Universidad de Granada

Departamento de Arquitectura y Tecnología de Computadores

Programa de Doctorado en Tecnologías de la Información y Comunicación

Optimization of 3D Hydrodynamic Models Applied to the Knowledge and Prediction of Inland Waters

Optimización de Modelos Hidrodinámicos 3D del Transporte y Mezcla Aplicados al Conocimiento y Predicción de Masas de Agua Continental

Memoria presentada para la obtención

del Grado de Doctor por la Universidad de Granada

con mención de Doctorado Internacional

Autor: Mario César Acosta Cobos

Directores: Dr. Mancia Anguita López y

Dr. Francisco J. Rueda Valdivia

Granada 2015

Editor: Universidad de Granada. Tesis Doctorales
Autor: Mario César Acosta Cobos
ISBN: 978-84-9125-139-2
URI: http://hdl.handle.net/10481/40298

El doctorando Mario César Acosta Cobos, y los directores de la tesis Mancia Anguita López y Francisco J. Rueda Valdivia. Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Abril 2015

Director/es de la Tesis

Doctorando

[illegible] [illegible]

Mario Acosta

Fdo.: Mancia Anguita López

y Francisco J. Rueda Valdivia

Fdo.: Mario César Acosta Cobos

Simplicity is a great virtue but
it requires hard work to achieve
and education to appreciate it.
And to make worse, complexity
sells better.

Edsger Dijkstra

Acknowledgement

I feel more comfortable expressing my gratitude in Spanish, so I apologize for the switch in the language.

El trabajo desarrollado en esta tesis no se puede considerar el trabajo de una sola persona, por eso agradezco a todos aquellos que han contribuido de una forma o de otra al desarrollo de la misma. Son muchas las personas que debo incluir, por ello, perdonad aquellos que no haya podido nombrar explícitamente.

Quería empezar dando las gracias a mis directores Mancia Anguita y Francisco Rueda. Esta tesis no sería posible sin su guía y su preparación a lo largo de estos años y por ello mis primeros agradecimientos deben ir dedicados a ellos. A Mancia Anguita por su esfuerzo y dedicación en el tutelaje de esta tesis. Trabajadora incansable, siempre con tiempo no solo para dirigir sino también para ayudar con cada uno de los detalles a realizar, enseñar con paciencia todo el conocimiento necesario y resolver los problemas que iban surgiendo en el desarrollo de esta tesis y en cualquier otra materia relacionada con mi propio aprendizaje. A Francisco Rueda por su paciencia en la enseñanza de un estudio multidisciplinar y su profesionalidad en el desarrollo de cada una de las tareas.

Quiero agradecer también a todos mis compañeros y profesores en la Universidad de Granada que han contribuido al desarrollo de esta tesis. En particular, al profesor Javier Fernández-Baldomero por los valiosos momentos “de café”, por su paciencia e implicación en el desarrollo de esta tesis y por explicarme con todo el detalle necesario cada pregunta que he podido formularle. A mis compañeras de despacho en el Instituto del Agua Anna, Cintia, Alicia y Andrea. Gracias por ayudarme con aquellas dudas multidisciplinares que no habría podido resolver de otro modo y por los buenos ratos que hemos compartido fuera y dentro de ese despacho. A mis compañeros de despacho en el CITIC Leo, Quique, Nolo, Karl y Fran por ayudarme en mi adaptación en el CITIC, diversos procesos administrativos y los buenos momentos vividos.

A todos mis amigos, los cuales han contribuido no solo de forma indirecta al desarrollo de esta tesis, sino también a que pudiera mantener fuera de la universidad la cordura necesaria para seguir con su proceso. Sois muchos todos los que debo nombrar

aquí, tanto Cordobeses como Granadinos, así que simplemente muchas gracias a todos. Especial mención a mi compañero de piso y amigo Manuel, con el que he compartido todos los años que he vivido en Granada y cuyo soporte agradeceré siempre.

Durante el desarrollo de esta tesis se realizó una estancia en la Universidad de California, Davis. Quería agradecer también al profesor Geoff Schladow por acogerme en la Universidad y por su tutelaje durante ese periodo. Además de sus revisiones, datos proporcionados y resolver cualquier duda que necesitara.

Por supuesto a toda mi familia. Sin ella sería imposible haber realizado esta tesis. En particular, a mis padres Luisa María y Mario César, cuyo amor incondicional, soporte infinito y esfuerzo por darme siempre lo mejor nunca dejan de sorprenderme y quienes han permitido ser la persona que soy y aprender con orgullo el tipo de persona que siempre he querido ser. A mi hermano Miguel Ángel, ingeniero brillante con un gran futuro y una persona excepcional en todos los sentidos, quería agradecer también cada momento que hemos pasado juntos.

Por último, pero no menos importante, quería dar un especial agradecimiento a Teresa. Tu forma de ser y brillantez es siempre fuente de inspiración y no tengo palabras para expresar lo importante que has sido no solo como apoyo en mi trabajo, sino también en cualquiera de las etapas vividas tanto en Granada como después. Simplemente espero que puedas seguir dándome tu apoyo, porque cualquier éxito que pueda conseguir será siempre en parte gracias a ti.

Resumen Extenso

Los procesos de circulación de agua en lagos y embalses se producen como resultado de flujos de energía térmicos y mecánicos través de las fronteras de un lago, disminuyendo desde procesos de larga escala (o escala cubeta) a micro escala y, finalmente, llevando a procesos de mezcla y disipación. Los patrones de mezcla inducidos por larga escala y ondas de alta frecuencia en lagos estratificados son inherentemente heterogéneos, llevando a variaciones en la densidad horizontal. En respuesta a estos gradientes de densidad horizontal, se forman patrones de flujo horizontales espacialmente complejos provocados por el forzamiento directo de viento y la rotación terrestre. Es a través de estos procesos de circulación horizontal que el sistema se reajusta volviendo a su estado de equilibrio con líneas de igual densidad, coincidiendo con el geopotencial en superficie. Los gradientes de densidad horizontal se producen en respuesta a un amplio número de mecanismos con diferentes escalas tanto espacial como temporal. Entre estos mecanismos se incluye la confluencia de ríos, variaciones espaciales térmicas producidas por flujos de calor en superficie, procesos de mezcla heterogénea debido a circulación de larga escala, procesos de mezcla inducidos por viento en superficie espacialmente variable y afloramiento o procesos de mezcla en la región béntica. Considerando este amplio número de mecanismos que afectan al gradiente de presión baroclínica, así como las diferentes escalas tanto espaciales como temporales que pueden presentar estos procesos, el problema de describir circulación horizontal en lagos y embalses todavía es difícil de abarcar de una forma sencilla.

Solo a partir de los últimos años, por un lado aprovechando los avances logrados en la recolección de datos mediante sensores remotos y el análisis de imágenes, los cuales permiten obtener información de velocidad en superficie y campos de temperatura y, por otro lado, haciendo uso de modelos numéricos tridimensionales (3D), capaces de resolver las ecuaciones de movimiento para simular el comportamiento del agua de un lago con suficiente resolución espacial y temporal, han permitido a ecólogos describir y comprender los complejos procesos de circulación que se producen en lagos y embalses. El estudio de estos procesos es de fundamental interés al ser los causantes de los procesos de transporte horizontales y la gran heterogeneidad presentes

en estos entornos. La mayoría de estos modelos numéricos están basados en la solución de las ecuaciones en tres dimensiones (3D) para aguas someras *(Shallow Water Equations,* 3D-SWE), cuya solución se trata de una forma simplificada del promediado de Reynolds de las ecuaciones de Navier-Stokes (*Reynolds averaged Navier-Stokes,* RANS), sujeta por las apropiadas condiciones de frontera. El uso de las 3D-SWE, debido a los límites físicos computacionales existentes y diversos estudios analíticos realizados a priori, está justificado para la descripción y estudio de los procesos de transporte de larga escala. Sin embargo, incluso haciendo uso de modelos basados en las 3D-SWE, ingenieros y científicos todavía se encuentran con un gran coste computacional al intentar simular los procesos de transporte horizontal y circulación presentes en la zona litoral de grandes lagos y embalses. Esta zona litoral es una parte fundamental en los ecosistemas acuáticos, con una enorme diversidad y siendo el hábitat de muchos de los organismos que viven en un lago. Por otro lado, esta zona se encuentra en continuo cambio al verse altamente afectada por la mano del ser humano, al ser el nexo de interacción entre el propio ser humano y el lago. Así, la zona litoral sufre numerosos cambios alrededor de sus costas debido a la construcción de residencias, zonas de ocio o pesca, tuberías para la extracción de agua o por el vertido de aguas residuales. Además, también es la entrada de una cantidad considerable de nutrientes a través de la desembocadura de ríos, como resultado de la agricultura y ganadería que hacen uso de estos ríos. Debido a estos motivos, hay una necesidad cada vez más demandante por comprender el entorno de la zona litoral tanto para el propio uso del ser humano como por la vida salvaje que compite por recursos en dicha zona. Sin embargo, simular correctamente estos entornos no es una tarea sencilla, el hábitat de la zona litoral puede ser sustancialmente heterogéneo tanto en la dimensión vertical como horizontal. Además, las condiciones físicas de esta área cambian de forma continua y muy dinámica (en escalas de tiempo muy pequeñas), como resultado de un forzamiento hidrodinámico intenso y la débil inercia característica de áreas poco profundas. Por otra parte, la zona litoral no puede estudiarse sin tener en cuenta los procesos que ocurren en la zona pelágica del lago y que influyen en la propia zona litoral. Esto conduce a que se debe simular tanto la circulación de larga escala, propia de la zona pelágica, como los procesos de pequeña escala que ocurren en la zona litoral. Adicionalmente, para que estos modelos obtengan resultados de interés, se deben simular durante largos periodos de tiempo. Como resultado, debido a estas enormes necesidades tanto temporales como espaciales, la mayoría de los fenómenos simulados producen simulaciones muy costosas (en términos de tiempo de ejecución y cantidad de memoria).

En los últimos años, la computación paralela está siendo cada vez más demandada y utilizada tanto en el área de Ingeniería Civil como en otras áreas de conocimiento, llegando a ser una práctica esencial para reducir los costes de computación requeridos en simulaciones numéricas de sistemas reales, o incluso para considerar abordar problemas de mayores dimensiones. Esta tesis doctoral pretende mostrar soluciones para reducir el coste computacional de modelos hidrodinámicos 3D, de forma que se puedan obtener resultados de simulaciones de grandes sistemas de agua continental durante largos periodos de tiempo y usando grids de alta resolución, todo en un tiempo aceptable y usando recursos fácilmente accesibles por cualquier científico. Diversas propuestas son explicadas y estudiadas. En primer lugar, se presenta un procedimiento conocido como anidamiento, utilizado para reducir el área de interés en alta resolución a una determinada zona del modelo, la cual está sujeta a condiciones de frontera dadas por otro modelo completo (el cual simula toda la cuenca) de baja resolución. En segundo lugar, se hace uso de técnicas de computación paralela que permitan realizar las simulaciones en plataformas de gama media/baja, tanto para realizar una implementación eficiente del procedimiento de anidamiento (de forma que el modelo completo de baja resolución y el modelo anidado de alta resolución se ejecuten en paralelo), como para dividir el trabajo a realizar entre los recursos disponibles, aplicando una optimización y paralelización del modelo hidrodinámico utilizado. Además, se presenta también una propuesta para simular áreas de alta resolución anidadas extensas en clusters de gama media/baja, dividiendo el área anidada en múltiples subdominios que puedan ser ejecutados (junto con el modelo completo de baja resolución) en paralelo, escalando casi de forma lineal a medida que más recursos para distribuir la zona de alta resolución anidada sean usados. Finalmente, también se demuestra que la adaptación y optimización de un modelo hidrodinámico a los recursos disponibles reducen considerablemente el coste computacional y el overhead generado por una implementación paralela, obteniendo con ello muy buenos resultados de escalabilidad incluso en arquitecturas de altas prestaciones.

Las implementaciones propuestas se han realizado sobre un ejemplo de modelo hidrodinámico basado en las 3D-SWE, Si3D (Figura r.1), aunque la mayoría de las mejoras pueden ser aplicables a otros modelos similares. Si3D fue implementado originalmente en su versión secuencial por E. P. Smith (2006). Las ecuaciones de gobierno en el modelo de Smith son discretizadas mediante un método de diferencias finitas usando un algoritmo semi-implícito de 3 niveles por pasos iterativos trapezoidal-salto de rana sobre un grid Cartesiano estructurado. En Si3D, el enfoque semi-implícito está basado en el tratamiento de ondas de gravedad y la difusión vertical de forma implícita para evitar limitaciones en el paso de tiempo tan estrictas como en el caso explícito debido a las condiciones dadas por Couram-Friedrich-Levy (CFL), y para garantizar la estabilidad del método. El resto de términos, incluyendo la advección, son tratados explícitamente. Aunque estos enfoques semi-implícitos evitan limitaciones estrictas en el paso de tiempo, también tienen la desventaja frente a los totalmente explícitos de que se deben formar y resolver largos sistemas de ecuaciones, normalmente resueltos mediante métodos iterativos, difíciles de paralelizar. En el caso de modelos 3D semi-implícitos, se debe formar y resolver un sistema de ecuaciones simétrico positivo definido con estructura pentadiagonal, resuelto mediante un método iterativo ampliamente utilizado conocido como Gradiente Conjugado (*Conjugate Gradient*, CG). Adicionalmente, al CG se le suele aplicar un precondicionador (*Preconditioner Conjugate Gradient*, PCG) que reduce el número de iteraciones que el método iterativo tiene que realizar para encontrar una solución óptima.

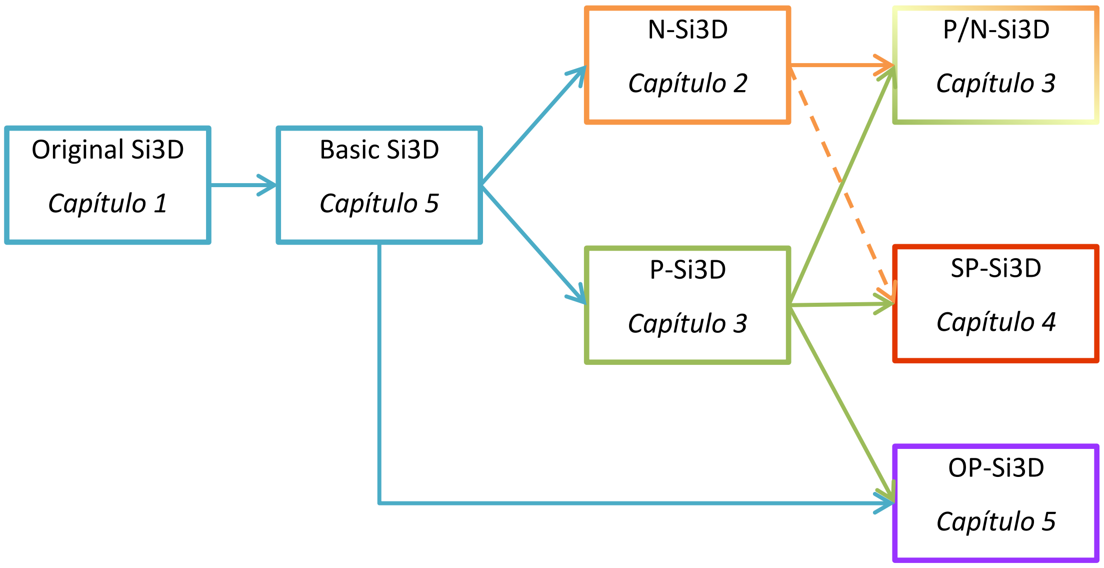


Figura r.1. Implementaciones de Si3D propuestas en este trabajo a partir del código original.

Resaltar que a este modelo original de Si3D se aplicaron diversas optimizaciones básicas (dando lugar a Basic Si3D, Figura r.1), necesarias como base en el desarrollo del resto de implementaciones realizadas (aunque esta implementación es el primer paso, se trata en detalle en el último capítulo, donde se explica todos los pasos a seguir en la completa optimización de un modelo hidrodinámico).

Las implementaciones propuestas fueron primero aplicadas en escenarios sintéticos y, posteriormente usadas en simulaciones de dos casos reales, 30 km de extensión del río Sacramento y el lago Tahoe, ambos situados en California (EEUU). Además, algunas de las implementaciones propuestas en esta tesis están siendo usadas también en otros casos de estudio, como el embalse de Beznar (España), la confluencia entro el río Ebro y el río Segre al final del embalse de Ribaroja (España), el lago Cayuga (EEUU) o el lago Tanganika (África). El modelo del río Sacramento es usado para comprender la influencia de la marea en la migración del salmón juvenil desde el propio río hasta el océano y reproducir la circulación lateral y secundaría en la zona de meandros del río. Por otra parte, el modelo del lago Tahoe es usado para caracterizar las rutas de transporte y migración de especies invasivas desde playas o bahías donde se encuentran establecidas a otras playas del lago, y el estudio de los procesos de transporte alrededor de la línea de costa que controlan el destino de contaminantes liberados por el ser humano en las playas.

La procedimiento de anidamiento propuesto e implementado en Si3D (N-Si3D, Figura r.1) se basa en un método de una vía (*one-way nesting*), en el cual se transfiere toda la información necesaria desde el modelo completo de baja resolución para solventar las dependencias que surgen en las ecuaciones discretizadas del modelo anidado de alta resolución. Opcionalmente, se añade también una zona de relajación 3D, la cual permite una transición suavizada del flujo de entrada al área anidada y evita reflexiones en la frontera en el flujo de salida, causadas por posibles diferencias en la solución de ambos modelos a medida que avanza la simulación. La propuesta aquí presentada se ejecuta de forma online siguiendo una estructura pipeline en la que el modelo completo de baja y el anidado de alta resolución se ejecutan en paralelo. La implementación paralela de una vía propuesta permite transferir toda la información requerida desde el modelo de baja al modelo anidado de alta resolución, tanto de la frontera como de la zona de relajación 3D, sin necesidad de almacenar ninguna información en ficheros y permitiendo realizar las transferencias incluso a cada paso de tiempo, eliminando así posibles errores por interpolación temporal. Además, las comunicaciones se encuentran solapadas con cálculo, por lo que la transferencia de datos no supone un coste computacional añadido para la implementación.

El procedimiento de anidamiento propuesto fue evaluado usando tanto modelos sintéticos como simulaciones de casos reales. La implementación anidada fue en todos los casos validada comparando los resultados de un área determinada, obtenidos tanto por el modelo anidado de alta resolución como con un modelo completo también en alta resolución. Las diferencias (Normal Root Square Error, NRMSE) entre los resultados de estos modelos fueron muy pequeñas (inferiores al 4%). Además, se muestra que estas diferencias pueden ser reducidas a 0 si (1) se usa la misma resolución en el modelo anidado y en el modelo completo que proporciona los datos de condiciones de frontera y (2) el sistema de ecuaciones pentadiagonal que se forma para resolver superficie libre, presente en modelos semi-implícitos, se resuelve mediante un método directo, el cual posee un alto coste computacional pero su solución es exacta. Por otra parte, a través de los resultados de estos casos de ejemplo se demuestra que la componente de velocidad tangencial debe ser transferida como condición de frontera desde el modelo completo de baja al modelo anidado de alta resolución. Los resultados demuestran que la ausencia de esta componente afecta considerablemente a la calidad de los resultados del modelo anidado, especialmente cuando existen corrientes intensas y paralelas a la frontera del modelo anidado. Este fue el caso, por ejemplo, de las simulaciones realizadas en *Claksburg bend* del río Sacramento, donde se producen fuertes corrientes de circulación *aguas arriba* del modelo. Por otro lado, los resultados se comparan con datos experimentales presentes en la literatura, mostrando buenas similitudes entre ambos. Por otra parte, se demuestra la necesidad de alta resolución en ambos casos, en el río Sacramento y en el lago Tahoe, comparando los resultados del modelo de anidado de alta con baja resolución en una misma zona, observando patrones de circulación que no son correctamente capturados en los modelos de baja y que, por otra parte, sí se capturan de igual forma en modelos completos de alta resolución como anidados en alta resolución.

A partir del procedimiento de anidamiento desarrollados en el Capítulo 2, se han explorado diversas propuestas para poder llevar a cabo simulaciones de líneas de costa extensas. En particular, este interés surge de la necesidad de simular toda la zona litoral del lago Tahoe. El interés en el estudio de toda la zona litoral se debe, como se indica en Rao and Schwab (2007), a que las corrientes en la zona litoral se producen generalmente a lo largo de zonas de igual profundidad, por lo que se crea una fuerte conexión física entre las playas y bahías existentes a lo largo de la zona litoral del lago. Esto lleva a que incluso reduciendo el coste computacional conseguido con anidamiento al simular una parte en baja resolución, la zona anidada en alta resolución siga teniendo un coste demasiado elevado y sea necesario buscar nuevas alternativas. La solución propuesta consiste en dividir los cálculos del modelo de alta resolución para simular la zona litoral entre los cores y computadores disponibles. Sin embargo, modificar un modelo semi-implícito para adaptarlo a la computación paralela no es una tarea trivial. A la hora de plantear nuevas soluciones en paralelo que alcancen una buena escalabilidad, hay que tener en cuenta el tipo de operaciones a realizar y su dificultad para realizarlas en una implementación paralela. Probablemente, en modelos 3D semi-implícitos la etapa computacional más difícil y que mayor *overhead* introduce en su paralelización es la resolución del sistema de ecuaciones pentadiagonal mediante métodos iterativos para obtener superficie libre. En Si3D este sistema se resuelve mediante el método del Gradiente Conjugado Precondicionado (*Preconditioned Conjugate Gradient*, PCG). Su implementación en paralelo requiere numerosas comunicaciones, tanto muchos-a-uno/uno-a-muchos entre todos los subdominios como comunicaciones entre subdominios vecinos, esto sumado a la necesidad del uso de reordenamiento como *red-black ordering* u otras técnicas típicas en la paralelización de métodos iterativos, produce un *overhead* importante que reduce considerablemente la escalabilidad del modelo y aumenta la complejidad de la implementación del mismo.

Una primera y plausible solución para introducir esta etapa en un modelo hidrodinámico 3D paralelo consiste en evitar la paralelización de la etapa que resuelve el sistema de ecuaciones pentadiagonal. Esto puede ser logrado (1) creando el sistema de ecuaciones en paralelo, repartiendo el trabajo mediante descomposición de dominio, (2) enviando cada parte del sistema de ecuaciones calculado a un solo proceso/hebra el cual lo resolverá de forma secuencial y (3) distribuyendo la solución obtenida entre todos los procesos/hebras usados. Las comunicaciones entre procesos ocurren dos veces mediante este método, antes de la resolución del sistema (muchos-a-uno) y después de resolver el sistema (uno-a-muchos). Esta implementación proporciona una solución paralela de Si3D (P-Si3D, figura r.1) adecuada para modelos de baja/media resolución, siempre que se utilice en pequeños clusters y que el cálculo del sistema pentadiagonal represente solo hasta un 2% del tiempo total de ejecución. Sin embargo, el *overhead* introducido por las comunicaciones hace que esta solución no sea escalable en clusters con un mayor número de recursos Una solución alternativa e inmediata que permite simular modelos de alta resolución de la zona litoral de un lago consiste en combinar P-Si3D con el procedimiento de anidamiento propuesto. La implementación resultante se ha llamado aquí P/N-Si3D (Figura r.1) y permitirá obtener resultados de alta resolución en pequeños clusters. El coste computacional haciendo uso de P/N-Si3D se reduce significativamente comparando con la versión secuencial de Si3D. Esto se consigue (1) simulando en alta resolución solo la zona litoral del lago mediante anidamiento y (2) haciendo uso de P-Si3D para la ejecución en paralelo del modelo de alta resolución anidado. Los resultados de este modelo son presentados y analizados para una bahía en particular al sureste del lago Tahoe (Marla Bay). En los resultados se observa que la concentración de trazador a lo largo del tiempo dentro de esta bahía son parcialmente debidos al trazador liberado de forma local dentro de la propia bahía, pero también parte de este trazador se concentra en la bahía como resultado de trazador exógeno que viajó a lo largo de la costa y que fue liberado en la costa sur del lago. Los resultados también muestran que los picos de concentración de trazador exógeno se producen durante fuertes periodos de viento, momento en el cual el agua de la zona sur del lago es rápidamente transportada a lo largo de la costa y atrapada en la bahía debido al desarrollo de procesos de pequeña escala de recirculación en forma de remolinos.

Dado que P-Si3D no escala correctamente, P/N-Si3D tampoco lo hace. Sin embargo, una implementación paralela y escalable puede ser construida si la solución de superficie libre del modelo completo de baja resolución es usada como condición de frontera del modelo anidado. Con este procedimiento, la zona anidada en alta resolución se divide en múltiples subdominios donde cada subdominio resuelve un subsistema de ecuaciones pentadiagonal de forma independiente. Con esta implementación, se evita las comunicaciones colectivas muchos-a-uno/uno-a-muchos que limitan la escalabilidad. Adicionalmente, esta propuesta utiliza una nueva y adaptada estructura pipeline además de la descomposición de dominio de P-Si3D. Esta estructura pipeline permite que el tiempo de ejecución del modelo completo de baja resolución no repercuta en el tiempo de ejecución total. La implementación SP-Si3D se ha evaluado y validado en un cluster de 9 nodos usando un modelo anidado en alta resolución y un modelo completo de baja resolución del lago Tahoe. Los resultados obtenidos muestran un escalabilidad lineal conforme el número de subdominios usados para distribuir la zona litoral en alta resolución del modelo anidado va aumentando y un error (NRMSE, comparando con los resultados de un modelo completo de alta resolución) muy pequeño.

Finalmente, se proponen y evalúan diferentes mejoras que optimizan P-Si3D, adaptando la implementación a la arquitectura disponible, ya sea a máquinas distribuidas o de memoria compartida. La implementación resultante se ha llamado aquí OP-Si3D (Figura r.1). El objetivo en este caso es reducir el *overhead* introducido por una implementación paralela. Entre otras mejoras, se evalúa la paralelización del sistema de ecuaciones pentadiagonal, evitando técnicas de reordenamiento o un mayor número de comunicaciones que las necesarias en otras etapas del modelo. Para llevar a cabo esta paralelización se hace uso de una variante del precondicionador Modified Incomplete Cholensky (MIC) que no añade cálculo o comunicaciones adicionales en su implementación paralela, obteniendo mejores resultados comparado a otras implementaciones usadas en la literatura, tanto en secuencial como en paralelo. Los resultados demuestran que una implementación secuencial o una implementación paralela no optimizada para resolver el sistema de ecuaciones pentadiagonal evitan que el modelo paralelo sea escalable, incluso aunque el resto de etapas estén optimizadas. La implementación resultante presenta una buena escalabilidad en cluster con más de 10 nodos.

Abstract

Water motions in lakes and reservoirs are initiated as a result of thermal and mechanical energy flowing through the lake boundaries, cascading down from the large- basin scale to the micro-scales, and ultimately leading to mixing and dissipation. Mixing patterns induced by basin-scale motions and high-frequency waves in stratified lakes are inherently patchy, setting up horizontal density variations. Spatially complex horizontal flow patterns develop in response to horizontal density gradients, when they are modulated by direct wind forcing and the Earth's rotation. It is through these horizontal motions that the system readjusts, returning to its equilibrium state with lines of equal density coinciding with geopotential surfaces. Horizontal density gradients have been shown to develop in response to a wide range of mechanisms with different spatial and temporal scales, including river inflows, thermal spatial variations response to surface heat fluxes, uneven mixing due to basin scale motions, spatially varying surface wind mixing, upwelling or mixing in the benthic boundary layer. Given the number of mechanisms leading to the development of baroclinic pressure gradients, and the wide range of spatio-temporal scales of these processes the problem of describing horizontal circulation in lakes and reservoirs has been elusive.

Only in the last few years, advances in remote sensing and quantitative imaging capable of retrieving surface velocity and temperature fields, and the use of three-dimensional (3D) numerical models solving the equations of motion applied to simulate lake motions with sufficient temporal and spatial resolution has allowed aquatic scientists to describe and understand the complex horizontal circulation patterns that develop in lakes and reservoirs, leading to horizontal transport and heterogeneity. Most of these models are based on the solution of three-dimensional form of the shallow-water equations 3D-SWE, a simplified form of the Reynolds averaged Navier-Stokes (RANS) equations, subject to the appropriate boundary conditions. Practical computational limits and *a priori* scaling analyses justify the use of the 3D-SWE in the description of these large-scale flows. But, even if using 3-D SWE models, engineers and scientists still face a serious challenge when trying to simulate horizontal transport and circulation in the near-shore environments of large water bodies. These are hot-spots in the aquatic ecosystems with large biodiversity and critical habitats for many

organisms in lakes. But, being the nexus of human interactions with lakes, littoral habitats are highly modified by human uses. Humans build structures, recreate, fish, extract water, or dispose sewage at lake edges. Significant inputs of nutrients also arrive to the coastal zone through rivers, as a result of agricultural practices or cattle-raising in the contributing watershed. Hence, there is an increasing need to understand near-shore environments where human uses and natural wild-live compete for resources. Simulating these environments, however, is not simple. Littoral habitats can be substantially heterogeneous in both vertical and horizontal dimensions. Moreover, physical conditions exhibit continuous and very dynamic changes, at short-time scales, as a result of strong hydrodynamic forcing and the weak inertia of shallow layers, and also as a result of the time varying nature of human activities. Furthermore, near-shore regions, though, cannot be understood in isolation from the pelagic. Hence, in trying to simulate the near-shore physical conditions both local and basin-scale circulation features need to be resolved simultaneously during long periods of time, and the computational cost of these simulations can be formidable.

Parallel computation platforms are being increasingly demanded and used in the area of Civil and Environmental Engineering, and others research areas, to reduce the computational time required to conduct numerical simulations of real systems, or deal with even larger problems. In this dissertation a series of solutions are proposed and tested to reduce the computational cost of 3D hydrodynamic models, so that simulations of water motion and transport in near-shore regions of large geophysical systems, during extended periods of time and using high resolution grids, can be conducted with acceptable execution time and using accessible resources. Several approaches are introduced and studied. First, we explore nesting-procedures in which only localized regions of the littoral zone are simulated using a very high-resolution or inner-model, with boundary conditions which are provided by an outer-model that solves the large-scale processes in the rest of the water body. Second, we use parallel computation techniques so that use can be made of mid-range or low-cost platforms to run the inner and outer models simultaneously. We also optimize and parallelize the model computations used in small commodity clusters, dividing the calculations in the near-shore regions among a large number of processors, as they become available in parallel platforms. The resulting optimized and parallelized model of near-shore regions scales almost linearly, so that the computational model is run faster as more resources are

used. We have additionally taken additional steps to adapt the hydrodynamic model to the architecture and tools available. With this work, we demonstrate that the adaptation and optimization of the model to the available resources can also be used to reduce significantly the computational cost, with very good scalability results even using High performance platforms.

The implementation was carried out on a particular 3D-SWE, which was originally implemented for serial architectures by P. E. Smith (2006) (Figure a.1). Most improvements proposed here, though, can be applicable to other similar models. The governing equations in Smith's model are solved using a semi-implicit, three-level, iterative leapfrog-trapezoidal finite difference algorithm on a staggered Cartesian grid. The semi-implicit approach is based on treating the surface gravity wave and vertical diffusion terms implicitly to avoid time-step limitations as strict as in an explicit case due to gravity wave Courant–Friedrich–Levy (CFL) conditions, and to guarantee the stability of the method. All other terms including advection are treated explicitly. Although this semi-implicit approach avoids strict time-step limitations, it also has the disadvantage compared to completely explicit approach that long system of equations must be formed and solved, usually using iterative methods which are very difficult to parallelize. Semi-implicit 3D-SWE models form a pentadiagonal system of equations which is symmetric positive definite. The system of equations is solved by an iterative method widely used known as Conjugate Gradient (CG). Additionally, the CG is usually applied using a preconditioner (PCG), which reduces the number of iterations of the CG to converge to a correct solution within a tolerance. The semi-implicit model used in this dissertation will be referred to as Si3D (Figure a.1).

Highlight that the original Si3D model was modified with several basic optimizations (Basic Si3D, Figure a.1). These optimizations are used in all the implementations made (although this implementation is the first step, it is covered in detail in the last chapter, where it is explained all the steps involved in the complete optimization of a hydrodynamic model).

The approaches proposed were first applied to simulate synthetic scenarios, and then used to conduct realistic simulations in two systems: a 30 km reach along the Sacramento River, and, Lake Tahoe, both of them in California (USA). Furthermore, some of the implementations developed in this thesis are being used in other study cases, including Beznar Reservoir (Spain), the confluence between Ebro and Segre

Rivers in the upstream end of Ribarroja Reservoir (Spain), Lake Cayuga (USA) and Lake Tanganyika (Africa). The Sacramento River model is used to understand the influence of tidal river dynamics on the migration of juvenile salmon towards the ocean and to reproduce the lateral and secondary circulations in the area of channel meanders. The Lake Tahoe model is used to study near-shore transport processes controlling the fate of contaminants released by humans in beaches, and the migration pathways of planktonic larvae of invasive species from bays where they have been able to settle to other beaches and bays which are free.

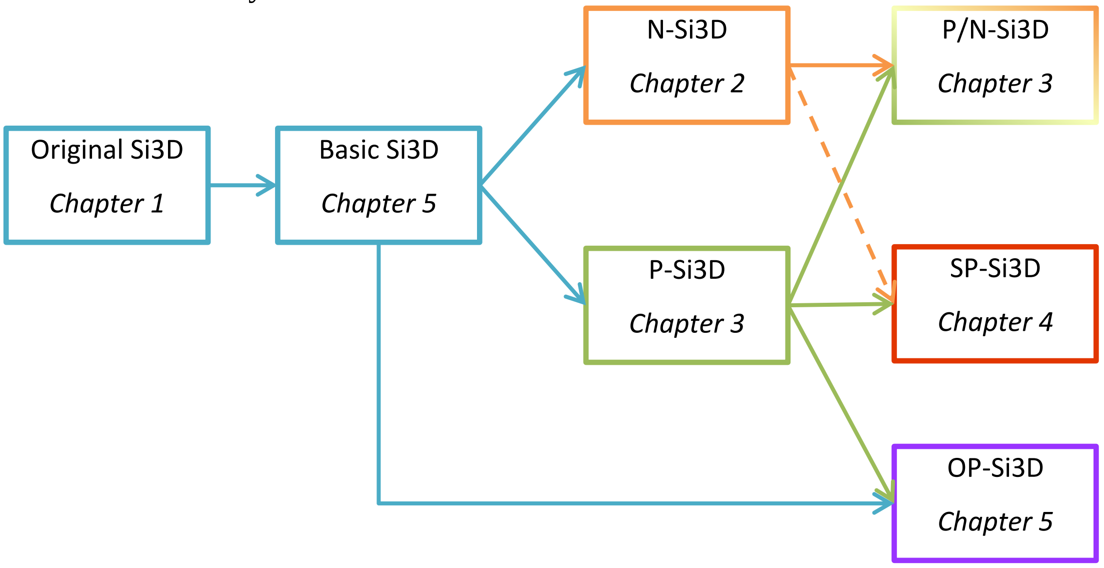


Figure a.1. Si3D implementations proposed in this work from the original version of Si3D

The nesting procedure developed and implemented in Si3D (N-Si3D, Figure a.1), is a one-way nesting method, in which all the necessary information is transferred from the outer- (or low resolution) basin scale model to the inner- (or high resolution) model, to solve the dependencies in the discretized equations of the latter. It optionally adds a 3D relaxation area, which allows a smooth transition from the inflow in the nested model and prevents reflections at the border in the outflow due to possible differences in the solution of both models when the simulation progresses over time. Furthermore, the inner and outer models are executed simultaneously in parallel in an online mode, following a pipeline structure. In the parallel one-way implementation all the information required is passed from the outer to the inner model without storing any information in files, both the nested open boundary and the 3D relaxation area, and allows the transfer of information even each time-step, eliminating possible errors by

temporal interpolation. In addition, communications are overlapped with calculation, so it does not represent an overhead in the implementation.

The nested approach was tested using both synthetic and realistic simulations. The nested implementation was in all cases validated by comparing the results of simulations in a small region (sub-domain) of a lake or river model. These differences in results between the nested and the complete model (error) are very small and even 0, using the same grid resolution both in the nested- or inner-model and the complete or outer-model and when the pentadiagonal matrix for water surface elevation built in the semi-implicit model was solved using a direct method, which is computationally demanding but exact. Through these case examples, we demonstrate that the tangential velocities need to be transferred from the low resolution to the high resolution model. If not they can affect significantly the quality of the nested solution, in particular when currents parallel to the inner-outer boundary are strong. This was the case, for example, of the high-resolution simulations of a river bend (Claksburg bend) along the Sacramento River, where strong lateral circulation develops upstream the inner-domain. The nested model results agree well with observations previously reported in the literature. Furthermore, the nested-model results compare well with the results from the high-resolution model of the whole reach, with differences (Normal Root Square Error, NRMSE) that are less than 4%. In other environments, with weaker currents, though, the need for passing tangential velocity information is not that strong. This was the case when simulating the local-scale circulation in a small bay (Marla Bay) of Lake Tahoe. In our realistic simulations of both the river bend in Sacramento River and in Marla Bay, Lake Tahoe, the use of a high-resolution grid in the inner-model reveals flow features which cannot be simulated with the low-resolution basin-scale model.

Using the nesting procedure developed in Chapter 2 we have explored approaches to conduct high-resolution simulations of extended near-shore regions. In particular, we are interested in simulating the littoral perimeter of Lake Tahoe. As reviewed by Rao and Schwab (2007), currents in the near-shore are largely aligned along isobaths, hence, creating strong physical links among beaches and bays existing along lake perimeters. The extension of the inner-domain (the littoral perimeter of a large lake) in these simulations can be large, and, high-resolution simulations in this domain can be very computationally demanding. The solution proposed consists of dividing the high-resolution computations of the littoral fringe among several

cores/computers. Modifying a semi-implicit model to conduct parallel computations, though, is not free of difficulties. Probably the computational stage in a semi-implicit 3D model which poses the largest difficulties to parallelize is the solution with iterative methods of the pentadiagonal matrix problem for the free surface elevation. In Si3D the matrix problem is solved using a Preconditioned Conjugate Gradient method. Its implementation in parallel requires numerous communications, both one-to-many/many-to-one where all sub-domains are involved and communications between neighbor sub-domains. In addition, the need of using reordering (as red-black ordering) or other techniques typically used to parallelize iterative methods produces an important overhead which may reduce significantly the scalability of the algorithm and increases the complexity of the implementation.

A first and plausible approach to parallelize the model computations consists of avoiding the parallelization of the pentadiagonal matrix solution. This can be achieved by (1) creating the pentadiagonal matrix in parallel by splitting the workload, through domain decomposition, among several threads/processes of the operating system, (2) then solving the matrix problem sequentially in one of the threads/processes; and, (3) distributing the solution of the matrix problem among all the threads/processes. Communications among processes in this stage occur twice: just before (many-to-one) and after (one-to-many) the matrix solution. This parallel implementation of Si3D (P-Si3D, Figure a.1) only works properly when the computational cost of the matrix solution represents only a small fraction (up to 2%) of the total runtime. This approach, however, by itself, scales poorly as a result of the overhead introduced by the communications involved in constructing the matrix and distributing the solution among processes and the sequential execution of the pentadiagonal matrix solution. Still one can use the parallel implementation of Si3D, conjunctively with the nesting procedures, to conduct high-resolution simulations of the littoral fringe. This implementation is here referred to as P/N-Si3D (Figure a.1) and can be used to conduct high-resolution near-shore simulations in small clusters. The cost of P/N-Si3D is significantly reduced, compared to the sequential version of the model, as a result of two strategies (1) only using a high-resolution grid in the nested near-shore model; and (2) simulating the littoral fringe using the parallel version of the model P-Si3D. This implementation was used to simulate the dispersion of passive tracers released at several locations along the perimeter of Lake Tahoe. These tracer simulations were intended to represent the fate of

water constituents entering the lake through a total of 51 outfalls existing around Lake Tahoe, discharging storm-water directly into the lake. The results of the model are presented and analyzed for a particular bay in the southeast of Lake Tahoe (Marla Bay). Tracer concentrations within the bay are partly explained as a result of tracer being released locally within the bay, but also as a result of long-shore currents carrying exogenous tracer released outside the bay, along the southern coast of Lake Tahoe. The concentration of exogenous tracer peaks during periods of strong winds, when water from the South is rapidly transported and trapped in the bay as a result of the development of local bay scale eddies.

Given that P-Si3D does not scale correctly, P/N-Si3D does not either. A scalable parallel and nested implementation, though, can be constructed if the free surface solution of the outer- low-resolution model is used as boundary conditions so that the high resolution nested model is divided into subdomains of the littoral fringe where each subdomain can solve a pentadiagonal matrix sub-problem independently. With this implementation, we can avoid collective communications many-to-one/one-to-many which limit scalability. The resulting implementation uses an adapted pipeline structure in addition to the domain decomposition of P-Si3D. The pipeline structure allows the runtime of the low resolution model added has no effect on the total execution time. We will refer to this implementation as SP-Si3D. It has been evaluated and validated in a cluster of 9 nodes using a high-resolution nested model and a low resolution model of Lake Tahoe. The results show a linear scalability when the number of subdomains used to distribute the high resolution littoral zone of Lake Tahoe increases and an error (NRMSE, comparing with the results of a complete high resolution model), very small.

Finally, several approaches are proposed and tested to optimize P-Si3D by adapting the parallel version of the code to the available architecture, for both distributed machines and shared memory platforms. The resulting implementation is called OP-Si3D (Figure a.1). Our goal was to reduce the overhead introduced by a parallel implementation. Among other approaches tested, we parallelized the pentadiagonal matrix solution using a new Modified Incomplete Cholensky preconditioner (MIC) which we propose that does not add any communications or reordering. Using this matrix solution method one gets better time execution results compared to other implementations used in the literature, both sequential and parallel methods. We proved that a parallel method scales poorly if either a sequential implementation or a non-optimized parallel implementation

of the matrix solution is used, even though other stages are fully optimized. The resulting implementation presented here has good scalability in high performance plataforms with more than 10 nodes.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

ADCP, Acoustic Doppler Wave and Current Profiler

ADI, Alternating Direction Implicit

ALU, Arithmetic Logic Unit

API, Application Programming Interface

BJ, Block Jacobi

CFD, Computational Dynamic Fluid

CFL, Courant-Friedrich-Lewy

CPU, Central Processing Unit

C/S, Communication and Synchronization

EFDC, Environmental Fluid Dynamic Code

ELM, Eulen-Lagrangian Method

EM, Excess Mass

E-W, East-West

GB, Giga Byte

GHz, Giga Hertz

GPU, Graphics Processing Unit

HPC, High Performance Computing

HR, High Resolution

IC, Incomplete Cholensky

IO, In-Out

IPO, Interprocedural Optimization

JAC, Jacobi

QETE, Quasi-Equilibrium Turbulence Model

LR, Low Resolution

MIC, Modified Incomplete Cholesky

MMIC, Modified Modified Incomplete Cholesky

MN-Si3D, Multiple Nested Semi-implicit Three Dimensional

MOM, Modular Ocean Model

MPI, Message Passing Interface

NRMSE, Nominal Root Square Error

N-S, North- South

N-Si3D, Nested Semi-implicit Three Dimensional

NSPCG, Non-Symmetric Preconditioned Conjugate Gradient

NUMA, Non-Uniform Memory Access

OBC, Open Boundary Condition

OP-Si3D, Optimized Parallel Semi-implicit Three Dimensional

OpenMP, Open Multi-Processing

PC, Personal Computer

PCG, Preconditioned Conjugate Gradient

PGO, Profile Guided Optimization

P/N-Si3D, Parallel Nested Semi-implicit Three Dimensional

POM, Princeton Ocean Model

POP, Parallel Ocean Program

PSi3D, Parallel Semi-implicit Three Dimensional

RAM, Random Access Memory

RANS, Reynolds Averaged Navier-Stokes

RMSE, Root Mean Square Error

RMSE_t, Temporal Root Mean Square Error

ROMS, Regional Ocean Modeling System

Si3D, Semi-implicit Three Dimensional

SLT, South Lake Tahoe

SMP, Symmetric Multiprocessing

SMT, Simultaneous MultiThreading

SP-Si3D, Scalable Parallel Semi-implicit Three Dimensional

SSE, Streaming SIMD Extensions

SWE, Shallow Water Equations

TERC, Tahoe Environmental Research Center

TKE, Averaged Kinetic Energy

USA, United States of America

USGS, U. S. Geological Survey

1D, One Dimensional

2D, Two Dimensional

3D, Three Dimensional

cm/s, centimeter per second

d, day

GB, Giga Bytes

GT/s, Gigatransfer per second

h, hour

KB, Kilo Bytes

km, kilometer

kg/m^3, kilograms per meter cubed

kg/s, kilograms per second

m, meter

MB, Mega Bytes

m/s^1, meter per second squared

m^3/s, meter cubed per second

µs, microsecond

s, second

W, watt

ºC, grade centigrade

Introducción General y Objetivos

De la cantidad total de agua puesta en juego en el ciclo hidrológico a escala planetaria, una cantidad ínfima (un 0.008%) se encuentra en forma de agua dulce en ríos, lagos y embalses. Esta fracción, no obstante, es de extraordinario valor para el ser humano, ya que casi el 80% del agua que consumimos procede de estas masas de agua superficial. A pesar de su importancia, el conocimiento de los ecosistemas de agua dulce, de acuerdo con el *Nacional Research Council* de EEUU, es aún escaso. Este conocimiento es particularmente pobre cuando se considera el comportamiento de estos sistemas de agua dulce a largo plazo, en torno a decenas de años (Harris and Durran 1986; Armengol et al. 1994), y a corto plazo, en escalas de tiempo diaria u horaria (Imberger and Parker, 1985). Esta falta de conocimiento se debe, en parte, a la multiplicidad y complejidad de los procesos físicos de transporte y mezcla. Estos procesos son un aspecto fundamental en el funcionamiento de ecosistemas acuáticos. Por un lado, determinan la posición y movimiento de partículas y el grado de disolución de sustancias presentes en el agua a lo largo del tiempo y, por otro, determinan las condiciones ambientales en las que ocurren las reacciones biogeoquímicas. Por ello, hay que tener en cuenta que para comprender la variabilidad espacial y temporal de las propiedades químicas y biológicas de un ecosistema acuático, será necesario primero comprender los procesos físicos de transporte y movimiento existentes en estos sistemas (Imberger 1998).

La utilización de modelos hidrodinámicos capaces de resolver las ecuaciones de movimiento de los fluidos en tres dimensiones (3D) espaciales, junto con la aplicación de tecnologías avanzadas para la observación de la velocidad del agua y la estructura térmica, han contribuido en las últimas décadas a iniciar la exploración del movimiento del agua en sistemas naturales (e.g. Hodges et al. 2000, Rueda and Schladow 2003). Muchos de estos modelos están basados en la solución numérica de una forma simplificada de las ecuaciones de Navier-Stokes para aguas someras (*Shallow Water Equations*, SWE), en las que se supone que la distribución de presiones en la dirección vertical es hidrostática. Esta forma de las ecuaciones se prefiere sobre las ecuaciones no-hidrostáticas por su menor coste computacional. Aun así, los modelos hidrostáticos siguen siendo ineficientes y pesados de ejecutar cuando se utilizan grids de alta resolución espacial, necesarios para resolver procesos de pequeñas escalas espaciales (giros y remolinos horizontales o corrientes de gravedad) y especialmente si se quieren utilizar plataformas de cómputo de uso masivo, como PC o servidores de gama baja.

Afortunadamente, los importantes avances que se han realizado en las últimas décadas en el área de la computación han permitido un impulso notable en la investigación en diferentes áreas de conocimiento, incluyendo la simulación de aguas con modelos hidrodinámicos. Actualmente resulta incuestionable la necesidad de utilizar computadores para realizar todo tipo de actividades científicas y de aplicar técnicas de computación avanzadas para reducir los enormes costes de computación así como la ingente cantidad de memoria necesaria en la ejecución de simulaciones 3D. Este avance ha propiciado que se plantee abordar problemas científicos cuyos requisitos computacionales no pueden ser simulados en secuencial, reduciendo los costes computacionales y permitiendo la simulación de problemas usando grids de alta resolución, impensables de realizar en un tiempo razonable apenas unos años atrás.

El objetivo general de esta tesis doctoral será mejorar la eficiencia computacional de los modelos hidrodinámicos existentes con el fin de abordar el estudio riguroso y detallado de los procesos de transporte y mezcla en sistemas de agua continental. Se mostrarán soluciones para reducir el coste computacional de estas simulaciones, de forma que se puedan obtener resultados útiles en un tiempo aceptable con grids de alta resolución, empleando tanto recursos fácilmente accesibles por cualquier científico como Arquitecturas de Altas Prestaciones (*High Performance Computing*, HPC). Las estrategias de optimización utilizadas han permitido mejorar la eficiencia de estos modelos, entendida esta como la relación entre prestaciones y coste y como la relación entre prestaciones y calidad de los resultados. Estas estrategias de optimización deberán facilitar y extender la utilización de los modelos 3D de transporte y mezcla como herramientas de trabajo en investigación aplicada al conocimiento de lagos, embalses y ríos, o como herramientas operacionales de predicción. Para ello se pretende realizar el procesamiento requerido en un tiempo aceptable en computadores personales, clúster de computadores personales y servidores de gama baja, evitando usar costosos servidores de gama alta. Para los casos más costosos, se presenta también una alternativa haciendo uso de HPC, implementada de forma eficiente.

Este objetivo general de mejorar la eficiencia computacional de los modelos hidrodinámicos existentes se pretende conseguir mediante dos propuestas diferentes, la primera aprovechando al máximo la arquitectura de los procesadores de propósito general (Anguita et al. 2009, Anguita and Martinez-Lechado 2005). El aprovechamiento de la arquitectura se ha comprobado muy rentable en diversas aplicaciones en las que se requiere tratar gran cantidad de datos y realizar gran cantidad de cálculos, además de aplicaciones relacionadas con el procesamiento de audio, gráficos, imágenes y vídeo (Anguita et al. 2009, Anguita and Martinez-Lechado 2005, Bhattacharjee et al. 2008, Guobin et al. 2005), también se ha comprobado su utilidad para aplicaciones en otros campos como la bioinformática, financieras o procesamiento de datos sísmicos (Ino et al. 2009, Lee et al 2009, Panetta et al. 2009, Zhang and Oosterlee 2009) y, por supuesto, en el campo de 3D-SWE (Fringer et al. 2006, Nesterov 2010, Amritkar et al. 2012, Tubbs and Tsai 2009)).

La segunda propuesta para reducir los costes de computación hace uso de un procedimiento conocido como anidamiento, ya usado en la literatura para reducir los costes de computación (Fox et al. 1995, Zavatarelli and Pinardi 2003). Usando este procedimiento es posible reducir la zona de alta resolución a solo el área que se pretenda estudiar, resolviendo el resto del grid con una resolución menor y, por lo tanto, reduciendo el coste computacional y el almacenamiento. Sin embargo, este procedimiento es insuficiente cuando las zonas de estudio en alta resolución siguen siendo demasiado extensas. Un ejemplo de ello se da en la zona litoral de lagos de grandes dimensiones. Los hábitats de esta zona litoral pueden ser sustancialmente heterogéneos tanto en la dimensión vertical como horizontal (Lodge et al. 1988), donde en cortos espacios de tiempo se producen cambios continuos y dinámicos en las condiciones físicas de esta zona, siendo el resultado del fuerte forzamiento hidrodinámico sumado a la débil inercia de estas zonas de poca profundidad (Lodge et al. 1988). Dado este extraordinario y continuo dinamismo de la zona litoral, existe en ella una gran biodiversidad (Vadeboncoeur et al. 2011) y hábitats únicos de muchos de los organismos que viven en lagos (Kalff 2001). Además, las playas o bahías alrededor de la costa no pueden ser estudiadas de forma individual sin tener en cuenta en el estudio a sus bahías vecinas, dado que estas se encuentran conectadas a través de diversos procesos físicos. Como se indica en Rao and Schwab (2007), las corrientes en la zona litoral se producen generalmente a lo largo de zonas de igual profundidad, por lo que se crea una fuerte conexión física entre la zona litoral a lo largo de todo el perímetro del lago.

En esta tesis se propone una implementación eficiente del procedimiento de anidamiento, combinándolo con clusters de computadores de gama media/baja, reduciendo aún más los costes computacionales y permitiendo resolver de forma eficiente toda la zona litoral (imposible en un tiempo aceptable mediante un procedimiento de anidamiento normal) de grandes lagos en alta resolución , del cual a pesar de su enorme diversidad e importancia como se ha explicado, se tiene un conocimiento bastante pobre (Kalff 2001). En línea a esto se muestran resultados útiles y de interés a la comunicad científica que indiquen el camino a seguir en el estudio de la zona litoral de grandes lagos y que demuestren la necesidad del uso de grids de alta resolución para poder realizar estos estudios con éxito.

Las soluciones presentadas en este trabajo se han implementado sobre un de modelo hidrodinámico basado en la solución numérica de las ecuaciones de Navier-Stokes para aguas someras, Si3D, aunque la mayoría de las mejoras pueden ser aplicables a otros modelos similares. En Si3D las ecuaciones son discretizadas mediante un método de diferencias finitas, usando un algoritmo semi-implícito de 3 niveles por pasos iterativos trapezoidal-salto de rana sobre un grid Cartesiano estructurado, donde la mayoría de los cálculos se realizan columna por columna de agua. Los algoritmos semi-implícitos existentes hoy día siguen la propuesta original diseñada por el Profesor Casulli (Casulli and Cheng 1992; Casulli and Cattani 1994) y, aplicados sobre mallas estructuradas y no estructuradas, han sido muy utilizados en la simulación de ecosistemas acuáticos continentales (e.g. Hodges et al. 2000, Appt et al. 2004, Laval et al. 2003) en los últimos años. El modelo semi-implícito al que se propone aplicar las estrategias de optimización computacional ha sido utilizado recientemente en un número importante de estudios sobre transporte y mezcla en ríos, embalses y lagos (Rueda et al. 2009, Rueda and Schladow 2009, Rueda and MacIntyre 2009, Hoyer et al 2014, Ramón et al 2013). Si3D fue desarrollado y propuesto originalmente por el *U.S. Geological Survey* (Smith 2006) y adaptado posteriormente para la simulación de lagos por Rueda (2001).

En el capítulo 1 se incluye una completa descripción de este modelo, incluyendo las ecuaciones de gobierno utilizadas, el modelo numérico, el flujo de ejecución del algoritmo y otros detalles computacionales de interés que demuestran la complejidad de algunas de las operaciones que se realizan en estos modelos y el enorme coste computacional (en términos de tiempo de ejecución y requerimientos de memoria) que precisan. El resto de capítulos han sido escritos como artículos independientes con su propio resumen, introducción, metodología, resultados, discusión y conclusiones. Parte de este trabajo ya ha sido publicado o está en proceso de revisión en revistas internacionales, otra parte de este trabajo está en preparación para ser enviado para su publicación. Debido a que estos capítulos han sido escritos como artículos individuales, algunos de los conceptos pueden repetirse de un capítulo a otro. Esto por otra parte, tiene como ventaja que cada capítulo puede leerse de forma independiente. La Figura i.1 explica las diferentes implementaciones desarrolladas a partir del código original de Si3D. El primer paso consistió en aplicar diversas optimizaciones básicas al código original de Si3D (Basic Si3D, Figura i.1), usadas en el resto de implementaciones realizadas (aunque esta implementación es el primer paso, se trata en detalle en el último capítulo, donde se explica todos los pasos a seguir en la completa optimización de un modelo hidrodinámico).

Los capítulos se desarrollan siguiendo un orden lógico en relación con cada una de las implementaciones desarrolladas, partiendo desde el modelo secuencial de Si3D con optimizaciones básicas (Basic Si3D). El segundo capítulo contiene la descripción, evaluación y validación del método de anidamiento propuesto (N-Si3D). En el capítulo 3 se presenta una implementación paralela adecuada para pequeños clusters (P-Si3D). En este capítulo se presenta un ejemplo práctico de aplicación en la zona litoral de Tahoe que requiere la combinación de la implementación paralela desarrollada con el procedimiento de anidamiento propuesto en el anterior capítulo (P/N-Si3D). En el capítulo 4 se describe y evalúa una de las implementaciones paralelas desarrolladas: SP-Si3D, que añade una implementación con grid de baja resolución a la implementación paralela con grid de alta resolución. La implementación propuesta modifica P-Si3D para que, haciendo uso de los resultados del modelo de baja resolución (haciendo uso del procedimiento de anidamiento N-Si3D), elimine las comunicaciones colectivas uno-a-muchos y muchos-a-uno necesarias en modelos hidrodinámicos semi-implícitos paralelos. Esta implementación consigue una escalabilidad prácticamente lineal. Finalmente en el capítulo 5, se muestran diversas propuestas para optimizar P-Si3D, que se engloban bajo una implementación denominada OP-Si3D. Las propuestas dadas en el capítulo 5 permiten adaptar la implementación a la arquitectura de una plataforma de propósito general actual, reduciendo al mínimo el *overhead* producido por una implementación paralela y permitiendo obtener una versión escalable incluso en arquitecturas HPC. A continuación se indican los objetivos concretos y se introduce brevemente los contenidos de cada capítulo:

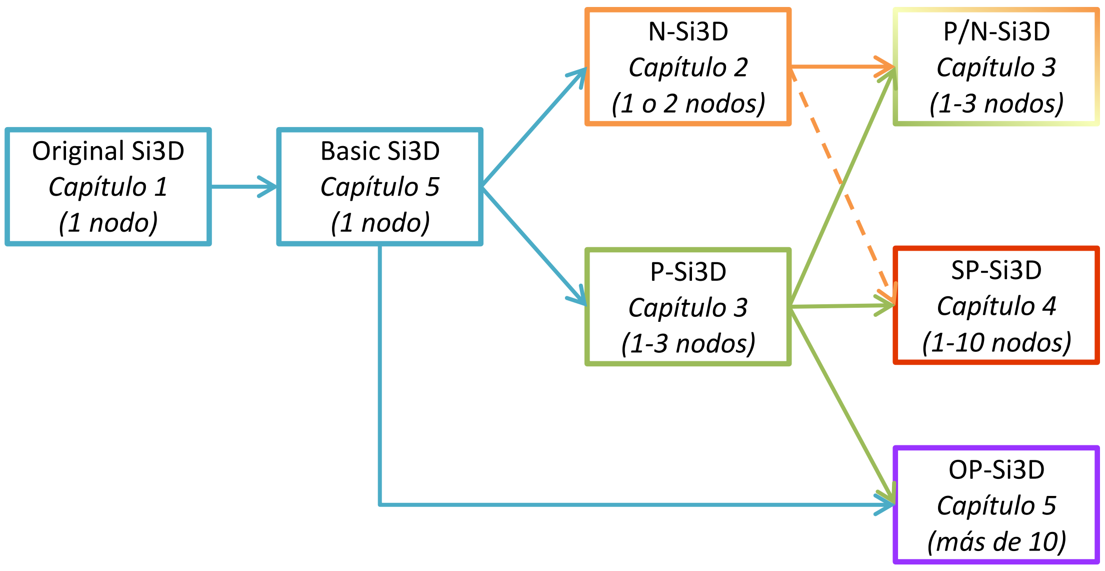


Figura i.1. Implementaciones propuestas para este trabajo desde la versión original de Si3D.

En el capítulo 2 el objetivo es obtener un procedimiento de anidamiento (N-Si3D) que permita aprovechar de forma eficiente la arquitectura de un computador, además de evaluar y conocer cuáles son los principales inconvenientes de esta técnica y reducirlos al mínimo. Usando este procedimiento es posible reducir la zona de alta resolución a solo el área que se pretenda estudiar (mediante un modelo anidado), resolviendo el resto del grid con una resolución menor (mediante un modelo completo de baja resolución) y, por lo tanto, reduciendo el coste computacional. En este procedimiento las ecuaciones del modelo anidado se resuelven teniendo en cuenta las condiciones de frontera dadas por el modelo completo de baja resolución. Para poder obtener un procedimiento de anidamiento correctamente acoplado, se hace especial énfasis en la importancia de calcular correctamente todos los términos en las ecuaciones de masa y momentum para aquellas columnas de agua adyacentes a la frontera en el modelo anidado. En particular, el estudio se centra en la necesidad de transferir el componente de velocidad tangencial desde el modelo completo de baja resolución al modelo anidado. También se presenta el procedimiento utilizado para ejecutar ambos modelos (anidado y completo de baja resolución) en paralelo al mismo tiempo. Este capítulo incluye una evaluación y validación completa del procedimiento de anidamiento indicando cuales son las principales fuentes de error. Finalmente, se justifica la necesidad de usar grids de alta resolución, mostrando patrones de circulación de pequeña escala que solo son correctamente capturados usando grids de alta resolución.

En el capítulo 3 el objetivo es obtener una implementación paralela de Si3D (P-Si3D) adecuada para pequeños clusters multinúcleos. En esta implementación la carga de trabajo se divide entre hebras/procesos disponibles, cada uno trabajando en un subconjunto distinto de columnas de agua del dominio. Solo el sistema de ecuaciones pentadiagonal necesario para obtener la solución de superficie libre es resuelto de forma secuencial, reduciendo así el número de comunicaciones y la complejidad que supone paralelizar esta etapa. La ejecución secuencial de dicha etapa supone la necesidad de usar una comunicación colectiva muchos-a-uno antes de iniciar la etapa para formar el sistema de ecuaciones completo y una comunicación colectiva uno-a-muchos que difunde la solución de superficie libre obtenida. Este modelo paralelo es sencillo de implementar y reduce el tiempo de ejecución de forma aceptable para modelos de baja y media resolución en pequeños clusters. Sin embargo, las funciones colectivas muchos-a-uno y uno-a-muchos impiden que la implementación escale en cluster de más de 3 nodos. Se incluye también un estudio del rendimiento de la implementación usando diferentes configuraciones del hardware y del tipo de corte usado en la descomposición de dominio. Finalmente, se combina el procedimiento de anidamiento creado (N-Si3D) con la implementación paralela desarrollada para pequeños clusters (P-Si3D) para simular la zona litoral del lago Tahoe en alta resolución. Esta implementación es usada para desarrollar un caso real de prueba que revela diversos patrones característicos en el transporte a lo largo de la costa que se suceden en el lago Tahoe.

En el capítulo 4 el objetivo es obtener una implementación paralela escalable en cluster multinúcleos de bajo precio de unos 10 nodos, la cual permita simular los procesos que ocurren en toda la zona litoral de grandes lagos. La implementación propuesta modifica P-Si3D para que, haciendo uso de los resultados de un modelo de baja resolución, elimine las comunicaciones colectivas uno-a-muchos y muchos-a-uno necesarias cuando se resuelve en secuencial el sistema de ecuaciones pentadiagonal para obtener superficie libre. Los resultados de esta implementación demuestran que se obtiene una escalabilidad casi lineal al incrementar el número de nodos usados. Se incluye también un estudio completo de los posibles errores que se pueden producir con esta implementación, demostrando que las diferencias (NRMSE) al comparar con los resultados de un modelo completo en alta resolución son pequeñas. Finalmente, se analiza la calidad de los resultados cerca de las fronteras entre subdominios. Los resultados demuestran que los patrones de recirculación son correctamente capturados, incluso aunque dichos patrones se encuentre en la zona de división entre fronteras.

Finalmente, en el capítulo 5 se pretende adaptar el modelo hidrodinámico a la arquitectura utilizada, con el objetivo de obtener una implementación paralela optimizada y escalable (OP-Si3D) que pueda usarse también en arquitecturas de altas prestaciones con más de 10 nodos. Se muestran los factores que influyen en la obtención de un modelo paralelo eficiente, tanto en máquinas de memoria compartida como distribuida. Los resultados demuestran que adaptarse a la arquitectura haciendo uso de las ideas propuestas mejora considerablemente los resultados de escalabilidad, que se pueden obtener buenos resultados con estas mejoras incluso en arquitecturas con componentes de bajas prestaciones y que todas las etapas de procesamiento deben ser optimizadas y paralelizadas para obtener los mejores resultados.

General Introduction and Objectives

Of the total amount of water in the hydrological cycle on a planetary scale, a tiny amount (0.008%) is in the form of fresh water in rivers, lakes and reservoirs. This fraction, however, is of extraordinary value to humans, since almost 80% of the water we consume comes from these surface water bodies. Despite its importance, our knowledge of freshwater ecosystems is still poor, according to the *US National Research Council*. Our knowledge gap is especially poor when considering the long-term behavior of these systems in periods of around ten years (Harris and Durran 1986, Armengol et al 1994), and short-time (daily or hourly) scales (Imberger and Parker, 1985). This lack of knowledge is partially due to the multiplicity and complexity of the physical processes of transport and mixing. These processes are a fundamental aspect of the functioning of aquatic ecosystems: not only they determine the position of particulate and dissolved substances contained in the water at any given time, but they also contribute to determine the environment in which biogeochemical reactions occur. Hence, understanding the chemical and biological properties of aquatic ecosystems, its spatial and temporal variability, requires first to build a through and sound understanding of the physical processes leading to motion and transport (Imberger 1998).

The use of hydrodynamic models capable of solving the equations of motion of fluids in three (3D) spatial dimensions, along with the application of advanced technologies for observing the water velocity and the thermal structure, have contributed in recent decades to initiate the study of the movement of water in natural systems (eg Hodges et al. 2000, Rueda and Schladow 2003). Many of these models are based on the numerical solution of a simplified form of the 3D Navier-Stokes equations known as the *Shallow Water Equations*, SWE. In these models it is assumed that the pressure distribution in the vertical direction is hydrostatic. This form of the equations is preferred over non-hydrostatic equations for its lower computational cost. Even so, hydrostatic models continue to be inefficient and heavy to run when high resolution spatial grids are needed to resolve processes with small spatial scales (such as horizontal vortices and eddies or gravity currents) and, especially, if PCs or low-end servers are used.

Fortunately, many important advances, which have been made in recent decades in the area of computing, have enabled a significant boost in research in different areas of knowledge, including the simulation of water with hydrodynamic models. Currently it is unquestionable the need to use computers for all kinds of scientific activities and the need to apply advanced computer techniques to reduce the enormous computational cost and the huge amount of memory required in the implementation of 3D simulations. As a result of these progresses large problems with large computational requirements, which cannot be solved sequentially, can now be addressed. This is possible by reducing the computational costs and allowing simulation problems which use high resolution grids, something unthinkable to perform within a reasonable computational time a few years ago.

The general objective of this thesis is to improve the computational efficiency of existing hydrodynamic models to address the rigorous and detailed study of processes of transport and mixing in inland water bodies. Different solutions will be shown to reduce the computational cost of these simulations, so that they can obtain useful results within an acceptable time with high-resolution grids, using both resources easily accessible for any scientist and High Performance Computing (HPC). The optimization strategies used have improved the efficiency of these models. The efficiency is understood as the relationship between performance and cost and the relationship between performance and quality of results. These optimization strategies should facilitate and expand the use of 3D models of transport and mixing as working tools in the investigation applied to the knowledge of lakes, reservoirs and rivers; or as forecasting tools. To make this possible, the approach presented is able to perform the processing required in an acceptable time on personal computers, low-end servers and small commodity clusters, avoiding the use of expensive high-end servers. For the more expensive cases, it is also presented an alternative using HPC, efficiently implemented.

The general objective of improving the computational efficiency of existing hydrodynamic models is achieved by two different proposals. The first one is efficiently taking advantage of the architecture of general purpose processors (Anguita et al. 2009, Anguita and Martinez-Lechado 2005). The efficient exploitation of the architecture has been proven very profitable in various applications where it is necessary to treat large amounts of data and perform lots of calculations. Among these applications are audio processing, graphics, images and video (Anguita et al applications. 2009, Anguita and

Martinez-Lechado 2005, Bhattacharjee et al. 2008, Guobin et al. 2005). The efficient exploitation of the architecture has been also found useful for applications in other fields such as bioinformatics, financial or seismic data processing (Ino et al. 2009, Lee et al 2009, Panetta et al. 2009, Zhang and Oosterlee 2009) and, of course, in the field of 3D-SWE (Fringer et al. 2006, 2010 Nesterov, Amritkar et al. 2012, Tsai and Tubbs 2009)).

The second approach to reduce the computation cost uses a procedure known as nesting. This method has been traditionally used in the literature to reduce computation costs (Fox et al. 1995 and Pinardi Zavatarelli 2003). Using this procedure it is possible to reduce the high resolution model only to the area of interest, solving the rest of the grid with lower resolution and, therefore, reducing the computational cost and storage. However, this procedure is insufficient when the high resolution area of study is large such as the littoral zone of large lakes. Littoral habitats of lakes can be substantially heterogeneous in both vertical and horizontal dimensions (Lodge et al. 1988). Physical conditions exhibit continuous and very dynamic changes, at short-time scales, as a result of strong hydrodynamic forcing and the weak inertia of shallow layers (Lodge et al. 1988). Given the extraordinarily variable dynamics of near-shore or littoral habitats, these are sites with large biodiversity (Vadeboncoeur et al. 2011) and critical habitats for many organisms in lakes (Kalff 2001). Furthermore, beaches or bays along the near-shore areas cannot be understood in isolation from neighbor bays, given that they are tightly linked through physical processes. As reviewed by Rao and Schwab (2007), currents in the nearshore are largely aligned along isobaths, hence, creating strong physical links between the littoral zone along the whole perimeter of lakes.

In this thesis an efficient implementation of a nesting procedure is proposed. Besides, this procedure is combined with computer clusters of medium/low range, thus further reducing the computational costs and allowing efficiently resolve all the littoral zone (impossible in an acceptable time even using a normal nesting procedure) of large lakes in high resolution, which despite its enormous diversity and importance as it is explained above, our understanding of these near-shore habitats is poor (Kalff 2001). According to this, some useful and interesting results are shown to the scientific community that indicate the path to follow in the study of the littoral zone of large lakes and demonstrate the need for using high resolution grids to perform these simulations correctly.

The solutions presented in this work have been implemented on a hydrodynamic model based on a numerical solution of the 3D shallow water equations, Si3D. In this model the equations are discretized using a semi-implicit, three-level, iterative leapfrog-trapezoidal finite difference algorithm on a staggered Cartesian grid, and the computations are carried mostly on water column by water column basis. The semi-implicit algorithms existing nowadays follow the original proposal designed by Professor Casulli (Casulli and Cheng 1992; Casulli and Cattani 1994). These algorithms applied to structured and unstructured grids have been widely used in the simulation of inland water bodies (eg Hodges et al 2000, Appt et al 2004, Laval et al 2003) in recent years. The semi-implicit model, where the optimization strategies have been inplemented, has recently been used in several studies about transport and mixing in rivers, reservoirs and lakes (Rueda et al. 2009, Rueda and Schladow 2009, Rueda and MacIntyre 2009, Hoyer et al 2014, Ramón et al 2013). Si3D was developed and originally proposed by the US Geological Survey (Smith 2006) and later adapted for simulation of lakes by Rueda (2001).

Chapter 1 contains a complete description of this model, including the governing equations and numerical model used, the execution flow of the algorithm and other computational details of interest that demonstrate the complexity of some of the operations performed in these semi-implicit models and the enormous computational cost (in terms of execution time and memory requirements) needed. The remaining chapters have been written as stand-alone independent articles with their own abstracts, introductions, methodology, results, discussion and conclusions. Some of these chapters are already published. Others are currently under review in international journals or in process of being submitted. The fact that they are written as individual articles means that some of the concepts can be repetitive for the reader but, on the other hand, each chapter can be read independently. Figure i.1 shows the different implementations developed from the original code of Si3D. The first step was to apply several basic optimizations to the original Si3D code (Basic Si3D). These optimizations are used in all the implementations made (although this implementation is the first step, it is covered in detail in the last chapter, where it is explained all the steps involved in the complete optimization of a hydrodynamic model). The chapters are arranged in a logical order related with each of the developed implementations. The second chapter contains the description, evaluation and validation of the nesting procedure proposed (N-Si3D).

In chapter 3 a suitable parallel implementation for small commodity clusters (P-Si3D) is presented. Besides, a practical application example is presented for the littoral zone of Tahoe. This simulation requires the combination of the parallel implementation developed and the nesting procedure proposed in the previous chapter (P/N-Si3D). Chapter 4 describes and evaluates other of the parallel implementations developed: SP-Si3D, which adds an implementation with a low-resolution grid to the parallel implementation with the high resolution grid. The proposed implementation modifies P-Si3D in order to use the results of the low resolution model to eliminate the collective communications one-to-many and many-to-one needed in parallel semi-implicit hydrodynamic models. This implementation scales almost linearly. Finally in chapter 5, several approaches are explored and tested to optimize P-Si3D, which are included under an implementation called OP-Si3D. With these approaches the parallel implementation is adapted to the architecture of a general purpose platform used nowadays. In this manner, the overhead caused by the parallel implementations is minimized resulting in a scalable version of the code, even if used in high performance platforms. Below the specific objectives of each chapter and their contents are briefly introduced.

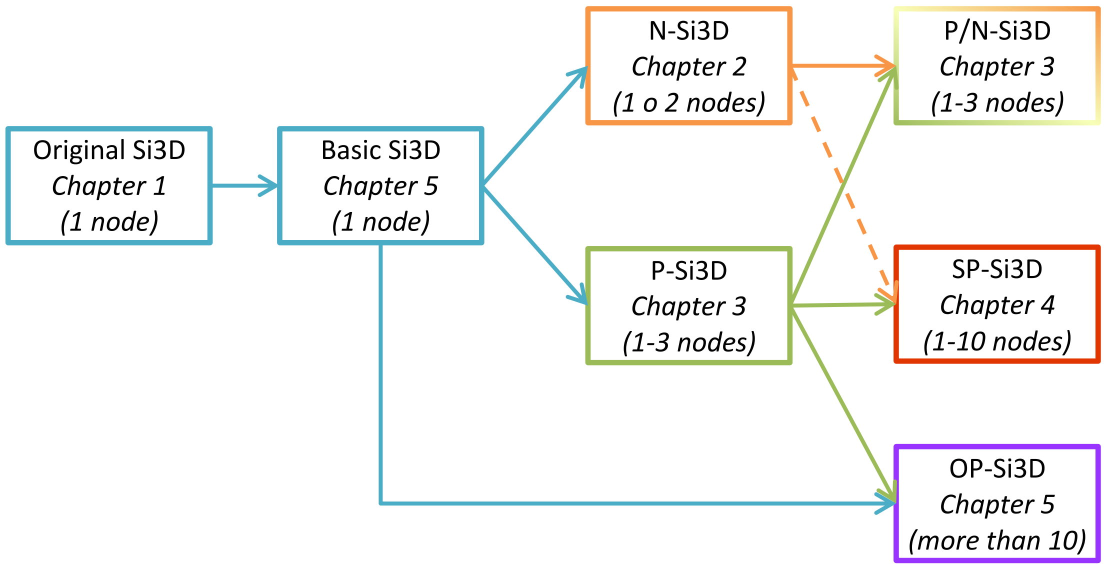


Figure i.1. Si3D implementations proposed in this work from the original version of Si3D

The goal in chapter 2 is to develop efficient procedures to create a seamless implementation of a nested model with Si3D (N-Si3D). In the nested model, only the area of interest is simulated using a high-resolution grid. This is the inner-model. The equations in the inner model are solved subject to boundary conditions provided by a low-resolution model of a larger domain, the outer-model. Hence, the computational cost of the simulations can be significantly reduced. A seamless nested solution can be obtained only if the value of all terms in the mass and momentum conservation equations for those nodes in the inner-domain adjacent to the boundary with the outer model can be correctly calculated. In particular, we focus on the need to transfer the tangential velocity components from the outer- to the inner-model. An efficient method to run the outer and the inner model simultaneously, in parallel, is also introduced. The sources of errors arising in the nested solution are studied. We further illustrate, through case examples, the benefits that we get, in terms of processes resolved, from using refined grids in local-scale regions of lakes and rivers.

In Chapter 3, the objective is to develop a parallel implementation of Si3D (P-Si3D) suitable for small multi-core clusters. In this implementation, the workload is split among different processes/threads, each one working with different sets of vertical columns in which the domain is decomposed. Only the matrix problem controlling the free-surface elevation is solved sequentially, hence, avoiding the complexity involved of parallelizing this stage in the computations. The sequential solution of the free-surface-elevation matrix requires that a collective communication many-to-one is used to collect the contributions of different water columns to that matrix before the solution process starts. It also requires the use of a collective communication one-to-many after finishing the stage to broadcast the solution obtained. As a result of the collective communications many-to-one and one-to-many and the sequential calculation of the free-surface-elevation equation system, this implementation does not scale correctly in clusters of more than 3 nodes. The parallel model though is simple to implement and reduces the execution time acceptably for models using low and medium resolution in small commodity clusters. We further analyze the influence of different hardware configurations and the domain decomposition process on model performance. Finally, the parallel implementation of Si3D is used to develop a nested model of the littoral perimeter of Lake Tahoe. This implementation is used, for illustrative purposes, to reveal some characteristic features of the long-shore transport in Lake Tahoe.

Our goal in Chapter 4, is to develop a parallel implementation of Si3D (SP-Si3D) to resolve near-shore processes in large lakes which scales in low range multi-core cluster of about 10 nodes. The approach proposed modifies the implementation P-Si3D in order to use the results of a low-resolution model. In this manner the collective communications one-to-many and many-to-one needed to solve the matrix for the free-surface elevation in serial mode are eliminated. The errors in the implementation are analyzed. In particular, we analyze the quality of the solution near the boundaries between sub-domains. The results show recirculation patterns are correctly simulated even if a subdomain boundary crosses the recirculation area.

Finally, in Chapter 5 several approaches are proposed to adapt the hydrodynamic model to the architecture used. This is done in order to obtain an optimized and scalable parallel implementation (OP-Si3D) that can also be used in high performance architectures with more than 10 nodes. It is shown which factors influence in obtaining an efficient parallel model, both in shared memory and distributed platforms. The results demonstrate that if the implementation is adapted to the architecture using the ideas proposed, (1) the scalability results improves significantly, (2) it is possible to obtain good scalability results with these improvements even in architectures with low-performance components and (3) all processing stages must be optimized and parallelized in order to obtain the best results.

Chapter 1

A Semi-implicit, three-dimensional hydrodynamic model

Abstract

In this chapter, it is presented the three-dimensional (3D) Semi-Implicit hydrodynamic model, Si3D, where the different optimization strategies presented in this thesis will be applied. Si3D is a semi-implicit, finite-difference method based on the numerical solution of the 3D Navier-Stokes Shallow Water Equations (3D-SWE) proposed by Smith (2006). The equations are first posed in layer-averaged form by integrating over the height of a series of horizontal layers separated by level planes. With this approach, the volumetric transports and not the velocities are the dependent variables in the model equations. These layered-averaged equations are discretized using a semi-implicit, three-level, iterative leapfrog-trapezoidal algorithm on a staggered Cartesian grid. The semi-implicit approach is based on treating the surface gravity wave and vertical diffusion terms implicitly to avoid time-step limitations due to gravity wave Courant–Friedrich–Levy (CFL) conditions, and to guarantee stability of the method. All other terms-including advection-are treated explicitly. Laplacian operators are used to represent mixing. Constant mixing coefficients are used to parameterize the effect of horizontal eddies. A two-equation turbulence model calculates the vertical eddy coefficients of mixing. The time-step of the resulting model is only subject to CFL restrictions associated to the explicit treatment of advection and baroclinic pressure gradient terms. The resulting method is mass conservative, efficient, and numerically accurate. However, the method is also computationally expensive, using a huge quantity of variables to save information of the Cartesian grid in three different states and solving some complex operations. At each time-step, Si3D solves a large number of small tridiagonal equation systems using a double-sweep method (known as Thomas algorithm) and then uses an iterative method known as Preconditioned Conjugate Gradient to solve a single, large, pentadiagonal equation system for the water surface elevation.

1.1 Introduction

The earliest 3D hydrodynamic models were developed in the 1960s for applications to oceans and lakes. These models are based in an approach known as the rigid-lid approximation in which the free surface is held constant and the surface gravity waves (including ocean tides and lake seiches) are filtered out of the solutions (i.e. Bryan (1969) for ocean model and Liggett (1969) for lakes). These models were able to employ large time-steps so that long-term simulations of large regions could be done economically. For studying large-scale oceanic circulation on a coarse numerical grid, rigid-lid models generally are adequate and are still being developed for modern applications. For lakes, rigid-lid models should only be used in studies focusing on time periods much longer than the dominant seiching period of the lake (Sheng, 1986b). Sheng et al. (1978) compared a rigid-lid model with a model that solves directly for the variable free surface (free-surface model) and showed the rigid-lid model gave poor results for periods of active seiching of Lake Erie under spatially and temporally varying winds. A study by Huang and Sloss (1981) used a rigid-lid model for Lake Ontario and obtained reasonably good results for the mean monthly circulation.

Leendertse et al. (1973) and Leendertse and Liu (1975, 1977) were the first to develop a 3D model for estuaries and coastal seas incorporating a fully time-varying free-surface location and using standard finite-difference grid boxes in all three dimensions. The details of the model formulation were thoroughly described in the series of Rand Corporation reports, which benefitted future investigators in the field. The significant drawback of the Rand model was its explicit finite-difference scheme that limited the size of the time-step to the time a surface gravity wave takes to travel between two adjacent horizontal grid points, a limitation referred to as the Courant-Friedrich-Lewy (CFL) stability condition for the gravity waves. When using a high resolution numerical grid in an estuary with areas of deep water, this limitation can be very severe.

In explicit finite difference models, the unknown hydrodynamic variables at any spatial point at next time level are calculated entirely from known values at neighboring points from one or more previous time levels. This scheme forms equation systems solved easily but, at the same time, requires higher computational time due to the integration time limitation of CFL. To reduce the computation time, a mode-splitting technique can be used. In general, mode-splitting is a separation of the 3D governing equations into a set of equations describing the two-dimensional (2D) depth-mean flow (the external mode) and a set describing the vertical structure of flow (the internal mode). The time discretization for the gravity-wave terms in the external-mode equations can be either explicit or implicit. In an implicit scheme, the unknown hydrodynamic variables at a given spatial point at next time level depend on the unknown variables at neighboring points at the same time level. The unknown variables are then obtained by the simultaneous solution of a system of algebraic equations.

In a mode-splitting approach, the internal-mode equations are treated explicitly except for the vertical diffusion terms, which usually are treated implicitly to avoid a time-step limitation in shallow water (Davies, 1985). By using an explicit time discretization for the external-mode gravity-wave terms, the time-step must be small enough to satisfy the gravity-wave CFL condition. By applying a time-splitting algorithm, however, the internal-mode solution can be integrated by using a much larger time-step than the external mode. Because the internal-mode solution can be expensive in terms of computation time, significant economies are gained by solving the internal mode explicitly with a large time-step (usually at least an order of magnitude larger than the time-step for the external mode). By using an implicit time discretization for the external-mode gravity-wave terms, the CFL limitation can be avoided, although this advantage is partially offset by the additional complexities involved in an implicit formulation (including the need to solve a matrix system of algebraic equations each time-step). Implicit mode-splitting, however, allows the external and internal modes to be solved using the same long time-step. In any case, the use of mode-splitting will usually lead to solutions that are far more computationally efficient than those that solve the primitive 3D equations directly using a fully explicit scheme. This method has since been adopted for use in 3D shallow-water circulation modeling (i.e. Blumberg and Mellor 1987, Sheng 1983, Jin 1993, Chapman et al. 1996). In fact, most 3D shallow-water models now being used are mode-splitting models. The mode-splitting technique is also being implemented for modeling the large-scale circulation of the ocean (Killworth et al. 1991, Dukowicz and Smith 1994, Semtner 1995).

Although mode-splitting has become widely accepted in 3D modeling, it has several important drawbacks that are often overlooked. If an explicit time discretization is used for both modes of a time-splitting scheme, the external mode (2D) velocities must be the exact depth average of the internal mode (3D) velocities; otherwise, the computations will become unstable (Dukowicz and Smith 1994). If an implicit time discretization is used for the external-mode gravity waves in a mode-splitting model, time-splitting errors can be eliminated if the external and internal mode time-steps are chosen to be equal. However, the separate calculations of the 2D and 3D variables lead to difficulties in consistently representing the magnitude of the bottom frictional stress between the external and internal modes.

Instead of mode-splitting, some researchers have used other forms of splitting methods. For example, the latest versions of the 3D Rand Corporation model (Leendertse 1989) and the 3D TRISULA model from Delft Hydraulics of the Netherlands (Uittenbogaard et al. 1992) are based on one of the best known splitting techniques—the alternating-direction-implicit (ADI) method. These models are basically 3D extensions of ADI methods successfully used in two dimensions (Leendertse 1987, Stelling 1984). These models are formulated with second-order accuracy and treat the vertical diffusion term implicitly. Leendertse (1989) is especially critical of the use of mode-splitting because it "degrades the accuracy of computation" to first order. To eliminate these inaccuracies, either the time-step or horizontal grid size of a simulation must be decreased, which can have a significant effect on model efficiency.

Two alternatives to the splitting models are the time-splitting and semi-implicit methods. The first one is the implicit, time-splitting finite-difference scheme presented in de Goede (1991). This model does not employ ADI methods so that inaccuracies caused by the ADI effect are absent, even for large time-steps. The 3D scheme has a strong resemblance to the 2D scheme described in Wilders et al. (1988). The 3D scheme, which neglects advection, density-forcing, and horizontal shear stress terms, is based on a two-stage splitting procedure in which the first stage requires the solution of a large number of independent tridiagonal systems of equations involving the implicit treatment of vertical diffusion. In the second stage, the terms describing the propagation of the surface gravity waves (that is, the water surface pressure gradient in the momentum equations and the velocity divergence in the continuity equation) are treated implicitly. This stage results in a pentadiagonal matrix system to be solved for the water surface elevation. Once the water surface elevation is known, the velocities are computed explicitly.

The other alternative to the splitting models is the semi-implicit method proposed by Casulli and Cheng (1992) and implemented in other models (Smith 2006, Hodges et al. 2000). This method is similar to the scheme presented by de Goede (1991), where gravity waves and vertical diffusion terms are treated implicitly and other terms

(Coriolis, horizontal friction, advection and baroclinic pressure term) are treated explicitly. However, this scheme does not use a splitting procedure to avoid the time-step limitation due to CFL condition. The scheme proposed by Casulli and Cheng (1992) and reformulated by Casulli and Cattani (1994) is a two-level scheme. In this model, the advection terms in the momentum equations are discretized by using an unconditionally stable, semi-Lagrangian or Euler-Lagrangian method (ELM) with linear interpolation. The schemes solve the governing equations in a nonconservative form that is consistent with the use of an ELM.

The semi-implicit scheme implemented in the model presented here (Si3D) and developed by Smith (2006) closely follows the approach outlined by Casulli and Cheng (1992) for the implicit inclusion of vertical diffusion in a 3-D calculation without recourse to mode-splitting. The details of the overall scheme, however, differ from those in Casulli and Cheng (1992) in many significant ways. The time integration used here is a three-time-level, semi-implicit, leapfrog-trapezoidal method. Except for a small amount of first-order error introduced by the uncentered (in time) treatment of horizontal diffusion for stability considerations, the scheme is second-order accurate in the truncation errors of the finite-difference approximations in both space and time. All terms (other than horizontal diffusion), as well as nonlinear coefficients, are centered in time during the leapfrog and trapezoidal steps of the scheme to achieve second-order accuracy in time. The accurate evaluation of the nonlinearities in the equation of motion is important, because nonlinearities have a significant effect on the tidally averaged circulation in shallow estuaries. The governing equations for the three-level scheme are prepared in a conservative form by integrating them over the height of each layer. The layer-integrated, volumetric transports replace velocities as the dependent variables so that the depth-integrated continuity equation that is used in the solution for the water surface elevation is linear. The advection terms in the momentum and transport equations are solved using explicit, leapfrog-trapezoidal integration rather than an ELM as used by Casulli and Cheng (1992). The leapfrog-trapezoidal approach does very well with the conservation of transport equation (salinity), which can be a problem with the ELM approach. Instead of the improvements implemented, the computational cost of the semi-implicit scheme, in terms of required memory and execution time, is still expensive. At each time-step, Si3D solves a large number of small tridiagonal equation systems using a double-sweep method known as Thomas algorithm and a single, large, pentadiagonal equation system for the water surface elevation using an iterative method known as Preconditioned Conjugate Gradient.

In this chapter, it is presented the hydrodynamic model based in the 3D-SWE and discretized using a semi-implicit, three-level, iterative leapfrog-trapezoidal finite difference algorithm on a staggered Cartesian grid. Extensive details about the governing equations, numerical algorithm and computational implementation of Si3D are presented in the next section. This model was developed for estuarine circulation and serial architectures by Smith (2006). Subsequently, Si3D was extended and adapted to lakes by Rueda (2001). Rueda (2001) modified the model to use temperature instead of salinity as the active scalar responsible for stratification, to use variable wind fields, to calculate heat fluxes based on the prognostic variables, and to calculate the momentum and scalar turbulent transfer coefficients with a high-order turbulence closure (Mellor-Yamada Level 2.5). The transport algorithm was modified to eliminate spurious oscillations in the temperature field that could interfere in the calculation of turbulent transfer coefficients. Additional modules have been developed to simulate the transport of tracers. Si3D is a public code programmed for serial architectures by the US Geological Survey (USGS), which provided us with a free version.

1.2 Hydrodynamic model

1.2.1 Governing equations

The model is based on the numerical solution of the 3-D Shallow Water Equations (Smith 2006), a simplified form of the Reynolds Averaged Navier-Stokes (RANS) (Cushman-Roisin 1994). Practical computational limits and a priori scaling analyses justify the use of the 3D-SWE in the description of these large-scale flows. In RANS models, state variables describing the fluid motion are decomposed into a mean and a fluctuating part, and separate governing equations are posed for the averaged variables and for the averaged products of the fluctuating quantities. Assuming that (1) density is negligible everywhere except in the buoyancy term (the Boussinesq approximation), (2) the weight of the fluid balances the pressure in the equation for vertical momentum (the hydrostatic approximation), and (3) a diffusion-like term can be used to represent turbulent fluxes of scalars and momentum (the eddy diffusivity concept), the governing equations can be written as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{1.1a}$$

$$\frac{\partial \zeta}{\partial \mathrm{t}} + \frac{\partial}{\partial x}\left[\int_{-H}^{\zeta} u dz \right] + \frac{\partial}{\partial y}\left[\int_{-H}^{\zeta} v dz \right] = 0 \tag{1.1b}$$

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv = -\mathrm{g}\frac{\partial \zeta}{\partial x} + \frac{g}{\rho_0}\int_z^{\zeta} \frac{\partial \rho}{\partial x} dz' \\ + \frac{\partial}{\partial x}\left(K_H \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y}\left(K_H \frac{\partial u}{\partial y} \right) + \frac{\partial}{\partial z}\left(K_V \frac{\partial u}{\partial z} \right) \end{aligned} \tag{1.2}$$

$$\begin{aligned} \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu = -g\frac{\partial \zeta}{\partial y} + \frac{g}{\rho_0}\int_z^{\zeta} \frac{\partial \rho}{\partial y} dz' \\ + \frac{\partial}{\partial x}\left(K_H \frac{\partial v}{\partial x} \right) + \frac{\partial}{\partial y}\left(K_H \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial z}\left(K_V \frac{\partial v}{\partial z} \right) \end{aligned} \tag{1.3}$$

$$\begin{aligned} \frac{\partial T}{\partial t} + \left[\frac{\partial (uT)}{\partial x} + \frac{\partial (vT)}{\partial y} + \frac{\partial (wT)}{\partial z} \right] = \\ \frac{\partial}{\partial x}\left(D_H \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y}\left(D_H \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z}\left(D_V \frac{\partial T}{\partial z} \right) + \frac{1}{\rho c_p}\frac{\partial I}{\partial z} \end{aligned} \tag{1.4}$$

$$\rho = \rho(T) \tag{1.5}$$

These equations comprise the 3D-SWE. They express the physical principles of conservation of mass for an incompressible fluid (Eqs. (1.1a)-(1.1b)), conservation of momentum (Eqs. (1.2)-(1.3)) and conservation of energy (Eq. (1.4)). The velocity components in the *x*, *y*, and *z* directions are denoted by *u*, *v*, and *w*, *t* is time, *f* is the Coriolis parameter, *g* is the acceleration of gravity, *ρ* represents water density variation with respect to a mean reference value ρ_0, ζ is the water surface elevation above an undisturbed level at z = 0, and z = -H (x, y) is the depth of the bottom boundary measured from z = 0. Equation 1a is the three-dimensional form of the continuity equation, while Eq. (1.1b) results from 1a by integration over the water depth, accounting for the kinematic boundary condition at the free surface, and is the equation governing the free surface position. Equation (1.5) is an equation of state which links temperature, T, and density, ρ. The last term in Eq. (1.4) is a source-sink term representing the surface heat flux, and c_p is the heat capacity of water: $\Delta s = {}^{1}\!/_{\rho c_p} {}^{\partial I}\!/_{\partial z}$. The coefficients K_H and K_V represent the horizontal and vertical turbulent momentum transfer coefficients (or kinematic eddy viscosity), and D_H and D_V are the horizontal and vertical turbulent transfer coefficients (eddy diffusivity) for temperature. Those terms represent the effects of the fluctuating quantities in the mean equations. Different approaches can be followed in order to estimate vertical turbulent diffusivities. Most commonly separate differential equations are solved for the turbulent kinetic energy and a turbulent length-scale determining variable (such as the dissipation of turbulent kinetic energy or the product of the turbulent kinetic energy and a turbulent macroscale l) from which turbulent transfer coefficients are calculated. Si3D was also adapted for 3-D lake models (Rueda 2001) using a modified formulation of the 2.5 level Mellor-Yamada turbulence closure scheme to parameterize vertical mixing (Kantha and Clayson 1994). This formulation incorporates the quasi-equilibrium turbulence model (QETE) of Galperin et al. (1988) with readjusted parameters, and a parameterization of shear-instability-induced mixing in the strongly stratified region below the surface mixed layer. It was developed to correct some of the deficiencies of the original formulation of the Mellor-Yamada 2.5-Level closure scheme, which was shown to misrepresent the depth of the surface mixed layer in oceanic applications (e.g. Martin 1985). Two partial differential equations are used to calculate the turbulent kinetic energy, $q^2/2$, and a turbulent macroscale, l, given by

$$\frac{\partial q^2}{\partial t} - \frac{\partial}{\partial z}\left(lqS_q \frac{\partial q^2}{\partial z} \right) = 2\left[P_s + P_b - \varepsilon \right] \tag{1.6}$$

$$\frac{\partial \left(q^2 l \right)}{\partial t} - \frac{\partial}{\partial z}\left(lqS_q \frac{\partial (q^2 l)}{\partial z} \right) = lE_1\left[P_s + P_b \right] - \frac{q^3}{B_1}\left[1 + E_2\left(\frac{l}{\kappa z} \right)^2 + E_3\left(\frac{l}{\kappa (H - z)} \right)^2 \right] \tag{1.7}$$

Here, κ is the von Karman constant, and P_s, P_b and ε stand for shear production, buoyant production and the dissipation of turbulent kinetic energy, given by

$$P_s = K_v\left(\frac{\partial u}{\partial z}\right)^2 + K_v\left(\frac{\partial v}{\partial z}\right)^2 ;\ P_b = \frac{g}{\rho_0} D_v \frac{\partial \rho}{\partial z} ;\ \varepsilon = \frac{q^3}{B_1 l} \tag{1.8}$$

in which B_1 is an empirical constant. The boundary conditions applied at the top and the bottom boundaries are calculated from the friction velocity u_* as (Blumberg 1986)

$$q^2 = B_1^{2/3}(u_*)^2,\ q^2 l = 0 ; \tag{1.9}$$

The turbulent fluxes of momentum (u, v) and temperature (T) are represented using a diffusion model and the concepts of eddy viscosity and diffusivity. The eddy viscosity and diffusivity are calculated from q and l as $K_V = S_m lq$ and $D_V = S_h lq$ in which S_m and S_h are stability functions that account for the effects of stratification on mixing. They are calculated as (Kantha and Clayson 1994)

$$S_h = \frac{A_2\left(1 - \frac{6A_1}{B_1}\right)}{1 - 3A_2 G_h (6A_1 + B_2(1 - C_3))} \tag{1.10}$$

$$S_m = A_1 \frac{\left(1 - \frac{6A_1}{B_1} - 3C_1\right) + 9\left[2A_1 + A_2(1 - C_2)\right] S_h G_h}{(1 - 9A_1 A_2 G_h)} \tag{1.11}$$

$$G_h = -\left(\frac{Nl}{q}\right)^2 \tag{1.12}$$

The values of the constants in the model are (A_1, A_2, B_1, B_2, C_1, C_2, C_3, S_q, E_1, E_2) =(0.92, 0.74, 16.6, 10.1, 0.08, 0.7, 0.2, 0.2, 1.8, 1.33) as suggested by Kantha and

Clayson (1994). G_h is bounded from above, so that the variance of the velocity fluctuations are positive definite, according to

$$G_h \leq \{A_2 [B_1 + 12A_1 + 3B_2 (1 - C_3)]\}^{-1} \tag{1.13}$$

The upper bound on G_h is set to 0.029 (Kantha and Clayson 1994). The lower bound on G_h reflects that under strong stable stratification conditions, there is a limit to the size beyond which eddies are incapable of overturning. This bound is dictated by the Ozmidov scale, $L_o = (\varepsilon/N^3)^{1/2}$, and is given by $G_h > -0.28$.

1.2.2 Numerical algorithm

The governing equations are first posed in layer-averaged form by integrating over the height of a series of horizontal layers separated by level planes. With this approach, the volumetric transports (the product of velocities by the layer thickness, U and V) and not the velocities (u, v) are the dependent variables in the model equations. This layer-averaged form of the equations is discretized using a semi-implicit, three-level, iterative leapfrog-trapezoidal finite difference algorithm on a staggered Cartesian grid (Arakawa-C grid, which has velocities and volumetric transports defined on cell faces and scalar quantities such as water surface elevation and temperature defined at the cell center). The semi-implicit approach is based on treating the gravity wave and vertical diffusion terms implicitly to avoid time-step limitations due to gravity wave Courant–Friedrich–Levy (CFL) conditions, and to guarantee stability of the method (Casulli and Cheng 1992). The vertical diffusion term in the transport equation is also treated implicitly. The remaining terms in the momentum equations (advection, Coriolis, horizontal momentum diffusion, and the baroclinic pressure gradient) and advection terms in transport equation are solved using explicit leapfrog-trapezoidal integration. The time-step of the resulting model Δt is only subject to CFL restrictions associated to the explicit treatment of advection and baroclinic pressure gradient terms. The resulting method is mass conservative, efficient, and numerically accurate (Smith 2006).

The Numerical Algorithm is divided into four stages (Figure 1.1). First, the finite-difference form of the continuity (Eq. (1.1b)) is written as follows

$$\zeta^{n+1}_{i,j} = \zeta^{n-1}_{i,j} - \frac{\Delta t}{\Delta x}\left[\sum_{k=1}^{km}(U^{n+1}_{i+1/2,j,k} - U^{n+1}_{i-1/2,j,k} + U^{n-1}_{i+1/2,j,k} - U^{n-1}_{i-1/2,j,k})\right] - \frac{\Delta t}{\Delta y}\left[\sum_{k=1}^{km}(V^{n+1}_{i,j+1/2,k} - V^{n+1}_{i,j-1/2,k} + V^{n-1}_{i,j+1/2,k} - V^{n-1}_{i,j-1/2,k})\right] \tag{1.14}$$

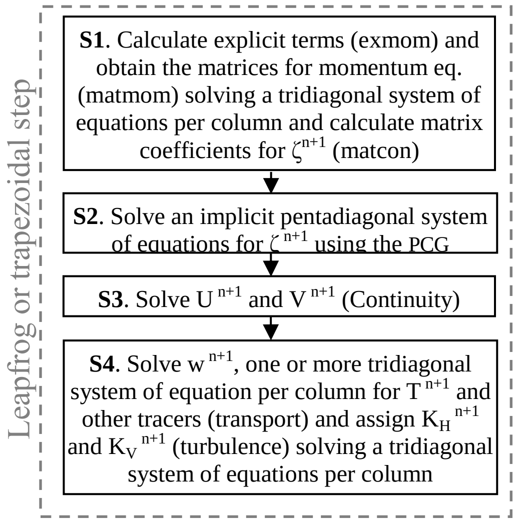

Leapfrog or trapezoidal step
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ^{n+1} (matcon)
S2. Solve an implicit pentadiagonal system of equations for ζ^{n+1} using the PCG
S3. Solve U^{n+1} and V^{n+1} (Continuity)
S4. Solve w^{n+1}, one or more tridiagonal system of equation per column for T^{n+1} and other tracers (transport) and assign K_H^{n+1} and K_V^{n+1} (turbulence) solving a tridiagonal system of equations per column

Figure 1.1. Solver stages of Si3D Numerical Algorithm.

The solution of the momentum equations is separated into two phases in which first the explicit terms and then the implicit terms are evaluated in the stage 1 column by column of the grid (S1, Figure 1.1). The finite-difference equation for the *x*-direction momentum equation is written so it is centered within layer *k* at the horizontal point *(i+1/2)* Δ*x*, *j*Δ*y;* the explicit phase is represented by

$$\hat{U}_{i+\frac{1}{2},j,k} = U^{n-1}_{i+\frac{1}{2},j,k}\ 2\Delta t[-(ADV_x)^n + (COR_x)^n - (BCLINIC_x)^n + (HDIFF_x)^{n-1}]_{i+\frac{1}{2},j,k} \tag{1.15}$$

The corresponding finite-difference equation for the explicit phase of the *y*-direction momentum equation is centered within layer *k* at the point *i*Δ*x*,*(j+1/2)*Δ*y* and is represented by

$$\begin{aligned}\widehat{U}_{i,j+\frac{1}{2},k} = {} & U^{n-1}_{i,j+\frac{1}{2},k} 2\Delta t \Big[-\left(ADV_y\right)^n + \left(COR_y\right)^n - \left(BCLINIC_y\right)^n \\ & + \left(HDIFF_y\right)^{n-1} \Big]_{i,j+\frac{1}{2},k}\end{aligned} \tag{1.16}$$

The bracketed terms (calculated in S1) are expanded fully in Smith 2006. The symbol (ˆ)in equation denotes a solution for the layer volumetric transport which includes only the contribution from the explicit terms. All terms in equations (1.15) and (1.16), except horizontal diffusion (*HDIFF*), are centered in time at time level *n* to achieve second-order numerical accuracy. The horizontal diffusion is written backward-in-time at time level *n* - 1 because the centering of that term can result in a weak instability.

Then the finite-difference equation for the implicit phase of the x-momentum and y-momentum equations is written as

$$\begin{aligned} U^{n+1}_{i+1/2,j,k} = {} & \hat{U}_{i+1/2,j,k} - g\frac{\Delta t}{\Delta x}\bar{h}^{n}_{i+1/2,j,k}\left(\frac{\bar{\rho}^{n}_{i+1/2,j,1}}{\bar{\rho}^{n+1}_{i+1/2,j,k}}\right)\left(\zeta^{n+1}_{i+1,j} - \zeta^{n+1}_{i,j} + \zeta^{n-1}_{i+1,j} - \zeta^{n-1}_{i,j}\right) \\ & + \Delta t\left(K^{\;n}_{V\,i+1/2,j,k-1/2}\cdot\left(\frac{(U/\bar{h})^{n+1}_{i+1/2,j,k-1} - (U/\bar{h})^{n+1}_{i+1/2,j,k}}{\bar{\bar{h}}^{n+1}_{i+1/2,j,k-1/2}} + \frac{(u)^{n-1}_{i+1/2,j,k-1} - (u)^{n-1}_{i+1/2,j,k}}{\bar{\bar{h}}^{n-1}_{i+1/2,j,k-1/2}}\right)\right) \\ & - \Delta t\left(K^{\;n}_{V\,i+1/2,j,k+1/2}\cdot\left(\frac{(U/\bar{h})^{n+1}_{i+1/2,j,k} - (U/\bar{h})^{n+1}_{i+1/2,j,k+1}}{\bar{\bar{h}}^{n+1}_{i+1/2,j,k+1/2}} + \frac{(u)^{n-1}_{i+1/2,j,k} - (u)^{n-1}_{i+1/2,j,k+1}}{\bar{\bar{h}}^{n-1}_{i+1/2,j,k+1/2}}\right)\right) \end{aligned} \tag{1.17}$$

$$\begin{aligned} V^{n+1}_{i,j+1/2,k} = {} & \hat{V}_{i,j+1/2,k} - g\frac{\Delta t}{\Delta y}\bar{h}^{n}_{i,j+1/2,k}\left(\frac{\bar{\rho}^{n}_{i,j+1/2,1}}{\bar{\rho}^{n+1}_{i,j+1/2,k}}\right)\left(\zeta^{n+1}_{i,j+1} - \zeta^{n+1}_{i,j} + \zeta^{n-1}_{i,j+1} - \zeta^{n-1}_{i,j}\right) \\ & + \Delta t\left(K^{\;n}_{V\,i,j+1/2,k-1/2}\cdot\left(\frac{(V/\bar{h})^{n+1}_{i,j+1/2,k-1} - (V/\bar{h})^{n+1}_{i,j+1/2,k}}{\bar{\bar{h}}^{n+1}_{i,j+1/2,k-1/2}} + \frac{(v)^{n-1}_{i,j+1/2,k-1} - (v)^{n-1}_{i,j+1/2,k}}{\bar{\bar{h}}^{n-1}_{i,j+1/2,k-1/2}}\right)\right) \\ & - \Delta t\left(K^{\;n}_{V\,i,j+1/2,k+1/2}\cdot\left(\frac{(V/\bar{h})^{n+1}_{i,j+1/2,k} - (V/\bar{h})^{n+1}_{i,j+1/2,k+1}}{\bar{\bar{h}}^{n+1}_{i,j+1/2,k+1/2}} + \frac{(v)^{n-1}_{i,j+1/2,k} - (v)^{n-1}_{i,j+1/2,k+1}}{\bar{\bar{h}}^{n-1}_{i,j+1/2,k+1/2}}\right)\right) \end{aligned} \tag{1.18}$$

The subscripts denote the spatial location in the computational grid, and the superscripts the time level at which the variable is evaluated. The symbols k_1 and k_m in Eq. (1.14) denote the first (shallowest) and last (deepest) layer in a given water column. The symbol ^ in Eqs. (1.17) and (1.18) denotes the contribution from the explicit terms to the layer-averaged transport. The overbars in Eq. (1.17) on a layer height or density variable is used to represent a spatial average in the *x*-direction between adjacent values. For example,

$$\bar{h}_{i+1/2,j,k} = \frac{h_{i,j,k} + h_{i+1,j,k}}{2} \tag{1.19a}$$

Also,

$$\bar{\bar{h}}_{i+1/2,j,k-1/2} = \frac{\bar{h}_{i+1/2,j,k-1} + \bar{h}_{i+1/2,j,k}}{2} \tag{1.19b}$$

The overbars in Eq. (1.18) represent a spatial average between adjacent values, as in Eq. (1.17), but this time in the *y*-direction, i.e.

$$\bar{h}_{i,j+1/2,k} = \frac{(h_{i,j,k} + h_{i,j+1,k})}{2} \tag{1.20a}$$

and similar to Eq. (1.20),

$$\bar{\bar{h}}_{i,j+1/2,k-1/2} = \frac{\bar{h}_{i,j+1/2,k-1} + \bar{h}_{i,j+1/2,k}}{2} \tag{1.20b}$$

In the solution process, the unknown values of the variables *U* and *V* at time n+1 (Eq. (1.17) and (1.18)) are expressed in terms of the unknown values of the free surface. The sum of the volume transports across any of the four sides of a given water column can be then expressed as follows (the details of these calculations can be found in Smith (2006) in terms of the free surface

$$\sum_{k=k_1}^{k_m} U_{i+1/2,j,k}^{n+1} = -g\frac{\Delta t}{\Delta x}\rho_0\left[\Xi A^{-1}R\right]_{i+1/2,j}(\zeta_{i+1,j}^{n+1}-\zeta_{i,j}^{n+1})+\left[\Xi A^{-1}G\right]_{i+1/2,j}$$
$$= -ar_{i+1/2,j}(\zeta_{i+1,j}^{n+1}-\zeta_{i,j}^{n+1})+ag_{i+1/2,j} \tag{1.21a}$$

$$\sum_{k=k_1}^{k_m} U_{i-1/2,j,k}^{n+1} = -g\frac{\Delta t}{\Delta x}\rho_0\left[\Xi A^{-1}R\right]_{i-1/2,j}(\zeta_{i,j}^{n+1}-\zeta_{i-1,j}^{n+1})+\left[\Xi A^{-1}G\right]_{i-1/2,j}$$
$$= -ar_{i-1/2,j}(\zeta_{i,j}^{n+1}-\zeta_{i-1,j}^{n+1})+ag_{i-1/2,j} \tag{1.21b}$$

$$\sum_{k=k_1}^{k_m} V_{i,j+1/2,k}^{n+1} = -g\frac{\Delta t}{\Delta x}\rho_0\left[\Xi A^{-1}R\right]_{i,j+1/2}(\zeta_{i,j+1}^{n+1}-\zeta_{i,j+1}^{n+1})+\left[\Xi A^{-1}G\right]_{i,j+1/2}$$
$$= -ar_{i,j+1/2}(\zeta_{i,j+1}^{n+1}-\zeta_{i,j}^{n+1})+ag_{i,j+1/2} \tag{1.21c}$$

$$\sum_{k=k_1}^{k_m} V_{i,j-1/2,k} = -g\frac{\Delta t}{\Delta x}\rho_0\left[\Xi A^{-1}R\right]_{i,j-1/2}(\zeta_{i,j}^{n+1}-\zeta_{i,j-1}^{n+1})+\left[\Xi A^{-1}G\right]_{i,j-1/2}$$
$$= -ar_{i,j-1/2}(\zeta_{i,j}^{n+1}-\zeta_{i,j-1}^{n+1})+ag_{i,j-1/2} \tag{1.21d}$$

The Eq. (1.21) form two tridiagonal system of equations (one in the x-direction and other in the y-direction) for each water column which can be uniquely solved in S1 (Figure 1.1) of SI3D with a double-sweep algorithm (here the Thomas algorithm is used to solve efficiently the tridiagonal systems). In executing the double-sweep algorithm, the results actually sought are only the two matrix products $\boldsymbol{A}^{-1}\boldsymbol{G}$ and $\boldsymbol{A}^{-1}\boldsymbol{R}$, which are column vectors equal in order to the number of model layers; the inverse of $\boldsymbol{A}$ (which is non-symetrical but diagonally dominant) is never computed by itself. Equation (1.21) is useful because it is an expression that can be formally substituted into the continuity equation.

With the new values of *ag* and *ar* calculated, the coefficientes *sx*, *sy*, *r* and *q* are calculated as follows at the end of S1 (Figure 1.1).

$$sx_{i+\frac{1}{2},j} = \frac{\Delta t}{\Delta x} ar_{i+\frac{1}{2},j} \tag{1.22a}$$

$$sx_{i-\frac{1}{2},j} = \frac{\Delta t}{\Delta x} ar_{i-\frac{1}{2},j} \tag{1.22b}$$

$$sy_{i,j+\frac{1}{2}} = \frac{\Delta t}{\Delta y} ar_{i,j+\frac{1}{2}} \tag{1.22c}$$

$$sy_{i,j-\frac{1}{2}} = \frac{\Delta t}{\Delta y} ar_{i,j-\frac{1}{2}} \tag{1.22d}$$

$$\begin{aligned} q_{i,j} = \zeta_{i,j}^{n-1} &- \frac{\Delta t}{\Delta x}\left(ag_{i+\frac{1}{2},j} - ag_{i-\frac{1}{2},j}\right) \\ &- \frac{\Delta t}{\Delta x}\left(ag_{i,j+\frac{1}{2}} - ag_{i,j-\frac{1}{2}}\right) \\ &- \frac{\Delta t}{\Delta x}\left[\sum_{k=1}^{km}(U_{i+1/2,j,k}^{n-1} - U_{i-1/2,j,k}^{n-1})\right] \\ &- \frac{\Delta t}{\Delta y}\left[\sum_{k=1}^{km}(V_{i,j+1/2,k}^{n-1} - V_{i,j-1/2,k}^{n-1})\right] \end{aligned} \tag{1.22e}$$

$$r_{i,j} = 1 + sx_{i-1/2,j} + sy_{i,j-1/2} + sy_{i,j+1/2} + sx_{i+1/2,j} \tag{1.22f}$$

Therefore, those summations are then included in Eq. (1.14) at the beginning of the stage 2 of SI3D (S2, Figure 1.1), resulting in the following equation for the free surface elevation at time n+1

$$-sx_{i-1/2,j}\zeta_{i-1,j}^{n+1} - sy_{i,j-1/2}\zeta_{i,j-1}^{n+1} + r_{i,j}\zeta_{i,j}^{n+1} - sy_{i,j+1/2}\zeta_{i,j+1}^{n+1} + sx_{i+1/2,j}\zeta_{i+1,j}^{n+1} = \mathrm{q}_{\mathrm{i,j}} \tag{1.23}$$

Equation (1.23) represents a pentadiagonal matrix which is symmetric and positive definite. This system of equations includes as many equations as water columns have the model and each equation includes up to five elements. These five elements relate each water column with the 4 neighbor columns (north, south east and west). Since free surface elevation is a 2D variable, each equation (and the system of equations in general) is created without paying attention to the number of layers of each water column. The system is then solved in S2 using a preconditioned non-symmetric conjugate gradient method (Golub and Loan 1996, Kincaid *et al.* 1989), which is an iterative method (hence approximate) that converges to the correct value within a tolerance.

After the free surface elevation at time n+1 is computed, the volumetric transport components are solved explicitly column by column of the grid using Eq. (1.21) in the stage 3 (S3, Figure 1.1) of SI3D. The vertical velocity is then updated column by column at the beginning of the stage 4 (S4, Figure 1.1) using the continuity equation (Eq. (1.1a)), which in finite difference form is

$$w_{i,j,k-1/2}^{n+1} = w_{i,j,k+1/2}^{n+1} - \frac{\left(U_{i+1/2,j,k}^{n+1} - U_{i-1/2,j,k}^{n+1}\right)}{\Delta x} - \frac{\left(V_{i,j+1/2,k}^{n+1} - V_{i,j-1/2,k}^{n+1}\right)}{\Delta y} \quad k = 2,3,...,km \qquad (1.24)$$

This equation is solved explicitly, starting from the bottom layer, where the vertical velocity is known to be zero.

The temperature transport equation is solved column by column after the hydrodynamic variables are determined in S4 of SI3D (Figure 1.1). Only the vertical diffusion terms are treated implicitly, in the numerical solution. Advection and horizontal diffusion terms are treated explicitly. The finite-difference form of Eq. (1.4), for a node with grid index i, j, k can be written as follows:

$$\mathrm{A_{i,j,k} s_{i,j,k-1}^{n+1} + B_{i,j,k} s_{i,j,k}^{n+1} + C_{i,j,k} s_{i,j,k+1}^{n+1} = D_{i,j,k}} \qquad (1.25)$$

where de coefficients A, B, C and D are:

$$A_{i,j,k} = -D_{v_{i,j,k-1/2}}^{n} / (h_{i,j,k-1}^{n} + h_{i,j,k}^{n})$$

$$B_{i,j,k} = h_{i,j,k}^{n+1}/(2\Delta t) + D_{v_{i,j,k+1/2}}^{n} / (h_{i,j,k}^{n} + h_{i,j,k+1}^{n}) + D_{v_{i,j,k-1/2}}^{n} / (h_{i,j,k-1}^{n} + h_{i,j,k}^{n})$$

$$C_{i,j,k} = -D_{v_{i,j,k+1/2}}^{n} / (h_{i,j,k}^{n} + h_{i,j,k+1}^{n})$$

$$D_{i,j,k} = F_{i,j,k}^{n} + \frac{\left(h_{i,j,k}^{n-1} + s_{i,j,k}^{n-1}\right)}{2\Delta t} - D_{v_{i,j,k+1/2}}^{n}\left(s_{i,j,k}^{n-1} - s_{i,j,k+1}^{n-1}\right)/\left(h_{i,j,k}^{n} + h_{i,j,k+1}^{n}\right) + D_{v_{i,j,k-1/2}}^{n}\left(s_{i,j,k-1}^{n-1} - s_{i,j,k}^{n-1}\right)/\left(h_{i,j,k-1}^{n} + h_{i,j,k}^{n}\right)$$

Here, the term *F* represent the explicit terms. Applying equation (1.25) to each of the layers at a computational point *i*Δ*x*, *j*Δ*y* results in a system of *km* equations involving *km* unknown values of $s_{i,j,k}^{n+1}$. This form a tridiagonal system of equations for each water column, which can be efficiently solved with the double sweep algorithm (Thomas algorithm). Once the new temperature fields are computed, they are used to update the density field using Eq. (1.5). The transport equation can be also used to simulate other passive tracers in the same way that temperature, solving an additional tridiagonal system of equations (per water column) for each passive tracer simulated.

Finally, the new scalar turbulent transfer coefficients (eddy viscosity and difussivity) are calculated column by column with the numerical algorithm of the turbulence equations (1.6)-(1.7) at the end of S4 (Figure 1.1). The approach adopted for the solution of the governing equations and the turbulent macroscale is a 3-level fully implicit algorithm (more details can be found in Rueda (2001)). The algorithm results in a tridiagonal system of equations for each water column that is solved using the double sweep algorithm (Thomas algorithm). The variables q^2 and q^{2}l are defined at vertical velocity points (interfaces between two cells aligned in the vertical direction). The discretized equations are written as

$$\begin{aligned}
&\frac{3}{2}\frac{(q^2)_{k-1/2}^{n+1} - (q^2)_{k-1/2}^{n}}{\Delta t} - \frac{1}{2}\frac{(q^2)_{k-1/2}^{n} - (q^2)_{k-1/2}^{n-1}}{\Delta t} \\
&- \frac{1}{\Delta z_{k+1/2}^{n+1}}\left[q_{k-1}^{n} l_{k-1}^{n} S_q \left(\frac{(q^2)_{k-3/2}^{n+1} - (q^2)_{k-1/2}^{n+1}}{\Delta z_{k-1}^{n+1}} \right) - q_k^n l_k^n S_q \left(\frac{(q^2)_{k-1/2}^{n+1} - (q^2)_{k+1/2}^{n+1}}{\Delta z_{k+1}^{n+1}} \right) \right] \\
&= 2A_{V\,k-1/2}^{\ n}\left[\left(\frac{u_{k-1}^{n} - u_k^n}{\Delta z_{k-1}^{n+1}} \right)^2 + \left(\frac{v_{k-1}^{n} - v_k^n}{\Delta z_{k-1}^{n+1}} \right)^2 \right] + 2\frac{g}{\rho_0} D_{V\,k-1/2}^{\ n} \left(\frac{\rho_{k-1}^{n} - \rho_k^n}{\Delta z_{k-1}^{n+1}} \right) - 2\frac{(q^2)_{k-1/2}^{n+1} q_{k-1/2}^{n}}{B_1 l_{k-1/2}^{n}}
\end{aligned} \qquad (1.26)$$

$$
\begin{aligned}
&\frac{3}{2}\frac{(q^2l)^{n+1}_{k-1/2}-(q^2l)^{n}_{k-1/2}}{\Delta t}-\frac{1}{2}\frac{(q^2l)^{n}_{k-1/2}-(q^2l)^{n-1}_{k-1/2}}{\Delta t}\\
&-\frac{1}{\Delta z^{n+1}_{k+1/2}}\left[q^n_{k-1}l^n_{k-1}S_q\left(\frac{(q^2l)^{n+1}_{k-3/2}-(q^2l)^{n+1}_{k-1/2}}{\Delta z^{n+1}_{k-1}}\right)-q^n_k l^n_k S_q\left(\frac{(q^2l)^{n+1}_{k-1/2}-(q^2l)^{n+1}_{k+1/2}}{\Delta z^{n+1}_{k+1}}\right)\right]\\
&=E_1 l^n_{k-1/2}A_{V\,k-1/2}^{\;n}\left[\left(\frac{u^n_{k-1}-u^n_k}{\Delta z^{n+1}_{k-1}}\right)^2+\left(\frac{v^n_{k-1}-v^n_k}{\Delta z^{n+1}_{k-1}}\right)^2\right]+E_1 l^n_{k-1/2}\frac{g}{\rho_0}D_{V\,k-1/2}^{\;n}\left(\frac{\rho^{\mathrm{n}}_{\mathrm{k-1}}-\rho^{\mathrm{n}}_{\mathrm{k}}}{\Delta z^{n+1}_{k-1}}\right) \qquad (1.27)\\
&-\frac{(q^2l)^{n+1}_{k-1/2}q^n_{k-1/2}}{B_1 l^n_{k-1/2}}\left[1+E_2\left(\frac{l^n_{k-1/2}}{\kappa \mathrm{z}^n_{\mathrm{k-1/2}}}\right)^2+E_3\left(\frac{l^n_{k-1/2}}{\kappa(\mathrm{H^n}-\mathrm{z}^n_{\mathrm{k-1/2}})}\right)^2\right]
\end{aligned}
$$

Once q^2 and q^2l at n+1 have been calculated, the eddy viscosity and diffusivity are calculated at n+1.

The finite difference equations for the semi-implicit trapezoidal scheme are nearly identical to those of the semi-implicit leapfrog scheme, except that the time interval over which the scheme is applied is halved to Δt from $2\Delta t$. The integration procedure is centered at time level $(n+ ½)\Delta t$ in the trapezoidal method and no longer involves the time level $(n-1)\Delta t$. The equations for the trapezoidal method can be easily derived from Eqs. (1.14)-(1.21) by making the changes in the subscripts indicated in Table 1.1 and replacing Δt by $\Delta t/2$. The variables needed in the scheme at time level $(n+ ½)\Delta t$ are determined by averaging the value of the variable at time level $n\Delta t$ and the estimate of the variable at time $(n+1)\Delta t$ obtained in a previous leapfrog or trapezoidal iteration.

Semi-implicit leapfrog scheme	*Semi-implicit trapezoidal scheme*
(n+1) Δt	(n+1) Δt
n Δt	(n+½) Δt
(n-1) Δt	n Δt

Table 1.1. Simple guidelines to derive the finite-difference equations for the semi-implicit trapezoidal scheme from the discretized form of the equations using the semi-implicit leapfrog method (Eq. (1.1)-(1.11) and (1.16)). The subscripts in the equations for the leapfrog method, on the left column, should be replaced by those shown in the right column to obtain the finite-difference equations for the semi-implicit trapezoidal method.

The hydrodynamic and transport model has been extensively validated against analytical solutions (Rueda and Schladow 2003) and field data sets collected in a wide range of lake environments (Rueda and Cowen 2005; Rueda et al. 2009), including

Lake Tahoe (Rueda et al. 2003). Furthermore, it has been used to analyze and study fundamental physical processes in lakes (Rueda and Schladow 2009).

1.2.3 Computational implementation

The computational implementation of Si3D is showed in detail in Figure 1.2. Si3D starts the execution reading the input data, initializing the model according to these input data and printing the model results in the initial state (in the time-step n=0). After this initialization, the real simulation of Si3D starts. Different variables control the flux execution, *Step* denotes if the next step will be a leapfrog (left side) or a trapezoidal step (right side), *Single* determine if the trapezoidal scheme will be used or not, *Ntrap* count the number of trapezoidal steps run in each time-step, the total number of trapezoidal steps is determined by *totaltrap*. Finally, the number of total time-steps is controlled by *niter* and when this variable is equal to *totaliter*, Si3D finishes. The main part of Si3D, where the numerical algorithm is executed, is indicated in the Figure 1.2 as Iterative Loop.

At the end of each step, the variables of the model must be updated with the new calculations. The type of data update will be different after a leapfrog step, a trapezoidal step or the last step before to start a new time-step. After a determined number of time-steps chosen by the user, Si3D save information as output data, which is stored using binary or text files. Finally, Si3D finishes when *niter* is equal to *totaliter*.

In computational terms, Si3D is a complex model that includes more than 20000 code lines, more than 100 subroutines and a huge quantity of variables (850). Among these variables. 250 save information for all the grid, either to store a specific value for each water column for 2D variables (as water free surface) or to store a specific value for each layer in a given water column for 3D variables (as vertical velocity).

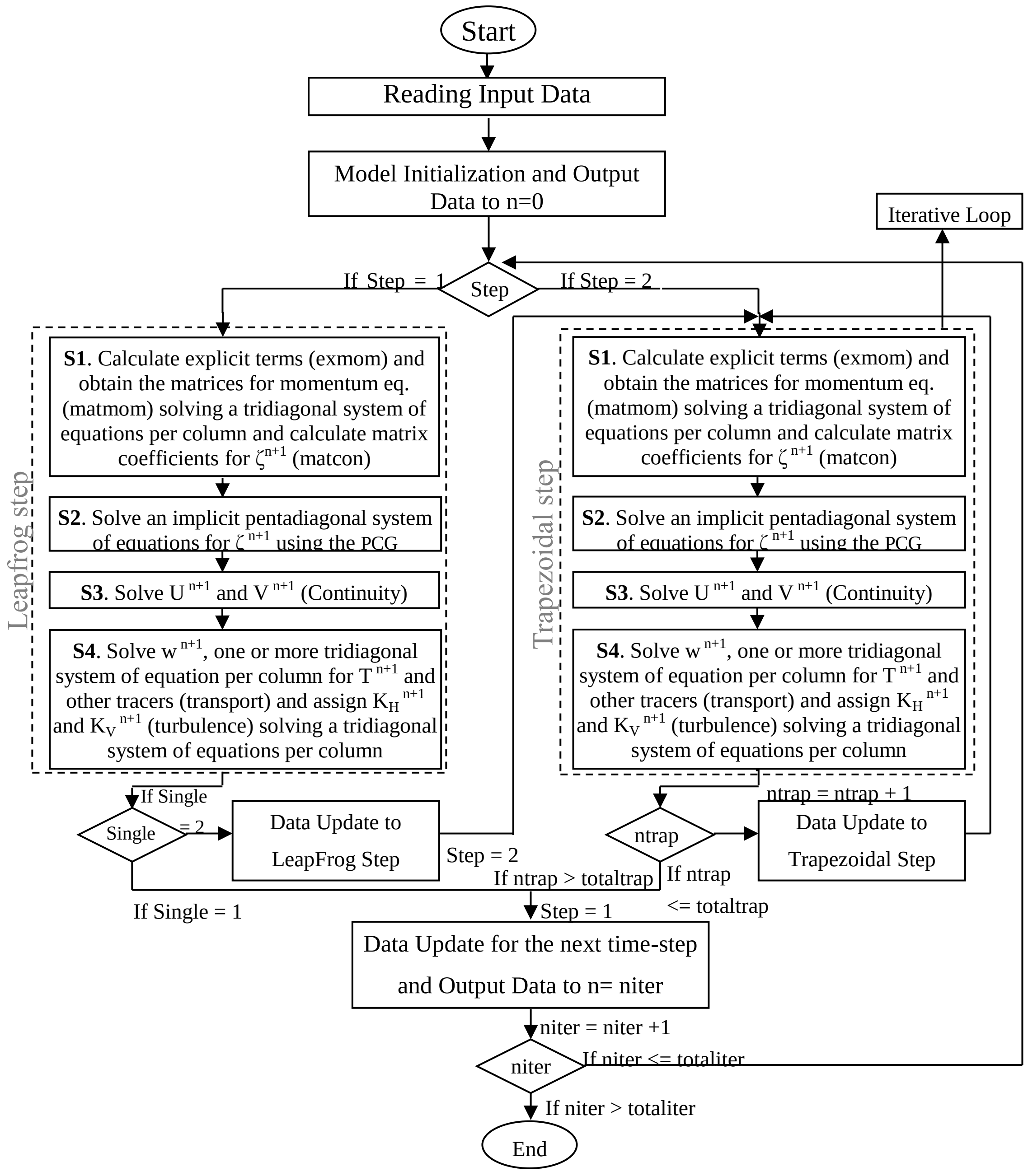

Start
Reading Input Data
Model Initialization and Output Data to n=0
Iterative Loop
If Step = 1
Step
If Step = 2
Leapfrog step
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ^{n+1} (matcon)
S2. Solve an implicit pentadiagonal system of equations for ζ^{n+1} using the PCG
S3. Solve U^{n+1} and V^{n+1} (Continuity)
S4. Solve w^{n+1}, one or more tridiagonal system of equation per column for T^{n+1} and other tracers (transport) and assign K_H^{n+1} and K_V^{n+1} (turbulence) solving a tridiagonal system of equations per column
Trapezoidal step
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ^{n+1} (matcon)
S2. Solve an implicit pentadiagonal system of equations for ζ^{n+1} using the PCG
S3. Solve U^{n+1} and V^{n+1} (Continuity)
S4. Solve w^{n+1}, one or more tridiagonal system of equation per column for T^{n+1} and other tracers (transport) and assign K_H^{n+1} and K_V^{n+1} (turbulence) solving a tridiagonal system of equations per column
If Single = 2
Single
Data Update to LeapFrog Step
Step = 2
ntrap = ntrap + 1
ntrap
Data Update to Trapezoidal Step
If ntrap > totaltrap
If ntrap <= totaltrap
If Single = 1
Step = 1
Data Update for the next time-step and Output Data to n= niter
niter = niter +1
niter
If niter <= totaliter
If niter > totaliter
End

Figure 1.2. Flow diagram for the Si3D model. S1, S2, S3 and S4 are the solver stages of Si3D.

1.3 Conclusions

The hydrodynamic model used as test example to apply the optimization strategies proposed in this work is presented. The model, Si3D, is a finite-difference model based in the 3D-SWE proposed by Smith (2006) and extended for lakes by Rueda (2001). Si3D uses a semi-implicit, leapfrog-trapezoidal numerical scheme that is efficient and essentially second-order accurate in the spatial and temporal numerical approximations.

The numerical scheme is based on treating the gravity-wave and vertical diffusion terms implicitly and all other terms in the governing equations (including advection) explicitly. The model does not use any form of vertical/horizontal mode-splitting to incorporate implicit vertical diffusion into the semi-implicit scheme. In this manner, the method is guaranteed to be stable without the time-step limitation imposed by the gravity wave CFL condition, stricter in explicit or mode-splitting methods. The governing equations for the multilevel scheme are arranged in conservation form by integrating them over the height of each horizontal layer. The layer-integrated volumetric transports replace velocities as the dependent variables so that the depth-integrated continuity equation used in the solution for the water surface elevation is linear. The advantage of the semi-implicit approach is that the solution for the water surface elevation is uncoupled in the model from the solution for volumetric transport. A pentadiagonal system of linear equations is solved at each time-step for the water surface elevation using an efficient preconditioned conjugate-gradient method. Volumetric transports are computed explicitly from the momentum equations.

Instead of the improvements founded in a semi-implicit approach, the computational cost, in terms of required memory and execution time, is still expensive. Si3D saves information of the entire grid in a huge quantity of 2D and 3D variables, in three different time states. Moreover, the type of operations presented in the semi-implicit approach increases the computational complexity and execution time of the model. Si3D must solve at each time-step a large number of small tridiagonal equation systems (1 in S1 and 2 or more (if passive tracers are used) in S4) using a double-sweep method known as Thomas algorithm. Besides, a single, large, pentadiagonal equation system must be also solved in S2 for the water surface elevation, using an iterative method known as Preconditioned Conjugate Gradient.

Chapter 2

Evaluation of a nested-grid implementation for 3D finite-difference semi-implicit hydrodynamic models

Abstract

This chapter evaluates the implementation of a nested Cartesian grid in a 3D semi-implicit hydrodynamic model with synthetic and real examples. The outer model provides all the values needed by the governing equations of the nesting (inner) subdomain at the boundary (including tangential velocities). A 3D flux relaxation scheme is applied to prevent mass and energy drift. The influence of tangential velocities in the solution is evaluated, showing a substantial reduction on the results' quality when they are considered negligible and lateral circulation exists. The inner/outer coupling implemented achieves a simulation time equal to the inner execution time and allow a transfer step equal to the inner time-step, removing time interpolation errors. This coupling makes feasible the 3D relaxation implemented. A dramatic improvement in memory requirements and simulation time is achieved, that allows the use of low-cost low-power consumption platforms in the simulations.

2.1 Introduction

In the last decade, considerable progress has been made in the development of three-dimensional (3D) transport and mixing models of shallow-water capable of resolving with reasonable accuracy and computational cost *large-scale* physical processes in rivers, lakes and reservoirs (Hinterberger et al. 2007, Huang et al. 2010, Leon et al. 2012, Schwab et al. 2009 Smith 2006). These models can also potentially be used to simulate *local-scale* processes, such as near-shore processes, where detailed topography and large changes in vorticity produce changes over small spatial scales. But the high resolution grid needed to simulate the local-scale processes requires disproportionately large amounts of CPU time and memory. Long simulation times are unacceptable when these models are part of decision support systems, where multiple simulations need to be run, or when the duration of a simulation exceeds the time period within which a result is required.

Unstructured or structured grid models can be used to conduct simulations of large basin-scale and local-scale processes. Unstructured models use grid cells of varying shapes and sizes that are pieced together to better represent topographic and bathymetric complexity. Small cell size can also be utilized where the local-scale processes are to be represented, increasing cell size away from the zone of interest. These models, however, are more computationally expensive than structured grid models (Griffies et al. 2000). The algorithmic details are more complex than with structured grids, and the accuracy of results is extremely sensitive to the quality of the generated mesh. Important related physical processes (such as suspended sediment transport) cannot be represented in an immediate and simple way as can be done with structured grids (Rueda and Schladow 2003).

Many studies in shallow water modeling have used finite-difference, structured Cartesian grid models with very satisfactory results (Huang et al. 2010, Leon et al. 2012, Rueda and Schladow 2003, Zavatarelli and Pinardi 2003). Grid generation tasks and the implementation of the numerical solution are simpler when compared with unstructured approaches. Local-scale processes can be simulated using a high resolution grid covering the entire domain. However, this comes at a very high-computational cost. Alternatively, nested-grid models can be used to reduce computational cost in oceans (Giunta et al. 2007, Rueda and Schladow 2003, Zavatarelli and Pinardi 2003), lakes (Leon et al. 2012) and rivers (Kolerski et al. 2010). In nested grid models, a higher resolution model (the inner model), is used to simulate the local-scale processes in a target zone. This inner model is embedded within a coarser resolution model (the outer model) that simulates the basin-scale processes and provides external boundary data to the inner model. Using this approach, the high-resolution inner model can be used to resolve small-scale circulation and mixing processes in the region of primary interest (target zone) while significantly reducing the overall computer time of simulations compared to the traditional approach of using only one uniform grid size.

A standard Cartesian-grid model typically uses a single, high-resolution grid of a fixed cell-size over the entire model domain. One important shortcoming of this approach is that these models when applied to medium-sized domains or larger can often require enormous amounts of computer memory. Memory requirements for these models can be especially high if all "dry" cells (cells located on dry land outside the boundaries of the water body being simulated) are saved in the active memory during simulations. A rectangular Cartesian grid overlaid upon an irregular-shaped water body, such as a meandering river, may have a high percentage of dry cells that are stored in memory. Curvilinear-grid models (see, for example Herzfeld (2009)) have been used to reduce memory requirements related to dry-cell storage, but these models can introduce other computational problems related to the use of a curvilinear coordinate system and also lack the desired simplicity of Cartesian grids. The model implemented here keeps memory requirements manageable by using a nested grid and storing only "wet" cells in active memory.

The first nested-grid one-way implementation for the Si3D model (Smith 2006), N-Si3D, is here presented and, his quality and computational efficiency, evaluated. Si3D is a semi-implicit second-order accurate, finite-difference Cartesian grid model for 3D shallow-water. Three kinds of tests applied to three kinds of example models verify and validate the implementation and demonstrate the excellence of the results. The examples are:

- Synthetic rectangular basins with constant wind. These examples allow to verify that the hydrodynamic fields in the inner and outer models are the same and to demonstrate that the implementation is good at conserving mass and energy.
- Sacramento River (USA). Scientists use this model to understand the influence of tidal river dynamics on the migration of juvenile salmon towards the ocean and to reproduce the lateral and secondary circulations in the area of channel meanders. The results here presented show that the nesting implementation proposed can reproduce lateral circulations not attainable by the lower resolution model of the entire basin.
- Lake Tahoe (USA). Scientists use this model to study the transport of contaminants and planktonic larvae in the near-shore (littoral) zone. The results show that the nesting implementation described can reproduce recirculation not observed in the lower resolution model of the entire basin.

The high computational cost required to apply the model using high resolution grids on large domains can lead to model run times that are slower than real time. This prevents the model from being used for forecasting the migration of salmon in the Sacramento River or the impact of contaminants and invasive species in Lake Tahoe for periods of days or more ahead of time, and/or prevents the use of low-cost and low-power-consumption processors or computers. This happens due to the huge number of operations and the large storage memory required by 3D high resolution grid, semi-implicit, second-order models, even when dry cells are not stored in memory. We have developed a hybrid shared and distributed memory parallel implementation with very good results for middle resolutions in computer clusters with multicore low-price/low-consumption servers, but with poor results for the high resolution needed in local-scale process simulations (Acosta et al. 2010). In order to deal with higher resolution models even in PCs or volume servers (of IDC Taxonomy, www.idc.com) a nesting implementation for Si3D is here proposed. Energy efficiency is increasingly important in computation, because the increased usage of computation, together with increasing energy costs and the need to reduce greenhouse gas emissions call for energy-efficiency. A basic way to contribute to energy-efficiency is using low carbon footprint computers. Expensive and high-power consumption computers can be used for even much larger problems.

The chapter is organized as follows. Section 2.2 presents the characteristics of the nesting implementations here evaluated and places this work within the related bibliography. Section 2.3 details the nesting approach implementation. Section 2.4 validates the nested implementation with three kinds of tests using both synthetic and real examples, and compares it with other approaches. Results demonstrate that the implementation overcomes mass and energy drift problems and it is able to reproduce recirculation in Lake Tahoe and lateral circulation in Sacramento River not attainable by lower resolution model of the entire basin. Section 2.4 also shows the importance of communicating tangential velocities in nested-grid simulations. Results about the computational efficiency are also given in this section. Finally, the last section gives conclusions.

2.2 Nesting implementation: characteristics, performance. Related works

A nesting implementation comprises several design and implementation decisions that affect the quality of simulation results and the computational efficiency:

Open Boundary Condition (OBC) type.

The grid border cells of a hydrodynamic model do not have adjacent cells in all its sides (there are no cells outside the grid), i.e. the grid has open boundaries. These border cells could resolve the governing equations if fictitious boundary cells were added to the model. The equations used at these boundary cells depend on the open boundary condition (OBC) used (Blayo and Debreu 2005, Chapman 1985, Marchesiello et al. 2001). The main purpose of OBCs is allowing the fluxes generated inside the model domain to leave it without reflection, avoiding the consequent deterioration of the model solution; although, at the same time, OBC should allow the model to receive external inward fluxes (e.g. inner models of nesting schemes). If the model does not receive inward fluxes the boundary is passive, otherwise it is active (Palma and Matano 2001). In pure OBCs, external values are unknown and set to 0 (the value itself, its change over time, its change over space or its change over time and space) (Chapman 1985, Jensen 1998, Martinsen and Engedahl 1987). External data used by OBCs can be sensor/climatology data, or data obtained from a larger-domain outer model (used by inner models in nesting approaches).

Several classifications (Blayo and Debreu 2005, Marchesiello et al. 2001, Oddo and Pinardi 2008) and comparisons (passive boundaries (Chapman 1985, Jensen 1998, Lavelle and Thacker 2008, Nycander and Doos 2003, Palma and Matano 1998), active boundaries (Cailleau et al. 2008, Herzfeld and Andrewartha 2012, Jensen 1998, Lavelle and Thacker 2008), active nesting boundaries (Cailleau et al. 2008, Herzfeld and Andrewartha 2012)) of OBCs can be found in the bibliography. Usual OBCs are radiation condition, Flather condition (often classified as radiation condition), Dirichlet or clamped condition (it can be considered a special case of radiation (Jensen 1998) or the most sharp form of relaxation (Blayo and Debreu 2006)), advective condition, and relaxation methods. A passive *clamped* condition sets to 0 the boundary variables ϕ ($\phi=0$) (Chapman 1985) or their change over time $\delta\phi/\delta t=0$ (Jensen 1998), active clamped conditions set the boundary variables to external values that change over time, $\phi=\phi^{ext}$ (Blayo and Debreu 2005). *Radiation conditions* assume a free outgoing wave propagation, often normal to the boundary, e.g. the passive OBC Sommerfeld condition $\delta\phi/\delta t + c\ \delta\phi/\delta n=0$, which corresponds to the transport of ϕ through the boundary with a velocity of c, n is normal to the boundary. The advective condition is frequently used for tracers (temperature, salinity),for instance in (Herzfeld and Andrewartha 2012, Herzfeld 2009, Oddo and Pinardi 2008, Zavatarelli and Pinardi 2003), $\delta\phi/\delta t + Vn\ \delta\phi/\delta n=0$, Vn is the normal velocity in the border cell. A *relaxation scheme*, in order to absorb outgoing flow generated in the model, relaxes the solution of the model toward 0 or to values calculated from interior values (passive) (Jensen 1998, Martinsen and Engedahl 1987, Palma and Matano 1998), or relaxes it to external data ϕ^{ext} (active).

Typically, the relaxation is applied in a relaxation band, e.g. the flux relaxation scheme, $\phi=\alpha\times\phi^{target}+(1-\ \alpha)\times\phi^{in}$, where ϕ^{in} is the interior solution, ϕ^{target} the specified target value, and α is the relaxation parameter which varies from 0 to 1 within the relaxation band; in active implementations $\phi^{target}=\phi^{ext}$. An important drawback of the flux relaxation scheme is its cost, in terms of communication time or storage required, which may prevent its use, especially in 3D models (Mason et al. 2010). Relaxation schemes can be applied to other OBCs; for instance, external values could be added in radiation OBCs by including a relaxation term (nudging term), $\delta\phi/\delta t + c\ \delta\phi/\delta n=-(\phi-\phi^{ext})/\tau$ (Marchesiello 2001).

Boundaries with clamped condition reflect any outgoing flow if they are passive and any outgoing flow not described by the external data if they are active. Comparisons show very poor performance of passive clamped conditions (Chapman 1985, Jensen 1998). Relaxation methods are recommended in comparative studies (Jensen 1998, Lavelle and Thacker 2008, Nycander and Noos 2003, Palma and Matano 2001). Most studies compare passive 1D or 2D OBCs. Jensen (1998) compares 3D passive clamped, radiation and relaxation schemes, recommending the relaxation scheme for general use but pointing out that reasonably correct target values, ϕ^{target}. are required. Lawrence and Ashley (2008) compares three different relaxation alternatives with shallow-water models recommending the simple implementation used in Martinsen and Engedahl (1987) but with correct external inputs as target value. In nesting approaches, the external information for the inner model, provided by the outer model, is mostly compatible with the inner model solution, especially if both resolve the same governing equations. Considering all these studies, N-Si3D uses a 3D active clamped condition with a flux relaxation scheme (cost is avoided by the particular coupling implementation, see below) to absorb outgoing flow generated in the inner model. Relaxation can be used if it is beneficial to prevent energy build-up, e.g. in long-term simulations and/or when it is difficult to find inner/outer boundaries with a good compatibility. The N-Si3D implementation without relaxation, just clamped condition, obtains good results. Comparison made here between clamped condition with or without relaxation reveals the goodness of the active 3D relaxation implemented (Section 2.4.2). Other alternatives of relaxation have been proposed such as the local flux adjustment in Herzfeld (2009), which is algebraically similar to the Flather radiation condition Herzfeld and Andrewartha (2012).

Numerical implementation of the inner/outer interface and coupling of the inner/outer grids

The N-Si3D OBC implementation guarantees that all the values used by inner cells are obtained by the same governing equations with the same discretization method and numerical implementation, even boundary values. The outer model provides all the values needed by the numerical implementation not computed by the inner model (the border cells use, for some variables, values (temporal and spatially) interpolated from outer values), so they are also obtained using the same governing equations, discretization and numerical implementation though with parameters adjusted for a lower resolution grid. Outer model provides all the information required by the inner model (flows) and without duplications, so under-specification and over-specification problems are reduced, although they are not removed. Incoming flows to and outgoing flows from the inner model boundaries depend on these outer values transferred. The quality of the solution will depend on distortions introduced at the boundary by the outer information transferred and on the noise caused by the reflection at the boundary of the outgoing flows generated in the higher resolution inner model and not present in the outer information. Distortions in the incoming fluxes lead to mass, momentum and temperature (tracers) conservation errors. Temporal and spatial interpolation of data transferred and mismatches between the outer and inner bathymetries can cause flux distortions and noise. The work (Cailleau 2008) compares results using temporal interpolation with two different transfer rates between inner and outer models, showing a clear improvement when transfer rate is reduced down to the outer-grid time-step. N-Si3D transfers data at each inner-grid time-step, removing temporal interpolation. This does not affect computational efficiency (execution time perceived by the user, resource cost) due to the parallel outer/inner implementation (Section 2.3.4). The inner OBC uses a bathymetry transition band in order to minimize the bathymetry mismatch effect on the quality (Section 2.3.2). Other works also used a bathymetry transition band (Herzfeld and Andrewartha 2012, Mason et al. 2010, Penven et al. 2006).

The work (Herzfeld 2009) is devoted to demonstrate that the relative positions of the variables supplied by OBCs may influence model solution; in particular two different relative positions is tested using different OBC types (passive radiation for elevation and tangential velocity and passive no-gradient, passive Flather and active Flather, for normal velocity). Here two different relative positions for Si3D hydrodynamic model are compared with the active OBC based on clamped and flux relaxation scheme, showing better results for the relative position used in N-Si3D (Section 2.3.3).

Offline nesting versus online nesting

The outer model may be run first independently of the inner model (*offline nesting* (Cailleau et al. 2008, Herzfeld and Andrewartha 2012, Mason et al. 2010)) or synchronously with the inner model (*online nesting* (Cailleau et al. 2008, Debreu and Blayo 2008, Penven et al. 2006)). In offline nesting the information to be transferred from the outer to the inner model is stored in a file, so a compromise must be met among storage, and number of values to exchange and transfer frequency. Here, an online implementation is presented that allows transferring at each simulation time-step of the inner model (so no temporal interpolation errors are added) and transferring all values required for the inner discrete equations, even those of a 3D model (to decrease under-specification). Results show no differences between inner and outer model outputs (water surface elevation, velocities, temperature, volume) when both inner and outer have the same resolution (Section 2.4.2).

One-way nesting versus two-way nesting

One-way nesting (Bonaventura and Rosatti 2002, Kolerski et al. 2010, Leon et al. 2012, Mason et al. 2010, Zavatarelli and Pinardi 2003) or *two-way nesting* (Cailleau et al. 2008, Debreu and Blayo 2008, Fox and Maskell 1995, Harris and Durran 2010, Zhang et al. 2007) have been used to exchange information between the components of an online nested-grid model. In conventional one-way nesting, the fine-resolution inner model is forced by the solution of the outer model (just boundary communication is needed). Alternatively, two-way interaction may be used between the inner and outer models. This approach adds feedback from all the inner model cells to the outer model, with the intention that the outer model benefits from the increased accuracy of the solution yielded by the inner model. Two-way nesting does not always improve the results (Heggelund and Beentsen 2002) and the interaction requires more execution time (computation and communication time). Two-way nesting approaches must have an online implementation, the execution time is inner plus outer model execution time plus communication time, and, as it is here discussed (Section 2.3.4), this does not change in *parallel* inner/outer implementations of two-way nested semi-implicit models because of the implicit variables. N-Si3D implementation execution time depends on the inner execution time or the outer execution time (Section 2.3.4).

Parallel Inner/Outer dynamic coupling

Online nesting implementations require coupling the inner and outer models at execution time. This is not an easy task as it is pointed out in (Blayo and Debreu 2006, Herzfeld and Andrewartha 2012, Mason et al. 2010). Other works use external software for coupling, which also implement the bathymetry transition band, (Cailleau et al. 2008, Debreu et al. 2012, Penven et al. 2006). Here a coupling implementation that allows the execution of the inner and outer models in parallel is proposed (Sections 2.3.4, 2.4.6). The parallel implementation is more challenging in semi-implicit hydrodynamic models because the inner model at the current time-step requires values of outer variables at a later outer time-step. The parallel scheme here presented can be used with other hydrodynamic models. The parallel execution time achieved is equal to the inner-model execution time or the outer-model execution time. Only one outer/inner transfer of values per time-step is needed and its communication time does not affect execution time because this transfer is overlapped with computation. Although not used here, both, inner and outer models, can be parallelized using a domain (data) decomposition structure combined with a master-slave structure as in Acosta et al. (2010) (there are several attempts to classify task/process structures in parallel computing, one of these classification can be found in Silva-Moura and Buyya (1998)).

External data provided from the outer model

The exchange of information from outer to inner model (carried out in both one-way and two-way nesting) usually includes velocities, active scalar concentration (e.g. temperature, salinity or suspended sediment concentration), non-active scalar transport (e.g. chlorophyll concentration, tracer concentration) and water surface elevation. Many models (Bonaventura and Rossatti 2002, Fox and Maskell 1995, Harris and Durran 2010, Zhai et al. 2008) transfer normal velocity components to ensure that mass and momentum-diffusion fluxes through the nested boundary are consistent, but they do not transfer *tangential velocity* components. While some models (Barth et al. 2005, Zavatarelli and Pinardi 2003) do transfer tangential velocities, they have not reported its influence on the accuracy of results or the increase in computational time required by the model. Here it is shown that the passing of tangential velocities in the boundary conditions does not materially affect execution time and can significantly improve the accuracy of the simulations, especially, when water currents and lateral circulations are strong and/or features such as vortices exist (Section 2.4.5).

2.3 Nesting specifications and implementation

Figure 2.1 illustrates a schematic set up of a nested grid for the problem of simulating circulation and transport in a square basin. The nesting implementation allows to define the nesting boundary as any path of segments in the *x* and *y* (horizontal) directions, selecting a connected inner domain inside the outer domain, being thus able to achieve a better adjustment to the region of interest and ignore areas of the domain in which high resolution is not required. The Figure 2.1 example uses a high-resolution grid to resolve the local-scale physical processes in the sub-domain formed by the northeastern corner of the outer model; those processes are partly driven by the large-scale basin circulation. The large basin is discretized using a structured grid with square cells of horizontal size Δx_{og}. The boundary that separates the sub-basin from the boundary cells will be referred to as the I/O (Inner/Outer) boundary. The sub-basin in the inner model is discretized with cells of size $\Delta x_{ig} \leq \Delta x_{og}$. Therefore, the number of water columns of the inner model, N_{im}, will be larger than the number of columns of the outer model, N_{om}, defined within the sub-basin where $N_{im} = (\Delta x_{og}/\Delta x_{ig})^2\ N_{om}$. The ratio $\Delta x_{og}/\Delta x_{ig}$ is the grid refinement parameter r_g which must be an integer value. The height of each layer in the vertical direction is equal in the outer and inner models (e.g., the *i*-layer height in the outer model is equal to the *i*-layer height of the inner), and the maximum number of layers remains equal in the inner model.

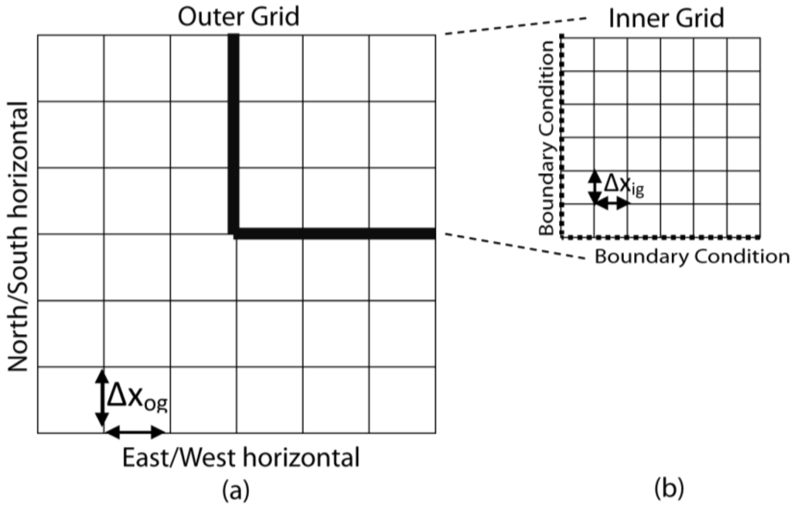


Figure 2.1. Nesting grid example, **(a)** outer grid model or basin where Δx_{og} is East-West and North-South horizontal resolution and **(b)** inner grid model or sub-basin inside the outer grid where horizontal resolution Δx_{ig} is half the outer model horizontal resolution. The sub-basin is simulated by the inner model with resolution Δx_{ig} and within the outer model with resolution Δx_{og}.

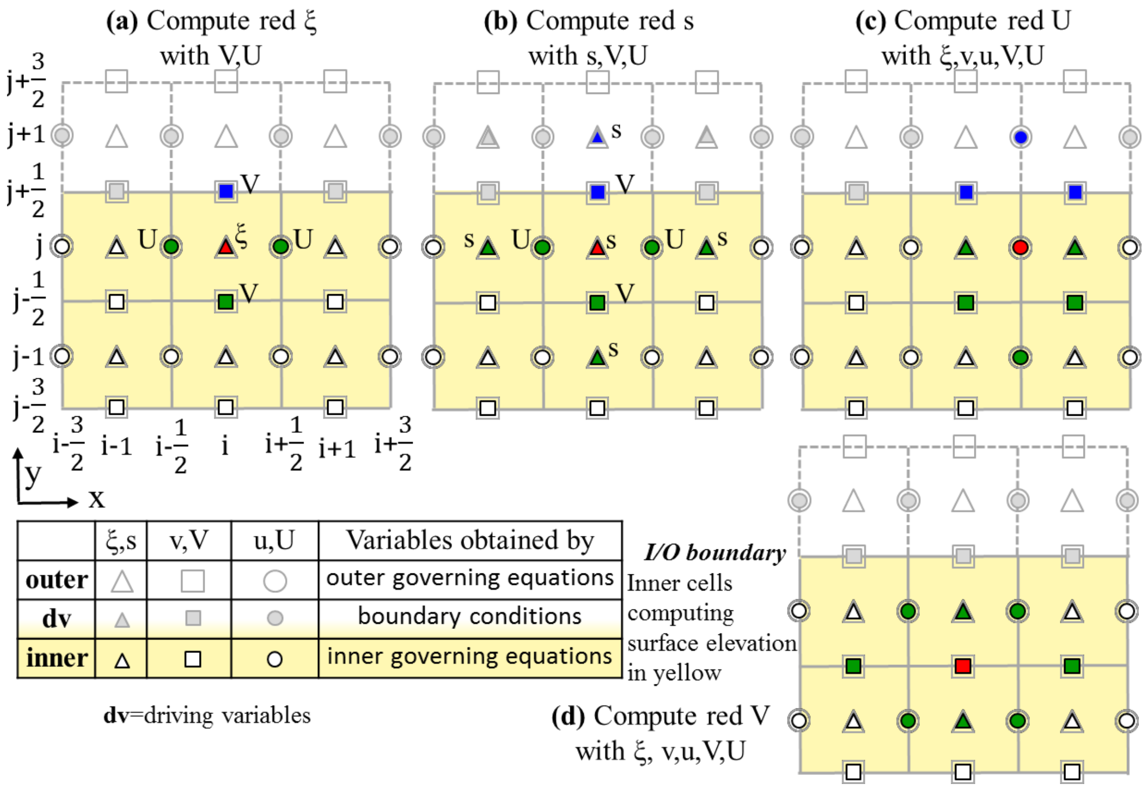
The nested implementation presented is applied here to the semi-implicit hydrodynamic model proposed by Smith 2006, Si3D (more details in Chapter 1) and modified to take advantage of several basic optimizations (more details about the basic optimizations can be found in Chapter 5, this version is called Basic Si3D). This optimizations also include the improvement of data structure of Si3D, explaining in detail in Section 2.3.5.

2.3.1 Discrete form of the equations near I/O boundary in the inner model

All inner model cells in the 3D grid resolve the same governing equations with the same numerical implementation. The outer model provides all the values needed by the numerical implementation not computed by the inner model. The border cells use, for some variables, values interpolated from outer values; these interpolated variables are called here *driving variables* and conform the boundary variables of the inner model. Consider, for example, the simple sub-domain with a northern I/O boundary shown in Figure 2.2(a). The value of ζ at the next time-step in any border column (*i,j*) depends on the unknown volumetric transports (*U, V*) at their 4 neighboring columns. The discrete form of the continuity equation in one such border column is:

$$\begin{aligned}&\zeta^{n+1}_{i,j} + \frac{\Delta t}{\Delta x}\left|\sum_{k=1}^{km}(U^{n+1}_{i+1/2,j,k} - U^{n+1}_{i-1/2,j,k})\right| - \frac{\Delta t}{\Delta y}\left|\sum_{k=1}^{km}(V^{n+1}_{i,j-1/2,k})\right| = \\ &\zeta^{n-1}_{i,j} - \frac{\Delta t}{\Delta x}\left[\sum_{k=1}^{km}(U^{n-1}_{i+1/2,j,k} - U^{n-1}_{i-1/2,j,k})\right] - \frac{\Delta t}{\Delta y}\left[\sum_{k=1}^{km}(\underbrace{V^{n+1}_{i,j+1/2,k} + V^{n-1}_{i,j+1/2,k}}_{driving\ .}) - \sum_{k=1}^{km} V^{n-1}_{i,j-1/2,k}\right.\end{aligned} \quad (2.1)$$

k = 1 and k_m denote the surface and bottom layers in the water column and *n+1* and *n-1* represent different time intervals. The two terms underlined in the right hand side of Eq. (2.1) (driving variables) are values of depth-integrated transport across the northern face, at times *n+1* and *n-1,* which are computed by the outer model and provided to the inner model (driving variables are highlighted in blue in the figure). One can easily derive the equations for other types of boundaries (eastern, western or southern). Note that in an implicit model, or a semi-implicit one as the model used here, in order to compute water surface elevation for next time-step *n+1* in the inner model, ζ^{n+1}, it is necessary to communicate the driving volumetric transport values (V^{n+1}, or U^{n+1} for an east-west boundary), computed in outer model at the end of time-step *n*, to the inner model at the beginning of time-step *n*. Therefore, the outer model proceeds one time-step ahead the inner model. The rest of equations are not shown here, they can be obtained in a similar way. Figure 2.2 also shows the dependencies for $s_{i,j,k}$ (b), $U_{i+1/2,j,k}$ (c), and $V_{i,j-1/2,k}$ (d).


(a) Compute red ξ with V,U
(b) Compute red s with s,V,U
(c) Compute red U with ξ,v,u,V,U
(d) Compute red V with ξ, v,u,V,U
j+3/2
j+1
j+1/2
j
j-1/2
j-1
j-3/2
i-3/2
i-1
i-1/2
i
i+1/2
i+1
i+3/2
y
x
U
V
ξ
s
I/O boundary
Inner cells computing surface elevation in yellow
dv=driving variables

	ξ,s	v,V	u,U	Variables obtained by
outer	△	□	○	outer governing equations
dv	▲	■	●	boundary conditions
inner	△	□	○	inner governing equations

Figure 2.2. For a northern I/O boundary ($r_g = 1$), dependencies for the calculation of (a) $\zeta_{i,j}$ (red triangle), (b) $s_{i,j,k}$ (red triangle), (c) $U_{i+1/2,j,k}$ (red circle), and (d) $V_{i,j-1/2,k}$ (red square). The values used in the computation are in green (if they are obtained by the inner model) and blue (if they are obtained by the boundary condition, driving variables)

2.3.2 Boundary interface between inner and outer model

The additional details in the high-resolution bathymetry will be important in creating flow features, which can be resolved by the inner model. Although, the inner and outer bathymetries need to match at the interface to reduce the mass flow differences between inner and outer models. Hence, the bathymetry information in the inner model will have a coarser resolution (equal to the outer model) at the I/O boundary, and will gradually transition to the high resolution bathymetry away from it. Preliminary tests showed that a *transition band* of up to three outer grid cells $tb_{om} = 3$, corresponding to $tb_{im} = tb_{om} \times r_g = 3 \times r_g$ inner grid cells, obtained good results. The new depth in each water column of the transition band is linearly interpolated between outer and inner grid depths.

The relaxation area implemented in the interface (Section 2.2), smoothing the flux changes between inner and outer models in order to maintain a strong consistency between their solutions in the area where both interact, preventing flux reflection. The inner solution on the interface, i.e. scalar concentration, height, normal and tangential velocities (volumetric transports are obtained from velocities and heights), is replaced at each time-step by (according to Davies (1976)):

$$\phi = (1-\alpha) \times \phi^{in} + \alpha \times \phi^{out} \qquad (2.2)$$

$$\alpha = 1 - \tanh(d/2) \qquad d = 1,2,3,\ldots\ tb_{im} \qquad (2.3)$$

where ϕ^{in} and ϕ^{out} are the inner and outer solution respectively, α is a relaxation function, and tb_{im} the size of the interface (equal by default to the transition band size).

2.3.3 *Relative position of the driving variables on the grid*

The relative position on the cell faces and centers of the velocity and surface elevation supplied by OBC, or OBC implementation (Herzfeld 2009), affects the quality of the results, as demonstrated in the cited reference. Herzfeld (2009) compares two alternatives of OBC implementations using different OBC types (passive radiation for elevation and tangential velocity and passive no-gradient, passive Flather and active Flather, for normal velocity). Other factors affect the OBC quality, as it is pointed out in Herzfeld (2009), such as the OBC type, the hydrodynamic model and the application.

Two alternatives for relative positions have been studied and tested here for Si3D and the type of OBC here used. Figure 2.3 shows the final OBC implementation included in N-Si3D (a) and the alternative approach studied (b). They have different relative positions of normal and tangential driving variables in the boundary grid (these implementations do not require external surface elevation per se). Readers can find similarities between the implementation in Figure 2.3(b) and the implementation tested in Herzfeld and Andrewrtha (2012). Although both use clamped OBC and compute the border surface elevations via the governing equations, here they are computed by the same equations used in the rest of the inner domain (it is a border cell of the inner domain). The OBC in Herzfeld and Andrewrtha (2012) modifies the equation so that the local flux adjustment in Herzfeld (2009) is applied. The local flux adjustment condition can be shown to be algebraically similar to the Flather radiation condition as pointed out in Herzfeld and Andrewrtha (2012). Moreover, the implementation in Herzfeld and Andrewrtha (2012) requires external surface elevations from the outer model to be used in the governing equations of these cells. They are boundary cells instead of border cells of the inner model.

Output quality is not the same for both approaches in Figure 2.3, as shown in Section 2.4 results for both synthetic and real examples. The implementations here presented do not introduce any errors in the computation when inner and outer models have the same resolution (as Section 2.4 shows). Therefore, the distortion in the incoming and outgoing fluxes is due to the interpolation required to obtain driving variables from outer values and to bathymetry interpolation. Thus, errors may depend on how many values of driving variables are used by the inner model. Differences are more noticeable when tangential velocities play an important role, such as in Sacramento River simulations. Thus, which terms in the equations use the driving variables is also important. The OBC implementation determines the usage of driving variables in the inner equations: number of times they are used, which terms in the equations use them (Herzfeld 2009). Herzfeld and Andrewrtha (2009) shows, for N-Si3D and the alternative OBC implementation of Figure 2.3(b), which terms in the governing equations use which driving variables (normal velocity or volumetric transport (N), tangential velocity or volumetric transport (T) and temperature (tp)). N-Si3D OBC implementation uses driving variables 14 times in the computation of the inner border variables and the alternative, Figure 2.3(b), 20 times (last row in Table 2.1). Border normal velocities (principal responsible of the incoming and outgoing fluxes) depend on 2 driving variables (1 normal and 1 tangential, column 4) for Figure 2.3(a) and on 8 (2 normal, 6 tangential, column 7) for Figure 2.3(b).

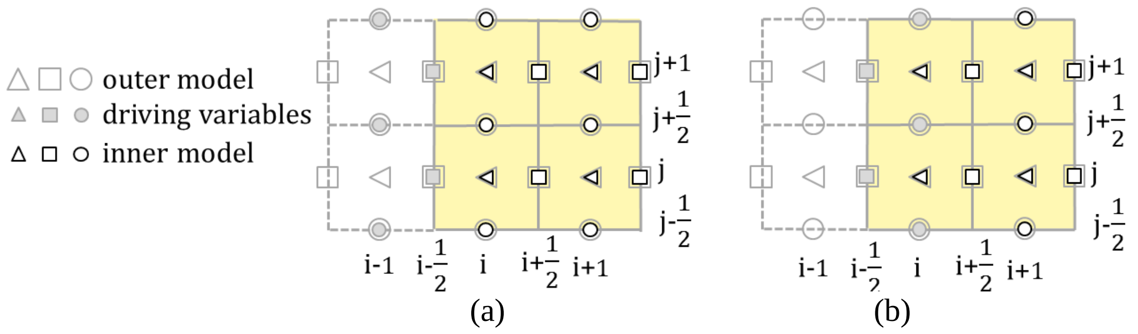


Figure 2.3. OBC implementation (western boundary) in (a) N-Si3D and (b) an alternative approach with tangential driving variables prescribed within the inner model. Refinement parameter r_g*=1* in this example. Yellow cells obtain elevation by the inner governing equation.

	OBC implementation	N-Si3D Figure 2.3(a) clp-et			Figure 2.3 (b) clp-it		
	Position of the interior variable in the cell	Center	Normal	Tangential	Center	Normal	Tangential
	Position of the interior variable in the grid	i	i+1/2	i	i	i+1/2	i+1
Momentum equation	Coriolis	---	---	2N	---	2T	---
	Advection	---	1N	2N/1T	---	2T/1N	1T
	Horizontal Diffusion	---	1T	1N	---	1N	1T
	Wind/bottom stress	---	---	2N	---	2T	---
Continuity equation	Water Surface Elevation	1N	---	---	1N/2T	---	---
	Vertical velocity	1N	---	---	1N/2T	---	---
Transport eq.	Temperature	1N/1tp	---	---	1N/2T/1tp	---	---
	Total per column	3N/1tp	1N/1T	7N/1T	3N/6T/1tp	2N/6T	2T
	TOTAL driving variables	11N/2T/1tp -> 14			5N/14T/1tp -> 20		

Table 2.1. Terms in the inner governing equations that use the driving variables (normal velocity or volumetric transport (N), tangential velocity or volumetric transport (T) and temperature (tp)). clp-et: clamped external tangentials, clp-it: clamped internal tangentials

2.3.4 Inner/Outer dynamic coupling

In order to design the inner/outer dynamic (at execution time) coupling, one needs to identify the outer variables required by each inner time-step. Figure 2.4 (a) shows the outer model variables used by the inner model at each inner time-step in the one-way OBC of the implementation here presented (non-active scalar transport is not included in the figures because it is not used in the applications of Section 2.4) and Figure 2.4 (b) also shows the inner model variables needed by the outer model each outer time-step in a two-way OBC version of the implementation (dashed line).

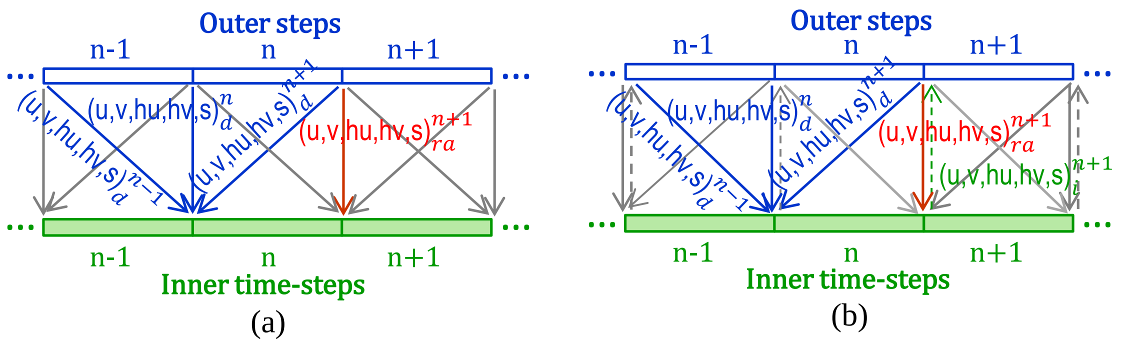


Figure 2.4. Data dependencies between inner/outer models. Driving variables (blue, denoted with the subscript *d*), needed by the boundary cells of the inner model at time-step *n*, for one-way (a) and two-way (b) implementations. Variables (red, denoted with the subscript *ra*) needed from the outer relaxation area in order to implement the relaxation scheme in the inner relaxation area. Variables (green, dashed, denoted with the subscript *i*) transferred from the inner model after time-step *n* to the outer model time-step *n+1* in a two-way implementation (b). Data dependencies (arrows) from outer to inner model (solid) and from inner to outer (dashed lines).

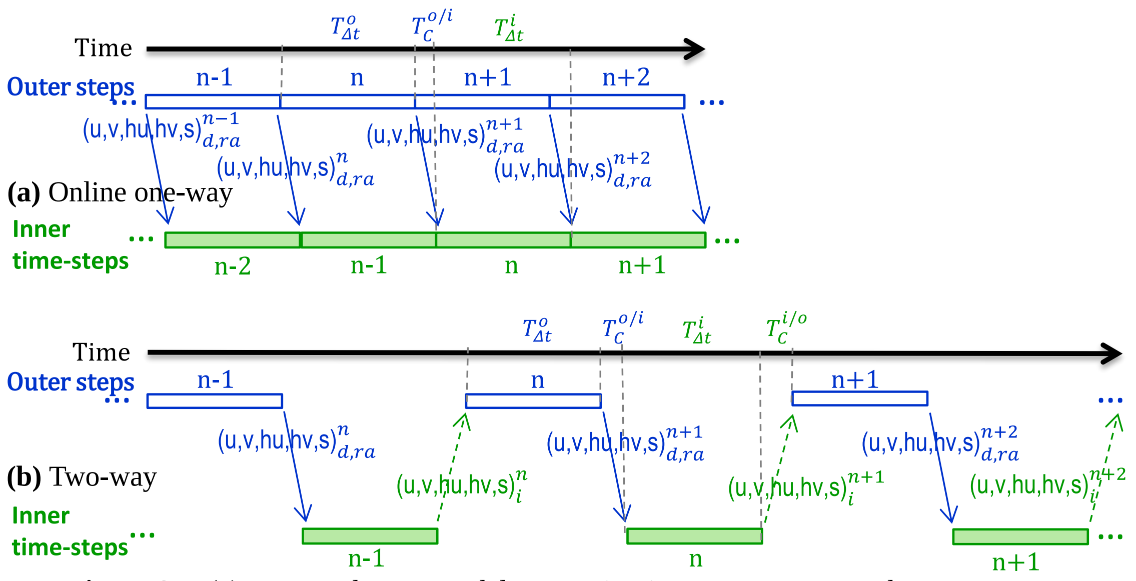
In offline nesting the information to be transferred from the outer to the inner model is stored in a file, so a compromise must be met among storage (capacity/speed/price), and number of values to exchange and transfer frequency. The implementation here presented is online, so there are no restrictions about transfer frequency and variables to be transferred, since reading from and writing to storage is not required. Moreover, in the implementation here proposed the number of communications from outer to inner model occurs in parallel to computation, so they do not affect execution time as the next equations show. The inner and outer models are executed in different processors (cores) with an execution time of (assuming same time-step Δt for inner and outer model, Figure 2.5 (a)):

$$\begin{aligned} T_{one-way} &= T_{\Delta t}^{o} + T_{C}^{o/i} + N_{\Delta t} \times T_{\Delta t}^{i} \ (\approx N_{\Delta t} \times T_{\Delta t}^{i}) \quad \text{when } T_{\Delta t}^{i} \geq T_{\Delta t}^{o} \\ T_{one-way} &= N_{\Delta t} \times T_{\Delta t}^{o} + T_{C}^{o/i} + T_{\Delta t}^{i} \ (\approx N_{\Delta t} \times T_{\Delta t}^{o}) \quad \text{when } T_{\Delta t}^{i} \leq T_{\Delta t}^{o} \end{aligned} \tag{2.4}$$

is the number of time-steps, $T_{\Delta t}^{i}$ is the run time for an inner time-step, $T_{\Delta t}^{o}$ is the run time for an outer time-step, and $T_{C}^{o/i}$is the time required for the communication (/synchronization) of the driving variables at each time-step.

The models can be adjusted to balance $T_{\Delta t}^{i}$ and $T_{\Delta t}^{o}$ (by increasing inner or outer domain or resolution, etc., to improve quality). An offline implementation of the online

Tahoe real example presented in Section 2.4.4 would generate a file of 54 TB, so an expensive and high power consumption storage (and storage network) would be required (shop price of tens of thousands of dollars, plus other costs such as installation, administration and maintenance). The online implementation here presented can be executed in parallel in one inexpensive and low power consumption PC (nowadays commodity inexpensive processors have at least two cores). The code can also run in parallel in a multithread core (also common nowadays) with a higher execution time because inner and outer models would share the core pipeline stages (in particular, the ALU or execution stage). This scheme can be applied to several nesting levels. Execution time will mainly depend on the most time consuming level (as in pipeline architectures) for a number of processing cores equal to the number of levels. The program implemented can also be executed sequentially without any change.



Time
Outer steps
n-1
n
n+1
n+2
$T^o_{\Delta t}$
$T^{o/i}_C$
$T^i_{\Delta t}$
$(u,v,hu,hv,s)^{n-1}_{d,ra}$
$(u,v,hu,hv,s)^{n}_{d,ra}$
$(u,v,hu,hv,s)^{n+1}_{d,ra}$
$(u,v,hu,hv,s)^{n+2}_{d,ra}$
(a) Online one-way
Inner time-steps
n-2
n-1
n
n+1
$T^{i/o}_C$
(b) Two-way
$(u,v,hu,hv,s)^{n}_{i}$
$(u,v,hu,hv,s)^{n+1}_{i}$
$(u,v,hu,hv,s)^{n+2}_{i}$

Figure 2.5. (a) Inner and outer models executing in two processors. Blue arrows represent data communications from outer to inner. Communication time is hidden by computations. (b) Inner and outer models in a two-way nesting must be executed sequentially. Blue and green arrows represent data communications between inner and outer. Total time depends on the inner time plus outer time plus communication time from inner to outer (variables of the whole 3D nesting grid) plus communication time from outer to inner (variables in the 3D boundary).

In a two-way implementation, the inner model cannot be executed in parallel to the outer model (Figure 2.5 (b)):

$$T_{two-way} = N_{\Delta t} \times \left(T^o_{\Delta t} + T^{o/i}_C + T^i_{\Delta t} + T^{i/o}_C\right) \qquad (2.5)$$

2.3.5 Improvement of data structures in the Si3D implementation

The 3D model variables are stored in one- or two-dimensional arrays that in order to decrease memory requirements and computational time do not reserve space for dry columns. 1D arrays store 2D variables defined in the *x* and *y* directions, such as ζ. 2D arrays store 3D variables defined in the *x*, *y* and *z* directions with the first dimension used for *x* and *y*, and the second for *z*. Neighboring north, south, east and west water columns are accessed using 4 arrays of one dimension (one for each direction). These arrays avoid the extra operations needed to identify neighbor columns, while taking up only minuscule memory space; for example, these arrays take up 0.05% of the total memory required in the Lake Tahoe model and 0.02% in the Sacramento River model.

2.4 N-Si3D model evaluation

2.4.1 Tests, metrics and examples

Three kinds of tests and three kinds of examples (synthetic rectangular channels with constant wind, Sacramento River and Lake Tahoe) are used to verify, validate and evaluate the model. The tests are:

A. A nesting test with outer and inner models using the same grid resolution ($r_g=1$) is done with the three kinds of examples to verify the implementation. This test demonstrates that the inner and outer domains are coupled seamlessly. In this test, the velocity fields *u* (East-West or E-W), *v* (North-South or N-S) and *w* (vertical), water surface elevation (ζ) and temperature (*s*, active scalar concentration) of both models are compared. A normalized error is obtained for each variable at every output epoch in several layers (see metrics below). ζ is a 2D variable, conceptually associated to the first layer, so only one *NRMSE* is computed per output epoch. In addition, the water volumes (outer and inner) in the nested area are compared in order to demonstrate volume conservation between outer and inner models. This volume is calculated by multiplying the sum of ζ by the product $\Delta x \times \Delta y$, where Δx and Δy are the horizontal dimensions of the cells (volume of the first layer).

B. A nesting simulation with an inner model using a grid resolution higher than the outer model ($r_g>1$) and a *non-nesting* simulation with grid resolution equal to the nested inner model is done with the three kinds of examples. This test is designed to demonstrate that the inner model obtains results similar to those of a complete high-resolution model (HR basin) with much lower computational cost. In this test, all variables (u, v, w, *ζ*, *s*) and the first-layer volume of the inner and the HR basin models are compared as in test A.

C. A nesting test with an inner model using a grid resolution higher than the outer model ($r_g>1$) is done with the two real examples in order to demonstrate that local-scale features in the flow cannot be accurately resolved using larger cell sizes (the LR model) or if local-scale irregularities in bathymetry and topography are not appropriately represented. An HR model, which is able to reproduce the length scales of these irregularities, is required to resolve these features and the nesting implementation here presented accomplishes it in much less time and using much less resources.

Differences between the outer and inner model (in tests A) and between the HR basin and inner model solutions (in tests B) after multiple simulation time-steps are taken as a measure of the validity of the nested algorithm implementation. This differences were quantified calculating a root mean squared error (*RMSE*) and a normalized root-mean-squared error (*NRMSE*) as in other validation works (Debreu et al. 2012, Halliwell et al. 2009, Kourafalou et al. 2009, Pairaud et al. 2011, Son et al. 2011):

$$RMSE = \sqrt{\frac{\sum_{c=1}^{N_{im}}(x_1^c - x_2^c)^2}{N_{im}}} \tag{2.6}$$

$$NRMSE = \frac{RMSE}{x_{max} - x_{min}} \tag{2.7}$$

Where, x_1 is a variable defined in cells of one layer in the outer model or HR basin and x_2 the same variable in the inner model; x_{max} and x_{min} are the maximum and minimum values found in that layer in the nested zone for both models being compared (as in (Halliwell et al. 2009, Son et al. 2011)); N_{im} is the total number of layer cells in the nested zone. A temporal *RMSE* is also used to compare scalars in a time-step, such as cross-sectional flow or total volume, along the whole simulation:

$$RMSE_t = \sqrt{\frac{\sum_{t=1}^{N_{\Delta t}}(v_1^t - v_2^t)^2}{N_{\Delta t}}} \tag{2.8}$$

Area averaged kinetic energy, excess mass, and energy flux are all calculated as in Herzfeld and Andrewrtha (2012) for the test B using a channel. These metrics are useful for identifying problems in the boundary implementation and for studying whether rim currents and other undesirable features may be present in the inner solution due to inconsistencies between the external data provided by the outer model and the evolution of the inner model. Averaged kinetic energy is computed by:

$$TKE = \frac{1}{2A}\int_A \rho(U^2 + V^2)dA \tag{2.9}$$

which computes kinetic energy using Eq. (1.24) of Kowalik and Murty (1993) (as in Herzfeld and Andrewrtha (2012)); $U = \int_{-H}^{\zeta} udz/D$ and $V = \int_{-H}^{\zeta} vdz/D$ are depth averaged velocities in the x and y direction, respectively, ζ is the water surface elevation, H is the bottom depth, and D is the total depth $D = \zeta + H$. Area averaged water level is computed using the excess mass metric of Palma and Matano (1998) (as in Herzfeld and Andrewrtha (2012)):

$$EM = \frac{1}{A}\int_A \zeta dA \tag{2.10}$$

It is used to check that the boundary implementation properly represents mass fluxes through the open boundaries (Palma and Matano 1998). Finally, energy flux is computed for a western I/O boundary by Eq. 6 of Palma and Matano (2001) (as in Herzfeld and Andrewrtha (2012)):

$$OBC_{flux} = \frac{1}{W}\int_W DU_{i/o}\left(g\zeta + \frac{1}{2}(U^2 + V^2)\right)dy \tag{2.11}$$

where W is the width of the frontier beside the I/O boundary and $U_{i/o}$ is the depth averaged normal velocity at the I/O boundary.

In the first example, the domain is a rectangular basin with a small number of grid cells (Section 2.4.2). The sub-basin is the southern end of this basin for the Test A and a southern central rectangle for the Test B. In the second example (Sacramento River, Section 2.4.3), the sub-basin is a bend in the river with high values of velocities (normal and tangential) and both lateral re-circulation and secondary circulation zones. In the last example (Lake Tahoe, Section 2.4.4) the model is used to resolve the circulation and transport patterns in the lake's near-shore regions. The sub-basin is the southeast corner of the lake. Tangential velocity can affect the quality of the results significantly as Section 2.4.5 shows. Table 2.2 summarizes the characteristics of the examples and the resolutions (for outer, inner and HR basin), temperature, simulation period, time-steps, output epochs and kind of tests used in each example.

2.4.2 *Case 1: Rectangular synthetic channels.*

Test A was applied to a rectangular flat bottom basin aligned in the N-S direction, with a length of 11000 km and a width of 110 km, both values greater than the barotropic Rossby radius (channel 1, 2nd row in Table 2.2). It has a constant depth of 10 m. The channel is discretized using square cells of 5.5 Km (Δx=Δy), and a vertical dimension of 0.5m (Δz). The inner grid covers the southern 2750 km of the channel. The model is forced with a southern-wind speed of 6 m/s. Temperature is 25ºC at free surface and decreases 0.5ºC per layer, reaching 15.5º at bottom layer. The period of time simulated is 60 days (steady state before 20 days) and the time-step is set to 50 seconds.

Examples	**Basin size**	**Inner size**	**Outer resolution**	**Inner resolution**	**HR basin**	**Temp-erature**	**Simulation period**	**Time-step**	**Output epoch**	**Test**
Channel 1	11000km× 110km×10m	2750km× 110km×10m	5.5km× 0.5m	5.5km× 0.5m		gr(25-15.5ºC)	60 days	50s	1h	A
Channel 2	1000km× 540km× (50m-110m)	420km× 280km× (50m-81.2m)	20km× (2m,5m)	4km× (2m,5m)	4km× (2m,5m)	cst(20ºC), gr(25-14.5ºC)	30 days, 40 days (with temp. gr.)	50s, 450s, 1800s	30'	B
Sacra-mento River	32km×170m × (1m-16m)	1670m×1720 m ×(1m-13m)	10m/ 49626c	10m/4593c 5m/17700c 2m/109067c	5m/ 198918c	cst	10 days (Julian Days 8-17, 2009)	4s/10m 1s/5m 1s/2m	1h	A B C
Lake Tahoe	20km×30km × (0.2m-0.5km)	7420m×6320 m ×(0.2m-430m)	100m/ 49215c	100m/3932c 20m/99722c	20m/ 1244896c	sensor outputs	30 days (Julian Days 185-215, 2008)	50s/100m 10s/20m	1h	A B C

Table 2.2. Example characteristics: size of the complete basin and the inner domain (length×width ×depth); resolution used in the nesting simulations (outer and inner grid), and in the high-resolution (HR) basin simulations (square cell horizontal side × cell depth / number of columns); temperature (cst=constant, gr: gradient from top to bottom, -0.5ºC per layer); time period of the simulations; simulation time-step for the different sizes of the square cells (second /square cell side); time period between output epochs (hours, minutes); and test conducted.

Tests A and B were applied to an open zonal channel (channel 2, 3rd row in Table 2.2) forced with constant alongshore wind (Chapman 1985, Herzfeld and Andrewrtha 2012, Palma and Matano 1998, Palma and Matano 2001) (Figure 2.6). (Chapman 1985, Palma and Matano 1998, Palma and Matano 2001) used 2D models, for which the linearized version has an analytical solution in an unbounded domain (Chapman 1985). Data bellow for channel 2 are equal to those of the idealized test case in Herzfeld and Andrewrtha (2012) (Sect. 3.1). The channel has a bottom slope in the N-S direction (from 110m at the northern coast to 50m at the southern coast). The outer model solution for the open zonal channel is achieved by mimicking an infinite coast using cyclic open boundaries. The model is run in the southern hemisphere, using constant Coriolis of -0,0001028s^{-1} (Rossby radius is 215.44km). There are 22 vertical layers, 2 m depth the surface and 5m depth the other layers. Both, the LR outer and the HR basin grids are 1000 km x 540 km and the HR inner grid is 420 km x 280 km. Both, the inner and basin grids have horizontal square cells of 4km and the outer grid of 20 km (r_g=5). Temperature is maintained at 20ºC. The bottom drag coefficient is 0.025. An alongshore wind of 8 m/s with drag coefficient of 0.00128 is applied, which creates an upwelling solution with an elevation increasing northwards to support an eastwards geostrophic flow. Time-steps of 50s and 450s have been used. The period of time simulated is 30 days. This test allows comparing the accuracy of the nesting procedure implemented with the accuracy of other methods and checking the sensitivity of some parameters in the quality of the results. An additional channel 2 simulation with a vertical temperature gradient has been conducted here. Temperature is 25º at free surface and decreases 0.5ºC per layer, reaching 14.5º at bottom layer.

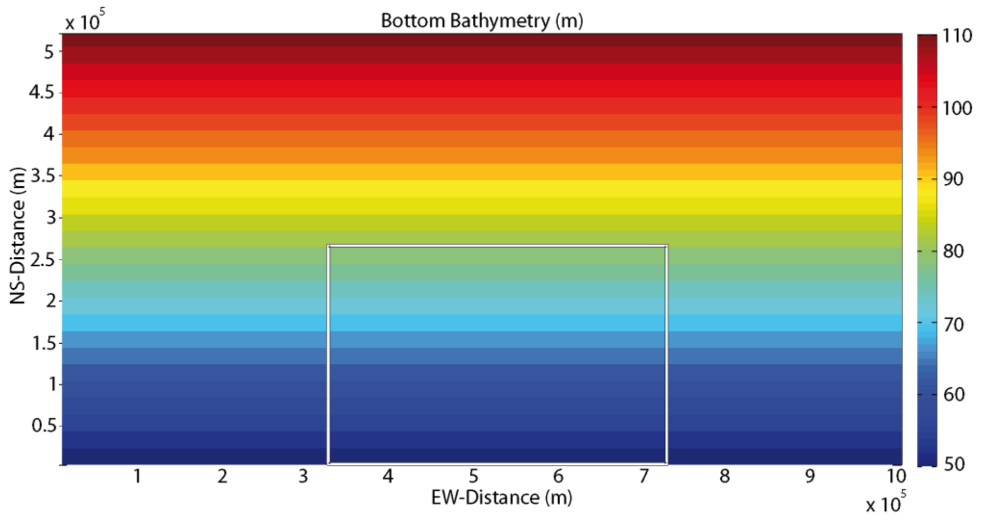


Figure 2.6. Channel 2: High-resolution basin and inner model bathymetry (white rectangle).

2.4.2.1 *Test A. Comparison of inner and outer models with the same resolution.*

Every hour, all the variables (u, v, w, ζ and s on the free surface plane) were collected, and the outer and inner solutions in the southern end of the basin were compared. The *NRMSEs* (Eq.(2.7)) for all variables and the temporal *RMSE* (Eq.(2.6)) for water volume (in % of average volume in the nested area of the outer solution) are all less than 4% in all output epochs and all layers (3rd column in Table 2.3). This error is low and is due to the preconditioned conjugate-gradient method used to solve the five-diagonal system of equations for water surface elevation. An iterative solver, such as the preconditioned conjugate-gradient (PCG) method, converges to the solution of the equation system with much lower computational cost than a direct solver. However, the solution using a direct solver is exact (excluding round-off error), while the solution with an iterative solver is approximate within a tolerance, which is set by the user. In real applications, using direct solver is not efficient due to the large systems of equations involved. However, for small problems, a direct solver can be used to eliminate the approximation error in the solution of the system. In this test case we used Gaussian elimination for both inner and outer models. The *NRMSEs* for u, v, w, ζ and s are then all zero (2nd column in Table 2.3). These results demonstrate that the nested approach here proposed does not introduce errors when inner and outer grid resolutions are the same. No volume drift over time is observed with or without relaxation area.

	Channel 1		Channel 2				Sacramento River				Lake Tahoe				
	Test A	Test A	Test B		Test B		Test A	Test A	Test B	Test B	Test A	Test A	Test B	Test B	
	inner/ outer (Gauss)	inner/ outer (PCGM)	inner/ HR basin (Gauss)		inner/ HR basin (PCGM)		inner/ outer	inner/ outer, clp0-t	inner/ HR basin, clp-it	inner/ HR basin	inner/ outer	inner/ outer, clp0-t	inner/ HR basin, clp-it	inner/ HR basin	Eq.
Resoluti.	5.5Km/ 5.5Km	5.5Km/ 5.5Km	20km/4km		20km/4km		10m/ 10m	10m/ 10m	5m/5m	5m/5m	100m/ 100m	100m/ 100m	20m/ 20m	20m/ 20m	
Temp.	gr	gr	cst	gr	cst	gr	cst	cst	cst	cst	sensor	sensor	sensor	sensor	
Time	60d	60d	30d	40d	30d	40d	10d	10d	10d	10d	30d	30d	30d	30d	
Time-step	50s	50s	50s		50s		4s	4s	1s	1s	50s	50s	10s	10s	
Epoch	1h	1h	30’		30’		1h	1h	1h	1h	1h	1h	1h	1h	
u (E-W)	0	3.36	1.32	1.39	4.27	4.41	1.94	6.15	5.03	4.12	2.19	2.93	3.32	2.99	7
v (N-S)	0	3.39	0.86	0.90	3.53	3.66	1.63	3.30	3.51	2.26	1.8	2.42	3.09	2.66	7
w (vert.)	0	3.51	1.11	1.16	3.61	3.72	0.62	2.26	4.36	3.42	0.88	0.98	1.66	1.24	7
ζ (wse)	0	3.07	0.57	0.61	2.99	3.11	0.10	0.44	0.37	0.33	1.57	1.91	1.91	1.82	7
s (temp.)	0	0.21		0.25		0.32					0.87	0.88	1.76	1.68	7
volume	0	0.95	0.21	0.22	0.89	0.91	0.1	0.72	0.09	0.08	0.13	0.13	0.21	0.2	8

Table 2.3. *NRMSEs* (Eq. (2.7), in %) for velocity fields u, v and w, water surface elevation (ζ) and temperature (s, active scalar concentration), and $RMSE_t$ (Eq. (2.8) in % of average volume) for volume ($\Sigma\zeta \times \Delta x \times \Delta y$) obtained for the different examples in the top layer. Inner model variables are compared to LR outer model (Test A) or HR basin model (Test B) variables. Outer/inner resolutions, simulation period (days), time-step (seconds), output epoch period (hours, minutes), and temperature (cst=constant, gr: gradient in Table 2.1, or sensor outputs) are also given. Gauss/PCGM indicates the solver used to obtained ζ in the synthetic examples. clp0-t=OBC for tangential velocities is clamped to 0. clp-it= tangential velocities within inner border cells are clamped to external outer values.

2.4.2.2 *Test B. Comparison of the inner model within a low-resolution outer model with a high-resolution model of the complete basin.*

The models were simulated for 30 days (as in Herzfeld and Andrewrtha (2012)), when the solution had reached a steady state. The resulting water surface elevation and velocity fields are shown in Figure 2.7. The figure shows the solution of the HR basin and the inner model superimposed for a simulation that transferred the information from outer to inner model every 1800s with a time-step of 50s (similar result are obtained for time-steps of 450s). The solution of the inner model is not deviated from, and shows a good continuity with, the solution achieved in the HR basin. TKE, EM and OBC_{flux} metrics (Eqs. (2.9), (2.10) and (2.11)) have also been used to compare the inner HR solution to the outer LR and HR basin models in the nested area.

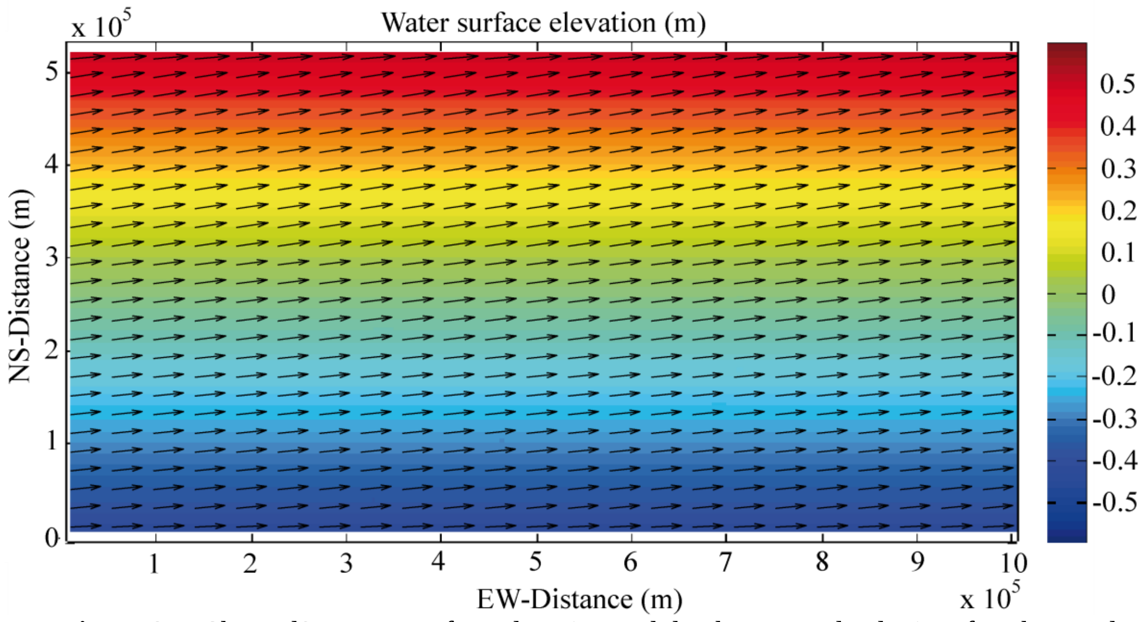


Figure 2.7. Channel2: Water surface elevation and depth averaged velocity after the steady state is reached (30 days after the simulation starts) for the HR basin model, with the nesting inner model superimposed.

Steady state TKE, EM and OBC_{flux} are shown in Table 2.4 for different parameter combinations. The differences between inner and HR basin solutions, when the steady solution is reached, are those in columns 7, 8 and 9. The differences between inner and LR basin solutions are similar, as columns 10, 11 and 12 show for some parameter combinations. Table 2.4 shows an important error reduction if the iterative solver (PCG) used to obtain water surface elevation ζ is substituted by a direct method (Gaussian elimination). The differences are zero when both inner and outer models have the same resolution, even without relaxation (row 7); the nesting implementation does not introduce any errors (under-specification or over-specification) when there are neither spatial interpolation nor bathymetry mismatch. The differences are slightly reduced, both with and without relaxation area, if the inner model is forced each time-step (row 4 vs. 5, 11 vs. 12) because time interpolation errors disappear, and when the time-step increases (from 50s to 450s) because interactions between inner and outer model decrease. The use of the OBC implementation in Figure 2.3(b) introduces additional errors, as can be seen comparing rows 12 and 13. The results obtained with the OBC implementation are good compared to those obtained in Herzfeld and Andrewrtha (2012) for different OBCs (clamped with local flux adjustment, radiation, Flather, etc., Table 4.1, Figure 4.2 in Herzfeld and Andrewrtha (2012)). Figure 2.8 (left axis, continuous line) shows the time evolution of EM for several configurations, in particular the configurations in rows 11, 14, 4, 7, 6, 8 and 9 in Table 2.4 . There is no EM drift when the flux relaxation scheme is applied to the clamped OBC and the EM drift without relaxation is hardly observed. The differences between inner and outer EM in the white rectangle in Figure 2.6 are also shown in the figure (right axis, dashed lines) so that the EM drift obtained with the clamped OBC without relaxation can be detected in the figure; in particular, those of the configurations in rows 7 and 14 (50s transfer step, No-RA). Simulations with 450s time-step with relaxation have similar slope than those obtained with 50s time-step. With the clamped OBC without relaxation, the EM drift has clearly lesser slope than the clamped OBC without relaxation in Herzfeld and Andrewrtha (2012) (Fig. 4.2). Figure 2.8 also shows data for transfer step of 7200s with and without relaxation. These graphs allow seeing more clearly the goodness of the relaxation implementation in avoiding mass drift. If the relaxation area is not used, the nesting simulation obtains low errors for transfer frequencies up to 3600s. With relaxation area, the water surface elevation starts to increment uncontrollably for transfer frequencies over 21600s.

		I&O	O->I	Relax-	OBC	inner vs. HR basin			inner vs. LR basin		
ζ solver	r_g	time-step	transfer step	ation area	impl.	TKE (%)	EM (%)	OBC_{flux} (%)	TKE (%)	EM (%)	OBC_{flux} (%)
PCGM	1	450s	450s	No	clp-et	2.46	2.11	1.96			
PCGM	5	450s	450s	Yes	clp-et	3.42	2.73	3.76			
PCGM	5	50s	50s	Yes	clp-et	3.53	2.82	3.8	3.55	2.84	3.79
PCGM	5	50s	1800s	Yes	clp-et	3.66	2.94	3.99	3.73	2.97	4.06
PCGM	5	50s	7200s	Yes	clp-et	3.84	4.39	6.12			
PCGM	5	50s	50s	No	clp-et	4.14	6.21	9.39	4.26	6.33	9.52
PCGM	5	50s	7200s	No	clp-et	7.51	10.66	15.66			
Gauss	1	450s	450s	No	clp-et	0	0	0	0	0	0
Gauss	5	450s	450s	Yes	clp-et	0.92	0.42	1.33			
Gauss	5	50s	50s	Yes	clp-et	0.99	0.47	1.39	1.01	0.48	1.41
Gauss	5	50s	1800s	Yes	clp-et	1.16	0.6	1.65	1.18	0.61	1.68
Gauss	5	50s	1800s	Yes	clp-it	1.49	0.76	1.73			
Gauss	5	50s	50s	No	clp-et	1.33	3.59	5.07	1.35	3.63	5.14

Table 2.4. Steady state TKE, EM, OBC_{flux} (western boundary) for different simulations. These are the *RMSE* errors for the 30 days simulation expressed as a percentage of the steady state of the HR basin (left) and of the LR basin (right). Columns (from left to right): iterative solver used to obtain water surface elevation ζ (PCGM or Gauss*)*, grid refinement parameter r_g, inner and outer time-step, frequency of transfers from outer to inner, interface with (yes) or without (no) relaxation area, and OBC implementation (clpt-et in Figure 2.3(a), clp-it in Figure 2.3(b)).

Additionally, simulations with a gradient in temperature have been executed, showing a slightly increase in errors as can be observed consulting the results simulations in Table 2.3 (columns 4 and 5, gr data). The relaxation area avoids EM drift in all the variables, including temperature.

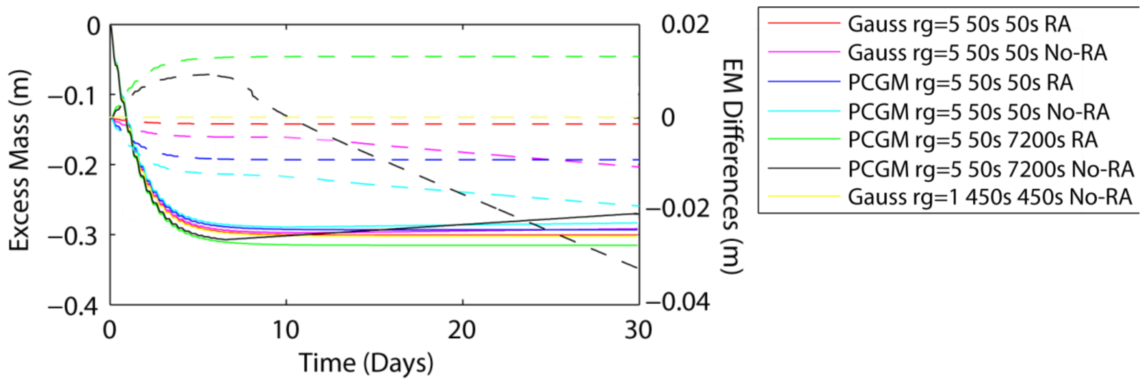


Figure 2.8. Time series of EM for the inner model (left axis, continuous line) and differences between inner and outer EM in the white rectangle in Figure 2.6 (right axis, dashed lines) for several configurations: type of ζ solver (PCG or Gauss), r_g (1 or 5), inner and outer time-step (50s or 450s), outer to inner transfer step (50s, 450s or 7200s), with (RA) or without (No-RA) relaxation area.

2.4.3 Case 2: Sacramento River

The hydrodynamics of the lower reach of the Sacramento River (Figure 2.9) with a length of 32 km, between the populations of Freeport (38º 27' 22'', 121º 30' 01'') and Walnut Grove (38º 14' 20'', 121º 31'18'') in California, were modeled (4th row in Table 2.2). The reach has an average depth of 6 m, with maximum depths of approx. 16 m, and an average width of approx. 170 m. The river hydrodynamics are influenced by both flows from the Sacramento River and tidal dynamics due to its proximity to the sea. Juvenile Chinook salmon migrates through this reach of the Sacramento River in their way to the sea, where the existence of different river distributaries (see for example the diffluence of the Sacramento River into Sutter Slough, USGS 11447830, Figure 2.9) allow juveniles to choose different migration routes to the sea (Perry et al. 2010). The percentage of salmon entering each of the routes is primarily driven by the amount of water from the Sacramento River that enters each of them (Perry et al. 2010); however, since many of these river distributaries are located where the Sacramento River bends (see Figure 2.9), the lateral circulation, characteristic of these river environments, could drive salmon towards the outer region of the bends where most entrances to the different migration routes are located, and hence, it could increase the entrainment of salmon towards these routes. Thus, being able to reproduce lateral circulation at bends is of key importance for any model trying to reproduce salmon route selection in the lower reaches of the Sacramento River. Reproducing the lateral circulation and small-scale vortices in the area of meanders requires a high resolution grid. Results here presented demonstrate that a bathymetry of 2m square columns clearly reveals effects not correctly simulated with lower resolution bathymetries (5m or 10m).

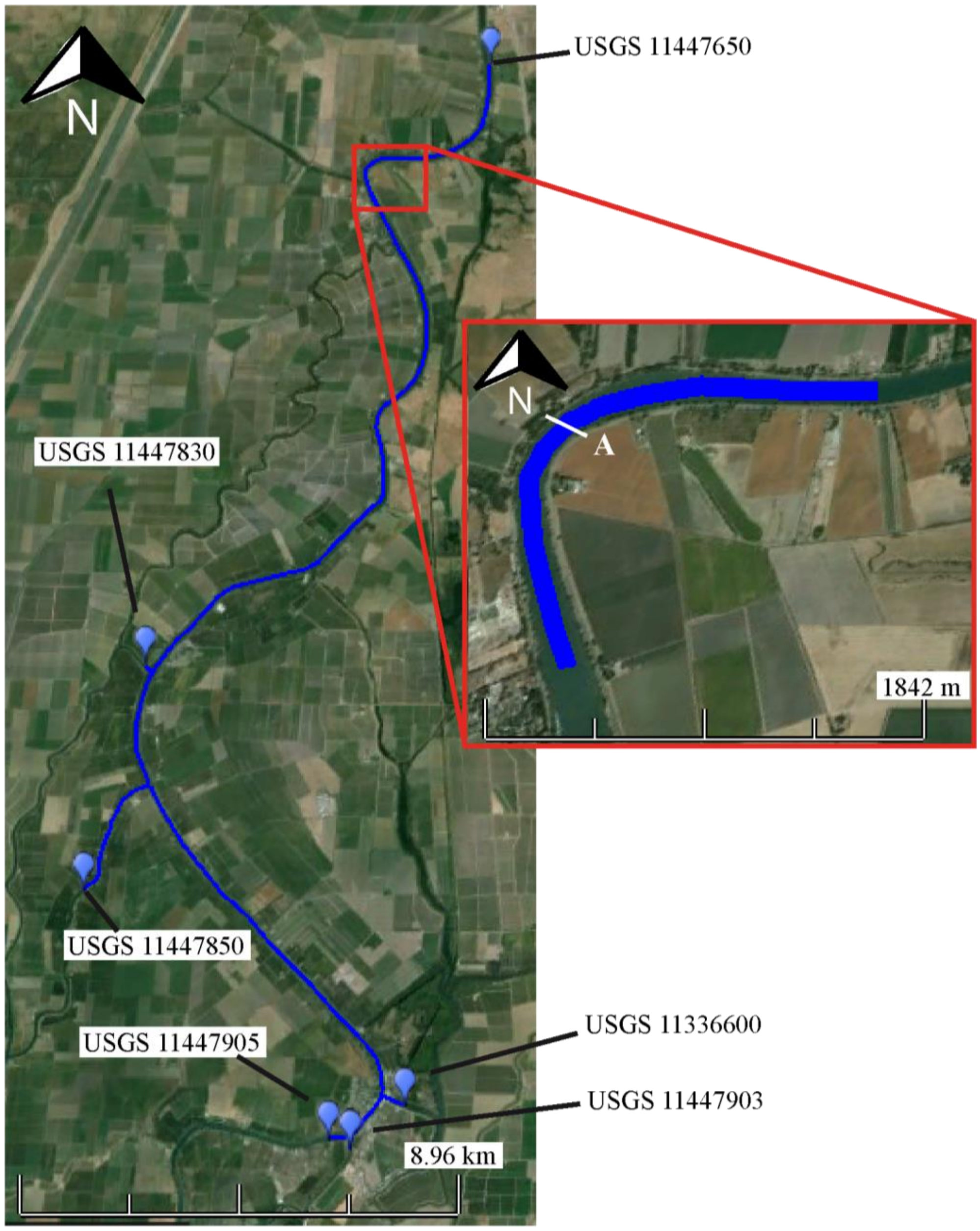


Figure 2.9. Sacramento River hydrodynamic model domain (in blue): domain for the low-resolution outer model and the high-resolution basin model (left) and domain for the inner model (right). Markers show the location of USGS gaging stations used as model boundary conditions. The location of section A inside Clarksburg Bend is also shown.

A complete 2m simulation would be computationally expensive and, for the complete reach, 5m square columns is the maximum resolution provided by the United States Geological Survey (USGS) we have accessed. The nested procedure here proposed allows resolving this problem with an acceptable computational time by using a 2m high resolution grid in the meander areas. Here, we present the results of the outer model, inner model and high-resolution (HR) basin for the region of Clarksburg Bend (enclosed region in Figure 2.9), where lateral circulation has been proven to occur Dinehart and Burau (2005). The outer domain includes the whole basin (shown in Figure 2.9 (left)) and is discretized using grid cells of size 10m x 10m in the horizontal plane with 49626 water columns while the inner domain uses 10m x 10m (Test A) with 4593 water columns, 5m x 5m (Test B) with 17700 water columns or 2m x 2m (Test C) with 109067 water columns, i.e. r_g=1, r_g=2 or r_g=5, respectively (4th row in Table 2.2). The HR basin uses 5m x 5m cells (test B) with 198918 water columns, which is the maximum resolution available for the complete reach provided by the USGS. The time-step is 4s for the test A simulations (10m) and 1s for tests B and C (5m and 2m) for stability purposes.

The boundaries of the outer domain were chosen to match the location of existing USGS gaging stations (Figure 2.9). Flows were used as the boundary conditions for locations USGS 11447650, 11447830, 11447850 and 11336600, based on observations collected at each of these gaging stations, in a given time period. Free surface elevations at the remaining two model boundaries were forced to vary according to observations collected at USGS 11447905 and 11447903, respectively. Temperature comparison is not provided for Sacramento River, because it was kept constant (with a uniform temperature equal to 8º C typical of the winter period) through all the simulation due to the negligible variations in Sacramento River temperatures during the study period (January 2009). Sacramento River temperatures normally begin to decline in October, remain uniform from December to March, and begin to increase in April. The model was simulated during a period of time of 10 days in 2009 starting on January 8th and ending on January 17th, with hourly output epochs.

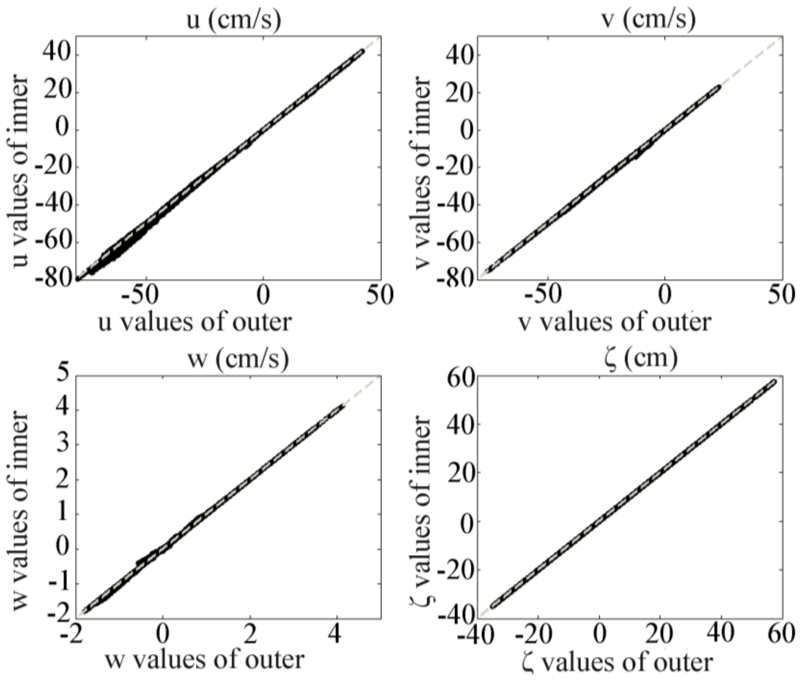


Figure 2.10. Sacramento River, test A: *u*, *v*, *w and ζ* compared between inner and outer model solutions for surface layer at all output epochs. Each point represents the value obtained by the inner model (y-axis) against the solution obtained by the outer model (x-axis). If both solutions coincide, the point is located on the dashed line.

2.4.3.1 *Test A. Comparison of inner and outer models with the same resolution.*

Inner and outer models have been compared in the nested area (enclosed region in Figure 2.9). Figure 2.10 compares, for each variable evaluated, the solution obtained by the inner model (y-axis) with that obtained by the outer model (x-axis), for all output epochs simultaneously. When both solutions are equal, that value is placed on the dashed line $y=x$. If the outer model solution is higher, the point is placed in the lower triangle while if it is lower, it is placed in the upper triangle. The compared variables are horizontal velocities (u and v) vertical velocity (w) and water surface elevation (ζ). The *NRMSEs* (Eq. (2.7)) in the top-most layer averaged over the simulation time are 1.94%, 1.63%, 0.62% and 0.1% for u, v, w and ζ, respectively (6th column in Table 2.3). Similar or better results are obtained in the other layers. Finally, the sum of ζ is used to check for volume conservation ($(\Sigma\zeta) \times \Delta x \times \Delta y$). The temporal *RMSE* (Eq. (2.8)) is 0.19m^3, which represents 0.1% of the average volume over the simulation time measured in the nested area of the outer model.

2.4.3.2 Test B. Comparison of the inner model within a low-resolution outer model with a high-resolution model of the complete basin.

The inner and HR models, both with 5m horizontal resolution, have been compared in the nested domain (Figure 2.9(right)) excluding the relaxation area. The *NRMSEs* (Eq. (2.7)) in the top layer averaged over the simulation time are 4.12%, 2.26%, 3.42% and 0.33% for u, v, w and ζ respectively (9th column in Table 2.3). Similar o better results are obtained in other layers. Finally, the sum of ζ is used to check for volume conservation ($(\Sigma\zeta) \times \Delta x \times \Delta y$). The temporal *RMSE* (Eq. (2.8)) is 0.16m^3, which represents 0.08% of the average volume over the simulation time measured in the nested area of the HR basin model. These results indicate that the inner model yields similar solutions in the meander area to those of the HR basin model, but with a much lower computational cost. No volume drift over time is observed in the simulation period.

The test has been repeated using the OBC implementation in Figure 2.3(b). The *NRMSEs* (Eq. (2.7)) in the top-most layer averaged over the simulation time are 5.03%, 3.51%, 4.36% and 0.37% for u, v, w and ζ, respectively (Table 2.3, clpt-it). The temporal *RMSE* (Eq. (2.8)) used to check for volume conservation is 0.09% of the average volume over the simulation time in the nested area.

2.4.3.3 Test C. Comparison with low-resolution outer model

We compared the outer and inner solutions looking for local-scale hydrodynamic features which are well resolved by the inner model and not by the outer model. Modeled values from the outer and inner grids were evaluated at section 4 of Dinehart and Burau (2005) in Clarksburg Bend (corresponding with section A in Figure 2.9), where lateral circulation is known to occur. Modeled values were averaged over a peak ebb tide (~0.5 hr) and then interpolated (nearest method) to the location of points in section 4 of Dinehart and Burau (2005). Lateral circulation was calculated as the velocity field parallel to the plane of the cross-section. No further reorientation of the cross-section was done.

The outer LR model (10m horizontal resolution) was unable to correctly reproduce the lateral circulation (Figure 2.11(a)), as well as the inner model using a horizontal resolution of 5m (Figure 2.11(b)). However, the 2m inner model (Figure 2.11(c)) was able to reproduce it, possibly as a result of a better representation of the outer bank, where the stair-stepping effect, characteristic of a Cartesian grid domain, is smoothed. Model and field data are not quantitatively comparable since (1) lateral velocities in the field were calculated in a time period when average discharges were > 650 m^3s^{-1} (while in our model discharges during peak tides are ≤ 400 m^3s^{-1}), and (2) Dinehart and Burau (2005) reoriented *u* and *v* to match suspension indicators (backscatter signal). On qualitatively basis, however, the pattern of lateral circulation in the model is similar to that observed in the field (Dinehart and Burau 2005), with velocities directed outwards near surface and directed inwards near the bed, with the zero velocity isoline located at middle depths. Our model computed outer velocities that are stronger than inner velocities in section A, while inner and outer field velocities were similar in magnitude. This is, however, the result of no further reorientation of the cross-section in the model.

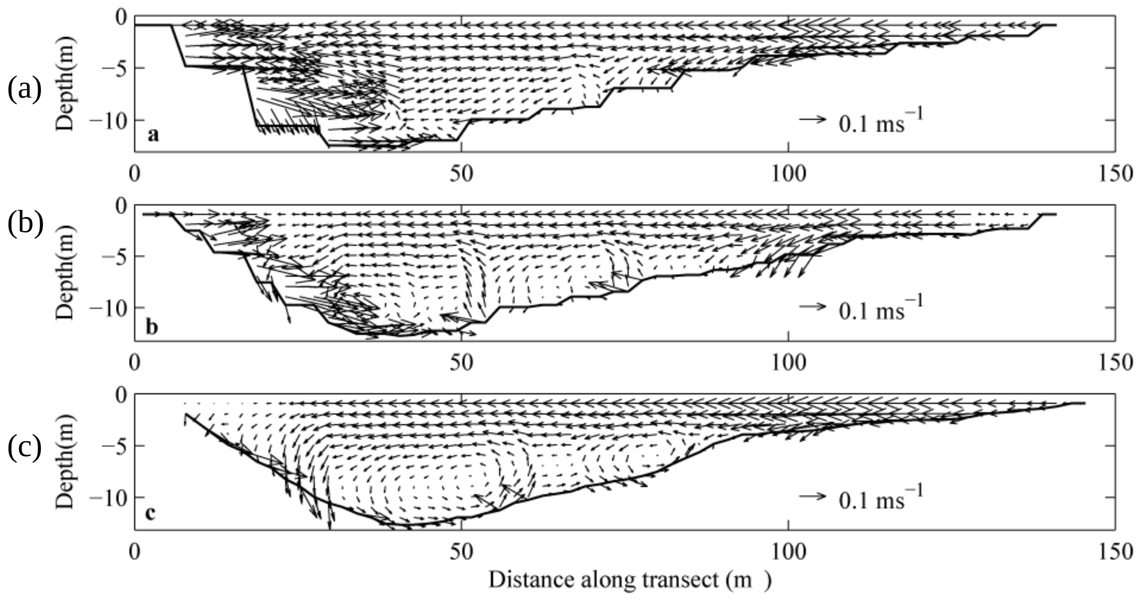


Figure 2.11. Sacramento River, test C: Lateral circulation at Clarksburg Bend (A in Figure 2.9), the location of section 4 of Dinehart and Burau (2005), according to (a) the outer 10m-resolution results, (b) the inner 5m results and (c) the inner 2m results. Views are upstream.

2.4.4 Case 3: Lake Tahoe

The nested-grid approach is being used to resolve near-shore circulation in Lake Tahoe (Figure 2.12). The extraordinary variability of the physical environment in the near-shore makes the task of characterizing it by means of observations a challenge.

Near-shore circulation can be used in different studies. For example, in Lake Tahoe, it is going to be used to develop a long-term risk assessment of Asian clam growth, spread and impact. In this study, the near-shore circulation can be used to develop a transport model of Lake Tahoe to characterize the pathways of transport of young life stages of Asian clams from the existing beds to other near-shore areas. To achieve this goal, a high resolution model must be used where fine-scale information is needed (such as in the near-shore). The nested approach to developing a high-resolution transport model of the near-shore is justified because the high-resolution simulation of the whole lake (roughly 20km x 30km in the horizontal dimension and a depth of up to 500 meters in the vertical dimension) would require expensive computation hours in expensive and high energy consumption parallel computers (5th row in Table 2.2).

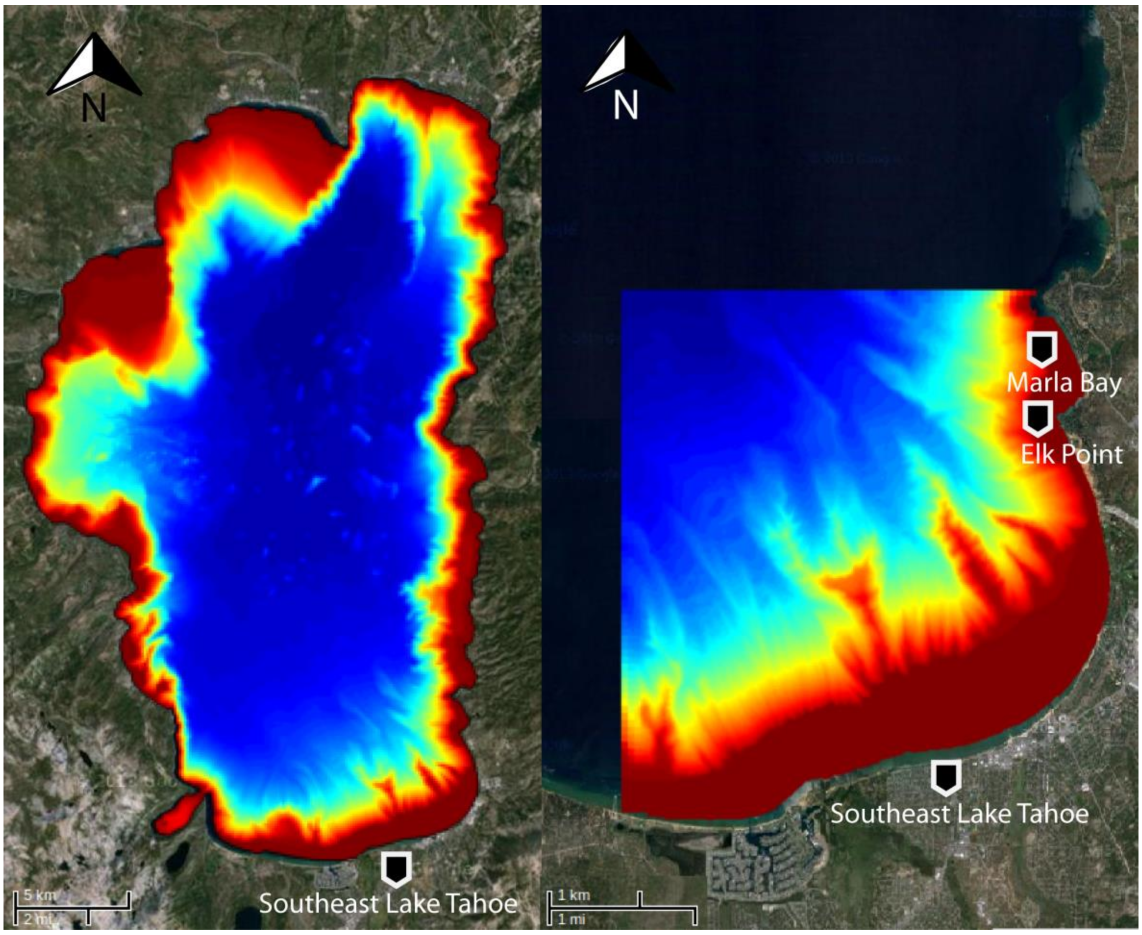


Figure 2.12. Lake Tahoe hydrodynamic model domain: domain for the low-resolution outer model and the high-resolution basin model (left) and domain for the inner model (right).

The selected study area for the observational experiments was the region adjacent to the largest urban area, Southeast Lake Tahoe (Figure 2.12). This is where the greatest anthropogenic effects are known to occur and declining water quality has been measured (Taylor 2004). The outer domain is discretized using grid cells of size 100m x 100m in the horizontal plane with 49215 water columns, while the inner domain uses 100m x 100m cells (test A) with 3932 water columns or 20m x 20m (tests B and C) with 99722 water columns, i.e. r_g=1 or r_g=5, respectively. The HR basin uses 20m x 20m cells (test B) for the complete bathymetry of Lake Tahoe with 1244896 water columns (5th row in Table 2.2). The Lake Tahoe bathymetry data used for the present study was obtained from USGS (Figure 2.12). The vertical resolution is set up as layers with variable depth increasing from 0.5 m near the top to nearly 10 m near the bottom. The time-step is 50s for the test A simulations (100m) and 10s for tests B and C (20m) for stability purposes.

In the simulations, the model was forced (input data) using surface heat and momentum fluxes estimated from local atmospheric variables (short and long wave radiation, air temperature, relative humidity, and wind speed and direction) obtained from meteorological data. These data were taken primarily from meteorological stations maintained by the Tahoe Environmental Research Center (TERC). There are ten shoreline and on-lake meteorological stations. All stations provide a near-continuous record of wind magnitude and direction and air temperature. The model was simulated from July 3rd 2008 (Julian Day 185) to August 2nd (Julian Day 215), with hourly output epochs.

2.4.4.1 Test A. Comparison of inner and outer models with the same resolution

The inner and outer model have been compared in the nested area (Figure 2.12 (right)), both with 100m square columns. The variables evaluated are horizontal velocities (u and v), vertical velocity (w), water surface elevation (ζ) and temperature (s). The *NRMSEs* (Eq. (2.7)) in top-layer averaged over time are 2.21%, 1.81%, 0.88%, 1.59% and 0.86% for u,v, w, ζ and s respectively, and similar or better in other layers (10th column in Table 2.3). Finally, the temporal *RMSE* (Eq. (2.8)) of the water volume ($(\Sigma\zeta) \times \Delta x \times \Delta y$) differences between outer-inner models is 0.22m^3 which represents 0.13% of the average volume over time measured in the nested area. No volume drift over time is observed in the simulation period.

2.4.4.2 *Test B. Comparison of the inner model within a low-resolution outer model with a high-resolution model of the complete basin.*

The inner and HR models, both with 20m square columns, have been compared in the nested domain (Figure 2.12 (right)) excluding the relaxation area. The *NRMSEs* (Eq. (2.7)) in the top layer averaged over the simulation time are 3.01%, 2.66%, 1.23%, 1.84% and 1.69% for u,v, w, ζ and s respectively, and similar or better in other layers (12th column in Table 2.3). The temporal RMSE for volume ($(\Sigma\zeta) \times \Delta x \times \Delta y$) is 0.34m^3 which represents 0.2% of the average volume over time measured in the nested area. These results indicate that the inner model yields similar results in the near-shore region to those of the HR basin model, but with a much lower computational cost. The test has been repeated using the OBC implementation in Figure 2.3(b), results worsen slightly (Table 2.3, clp-it).

2.4.4.3 *Test C. Comparison with low-resolution outer model*

The outer and inner results have been compared looking for local-scale hydrodynamic features well resolved by the inner model but not by the outer. Vorticity in Marla Bay is an example (Figure 2.13), where recirculation is likely to occur as a result of flow separation (Rueda and Vidal 2006). The vorticity field in Marla Bay at any given time t was computed from surface velocity predictions for the rectangular region in Figure 2.13 (study area), as follows

$$\begin{aligned}\omega(i+1/2\,,j+1/2) \\ &= \frac{v(i+1,j+1/2)-v(i,j+1/2)}{\Delta x} \\ &- \frac{u(i+1/2\,,j+1)-u(i+1/2\,,j)}{\Delta y}\end{aligned} \tag{2.12}$$

Although basin-scale features captured in both models are similar, sometimes the differences in the vorticity fields reveal the location of features in the inner model that are not captured by the outer model. Figure 2.13 shows an example of the velocity field in Marla Bay from Julian Day 207 in the simulated period. As shown in the graphs, the inner model simulates recirculation in Marla Bay, while the outer model only captures a weak divergence in the velocity field. Being able to simulate these eddies in bays and other lake shore irregularities, is important in trying to understand coastal transport processes (Rueda and Vidal 2006). As a result of re-circulating eddies, bays can trap particles in suspension and other water constituents, hence, decreasing the longshore dispersion rates. This trapping effect has been reported previously in the literature; for example, Brooks et al. (1999) shows that eddies in Cobscook Bay, Maine, could trap particulates in the side-arms of the estuary. The local residence time of water within bays tends to increase as a result of recirculating eddies; hence becoming hot-spots for the reproduction of species looking for quiet conditions. For example, high concentrations of juvenile fish in the center of a large eddy in the Santa Barbara Channel are observed in Nishimoto and Washburn (2002).

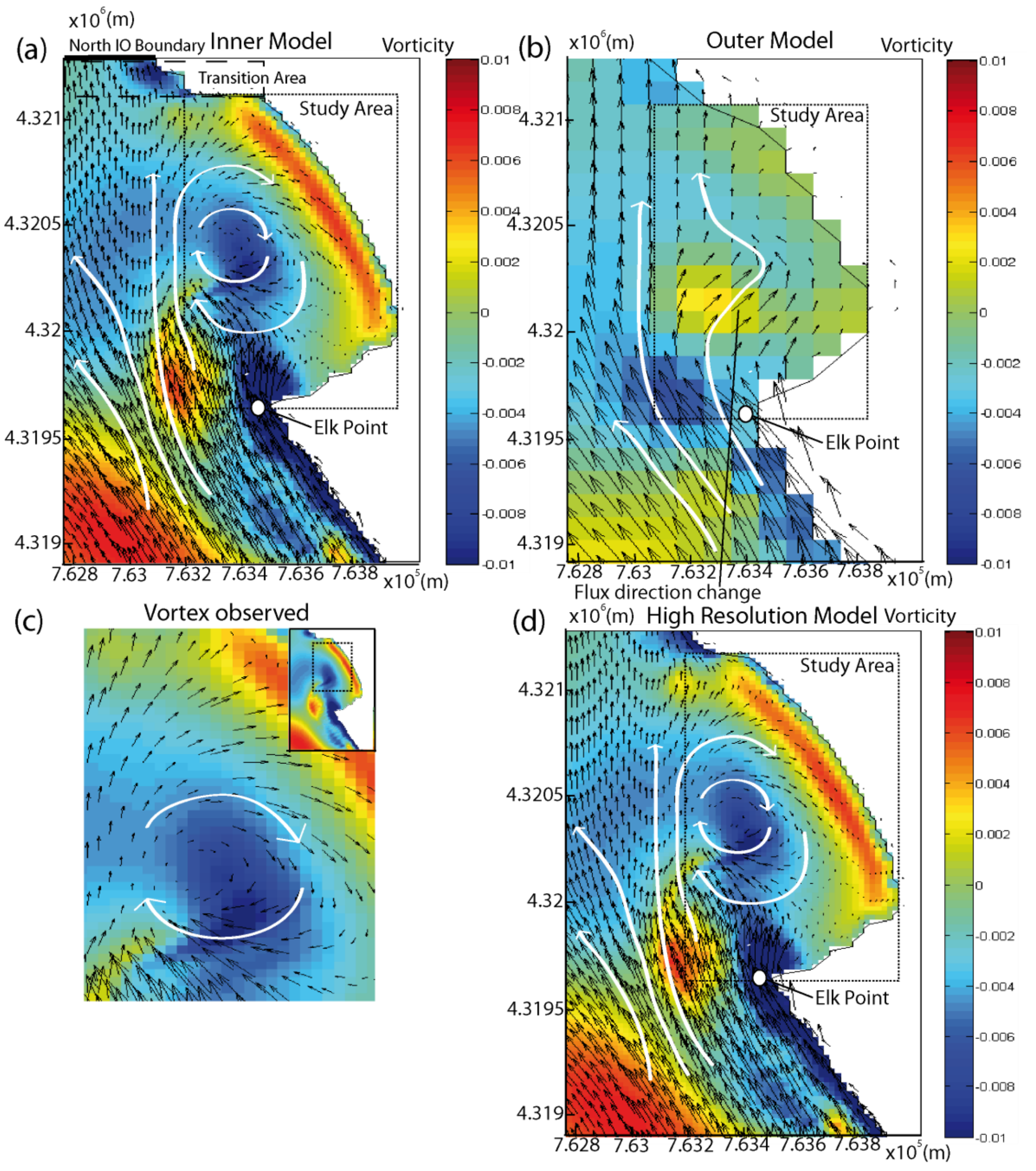


Figure 2.13. Lake Tahoe, test C: Vorticity in Marla Bay area at a snapshot in time on Day 207. Vorticity (color scale) and u+v velocity field (black arrows) for **(a)** inner model, **(b)** outer LR model and **(d)** HR basin model. **(c)** Zoom of the captured local-scale vortex.

Figure 2.14 shows the evolution of average surface vorticity in Marla Bay during a period of 18 days. Note that the flow field in Marla Bay tends to exhibit a negative vorticity, corresponding to a clockwise circulation (see Figure 2.14, Figure 2.13). The circulation strength changes with time, increasing in response to pulses of strong northward currents at Elk Point (Figure 2.14 (d)). These pulses, in turn, tend to respond to local wind variations (Figure 2.14 (b-c)), occurring with diurnal periodicity. Peak northward currents in Elk Point tend to occur after diurnal wind events, blowing predominantly from the north. The vorticity tends to be larger in magnitude in the inner model, partly due to its higher resolution, and partly due to the lower value of K_H used (Eq. 3.3), which depends on the grid resolution. Note, also, that the largest differences in vorticity between inner and outer model tend to occur when the current in Elk Point (and circulation in Marla Bay) is stronger.

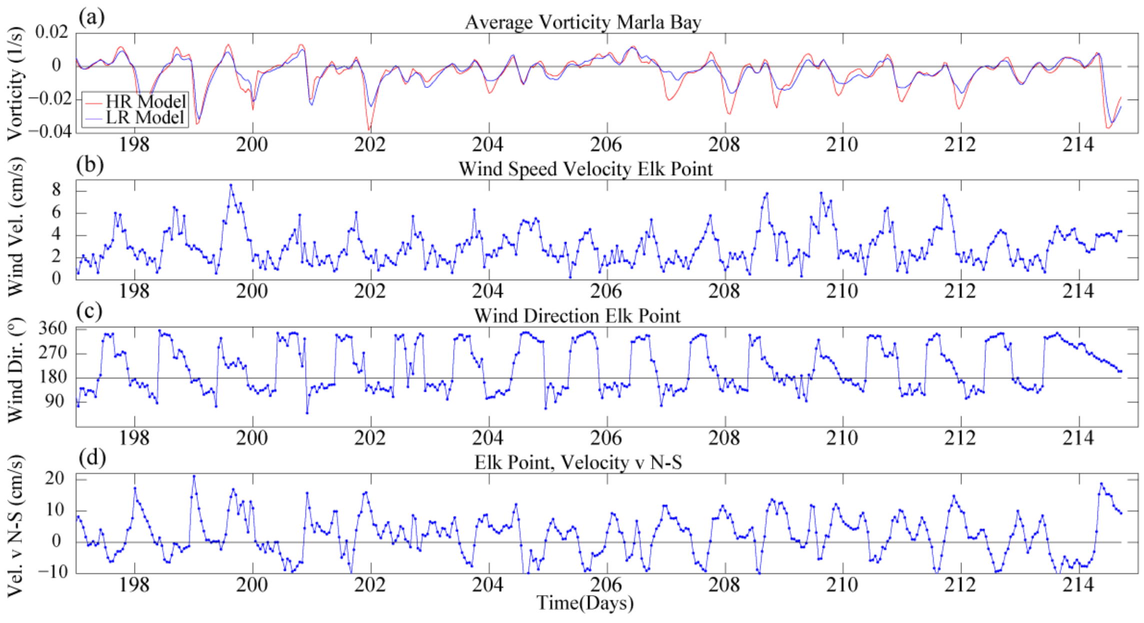


Figure 2.14. Lake Tahoe, test C: (a) evolution of vorticity calculated by HR model (red line) and LR model (blue line) in Marla Bay. (b) wind speed velocity (cm/s) in Elk Point. (c) wind direction in Elk Point (º, 360º=North) (d) *v* velocity (cm/s) in free surface in Elk Point.

2.4.5 Influence of tangential velocities in Lake Tahoe and Sacramento River.

The correct construction of I/O boundary is a fundamental task to get a correct coupling between outer and inner models. It is necessary to prevent a source of error that may impair the quality of the results of the nested model, and to avoid problems in the conservation of mass and volume. Figure 2.2 shows that the two velocity components (normal and tangential) are among the variables that must be communicated. Some nested implementations (Bonaventura and Rosatti 2002, Fox and Maskell 1995, Harris and Durran 2010, Zhai and Sheng 2008) use just the normal component in the construction of the I/O boundary, probably because tangential values for the simulated models are very small compared to other forces and their use as driving variables barely affects the results of lowly energetic environments, as is the case of the nested area simulated in Lake Tahoe. However, in highly energetic environments such as Sacramento River, characterized by high river discharges, river bending and with high values of lateral circulation and flow directional gradients in the nested area (a curve in the domain, Figure 2.9(right)), tangential velocities reach large values, being comparable to normal velocities. In this case, their absence in the I/O boundary construction can lead to errors in the nested model and a consequent loss of quality in the results.

To assess the importance of a complete communication of both velocity components in the construction of the I/O boundary in different models, test A was repeated not communicating the tangential velocity components this time (i.e. clamping to 0 the tangential velocities using a passive pure clamped OBC). The remaining variables are sent as in the previous case. In both real examples (Lake Tahoe and Sacramento River) *NRMSEs* averaged over time for the whole inner grid are greater when the tangential velocities are not transferred, as can be noticed comparing the 6th and 7th columns, and the 10th and 11th columns in Table 2.3, reaching maximum *NRMSEs* over time of an order of magnitude greater without tangential velocities for Sacramento River, as Table 2.5 shows. The differences are small in Lake Tahoe (Table 2.5, columns 2 and 3). However, in Sacramento River, differences are very important (Table 2.5, columns 4 and 5 and Figure 2.15).

Variable	Tahoe	Tahoe – clp0-t	Sacramento	Sacramento – clp0-t
Vel. ***u*** **(E-W)**	3.82	4.63	4.01	34.8
Vel. ***v*** **(N-S)**	3.17	3.74	2.21	17.5
Vel. ***w*** **(vert.)**	3.46	3.89	3.13	28.3
ζ **(wse)**	2.95	3.21	0.24	1.5
s (temp.)	1.89	2.04	Not measured	Not measured

Table 2.5. Test A. *NRMSEs* (%) in top-layer of nested area, maximum over time, for Lake Tahoe and Sacramento River, comparing the construction of I/O boundary using the tangential velocity components and without them (clp0-t, clamped to 0).

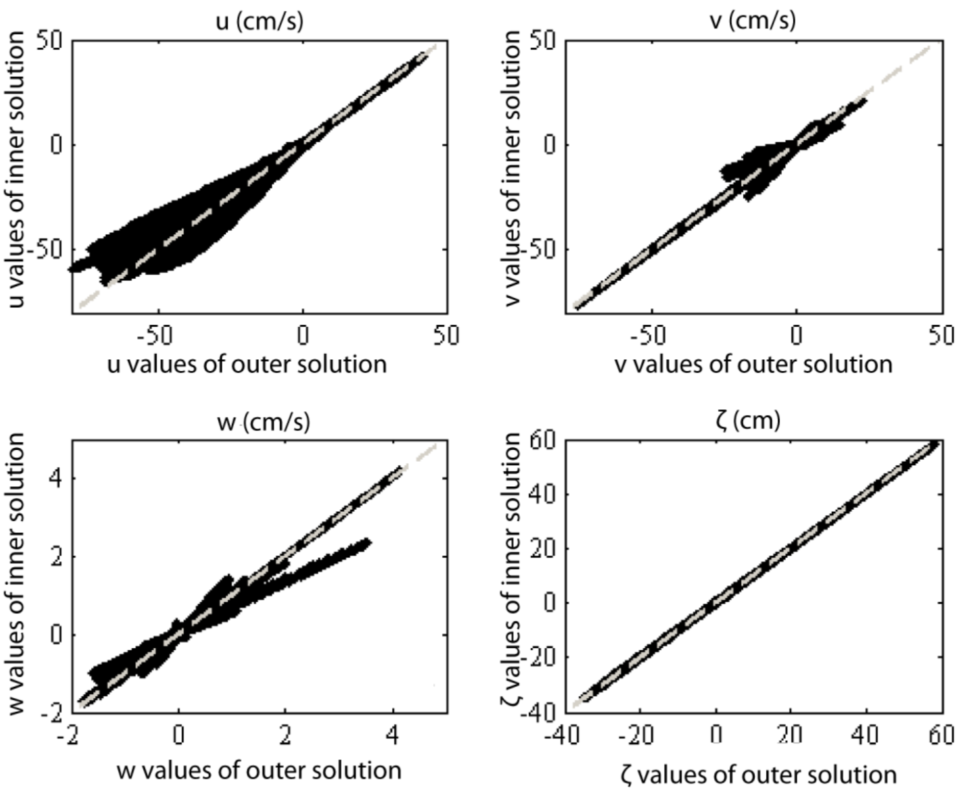


Figure 2.15. Sacramento River, test A, I/O boundary without tangential velocities: *u, v, w and ζ,* compared between inner and outer model solutions for surface layer at all output epochs. Each point represents the value obtained by inner model (y-axis) against the solution obtained by outer model (x-axis). If both solutions coincide, the point is located on the dashed line.

The magnitude of the error is directly related to the tangential velocities missing in Clarksburg Bend I/O boundaries, particularly the eastern one (Figure 2.9, Figure 2.15). The flow in Clarksburg Bend, and in Sacramento River in general, depends on the flood and ebb tides from the ocean. Near to the eastern I/O boundary, a pattern occurs with tidal periodicity. The flow is generally westward, peaking on low tides (negative *u* velocity, see Figure 2.16 (a) for discharge *Q* and *u* at time marks T4 and T8), reaching maximum *Q* and maximum negative normal and tangential velocities, *u* and *v*, on *lower low waters* (T4 in Figure 2.16 (a)). The eastern I/O boundary is aligned in the N-S direction, the river crosses it at a slight angle (turning west, Figure 2.9), and mean tangential *v* velocity in top layer in I/O boundary is a fraction (around 25%) of normal *u* velocity, negative too (southward).

Since tangential velocities are not being communicated, the inner model error is maximum just at this time (*lower low waters*) in the I/O boundary, and propagates from the boundary to the inner model (Figure 2.17 for T4). Figure 2.17 for T5-T6 show that the inner model error decreases with the magnitude of the missing tangential velocity *v*, until it becomes relatively not important (at *lower high waters*, T6), though velocities of around -2.5cm/s (around 20% of peak -12.5cm/s, see Figure 2.16 (a) for *v* at time marks T6, T4) are not being passed to the inner model. As the next low tide approaches (*higher low waters*, Figure 2.17 T7-T8), similar errors appear again in the inner solution, perhaps smaller than those of the *lower low waters* (compare *NRMSEs* for T4 and T8 in Figure 2.16 (b)).

On high tides, velocities approach to 0 (*lower high waters*, T6) or even become positive (*higher high waters*, T1 in Figure 2.16), which means negative discharge and, for the eastern I/O boundary, eastward flow (u positive, v ≈ 25% u positive too). In the southern boundary (aligned in the E-W direction) normal velocity *v* follows the same pattern while the tangential velocity *u* exhibits opposite sign (Figure 2.16 (a)). The river crosses the boundary at a larger angle (Figure 2.9) and *u* is a larger fraction of *v* (approximately -2/3, compared to v ≈ 0.25 u in the northern I/O boundary). Maximum tangential velocities at this boundary, over 25cm/s (Figure 2.16 for $u_{S\text{-}boundary}$ at T4 or any other mark but T1-T2) occur, however, when the flow is exiting the inner domain, so the error incurred by not communicating the tangential velocity *u* in the southern I/O boundary is much smaller than in the eastern I/O boundary (Figure 2.17).

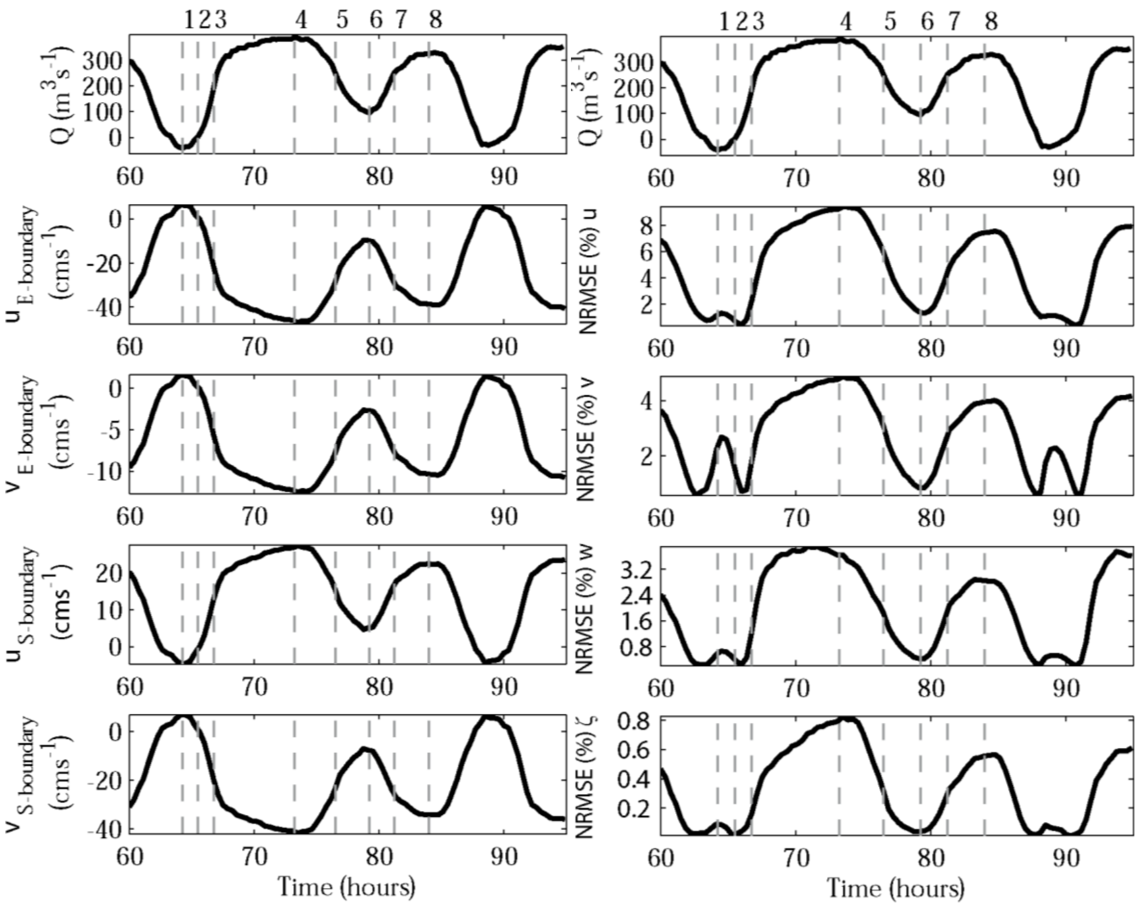


Figure 2.16. Sacramento River, Test A, I/O boundary with tangential velocities clamped to 0: (a) evolution of discharge (*Q*) and mean velocities in top-layer in eastern I/O boundary (*u* E-W normal, *v* N-S tangential) and in southern I/O boundary (*u* tangential, *v* normal), and (b) evolution of *Q* and *NRMSEs* for *u*, *v*, *w* and *ξ* in the top-layer of the nested area.

These results show that the error in the explicit source terms in the momentum equations, and the contribution of those terms in the momentum equations largely depend on the magnitude of the tangential components. The error in the momentum equations are particularly large in highly energetic environments such the test case of Sacramento River. The error, in this case, is mainly the result of the calculation of advective accelerations along the I/O boundaries (the terms in the inner model solution that use tangential driving variables are identified in Table 2.1).

The simulation time is imperceptibly affected by including tangential velocities (for example, it increases just a 0.6% in Lake Tahoe), so the model can be programmed to always use tangential velocities, freeing the researcher or end-user from deciding about it.

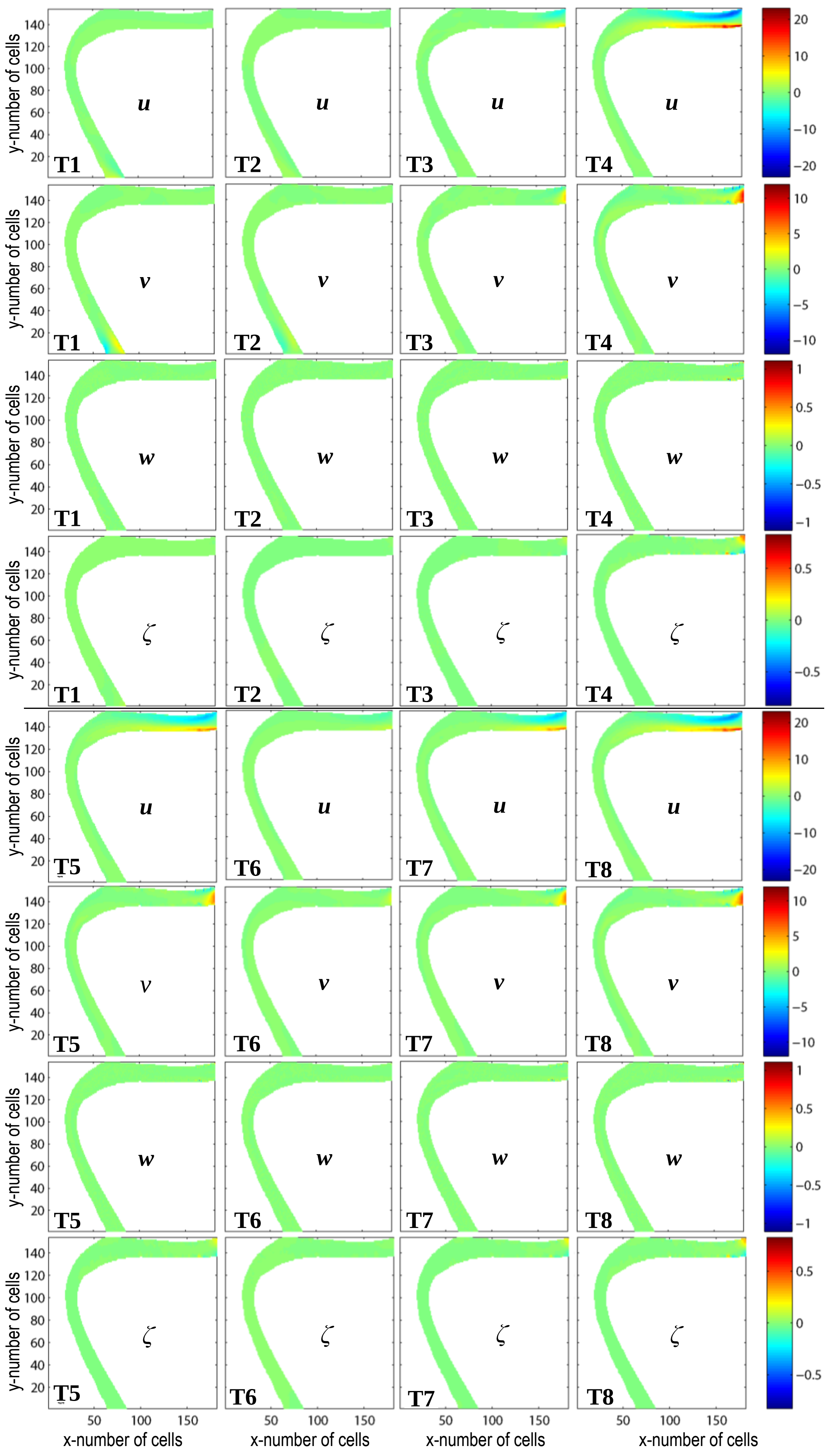


Figure 2.17. Sacramento River, Test A, tangential velocities clamped to 0: snapshots of NRMSEs for variables u, v, w, ξ for the different time marks in Figure 2.16: T1-T8.

2.4.6 N-Si3D performance evaluation

Performance (memory and time) improvement have also been analyzed. Two platforms have been used:

- A low-price and low-consumption *entry-level/volume* Intel® Xeon® CPU L5506 *processor* (4 cores, 2.13 GHz, 4MB last-level L3 cache, 4.80 GT/s Intel® QPI, low thermal design power 60W, Intel recommended customer price $423) in an entry-level/volume server with 16 GB of memory.
- A *midrange* Intel® Xeon® X7550 *processor* (8 cores, 2.00 GHz, Turbo Boost deactivated, 18M last-level L3 Cache, 6.40 GT/s Intel® QPI, low thermal design power 130 W, Intel recommended customer price $2837.00) included in a high-end shared-memory CC-NUMA server with 1TB of memory, of which 128 GB are local to the processor. This server has enabled the execution of some tests without nesting that required more than 16 GB of memory.

Performance improves significantly by both storing just columns with water and applying the nested implementation in both Sacramento River and Lake Tahoe. For the Sacramento River simulation of the 5m square cell basin model in the midrange processor, the memory requirements were reduced by approx. 95% and execution time by approx. 16% with our first Si3D-code improvement based mainly on avoiding dry cells storage. With this reduction in memory the entry-level processor can execute the application but not in real time (time per time-step is 4.84s>1s in the entry-level processor). By applying the online parallel nested implementation (outer-model 10m cells and inner-model 5m cells, see Table 2.2), the memory is reduced an additional 91% approx. and the sequential time is reduced an additional 91% approx. using the 4 cores (1 core for the inner and 3 cores for the outer model in order to balance the inner and outer time-step execution time, see Section 2.3.4, Figure 2.4) of the entry-level processor (the speedup is ~10). This reduction allows real time execution (0.45s per time-step).

For the Lake Tahoe simulation of the 100m square cell basin model in the entry-level processor, the memory requirements were reduced by approx. 29% and execution time by approx. 39% with our first Si3D-code improvement based mainly on avoiding dry cells storage. By applying the parallel nested implementation with outer-model 100m cells and inner-model 20m cells, memory reduction of approx. 88% and time reduction of approx. 98% (speedup of ~64) were measured in the high-end server using 3 cores (2 cores for the inner model and 1 core for the outer in order to balance the inner and outer time-step execution time). Performance figures cannot be offered for the entry-level processor because HR basin required more than 16GB; but by applying the parallel nested implementation, the lake can be simulated in the entry-level processor and in real time. These reductions enable the use of the low-priced and low-power consumption test computer in the research or predictions in Lake Tahoe. The execution time using 3 of the 4 cores of the entry-level processor (6.5s<10s, see time-step Table 2.2) is less than the execution time in 3 cores of the midrange processor (7.1s); clock frequencies are 2.13 GHz for the former and 2 GHz for the latter. Multiple nested grids of the lake can be simulated all at once in the in staff's personal computers, or in a low-cost cluster.

2.5 Conclusions

This chapter presents the verification, validation and performance evaluation of a nesting grid approach, N-Si3D, for 3D finite-difference semi-implicit hydrodynamic models with Cartesian grid, Si3D. The objective was to obtain an implementation with a good relation between quality of the results and execution time with low-cost low-power resources of computing.

The Cartesian grid memory requirements are drastically reduced by using nesting and a linear data representation that stores just information of columns with water. This made the implementation suitable for simulations of irregular domains, such as rivers.

The evaluation and validation of the test results show that:

- The nested approach proposed does not introduce errors when the inner and outer models have the same resolution (Tests A in Section 2.4.2). The errors in the nesting implementation are due to the iterative method for solving elevation, and to spatial interpolation and bathymetry mismatch (both consequence of the different inner and outer grid resolutions). When N-Si3D uses the clamped OBC implemented without relaxation, mass drift was observed but with a low slope. The 3D flux relaxation scheme implemented avoids mass and energy drift (Test B in Section 2.4.2.2).

- The implementation can model 3D physical processes that cannot be accurately simulated by a non-nested low-resolution model applied to the complete domain: horizontal recirculation in Lake Tahoe (Test C, Section 2.4.4.3) and vertical recirculation in Sacramento River (Test C, Section 2.4.3.3). Comparisons of high-resolution basin results with high-resolution inner results show similar ability to capture local-scale processes (Test C) and low NMRSE errors (Test B, Sections 2.4.3.2 and 2.4.4.2). Grid refinement ratios of $r_g=5$ may be necessary in order to simulate some local features, such as vertical recirculation in Sacramento River (Test C, Section 2.4.3.3).

- The inclusion of tangential velocities in the boundary conditions strongly affects the quality of results when very strong currents, lateral circulation and/or vortices exist, with negligible effect on computing time (Section 2.4.5). Maximum error percentages even of tens in Sacramento River have been obtained when they are not included. The simulations show the error is related to the momentum missed at the I/O boundary when tangential velocities are eliminated, in particular, the influence on the advective term.

- The online inner/outer dynamic coupling achieves a computing time equal to the inner execution time or the outer execution time (communication time does not affect) by executing the inner and outer model in parallel with a structure that resembles a pipeline processing (Sections 2.3.4 and 2.4.6). This online coupling (1) avoids the use of expensive storage resources to store the values transferred from outer to inner, and (2) makes feasible a transfer step from outer to inner equal to the inner time-step, avoiding errors from time interpolation and without affecting computing time. Comparing the simulation times of Si3D without storing dry cells (Section 2.3.5) and N-Si3D, speed-ups of ~10 (for $r_g=2$) and ~64 (for $r_g=5$) are achieved in Sacramento River and Lake Tahoe respectively using a processor with several cores. The improvement in memory and time allows the use of low-cost and low-power consumption processors in real time simulations and that multiple simulations can be run all at once on personal computers, or in a cluster of low-cost computers.

Chapter 3

Parallel implementation of a 3D semi-implicit hydrodynamic model. Simulation of the near-shore physical processes of a lake in a small commodity cluster

Abstract

The parallel implementation of a three-dimensional (3D) to simulate physical processes in lakes with Cartesian grid models in a small commodity cluster of three multi-core nodes is presented. The parallel program, P-Si3D, uses the three nodes in the cluster (by using the message passing standard MPI) and the four cores in a node (by using the shared memory standard OpenMP). This work analyzes the influence in performance of using different platform configurations, several workload distributions, several parallel implementations, and block-driven processing. The approach is implemented using the semi-implicit model presented in Chapter 1. Additionally, P-Si3D and N-Si3D, last one presented in Chapter 2, are here used to develop a high resolution model of the perimeter of Lake Tahoe (USA) in order to simulate the near-shore (i.e. small-scale) physical processes with the resolution required. The result lake model, P/N Si3D, is applied, for illustrative purposes, to conduct tracer transport simulations revealing the pulsating nature of along-shore transport processes in lakes, and the effect of bays and shoreline irregularities on long-shore transport.

3.1 Introduction

High Performance Computing is being increasingly demanded in water sciences to get detailed descriptions of the flow fields that develop in natural ecosystems within reasonable lengths of time. It has been through these detailed descriptions of the flow fields, obtained either by means of simulations conducted with three-dimensional (3D) numerical algorithms solving the governing equations of fluid motion, or through field observations collected with high-resolution experimental techniques that water scientists have gained, in the last years, some understanding of transport processes in natural lakes and reservoirs (Hodges et al. 2000, Rueda et al. 2003). This understanding, however, is still far from complete.

Lakes and reservoirs are complex ecosystems composed of several subsystems with distinct physical, chemical and biological characteristics (Schindler and Scheuerell 2002). Those contrasting characteristics are particularly evident when comparing the littoral and pelagic environments. The pelagic habitat of lakes is relatively homogeneous in the horizontal dimension, but can have substantial vertical heterogeneity associated with seasonal thermal and chemical stratification. Littoral habitats, in turn, can be substantially heterogeneous in both vertical and horizontal dimensions (Lodge et al. 1988). Physical conditions exhibit continuous and very dynamic changes, at short-time scales, as a result of strong hydrodynamic forcing and the weak inertia of shallow layers (Lodge et al. 1988). Given the extraordinarily variable dynamics of near-shore or littoral habitats, these are sites with large biodiversity (Vadeboncoeur et al. 2011) and critical habitats for many organisms in lakes (Kalff 2001). Littoral and pelagic habitats, however, cannot be understood in isolation, since they are tightly coupled through a wide range of physical and biological processes. Circulation and mixing in the near-shore regions, for example, move nutrients, heat, organic carbon, and other tracers across isobaths, from the lake edge to the inner-shelf, and vice-versa. Organisms with complex life cycles, such as benthic species, can use both habitats in different life stages. Larger organisms, such as fish, also migrate off and on-shore, hence, linking the dynamics of benthic and littoral habitats. Hence, there is a pressing need for understanding the spatial and temporal dynamics of near-shore areas, and their interaction with the pelagic habitats. Furthermore, beaches or bays along the near-shore areas cannot be understood in isolation from neighbor bays, given that they are tightly linked through physical processes. As reviewed by Rao and Schwab (2007), currents in the nearshore are largely aligned along isobaths, hence, creating strong physical links between the littoral zone along the whole perimeter of lakes.

In spite of their importance for lake ecosystem function, our understanding of near-shore habitats is poor (Kalff 2001). This is in part the result of the extraordinary variability of the physical environment, which makes the task of characterizing it challenging, at least through observations. In the last decades, though, considerable progress has been made in developing numerical transport and mixing models capable of resolving with reasonable accuracy and computational cost large-scale physical processes in lakes and reservoirs (Schwab et al. 1994; Hodges et al. 2000; Rueda and Schladow 2003). These models are based on the numerical solution of the shallow-water equations (SWE), a simplified form of the Reynolds averaged Navier-Stokes (Cushman-Roisin 1994). The choice of these governing equations is based on practical computational limits and a priori scaling analyses that justify their use in the description of large-scale flows. These models can potentially be used to simulate local-scale processes in the littoral zone such as circulation and mixing, and the exchange of water, suspended and dissolved constituents with the pelagic environment. However, these local near-shore simulations are not straightforward to carry out. First, they cannot be conducted in isolation from the basin scale, given that the physical dynamics of the littoral areas are driven by and interact with basin scale processes such as internal waves, wind waves, and the large-scale circulation. Second, and more importantly, these simulations need to be conducted on high-resolution grids so that the spatial scales characterizing circulation and transport patterns in the near-shore regions are correctly captured. Flow fields in the littoral zone are tightly linked to small-scale bathymetric variations, shoreline irregularities (such as headlands, islands and bays), river inflows and water withdrawals (Rueda and Vidal, 2009), which are poorly resolved by the low-resolution grids typically used in basin-scale simulations.

A plausible approach to capture local-scale processes using high resolution grids consists of the application of parallel computational techniques to the solution of the governing equations of motion in clusters. Several SWE models have been implemented in parallel that take advantage of their data parallelism. Implementations that use the message passing paradigm with MPI for both 2D (Rao 2004, with one and also two layers in Castro et al. 2006, Tubs and Tsai 2009, Nesteron 2010) and 3-D models (Nesterov 2010) can be found. The implementation of Tubs and Tsai (2009) parallelizes a 3-D lattice Boltzmann model using the shared memory paradigm with OpenMP. These MPI and OpenMP implementations use domain decomposition to divide the workload among processes or threads. Performance can also be increased using Streaming SIMD Extensions (SSE) instructions either explicitly (manually) or through libraries, or, alternatively, using GPUs. Dyk et al. (2009), for example, presents results of a SSE optimized implementation of a 2D SWE-model. Castro et al. (2008), in turn, solves a 2D SWE-model using the Intel Integrated Performance Primitives library. An implementation of a 2D SWE-model in several GPUs supporting CUDA programming toolkit is presented in Asunción et al. (2010).

In this chapter, a parallel implementations for small commodity clusters of a semi-implicit 3D hydrodynamic model, P-SI3D, is presented and evaluated. The parallel implementations of the 3D model combine both message passing (with MPI) and shared memory paradigms (with OpenMP). Implementations with redundant operations (workload overlapping) are compared to non-redundant implementations. Workload overlapping increases the number of operations and decreases communications. This work also analyzes the influence of different platform configurations (such as simultaneous multithreading, Intel SpeedStep and Turbo Mode technologies, and prefetching hardware), and different domain decompositions have on code performance. Different compiler optimization options and a block-driven processing implementation were also tested.

Our final goal here is to develop an efficient procedure to conduct high-resolution simulations of the continuous littoral fringe of a lake using a Cartesian grid model; the model of Lake Tahoe (USA) is implemented with that purpose. P-Si3D is used successfully to conduct simulations with low and mid resolution of Lake Tahoe in a small commodity cluster with three nodes. The mid resolution model is used to evaluate the parallel implementation. However, high-resolution models of Lake Tahoe are very expensive computationally and not possible to simulate in small commodity clusters. One alternative with a much lower computational cost and traditionally used with Cartesian grids is to use a nesting procedure. In a nesting implementation, a high-resolution HR near-shore model that resolves physical dynamics in the littoral zone, is embedded inside a low-resolution LR outer model that simulates the basin-scale processes (Fox et al. 1995; Zavatarelli et al. 2003). Still, the cost of high-resolution nested-grid simulations, evaluated in terms of computational time and memory requirements, can be high in large lakes where the littoral zone extends over tens or hundreds of kilometers, as Lake Tahoe. Those computational costs are unacceptable when the simulation models need to be used for water quality management to assess, for example, the risk of introduction and dispersion of invasive species (Hoyer et al. 2014) or to evaluate the environmental effects of large infrastructures (e.g. Rueda et al. 2009).

The parallel implementation (P-Si3D) presented and evaluated here is combined successfully with the nesting procedure (N-Si3D) presented in Chapter 2 in order to simulate local scale processes of Lake Tahoe, which are present in the littoral zone. With the pipeline structure used by the nesting procedure (see more information in Chapter 2), HR and LR models are solved in parallel by different sets of nodes in the cluster. Additionally, HR and LR model use P-Si3D to distribute the work using domain decomposition. The models constructed with this approach will be referred to as Parallel/Nested (or P/N-Si3D) models. P/N-Si3D is here applied to Lake Tahoeand used, for illustration purposes, to simulate the dispersion of storm-water from existing outfalls in the lake to neighboring beaches. The solution of the P/N-Si3D model is evaluated using the case example.

The chapter is organized as follows: Section 3.2 explains the parallel implementation P-Si3D for small commodity clusters. Section 3.3 evaluates P-Si3D performance and compares several parallel implementations and platform configurations. Section 3.4 explains the combination of P-Si3D and N-Si3D, P/N-Si3D, to simulate the littoral area of Lake Tahoe using a high resolution nested model. Section 3.5 evaluates and validates the simulation obtained using P/N-Si3D and discusses some experimental results. Finally, Section 3.6 summarizes conclusions.

3.2 P-Si3D implementation

The parallel implementation presented (P-Si3D, Figure 3.1) is applied here to the Semi-Implicit hydrodynamic model proposed by Smith (2006), Si3D (Chapter 1) and modified to take advantage of several basic optimizations and an improvement of the data structure (more details about the basic optimizations and the new data structure can be found in Chapter 5, this version is called Basic Si3D). Implicit or Semi-implicit schemes are used to avoid the strict time-step limitation due to Courant-Friedrich-Levy (CFL) condition founded in explicit schemes. On the other hand, an implicit or semi-implicit approach requires solving long systems of nonlinear equation for free surface elevation over the entire domain each time-step. These systems of equation are usually solved using iterative methods, Si3D uses a method widely used in the literature, the Preconditioned Conjugate Gradient (PCG). More details about the implementation of Si3D can be found in Chapter 1. P-Si3D is able to simulate in an acceptable execution time models with a computational cost that would be impossible in a sequential execution.

Several parallel Si3D versions have been implemented. The speedup achieved in a parallel implementation of Si3D in which OpenMP construct !$OMP PARALLEL DO-END PARALLEL DO is used to locate parallelism was 1.22 with the four cores of a processor. The performance has been increased when the programmer has also done explicitly these tasks: assign jobs to threads; create and destroy threads; communicate and synchronize threads. Moreover, several parallel versions have been implemented in order to compare implementations with redundant operations, which avoid communications and synchronizations (C/S), with non-redundant operation implementations. The results show that redundant operations improve the MPI version performance of Si3D but the OpenMP version increases performance when the redundant operations are reduced by adding some extra synchronization.

The parallel model (Figure 3.1) uses MPI to assign work to computers and OpenMP to assign work to cores. C1, C2, C3 and C4 are communications among computers through message-passing. S1, S2, S3 and S4 are the solver stages of Si3D (more details about the solver stages can be found in Chapter 1). In this implementation, each thread/process is assigned the task of performing the calculations of one of the sub-domains. The domain decomposition (Chan and Mathew 2008; or Passoni et al. 2001) is done prior to the start of the computations in such a way that (1) all sub-domains have similar numbers of wet cells; and (2) sub-domain data are stored in contiguous memory positions.

The boundaries between sub-domains are vertical surfaces. The computations done by different threads/processes are not entirely independent, given that they involve the calculation of horizontal gradients. P-Si3D uses overlapping and redundant calculation to reduce communications so the overlapping area is calculated in two neighbor subdomains. Using redundant computation, the communications between computers to solve the horizontal gradients dependences are needed only at the end of each iterative loop (C4, Figure 3.1). In the non-redundant MPI version there are additional data interchanges between processes: 9 in S1, 4 in S3 and 12 in S4. On the other hand, P-Si3D does not overlap subdomains among cores, since threads can use shared memory and synchronization to avoid explicit communications.

S2 is the only stage where the calculation is not done column by column of the model. In this stage, the system of nonlinear equations is solved using an iterative method known as Preconditioned Conjugate Gradient (PCG). A parallel implementation of PCG is not something trivial, requiring many reduction (many-to-one) communications to perform multiple dot-product operations and communications between neighbor subdomains to perform a matrix-vector multiplication (see for example Nesterov 2010). These operations occur multiple times each iteration of the PCG and, in general, in other iterative methods. Priory test concluded that S2 consumes a very little time in the total execution of Si3D (a 2% of the simulation time of Lake Tahoe using the mid-resolution model). Taking into account these results, the implementation of a sequential PCG into the parallel model to avoid the communications explained before seems reasonable, as long as the parallel version is executed in small commodity clusters. With a sequential PCG, only one collective gather (many-to-one, to collect all the equations of the system) communication before S2 (C2, Figure 3.1) and one collective scatter (one-to-many, to distribute the new solution of water free surface) communication after S2 (C3, Figure 3.1) are needed.

The overhead of the parallel implementations of P-Si3D, like in other related applications, is mainly affected by:

- Load unbalance. The irregular grid dimensions make difficult to obtain an even distribution.
- Communication time. It depends both on the number of communications and on the amount of data being transferred in each communication (between subdomains (C4) and collectives (C2 and C3)). In the data interchanges between processes, both of them depend on the domain decomposition approach used.
- Extra operations due to sub-domain overlapping. The number of communications can be reduced by overlapping sub-domains. In these overlapping regions, computations are redundant. The overhead that results from redundant calculations depends on the extent of overlapping regions, and this, in turn, depends on the particular domain decomposition approach used.

Therefore, domain decomposition affects performance, several approaches are possible in 3D models. Either horizontal-cut or vertical-cut (depth) decomposition can be applied in these cases. Horizontal-cut decomposition distributes layers among sub-domains, i.e. among processors/cores. The degree of parallelism in this case equals the number of layers and communication depends on the horizontal resolution and the horizontal extent of the lake. Given that large differences exist between the horizontal and vertical dimensions of large-scale geophysical systems, the degree of parallelism in the horizontal-cut decomposition tend to be lower than in a vertical-cut decomposition.

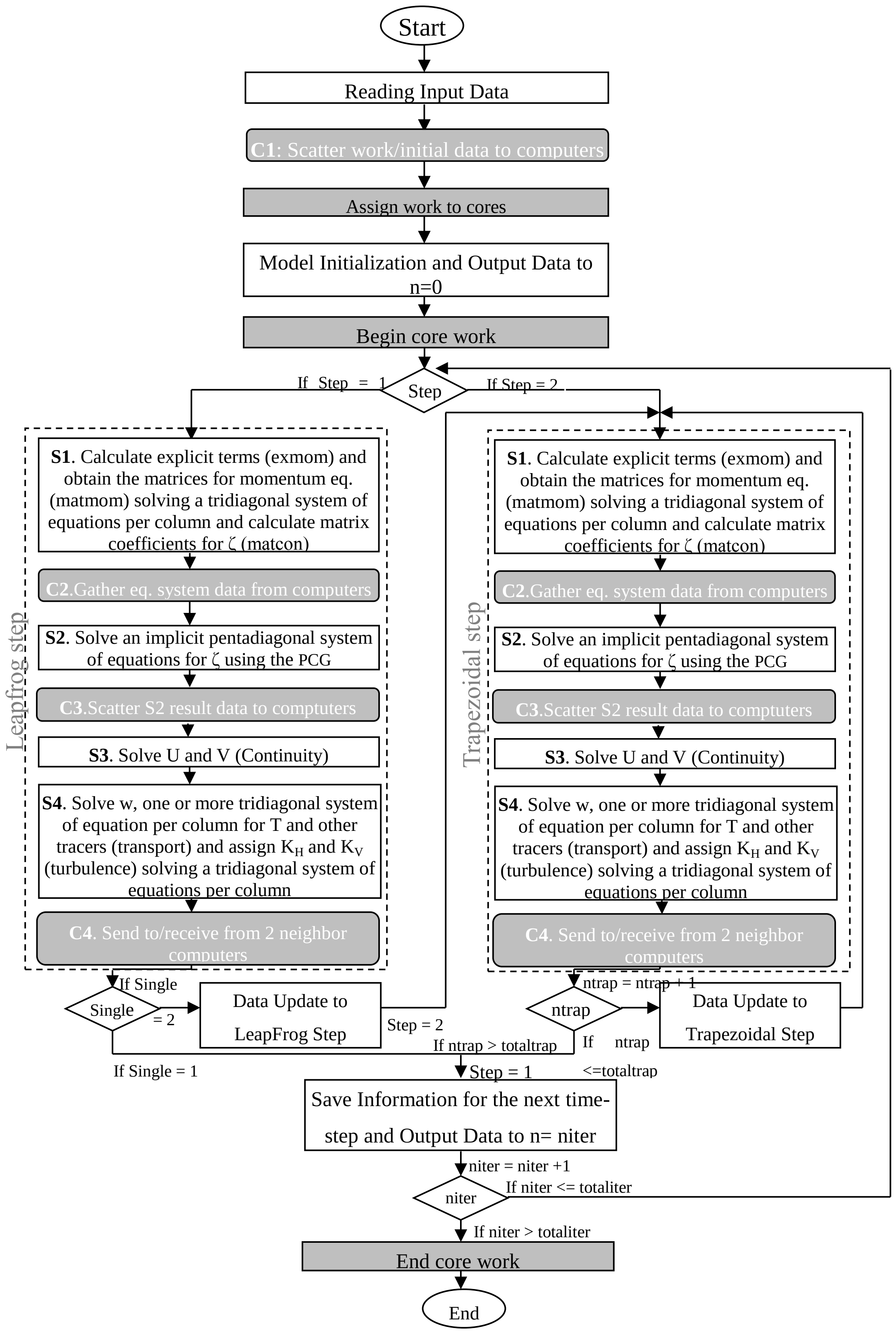

Start
Reading Input Data
C1: Scatter work/initial data to computers
Assign work to cores
Model Initialization and Output Data to n=0
Begin core work
If Step = 1
Step
If Step = 2
Leapfrog step
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ (matcon)
C2.Gather eq. system data from computers
S2. Solve an implicit pentadiagonal system of equations for ζ using the PCG
C3.Scatter S2 result data to comptuters
S3. Solve U and V (Continuity)
S4. Solve w, one or more tridiagonal system of equation per column for T and other tracers (transport) and assign K_H and K_V (turbulence) solving a tridiagonal system of equations per column
C4. Send to/receive from 2 neighbor computers
Trapezoidal step
If Single
Single
= 2
Data Update to LeapFrog Step
Step = 2
If Single = 1
ntrap = ntrap + 1
ntrap
Data Update to Trapezoidal Step
If ntrap > totaltrap
If ntrap <=totaltrap
Step = 1
Save Information for the next time-step and Output Data to n= niter
niter = niter +1
niter
If niter <= totaliter
If niter > totaliter
End core work
End

Figure 3.1. Flow diagram for the P-Si3D model. Grey Boxes represent the parallel implementation added to the hydrodynamic model. S1, S2, S3 and S4 are the solver stages of Si3D and C1, C2, C3 and C4 are communications among computers.

Three types of vertical-cut decomposition (of a river, lake, etc.) are possible (Figure 3.2). The data interchange between the sub-domains is indicated by the arrows in the Figure 3.2.

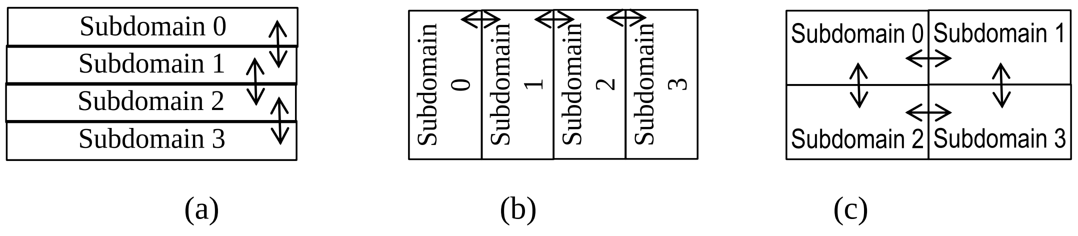


Figure 3.2. Three domain decomposition alternatives with vertical cut: (a) wide cut distribution, (b) narrow cut distribution, (c) two-direction cut distribution. Arrows show the communication needed among sub-domains in this kind of applications

The length of the boundaries between any two given sub-domains reflects the amount of data exchanged between them. It is also indicative of the amount of redundant calculations if the number of communications is reduced by overlapping sub-domains. The total length of the sub-domain boundaries will depend on the particular geometry of the water body being simulated, and on how the domain is partitioned among processes. The number of interchange communications is larger if one uses the *two-direction* cut distribution, as shown in Figure 3.2(c). The total amount of data exchanged among processes and the number of redundant calculations, though, could be less than in the other two distributions, depending on the particular geometry and number of sub-domains. With this distribution a process can both send to and receive from more than two processes. Both *narrow* (Figure 3.2(b)) and *wide* (Figure 3.2(a)) cut distributions have the same number of interchange communication operations. A process will send to and receive from just one or two processes. Larger amounts of data are exchanged and more redundant operations are done in the distribution shown in Figure 3.2(a) (wide-cut distribution). Nesterov (2010) compares the alternatives in Figure 3.2(b) and Figure 3.2(c) using MPI in a cluster of four AMD Opteron 2.2 GHz nodes (2 cores each) connected through Gigabit Ethernet. The results for different grid sizes show that better performances are achieved if the decomposition is done using narrow cut distribution compared to the two-direction distribution. Rao (2004) compares the alternatives in Figure 3.2(a) and Figure 3.2(b) using MPI in a CC-NUMA HP/Convex Exemplar X-Class (SPP2200) with 64 processors distributed in four hyper-nodes. The eight nodes of a hyper-node are connected through a network (switch) of 960 MB/s bandwidth in each link direction Messina et al. (1998) (the network of the

Exemplar is an implementation of the standard SCI). The results show that narrow-cut distribution reduces execution time compared to wide-cut distribution. Here some tests (Section 3.3.4) compare wide and narrow-cut distributions with both message passing and shared memory paradigms in SI3D. Note that the alternative in the Figure 3.2(c) has less data locality compared to the alternatives in (Figure 3.2(b)) and (Figure 3.2(a)). The data of a sub-domain in the wide and narrow distribution were stored in disk and memory in contiguous positions in order to improve locality. The lack of locality decreases performance, especially in shared memory implementations.

A block-driven processing approach was also tested as in the shared memory implementation in Tubs and Tsai (2009). Extra communication and block-driven implementation are also suitable in a process-level parallel implementation when the memory of the processing node is not enough for the application (Paglieri et al. 1997, Castro et al. 2006).

3.3 Performance evaluation of P-Si3D with different platform configurations and parallel alternatives

3.3.1 Platform

The results have been obtained in a small commodity cluster known as ACII of three nodes connected through a Gigabits Ethernet switch. Each node has 6 GB of memory and a Core i7 CPU 920 (launch date: fourth quarter of 2008). The Core i7 920 has four cores of 2.667 GHz (two threads per core if Hyper-Threading is active), L3 cache of 8MB shared by all the cores, and QuickPath of 4.8 GT/s. The cluster price was of 3,000 € (first quarter of 2009) approximately with all the components, including the cabinet. It runs Linux Fedora 10 (kernel 2.6.27.41). Cluster communication system has a bandwidth of 115 MB/s, near to the theoretic 125 MB/s.

The program is compiled using Intel Fortran 11.1 compiler. The OpenMP of this compiler is used for the shared memory implementation and MPICH-1.3 for the MPI message passing implementation. The source-code versions implemented were compiled using options that drive classic optimizations and vectorizations. Table 3.1 summarizes the optimization options checked. Similar execution times are obtained with O2 and O3. When the options ipo and/or SSE4.2 are added to O2 or O3 performance does not improve. PGO does not improve the execution time compared to a version with the same optimization options but without PGO. The executables used in this section have been obtained with O2 and openmp compiler options.

O2: inline expansion and cloning of functions, classical optimization (loop unrolling, constant and copy propagation, strength reduction, variable renaming, dead store elimination, global instruction scheduling and control speculation …) and vectorization (this tries to generate MMX, SSE, SSE2 instructions). O2 is the generally recommended optimization level for reducing execution time.

O3: O2 optimizations plus more aggressive optimizations, such as prefetching, scalar replacement to reduce memory references, and loop and memory access transformations.

ipo: multifile interprocedural optimization (this, for instance, allows inline expansion and cloning for calls to functions defined in separate files).

openmp: this option enables the parallelizer to generate multi-threaded code based on the OpenMP directives included by the programmer.

xSSE4.2 (architecture-specific optimization): this tries to generate MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1 and SSE4.2 instructions (vectorization) and can optimize for the Intel Core i7 processor family.

prof_gen and prof_use (Profile Guided Optimization or PGO): PGO allows optimization by taking into account real benchmark data instead of heuristic data.

Table 3.1. Optimization options (Intel C compiler 11.1).

3.3.2 Test model for Lake Tahoe

The test application is a simulation of the transport model in Lake Tahoe. The ultimate goal of these simulations is the risk of introduction and dispersion of invasive species (Hoyer et al. 2014) or to evaluate the environmental effects of large infrastructures (e.g. Rueda et al. 2009). Given that O(10^2) m (hundreds) features of the velocity fields, characteristics of nearshore regions, should be resolved in Lake Tahoe, the computational grid cells should have horizontal dimensions of at least O(10) m (tens). Table 3.2 shows the details of the models for Lake Tahoe developed. Simulating a lake of the size of Lake Tahoe (roughly 20 km x 30 km) with O(10)m horizontal size cell columns, poses a serious computational problem which can only be addressed through the use of parallel computers. For example, the ratio of real to computational time in simulations conducted with 50 m wide grid cells (a mid-resolution among the models implemented) in a single core of the cluster is approximately 1:1. Moreover, the complete high-resolution (HR) model, which is implemented using horizontal square columns of 20x20m, was impossible to simulate in the small commodity cluster used here because of the requirements of memory, needing at least 16 GB to fit the variable information of the model. On the other hand, two complete models for Lake Tahoe are conduced successfully using the parallel implementation in an acceptable execution time. These models are the mid-resolution (MR) model and the low-resolution (LR) model. The MR model is implemented with horizontal square columns of 50x50m and a time-step of 25s, this model is used to evaluate the parallel implementation developed in small commodity clusters. The LR model is implemented with horizontal square columns of 100x100m and a time-step of 10s, this model is used to create a high-resolution near-shore model for Lake Tahoe as it is explained in Section 3.4. The time-step could be up to 50s without CFL limitations in the LR model. However, it is set to the same time-step of the HR nested model to avoid additional temporal interpolation errors using the nesting procedure (more information in Chapter 2). The vertical resolution of the grids was the same in all the models (HR nested, MR and LR models), changing progressively from 0.5 m near the surface to nearly 10 m near the bottom. The bathymetry data (Gardner et al. 1998) was downloaded (http://tahoe.usgs.gov/bath.html) and corrected in the near-shore region in the southern shore.

Models of Lake Tahoe	**Horizontal cell side**	**Columns of water**	**Total cells**	**Time-step**
Basin HR model	20x20m	1,244,896	94,691,170	10s
Nested HR model	20x20m	493,317	21,181,918	10s
Basin MR model	50x50m	197,781	14,654,639	25s
Basin LR model	100x100m	50,383	3,657,268	50s/10s

Table 3.2. Computational data of Lake Tahoe simulations: Horizontal resolution (square columns), total number of water columns, total number of cells and time-step used.

3.3.3 Performance of different platform configurations

This work analyzes the influence in performance of the multiple cores in a node, the prefetching hardware, the Intel Hyper-Threading technology, and the Intel SpeedStep and Turbo Mode technology. These simulations are conducted using the MR model of Lake Tahoe (Table 3.2).

Hardware prefetcher monitors data access patterns and prefetches data automatically into processor caches. Core i7 cores can track 16 forward streams and 4 backward streams each. Simultaneous multithreading allows the execution of multiple threads in a core; in particular, two threads with Intel Hyper-Threading. Intel SpeedStepTechnology allows the operating system to control the core speed. Intel Turbo Mode Technology allows processor cores to run faster than the assigned frequency under specific conditions.

Table 3.3 shows the seconds per iteration obtained for different platform configurations and different number of processes and threads. The narrow-direction distribution and the MPI redundant operation version have been used. The communication time due to data distribution or collection is not included because it does not depend on the number of iterations. Up to four threads are used to each node; a higher number of threads makes performance worst despite of Hyper-Threading being enabled. The column HSTP shows the results for the default configuration. In the default configuration the BIOS and the operating system have enabled Hyper-Threading (H), SpeedSteep (S) and Turbo Mode (T), and prefeching hardware (P). In particular, *ondemand* is the default *CPUfreq governor* of the cluster operating system, which means the governor sets the frequency depending on the current usage, between a minimum of 1.6 GHz and a maximum of 2.667 GHz, last one can increase due to Turbo Mode. The time in the default configuration is less reproducible due to the thread distribution of the operating system among the eight logical cores of a node. If Hyper-Threading is disabled (column -STP) performance improves, but if either SpeedStep/Turbo Mode (column ---P) or Prefetching (column -ST-) are also disabled, time increases slightly. The results in the columns -STP and ---P show an increment in the clock frequency due to the Turbo Mode. The results in the columns -STP and -ST- suggest that the prefetching hardware is being weakly used.

Block-driven processing was added to try to reduce cache miss by facilitating data locality. It reduces the execution time by 4% with horizontal cell size of 100 m x 100 m and one process with four threads. Block processing improves only marginally this implementation's performance, although it never makes performance worst as it was observed in the block processing implementation of Tubs and Tsai (2009). The results in Tubs and Tsai (2009) are obtained in a platform of IBM with Power5+ 1.9 GHz. For a grid of 1024x1024x10 (=10,485,760 cells), from 1 to 8 processors block-driven processing makes performance worst but from 12 to 16, the maximum number of processors tested, the block-driven implementation improves performance Tubs and Tsai (2009).

Tahoe MR model		Platform configurations			
No. Processes	**No. Threads**	**HSTP**	**-STP**	**---P**	**-ST-**
1	1	21.1	**21.15**	21.92	22.54
1	2	11.07	11.1	11.56	11.79
1	3	7.77	**7.73**	8.07	8.22
1	4	9.75	**6.12**	6.12	6.51
2	1	10.96	10.96	11.45	11.65
2	2	5.96	6.05	6.21	6.38
2	3	6.75	**4.27**	4.48	4.56
2	4	5.15	**3.42**	3.57	3.68
3	1	7.62	**7.65**	7.9	8.04
3	2	4.23	**4.26**	4.4	4.48
3	3	4.81	**3.1**	3.21	3.29
3	4	3.65	**2.51**	2.68	2.71

Table 3.3. Performance of different platform configurations (seconds per iteration). In HSTP, H means Hyper-Threading enable, S means SpeedStep enable, T means Turbo enable, and P mean Prefeching enable. " -" means Disable.

The results presented in the next subsections are obtained with the configuration –STP and *ondemand* as the *CPUfreq governor.*

3.3.4 Comparison of wide-direction and narrow-direction distributions in both MPI versions, with and without redundant operations

Both, wide-direction and narrow-direction distributions, have the same number of communication in both MPI versions, but they are of different sizes. Also, in the MPI version with redundant operations, the wide-direction distribution has more redundant operations than the narrow-direction approach (because it has larger border length).

Table 3.4 shows the execution time per iteration and speedup for both wide and narrow-direction distribution and both the MPI implementation with redundant operations (R) and the MPI approach with non-redundant operations (NR). These simulations are conducted using the MR model of Lake Tahoe (Table 3.2). As can be observed narrow-cut distribution also improves sequential execution time. The best approach is to use the MPI implementation with redundant operations and the narrow-cut distribution. Speedup improves more with the narrow-cut approach because this approach has lesser border length than the wide-cut approach; the border size is decreased a 25% approximately.

Tahoe 50m		Sec./iteration				Speedup			
No. Pr.	**No. Th**	**Narrow**		**Wide**		**Narrow**		**Wide**	
		R	**NR**	**R**	**NR**	**R**	**NR**	**R**	**NR**
1	1	21.1	21.1	21.32	21.32	1	1	1	1
1	2	11.1	11.1	11.34	11.34	1.9	1.9	1.88	1.88
1	3	7.73	7.73	7.97	7.97	2.73	2.73	2.68	2.68
1	4	6.12	6.12	6.29	6.29	3.45	3.45	3.39	3.39
2	1	**10.96**	11.18	11.21	11.51	**1.93**	1.89	1.9	1.85
2	2	**6.05**	6.21	6.19	6.43	**3.49**	3.41	3.44	3.32
2	3	**4.27**	4.41	4.51	4.76	**4.94**	4.8	4.73	4.48
2	4	**3.42**	3.58	3.61	3.79	**6.17**	5.91	5.91	5.63
3	1	**7.65**	7.85	7.88	8	**2.76**	2.69	2.71	2.67
3	2	**4.26**	4.48	4.43	4.66	**4.95**	4.72	4.81	4.58
3	3	**3.1**	3.31	3.34	3.52	**6.81**	6.39	6.38	6.06
3	4	**2.51**	2.69	2.77	3.02	**8.41**	7.86	7.7	7.06

Table 3.4. Wide and narrow distributions in both MPI versions: with (R) and without (NR) redundant operations.

3.4 *Simulation of Lake Tahoe small-scale processes in small commodity clusters by using P-Si3D and N-Si3D*

3.4.1 *A high-resolution near-shore model for Lake Tahoe*

P-Si3D and the nesting procedure explained in Chapter 2 (N-Si3D) are combined to take advantage of small commodity clusters of computers in the simulation of small scale processes in a lake. It couples both, the parallel implementation and a nesting algorithm in an efficient parallel method, using processes (MPI) among the available nodes and threads (OpenMP) among the multiples cores in each node. It solves the 3D-SWEs on a HR grid of the littoral perimeter of the lake (Figure 3.3, (nested HR model in Table 3.2)), subject to velocity boundary conditions (HR-LR boundary) which are taken from the LR lake model of the basin-scale hydrodynamics (basin LR model in Table 3.2). HR and LR models are solved in parallel by different sets of nodes in the cluster. In the small commodity cluster used, the LR model (with a smaller computational cost) is set in one core of the first node. At the same time, the HR nested model is set in the three nodes, using P-Si3D to divide the workload among processes and threads, using 3 cores in the first node and 4 cores in the other 2 nodes. The communications from the LR model to the HR nested model use MPI too. This configuration could be changed in other clusters with different number of nodes, using P-Si3D to divide the workload of the LR model or the HR nested model with a different configuration. Besides, the 3D-SWEs in the LR model or in the HR nested model are executed in parallel at the same time using a pipeline structure explained in detail in Chapter 2. Models constructed with this approach will be referred to as Parallel-Nested (or P/N-Si3D) models. The use of P/N-Si3D allows the simulation in the small commodity cluster (ACII) of the near-shore of Lake Tahoe with enough resolution to observe small-scale processes and with a total execution time (including both the LR and HR model execution) enough so that the simulation can be possible (7.36 seconds per time-step).

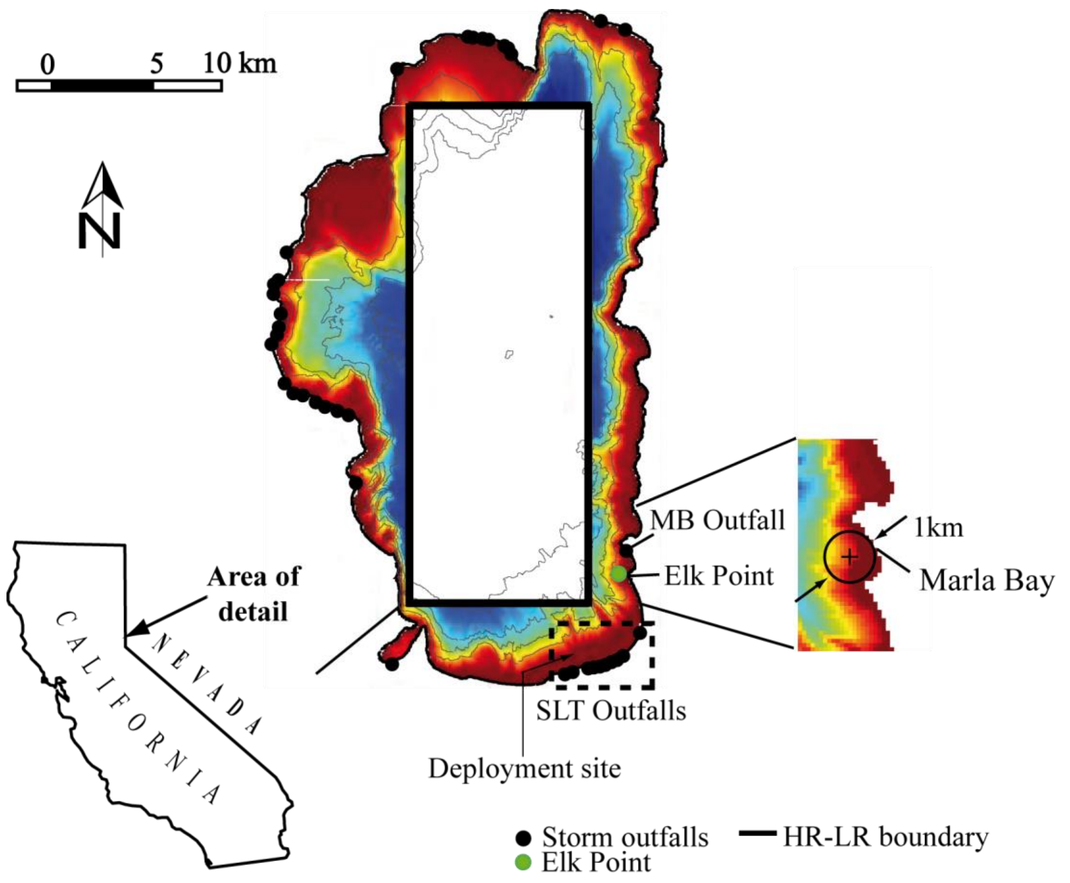


Figure 3.3. Lake Tahoe between California and Nevada in USA. The littoral zone is computed in high-resolution with communications from the low resolution model (HR-LR, solid line). Storm outfalls known around Lake Tahoe (http://tahoepipeclub.com/uploads/tahoe_pipe_list.pdf) and the particular areas studied (Marla Bay and South Lake Tahoe (SLT)) are shown too.

Therefore, P/N-Si3D was used to simulate the near-shore circulation and transport during the period in 2008 from day 191 (July 3) to day 215 (August 1st). Our focus is on two sub-periods characterized by different intensities of wind forcing. From day 192 to 197 (Figure 3.4, sub-period 1) the wind was weak with an average magnitude of approximately 2 ms^{-1}. From day 207 to day 212 (Figure 3.4, sub-period 2) winds were higher with magnitudes of up to 7 ms^{-1}. Flow features in the near-shore exhibit spatial scales on the same order of magnitude as the bays or other shoreline irregularities existing around the lake (see Rueda and Vidal 2009). In the case of Lake Tahoe, the length scales of these bays are of O (10^2) m (see Figure 3.3, Marla Bay). To resolve adequately O(10^2) m features, in turn, the model will need to accommodate at least 2 and preferably 4 computational cells in ca. 100 m, hence, the grid resolution needs to be ca. 20 m. Hence, a HR nested model of the littoral perimeter of Lake Tahoe was constructed using a 20 m grid resolution, and driven with the 100-m grid LR basin-scale model.

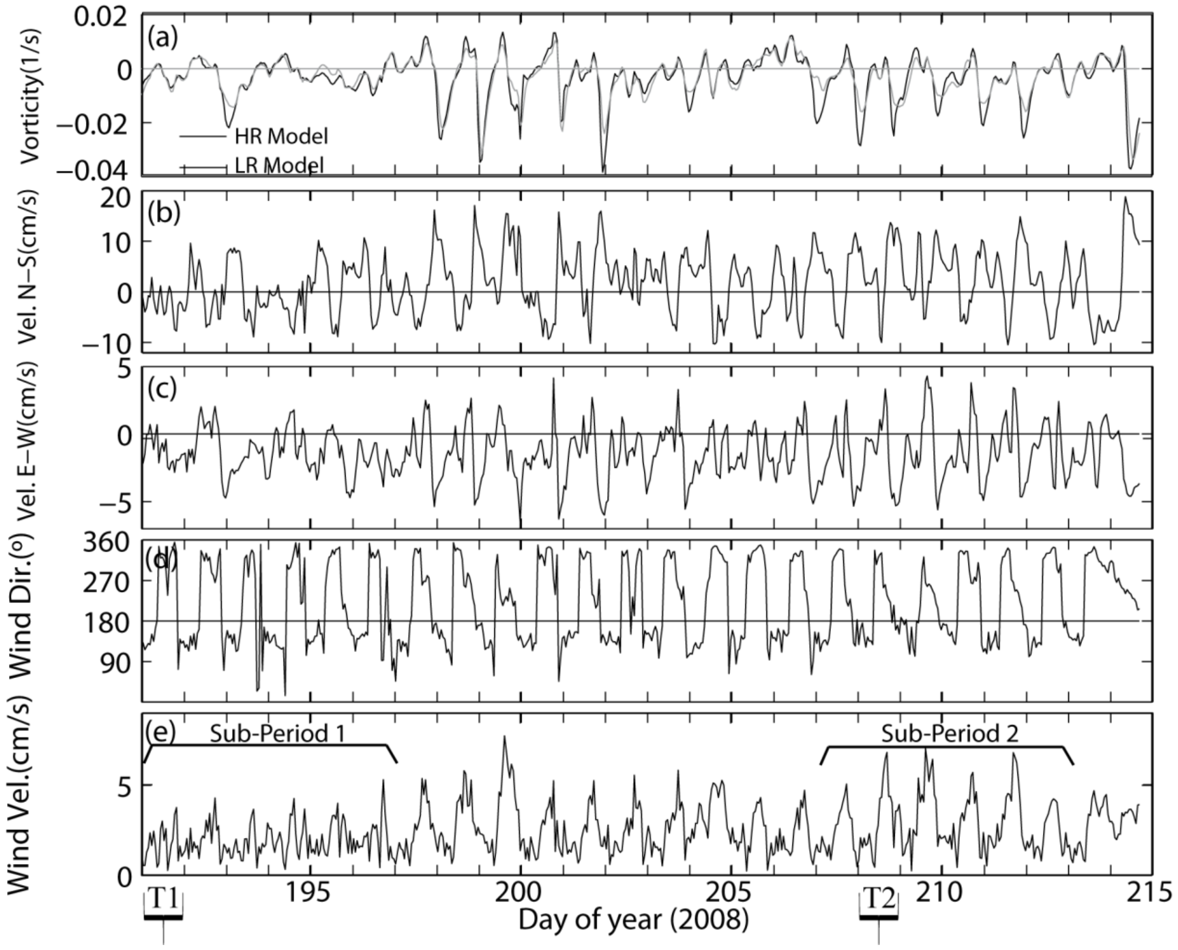


Figure 3.4. Period of time (study time) simulated in 2008. (a) Vorticity calculated for HR model (black line) and LR model (grey line) along the time in Marla Bay. (b) *v* velocity (North-South) in free surface in Elk Point. (c) *u* velocity (East-West) in free surface in Elk Point. (d) Averaged wind direction in the Southeast coast of Lake Tahoe. (e) Averaged wind speed velocity in the Southeast coast of Lake Tahoe.

The model run was initiated six days before the beginning of the study period, so that the simulation results were free from the effect of artificial initial conditions. The lake was assumed initially quiescent with horizontal isotherms. The temperature in the initial conditions was assumed to vary vertically following a temperature profile collected at the lake center (see Hoyer et al. 2014). The model was forced using surface heat and momentum fluxes estimated from local atmospheric variables (short and long wave radiation, air temperature, relative humidity, and wind speed and direction). Meteorological data were taken primarily from meteorological stations maintained by the Tahoe Environmental Research Center (TERC). In total, there are ten onshore and on-lake meteorological stations. All stations provide a near-continuous record of wind magnitude and direction and air temperature. Those records were averaged in space to the grid points using an iterative method originally proposed by Barnes (1964) (see also Barnes 1994a, Barnes1994b, Koch et al. 1983). The bottom drag coefficient was set to 0.02, following Rueda et al (2003). The horizontal eddy diffusivity K_h was set to a constant value, varying depending on the grid resolution. We used the empirical equation proposed by Lawrence et al. (1995) for lakes, between dispersion coefficient and the length scale of tracer clouds, to link the value of K_h with grid size. The value of K_h was set to 0.0086 m^2s^{-1} in the HR grid models while 0.05 m^2s^{-1} in the LR grid model. Different number of nodes *nn*, varying from one to eight, was used in the simulations depending on the goal of the runs. All cores were used in each of the nodes employed in the simulation.

Two forms of validation of the simulations conducted with the P/N-Si3D model were used. First, we used the differences in the solution of the governing equations (U, V, T, K_z and ζ) in the near-shore region of Lake Tahoe (Figure 3.3) calculated with the littoral HR (20 x 20 m grid) included in the P/N-Si3D simulation and with a basin-scale HR model (the basin HR model in Table 3.2). The solutions were output and compared every hour, and the comparison was done layer by layer. The results, shown here, are differences evaluated for the surface layer, two layers located at a depth of 10 and 20 m, and one more set of cells occupying the bottom of all water columns in the near-shore. The differences between two fields at any given time were quantified using a normalized form of the root-mean-squared error (NRMSE), calculated as follows

$$RMSE = \sqrt{\frac{\sum_{l=1}^{N}(x_b(l) - x_n(l))^2}{N}} \tag{3.1}$$

$$NRMSE = \frac{RMSE}{x_{\max} - x_{\min}} \tag{3.2}$$

Here x_b represents the value of a variable calculated by the HR basin-scale model; x_n is the value calculated by the HR nested model; $x_{\max}$ and $x_{\min}$ represent the maximum and minimum of each variable; and N is the total number of water columns in the near-shore domain. Second, we compared the HR near-shore simulations, constructed with the P/N-Si3D procedures outlined above, with velocity observations collected at a site close to South Lake Tahoe, ca. 1000 meters offshore, with a depth of 5-7 m (see Figure 3.3). The observations consist of near-continuous profiles of current magnitude and direction in 0.50 meter vertical bins collected with a NORTEK Acoustic Doppler Wave and Current Profiler (ADCP). The differences between observations and simulations are quantified along the period with experimental data available using a temporal root-mean-squared error (RMSE$_t$) defined as follow:

$$\mathrm{RMSE}_t = \sqrt{\frac{\sum_{t=1}^{N_{\Delta t}} (v_1^t - v_2^t)^2}{N_{\Delta t}}} \tag{3.3}$$

3.4.2 Application of the near-shore model to case studies in Lake Tahoe

The velocity fields predicted by the P/N-Si3D model during the study period were used to drive a series of tracer transport simulations. The simulations consist of the release of pulses of tracers from different sites around Lake Tahoe where storm water outfalls are known to exist – see Figure 3.3. They are intended to represent the fate of storm water contaminants entering the lake through outfalls. Tracers are assumed conservative and are released at a rate R_0 = 1 kg s^{-1} over a period of 24h. Two release periods were simulated: one, on day 191 (T1), under weak forcing conditions, and a second on day 208 (T2), during a period of persistently strong winds. The outcome of the simulations was a time series of tracer concentration evaluated in the areas with a depth less than 1.6 m of Marla Bay and South Lake Tahoe (SLT). Our interest in Marla Bay arises from recent work conducted to understand the dispersion of an invasive species from these sites (Hoyer et al. 2014). Furthermore, they represent features where the HR model could potentially provide details of the flow field, unresolved in the LR model (see Chapter 2, Section 2.4.4), given the spatial scales of the bays of O(10^2) m. Through tracer fields and the time series of tracer concentrations we assess the importance of alongshore transport.

3.5 *Evaluation of the Tahoe P/N-Si3D model*

Time averaged and maximum normalized differences between state variables computed during the simulation period with the HR-nested and the HR-basin models in different layers of the lake perimeter are shown in Table 3.5.

	Surface			**10m**			**20m**			**Bottom**		
NRMSE(%)	Av.	Max	Min	Av.	Max	Min	Av.	Max	Min	Av.	Max	Min
U	3.1	4.21	0.16	2.94	4.04	0.16	2.89	4.12	0.14	3.27	4.33	0.17
V	2.94	4.01	0.14	2.82	3.92	0.13	2.97	4.14	0.15	3.01	4.18	0.15
T	1.43	3.24	0.11	1.16	2.98	0.11	1.25	3.12	0.09	1.29	3.06	0.11
$\boldsymbol{K_v}$	0.93	2.46	0.12	0.96	2.44	0.13	0.89	2.36	0.12	0.92	2.27	0.11
$\boldsymbol{\zeta}$	2.61	4.9	0.17									

Table 3.5. Average, maximum and minimum NRMSE (%) between HR-nested and HR-basin models in surface layer, layer at 10m, layer at 20m and bottom layer for U, V, T, K_v, and ζ variables.

The time-averaged normalized differences are 3.10% and 2.94% for *U* and *V* respectively, 1.43% for temperature (*T*), 0.93% for vertical diffusivity (K_v) and 2.61% for ζ. Note that the averaged differences (NRMSE) were in all cases < 4%. The largest differences (of ca. 4%) were for the velocity and the free surface elevation. The lowest error norms (< 1.5%) correspond to water temperatures and turbulent diffusivity. These results indicate that the HR nested model yield similar solutions in the near-shore regions to those of the HR-basin model using P-Si3D to reduce the computational cost and N-Si3D to reduce the extension of the grid. As demonstrated in Chapter 2 (Section 2.4.2) the differences are, at least, partly due to the iterative (hence, approximate) nature of the matrix solver used to calculate the free surface elevation. The normalized differences between nested and basin-scale HR models (not shown) do not follow any significant trend in time, and only exhibit random variations. When compared against the velocity observations, we note that the P/N-Si3D model provides an accurate representation of the horizontal velocity records collected at the deployment site (ADCP, Figure 3.5), with temporal errors (Eq. (3.3)) of 4.12 x 10^{-2} ms^{-1} for the E-W velocity component and 2.61 x 10^{-2} ms^{-1} for the N-S velocity component. Other works in the literature have reported similar results. Jin et al. (2000) found errors between velocity measurements and simulation results from 1.52 to 4.76 x 10^{-2} ms^{-1} and Rueda and Schadow (2003) showed errors from 2 to 5 x 10^{-2} ms^{-1}.

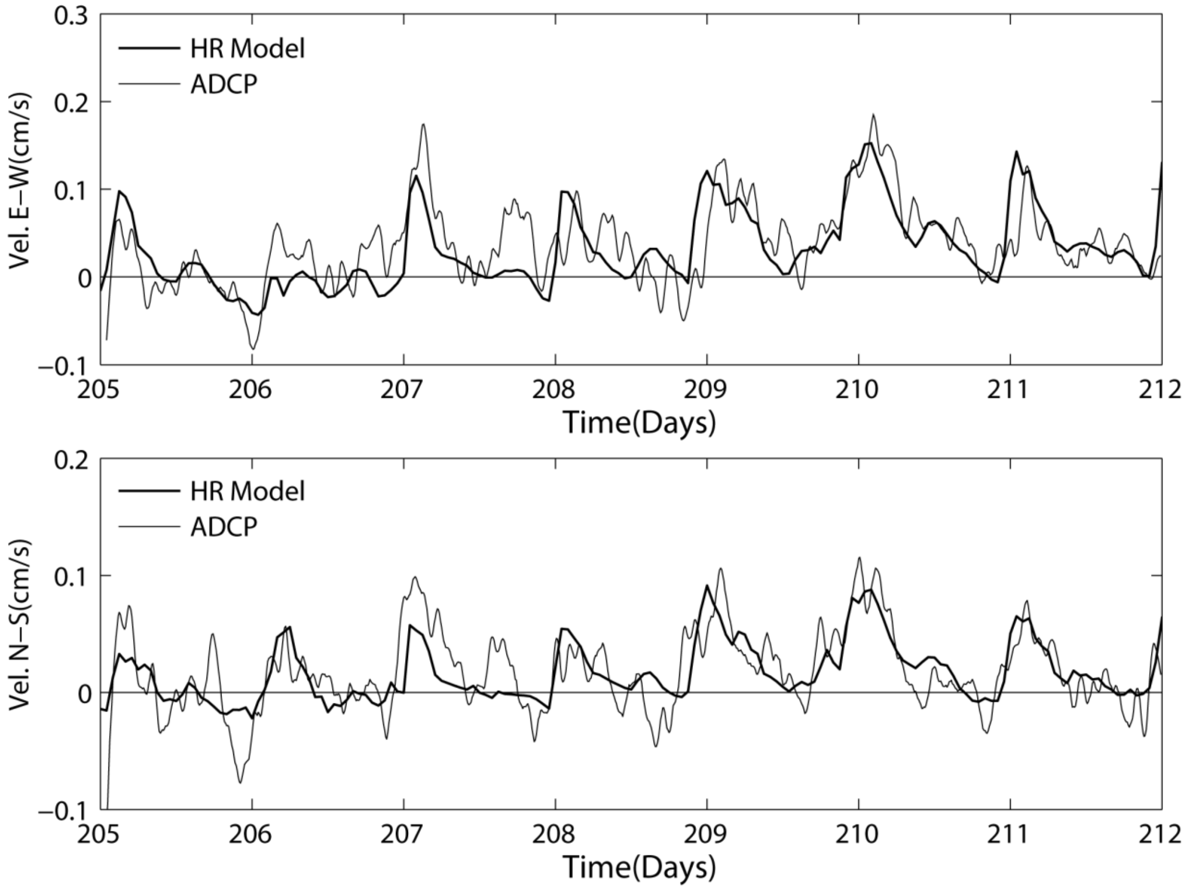


Figure 3.5. East-West Horizontal velocity (*u*) and North-South horizontal velocity (*v*) in a point in the surface-most layer in the period simulation from the Day 205 to the Day 212 in 2008, for experimental data measured by ADCP and simulation data of the high resolution model.

3.5.1 Simulating the fate of storm-water outfalls

Tracer T1 concentrations in the nearshore of Marla Bay are shown in Figure 3.6(e) and Figure 3.6(g). Figure 8(e) represents the concentration of tracer that had been released in the only outfall existing in Marla Bay (Figure 3.3), referred to as local; Figure 3.6(g) represents the concentration of tracer that had been released in South Lake Tahoe, SLT (Figure 3.3), referred to as exogenous. The average concentration of the local tracer increased during the release day (Day 191) and decreased rapidly thereafter. Two days after the start of the release, the average concentration within the shallowest 1.6 m had decreased almost 87%, as a result of near-shore dispersion. The decrease in local tracer concentration occurred mainly during the first half of day 192, coinciding with peak northerly currents offshore in Elk Point (Figure 3.6(c)). The concentration decreased continuously during that time and remained constant, thereafter, until the beginning of day 193. At that time and coinciding with strong northerly currents offshore Elk Point, the local tracer concentration decreased again. The concentration of exogenous tracer released in SLT (Figure 3.6(g)) also increased in Marla Bay early on day 193. These results suggest that the local tracer was displaced by exogenous material that reached Marla Bay (Figure 3.6(g)) as a result of along-shore transport processes. Maximum concentrations of the exogenous material released in SLT were a factor of 10 lower than the maximum concentrations induced by the local outfall. The exogenous tracer was also dispersed rapidly after peaking, at noon on day 193. On day 195 for example, average concentrations of exogenous material in Marla Bay was only 10% of the maximum concentrations.

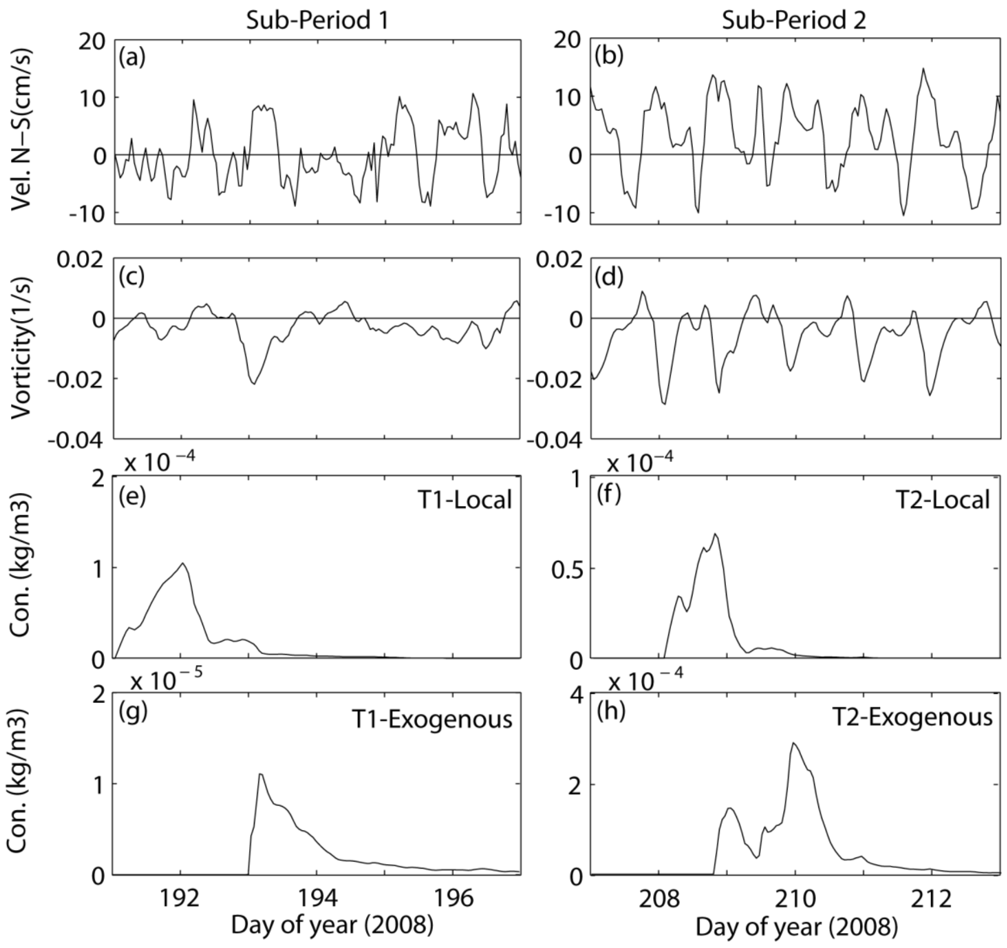


Figure 3.6. Release periods for T1 tracer (Sub-period 1, left) and T2 tracer (Sub-period 2, right). (a) *v* velocity (cm/s) in free surface in Elk Point in the Sub-period 1. (b) *v* velocity (cm/s) in free surface in Elk Point in the Sub-period 2. (c) vorticity calculated

for HR model in Marla Bay in the Sub-period 1. (d) vorticity calculated for HR model in Marla Bay in the Sub-period 2. For the Sub-period 1, (e) T1 averaged local tracer concentration released in Marla Bay and (g) averaged exogenous tracer concentration released in SLT. For the Sub-period 2, (f) T2 averaged local tracer concentration released in Marla Bay and (h) averaged exogenous tracer concentration released in SLT.

A series of tracer T1 concentration fields near the lake surface are shown in Figure 3.7 and Figure 3.8, under weak wind forcing. The tracer shown in Figure 3.7 is of local origin. In Figure 3.8, the exogenous tracer is shown. The tracer cloud in Figure 3.7 at the time of release on day 191 was concentrated near the only outfall in Marla Bay; on day 193, a plume of water entered Marla Bay from the south displacing the local tracer to the north and to the interior of the lake. The tracer released in SLT (Figure 3.8) was largely concentrated near the outfalls at the beginning of day 192. This plume was partially dispersed to the north on day 193, leading to small increases in tracer concentrations within Marla Bay, and rapidly dispersing on day 194. Note, however, that the bulk of the tracer remained largely un-dispersed on SLT towards the end of the experiment.

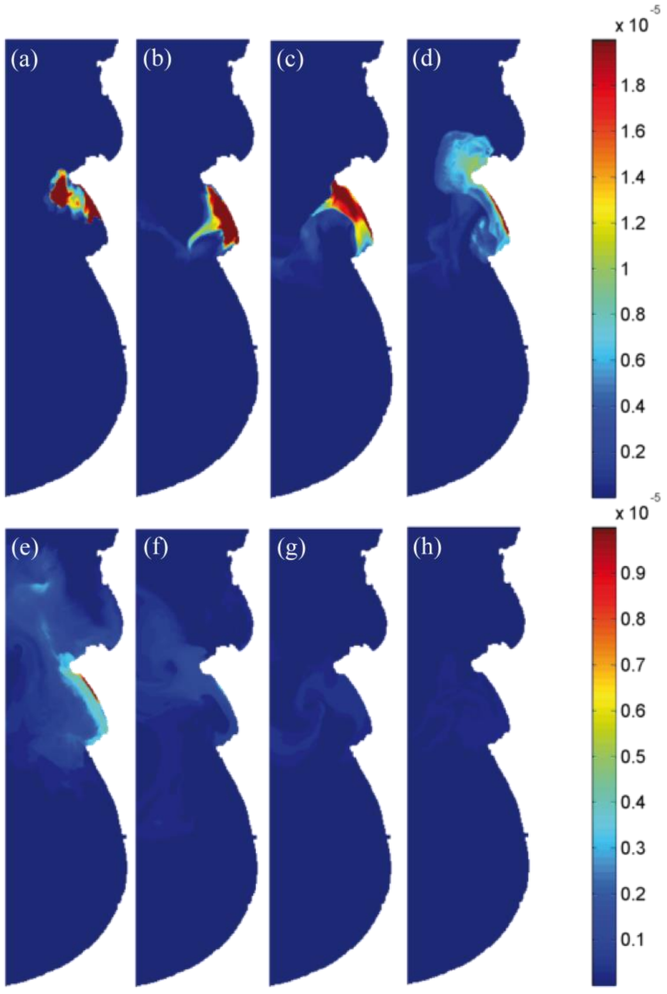


Figure 3.7. Near surface concentration of tracer T1 released locally in Marla Bay (Figure 3.3) in Sub-period 1 at different days (D) and hours (H): (a) D: 191 H: 19:00 (b) D: 192 H: 15:00 (c) D: 193 H: 03:00 (d) D: 193 H: 12:00 (e) D: 194 H: 06:00 (f) D: 195 H: 01:00 (g) D: 196 H: 14:00 (h) D: 198 H: 00:00.

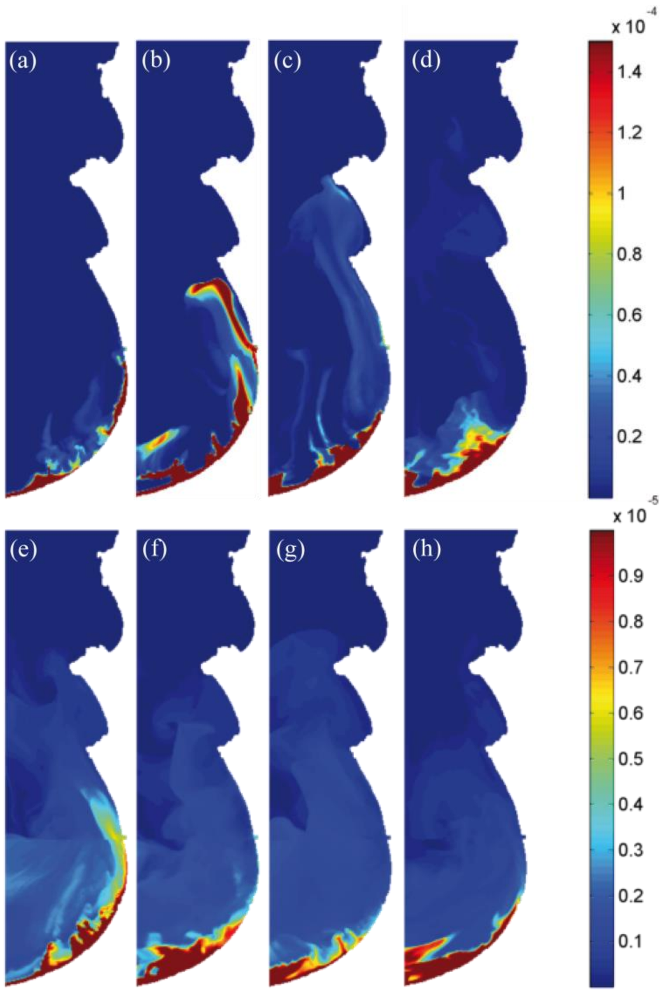


Figure 3.8. Near surface concentration of exogenous tracer T1 released in SLT Figure 3.3) in Sub-period 1 at different days (D) and hours (H): (a) D: 192 H: 12:00 (b) D: 193 H: 00:00 (c) D: 193 H: 06:00 (d) D: 194 H: 00:00 (e) D: 195 H: 23:00 (f) D: 196 H: 17:00 (g) D: 196 H: 23:00 (h) D: 198 H: 00:00.

The average tracer T2 concentrations in Marla Bay for release experiments conducted under strong and persistent wind forcing are shown in Figure 3.6(f) and Figure 3.6(h). Note first that the maximum local tracer concentration (Figure 3.6(f)) was lower than the maximum reached under weak forcing (T1, Figure 3.6(e)), likely a result of stronger dispersion rates occurring under strong wind forcing. The maximum concentrations of tracer released in SLT (Figure 3.6(h)), though, were almost five times larger than those observed under weak forcing (Figure 3.6(g)). The peak concentrations of local and exogenous tracer were similar in these experiments. Note also that two peaks occurred in the time series of exogenous tracer concentrations (see Figure 3.6(h)), on days 209 and 210, and not one as shown in Figure 3.6(g). The timing of those pulses also coincides with decreasing concentrations in Figure 3.6(f). These results are partly the result of stronger along-shore currents (see Figure 3.6(b)), transporting material rapidly from SLT. But, it is also a result of the development of stronger re-circulating eddies of negative vorticity trapping material within Marla Bay during the study sub-period 2 (see Figure 3.6(d)). The speed of rotation of local-scale eddies in the near-shore irregularities are shown to exhibit variations in time, increasing in response to pulses of strong long-shore currents. These pulses, in turn, tend to follow the local wind variations.

As is proved in Chapter 2 (Section 2.4.4), these local-scale hydrodynamic features, such as flow separation and recirculation eddies, occurring in the near-shore region can only be well resolved in the HR model and not by the LR model. Marla Bay is a prototypical example of bays in lake shores Bay-scale where vorticity tends to be larger in magnitude in the high-resolution computations, partly due to the higher resolution of the grid, and partly due to the lower values of K_h used. In Chapter 2 (Section 2.4.4), examples of the vorticity field in Marla Bay calculated by the HR and LR models are shown in Figure 2.13, day 201 in the simulated period. Note that the HR model simulates recirculation in Marla Bay, while the LR model only captures a weak divergence in the velocity field. Being able to simulate these eddies in bays, and other lake shore irregularities, is important in trying to understand coastal transport processes (Rueda and Vidal 2009). As a result of re-circulating eddies, bays can trap particles in suspension and other water constituents, hence, increasing the long-shore dispersion rates. This trapping effect has been reported previously in the literature. For example, Brooks et al. (1999) showed that eddies forming in Cobscook Bay, Maine, could trap particulates in the side-arms of the estuary. The local residence time of water within bays tends to increase as a result of recirculating eddies, hence, becoming hot-spots for the reproduction of species looking for quiet conditions. Nishimoto and Washburn (2002), for example, observed high concentrations of juvenile fish in the center of a large eddy in the Santa Barbara Channel.

Near surface concentration fields of tracer released through outfalls existing in SLT are shown in Figure 3.9. Note, first, that the tracer released in SLT had already reached Marla Bay towards the end of the release, on day 208 (Figure 3.9). The tracer, at that time, appeared concentrated in a narrow plume, along the southeastern shore, far away from the release area. Note, also, that the tracer appeared trapped in Marla Bay on day 210, as a result of recirculating eddies. The tracer leaves Marla Bay finally on days 211 and 212, almost 4 days after the release.

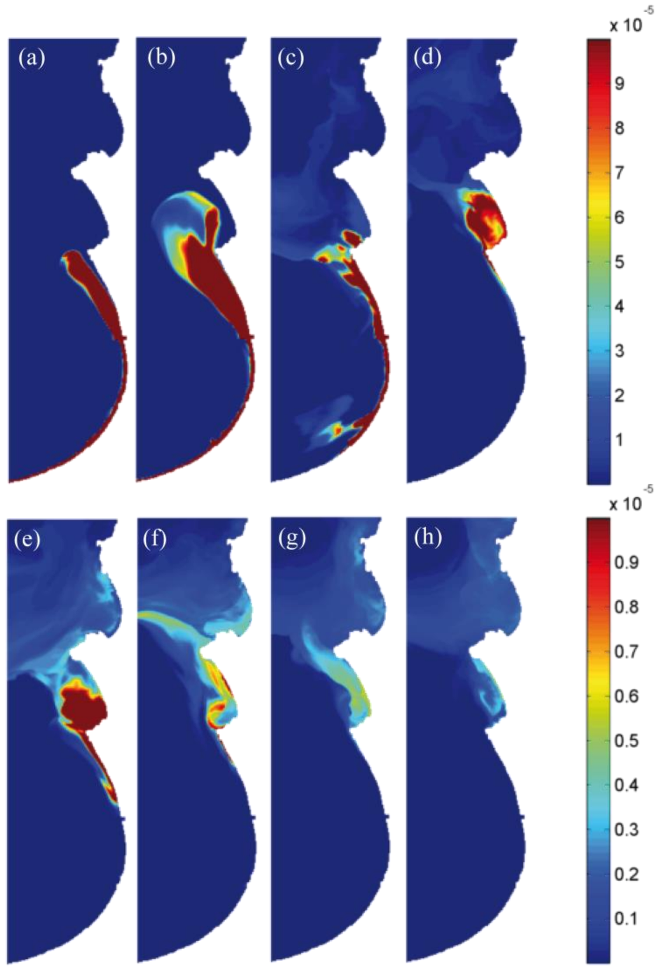


Figure 3.9. Near surface concentration of exogenous tracer T2 released in SLT (Figure 3.3) in Sub-period 2 at different days (D) and hours (H): (a) D: 208 H: 19:00 (b) D: 208 H: 21:00 (c) D: 209 H: 11:00 (d) D: 210 H: 02:00 (e) D: 211 H: 00:00 (f) D: 211 H: 08:00 (g) D: 212 H: 00:00 (h) D: 212 H: 07:00.

3.6 Conclusion

This chapter presents a parallel implementation (P-Si3D) that is able to simulate successfully a Semi-Implicit 3D lake hydrodynamic model in small commodity clusters. The implementation is proved successfully and with an acceptable execution time using mid-resolution and low-resolution of a real test example (Lake Tahoe). Additionally, the combination of P-Si3D and the nesting procedure developed and explained in Chapter 2

(N-Si3D) is able to conduct high-resolution simulations of the littoral fringe of Lake Tahoe, used to conduct tracer transport simulations revealing the pulsating nature of along-shore transport processes in lakes, and the effect of bays and shoreline irregularities on long-shore transport.

This work discusses the performance of several thread- and process- level implementations of the parallel implementation and the influence of different platform configurations and domain decompositions. It has been found that:

- The program makes a weak use of the prefetching hardware (prefetching decreases execution time by between 5% to 8%).
- Intel® Turbo Mode Technology decreases slightly the execution time (by between 3% to 7%).
- Performance is worse if the default BIOS and operating system configuration is used (time increases by between 40% to 60%, depending on the number of processes and threads). This is due to the thread distribution of the operating system among the eight logical cores of a node when Hyper-Threading is enabled. Thread affinity could be used to avoid this problem instead of disable Hyper-Threading.
- Block-driven processing reduces execution time too slightly (4% improvement approximately).
- Process level implementation reduces execution time using overlapping sub-domains (redundant operations).
- The comparison of wide-direction or narrow-direction distribution in a parallel implementation, with MPI communications and with or without redundant calculation, shows that though the number of communications is the same, the quantity of data to calculate or communicate varies. The distribution approach more efficient is the one with a lower border length.
- With the best parallel implementation and performance configuration, and with narrow-cut domain decomposition the simulation of 24 hours with the MR model in a core of the cluster requires proximately 6 hours with one processor (4 threads) instead of 20 hours and 30 minutes (with 1 threads) and approximately 2 hours and 30 minutes with the three processors (12 threads).

Additionally, the high-resolution near-shore model for Lake Tahoe using P-Si3D and N-Si3D is evaluated, validated and used to conduct tracer transport simulations founded that:

- The quality of the results obtained with P/N-Si3D is similar to the solution of a complete HR model. The averaged differences comparing the results of P/N-Si3D with the results of a complete high-resolution model are in all cases less than 4%.
- The physical and chemical environment in specific bays is tightly linked to neighboring bays through along-shore transport processes. For example, water quality in Marla Bay, in the southeastern shore of Lake Tahoe, appears to be strongly influenced by the quality of water in South Lake Tahoe. The influence appears to be stronger during periods of strong winds, when water from SLT is rapidly transported and trapped in the bay as a result of the development of local bay scale eddies.

Chapter 4

Scalable parallel implementation for 3D semi-implicit hydrodynamic models

Abstract

This chapter presents a parallel implementation for semi-implicit hydrodynamic models that scales in low-cost clusters of computers. The scalability of semi-implicit hydrodynamic models is limited due to the need of all-to-one/one-to-all communications at each simulation time-step. These communications are here avoided taking advantage of a nesting implementation, which resolves, in addition to the model with the original grid resolution (nested), a model with a lower grid resolution (parent). Nesting implementations are normally used to simulate both global and local processes with less memory and execution time by using as nested domain just the area where local processes occur while the parent model simulates the complete domain; but here, the nesting implementation is used to improve scalability. A two-level processing structure is proposed for the parallel implementation: pipeline plus domain-decomposition. The resulting two-level parallel structure scales ideally. The computer performance and the quality of the results are evaluated using Lake Tahoe.

4.1 Introduction

Several software packages used to simulate three-dimensional shallow water (3D-SW) are summarized in Table 4.1: EFDC (Environmental Fluid Dynamic Code, Hamrick 1992), MOM (Modular Ocean Model, Griffies et al. 2008), POM (Princeton Ocean Model, Blumberg and Mellor 1987), POP (Parallel Ocean Program, Smith et al. 2010, Dukowicz and Smith 1994), ROMS (Regional Ocean Modeling System, Shchepetkin and McWilliams 2005), and Si3D (Smith 2006). MOM and POP models are more suitable for global processes simulations (global-scale problems). Several parallel proposals for these softwares have been discussed in the bibliography (Table 4.1, 5th column); for example, parallel implementations for EFDC (O'Donncha et al. 2014), MOM (Beare and Stevens 1997,Griffies et al. 2008), POM (Giunta et al. 2007, Jordi and Wang 2012), POP (Smith et al. 2010), ROMS (Wang et al. 2005), or Si3D (Acosta et al. 2010). The difficulty to obtain a parallel implementation and its scalability (i.e. the time improvement when new computing resources are added) depends on the time-discretization scheme used for solving the 3D governing equations: explicit, (semi-

)implicit, or splitting. *Explicit* schemes require higher computational time due to the limitation of the integration time-step to the time a surface (external) gravity wave takes to travel between two adjacent horizontal grid points: this limitation is referred to as Courant or CFL (Courant-Friedrichs Lewy) stability condition for gravity waves. In order to reduce computational time, allowing the use of higher time-steps while retaining free-surface effects, splitting and semi-implicit methods are preferred. *Semi-implicit* approaches (Casulli and Cheng 1992) avoid the time-step limitation due to CFL condition by treating implicitly the gravity-wave terms in the model equations, while other terms are treated explicitly, so that the time-step can be increased. Fully implicit implementations for 3D-SW equations are avoided due to the requirements in computational time and memory; implicit schemes require solving a coupled system of nonlinear equations for velocity and surface elevation over the entire domain each time-step. In shallow water modeling with a semi-implicit scheme (used for example in Si3D), the solutions for surface elevation and velocity are uncoupled, a system of linear equations over the entire domain is solved at each time-step for surface elevation, and velocities are obtained explicitly using the computed surface elevations. The coefficient matrix for this system is symmetric and positive-definite so that the equations can be resolved efficiently using an iterative technique, such as the widely used preconditioned conjugate gradient (PCG). For its part, *splitting* methods (Blumberg and Mellor 1987) separate the 3D governing equations into the so called external or barotropic mode, a 2D model for the depth-averaged flow (associated with the fast moving waves), and the internal or baroclinic mode, a 3D model for the vertical structure of flow (slower moving waves). The coupling of these internal and external modes is required. This splitting allows different time-steps for the 2D and 3D models, enabling the use of explicit integration with a short time-step that satisfies the CFL condition for the fast-moving surface waves and with a longer time-step for the 3D model. However, the problem of coupling the external and internal modes with different time-steps comes up in this case. Several variations of splitting methods are used. Usually, the *internal mode* uses an explicit scheme except for the vertical diffusion terms, which are usually treated implicitly for stability reasons. *External mode* can be either explicit (explicit splitting methods), or it can be implicit or semi-implicit (implicit splitting methods). *Explicit splitting methods* (used for example in ROMS, MOM4.0 and later MOM releases, and POM) avoid the need to solve a system of equations over the entire domain at each external mode time-step (simplifying their numerical and parallel implementation), while *implicit splitting methods* (e.g. EFDC, POP) makes external and internal modes coupling easier allowing the same time-step for both modes. Discussions about the accuracy of (semi-)implicit schemes and (explicit and implicit) splitting methods can be found for example in Smith 2006, and Dukowicz and Smith 1994.

Soft.	References	Time integration	Nesting implementation	Parallel implementation	Programming paradigm	Parallel structure
EFDC	EPA 2002; Hamrick 1992	implicit splitting (PCG solver recommended)		O'Donncha et al. 2014	MPI	domain-decomposition
MOM	Griffies et al. 2008	explicit splitting		Griffies et al. 2008	FMS(1) (MPI)	domain-decomposition
				Beare and Stevens 1997	PVM(2)	domain-decomposition
POM	Blumberg and Mellor 1987	explicit splitting	Giunta et al., 2007 (using the nesting of RSL(3) interface)	Jordi and Wang 2012	MPI	domain-decomposition
				Giunta et al. 2007	RSL(3) (MPI)	domain-decomposition
POP	Smith et al. 2010; Dukowicz and Smith 1994	implicit splitting (PCG solver recommended)		Smith et al. 2010	hybrid OpenMP-MPI	domain-decomposition
ROMS	Shchepetkin and McWilliams 2005	explicit splitting	Debreu et al., 2012, Penven et al., 2006 (using multi-grid of AGRIF(4) interface)	Wang et al. 2005	MPI	domain-decomposition
Si3D	Smith 2006	semi-implicit (PCG solver recommended)	Acosta et al. 2015	Acosta et al. 2010	hybrid OpenMP-MPI	domain-decomposition

(1) FMS (Flexible Modelling System, Balaji 2002). It provides an interface to MPI and to SHMEM (library of Cray)
(2) PVM (Parallel Virtual Machine). Popular tool in the 90s for message-passing programming based on a library of functions. The experience in PVM helped to develop MPI, current de-facto standard based on a library of functions for message-passing programming.
(3) RSL (Runtime System Library, Michalakes 2000). It provides an interface able to define levels of grids and to parallelize the grid levels (domains) over the same set of processors, where each one has a piece of every domain. It uses MPI.
(4) AGRIF (Adaptive Grid Refinement in Fortran, Debreu et al. 2008): provides an interface to define levels of grids.

Table 4.1. Several software packages used to simulate 3D-SW and several parallel implementations proposed for them

The solution of the equation system over the entire domain each time-step makes hydrodynamic models with (semi-)implicit (splitting and non-splitting) schemes more difficult to parallelize and less scalable than explicit splitting and fully explicit schemes. This is a disadvantage of (semi-)implicit (splitting and non-splitting) schemes (also stated in Griffies et al. 2000 and Weller et al. 2013). This disadvantage is especially important nowadays because computational capacity is increasing by adding processing cores also in the low-cost computer market. The scalability is limited due to the all-to-one/one-to-all collective communications required each time-step to obtain surface elevation. To be more precise, the all-to-all reduction communications (i.e. all-to-one reduction plus one-to-all broadcast communications) used by the parallel solver that obtains the surface elevation (see for example Nesterov 2010, Hu et al. 2013) or the couple all-to-one gather and one-to-all scatter communications required for using a sequential solver within a parallel code (see for example Acosta et al. 2010,O'Donncha et al. 2014 and Section 4.2.1). The solver preferred for its efficiency is PCG; in particular, POP (Smith et al. 2010), EFDC (EPA 2002) and Si3D (Smith 2006) recommend it. Note that the smaller computational load (due to the time-step increment) also worsens the scalability of these schemes compared to explicit schemes, i.e. the task of parallelizing explicit schemes is more rewarding for the programmer. To tackle the scalability problem, a less efficient (low accuracy and/or slow convergence) parallel solver for surface elevation with less or no all-to-one/one-to-all communications can be used, such as Chebyshev iteration (used for example in Hu et al. 2013 for POP). The implementation here proposed avoids the scalability limitation by taking advantage of an *online* nesting implementation to eliminate all-to-one/one-to-all communications (Chapter 2 clarifies the difference between *online* and *offline* implementations). The results show that this implementation does not slow convergence, but accelerates it. The robustness of this approach is here shown.

Nesting implementations are used in hydrodynamic models with structured grids to allow simulating both base-scale (global) processes and regional (local) processes reducing both the memory and run-time requirements because they avoid the simulation of the entire basin in the high resolution required to simulate local processes (Figure 4.1). Online nesting implementations (Table 4.1, 4th column) have been included for instance in POM (Giunta et al. 2007), ROMS (Debreu et al. 2012,Penven et al. 2006), or Si3D (Acosta et al. 2015). In nesting schemes a high-resolution (HR) model (nested model), which resolves local physical dynamics in the region of interest, is embedded inside a basin lower-resolution (LR) model (parent model) that simulates the basin-scale processes (Figure 4.1(b)). The different grid cell size of parent and nested models makes the execution time of the nested HR model much higher than the execution time of the parent LR model (the number of columns and cells to be processed are much higher and, additionally, the necessary time-step is smaller).

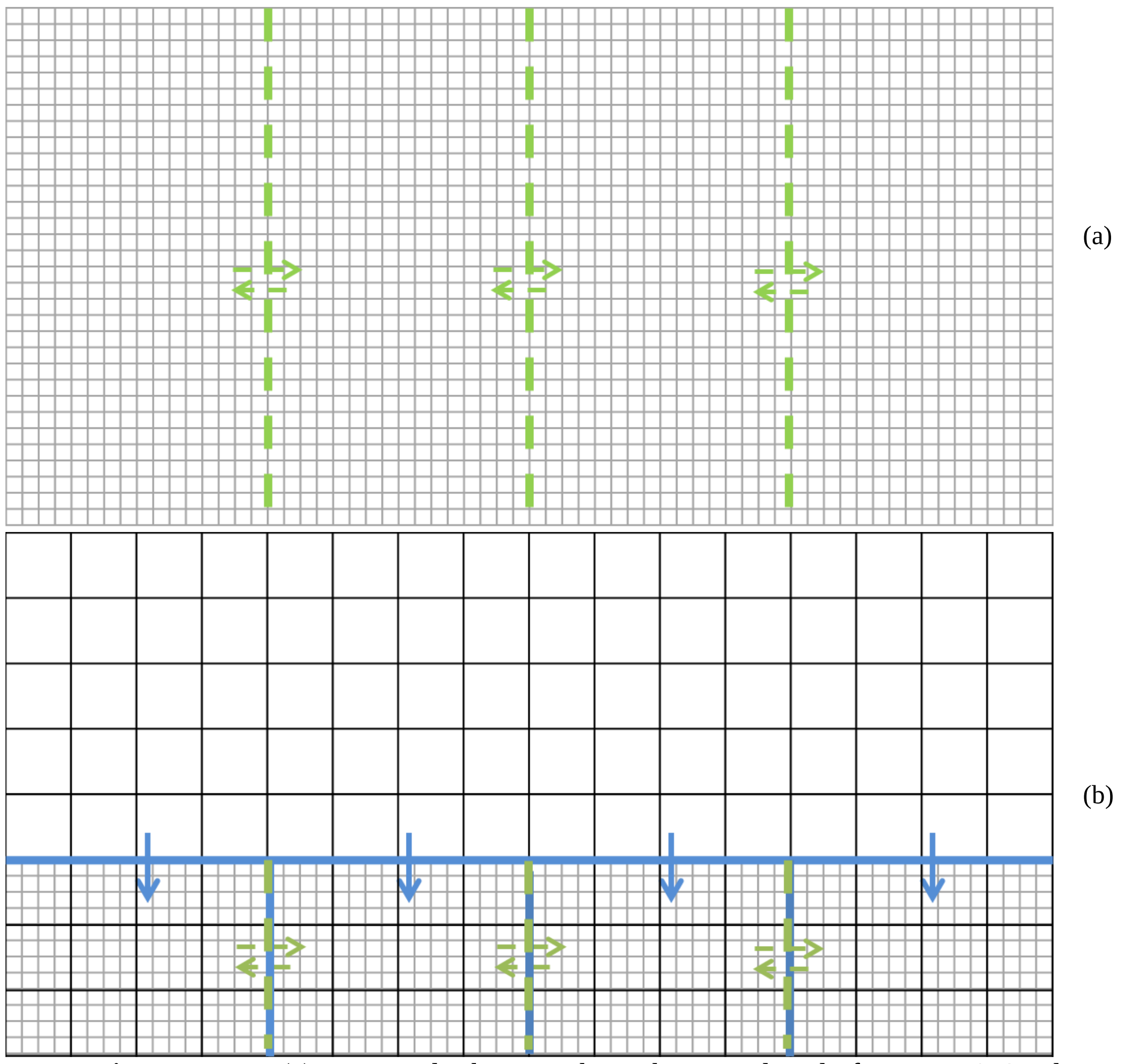


Figure 4.1. (a) Rectangular basin with HR horizontal grid of 32x64=2048 columns (each square is a column). Vertical cuts (green dashed lines) divide the domain into four subdomains (of 512 columns/subdomains). (b) HR grid on a region of interest (gray lines) of 12x64=768 columns nested in a basin LR grid (black lines) of 16x8=128 columns (cell side ratio of 4:1 in the horizontal grid). The nested grid is divided into four subdomains (green dashed lines) of 192 columns. The size of the messages interchanged between adjacent subdomains (green arrows) depends on the length of the border and the depth of the columns. The border with the LR grid is in blue. The amount of data to be transferred from the basin LR model to the nested HR model depends on the length of the border and the depth of the columns. The figures do not show the all-to-one/one-to-all communications among subdomains required by the parallel implementation of hydrodynamic models with (semi-)implicit (with or without splitting) schemes.

The parallel approach here presented uses a nesting scheme to obtain a scalable implementation for a semi-implicit model in a *low-cost commodity cluster* of low-price computers (around ten low-price computers, each one with multiple cores sharing main memory), connected by an inexpensive network. That is, connected by a switch, links and interfaces of Gigabit Ethernet (nowadays motherboards include a 1GbE interface), instead of a custom made network for a particular platform or a proprietary one (e.g.

Gemini of Cray) or a more expensive commodity network (e.g. 10GbE, Infiniband, Myrinet, QsNet …). Scalability is more difficult to achieve in low-price clusters due to the lower performance of the network interconnecting the computers (latency and bandwidth) and the higher number of computers for the same number of processing cores (which increases mean communication time among cores), being these nowadays, the main differences between low-price platforms and other more expensive platforms for general-purpose processing (mid-range and high-end servers in the International Data Corporation taxonomy). The core microarchitecture, core clock-frequency, core local cache, last-level cache size divided by the number of cores, and maximum bandwidth divided by the number of cores is equivalent in low-price, mid-range and high-price platforms. The proposed parallel approach is here applied to a semi-implicit hydrodynamic model, in particular Si3D (Table 4.1, last row), although it can also be used with implicit schemes and (semi-)implicit splitting methods. Other parallel platforms for specific-purpose processing, such as GPUs, and for general-purpose processing can benefit from this approach. The nested model can comprise the entire basin (parent model) or a region of the basin; run-time is additionally improved and total memory requirement is reduced with this last option.

The scalable-parallel approach uses a *hybrid parallel programming paradigm* and a *two-level parallel structure*. The instruction flows assigned to computers are managed by the message-passing programming-paradigm of MPI and those assigned to cores by the shared-memory programming-paradigm of OpenMP. MPI is nowadays a de-facto standard tool for message-passing-based parallel programming, while OpenMP is a de-facto standard tool for shared-memory-based parallel programming. MPI is founded on a function library while OpenMP is founded on compiler directives and a function library. Implementations of MPI and OpenMP tools can be found for Fortran and C/C++ sequential programming languages. Table 4.1 shows the programming paradigm used in several parallel implementations (6th column). The parallel implementation of MOM in Griffies et al. 2008 and that of POM in Giunta et al. 2007 use an API (Application Programming Interface), FMS (Balaji 2002) and RSL (Michalakes 2000) respectively, on top of the more general (flexible) and more low-level parallel programming tool, which is MPI in both cases. An API in general facilitates programmer work at expense of worsening flexibility and execution time.

The *two-level parallel structure* combines a pipeline-like processing at the first level with a structure resulting from a domain-decomposition data-distribution at the second level. There are several attempts to classify task/process structures or task distributions in parallel computing (both are related); some of these classifications can be found in Silva-Moura and Buyya 1998; another term in the bibliography for pipeline is data flow and for domain-decomposition is data-structure decomposition. The nested HR model will be executed in parallel to the basin LR model in a pipeline-like processing and, due to the much higher execution time of the nested HR model, this model will be executed in parallel in several computers applying domain-decomposition. The domain-decomposition implemented here avoids all-to-one/one-to-all communications (and consequent communication delay and run-time increase) by making each computer solve an independent equation system, taking advantage of the online nesting implementation. As a result, the parallel domain-decomposition implementation of the nested HR model decreases the execution time almost linearly with the number of cluster computers. The complete parallel code that includes the two-level structure (pipeline plus domain-decomposition) also scales almost ideally. We have not been able to find any parallel implementation with this two-level structure in the bibliography. The parallel-nested implementation of POM in Giunta et al. 2007(Table 4.1) is built upon the functionality of the RSL interface (Michalakes 2000). In RSL the nested HR model and the basin LR model use the same set of processors. As a result, each one has a piece of every domain (as Michalakes 2000 clarifies), so no pipeline structure is reported.

The performance figures (computer performance and quality of the results) are here obtained using a model of Lake Tahoe (USA), which has a size of roughly 20 km x 30 km in the horizontal dimension and a depth of up to 500 meters in the vertical dimension. Engineers use this model, for example, to study the transport of contaminants and planktonic larvae in the near-shore (littoral) zone (see Hoyer et al. 2014). The proposed implementation can simulate local processes for this example in real time in a low-cost cluster. Much larger size problems would require more expensive platforms.

This chapter is organized as follows. Section 4.2 presents the scalable-parallel implementation and Section 4.3 evaluates it. Section 4.2 justifies the benefit of the scalable-parallel approach proposed taking into account the influence in performance of the different processing stages and of the different kinds of communications among computers. Section 4.3 evaluates run-time and scalability of the parallel implementation and evaluates the quality of the results. Last section gives conclusions.

4.2 Scalable parallel implementation

The parallel implementation presented, that takes advantage of the nesting implementation N-Si3D to achieve scalability and is based on a parallel processing structure of two levels (more details in Chapter 2), is applied here to the semi-implicit hydrodynamic model proposed by Smith 2006, Si3D (Chapter 1) and modified to take advantage of several basic optimizations and an improvement of the data structure (more details about the basic optimizations and the new data structure can be found in Chapter 5, this version is called Basic Si3D). The scalable parallel implementation modifies the parallel implementation presented in Acosta et al. 2010 and explained in detail in Chapter 3, P-Si3D. Additionally, some details of P-Si3D are showed here again (Section 4.2.1) in order to explain the new implementation consistently. The new scalable parallel implementation also uses the nesting implementation proposed and validated in Acosta et al. 2015 and explained in detail in Chapter 2.

4.2.1 P-Si3D implementation

Figure 4.2 shows a simplified flow diagram of a hybrid parallel implementation for multicore clusters, P-Si3D, introduced in Acosta et al. 2010 (Table 4.1) and explained in detail in Chapter 3. The implementation here proposed modifies this parallel implementation making it more scalable in low-cost clusters of computers. P-Si3D has the following design characteristics:

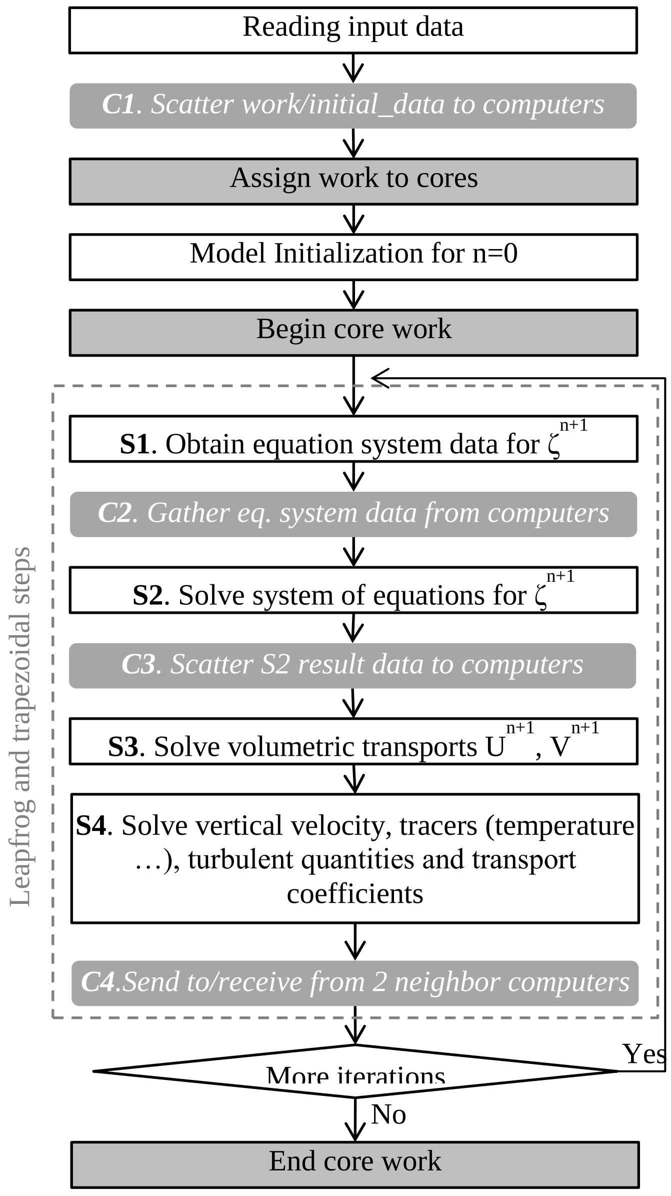

Reading input data
C1. Scatter work/initial_data to computers
Assign work to cores
Model Initialization for n=0
Begin core work
Leapfrog and trapezoidal steps
S1. Obtain equation system data for ζ^n+1
C2. Gather eq. system data from computers
S2. Solve system of equations for ζ^n+1
C3. Scatter S2 result data to computers
S3. Solve volumetric transports U^n+1, V^n+1
S4. Solve vertical velocity, tracers (temperature …), turbulent quantities and transport coefficients
C4.Send to/receive from 2 neighbor computers
More iterations
Yes
No
End core work

Figure 4.2. Solver stages, S1, S2, S3 and S4, for a semi-implicit hydrodynamic model (Si3D) and simplified flow-diagram for a parallel implementation in a cluster of multicores (P-Si3D). All computers execute the same code (they all take part in the gather and the scatter collective communications). The diagram does not show output epochs that store output data in disk. Gray boxes represent processing added for the parallel implementation. C1, C2, C3 and C4 are communications among computers, communications/synchronizations of cores are not shown.

1. It distributes the work among computers (C1 in Figure 4.2) and among the cores in the computers (as Figure 4.2 also shows). The instruction flows assigned to computers are managed by the message-passing programming-paradigm of MPI and those assigned to cores by the shared-memory programming-paradigm of OpenMP. C1, C2, C3 and C4 are communications among computers through message-passing (communications/synchronizations of cores, through the main memory they share,

are not shown). S1, S2, S3 and S4 are Si3D solver stages (see Chapter 1 for more details about the numerical algorithm of Si3D).

2. P-Si3D splits the simulation domain into sub-grids or subdomains assigning contiguous columns in main memory to a subdomain. All the subdomains, including those assigned to the cores of a computer, process columns that are contiguous in memory. All columns are wet (columns with water), dry columns are not considered nor stored in memory. The distribution is made taking into account the number of cells in the columns for improving work balance (the amount of work in S1, S3 and S4 depends on the number of cells). In O'Donncha et al. 2014 can be seen an example of how work balance can affect parallel scalability (a distribution that includes dry columns versus other distribution that does not). A subdomain requires values from the neighbor subdomain for the computation (gradients are obtained). This kind of distribution of work among computers/cores is usually named domain-decomposition distribution.

3. A parallel implementation of shallow water simulation with both explicit or (semi-)implicit methods requires interchange communications (see Figure 4.1), the latter ones also require all-to-one/one-to-all communications (see next item 4). P-Si3D, in order to reduce interchange of data by MPI message passing, overlaps the grids assigned to computers (each subdomain has replicated columns from its neighbors), such that some computations in the overlapped points are performed in two neighbor grids. Thanks to this overlapping, data interchange between computers is needed just once, at the end of S4 stage (C4 in Figure 4.2). Other parallel implementations also report the use of overlapping sub-grids or ghost cells or hallo regions to decrease interchanges; for example, Giunta et al. 2007 (Table 4.1), which executes the parallel code in a platform with single-core computers with two logical cores per computer (2-way Simultaneous MultiThreading, SMT), and O'Donncha et al. 2014 (Table 4.1), which uses a cluster of 5 computers (in a blade packaging) with one 6-core (2-way SMT) processor per computer. P-Si3D does not overlap the sub-grids assigned to cores, since they share memory and can access neighbor’s data with memory reads instead of slower network message-passing, so replicating overlapped computations is no longer an advantage, and a worse execution time is obtained if done so. Due to the domain-decomposition implemented, each subdomain has only two neighbor subdomains as in Figure 4.1.

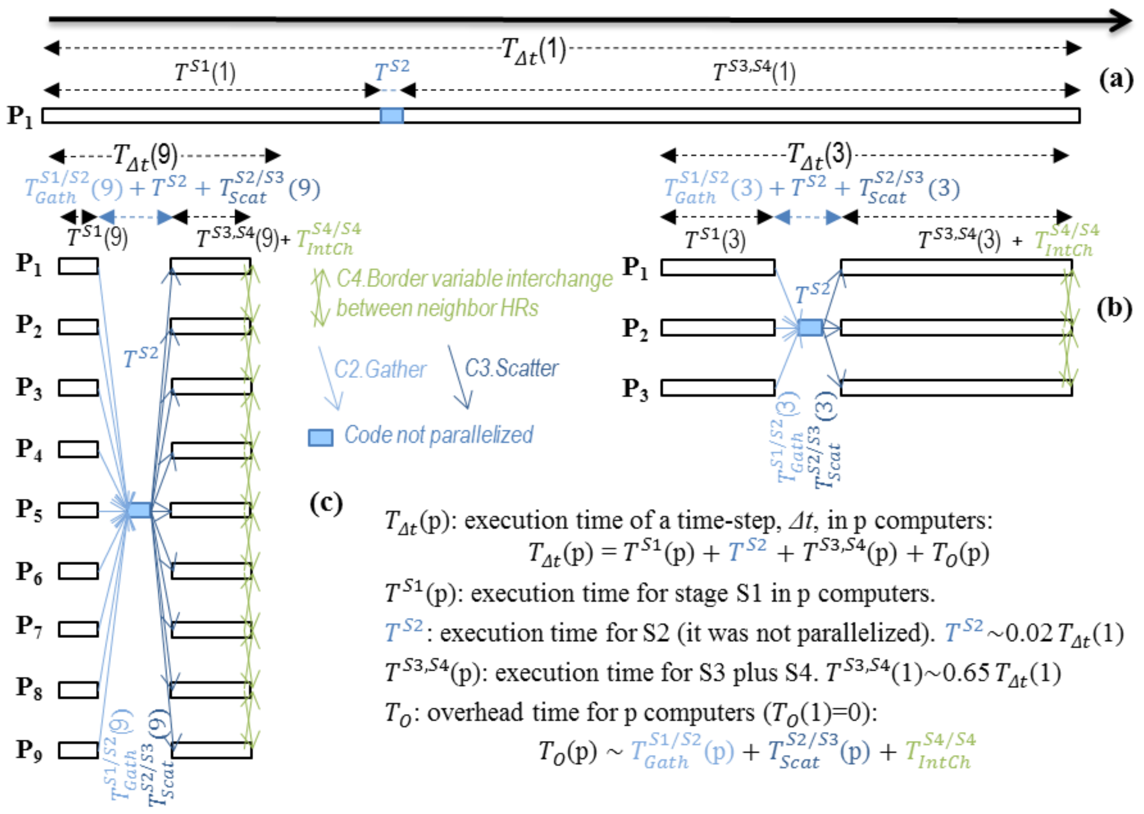
4. S2 solver stage, which resolves a system of equations for obtaining surface elevation ζ^{n+1} with an equation for each of the domain columns by using an iterative method, has a small sequential execution time compared to that of a time-step (sequential time of S1, S2, S3 plus S4 in Figure 4.2). For example, S2 consumes a 2% of the simulation time of Lake Tahoe (with cells of 50 m x 50 m side in the horizontal direction) in Chapter 3 using modified incomplete Cholesky as preconditioner for PCG. This preconditioner was selected by Smith 2006 for Si3D after comparison with other alternatives. We have obtained, for example, improvement of 89-90% in S2 time using modified incomplete Cholesky instead of a Jacobi preconditioner. Due to the low percentage of execution time of S2, executing it sequentially seems reasonable as long as the parallel implementations are executed in just a few computers (as it is going to be clarified below). P-Si3D executes S2 sequentially; in consequence, one collective gather (all-to-one) communication before S2 (to collect all the equation system coefficients obtained by the subdomains) and one collective scatter (one-to-all) communication after S2 (to distribute the calculated ζ^{n+1}) are required (C2 and C3 in Figure 4.2). A parallel execution of S2 would avoid the gather plus scatter communications each time-step but would need, multiple all-to-all reduction (i.e. all-to-one reduction plus one-to-all broadcast) and interchange communications in the solver used in S2 to resolve the equation system each time-step (see, for instance, Nesterov 2010). The number of these communications depends on the number of iterations required by the PCG solver, which in turn depends on the preconditioner used and the size of the equation system. Each PCG solver iteration requires one or two all-to-all reduction plus an interchange communications. A test of these two approaches (gather + sequential S2 + scatter vs. parallel S2) can be seen in O'Donncha et al. 2014 for POP.

As Figure 4.3 clarifies, the run-time of a time-step (Δt) with P-Si3D in p computers ($T_{\Delta t}(p)$) depends on the execution time in p computers of the Si3D solver stages S1, S3 and S4 ($T^{S1}(p)$ and $T^{S3,S4}(p)$), the sequential execution of S2 (T^{S2}) and the overhead ($T_O\ (p)$) introduced by the parallel implementation:

$$T_{\Delta t}(p) = T^{S1}(p) + T^{S2} + T^{S3,S4}(p) + T_O\ (p)$$

$$T_O\ (\mathrm{p}) = (T_{Gath}^{S1/S2}(p)\ +\ T_{Scat}^{S2/S3}(p) + T_{IntCh}^{S4/S4})\ + T_{work-imbalance} + T_{redundant-work} \qquad (4.1)$$

Overhead limits scalability and when it grows as more resources (p) are used, it could cause, after some certain point, that execution time increases instead of decreasing as yet more resources are added. The overhead, in a parallel implementation in general, depends on (1) the communication time (gather, scatter and interchange time in P-Si3D), (2) extra operations (as the previously mentioned redundant computations in the overlapping areas, Section 4.2.1), and (3) the penalty for an imperfect work distribution among computing resources, which causes that some of these resources finish their assigned work after others do (P-Si3D reduces imbalance by distributing columns with water and trying to balance the number of cells with water, as was pointed out before).



$T_{\Delta t}$(p): execution time of a time-step, Δt, in p computers:

$$T_{\Delta t}(\mathrm{p}) = T^{S1}(\mathrm{p}) + T^{S2} + T^{S3,S4}(\mathrm{p}) + T_O(\mathrm{p})$$

T^{S1}(p): execution time for stage S1 in p computers.

T^{S2}: execution time for S2 (it was not parallelized). $T^{S2} \sim 0.02\, T_{\Delta t}(1)$

$T^{S3,S4}$(p): execution time for S3 plus S4. $T^{S3,S4}(1) \sim 0.65\, T_{\Delta t}(1)$

T_O: overhead time for p computers ($T_O(1)=0$):

$$T_O(\mathrm{p}) \sim T^{S1/S2}_{Gath}(\mathrm{p}) + T^{S2/S3}_{Scat}(\mathrm{p}) + T^{S4/S4}_{IntCh}$$

Figure 4.3. P-Si3D. Execution time of P-Si3D for a given problem size (the lengths of the bars are proportional to the time required in a Lake Tahoe simulation with cells of 50 m x 50 m side in the horizontal direction) in (a) one computer or process (P_1), (b) three computers (P_1,P_2,P_3), and (c) nine computers (P_1,…,P_9). $T_{\Delta t}(1)$, $T_{\Delta t}(3)$ and $T_{\Delta t}(9)$ are the total execution time of a time-step in 1, 3 and 9 computers respectively. $T^{S4/S4}_{Intch}$ (in green) is the data interchange time at stage S4 (the first and last processor can also interchange data). One of the computers gathers (in blue) the coefficients of the equation system for obtaining ζ ($T^{S1/S2}_{Gath}$), solves the equation system (T^{S2}), and scatters (blue) the results among computers ($T^{S2/S3}_{Scat}$). The run-time depends on the number of computers (as figure shows), but also on the size of the problem simulated.

The communications in each time-step of P-Si3D include one gather (C2 in Figure 4.2), one scatter (C3) and an interchange of boundary data between subdomains (at the end of S4, C4 in Figure 4.2, see also Figure 4.3). The performance (latency and bandwidth) of these collective communications depends on the time of a point-to-point communication. The communication time between two computers connected through a network (point-to-point communication), such as a low-cost network based on a 1Gbit Ethernet switch (which typically connect tens of computers (8, 16, 24, 48)), depends on the network minimum latency (L, seconds) and maximum bandwidth (*B*, bytes per second) and the size of the message (*m*, bytes). It could be roughly approximated by this expression (as Section 4.3.3 shows):

$$T_{point-to-point}(m) = L + \frac{m}{B} \tag{4.2}$$

L is the time required for a message of small size; around 30 µs for the 1GbE links and switch used here in the test cluster (using MPI), around hundreds of nanoseconds with links of high-performance networks. *B* is the bandwidth for large message size *m*. The ideal value for *B* is the capacity of the link (1 Gbit/s = 125 MB/s for 1GbE links in one direction), *B* is around 117.6 MB/s per direction in the test cluster with MPI, tens of gigabytes per seconds in one direction with links of high-performance networks. For scatter (one-to-all) and gather (all-to-one) collective communications, the time depends on the size *m* of the message scattered from or gathered to one node (name here the root node) and the number of computers *p* involved. If the node that scatters also receives data (as in C3) or the node that gathers also sends data to itself (as in C2), both the minimum latency *L* and the maximum bandwidth *B* depend on the number of computers involved. The total latency of these scatter and gather communications could be roughly approximated by this expression:

$$T_{Scat}^{one-to-many}(m,p) = T_{Gath}^{many-to-one}(m,p) = L\,(p) + \frac{m}{B\,(p)} \tag{4.3}$$

When the number of computers increases, the minimum latency *L(p)* will increase, because there is just one link connecting the root node to the switch that every packet sent (scatter) from or received (gather) by this node must traverse, and *B(p)* will decrease, because the number of bytes truly sent or received through the network increases (notice that the bytes from or to the root node are not really transferred through the network). The lowest limit of this bandwidth is the point-to-point bandwidth and, the highest, twice the point-to-point bandwidth. If the node that scatters does not receive data or the node that gathers does not send data, also the minimum latency *L(p)* would decrease with the number of computers and the total bandwidth will increase, but the maximum bandwidth would not depends on the number of computers. It would be the maximum point-to-point bandwidth regardless the number of computers involved. This happens because there is just one link connecting the root node with the switch and all the *m* bytes are transferred through the network. More precise communication models than those of the Eq. (4.2) and (4.3) can be found in the bibliography (for example in Pjesivac-Grbovic et al. 2007), which can model packets (networks split messages into packets, which are routed individually through the interconnection network), more complex networks (based on multiple switches), and/or network congestions (useful, in particular, when the network is shared by several applications at once).

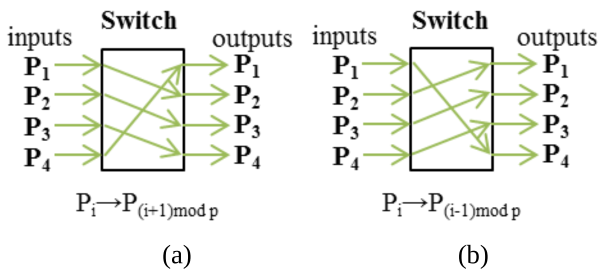

inputs
Switch
outputs
P_1
P_2
P_3
P_4
$P_i \rightarrow P_{(i+1)\mathrm{mod}\,p}$
(a)
inputs
Switch
outputs
P_1
P_2
P_3
P_4
$P_i \rightarrow P_{(i-1)\mathrm{mod}\,p}$
(b)

Figure 4.4. Interchange collective communication includes two shift permutations (a) right and (b) left. Each of these permutations can be performed in parallel in a network based on a crossbar full-duplex switch. The switch in the figure connects four computers (P_1 to P_4)

The communications in each time-step of P-Si3D also include an interchange of boundary data between subdomains (at the end of S4, C4 in Figure 4.2, see also Figure 4.3). Due to the domain-decomposition implemented, each subdomain has only two neighbor subdomains. The interchanges of data between neighbor subdomains can proceed in parallel, as Figure 4.3 shows, with a time that does not depend (at least, noticeably) on the number of computers in a network based on a crossbar switch (usual in clusters). A crossbar switch allows that all or several switch inputs transfer data to outputs in parallel as long as each input requires a different output. The interchange of data between neighbor subdomains comprises two shift permutations (bijective functions) because each subdomain has two neighbor subdomains in P-Si3D. Each of these permutations can be implemented in parallel in a network with a full-duplex switch, full-duplex links, and interfaces with independent input and output queues, as Figure 4.4 illustrates for a switch connecting four computers. Therefore, it is expected that the interchange latency and local bandwidth (bandwidth of a node) will be roughly twice the latency and bandwidth of a point-to-point communication. In practice, the number of computers will affect the interchange performance slightly because the switch has to cope with more packets coming from more sources. Moreover, C4 impact could be reduced or even avoided by overlapping it with computation (the border data to be sent can be obtained first as for example in Beare and Stevens 1997). This overlapping is not possible with C2 and C3 (S2 can start when S1 finishes, S2 obtains all surface elevations at once, so S2 must finish to continue as Figure 4.3 illustrates). Therefore, for large amount of bytes transferred through the network, the latency for a gather plus a scatter and for interchange is expected to be similar (for the same number of bytes) and near twice the point-to-point latency (i.e. the latency of a Ping-Pong). For small amount of bytes, gather+scatter latency for more than two nodes (plus the root node) is expected to be higher than the interchange latency, since the latter is approx. twice the point-to-point latency while the former is predicted to depend on both the point-to-point latency and the number of nodes (in the worst-case, point-to-point multiplied by the number of nodes). Section 4.3.3 shows test results (see, for example, Figure 4.12).

For C4, the size *m* of the messages depends on the number of grid layers (grid depth) and the length of the border between neighboring subdomains (see Figure 4.1). For the gather C2 and the scatter C3, the size of the scattered or gathered message (*m* in Eq. (4.3)) depends on the number of columns in the entire grid, i.e., number of equations and unknowns in the S2 equation system. The amount of data to be transferred in C2, C3 and C4 depends on the particular simulation. Generally, the amount of data to be interchanged (C4) will be less than the amount of data trasferred by the gather+scatter required (C2-C3). Moreover, C4 can be overlapped with computation.

Redundant operations and work imbalance (with the P-Si3D work distribution mentioned before, Eq. (4.1)) affect overhead in a lesser degree than the S2 sequential execution and the C2-C3 communications, and their overheads do not depend on the number of computers. Figure 4.3 ((b) and (c)) assumes a perfect balance (see length of the bars):

$$T^{S1}(p) = \frac{T^{S1}(1)}{p} \qquad T^{S3,S4}(p) = \frac{T^{S3,S4}(1)}{p} \tag{4.4}$$

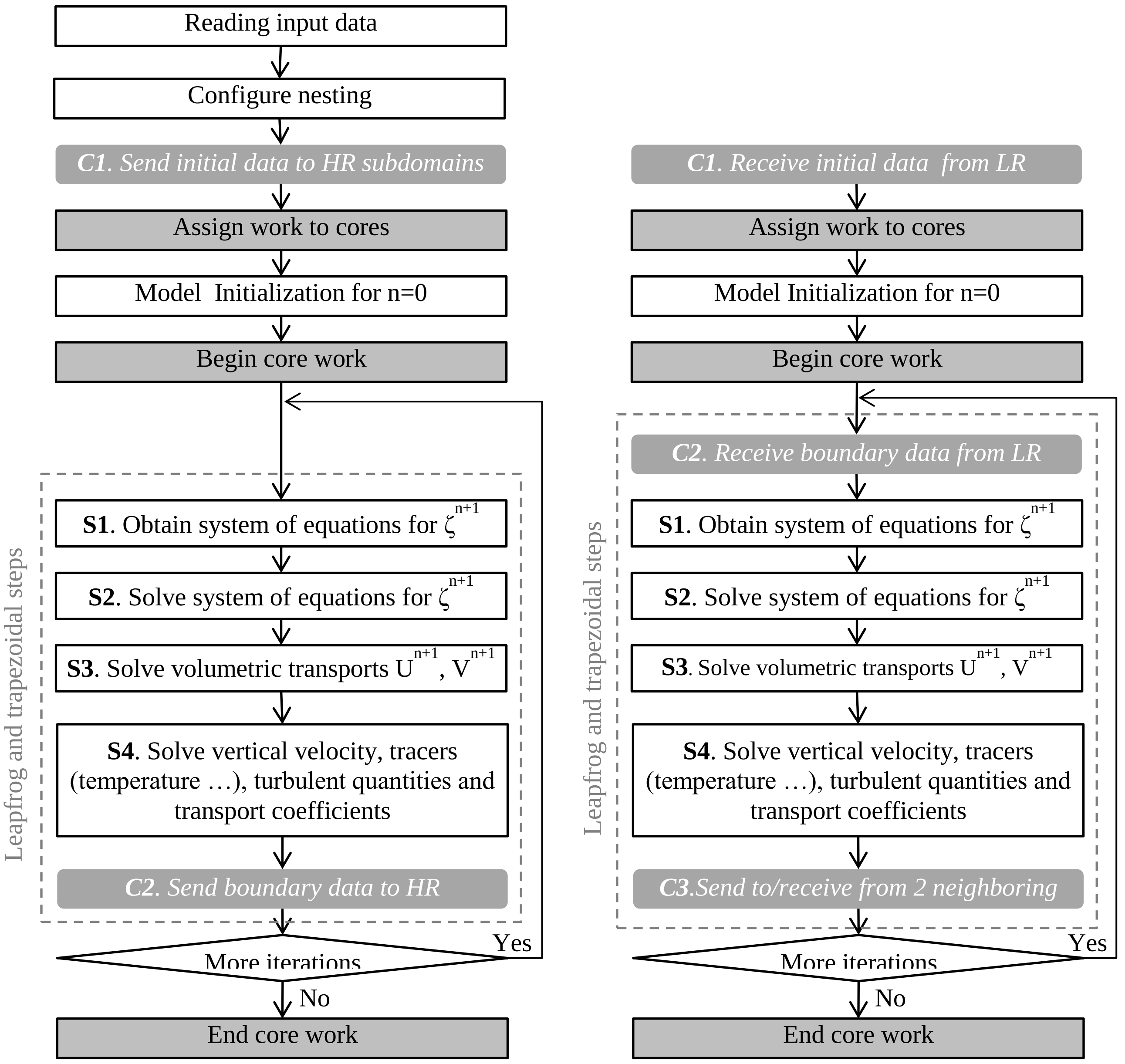
Taking into account the above explanation and, as Figure 4.3 illustrates, P-Si3D scalability is mainly limited by (1) the non-parallelized code, S2, and (2) the gather and scatter communications. Figure 4.3 shows that for nine computers the time of gather plus S2 plus scatter becomes an important percentage of the total time-step run time. The parallel implementation here proposed eliminates these problems by executing S2 in parallel and without all-to-one/one-to-all communications; as a result, the implementation is going to scale linearly with the number of computers.

4.2.2 Scalable parallel semi-implicit implementation

The parallel implementation proposed, SP-Si3D, uses online one-way nesting in order to obtain a scalable speedup as more computers are added. The nested grid can be the entire basin model (equal in extension to the parent model) or a region of the basin. In this last case, the requirements of memory and execution time are additionally reduced by the nesting approach. The different cell size of parent and nested models (even of 5:1, with number-of-columns ratio of 1:25, and time-step ratio of 1:5, yielding ratios of total work load of even 1:125 for the same simulation period) makes the execution time of the nested HR model much higher than the execution time of the parent LR model. It is most likely in low-cost clusters that parent LR model execution in real time would not require more than one computer while the nested HR model would require for real time performance multiple computers, each one executing a subdomain of the nested HR grid. Figure 4.5 shows a simplified flow diagram of the parent LR model code executed by a computer and the flow diagram for a nested HR subdomain. Figure 4.6 (a) shows the execution in parallel of LR and HR models during several time-steps (in particular, n-1, n, n+1 and n+2 time-steps of parent LR model and n-2, n-1, n and n+1 of nested

HR model are shown). The parent LR model is executed in a computer in parallel to the execution of the nested HR model (Figure 4.6 (a)) with a structure that resembles a pipeline architecture processing (Figure 4.6 (b)) with three stages (LR stage, interconnection network stage and HR stage). Due to the much higher execution time of the nested HR model, this model is also executed in parallel in p computers, with a domain-decomposition structure (Figure 4.6 (b)). The parallel implementation of the nested HR model departs from the MPI parallel implementation of P-Si3D in order to obtain a scalable speedup as more computers are added by taking advantage of the parent LR model. The parallel HR model implementation executes S2 solver stage in parallel without all-to-one/one-to-all collective communications. S2 equation system used to obtain surface elevation ζ^{n+1} is divided into p independent equation systems (one per subdomain). These equation systems need unknown elevations values from the n+1 time-step (in particular, in the HR equations formulated for border/corner columns) that are going to be calculated at the same time by the equation system of a neighbor HR subdomain. To overcome this problem, these values are obtained from the parent LR model by interpolation. The LR values of the neighbor (obtained from the parent) are multiplied by the respective coefficients and moved to the independent term in the right hand side, making the p equation systems independent of each other. As the equation systems are smaller than the original equation, the number of iterations required for each one to converge to the same required tolerance will be smaller as well.

The time required by the interchange of data between neighboring HR subdomains, $T_{IntCh}^{HR/HR}$ in Figure 4.6 ($T_{IntCh}^{S4/S4}$ in Figure 4.3) depends on the size of the data to be transferred (m in Eq. (4.2)), which depends on the length of, and number of layers in, the frontier between neighboring subdomains, as Section 4.2.1 pointed out. In SP-Si3D, the LR model sends data to all HR subdomains (C2 in Figure 4.5). The time of this scatter depends on the number of computers, p, and the amount of data to be transferred, m. Therefore, the time of this scatter from LR model to HR subdomains for a given problem size depends on the number of computers (subdomains), p, used to execute the nested model, but this time ($T_{Scat}^{LR/HR}(p)$) is not perceived because these communications occur in parallel to computations, as Figure 4.6 (a) shows, due to the pipeline implementation. The pipeline of SP-Si3D has three stages as Figure 4.6 (b) shows:


Reading input data
Configure nesting
C1. Send initial data to HR subdomains
Assign work to cores
Model Initialization for n=0
Begin core work
Leapfrog and trapezoidal steps
S1. Obtain system of equations for ζ n+1
S2. Solve system of equations for ζ n+1
S3. Solve volumetric transports U n+1, V n+1
S4. Solve vertical velocity, tracers (temperature …), turbulent quantities and transport coefficients
C2. Send boundary data to HR
More iterations
Yes
No
End core work
C1. Receive initial data from LR
Assign work to cores
Model Initialization for n=0
Begin core work
C2. Receive boundary data from LR
Leapfrog and trapezoidal steps
S1. Obtain system of equations for ζ n+1
S2. Solve system of equations for ζ n+1
S3. Solve volumetric transports U n+1, V n+1
S4. Solve vertical velocity, tracers (temperature …), turbulent quantities and transport coefficients
C3.Send to/receive from 2 neighboring
More iterations
Yes
No
End core work

(a) Processing of the parent LR model by a computer **(b)** Processing of a nested HR subdomain in a computer

Figure 4.5. SP-Si3D. Simplified flow diagrams for LR model and HR subdomains. The diagrams do not show output epochs that store output data in disk. Gray boxes represent processing added for the parallel implementation. C1, C2 and C3 are communications among computers; communications of cores are not shown. S1,…S4 are Si3D stages.

- (1st stage, LR) parent LR model ($T_{\Delta t}^{LR}(p^{LR})$) executed in one computer using one or several of the computer's cores (p^{LR} = #cores_used_by_LR / #cores_per_computers). The unused computer cores can be assigned to another user or to another task of the same user.
- (2nd stage, C2) communication C2 in Figure 4.5 ($T_{Scat}^{LR/HR}(p)$) performed by the interconnection network.

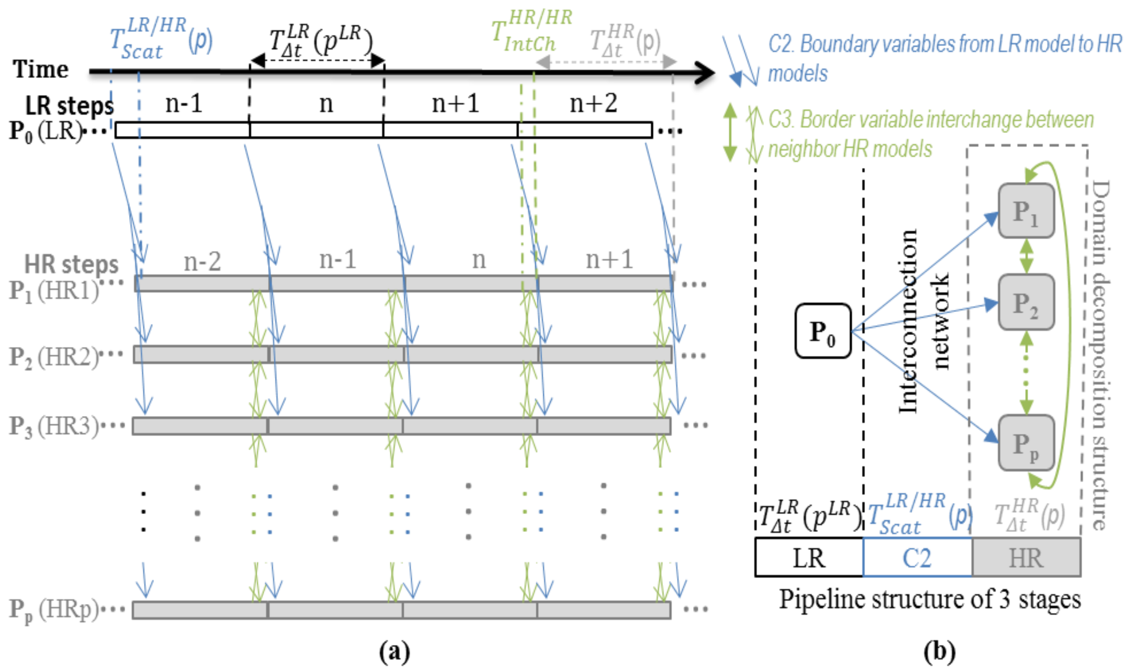


Figure 4.6. SP-Si3D. (a) Parallel execution of SP-Si3D, with a particular problem size, in time-steps n-2, n-1, n, n+1 and n+2 with several computers (processes) $P_0 \ldots P_p$. (b) Pipeline plus domain-decomposition structure. The figure assumes same time-step for parent LR an nested HR models ($\Delta t = \Delta t_{LR}$=Δt_{HR}) for simplicity and also assumes that load balance is perfect, i.e. the HR models (HR1,…,HRp) and the LR model complete a simulation time-step in the same time ($T^{LR}_{\Delta t}(p^{LR})$= $T^{HR}_{\Delta t}(p)$). Communications C2 (blue arrows) and C3 (green arrows in 0 are shown (the first and last processor can also interchange data as figure (b) shows, figure (a) does not include it for a cleaner drawing). $T^{LR/HR}_{Scat}(p)$ is the communication time from LR to all HRs models (C2) and $T^{HR/HR}_{IntCh}$ is the time spent on data interchange between neighboring HR models (C3).

- (3rd stage, HR) nested HR model ($T^{HR}_{\Delta t}$(p)) executed in parallel by *p* computers.

These three stages use different hardware resources so they are executed in parallel. In a pipeline structure, a result is obtained in a time equal to the time of the slower stage. In this pipeline, the slower stage will be usually the 3rd (even though the LR model uses less resources, work load ratios LR:HR of even 1:125 can be found). This is due to the different grid resolution of the parent and nested models (number of cells, time-step value), the extension of the nested model (which can include the entire basin), and the size of low-cost clusters. The resources (p^{LR}) to be used in the execution of the LR model can be chosen in order to approximate its run time to the run time of the HR model (($T^{HR}_{\Delta t}(p)$ ~$T^{LR}_{\Delta t}(p^{LR})$). The execution time of a time-step, Δt, in SP-Si3D is (taking into account that $T^{HR}_{\Delta t}(p) \geq T^{LR}_{\Delta t}(p^{LR})$ and $T^{HR}_{\Delta t}(p) \geq T^{LR/HR}_{Scat}(p)$):

$$T_{\Delta t}\left(p+p^{LR}\right) = \max\left(T_{\Delta t}^{HR}(p),\ T_{Scat}^{LR/HR}(p), T_{\Delta t}^{LR}\left(p^{LR}\right)\right) = T_{\Delta t}^{HR}(p)$$

$$T_{\Delta t}^{HR}(p) = T^{S1}(p) + T^{S2}(p) + T^{S3,S4}(p) + T_O \qquad where \qquad T_O \approx T_{IntCh}^{HR/HR} = T_{IntCh}^{S4/S4} \tag{4.5}$$

N simulation time-steps are executed using SP-Si3D (Figure 4.6 (a)) with an execution time of (LR model must start first):

$$T_{simulation}(N,p) = T_{\Delta t}^{LR} + T_{Scat}^{LR/HR}(\mathrm{p}) + \mathrm{N} \times T_{\Delta t}^{HR}(p) \approx \mathrm{N} \times T_{\Delta t}^{HR}(p) \tag{4.6}$$

Therefore, SP-Si3D run-time (Eq. (4.5) and Eq. (4.6)) does not depend on the computation of a sequential code and on gather and scatter communications, as the execution time of P-Si3D (Eq. (4.1)). Moreover, as it can be observed, the overhead of SP-Si3D does not depend on the number of computers. The communication time depends on the interchange communication, which can be overlapped with computation as was pointed out before.

To the best of our knowledge, the combination of task structures applied here, pipeline and domain-decomposition, has not been previously used in fluid dynamics simulations. It can be used with explicit, (semi-)implicit and (semi-implicit) splitting methods with nesting.

4.3 SP-Si3D performance evaluation and validation

The computing performance (Section 4.3.2) and the quality of the results (Section 4.3.5) obtained by SP-Si3D are evaluated using Lake Tahoe (Section 4.3.1).

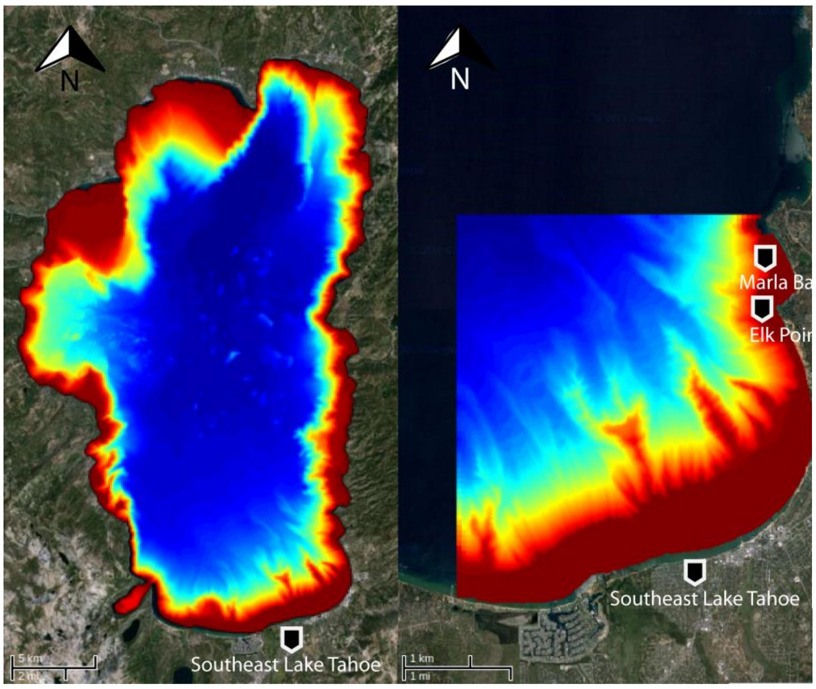
4.3.1 Lake Tahoe simulation figures

The objective is to simulate the local processes in the littoral zone of Lake Tahoe, which has a size of roughly 20 km x 30 km in the horizontal dimension and a depth of up to 500 m in the vertical dimension (Table 4.2, Figure 4.7(a)). Near-shore circulation can be used in different studies. For example, in Lake Tahoe, it can be used to develop a long-term risk assessment of invasive species (such as Asian clam) growth, spread and impact (Hoyer et al. 2014). In this study, the near-shore circulation can be used to develop a transport model of Lake Tahoe to characterize the pathways of transport of invasive species from the existing beds to other near-shore areas. To achieve this goal, a high resolution must be used where fine-scale information is needed, such as in the near-shore. In this case, it is not necessary to simulate the entire lake with a high resolution, so the nested HR model can encompass just the littoral zone (Figure 4.7(b)). In order to make this study, cells of 20 m x 20 m are required in this region (this size has been used for example in Hoyer et al. 2014). With 20 m x 20 m square cells, the time-step must be set to 10 s to guarantee model stability. The parent LR model has 100 m x 100 m horizontal cells, i.e. the cell side ratio LR:HR is 5:1 (number-of-columns ratio LR:HR of 1:25). The time-steps of the parent LR and nested HR models (Δt_{HR} and Δt_{LR}) do not need to be the same; in fact, Δt_{LR} could be larger than Δt_{HR} given the larger size of the grid cells (50 s instead of 10 s, Table 4.2, time-step ratio LR:HR of 1:5). The time between consecutive LR/HR communication events could also be a multiple of the time-step in the parent LR model. The boundary condition information, in those cases (different LR and HR time-steps and/or LR/HR communication events of multiple LR time-steps), is interpolated in time. This interpolation increases the errors in the nesting implementation (as shown in Chapter 2, Section 2.4)), although in a lower degree than the iterative solver used for obtaining free surface (Table 2.4 in Chapter 2). In the applications shown here (Section 4.3), HR and LR models are set to run with the same time-step (i.e. $\Delta t = \Delta t_{LR} = \Delta t_{HR}$ =10 s, Table 4.2) and the communications were forced to occur after every iteration, so no temporal interpolation error is introduced at expense of a higher run time of the parent LR model. By using a time-step of 50 s instead of 10 s for the LR model, its run time could be divided by approx. 5 or, alternatively, the number of resources assigned to the LR model could be divided by approx. 5.

Models of Lake Tahoe	Horizontal cell side	#Wet_ columns	Column ratio	# Wet_cells	Cell ratio	Simulation period	Time-step	# Time-steps
LR basin	100 m	50,383	0.04	3,657,268	0.04	30 days	10 s / *50 s*	259,200 (10 s), *51840 (50 s)*
HR littoral	20 m	493,317	0.4	21,181,918	0.22	30 days	10 s	259,200
HR basin	20 m	1,244,896	1	94,691,170	1	30 days	10 s	259,200

Table 4.2. Simulation figures: resolution (square cell horizontal side); total number of columns and number of cells; period of simulation; simulation time-step used / *time-step that could be used taking into account the cell size*; number of time-steps for the whole simulation period.

The bathymetry data was downloaded from http://tahoe.usgs.gov/bath.html. In the simulations, the model was forced (input data) using surface heat and momentum fluxes estimated from local atmospheric variables (short and long wave radiation, air temperature, relative humidity, and wind speed and direction) obtained from meteorological data. These data were taken primarily from meteorological stations maintained by the Tahoe Environmental Research Center (TERC). There are ten shoreline and on-lake meteorological stations. All stations provide a near-continuous record of wind magnitude and direction and air temperature. The model was simulated from July 3th 2008 (Julian Day 185) to August 1nd (Julian Day 214), with hourly output epochs.



(a)

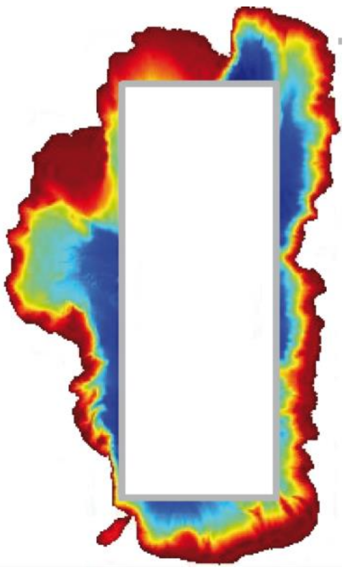

(b)

Figure 4.7. Lake Tahoe (between California and Nevada in USA). (a) Bathymetry and location of Marla Bay. The basin HR model and the basin LR model (or parent model) comprise the entire Lake Tahoe basin. (b) Littoral zone simulated in the nested HR model.

4.3.2 Computing performance

The parallel simulations were conducted in ARCHIMEDES, a standard rack-mount cluster of nine rack computers (including the front-end) connected by a Gigabit Ethernet switch, the Dell PowerConnect 2824, which is a full duplex switch with 2 Mb of packet buffer memory. The computers have 12 GB of main memory and two Intel® Xeon® CPU L5506 processors (4 cores, hyperthreading (Intel SMT), 2.13 GHz, 4 MB last-level L3 cache, 4.80 GT/s Intel® QPI, low thermal design power 60 W, Intel recommended customer price $423). Each core is two logical cores, so it can execute two instruction flows in parallel. SP-Si3D run-time improves by using logical cores when only one computer is available to execute both LR and HR models, for more than one computer, one logical core per core improves run-time. The cluster works under Rocks Linux distribution. The code was compiled using the Intel Fortran 11.1 compiler. The OpenMP included in this compiler is used to manage the instruction flows assigned to (logical) cores. The MPI implementation that manages the instruction flows assigned to computers is MPICH2-1.2.

4.3.3 Communication performance

This section evaluates the performance (latency and bandwidth) for a point-to-point communication and for the collective communications used in P-Si3D and SP-Si3D, which depends on the point-to-point performance. The results support the deductions made in Section 4.2.1. The cores, the network (links, switch and interfaces), the operating system (network interface drivers) and the MPI implementation affect performance.

Figure 4.8 shows the performance of a point-to-point communication, i.e. a communication between two nodes, with MPI in the test cluster for different message size *m*. The latency and bandwidth was obtained using a Ping-Pong test with two nodes, i and j: first, the node i sends a message of *m* bytes to node j (ping), when node j receives this message it sends the same *m* bytes back to the i (pong). The node i obtains the time spent on these two point-to-point transfers. The latency in the figure is obtained dividing this time by two. The figure also shows the approximation of the point-to-point time by the Eq. (4.2) and shows also the L and B parameters used to model this point-to-point time according to Eq. (2).

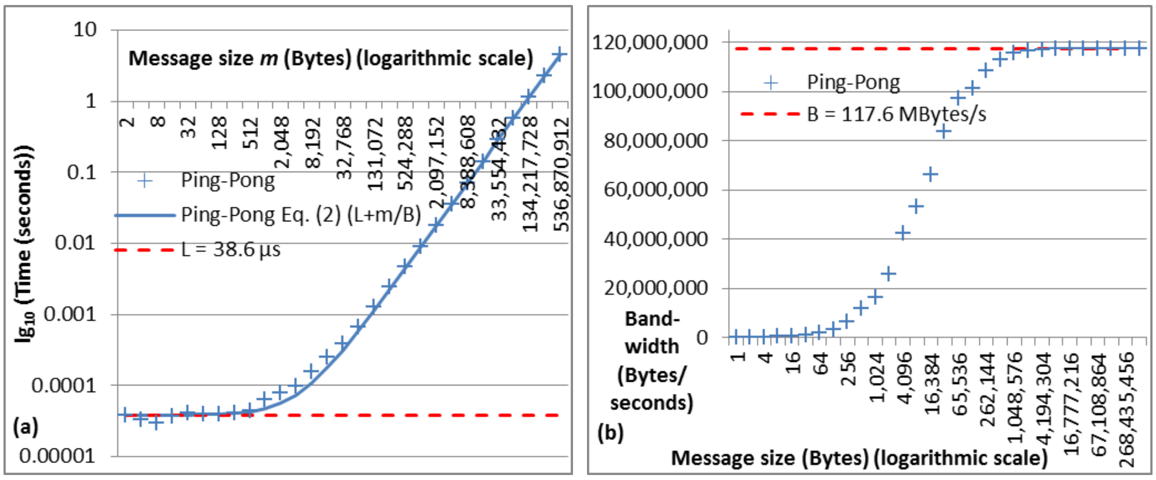


Figure 4.8. Point-to-point performance in the test cluster: (a) Latency vs. message size *m*, minimum latency (*L* in Eq. (4.2)) and latency approximation by Eq. (4.2) using *L* and *B*. (b) Bandwidth vs. message size *m* and maximum bandwidth (*B* in Eq. (4.2)).

As pointed out in Section 4.2.1 the performance of the scatter, gather and interchange collective communications involved in the processing depends on the particular input to be simulated, since the size of the data to transfer depends on this input (bathymetry, depth, geometry, …). Figure 4.9 shows the performance of a scatter/gather collective communication for different size *m* and number of computers *p*. Performance was obtained with a code that performs a scatter of *m* bytes (ping) followed by a gather (pong) of these *m* bytes spread by the scatter. Scatter and gather were implemented by MPI *MPI_scatterv()* and *MPI_gatherv()* functions. Between the scatter and gather the nodes execute a barrier (*MPI_barrier()*). The node that scatters and gathers the data is the same root node (for example, node i): it sends *m* bytes with *MPI_scatterv()*, *m/p* to each computer, and receives them with *MPI_gatherv()*. The scattered variable receives the gathered values. The latency was obtained dividing by two the time of the scatter plus the gather obtained on node i. In Figure 4.9, the root node i also receives data in scatter and sends in the gather, as in the scatter/gather communications of P-Si3D. The local root bandwidth is higher than the point-to-point bandwidth for the same size *m* because the root node receives in scatter and sends in gather *m/p* bytes (at the speed of a local memory copy). As it can be noticed the minimum latency (*L(p)* in Eq. (4.3)) increases and the maximum bandwidth (*B(p)* in Eq. (4.3)) decreases with the number of computers. Therefore, performance decreases when the number of computers increases for all message sizes, as it was pointed out (Section 4.2.1). The highest maximum bandwidth (reached with 2 nodes) is near twice the point-to-point maximum bandwidth, the lowest maximum bandwidth will be close to the maximum point-to-point bandwidth. In Figure 4.10, the root node does not receive or send data as in the scatter communications of SP-Si3D. The bandwidth is limited to the maximum point-to-point bandwidth (Figure 4.8) as expected (Section 4.2.1). Notice that the local root bandwidth (MB/s received or sent by the root node) coincides with the global bandwidth (MB/s through all the network).

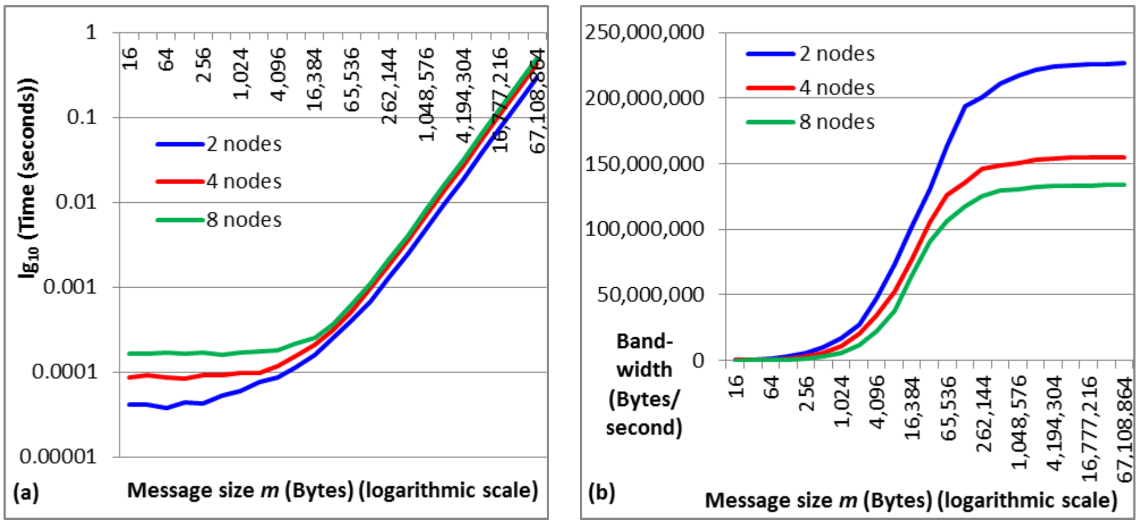


Figure 4.9. Scatter/gather performance in the test cluster for different number of nodes (computers). The node that scatters and gathers the data is the same and it takes a share on them (it receives data in scatter and sends in gather). This node measures the time of scatter plus gather. (a) Latency vs. message size *m* (time of scatter+gather divided by 2). (b) Bandwidth vs. message size *m*.

Figure 4.11 shows the performance of the interchange collective communication with MPI for different size and number of computers. Interchange is implemented with two *MPI_sendrecv()* functions (see Figure 4.4). The computers start the interchange after a barrier. All the nodes measure time (the measure starts after the barrier). The times obtained by the nodes for a particular size *m* are similar, the figure uses the mean value. As it can be observed, the performances are mostly equal for different number of computers. The minimum latency (75 µs) is higher than twice the point-to-point latency (30 µs) because the nodes cope with input and output messages in parallel and the switch manages packets from/to multiple nodes in parallel (data packets and also control packets), see also Figure 4.4. The maximum local bandwidth of a node (~206 MB/s) is almost independent of the number of computers. Figure 4.11(b) compares the local bandwidth with the point-to-point bandwidth multiplied by two. The local bandwidth is nearly twice the point-to-point bandwidth until the message size exceeds 64 KB. This reflects a change from an eager to a rendezvous flow-control protocol of MPICH. The rendezvous protocol requires interchange of packets between the sender and receiver before transferring data to make sure there is enough buffer space in the receiver for storing the data (note that the switch has to cope with a higher number of packets).

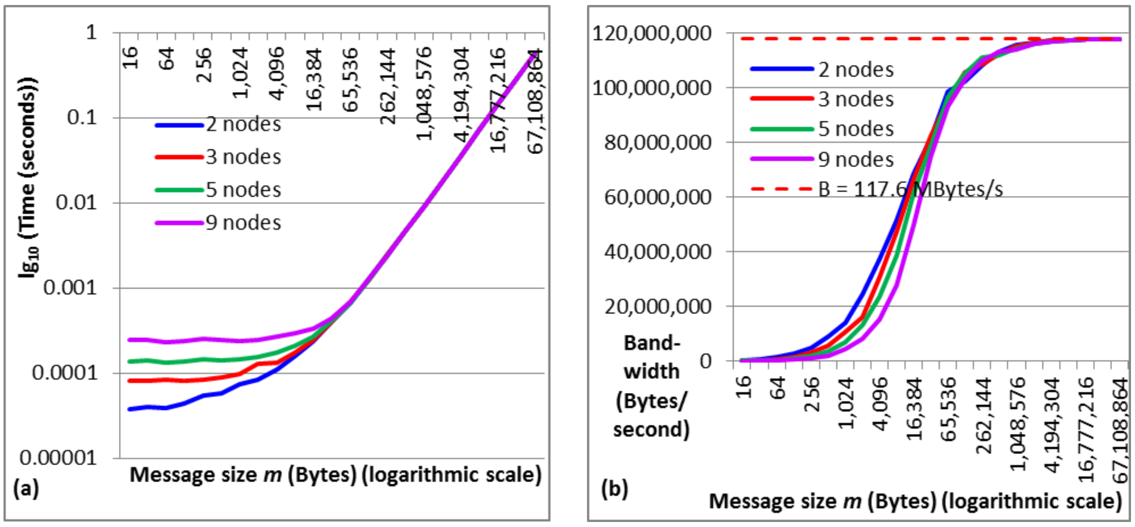


Figure 4.10. Scatter/gather performance in the test cluster for different number of nodes (computers). The same node scatters and gathers the data, but it does not take a share on them (neither receives in scatter nor sends in gather).This node measures the time of scatter plus gather. (a) Latency vs. message size *m* (time of scatter+gather divided by 2). (b) Bandwidth vs. message size *m*.

As Figure 4.12 shows the latency of both the scatter+gather and interchange are similar to the Ping-Pong latency (point-to-point latency multiplied by two) for large *m*. For small *m*, scatter+gather latency is greater than Ping-Pong and interchange latency and increases with the number of nodes. Moreover, as pointed out in Section 4.2.1, generally, the amount of data to be interchanged (C4) in P-Si3D will be less than the amount of data trasferred by the gather+scatter (C2-C3 in Figure 4.2 and Figure 4.3), so C4 time will be lower than C2-C3 time. Figure 4.12 also compares bandwidth, local of a node and global (i.e. bandwidth in the whole network). The local bandwidth of the root node in a scatter+gather communication with *m* bytes is also the global bandwidth because *m* is the total byte that the network transfers in both scatter and gather. The interchange local bandwidth is near twice the point-to-point (i.e. Ping-Pong) bandwidth, especially up to *m*=64 KB, because all nodes both send and receive *m* bytes in parallel. The global interchange bandwidth is near the point-to-point bandwidth multiplied by the number on nodes (5 in the figure), especially up to *m*=64 KB, because all the nodes send two messages, each one of *m* bytes.

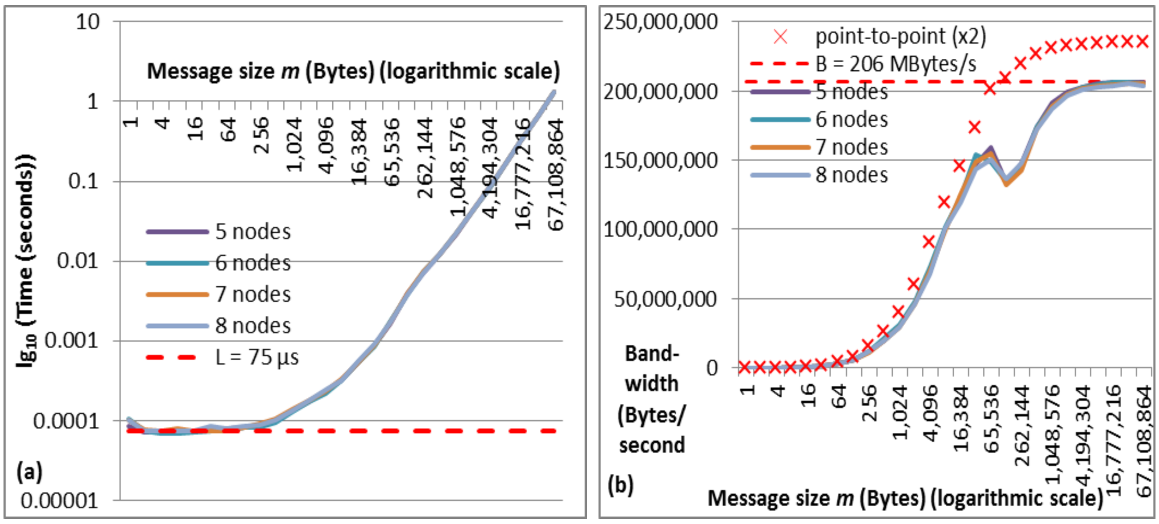


Figure 4.11. Interchange performance in the test cluster for different number of nodes (computers). The nodes start interchange after a barrier. All the nodes measure time (measuring starts after the barrier). (a) Latency vs. message size *m* and minimum latency *L*. (b) Bandwidth of a node vs. message size *m* (each node sends two messages and receive two messages, each one of *m* bytes), bandwidth of the point-to-point test multiplied by 2 and maximum bandwidth *B*.

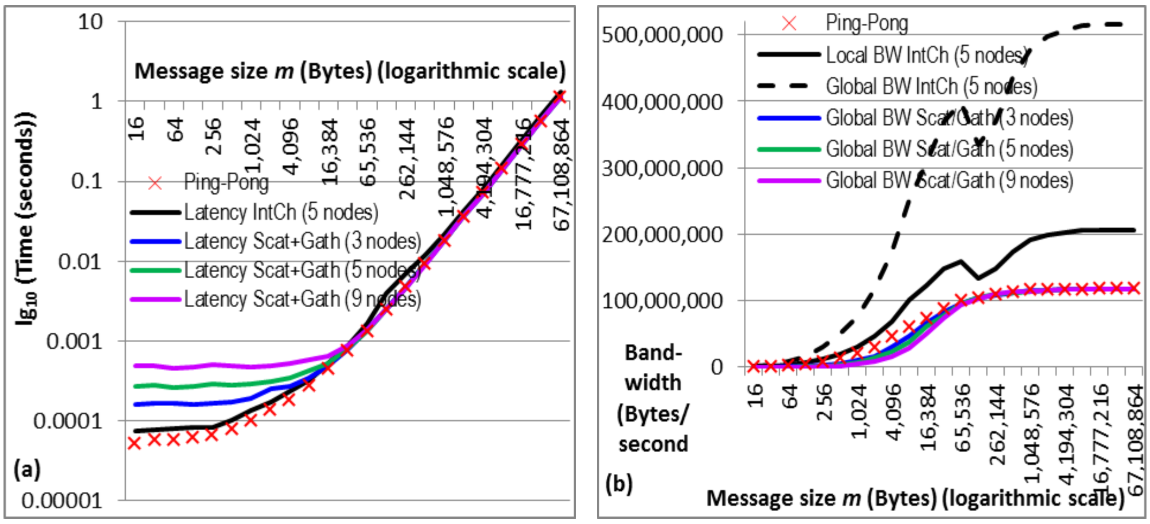


Figure 4.12. Comparison of Ping-Pong, scatter+gather and interchange performance in the test cluster. (a) Latency vs message size (b) Ping-Pong and interchange bandwidth for a node, interchange and scatter+gather global bandwidth (scatter+gather global bandwidth is also the local root bandwidth)

4.3.4 *SP-Si3D computing performance*

SP-Si3D was executed with a basin LR model (parent model) of 100 m x 100 m horizontal cells and a littoral HR model (nested model) of 20 m x 20 m cells (Table

4.2). SP-Si3D main memory requirements (basin LR model plus nested HR model in Table 4.2) decrease compare to a P-Si3D simulation of the basin HR model due to the reduction in wet columns and cells. Columns decrease a 56% and cells decrease a 74%. Due to the memory requirements, the basin HR model needs at least three computers to be executed with P-Si3D while the SP-Si3D can be executed in one computer of the low-cost cluster.

Section 4 pointed out that the number of S2 solver iterations for a particular tolerance depends on the equation system size. SP-Si3D equation subsystems decrease in size when the number of computers increase. Table 4.3 shows the number of iterations required by SP-Si3D to solve the equation system of the S2 stage in the parent LR model (2nd column) and the nested HR model (3nd column). It also shows the iterations required for each of the equation subsystems (one per subdomain) resolved when SP-Si3D is executed in eight computers (4th column). As can be noticed (see also number of columns in Table 4.2), the number of iterations decreases as the size of the S2 equation system decreases (littoral HR model in eight computers requires less iterations than in one computer, and LR model much less than any of them).

Step	LR model	Littoral HR model in 1 computer	Littoral HR model in 8 computers
LeapFrog	~11	~30	~26
Trapezoidal	~7	~21	~18

Table 4.3. SP-Si3D. Mean number of iterations of SP-Si3D PCG solver for the LR model, for the littoral HR model when executed in one computer, and for one HR subdomain when the nested model is executed in eight computers. All simulations use the same tolerance.

The scalability of the littoral HR model was studied up to eight computers comparing with one computer, using all cores in each computer. Speedup of the littoral HR model, $S^{HR}(p)$, for p computers is obtained as follows:

$$S^{HR}(p) = \frac{T_{\Delta t}^{HR}(1)}{T_{\Delta t}^{HR}(p)} \tag{4.7}$$

Where $T_{\Delta t}^{HR}(p)$ is the mean run-time of a time-step. Efficiency of the parallel littoral HR model, $E^{HR}(p)$, for p computers is obtained with:

$$E^{HR}(p) = \frac{Ideal\ time\ with\ p\ computers}{Actual\ time\ with\ p\ computers} = \frac{T_{\Delta t}^{HR}(1)/p}{T_{\Delta t}^{HR}(p)} = \frac{S^{HR}(p)}{p} \tag{4.8}$$

Speedup for SP-Si3D when the number of computers p assigned to the littoral HR model increases from 1 to 8 comparing with an execution in one computer of both HR and LR models, $T_{\Delta t}(1)$, is:

$$S(p+p^{LR}) = \frac{T_{\Delta t}(1)}{T_{\Delta t}(p+p^{LR})} = \frac{T_{\Delta t}(1)}{\max\left(T_{\Delta t}^{HR}(p),\, T_{C}^{LR/HR}(p), T_{\Delta t}^{LR}(p^{LR})\right)} = \frac{T_{\Delta t}(1)}{T_{\Delta t}^{HR}(p)} \tag{4.9}$$

$$p^{LR} = \frac{\#\ cores\ used\ by\ the\ LR\ model}{\#\ cores\ per\ computer}$$

where p^{LR} is the ratio between the number of cores used by the basin LR model and the number of cores in a computer. Efficiency is:

$$E(p+p^{LR}) = \frac{S(p+p^{LR})}{p+p^{LR}} \tag{4.10}$$

SP-Si3D run-time in one computer $T_{\Delta t}(1)$ is obtained running parent LR and nested HR models in parallel in the computer. The best single-computer time is used here, obtained when the 16 logical cores of a computer is used and both LR and HR are executed in 8 logical cores belonging each to a different physical core.

Figure 4.13 shows run-time per iteration $T_{\Delta t}^{HR}(p)$, speedup $S^{HR}(p)$, and efficiency $E^{HR}(p)$ for the domain-decomposition execution of the littoral HR model. Figure 4.14 shows run-time per iteration $T_{\Delta t}(p+p^{LR})$, speedup $S\ (p+p^{LR})$, and efficiency $E\ (p+p^{LR})$ of SP-Si3D. Alternative speedup (S') and efficiency (E') figures are also shown, comparing the SP-Si3D parallel run time $T_{\Delta t}(p+p^{LR})$ to $T_{\Delta t}^{HR}(1)$, the run time of the littoral HR model in one dedicated computer not shared with the LR model (i.e.

not including the LR model run time in the reference time used for the comparison). This figure also shows the run-time of the LR model depending on the number of cores used. The run-time decreases as the computational load is split among a larger number of computers. In all the simulations conducted, the littoral HR model was the most costly (pipeline stage "HR" in Figure 4.6(b)), independent of the number of subdomains used in the HR model (*p*=1…8). As Figure 4.13 shows, the speedup of the littoral HR model increases linearly with a constant slope of almost one (efficiency of near 100% in Figure 4.13), so it scales almost ideally and the run-time decreased almost inversely proportional to the number of computers used. As Figure 4.14 shows SP-Si3D also scales almost ideally as the number of computers increases, with efficiency (E(p´)) around 101%. Without taking into account the run time of the LR model, efficiency is slightly lower (E´(p´) in Figure 4.14). Note that efficiency is inversely proportional to resources used and that the resources (in particular, the number of cores of the parent LR model) can decrease using a higher time-step for the LR model (50 s instead of 10 s) as it was pointed out before.

Nested HR mean time per time-step, speedup (Eq. (4.7)) and efficiency (Eq. (4.8)):

p	Nested HR T $T_{\Delta t}^{HR}(p)$ (s)	Nested HR S $S^{HR}(p)$	$E^{HR}(p)$ %
1	20.8223	1.0000	100.00
2	10.4600	1.9907	99.53
4	5.2407	3.9732	99.33
8	2.6129	7.9790	99.61

(a)

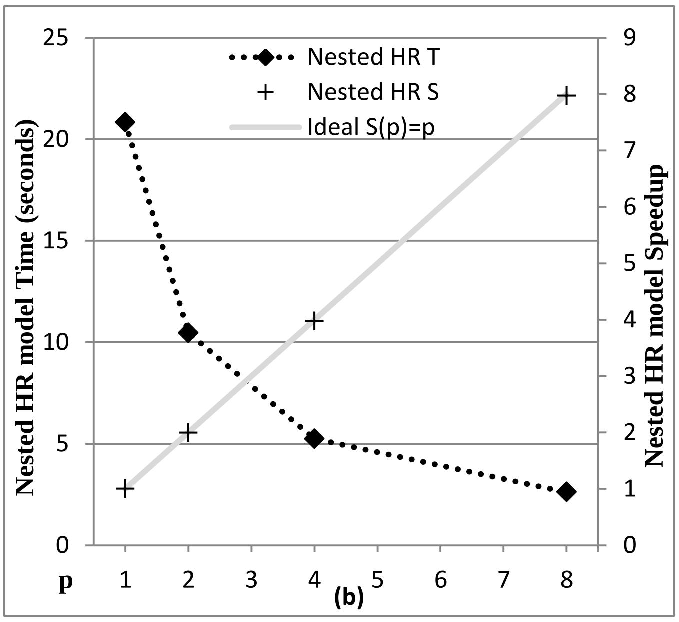


Figure 4.13. SP-Si3D. (a) Run-time per iteration, speedup and efficiency (in %) of the littoral HR model in Lake Tahoe. (b) Graph of time and speedup for *p=1,2…8* computers comparing with time in 1 computer (using all cores in each computer). The ideal linear speedup (speedup of *p* for *p* computers) is shown in the graph (gray line).

SP-Si3D time, speedup (Eq. (4.9)) and efficiency (Eq. (4.10)):

p'	SP-Si3D T $T_{\Delta t}$(p') (s)	SP-Si3D S S(p´)	E(p') %	SP-Si3D S´ S´(p´)	E´(p') %
1.000	22.5120	1.0000	100.00	0.9249	92.49
1.125	20.8223	1.0811	96.10	1.0000	88.89
2.125	10.4600	2.1522	101.28	1.9907	93.68
4.250	5.2407	4.2956	101.07	3.9732	93.49
8.500	2.6129	8.6157	101.36	7.9690	93.75

p'= p +p^{LR} when more than 1 computer is used.

LR time:

p^{LR}	#cores	$T_{\Delta t}^{LR}(p^{LR})$
0.125	1	7.04
0.250	2	3.73
0.500	4	2.16
1.000	8	1.15

$$p^{LR} = \frac{\#\, cores\ used\ by\ LR}{8\ cores\ per\ computer}$$

(a)

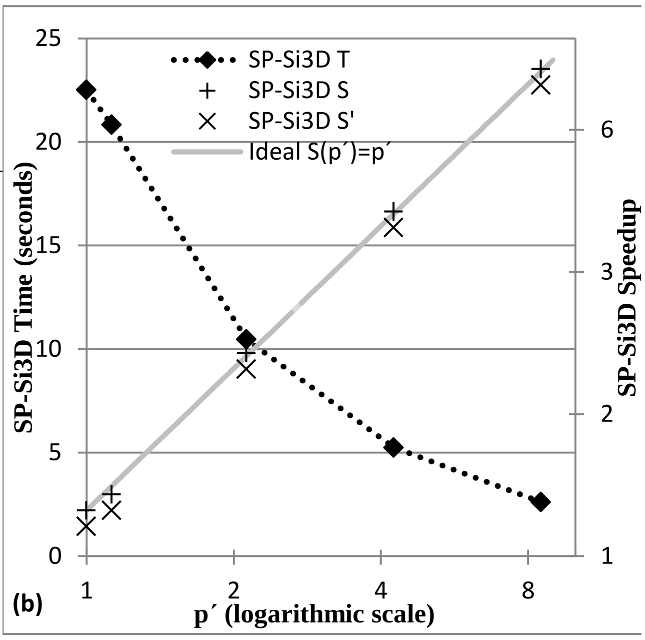


Figure 4.14. SP-Si3D. (a) Run-time per iteration, speedup and efficiency (%) in Lake Tahoe. Run-time per iteration for the LR model is also shown using 1, 2, 4 and 8 cores of a computer. The number of cores in the LR model for SP-Si3D is chosen in order to approach the HR parallel time. (b) Speedup. The ideal linear speedup is shown in the graph (gray line).

The speedup of SP-Si3D compared to the HR model of the complete basin executed in one computer would be super-linear (above ideal) due to the nesting. This speedup cannot be obtained because the basin HR model does not fit in the memory of a computer, requiring at least three computers. SP-Si3D achieves real time execution (a run-time per iteration lesser than the time-step of 10 s) with three computers, while P-Si3D does not achieve real time for the number of computers available in the cluster (the minimum run-time per iteration obtained by P-Si3D is 12.25 s).

4.3.5 Quality of the SP-Si3D results

The quality of SP-Si3D results depends on (1) the quality of the nesting implementation and on (2) the scalable parallel implementation of the nested HR domain. The nesting implementation was evaluated in Chapter 2 (Section 2.4). Results show that the differences between N-Si3D results and a HR execution of the entire basin are mainly due to the iterative PCG solver used to obtain free surface elevation (Table 2.4 in Chapter 2). No differences are obtained when the iterative PCG solver is replaced by a non iterative method and both the nested and the parent models have the same resolution.

The quality of the SP-Si3D results is evaluated qualitative and quantitatively in a region where there exist local-scale hydrodynamic features well resolved by a basin HR model but not by a basin LR model. Local-scale hydrodynamic features, such as flow separation and recirculation eddies occurring in the near-shore region, can only be well resolved in the basin HR model and not by the basin LR model. Global-scale features captured in both basin HR and basin LR models are similar, but differences in the vorticity fields reveal the location of features in the basin HR model that are not captured by the basin LR model. Marla Bay (Figure 4.7(a) shows its location) is here taken as a case example where recirculation is likely to occur as a result of flow separation. The vorticity field in Marla Bay at any given time t was computed from surface velocity predictions for the rectangular region in Figure 4.15, as follows

$$\omega(i+1/2\,,j+1/2) = \frac{v(i+1,j+1/2)-v(i,j+1/2)}{\Delta x} - \frac{u(i+1/2\,,j+1)-u(i+1/2\,,j)}{\Delta y} \qquad (4.11)$$

As Figure 4.15 demonstrates, with the nested HR model of the Marla Bay region shown in the figure, recirculation in Marla Bay is captured (d) in the same way that in the basin HR model (a), while the basin LR model (b) only captures a weak divergence in the velocity field. Being able to simulate these eddies in bays and other lake shore irregularities, is important in trying to understand coastal transport processes (Rueda and Vidal 2009). As a result of re-circulating eddies, bays can trap particles in suspension and other water constituents, hence, decreasing the longshore dispersion rates. This trapping effect has been reported previously in the literature; for example, Brooks et al. 1999 shows that eddies in Cobscook Bay, Maine, could trap particulates in the side-arms of the estuary. The local residence time of water within bays tends to increase as a result of recirculating eddies; hence becoming hot-spots for the reproduction of species looking for quiet conditions. Nishimoto and Washburn 2002, for example, observed high concentrations of juvenile fish in the center of a large eddy in the Santa Barbara Channel.

In order to show the quality differences between the results of SP-Si3D and those of a parallel implementation based in the execution of multiple simple nested HR grids in parallel (called here as MN-Si3D), one for each SP-Si3D subdomain, both SP-Si3D and MN-Si3D have been used with a subdomain boundary in the middle of the local-scale vortex of Figure 4.15(c). Figure 4.16 shows the results obtained (vorticity and velocity field). SP-Si3D output is very similar to the output of the basin HR model (Figure 4.15(a)), while MN-Si3D output is clearly quite different. The differences of variables (velocity, free surface elevation, temperature, vertical diffusivity) at the end of the simulation period between the simulation of the basin HR model and the alternative simulations to be evaluated (SP-Si3D, MN-Si3D, N-Si3D, basin LR sequential simulation) were quantified using a normalized form of the root-mean-squared error (NRMSE), usual in validation works (Debreu et al. 2012,Kourafalou et al. 2009,Pairaud et al. 2011,Son et al. 2011), calculated as follows

$$RMSE = \sqrt{\frac{\sum_{c=1}^{N}(x_b^c - x_n^c)^2}{N}} \qquad (4.12) \qquad NRMSE = \frac{RMSE}{x_{max} - x_{min}} \qquad (4.13)$$

Here x_b represents the value of a variable calculated by the basin HR model executed sequentially; x_n is the value calculated by the simulation to be evaluated (SP-Si3D, MN-Si3D, N-Si3D, basin LR sequential simulation); $x_{\max}$ and $x_{\min}$ represent the maximum and minimum of each variable; and N is the total number of water columns in the region of interest. The NMRSE errors in the area named “study area” in Figure 4.15 and Figure 4.16 for velocities (u, v), temperature (T), vertical diffusivity (K_v) and surface elevation ζ can be seen in Table 4.4. They are obtained for the Si3D simulation of the basin LR model (Figure 4.15(b)), the N-Si3D simulation of Figure 4.15(d), the SP-Si3D simulation of Figure 4.16(a) and the MN-Si3D simulation of Figure 4.16(b). The region simulated in HR with N-Si3D, SP-Si3D and MN-Si3D is the Marla Bay zone shown in the figures. The comparison of N-Si3D and SP-Si3D errors allows concluding that the SP-Si3D errors are due to the nesting implementation, the parallel implementation of the nested HR model do not introduced important errors. However, the errors of MN-Si3D are quite important, as Figure 4.16(b) also shows. Therefore, MN-Si3D is not that good as parallel implementation because the distribution of work among subdomains must care about where the boundaries between subdomains are established in order to avoid errors, making it difficult to obtain a good work-load balance among subdomains.

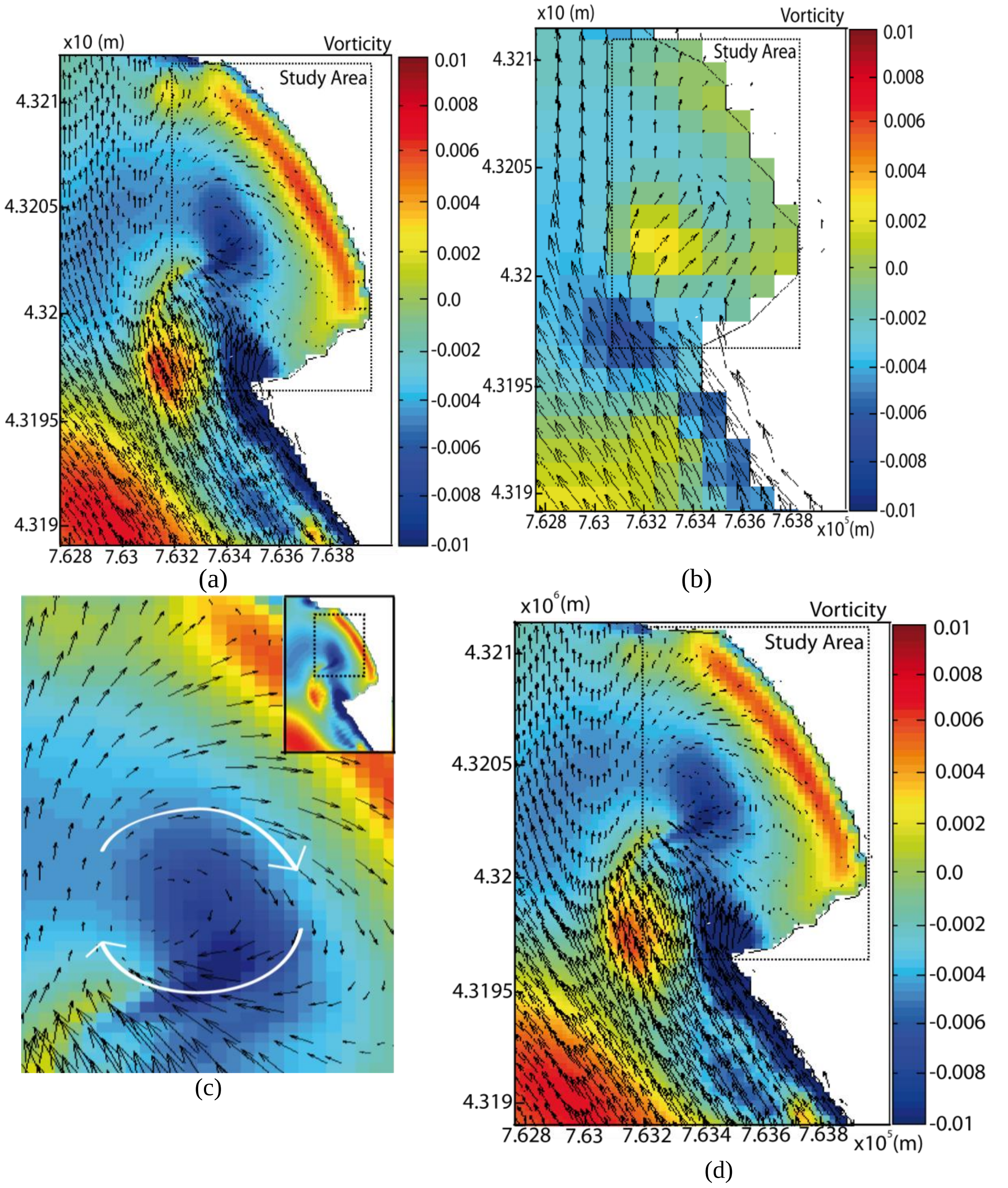
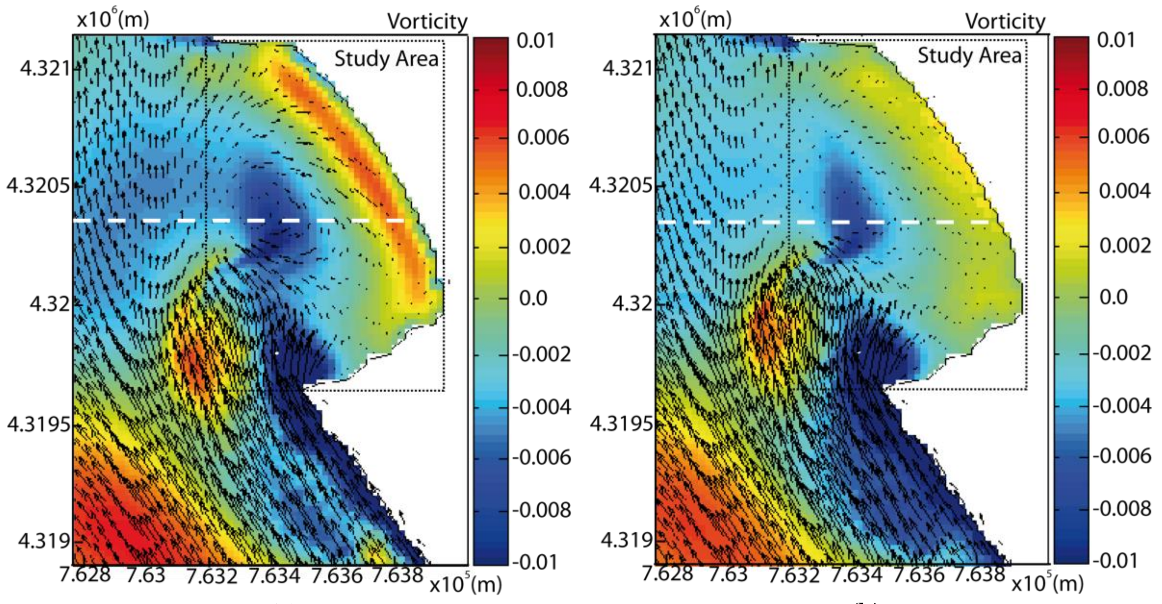


Figure 4.15. Vorticity (color scale) and u+v velocity field (black arrows) in Marla Bay area at a snapshot in time on Day 207. (a) Results from Si3D of the basin HR model, (b) Results from Si3D of the basin LR model (parent model), (c) Zoom of the captured local-scale vortex. (d) Results from N-Si3D, where the nested HR region simulated is just the region shown in the figure.



(a) (b)

Figure 4.16. Vorticity (color scale) and u+v velocity field (black arrows) in Marla Bay area at a snapshot in time on Day 207. (a) Results from SP-Si3D. (b) Results from MN-Si3D. Both results from SP-Si3D and MN-Si3D simulate in HR just the region shown in the figures, divided into two subdomains. The boundary between subdomains is highlighted by a white dashed line.

NRMSE (%)	**Si3D basin LR Figure 4.15(b)**	**N-Si3D Figure 4.15(d)**	**SP-Si3D Figure 4.16(a)**	**MN-Si3D Figure 4.16(b)**
$\boldsymbol{u}$	21.81	2.21	2.23	6.73
$\boldsymbol{v}$	24.46	2.14	2.15	6.51
$\boldsymbol{T}$	9.23	1.02	1.01	2.59
$\boldsymbol{K_v}$	8.46	0.61	0.61	1.66
$\boldsymbol{\zeta}$	13.61	1.87	1.89	5.41

Table 4.4. NRMSE (%), for velocities u and v, temperature T, vertical diffusivity K_v and surface elevation ζ, obtained for Marla Bay (region named "study area") in the surface layer with the basin LR model (Figure 4.15(b)), N-Si3D (Figure 4.15(d)), SP-Si3D with a boundary in the vortex (Figure 4.16(a)) and MN-Si3D with a boundary in the vortex Figure 4.16(b) all of them compared against the results of the basin HR model in this region (Figure 4.15(a)). N-Si3D, SP-Si3D and MN-si3D simulate in HR just the region shown in the figures.

4.4 Conclusions and future work

This work proposes a scalable parallel implementation with a two-level parallel structure, SP-Si3D, for semi-implicit hydrodynamic models, that achieves a scalable speedup in low-cost clusters taking advantage of a nesting implementation. Scalability is more difficult to achieve in low-cost clusters due to the lower performance of the network interconnecting the computers (latency and bandwidth) and the higher number of computers for the same number of processing cores (which increases mean communication time among cores). SP-Si3D has been obtained modifying P-Si3D, a parallel implementation of Si3D code (Smith 2006). This chapter shows that:

- P-Si3D scalability limit is mainly due to the sequential execution of the surface elevation solver and the required all-to-one/one-to-all communications.
- SP-Si3D run-time decreases almost linearly with the number of computers in a low-cost cluster. Thus, it scales almost ideally, i.e., its speedup (*speedup_with_p_computers = time_with_a_computer / time_with_p_computers*) increases with the number of computers used with a constant efficiency (*efficiency_with_p_computers = speedup_with_p_computers / p*) of approx. one. The scalability is achieved by executing in parallel the surface elevation solver avoiding, in addition, all-to-one/one-to-all communications; just one communication remains in SP-Si3D, an interchange communication, as in parallel implementations of explicit hydrodynamic models. The method proposed to improve scalability is also applicable to other (semi-)implicit and implicit-splitting models. Other parallel platforms, such as GPUs and high-end servers, can also benefit from the modification proposed.
- The time of the SP-Si3D interchange communication in a low-cost cluster does not depend on the number of nodes.
- The parallel processing structure of two-level proposed (pipeline plus domain-decomposition) allows the execution of the parent LR model and the nested HR model in parallel with an execution time equal to the execution time of the nested HR model. This two-level approach is also applicable to online nested implementations of explicit or implicit (splitting or not) hydrodynamic models.
- Lake Tahoe local-scale simulations with SP-Si3D achieve real time (a run-time per iteration smaller than the time-step of 10 s) with three computers, while with P-Si3D (which has all-to-one/one-to-all communications) does not achieve real time for the number of computers available in the low-cost test cluster and requires at least three computers for its execution due to the main memory required by the HR simulation of the entire basin.

- The modification applied to achieve scalability barely affects the quality of the results. The differences with the results obtained with a basin HR model are due to the nesting implementation, the errors of which are mostly due to the iterative solver used for obtaining surface elevation.
- The results show that the parallel implementation can reproduce recirculation not observed in the lower resolution model of the entire Lake Tahoe basin even if a subdomain boundary crosses the recirculation area.
- The implementation that divides the HR grid to be simulated into multiple nested HR sub-grids that are processed in parallel (named here MN-Si3D) is less recommended as parallel implementation because the boundaries between nested HR sub-grids must be chosen with care to avoid errors, making it then difficult to balance the work among computers.

The execution of much higher size models in high-end platforms could require the modification of the parent LR code so that it can be executed in more than one computer. To achieve this objective this code could use P-Si3D MPI implementation (only the OpenMP implementation was used here). The implementation can be extended to more than two levels of grids (parent grid + nested grid). In that case, the number of stages of the pipeline structure will depend on the number of grid levels.

Chapter 5

A hybrid parallel implementation of a 3D hydrodynamic model optimized and adapted to the architecture

Abstract

In this chapter, a number of proposals to obtain an optimized and scalable parallel implementation are presented. This is achieved by adapting the operations typically found in a 3D hydrodynamic model to the architecture used in both shared and distributed memory machines. The results show that the adaptation of the model to a NUMA architecture, and minimizing the overhead produced at points of communication/synchronization, significantly reduces the computational cost of the model (in both required execution time and memory). Good results can be obtained with these improvements even in low performance architectures, even showing that the optimization and reduction of communications allows us to obtain similar results with Infiniband and Gigabit Ethernet. It also shows that all stages must be optimized equally to obtain the best results, including the resolution of the long sets of equations characteristic of implicit models, solved by iterative methods such as the Preconditioned Conjuguent Gradient (PCG) whose parallel implementation is very complex. This work includes an efficient and optimized parallelization of the PCG in the same way as the rest of the model. This allows an implementation in parallel without adding a higher computational cost than in other stages, at the cost of reducing their convergence in finding an acceptable solution when the number of subdomains that run in parallel is incremented. However, this reduction is significantly enhanced by a modification to the Modified Incomplete Cholensky preconditioner developed and implemented in this work.

5.1 Introduction

In Computational Fluid Dynamics (CFD), the numerical simulation of water using 3D hydrodynamic models is one of the most challenging problems in engineering applications. In these simulations, small spatial scale patterns have a significant impact on large-scale circulation. It is therefore necessary to use large high-resolution grids which are able to properly simulate these patterns properly. Moreover, the time scale is also usually large, with small time-steps to avoid problems of stability, thus making it necessary to run these models a large number of time-steps for long periods of time to obtain useful simulation results. Because of these enormous requirements of timescale and spatial scale, most of the simulated phenomena produce costly simulations (in terms of execution time and the amount of memory needed). For critical situations, for example the study of floods in order to predict which areas might be affected and establish preventive measures in time, high execution times would be unacceptable. Given these requirements, the use of High Performance Computing (HPC) to reduce the computational cost is becoming an essential practice. These high-performance platforms not only reduce the computational time, but allow simulations with higher spatial resolution or repeat costly simulations for calibration and validation and, consequently, provide better results.

Different paradigms can be used in order to take advantage of these parallel environments. In the state of the art simulations of large-scale water, it shows that the message-passing paradigm is the option which is mostly used (Ecer et al. 1999, Fischer and Patera 1994 Fringer et al. 2006, Manzini and Stolcis 1999, Rao 2004, Semtner and Chervin 1988, Smith et al. 1992), where each processor has its own "local space name". The existence of a standard for this paradigm, Message Passing Interface (MPI) (Message Passing Interface Forum), and its ability to scale to a large number of processors (Gropp 2001), has contributed to its main use. However, the shared memory paradigm OpenMP (OpenMP Architecture Review Board) has also been used more frequently in recent years to efficiently exploit shared memory machines (SMP), which generally have NUMA Architectures (Non-Uniform Memory Access), where you can take advantage of the faster local memory of each processor compared to the memory of other processors to perform more efficient implementation, as well as its simplicity and lower computational cost when accessing data from other threads using shared memory.

Both paradigms have been compared in performance in the past. In (Luecke-Hua and Wei 2001) 7 tests were conducted (2 to measure communications and 5 kernels), MPI performed better than OpenMP in most cases. In another study to analyze the efficiency of a hydrodynamic model (Resch et al. 1999), it found that results with OpenMP had poor scalability compared to MPI for 8 processors. However, in recent years various studies have shown that OpenMP can be as competitive as MPI in SMP if explicit optimizations are developed by the programmer. (Norden et al. 2006) makes a comparison, without going into detail of how each implementation was done, between MPI and OpenMP in NUMA and UMA architectures concluding that OpenMP is as efficient as MPI, provided that care is taken with the initial placement of variables in memory (first-touch). (Amritkar et al. 2012) also showed optimal performance of OpenMP implementation in a SMP architecture of up to 256 cores when first-touch and appropriate affinity threads were used. Thus, the current trend is moving towards hybrid implementations that allow us to take advantage of OpenMP within a node and scalability of MPI in an environment of multi-cores nodes. However, to obtain good scalability in these architectures, it is necessary that the deployed application is optimized for both parallel paradigms, OpenMP and MPI.

Whether using OpenMP or MPI, optimizations are performed to obtain an efficient parallel implementation, which will largely depend on the type of model used and, therefore, the type of operations executed. Usually 3D hydrodynamic models use numerical solutions of partial differential equations based on the Navier-Stokes equations for shallow water (Shallow Water Equations, SWE, see for example Blumber and Mellor 1987, Chapman et al. 1996, Fringer et al. 2006, Smith 2006, Griffies et al. 2008). Typically, these equations are discretized in space using methods of finite volumes (i.e. Daoud 2008), finite elements (i. e. Ferrarin et al. 2008) or finite differences (i.e. Hodger et al. 2000) in grids which can be structured (i. e. Blumber and Mellor 1987) or unstructured (i. e. Fringer et al. 2006). On the other hand, the equations can be discretized in time explicitly, implicitly or the alternative usually used with a combination of both of them, the splitting methods. Some of these splitting methods are the mode-splitting scheme presented by Blumber and Mellor (1987), the semi-implicit approach presented by Casulli and Cheng (1992) or the time-splitting method presented by de Goede (1991). The explicit case is presented simpler than the implicit case, given the direct computation of the dependent variables which can be made in terms of known quantities. Conversely, in an implicit case the dependent variables are defined by sets of coupled equations, generating large 2D equation systems which typically have the structure of a definite positive pentadiagonal symmetric matrix. This system of equations can be solved in each time-step by iterative methods such as the conjugate gradient (CG) (Hestenes and Stiefel 1952), which is one of the most widely used because of its simplicity and good convergence ratio (Zou and Hoffman 1993).

These models based on the SWE are computationally implemented so that work can be resolved column by column of the grid. However, the computational cost can be very significant, even in the calculation of each column, for example having tridiagonal systems of equations to solve (Stone 1973) for each of the columns of the grid (in the case of 3D models). In addition, the calculation of each column is dependent on other columns (usually with their neighbors to the north, south, east and west) that must be resolved efficiently in a parallel implementation. Similarly, the implicit case for solving the system of pentadiagonal equations involves additional work in both sequential and parallel implementation. These systems of equations are solved by iterative methods like CG (Meyer et al. 1989, Hill 1990). Additionally, preconditioning techniques are often used to improve the convergence of the iterative method *(*Preconditioned Conjugate Gradient, PCG) which, while reducing the number of iterations to find the solution, may present a computational cost at or above the CG itself. In the case of a parallel implementation, regardless of how the load balance is done, each process or thread normally only knows the part of the equation system that has been calculated and additional procedures such as reordering, factorization and/or communication/synchronization, with its associated computational cost, must be applied by using a PCG in parallel. Moreover, in the parallel implementation of the PCG, it must be taken into account that (1) the instructions added for a proper parallelization (computational cost added) and (2) adding synchronization and data exchange too often at each iterative step, degrade performance on both the execution time of each of the iterations of the PCG (Amestoy and Duff 2000) and the number of iterations required to converge to an acceptable solution (Benzi 2002, and Van der Vorst Magoulu 2000 Naff and Wilson 2006, Teranishi and Raghavan 2007).

The choice of a suitable preconditioner when performing a parallel implementation is not a trivial task. In literature Jacobi and Block Jacobi (Pellissetti and Ghanem 2000) are among the methods most used in preconditioning hydrodynamic models, for its simplicity and relative ease in implementing in parallel without adding a significant additional cost compared to CG itself. Other more complex preconditioners such as Incomplete Cholensky (IC) or Modified Incomplete Cholensky (MIC) greatly reduce the number of iterations needed to converge to a certain tolerance, but require a much higher computational cost per iteration and their implementations are difficult to perform in parallel. (Benzi 2002 Eijkhout 1992) present a parallel implementation of the preconditioner MIC using red-black ordering where scalability of the implementation is reduced by increasing the number of processes due to increased communication. (Jiaquan et al. 2013) describes a parallel implementation of MIC on GPUs using a technique called *wave front*, where all the threads have full access to the pentadiagonal system to solve step by step those equations that do not present unresolved dependencies. This method, however, would not be efficient in distributed architectures. Other works have PCGs (Benzi 2002 Husbands and Yelick 2007, 2003 Kim and Im, and Van der Vorst Magoulu 2000, 2006 and Wilson Naff, Teranishi and Raghavan 2007) implemented in parallel which require preparation of the system of equations with additional factorizations of the LU or LDLT type and/or a new reordering of the equations (such as red-black ordering). Though a parallel implementation is reached using these methods, it hinders implementation, adds computational load and requires more communication between subdomains to solve the dependences. Furthermore, many proposals found in literature solely presented the parallelization of the PCG, regardless of what type of application will be used. This can present possible additional imbalances because the workload is usually divided in these types of hydrodynamic models taking into account all stages of the model, and not just the pentadiagonal system of equations. This could mean that certain proposals, although they present an efficient parallel PCG, are not efficient enough when they are included in a hydrodynamic model, having to adapt the parallel PCG performed with new reordering and/or communications.

Given the importance of a good load balance, another of the critical points in the development of an efficient parallel 3D hydrodynamic model is how the work is divided. Domain decomposition is generally the proposal mostly used. The type of decomposition used, and the procedures that will be needed to add to perform the exchange of information between subdomains, will produce an added overhead in the parallel implementation, both in the number of communications and the amount of data to be communicated. The implementation of distinct alternatives of domain decomposition in various parallel architectures has been widely studied (Bjorstad et al. 1993, Chan et al. 1990, Hackbusch 1991 Keyes 1987) testing different types of cuts (horizontal or vertical to the grid) and the complexity of each cut (from simple cuts in one direction, reducing interaction with other subdomains, to irregular cuts in two or more directions, increasing the interaction with other subdomains). In this work (Chapter 3, Section 3.3.4) it is also studied how the type of cut affects to the performance in a parallel implementation. It is also possible to use tools like ParaMETIS, which automatically allows domain decomposition (Karypis 2000).

Making an optimal domain decomposition of grids with irregular geographies can be a complicated task. This is because to spread the workload as evenly as possible, it could also increase the amount of data to be communicated and the number of subdomains that must interact with each other (by increasing the number of cuts between neighbors for example). Delis (2009) and Yu (2010) compared different domain decompositions with different ways of distribution work. They concluded that simple cuts in one direction (vertical or horizontal) obtained the best results in producing a load balance. Simple cuts were only slightly more unbalanced compared to other decompositions with various cuts and conversely, reduced the amount and the number of communications significantly.

Considering all the above-mentioned points, the parallel implementation of a 3D hydrodynamic model is not a trivial task, where you must consider the type of model to be parallelized and how to take advantage of the architecture in which the model will be efficiently executed. This work presents OP-Si3D, the optimization of the parallel implementation of a 3D semi-implicit model based on the SWE using domain decomposition. The implicit or semi-implicit models are harder to parallelize than the explicit cases (Naik 1993). This is because during the execution of each time-step a large system of pentadiagonal equations of the entire domain to obtain water surface elevation must be solved and its implementation in parallel is not a trivial task (Grindbaum 1998). In this chapter it is intended to emphasize those tasks, methods and tools necessary to take full advantage of a hybrid parallel hydrodynamic model, making an efficient use of available resources and getting good results from both shared and distributed memory machines. This will not only achieve greater scalability and efficiency in high performance architectures, but also achieve results almost as efficient with low cost machines connected with to cheaper interconnection networks.

Improvements in this chapter to adapt the parallel implementation to the architecture and reduce the overhead caused by a parallel implementation include:

- How to implement procedures such as first-touch and an efficient affinity map, showing that distinctive affinity options offer different results depending on the type of architecture used.
- Synchronization points common between threads (ompbarrier) are replaced by our own more efficient implementations, adapted to a NUMA architecture and to the domain decomposition used.

- We show that algorithms typically used in linear algebra such as the Thomas algorithm for solving tridiagonal system of equations can reduce their executional time, with basic optimizations such as loop unrolling.
- Alternative storage variables are presented in the case of threads, showing that the use of data locality cannot be efficient if the storage of variables in threads is not properly implemented.
- All communications are overlapped with computation, performing the computation of the border columns before the rest of the interior columns of each subdomain, showing that similar results can be obtained with Infiniband and Gigabit Ethernet despite the volume of data to be communicated.
- We show that the efficiency of implementation can be improved if the number of synchronization/communication points is reduced and redundant computation is added.

As domain decomposition method is used the implementation with vertical cuts in one direction explained in Chapter 3 and presented in other works (Passoni et al. 2001). This decomposition leads to load imbalances of lower than 0.5% even in large high resolution grids, and greatly reduces the amount and number of communications (each subdomain has to communicate with a maximum of two other subdomains). The same domain decomposition is used even during the parallel calculation of CG and of the preconditioner, maintaining the same type of communication as in the rest of the model (with a maximum of two neighboring subdomains) and avoiding additional procedures such as reordering. Similarly, distinct preconditioner alternatives are evaluated, presenting methods traditionally used, such as Jacobi or Block Jacobi, and implemented efficiently. Additionally, a modified version of MIC is presented. This modification has not been found by the authors in the literature, and it can be easily implemented in parallel without any additional computational cost with respect to a sequential MIC, reducing the loss of efficiency thanks to the presented modification and performing better than the second best method evaluated, Block Jacobi, in a machine up to 256 cores.

5.2 OP-Si3D implementation

The 3D hydrodynamic model used here was originally developed by Smith (2006) and later adapted for simulation in lakes (Rueda 2001). Si3D is a public code programmed for serial architectures by the US Geological Survey (USGS), which provided us with a free version.

The optimized parallel implementation proposal here is applied to Si3D (more details about the model description can be found in Chapter 1), based on the code provided by the USGS. This implementation is called here OP-Si3D.

5.2.1 Basic optimizations

From the original sequential algorithm, various optimizations have been applied in order to reduce and avoid unnecessary computations, to facilitate compiler optimization of the code and reduce the amount of memory required. The optimizations include:

- Deleting unnecessary initializations for each time-step.
- Fusion of loops.
- Replacing explicit loops with implicit loops, implemented as a standard in Fortran and recommended to facilitate the optimization of the program by the compiler.
- Re-using the same calculation variables at different stages.
- Padding in some loops.
- Elimination of unnecessary copies, avoiding some variables being calculated into an array and then copied to another just for reading in later stages.
- Replacing the calls of small subroutines directly by the code.

Additionally, we have implemented an optimization to the standard algorithm known as the Thomas algorithm, widely used to efficiently solve tridiagonal systems of equations (Thomas 1949) and that in Si3D it consumes over 50% of the execution time of each time-step (see Section 5.3.4) . The Thomas algorithm works with each row (equation) of the system, similar to the Gaussian elimination method. In the first-step a forward substitution is done, where each row is replaced by a suitable linear combination of rows so that the lower diagonal elements are eliminated and the main diagonal elements are equal to 1. Once the upper triangular matrix is obtained, the algorithm applies a backwards substitution to solve the unknown variables. Although the Thomas algorithm is very economical and grows linearly with the number of unknown variables, this is still computationally expensive because of the two-step procedure used. To facilitate optimization and the efficient use of the architecture, the Thomas algorithm has been modified to include a loop unrolling.

5.2.2 Data distribution and parallelization

Si3D uses about 250 variables to store 3D model information. For this, arrays are used that only store information of columns with water, this allows us to reduce memory costs by not having to store information of dry columns and reduce the computational cost by not having to assess whether a column is dry or not before performing a calculation. The new variables are one or two dimensional arrays, one dimensional for 2D variables defined in the *x* and *y* directions such as water surface elevation, two dimensional for storing 3D variables defined in the *x, y* and *z* directions such as vertical velocity, with the first dimension used for *x* and *y*, and the second for *z*. Access to neighboring columns (north, south, east or west) is achieved by 4 one-dimensional arrays (one for each direction). These arrays avoid extra operations to find the neighbors of a given column while supposing a miniscule memory cost, for example only 0.05% of total memory is required in the simulations of the high-resolution model used in this chapter.

Each one of the Si3D stages (except in the stage where the solution of the pentadiagonal system of equations is performed, further details of the numerical model can be found in Chapter 1) and, therefore, the calculation of the governing equations, are done column by column as per the original Si3D model developed by Smith (2006). This facilitates data locality in the calculation of each stage of the model, such as the resolution of the tridiagonal systems that are formed for each water column in 3 points (or more than 3 if passive tracers are used) for each Leapfrog or Trapezoidal step. The tridiagonal system calculations represent more than 50% of the execution time in all cases studied (see Section 5.3.4). As mentioned, the execution of Si3D is done column by column, advancing along each horizontal row of the grid from the southernmost column to northernmost column, beginning at the same time with the most westerly row of columns and ending with the most easterly. Taking this into account for the management of data locality, the variables are stored so that the information in each column is contiguous in memory and neighboring north and south columns are contiguous too. This ensures that when you access a column of the domain and the data is brought into a block memory cache (the fastest memory), this block includes data from the entire column and other neighbor columns that are then calculated immediately, so avoiding penalties of higher computational cost because of having to access the main memory too often.

Given the way Si3D works, just like other existing hydrodynamic models, the model can be parallelized using domain decomposition, assigning a specific number of columns to each subdomain taking into account an overlapping area to resolve dependencies during the calculation. In this work, as well as others (see for example Yu 2010), we have chosen a domain decomposition with vertical cuts in one direction, similar to those in Figure 5.1. This is done by assigning a group of consecutive grid horizontal rows to each subdomain and reducing to a maximum of only two overlapping subdomain areas to resolve dependencies with neighbors to the east and west.

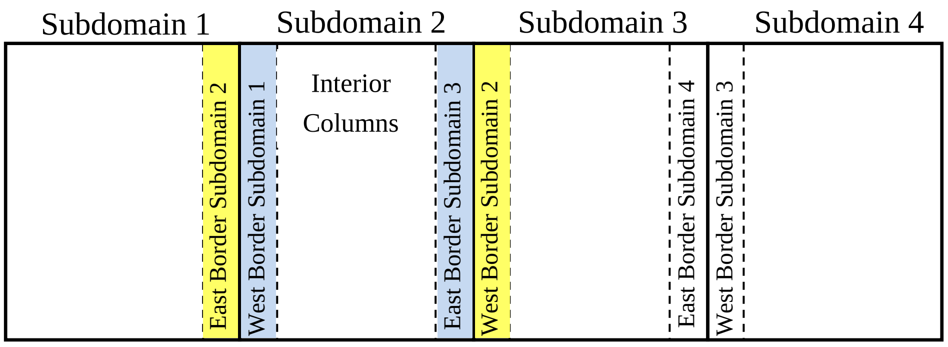


Figure 5.1. Domain decomposition by horizontal rows in Si3D, each subdomain consists of those columns within its subdomain (Interior Columns) and those representing the boundaries of other subdomains to the east and west (blue area). The yellow area represents the dependencies of a subdomain with its neighbors (overlapping area), solved by adding communication and/or synchronization.

For example in Figure 5.1, the subdomain 2 consists of those columns of water known as Interior Columns and additionally the columns which are shown in blue. On the other hand, the data outside the subdomain, but which is necessary to solve the dependences, will be considered as areas of overlap, represented in the figure with yellow, the blue borders represent overlapping areas of subdomains 1 and 3 respectively. Thanks to this type of domain decomposition, the communication/synchronization between two subdomains is reduced to only one area of overlap. This also ensures that the data to be communicated is contiguous in memory (given the way the variables are stored). Besides, this same domain decomposition can be used in the parallelization of the Conjugate Gradient (as explained in Section 5.2.6) in spite of which at this stage the calculation is not column by column of the grid as in the other stages. Although the domain decomposition is done by assigning a number of grid rows to each subdomain, the distribution is made taking into account the depth (number of cells) of each column. Taking into account this, a group of variable rows is assigned to each subdomain, trying to balance the number of cells that are distributed among subdomains as equally as possible. This type of domain decomposition has shown minimal imbalance, less than 0.5%, for all cases tested. The final Si3D parallel model (OP-Si3D) is presented in Figure 5.2, adding those points of communication and synchronization required to resolve the dependencies. S1, S2, S3 and S4 represent the solver stages of Si3D (see Chapter 1 for more details). C1, C2 and C3 represent the necessary communications/ synchronization between computers (communications within S2 will be explained in Section 5.2.6), including those variables that have to be communicated. OP-Si3D is based on the parallel implementation for small clusters (P-Si3D) presented in Chapter 3 (Figure 3.1).

5.2.3 Implementation of the affinity map

Affinity allows using software to run a particular thread or process within specified processing resources. This allows us to take advantage of NUMA type Architectures, which are common in most current machines, where the access to processor memory related to the process or thread is faster than accessing memory from other processors. Affinity has been studied and recommended for this type of architecture in the past and, however, still not commonly used in the field of Computational Fluid Dynamics, probably because of the need to prepare the code explicitly to get the maximum good affinity performance. This paper explains in simple steps how to extract the maximum affinity performance from any parallel hydrodynamic model using domain decomposition. This approach has reported a reduction of over 54% in the runtime in the case of threads in a shared memory machine.

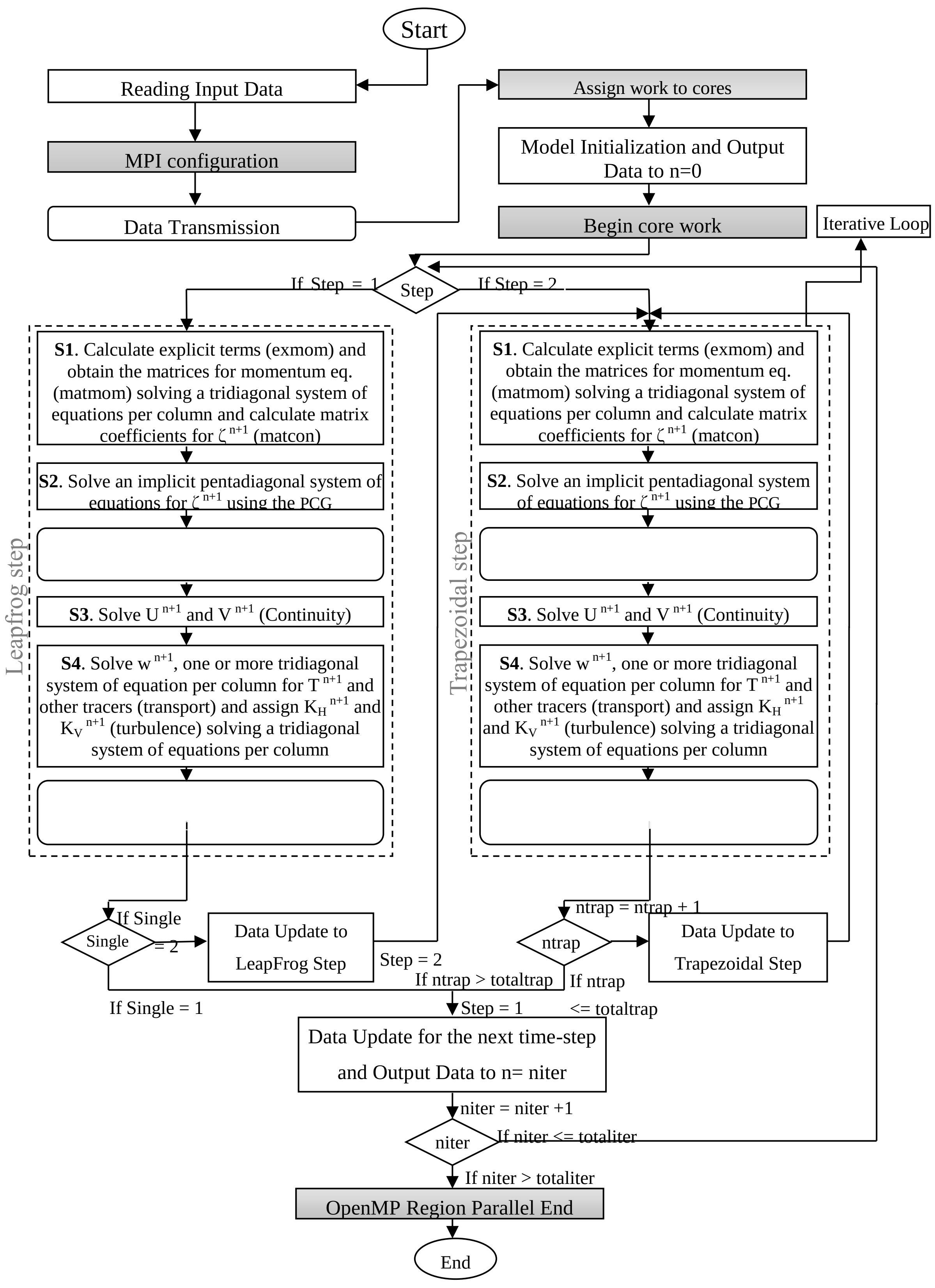

Start
Reading Input Data
MPI configuration
Data Transmission
Assign work to cores
Model Initialization and Output Data to n=0
Begin core work
Iterative Loop
Step
If Step = 1
If Step = 2
Leapfrog step
Trapezoidal step
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ n+1 (matcon)
S2. Solve an implicit pentadiagonal system of equations for ζ n+1 using the PCG
S3. Solve U n+1 and V n+1 (Continuity)
S4. Solve w n+1, one or more tridiagonal system of equation per column for T n+1 and other tracers (transport) and assign KH n+1 and KV n+1 (turbulence) solving a tridiagonal system of equations per column
If Single = 2
Single
Data Update to LeapFrog Step
Step = 2
If Single = 1
ntrap = ntrap + 1
ntrap
Data Update to Trapezoidal Step
If ntrap > totaltrap
If ntrap <= totaltrap
Step = 1
Data Update for the next time-step and Output Data to n= niter
niter = niter +1
niter
If niter <= totaliter
If niter > totaliter
OpenMP Region Parallel End
End

Figure 5.2. Simplified flow diagram for OP-Si3D. Gray boxes represent processing added for the parallel implementation. C1, C2 and C3 are communications among computers. S1, S2, S3 and S4 are Si3D stages.

The NUMA architectures are organized by a topology of two or three levels of hierarchy, with the number of cores per socket on the first level, a number of sockets per node on the second level and, finally for large shared memory machines, a third level with multiple nodes. OP-Si3D makes use of the hwloc tool to recognize this architecture (number of cores per socket, number of sockets per node and number of nodes) and organizes data accordingly. The procedure consists of using affinity to bind each thread to a particular core, also in placing threads with neighboring subdomains in neighboring cores, as shown in Figure 5.3. In the case of processes this affinity scheme is also used, ensuring that neighboring subdomains are found in the nearest neighboring core. This organization ensures that only a maximum of two subdomains per node will have to perform explicit communications with other nodes in distributed memory machines or access the overlapping areas of subdomains in other nodes in shared memory machines in a slower way. For the example shown, the communications between different nodes is reduced to only one between subdomains 4 and 5, the remaining subdomains rapidly access their data and ensure that neighboring subdomains are found in the shortest possible distance.

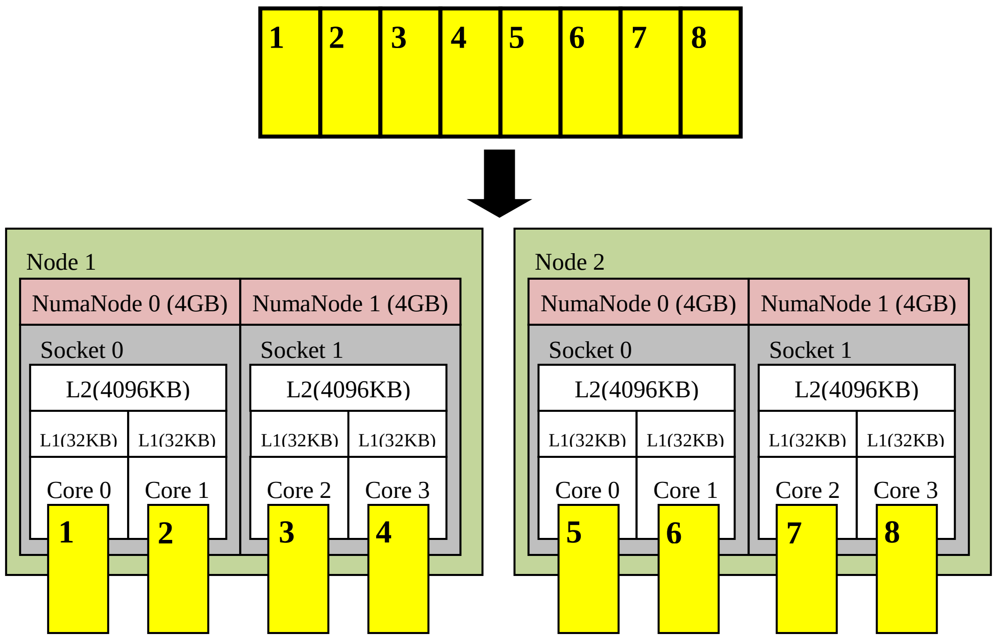


Figure 5.3. Example of NUMA architecture with two nodes, each node has two sockets, and each socket has 2 cores. The domain is decomposed according to the number of resources available and each thread or process is assigned to a core. This is always done consecutively by placing neighboring subdomains in the architecture.

To make this possible efficiently, each thread must initialize the data of its subdomain (first-touch) to ensure thses data is taken to its part of the memory. Additionally, the data is stored in memory alignments to 4KB, and also with a minimum size of 4KB (including control variables). This is done to ensure that when a memory block is stored in cache memory it contains only information from a particular subdomain and, at the same time, another thread cannot invalidate this data by writing in another area of the same block. In the case of processes, only the use of a suitable affinity map will have a positive effect on performance. This affinity map accompanied by the domain decomposition facilitates communications. In the example shown and assuming that the nodes are found in different machines, only subdomains 4 and 5 will have to make communications across an interconnection network. Besides, it is ensured that for any case with more nodes and subdomains, only a maximum of two subdomains per node will have to communicate with other nodes, as may be the case in subdomains 4 and 8 in the example shown if a third node was used.

In the case of threads, the type of variables used (shared or private), and the distribution of the data, affect the performance of the affinity procedure. One of the advantages of using threads is the availability of shared memory, which allows communication between threads to be more simple and efficient. It is therefore necessary to allow, whenever possible, the use of shared memory to obtain information on the overlapping areas of other subdomains, this is possible without loss of efficiency when accessing variables as read-only mode, so those variables that are written in one time-step, and the next one are only accessed in read-only mode, are created as shared variables. So for these variables each thread accesses to read and write to its part of the subdomain during its calculation and only in read mode to the overlapping areas of other subdomains in successive time-steps. However, there are many other calculations that are accessed by both modes (reading and writing) in the same stage and can cause two neighboring threads to compete for the same memory block, causing additional penalties and having access to slower memory to take the updated data. Several alternatives have been tested to determine the effect of these variables on performance, and to find which one offers the best performance.

In addition, two alternatives are evaluated to do the calculation of each stage and to solve dependencies with other subdomains. During S1, S3 and S4 of OP-Si3D (Figure 5.2), there are certain points where other subdomains need information from neighboring columns to the east or west immediately afterwards to calculate some variables. This has been solved in the first alternative by using communication in the case of processes and by synchronization in the case of threads, so that the stage is not continued until the required calculated information is available for another thread or process. However, this implies adding numerous points of communication/synchronization that may reduce performance due to the active blocking. A second alternative to replace these points of synchronization/communication with redundant calculation is evaluated, so that the same thread or process performs the calculation of the overlapping area, which also avoids active blocking between threads or processes.

On the other hand, three storage alternatives have been evaluated for the calculation variables in the case of threads.

- V1: A single shared variable size of the entire domain, in this case each thread must access the information calculated by other threads in that variable when it needs information from the overlapping area, directly accessing data from other subdomains.
- V2: A single shared variable where, in addition to each subdomain, it is adding the overlapping area and filling or garbage so that the total number of bytes for each variable is a multiple of 4KB, ensuring that the block with the subdomain of each variable can not be accessed by another thread. For this case, it is necessary to use redundant computation.
- V3: A private and distinct variable for each thread of subdomain size and adding only the overlapping area, avoiding any penalties for data access between threads and reducing the amount of memory used. For this case, it is necessary to use redundant computation.

5.2.4 Synchronization points

Synchronization points, barriers in the most common case, specify a point in the execution flow of a parallel program where all the threads or processes must wait until they all reach that particular point. In the case of a parallel hydrodynamic model, they ensure that the necessary dependencies needed to continue the execution of a particular subdomain have been resolved and this information is accessible to another thread or process.

In the case of processes, the most efficient is to use the same communications as synchronization, using blocking communication or both unblocking communication and mpi_wait to ensure that data is available for a given process before continuing with the execution.

In the case of threads, the most common procedure is the use of barriers that prevent further execution until all the threads pass by that point. Although each software already offers its own barriers (ompbarrier for example), the need of a generic implementation to work in the same way in any architecture makes these barriers non-optimal for each particular architecture used, as in the case of a NUMA architecture, where the memory is organized into levels. Si3D presents two points where all threads are synchronized at the same time (before and after S2) when a sequential Conjugated Gradient is used (more details of this implementation are explained in Chapter 4, Section 4.2.1). To accomplish this, ompbarrier uses a unique variable lock that all the threads must access (Figure 5.4 (a)), so that all threads compete for the said variable regardless of where the socket or node is found (Al Khalissi 2013). However, here we have implemented a barrier where there is a variable lock per level, which is only accessed by the threads of the same socket on the first level, one thread per socket on the second level and one thread per node on the third level (Figure 5.4 (b)). In addition, these barriers have been implemented so that in compilation time the compiler itself can evaluate how many levels are available in the architecture, so that it is possible to eliminate the computational cost of additional variable locks if any of the levels do not exist. Similarly, these variables locks are 4KB in size, ensuring that no other thread invalidates the data by accessing another area of the block containing each variable lock.

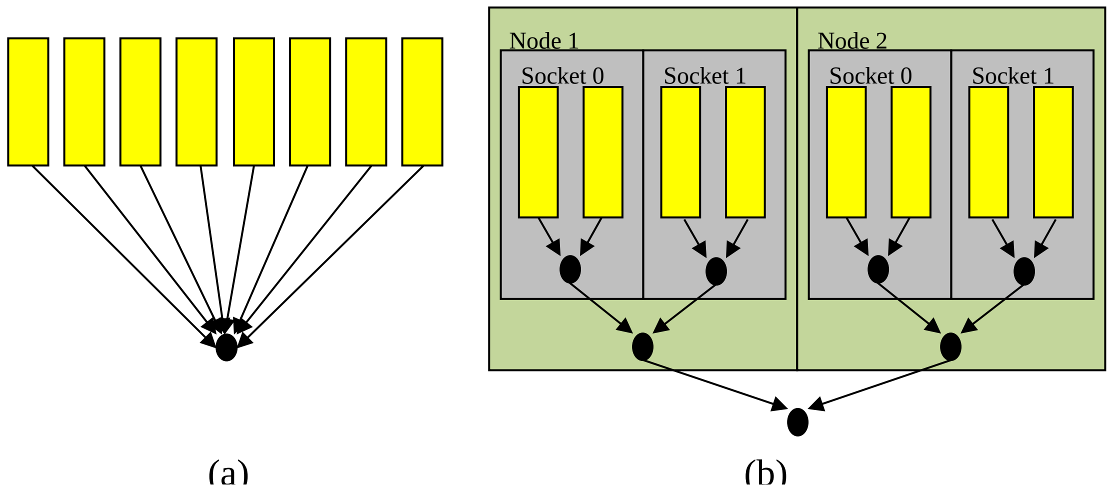


Figure 5.4. (a) Classic ompbarrier barrier, each thread updates a shared global variable lock. (b) Levels barrier. There is a variable lock size of 4KB per level. In level 1 all the threads in the same socket are blocked, in level 2 the threads within the same node and finally in level 3 all the threads.

On the other hand, during S2 it is necessary for some kind of synchronization that allows to solve the three existing Dot-product operations in any Conjugate Gradient (Figure 5.5, stages 2.2 and 2.5) so that the threads do not continue their execution until the final result of the Dot-product is available. Although OpenMP offers an efficient solution with clause reduction, this is not found to be adapted to a NUMA architecture. With clause reduction the partial result obtained for each thread is added to a single global variable (Bliznak et al 2014), presenting the same problem as the general ompbarrier discussed above.

The alternative presented here allows the use of a structure similar to the new implemented barriers, combining it with the Dot-product operation so that each partial sum is done by using variables per level, avoiding all threads competing to make the global sum in a single variable.

Finally, thanks to the domain decomposition used, the dependencies of each subdomain are reduced to the areas of overlap (Figure 5.1), so although synchronization points in the flow of execution are required for all subdomains, the dependencies and consequent blocking of a subdomain depends only on the two neighboring subdomains to the east and west. This means that each blocking may be selectively made between each pair of neighboring subdomains, so that synchronization is reduced between three subdomains (since each subdomain has two borders (Figure 5.1), it must synchronize itself with its two neighboring subdomains) in these points of synchronization.

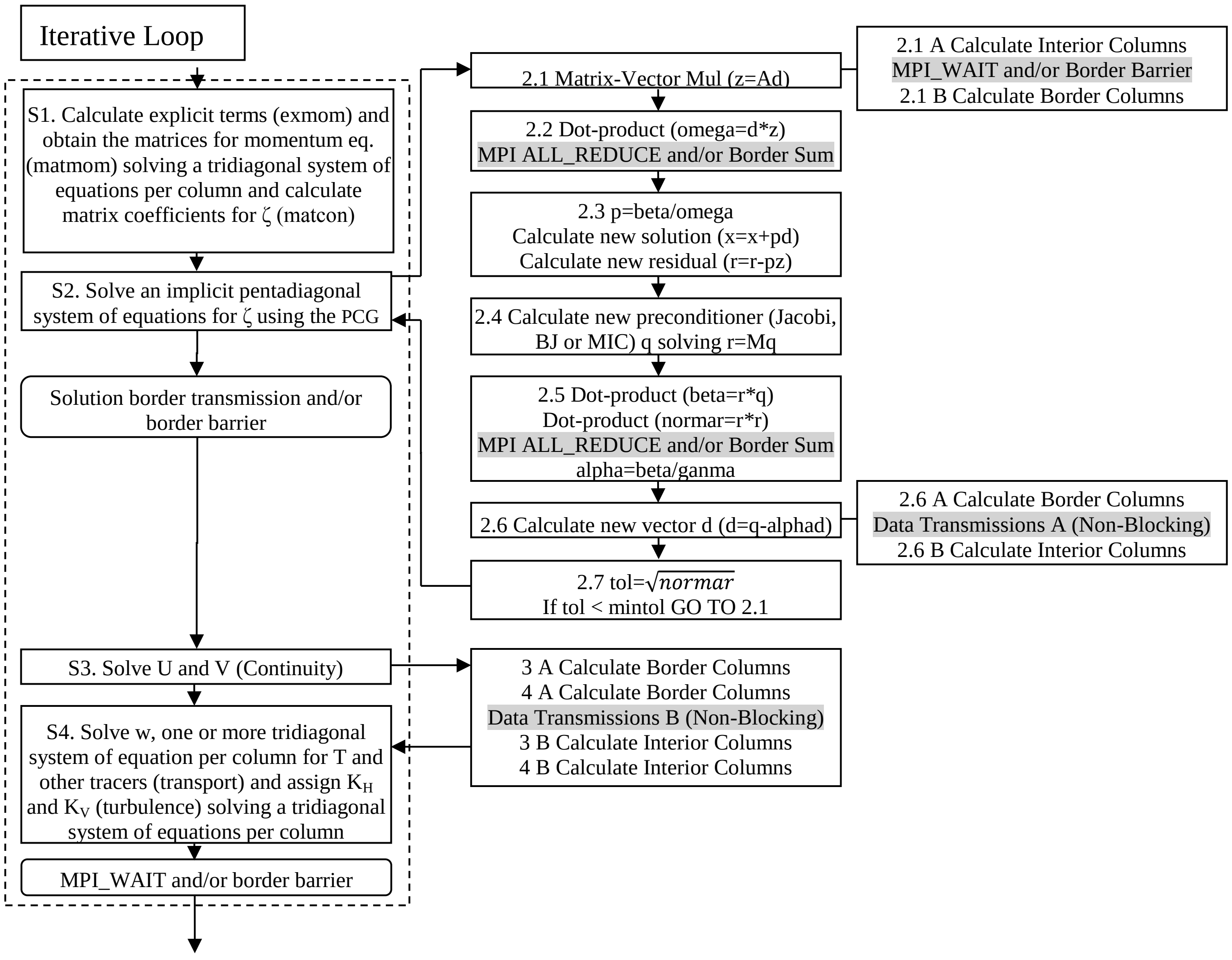

Iterative Loop
S1. Calculate explicit terms (exmom) and obtain the matrices for momentum eq. (matmom) solving a tridiagonal system of equations per column and calculate matrix coefficients for ζ (matcon)
S2. Solve an implicit pentadiagonal system of equations for ζ using the PCG
Solution border transmission and/or border barrier
S3. Solve U and V (Continuity)
S4. Solve w, one or more tridiagonal system of equation per column for T and other tracers (transport) and assign K_H and K_V (turbulence) solving a tridiagonal system of equations per column
MPI_WAIT and/or border barrier
2.1 Matrix-Vector Mul (z=Ad)
2.1 A Calculate Interior Columns
MPI_WAIT and/or Border Barrier
2.1 B Calculate Border Columns
2.2 Dot-product (omega=d*z)
MPI ALL_REDUCE and/or Border Sum
2.3 p=beta/omega
Calculate new solution (x=x+pd)
Calculate new residual (r=r-pz)
2.4 Calculate new preconditioner (Jacobi, BJ or MIC) q solving r=Mq
2.5 Dot-product (beta=r*q)
Dot-product (normar=r*r)
MPI ALL_REDUCE and/or Border Sum
alpha=beta/ganma
2.6 Calculate new vector d (d=q-alphad)
2.6 A Calculate Border Columns
Data Transmissions A (Non-Blocking)
2.6 B Calculate Interior Columns
2.7 tol=√normar
If tol < mintol GO TO 2.1
3 A Calculate Border Columns
4 A Calculate Border Columns
Data Transmissions B (Non-Blocking)
3 B Calculate Interior Columns
4 B Calculate Interior Columns

Figure 5.5. Improvements made in the OP-Si3D parallel model, Iterative Loop stage.

This synchronization among three is the same for all subdomains, except of course at the beginning and at the end of the domain, as in the case of subdomains 1 and 8 in Figure 5.5, which should only be synchronized with a neighboring subdomain to the east or west respectively. This type of synchronization between neighbors is at least necessary after S4 (Figure 5.2, border barrier), before and after S2 if a parallel Conjugate Gradient is being used (Figure 5.2) and within the parallel Conjugate Gradient itself, before the operation of Matrix-Vector multiplication. Additionally, in more points of S1, S3 and S4 if redundant calculation is not used. In this approach the new border barriers are used, replacing the generic barriers (ompbarrier) given by OpenMP. To perform this, it will have available an array of border barriers, separating each barrier in sizes of 4KB and where each boundary between subdomains uses a different border barrier. Once a thread reaches these synchronization points, each subdomain will only use the border barriers corresponding to its east and west borders, so that it will block only until its two corresponding neighbors reach the same point.

5.2.5 Points of communication

Communications obtain information that is not calculated by a subdomain from other subdomains to resolve dependencies. In the case of threads using shared memory, the synchronization points implicitly allow the performance of such communications. In the case of processes, the communications must be done explicitly.

Si3D, like other hydrodynamic models, requires communication to resolve the dependencies of the overlapping areas, thanks to the domain decomposition chosen and the distribution of data in memory, these overlapping areas are reduced to the borders of each subdomain, which are contiguously found in memory (Figure 5.1). This implies that each process must communicate with a maximum of two other subdomains. In OP-Si3D these border communications are necessary after S4 (Figure 5.2, border transmission) and during a parallel Conjugate Gradient for Matrix-Vector multiplication. Additionally in more points of S1, S3 and S4 if redundant computation is not used. Furthermore, additional communications are required for the solution of the Dot-product operations performed during the Conjugate Gradient (Figure 5.5, stages 2.2 and 2.5). In this case we make use of the function given by MPICH MPI_AllReduction, which is implemented efficiently even for NUMA architectures, performing the partial sums in a binary tree structure between the nearest neighbors (Mamadou et al. 2006). On the other hand, in the case of a sequential Conjugate Gradient being used, additional communications must be added before S2 to form the complete pentadiagonal matrix into a single process, and after S2, to distribute to each process its part of the solution of water surface elevation (more details can be found in Chapter 4 about the sequential implementation of CG in a parallel implementation of Si3D, Figure 4.2)

To do these communications as efficiently as possible, all the border communications are overlapping with calculation, conducting a reordering of the calculation of each subdomain. This reordering, found in each subdomain, first calculates those columns that form the overlapping area of other neighboring subdomains (Figure 5.1, subdomain 2, blue area). When these columns are calculated, they are sent in a non-blocking form and, at the same time, its own areas of overlapping

(Figure 5.1, subdomain 2, yellow areas) are received in a non-blocked form too. Subsequently each subdomain continues its execution by calculating the interior columns (Figure 5.1, Interior Columns), while it is completed in parallel the process of sending and receiving to/from the borders. This procedure is applied during S3 and S4, calculating and sending first the overlapping areas of the these stages (Figure 5.5, Data Transmissions B), and continuing the calculation of the rest of the subdomain and getting blocked with MPI_Wait before the next time-step, in case communications have not been completed yet. This reordering is also used to calculate the vector *d* in each iteration of PCG (Figure 5.5, Data Transmissions A during step 2.6), in order to solve the dependencies in the operation of Matrix-Vector multiplication (Figure 5.5, step 2.1). In this particular case, an additional inverse reordering is applied, where the interior columns of step 2.1 are first calculated, so adding the maximum calculation possible before the overlapping area is necessary to complete the calculations. In case that such communications are not received before they are needed to solve the dependences from the previous stage 2.6, the corresponding process is blocked with an MPI_Wait before finally calculating the borders of stage 2.1.

5.2.6 Parallel implementation of the Conjugate Gradient and preconditioner

5.2.6.1 Conjugate Gradient

During S2 of Si3D (Figure 5.2), as in other semi-implicit models, a symmetric positive system of equations of the type $Ax = b$ (where A is a matrix, x an array where the new solution is stored and b the independent vector) is defined with a pentadiagonal structure (see for example Nesterov 2010). This kind of system of equations is typically solved using an iterative method, which converge to a solution with a certain tolerance determined by the user. These methods have the advantage of being computationally much faster than direct methods like Gaussian substitution, which directly resolve the system. Among known iterative methods, those based on Krylov techniques are the most widely used, and among them Conjugate Gradient (Conjugate Gradient, CG) is usually given in literature as one of the best options (Barret et al. 2004). This is due to the number and type of operations that are presented in the CG with respect to other more expensive alternatives and that only a reduced group of vectors should be kept in memory during its execution.

The formation of the system of equations generated in S2 is closely related to the model geometry and dependencies between water columns. Firstly, the system consists of one equation (one row of the matrix *A*) per water column of the model has, as well as one element of the vector *x* and one element of the vector *b*. This makes the matrix *A* have as many rows and the vectors *x* and *b* have as many elements as the model has columns of water. Secondly, each equation has only 5 elements, determined by the position of each water column and its 4 neighbors to the north, south, east and west, thereby generating a pentadiagonal matrix as resulting in Figure 5.6(a) for the geometry presented.

At the beginning of S2, each process or thread has a part of the matrix or system of equations equivalent to the number of water columns that belong to its subdomain, also having the corresponding part to this subdomain of the vector *x* and the vector *b* . Given the domain decomposition and the storage of variables used, both the rows of *A* and the part of the vector *x* and *b* calculated by each process or thread are consecutive. The parallelization of CG can be described as the parallelization of the mathematical operations that comprise it. Figure 5.5 (S2) shows the operations that compose the CG, some of these operations (2.3-2.4-2.6-2.7) can be solved in parallel without adding communication or synchronization. Meanwhile, the operations of Dot-product (2.2-2.5) and Matrix-Vector (2.1) multiplication require synchronization points and/or communications as explained in Sections 5.2.4 and 5.2.5.

In the case of Dot-product operation, each thread or process has one part of the two vectors to multiply and can calculate the local product in parallel, the result is a scalar to be added, along with the other scalar values calculated by other processes or threads, to obtain the final result.

In the case of Matrix-Vector multiplication ($z = Ad$) its parallelization means that each process multiplies its matrix *A* rows by the vector *d*. To make this possible, each process or thread should have a complete copy of the vector d, adding any necessary communications and/or synchronization. However, this operation is simplified for these types of hydrodynamic models. Since each row contains only 5 elements, you just need the corresponding 5 positions of vector *d* to multiply it with each row of the matrix. In addition, these 5 elements correspond to each column of water and its neighbors to the north, south, east and west. For example, following the case of Figure 5.6(a), the operation of multiplying water column 3 of the matrix *A* (which contains 4 non-zero elements) by the vector *d*, actually needs only four elements of that vector, those in positions 2, 3, 4 and 7, equivalent to the number of the column (3) and its neighbors to the north (2), south (4) and east (7). In the parallel implementation presented for Si3D,

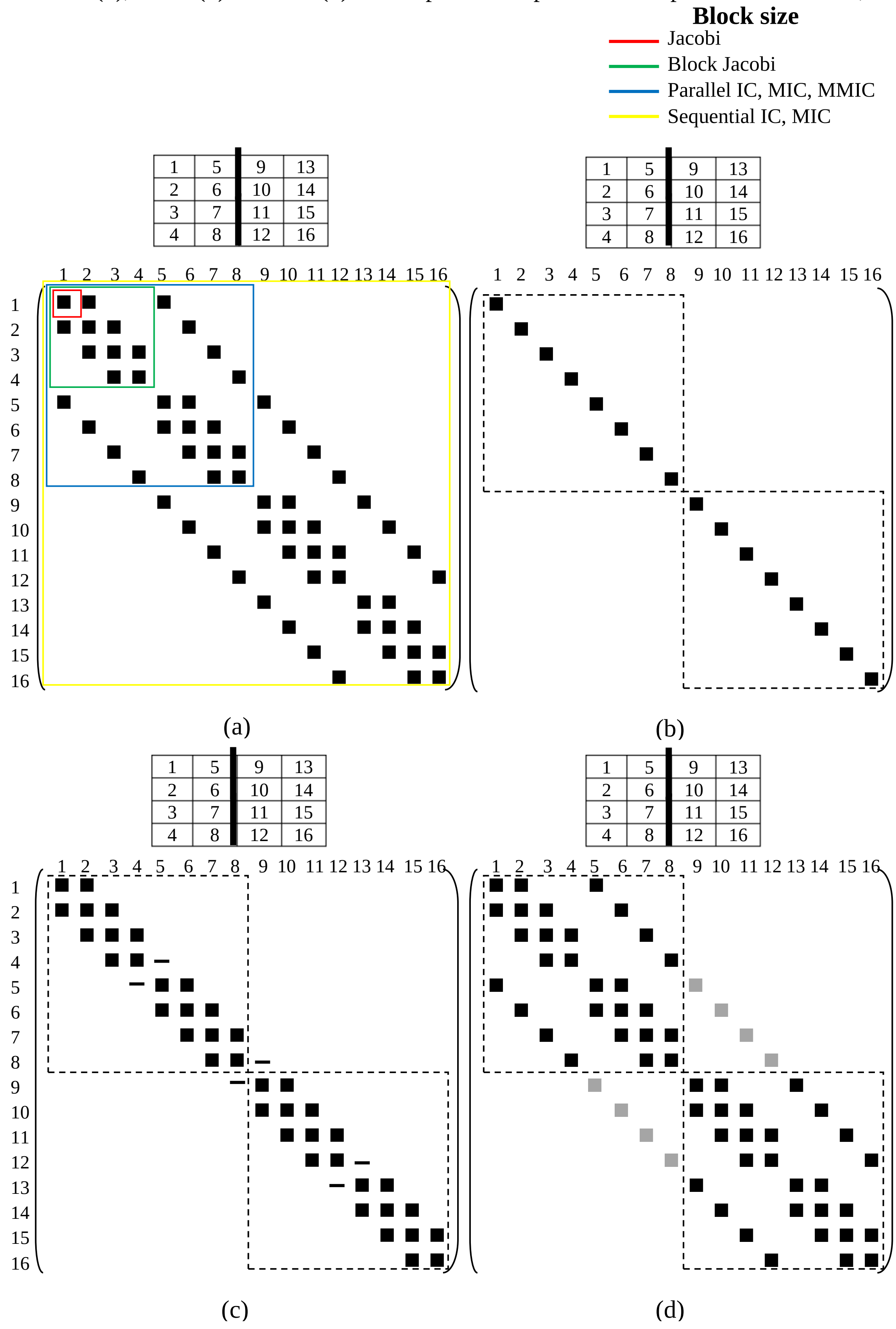


Figure 5.6. (a) The pentadiagonal matrix *A* resulting in forming the system of equations to solve water surface elevation in S2 according to the geometry of the domain given. The block size for each preconditioner is also indicated. (b) Preconditioned matrix used for the Jacobi preconditioner using two subdomains. (c) Preconditioned matrix used for BJ using two subdomains. (d) Preconditioned matrix for IC, MIC and MMIC in parallel using two subdomains.It also shows those elements (gray color) between subdomains which take a value of 0 and are added to the main diagonal during the construction of the preconditioner in MMIC.

the domain decomposition used ensures that only those water columns of the matrix located in the eastern and western borders of the subdomain will require data from other subdomains to carry out its part of the Matrix-Vector multiplication, solved by border communications or synchronization points. This ensures that the parallelization of CG does not require the addition of any extra computational cost, such as reordering or factorization, and that the same type of communications used in the rest of the model (border only) can also be used for S2.

5.2.6.2 Preconditioner

Because CG is an iterative method that converges to the solution according to a given tolerance, the number of iterations required to converge to the solution significantly affects the total execution time of this stage. Usually an M^{-1} preconditioner is used which accelerates the convergence of the PCG, obtaining the value of $q = M^{-1} r$ (Figure 5.5, step 2.4), where r is the residual vector and q is the new direction of convergence. Ideally, the preconditioner itself would be the inverse of *A*, thus solving the system of equations would be immediate. However, obtaining the inverse of *A*, or of any matrix in general, is usually a very costly process, so what is usually chosen as preconditioner *M* is a matrix that is as similar as possible to *A* , thus solving the system of equations $qM = r$ at the lowest possible cost without actually computing the inverse of *M*.

Four different preconditioners have been implemented in Si3D to assess their performance in an optimized parallel implementation and their efficiency to accelerate convergence. According to the literature, the four implementations can be considered as Block Jacobi, each one using a different block size and number of blocks to resolve the PCG sequentially or in parallel. Additionally, in each case the system of equations r = qM (to calculate the preconditioner) is solved by a different method, giving the name to each one of four preconditioners implemented. The Table 5.1 shows the equivalence between each of the preconditioner and the Block Jacobi implementation used.

Preconditioner	**Block Jacobi Size**	**Number of blocks per subdomain**
Jacobi	1 water column of the grid	Many blocks as water columns have the subdomain
BJ	1 horizontal row of the grid	Many blocks as horizontal rows have the subdomain
Parallel IC, MIC, MMIC	The number of horizontal rows into the subdomain	1 block
Sequential IC, MIC	The number of horizontal rows into the complete domain	1 block

Table 5.1. Equivalence between each preconditioner implemented with its Block Jacobi implementation performed.

- Jacobi: it uses the inverse of the main diagonal of A as the preconditioner M^{-1} and directly resolves $q = M^{-1} r$. This is equivalent to the use of a block Jacobi size equal to one water column of the grid (Figure 5.6(a), red box). Each subdomain applies Jacobi in parallel using as many blocks as water columns have the subdomain (Figure 5.6 (b)).
- Block Jacobi (BJ): it uses the three main diagonals of the matrix A as the preconditioner M where the system $qM = r$ is solved by the Thomas algorithm typically used for solving tridiagonal systems. This is equivalent to the use of a block Jacobi size equal to one horizontal row of the grid (Figure 5.6(a), green box). Each subdomain applied BJ in parallel using as many blocks as horizontal rows have the subdomain (Figure 5.6(c)). Given the structure of the matrix obtained (Figure 5.6(c)), it is possible to solve this system of equations in parallel without following a strict sequential order or adding synchronization or communication points. The Thomas algorithm involves two phases with a forwards and backward substitution where, if all elements of the 3 diagonals were not zero, the process would be strictly sequential. However, the tridiagonal system has some zero elements generated by the absence of neighbors to the north and south at the end and start of each horizontal row of the domain respectively. For example, these zero elements can be seen in Figure 5.6(c), marked with dashes, to generate the equation 8 and 9 of the matrix M, this is

because in the first subdomain, the water column 8 has no neighbor to the south, being the last in that horizontal row, at the same time, the water column 9 has no neighbor to the north, being the first of its horizontal row. This allows the solution of the full tridiagonal system as a set of independent tridiagonal subsystems, as long as each subset is composed of complete horizontal rows of the grid. For example, in Figure 5.6(c) both subdomains (highlighted in a box) can be independently solved.

- Incomplete Cholensky and Modified Incomplete Cholensky (IC and MIC): it uses the factorization of type LDL^{T} of matrix A as preconditioner M, L is a lower unit triangular matrix, D is a diagonal matrix an L^{T} is the conjugate transpose of L. This allows us to solve the system $Mq = r$ as $LDL^{T}q = r$ in two steps. Solving first $LDx = r$ to obtain x and in a second step by solving $L^{T}q = x$ to finally obtain q. Both steps are solved by a substitution algorithm, using a forwards/backwards substitution (Dongarra 1998). Both IC and MIC should be made so that M retains the pentadiagonal structure of the original matrix A. In order to maintain this same structure, those elements which are different to 0 out of the 5 main diagonals (known as filling values) take a value of 0 using IC and, additionally using MIC besides taking 0 value in its original position they are added to the main diagonal of M . To determine whether IC or MIC is used, a parameter w is used which indicates if the filling values are or not added to the main diagonal, with $w = 0$ for IC, $w = 1$ for MIC and a value between 0 and 1 a compromise between IC and MIC. The parameter w is usually adjusted for adding convergence by the preconditioner. For example, Dongarra (1998) indicates that w should be set to 0.95 to obtain the best convergence ratio.

For the sequential implementation, MIC and IC are directly applied to the matrix A (Figure 5.6 (a)). This is equivalent to the use of a block Jacobi size equal to the complete domain (Figure 5.6(a), yellow color) and only one block.

The main goal to obtain a parallel implementation of this preconditioner is that the implementation must not require any type of additional extra operation (such as reordering), communication/synchronization points or anything that cannot be fully parallelized. However, this is not a trivial task taking into account that its implementation, both the forward and the backward substitution, must be performed in a strictly sequential way. Given this requirement in the parallelization of this preconditioner, a similar solution to the idea used in Block

Jacobi has been chosen here, so that the calculation of the preconditioner in each subdomain is done independently. To carry out this process, the elements corresponding to the overlapping area of each subdomain take a value of *0*. This is equivalent to the use of a block Jacobi size equal to the number of horizontal rows of the grid into each subdomain (Figure 5.6(a), blue box). Each subdomain applies IC or MIC in parallel using only one block (Figure 5.6 (d)). The Figure 5.6(d) shows an example where the first subdomain consists of the first 8 water columns of the model. To compute in parallel the corresponding part of the preconditioner to this subdomain from the matrix *A*, those elements of the upper diagonal from positions 9 to 12, as well as the lower diagonal elements from positions 5 to 8, takes 0 value (elements in gray). Making these elements take a value of 0, it is possible to solve the preconditioner in an independent parallel way for each subdomain, adapting the parallel implementation of the preconditioner with the domain decomposition used in other stages. However, moving these elements to *0* value on the borders between subdomains, the new preconditioner *M* is created from a matrix that is less like *A* (since these values do have a non-zero value in the matrix *A*) and, therefore, the convergence acceleration of the preconditioner decreases. Furthermore, increasing the number of subdomains, the number of borders where these elements must take a value of *0* increases, slowing convergence with each new division of the domain. To reduce this loss of convergence, when the values between subdomains are 0, a modification to the preconditioner MIC is implemented. This implementation is called here MMIC. This modification is similar to the idea followed in the MIC implementation itself to remove the filling data and maintain the structure of the original matrix *A*. In this case the values converted to *0* (gray elements in Figure 5.6) are added to the main diagonal of the preconditioner, thus considerably reducing the loss of convergence. This modification is controlled by a parameter *w2*, where for *w2 = 0* these values are not added to the main diagonal, for *w2 = 1* these values are added to the main diagonal and a value between *0* and *1* provides a compromise between both of them. This parameter can be adjusted to find what value of *w2* provides greater convergence acceleration.

5.3 Performance evaluation

Results of execution time and scalability using OpenMP, MPI or both simultaneously are presented in this section. The execution time results are presented as an average of 10 iterative loops (Figure 5.2), neglecting the first iteration which is different to the rest (the leapfrog step is not possible until the second iteration). The time before and after the iterative loop is considered negligible for these types of simulations, wherein the said iterative loop must repeat the time-steps needed to cover long periods of simulation. For example in the high resolution case of Lake Tahoe, the number of iterations to make a simulation of 30 days with a time-step of 10s is 259,200 (Table 5.2). Conversely, the time before and after the iterative loop is less than 0.1% with respect to the total time required to execute all the time-steps.

We also evaluated the potential load imbalance between subdomains produced by the selected domain decomposition (Section 5.2.2), counting the number of cells that each subdomain has to calculate. This study was done counting the number of cells in each subdomain with divisions from 2 to 256. In all cases the difference in cells between the subdomain with the highest number and the subdomain with the least cells was from 0.15% to 0.5% for all cases evaluated.

5.3.1 Platform and tools

OP-Si3D has been tested in 3 architectures with different features in order to evaluate its performance and scalability, both in small commodity clusters (including clusters of distributed memory and shared memory machines) and high performance architectures. This will not only evaluate the performance and scalability of the parallel implementation in architectures with good performance, but also compared to other clusters of low performance to determine whether the implementation can achieve good results in a type of cluster which most scientists can access.

ACII has been used as a low-end cluster, a machine of 3 nodes. Each node has 6 GB of memory and an i7 CPU920 processor with 4 cores of 2.667 GHz. In the tests performed both Infiniband and Gigabit Ethernet have been used as network connection between nodes. Each node has only one socket, with a processor with four cores in each socket.

ALHAMBRA has been used as high performance architecture with distributed memory, a cluster with 16 Fujitsu CX250 (S1) nodes. Each node has 32 GB of memory and two Intel Xeon E5-2680 processors with 8 cores each of 2.7 GHz. The nodes are connected by Infiniband. Each node has two sockets, with a processor with 8 cores in each socket.

CIEMAT has been used as a shared memory midrange machine, a machine with two CC-NUMA shared memory nodes. Each node has 512 GB of memory and 4 Intel Xeon X7750 processors with 8 cores each of 2 GHz. Each node has 4 sockets, with a processor with 8 cores in each socket.

Given that Si3D, from the original model, is developed in Fortran, the execution is generated with the Intel Fortran (version 11) compiler, also using the OpenMP version included in the compiler. For message passing paradigm MPICH2 3.0 has been used. The code was compiled using standard compilation optimizations given with O2. In addition, the Intel V-Tune tool was used to detect those subroutines of code that consume more CPU time and to apply various basic optimizations as explained in Section 5.2.1.

5.3.2 Test model

It has been used, as an application to evaluate the parallel implementation, a realistic simulation of Lake Tahoe (USA). Currently several groups of researchers are using this model to characterize transport routes in the coastal area (around the littoral zone) of pollutants (Hoyer et al. 2015) and larvae of a species known as invasive Asian clam (Corbicula bivalve fluminea) (Hoyer et al. 2014).

Lake Tahoe has horizontal dimensions of 20 km x 30 km and up to 500 m of depth in the vertical. Two tests with Cartesian grids have been used for simulation with square columns of 50m x 50m and 20m x 20m in the horizontal (Table 5.2). The vertical resolution is the same for both cases with 95 layers of depth. The case of lower resolution obtained good results in the pelagic zone of the lake and can be run on platforms with less computational resources. However, it is showed in chapter 2

(Section 2.4.4) that some of the circulation patterns in the coastal zone can only be properly reproduced if a similar scale to those found in the littoral bays is used, requiring a resolution of at least 20m x 20m. However, this simulation using a high-resolution grid is not trivial and requires an unaffordable computational cost. The computational time of this simulation using the Si3D sequential model, even including several basic optimizations, keeps a real-time/simulation time relationship of approximately 768:30, meaning it would take 768 days of computing to simulate 30 days. Likewise, the cost of memory is too high, requiring more than 16 GB for its execution. This huge computational cost prevents its simulation sequentially or even in parallel in small commodity clusters with little memory as in the case of ACII.

The lake bathymetry data was obtained from the US Geological Survey. The vertical resolution comprises up to 95 layers with a layer depth variable from 0.5m at the surface to about 10m near the bottom. The time-step is set to 25s in the case of lower resolution and 10s in the case of higher resolution, thus fulfilling the condition of CFL and ensuring the stability of the simulation. The model was forced (input data) using Surface heat and Momentum fluxes estimated from local atmospheric variables (long and short wave radiation, air temperature, relative humidity and wind speed and direction) obtained from meteorological data. This data was obtained from meteorological stations maintained by the Tahoe Environmental Research Center (TERC), having a total of 10 stations around the coast and around the lake. All the stations provide continuous information about wind speed and direction and air temperature.

Models of Lake Tahoe	**Horizontal cell side**	**Columns of water**	**Total cells**	**Simulation period**	**Time-step**	**Total Time-steps**
Tahoe50	50x50m	197,781	14,654,639	30 days	25s	103,680
Tahoe20	20x20m	1,244,896	94,691,170	30 days	10s	259,200

Table 5.2. Computational data of Lake Tahoe simulations: Horizontal resolution (square columns), total number of columns and total number of cells, period of simulation, time-step used and the total number of time-steps needed to finish the simulation.

5.3.3 Basic optimizations

The original model of Si3D kept information from dry columns, requiring a higher memory cost and execution time because of having to evaluate the calculation of all columns in the grid, regardless of whether they are columns with water or not, a first optimization was performed to store only columns with water as explained in Section 5.2.1. The Figure 5.7 shows the results of different sequential runs of Si3D from the original model (using O0 and the standard compiler optimizations (O2)) and the improvements achieved in execution time by applying the optimizations shown in the figure for the case of Tahoe50 in ACII, using the sequential MIC as the preconditioner in the CG. Each successive improvement from left to right is evaluated including each one of the previous optimizations, using the optimization option O2.

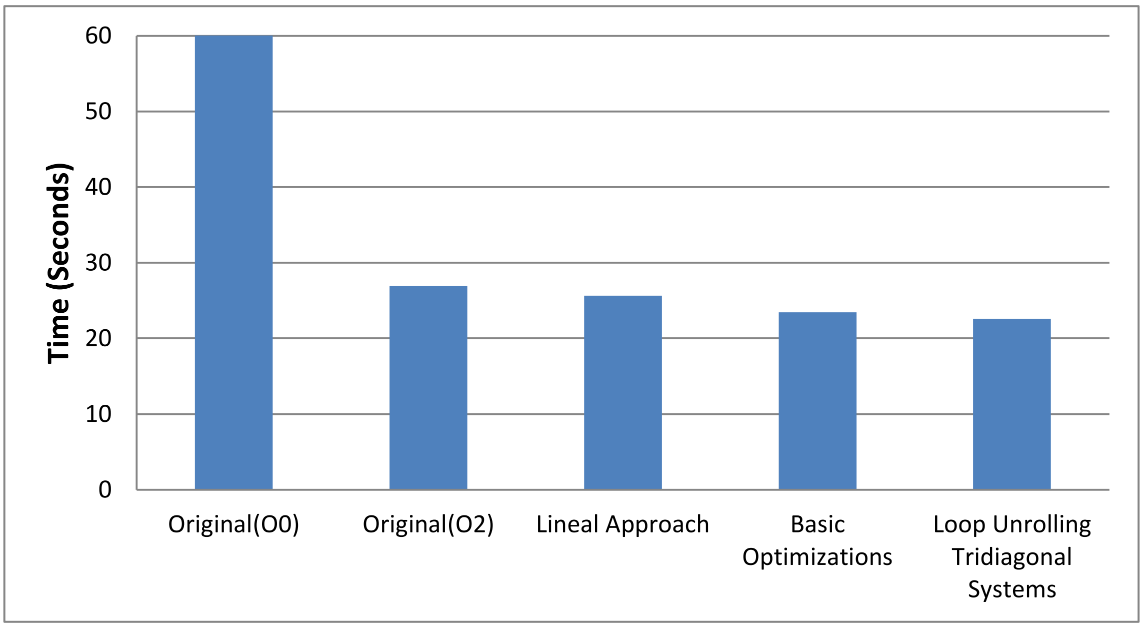


Figure 5.7. Iterative Loop runtime for a sequential version of SI3D adding several basic optimizations.

As it is shown, storing only columns with water reduces the runtime by 4.7%. In addition, other basic optimizations outlined in Section 5.2.1 reduce the execution time by up to 7.76%. Finally, the unrolling loop applied in order to help the compiler to optimize the Thomas algorithm used in solving tridiagonal systems reduces runtime by another 4%. Once all the basic optimizations are applied, the total reduction is 15.5% compared to the original version optimized by the compiler (O2).

The implementation of Si3D with all the basic optimizations (known as Basic) is the first step taken in the modifications made to Si3D, this implementation has been used as a base in the development of the rest of implementations presented in other chapters.

5.3.4 SI3D profiling

SI3D is an extensive code with lots of subroutines and a huge amount of variables so that a study of those stages of the code that consume more time is vital. This profiling is obtained from a sequential execution of Si3D, including all basic optimizations and using O2 in an ALHAMBRA node for Tahoe20, using MIC as the preconditioner in the CG.

The Figure 5.8 shows the percentage of execution time compared to the total time of each stage of Si3D (Figure 5.2), including a complete iteration of Si3D with one leapfrog and one trapezoidal step. As it is shown, the two stages with most cost are S1 and S4, with a cost of 57% and 29% with respect to the total. This is because it is in these stages where the formation and resolution of as many tridiagonal systems as the model has water columns must be done, the formation and resolution of these tridiagonal systems representing the 53% of the total time and must be performed up to 3 times in the iterative loop. The first systems of tridiagonal equations (Momentum Trid.) is calculated for S1 and represents 31% of the total time of this stage, the remaining time of S1 is consumed in the calculation of the explicit terms of the equations of momentum (exmom, with 67%) and the formation of the pentadiagonal system that will be resolved in S2 (matcon, with 2%). Additionlly, the resolution of the tridiagonal systems in calculating (1) temperature in the transport equation and (2) the horizontal coefficients of viscosity in the turbulence equation represent 76% (Transport and Turbulence Trid.) of the S4 runtime, the remaining 24% being the calculation of other variables. Furthermore, the resolution of the system of pentadiagonal equations in S2 represents 3.54% (PCG) of the total. Solving the equation of continuity in S3 to obtain the new volumetric transports represents 2.65% (Continuity) of the total. Finally, the updating of variables (Data Update), both after the leapfrog and before the next time-step represents 7.08% of the total time.

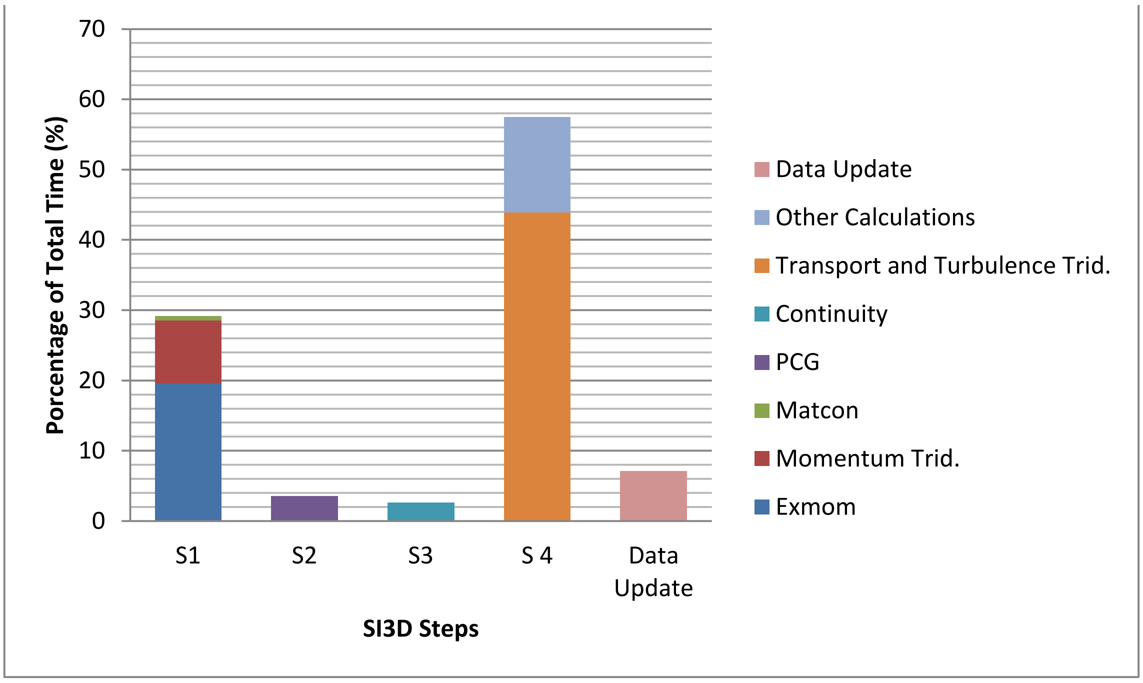


Figure 5.8. Percentage of runtime of the SI3D stages in the Iterative Loop.

5.3.5 Hybrid Model Optimizations

In this section the results of the implementations performed to take full advantage of the parallel implementation are evaluated, with explicit optimizations for both threads with OpenMP and for processes with MPI. The results in this section are intended to determine whether the procedures presented enhance the performance of the parallel implementation or which alternative from those presented is the most efficient. These results are obtained for Tahoe50, in ACII using only processes, and in CIEMAT using only threads, starting with the already optimized model presented in Section 5.3.3 (basic optimizations). This model includes the original implementation used by Smith (2006) of the CG using MIC as the preconditioner, which is included in the NSPCG math library (Non-Symmetric Preconditioned Conjugate Gradient, Kincaid et al. 1989).

5.3.5.1 OpenMP optimizations

The Figure 5.9 shows the results of SpeedUp comparing the basic model and other implementations of the OpenMP parallel model. Basic presents results (using threads) of the model with basic optimizations obtained in Section 5.3.3, which shows a rather poor performance. The Affinity model is the same implementation but adding the first-touch and an explicit affinity (first fully occupying each CIEMAT socket). The results of this model show a significant improvement of up to 45% in the case of 64 threads. This demonstrates the importance of making use of data locality in NUMA type architectures, with an initialization of data in parallel by each subdomain and the importance of binding each thread with the core where the initialization is performed, which also ensures that neighboring subdomains are founded in neighboring cores to achieve a faster access to their overlapping areas.

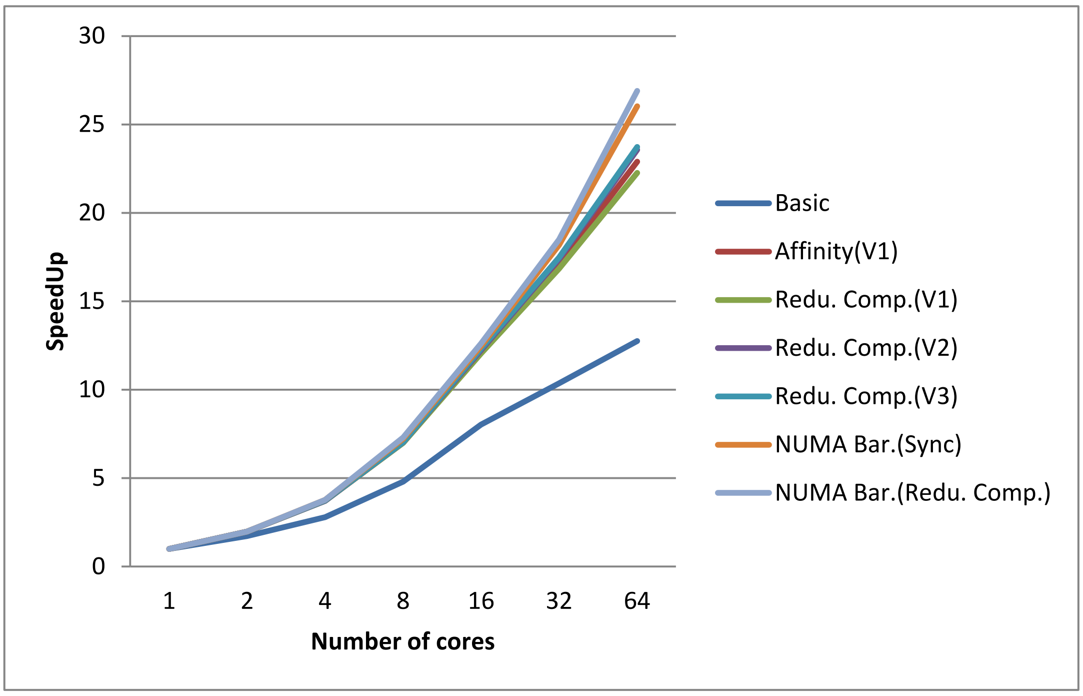


Figure 5.9. Comparison of SpeedUp results between the basic model and other implementations of the OpenMP parallel model for Tahoe50 in CIEMAT.

The new Affinity model, like other parallel models, includes synchronization (Ompbarrier barriers) before any calculation loop that requires information dependent on neighboring subdomains, which in the chosen decomposition domain is reduced to the overlapping areas between neighboring (Figure 5.1). Besides the synchronization points shown in Figure 5.2, 9 barriers must be added in S1, 4 in S3 and 12 in S4 to resolve all dependencies. Given the large number of synchronization points, a study of the type of calculation between these synchronization points was performed, noting that these dependencies in S1, S3 and S4 occur frequently between nearby loops with little calculation between them. Therefore a new implementation which replaces the synchronization points of S1, S3 and S4 for redundant computation (Redu. Comp. in Figure 5.9) was evaluated.

Moreover, the storage type of the variables has also been evaluated with the 3 types of storage explained in Section 5.2.3. In the model with redundant calculation the variables are initially stored using V1 (redu. Comp. (V1)), showing poorer results than using synchronization and the same storage (Affinity). However, storage types V2 and V3 (Redu. Comp. (V2) and Redu. Comp. (V3) respectively) obtain better results than V1, both using redundant computation and synchronization, being V3 the best option, 0.72% better with respect to V2 and 6.16% with respect to V1 with redundant computation and 3.52% with respect to V1 with synchronization, in each case comparing the results obtained with 64 threads. This is because both V2 and V3 do not need, to solve dependencies, access to data for calculation of the overlapping area that neighboring threads are continually accessing at the same stage in both reading and writing mode. The new storage ensures that the information that each thread brings to its cache memory will not be invalidated by neighboring threads. Meanwhile V3 slightly reduces the runtime of V2 and reduces the amount of memory required, not needing to include garbage or filling between subdomains to ensure that each memory block is only accessed by the local thread.

Finally, the results of a model fully adapted to the NUMA architecture were presented, using the border barriers between subdomains and the barriers per level outlined in Section 5.2.4. Thus, the barriers used and given by OpenMP (ompbarrier) are replaced by the new approach in the best version available at this point (Redu. Comp. (V3)) obtaining a new implementation (NUMA Bar. (Redu. Comp.)). As it can be seen, the new implementation improved scalability of the parallel model with respect to the same version using Ompbarrier, reducing runtime by up to 11.67% in the case of 64 threads. Having demonstrated the efficiency of the newly developed barriers, the redundant computation was replaced again by synchronization points using the improved border barriers (where redundant computation is used in S1, S3 and S4) (NUMA Bar. (Sync)). The results show that it is still better by 3.2% to use redundant computation using 64 threads, despite the better performance of the new barriers.

5.3.5.2 MPI optimizations

The Figure 5.10 shows the results of SpeedUp between the base model with basic optimizations and other implementations of the MPI parallel model for Tahoe50 in ACII, using Gigabit Ethernet as the connection network. Basic presents the results (using processes) of the basic model including the basic optimizations presented in Section 5.3.3, which has poor scalability due to the large amount of data to be communicated and several blocking waits while those communications are performed. The Affinity model slightly improves the results of the Basic model, by up to 3.28% with 12 processes, since it ensures that communications between different nodes will be minimal. However, this model still shows very poor scalability, this is because both models (Basic and Affinity) include communication before any calculation loop that requires information dependent on neighboring subdomains, which in the chosen domain decomposition is reduced to the overlapping areas between neighbors (Figure 5.1). In addition to the communication points shown in Figure 5.2, 9 points of communication must be added in S1, 4 in S3 and 12 in S4 to resolve all dependencies. This large amount of both, the quantity of data to communicate and the number of communications between loops with little calculation, leads to lack of scalability of the model. To solve this problem, a new approach is done (Redu. Comp. (Block. Comm.)) where these communications of S1, S3 and S4 are replaced by redundant computation following the same procedure outlined for threads in Section 5.3.5.1. As shown in Figure 5.10, a significant improvement in efficiency is obtained for the parallel model of up to 14.8% in the case of 12 processes. This is because the number of communications has been reduced considerably.

In the process of reducing communication costs to the minimum, the results of (Redu. Comp. (Reorder)) shows the same implementation as (Redu. Comp. (Block. Comm.)) but substituting the communications at the end of S4 (Figure 5.2) by non-blocking communication and performing a reordering of S3 and S4, as explained in Section 5.2.5. The results show a clear improvement when these communications are overlapped with calculation, with an improvement of 6.55% in the case of 12 processes.

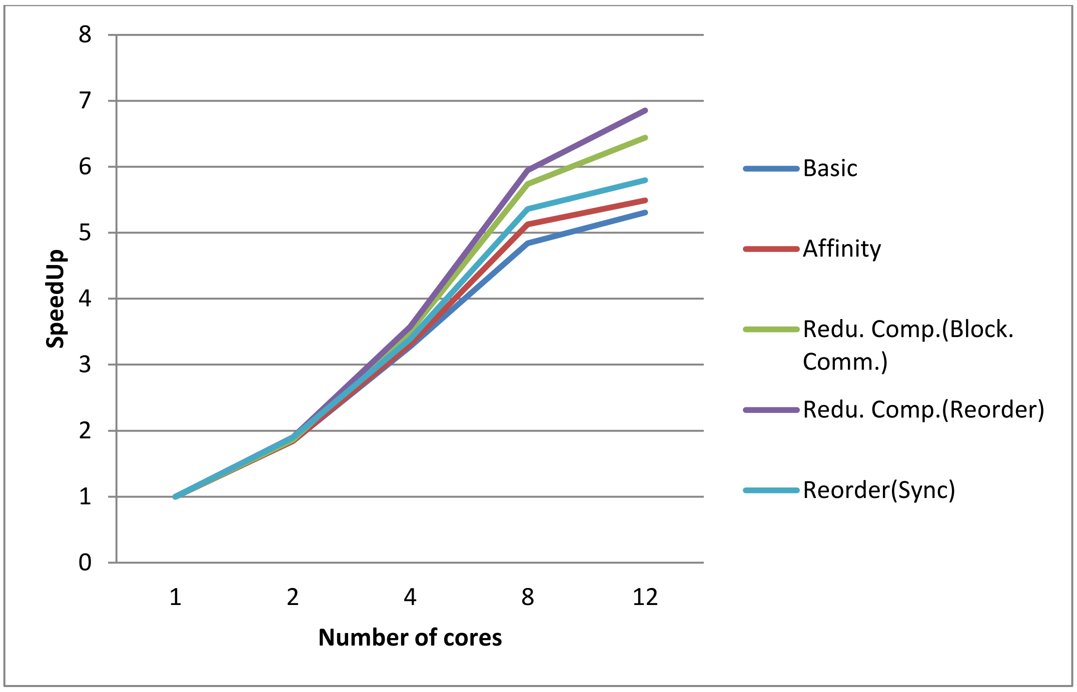


Figure 5.10. Comparison of SpeedUp results between the basic model and other implementations of the MPI parallel model for Tahoe50 in ACII.

Once implemented the new reorganization and non-blocking communications, the redundant computation is replaced by non-blocking communication points and reordering of those calculation loops in S1, S3 and S4 with dependencies. The results (Reorder (Sync)) show that it is still better to use redundant calculation up to 15.5% using 64 threads. This is because the number of communications between loops is so frequent that the overlapping of calculation of small loops with the time needed to carry out communications is not possible, reaching an overhead added by the communications between neighboring subdomains.

Two different types of affinity were also evaluated using the best implementation (Redu. Comp. (Reorder)) once the cost of communications has been minimized. ACII and CIEMAT were used for these tests. Tests were conducted with 2, 4 and 8 processes. With the first affinity map each socket is filled first (in ACII since each node has only one socket, the affinity is associated with occupying each node). Thus for the case of four processes for example, the four processes are co-located in the same node of ACII or the first CIEMAT socket. With the second affinity map, an attempted distribution to occupy the largest possible number of nodes (sockets) was used. For example, for the case of 4 processes, the four processes are co-located in the first four different sockets of the first CIEMAT node. In the case of ACII, two neighboring processes are co-located in the first node and only one process is placed in the other two nodes respectively. As another example, for the case of eight processes, only one process is allocated per CIEMAT socket while in ACII, three neighboring processes were co-located in the first node and two neighboring processes were placed for each remaining node. Table 5.3 shows that the results obtained in ACII and CIEMAT are not the same. In CIEMAT, with an amount of memory much higher than ACII, it is preferable to fill each socket completely before occupying another socket (minimum distribution results against maximum distribution results), favoring the data locality to locate information from neighbors (to access the overlapping areas) in the memory of the same socket. However, in ACII, as seen in all cases, it is preferable to occupy the largest number of nodes (maximum distribution) rather than totally occupy each node (minimum distribution), thus providing more memory to run the simulation. This indicates that despite the overhead added by the communications between nodes and the reduction of the data locality to distribute the processes in different nodes, it is slightly more efficient to have more memory for its execution.

Affinity Test(Infiniband)	**ACII Time (seconds)**		**CIEMAT Time (seconds)**	
Threads	maximum distribution	minimum distribution	maximum distribution	minimum distribution
1	**22.48**	22.48	32.54	**32.54**
2	**11.42**	11.53	16.43	**16.41**
4	**6.12**	6.24	8.72	**8.63**
8	**3.28**	3.46	4.58	**4.46**

Table 5.3. The results of execution time (seconds) per time-step of the parallel model using two types of affinity are presented. One where each socket is completely occupied first (minimum distribution) and another where the threads are distributed occupying the highest number of sockets first (maximum distribution).

5.3.6 Conjugate Gradient Optimization and preconditioner

Table 5.4 presents the number of collective communications, non-collective communications and the results of different alternatives of CG and different preconditioners, using Tahoe50 in ALHAMBRA, which is the available architecture where a greater number of subdomains can be studied (up to 256). The parallel implementation developed for this model (PCG) has been compared to the sequential implementation included in the NSPCG math library (rows 2, 3 and 4) and originally used in the Smith model (2006) (Chapter 1). Comparison with the NSPCG library can test whether the implementation of the Conjugate Gradient used obtains similar results when the model is executed sequentially, both the number of iterations to converge and the runtime.

PCG	many-to-one	Interchanges	No. iterations		Total Time	
			1 thread	256 threads	1 thread	256 threads
PCG (NO Preconditioner)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	882	882	6.84	6.84
NSCPG(MIC)	Not applicable	Not applicable	16	16	1.89	1.89
NSPCG(JAC)	Not applicable	Not applicable	760	760	5.96	5.96
NSPCG(BJAC)	Not applicable	Not applicable	537	537	4.97	4.97
PCG(JAC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	761	761	5.86	0.22
PCG (reorder_JAC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	761	761	5.86	0.1
PCG (BJAC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	538	538	4.83	0.16
PCG (reorder_BJAC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	538	538	4.83	0.07
PCG (IC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	57	405	5.74	0.18
PCG (MIC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	17	129	1.84	0.06
PCG (MMIC)	2 ALLREDUCE per iteration + 1 ALLREDUCE	2 ISEND + 2 IRECV per iteration + 2 ISEND/2IRECV	17	74	1.84	0.03

Table 5.4. Conjugate Gradient and preconditioners used both sequentially using the NSPCG library and in parallel with the developed code (PCG).

The new implementation has also been compared with 4 different alternatives for the preconditioner: Jacobi (Rows 5 and 6), Block Jacobi (BJ, Rows 7 and 8), Incomplete Cholensky (IC, Row 9) and Modified Incomplete Cholensky (MIC, Row 10) as well as a modification of MIC itself for the parallel model (MMIC, Row 11). In addition, for the Jacobi and BJ cases, they are compared to the use of blocking communication (rows 5 and 7) and non-blocking communication with reordering of calculation (Rows 6 and 8) described in Section 5.2.6.1, showing the best results in both cases with the second option, where most communications are efficiently overlapped with calculation in the CG.

In the first place, the results show that the number of iterations to converge and the total execution time between the NSPCG method and the implementation developed (PCG) are very similar. Besides, in the case of the evaluation of the PCG developed, comparing execution times and the number of iterations to converge using different preconditioner alternatives, the results show that the minor computational cost per iteration is obtained by Jacobi followed by BJ, IC and MIC. This is because the number of operations needed in the computation of each preconditioner. Jacobi is only to reverse the main diagonal of *A*. On the other hand, BJ must solve a system of tridiagonal equations of the size of the number of water columns of the domain. In IC and MIC, the cost is even higher, since the matrix *A* must be factorized in the form of LDL^T and proceeds to the resolution of the new system as explained in Section 5.2.6.2, adding more computation. However, the number of iterations to converge varies contrary to the order of the cost of each iteration, making the final execution time less using MIC, followed by IC, BJ and finally Jacobi, which despite having a very small computational cost does not offset with respect to the number of iterations needed to converge.

As in other stages of the parallel model, the PCG developed has only dependencies with other subdomains in the overlapping areas (Figure 5.1), solved by border communication/synchronization, without adding any additional reordering or factorization. Furthermore, it is not necessary to add any computational cost to the parallel implementation or calculation of the preconditioner. To make this possible, as explained in Section 5.2.6.2, some convergence is lost when the number of parallel subdomains is incremented. Table 5.4 shows the evolution of the different preconditioners based on the number of subdomains used. The group of columns 1, 2 and 3 show the number of iterations to converge using MIC, IC and an intermediate solution respectively (variable w). As it can be seen, it is better to use the intermediate solution (except in a sequential execution where MIC is better). To control in what proportion a solution between IC and MIC is used, the value of *w* parameter is changed, taking the value 0 for IC (column 3), 1 for MIC (column 6), and an intermediate value which is changed when the number of subdomains increase to obtain the best convergence results. Column 9 shows the optimal values of *w* for each number of subdomains when a parallel MIC with variable w is used. As it is seen in Table 5.3, MIC in parallel with the optimized *w* parameter gets the best results even although the number of iterations to converge increase, both with respect to a sequential execution of MIC (with a fixed number of iterations to converge) as with respect the second best option in parallel, BJ.

Finally, the results of the modification to MIC (MMIC) explained in Section 5.2.6.2 are shown. This modification reduces the loss of convergence when the number of subdomains increases. These results can be seen in the group of columns 4 and 5 of 0, noting that the number of iterations to converge with respect to the parallel MIC improves. Again this change is controlled by a parameter *w2*, whose optimum value takes a value between 0 and 1, depending on the number of subdomains used. The results using this modification of MIC can be compared in 0 using a value of *w2* = 1 (column 13) or using an intermediate value between 0 and 1 which is changed according to the number of subdomains used (column 14), the results show that it is better to use a variable value of *w2*. The best value of *w2* according to the number of sub-domains can be seen in column 16 of the same table. As shown in Table 5.4, MMIC improves the results obtained by the parallel MIC up to a total of 256 subdivisions, presenting itself as the best option and providing a parallel preconditioner that does not add any costs in comparison to the implementation of a sequential MIC. This is achieved by reducing the convergence, but significantly reducing the loss of convergence when (1) two parameters *w* and *w2* which vary according to the number of subdomains are used and (2) a modification to MIC presented as MMIC is used.

ALHAMBRA	MIC			IC			MIC w variable			MMIC			MMIC w variable		
Threads	**No. of Iterations**	**w**	**w2**	**No. of Iterations**	**w**	**w2**	**No. of Iterations**	**w variable**	**w2**	**No. of Iterations**	**w**	**w2**	**No. of Iterations**	**w variable**	**w2 variable**
1	17	1	0	57	0	0	17	1	0	17	1	1	17	1	1
2	18	1	0	57	0	0	18	0.99	0	34	1	1	17	1	0.8
4	50	1	0	59	0	0	34	0.96	0	36	1	1	22	1	0.95
8	61	1	0	59	0	0	42	0.95	0	39	1	1	23	1	0.93
16	69	1	0	61	0	0	55	0.9	0	45	1	1	29	0.99	0.89
32	90	1	0	63	0	0	68	0.85	0	60	1	1	36	0.99	0.88
64	107	1	0	331	0	0	89	0.8	0	85	1	1	47	0.99	0.87
128	760	1	0	367	0	0	135	0.75	0	218	1	1	65	0.99	0.87
256	981	1	0	405	0	0	151	0.7	0	353	1	1	74	0.98	0.86

Table 5.5. Number of iterations required by the parallel PCG to converge to a solution with a certain tolerance depending on the preconditioner, value w and value w2 used, and the number of subdomains used. For MIC and MMIC with variable w, the best possible values of w and w2

5.3.6.1 OpenMP optimizations

The Dot-product operation performed up to 3 times in the CG is calculated by the threads, obtaining in each one a local scalar product of its subdomain, and then adding it to a global variable. Here the operation given by OpenMP (OpenMPReduction) is compared to the implementation developed (NUMA Barrier Dot Product) which allows adapting the Dot-product operation to the NUMA architecture, as explained in Section 5.2.4. The results presented in Figure 5.11 show that the actual implementation reduces runtime by up to 12% for 64 threads in CIEMAT, allowing us to exploit the data locality when the partial sums are performed.

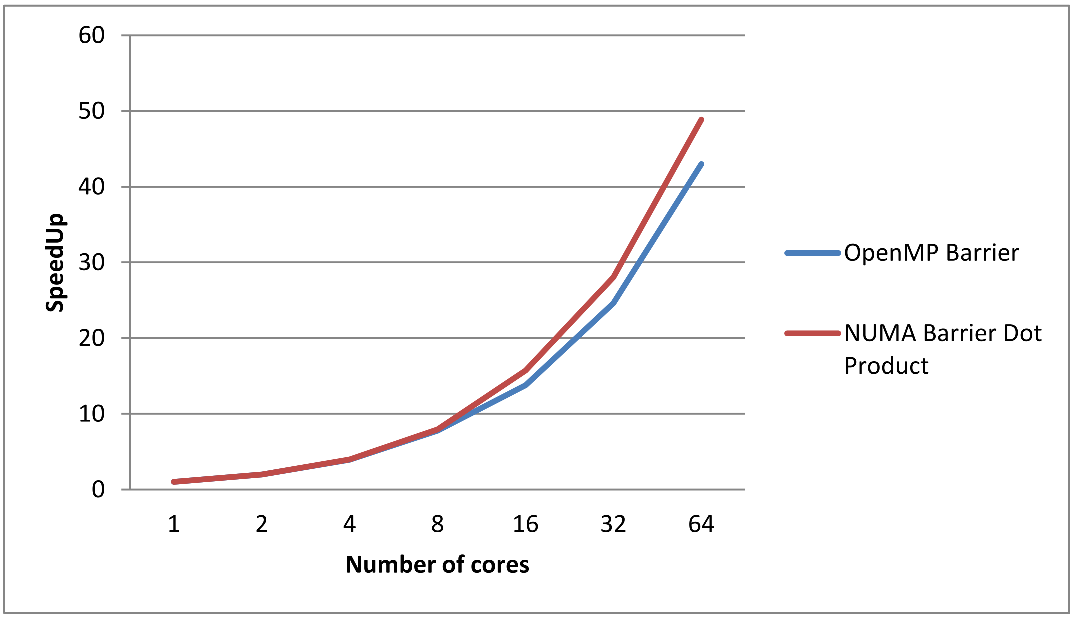


Figure 5.11. Comparison of SpeedUp results between the model with a parallel CG using NUMA barriers and other implementation using ompbarrier and ompreduction for Tahoe50 in CIEMAT.

5.3.6.2 MPI optimizations

It was noted that the same overlapping of communication with computation performed in S3 and S4 (Reorder), whose results are presented in Section 5.3.5.2, can be carried out with all communications made in the parallel implementation of the CG, as explained in Section 5.2.5. The results (Figure 5.12) show a substantial reduction in the execution time of up to 63% in the case of 12 processes in ACII with Gigabit Ethernet, with respect to another implementation where communications are blocking without reordering (Block. Comm), showing that all communication can be effectively overlapped with computation.

Once all communications are overlapped with calculation, both in CG and in other stages, results of the parallel model were compared in ACII using Gigabit Ethernet and Infiniband as interconnection networks, both the reordering version and the version with blocking communication. The results (Figure 5.12, Gigabit vs InfiniBand) show that communications produce a significant overhead in slower networks such as Gigabit Ethernet, whose results are much worse than those obtained by Infiniband. However, the difference using non-blocking communications and reordering between the two networks is only 2.51% in the case of 12 processes, showing that almost all communications can be overlapping and a good parallelization implementation can avoid using more expensive networks.

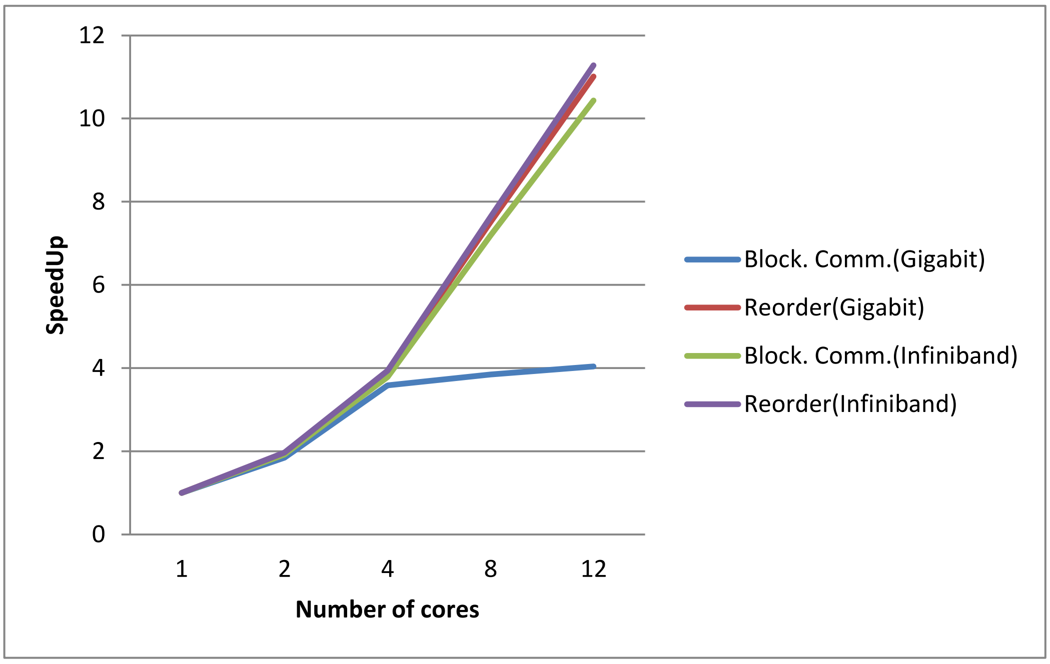


Figure 5.12. Comparison of SpeedUp results between the model with a parallel CG using blocking communication and another implementation using non-blocking communication and reordering of calculation for Tahoe50 in CIEMAT. Tests have been carried out with both Gigabit Ethernet and Infiniband.

5.3.7 Scalability study

In this section results of the final version of the hybrid parallel model using OpenMP and MPI are shown (Table 5.7), both using threads and processes in ALHAMBRA, in order to check its performance with up to a total of 256 subdivisions. The final version including all improvements, a parallel CG and MMIC as preconditioner, is also compared with other versions to assess how each one scales. The tests were performed by running a process by node (with a total of 16 processes in 16 nodes), each process also uses 16 threads, completely occupying each node.

	Basic		**ParallelCG(BJ)**		**ParallelCG(MMIC)**	
Threads	**Tiempo**	**SpeedUp**	**Tiempo**	**SpeedUp**	**Tiempo**	**SpeedUp**
1	103.81	1	107.73	1	103.36	1
2	60.02	1.72	55.31	1.95	52.16	1.98
4	34.51	3.01	28.11	3.83	26.62	3.88
8	19.52	5.32	14.02	7.68	13.25	7.80
16	12.35	8.41	9.04	11.92	8.42	12.27
32	11.13	9.33	4.31	24.99	4.05	25.52
64	10.52	9.87	2.43	44.33	2.08	49.69
128	10.24	10.14	1.43	75.33	1.15	89.88
256	10.11	10.27	0.72	149.62	0.67	154.27

Table 5.7. Results of scalability for the Basic parallel model and the parallel model using a parallel CG. On the one hand using Block Jacobi as the preconditioner and on the other hand the modified version of MIC (MMIC), using as parameters w and w2 those optimal values specified in Table 5.4.

The results show that the version that only includes the basic optimizations using O2 (Basic, Columns 1 and 2) scales very little above 16 threads, showing nearly constant execution time from that point. One can also observe the importance in the efficiency of the inclusion of a parallel PCG, both using the BJ preconditioner and MMIC (Columns 3-4 and 5-6 respectively). The best results of scalability are achieved for the last case with the MMIC preconditioner, adjusting the *w* and *w2* parameters as it is indicated in Table 5.4 according to number of subdomains used.

Figure 5.13 summarizes progress made from the implementation of the first parallel version to the best version used, including all the optimizations that have been made and the parallel PCG using MMIC as the preconditioner. The results are shown both in ALHAMBRA and in CIEMAT. CIEMAT is also used with the objective of representing this improved scalability (in high resolution grids such as Tahoe20) in shared memory platforms, using in this case from 1 to 64 threads. The scalability in ACII with Tahoe20 is not evaluated because its execution is not possible because of the high memory requirements.

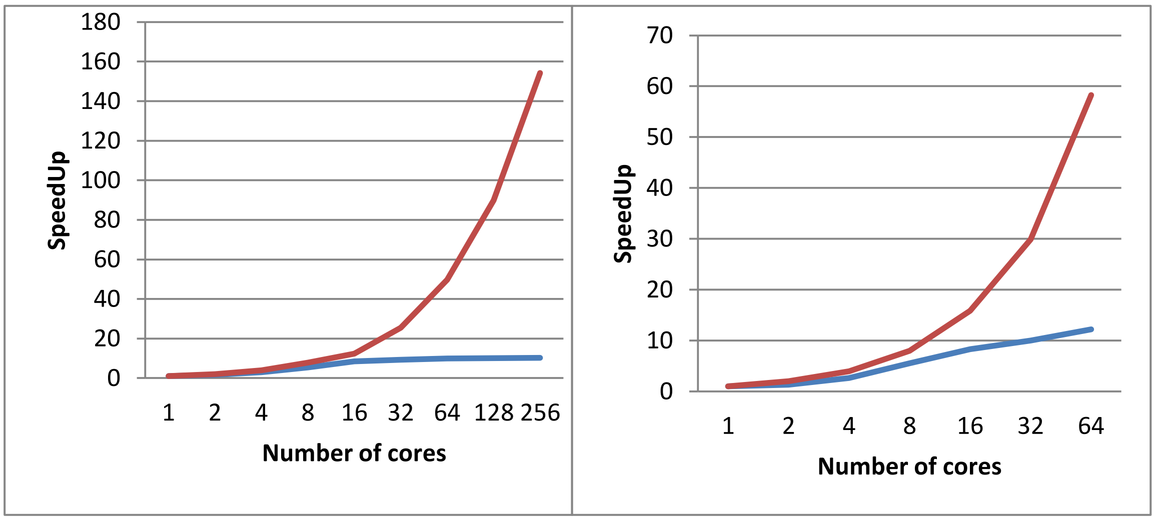


Figure 5.13. Results of scalability in ALHAMBRA (left) and CIEMAT (right) comparing the initial parallel version (Basic, blue line) and the parallel version including all the proposed improvements (ParallelCG (MMIC), red line).

The results show significant differences between the two versions. In the case of ALHAMBRA, the basic version is barely able to reduce the execution time from 16 cores, the loss of posterior efficiency is highly influenced by the explicit communications between nodes on a distributed memory platform such as ALHAMBRA. In CIEMAT, a shared memory machine, although the basic version scales slightly when more cores are used, the improvement is minimal. Conversely, good results of scalability are seen for the parallel version that includes all the improvements, in both ALHAMBRA where SpeedUp reaches 154 with 256 cores, and CIEMAT, where it reaches a SpeedUp of 58 with 64 cores.

In comparison with the original code using O2, substantial improvements have been achieved in successfully reducing the computational cost. The execution time of the basic optimizations Si3D is 120.36s in ALHAMBRA and 238.13s in CIEMAT.

These times present an approximate ratio of 360: 30 and 714: 30, respectively, meaning that it would take 360 days in ALHAMBRA and 714 days in CIEMAT to complete a computer simulation of Tahoe20 of 30 days. On the contrary, given the results of the best parallel version, this ratio is reduced to about 2:30 in ALHAMBRA using 256 cores and up to 10:30 in the case of CIEMAT using 64 cores, allowing us to obtain, with enough resources, useful results in an acceptable time.

5.4 Conclusions

- In this chapter various proposals are presented that, applied to a 3D hydrodynamic model, provide an optimized parallel implementation, both in shared memory machines using OpenMP and in distributed memory machines using MPI. For this, emphasis is made on adapting the parallel implementation specifically to the characteristics and type of operations that can be found in hydrodynamic models. Furthermore, we demonstrate that if the parallel implementation is adapted to the type of architecture used, the efficiency and scalability of the parallel model will improve considerably. In relation to this the following conclusions are obtained:
    - The use of affinity improves performance on platforms with shared memory and distributed memory, with an improvement up to 45%. The results show that the most appropriate map affinity varies in relation to the amount of memory available in the architecture used.
    - The use of synchronization points that, in the case of shared memory, are adapted to the NUMA architecture and to the domain decomposition used, improves the execution time obtained compared to the use of OpenMP barriers up to 11.67%. To achieve this, barriers have been implemented with a hierarchical structure and a specific synchronization for exchanges between neighboring subdomains.
    - Basic optimizations, like loop unrolling, improve the runtime of algorithms typically used in linear algebra such as the Thomas Algorithm for solving tridiagonal systems. This reduction is up 4% of the total execution time when the loop unrolling is applied to S1 and S4 of Si3D.

    - The type of storage used to exploit the data locality cannot be efficient if, in the case of shared memory, two threads compete for the access to information located in the overlapping area of shared variables. The results show that it is more efficient the use a local variable for each thread, allowing the access to these variables more frequently from cache memory and avoiding penalties in the access to main memory. Using local variables for specific calculations allows to obtain an improvement of 6.16% with respect to the use of shared variables.
    - Performing the computation of the water columns located on the border of each subdomain before the rest of interior columns, both during the calculation of PCG and in S3 and S4 of Si3D, it is possible to overlap all communication with computation, obtaining an improvement of the execution time up to 63%. With this alternative similar execution times are obtain using Infiniband and Gigabit Ethernet with differences lower than 2.51%.
    - For loops with little calculation and whose results are needed by neighboring subdomains, the results show that it is more efficient to solve the dependencies reducing the points of synchronization/communication by adding redundant computation. The redundant computation improves execution time up to 3.2% in the case of synchronization using threads and up to 15.5% in the case of communications using processes.
- The development of a parallel and optimized implementation stage where the system of pentadiagonal equations is solved improves the results considerably, thereby obtaining a model with good scalability results in high performance platforms with more than 10 nodes. The proposal presented here has been achieved by adapting the PCG to the architecture used and to the domain decomposition used in the other stages of Si3D, without adding any additional computational cost such as reordering or extra communications. Thanks to these optimizations, OP-Si3D is able to simulate a complete model of Lake Tahoe using high resolution with a relation of 2:30 using 256 cores in ALHAMBRA and a SpeedUp of 154.
- The choice of the preconditioner used by the PCG to solve the system of pentadiagonal equations significantly affects the efficiency of a parallel

implementation due to the added overhead by new factorization, reordering and/or communications. The alternative preconditioner MIC presented here (MMIC) is implemented without adding any type of calculation or communication/synchronization in its implementation, at the cost of reducing their efficiency when the number of subdomains used increases. However, this reduction is greatly enhanced by the alternative presented here, and the use of a parameter whose value varies according to the number of subdomains used. The results show that this alternative performs better than other alternatives evaluated, implemented both sequentially and in parallel.

Conclusiones Generales

Las estrategias de optimización propuestas en este trabajo permiten mejorar la eficiencia de modelos hidrodinámicos 3D ampliamente utilizados en la simulación de procesos de circulación, transporte y mezcla de aguas continentales. Estas propuestas han sido implementadas con éxito en un modelo de diferencias finitas semi-implícito (Si3D), aunque son suficientemente genéricas para ser implementadas de igual forma en modelos similares. Los resultados demuestran que se ha conseguido reducir el coste computacional de simulaciones excesivamente costosas por usar grids de alta resolución, las cuales serían imposibles de realizar sin las optimizaciones propuestas. Estas estrategias permitirán realizar estas simulaciones en clusters de gama baja en un tiempo de ejecución aceptable o incluso aprovechar de forma eficiente arquitecturas de altas prestaciones para la simulación de modelos de mayor coste computacional. Las propuestas realizadas en este trabajo han sido o están siendo usadas en varios proyectos. Por ejemplo, se han utilizado para llevar a cabo simulaciones con modelos de alta resolución en el estudio de procesos de media y pequeña escala en diversas zonas de interés como la confluencia de ríos, el meandro de un río o la entrada de un río en un lago.

También se han utilizado en el estudio de procesos de pequeña escala en la zona litoral de lagos y embalses (http://terc.ucdavis.edu/research/modeling/three-d-tahoe.html), o en el estudio de procesos de circulación de gran escala en lagos de gran tamaño, como son el lago Tahoe en California y el lago Tanganika en el este de África (http://terc.ucdavis.edu/publications/newsletters1/winter2014-15.pdf, http://terc.ucdavis.edu/research/world/lake-tanganyika.html). Además, haciendo uso de la versión optimizada de Si3D propuesta aquí, ha sido posible el desarrollo de varias publicaciones en revistas internacionales JRC de prestigio y varias tesis doctorales del Dpto. de Ingeniería Civil de la Universidad de Granada.

Como ejemplo de las mejoras logradas, la simulación completa del lago Tahoe utilizando una versión secuencial de Si3D y un modelo de alta resolución con columnas de agua de 20x20m, tiene una relación de 3000:30 aproximadamente con respecto a la versión original de Si3D y una relación de 360:30 aplicando diversas optimizaciones básicas, esto quiere decir que se tardarían unos 3000 o 360 días de cómputo respectivamente en simular 30 días de forma secuencial. Sin embargo, gracias a la implementación paralela realizada siguiendo las estrategias propuestas en este trabajo, esa relación puede reducirse a 6:30 usando 64 cores de una arquitectura de memoria distribuida y a 2:30 usando 256 cores.

A continuación se enumeran una serie de conclusiones específicas de cada capítulo:

- Implementación anidada N-Si3D:
    - Se presenta un procedimiento de anidamiento de una vía para simular procesos de circulación en modelos de larga escala. La implementación anidada fue validada comparando los resultados de diversas simulaciones de zonas concretas (subdominios) de los modelos de un lago y un río, obteniendo resultados de cada zona concreta tanto por un modelo completo como por el modelo anidado. En las pruebas de validación realizadas, las diferencias comparando los resultados del modelo anidado en alta resolución con un modelo completo también en alta resolución son en todos los casos menores al 4%. También se demuestra que estos errores pueden llegar a ser 0 cuando ambos modelos (el de baja completo y el de alta anidado) tienen la misma resolución y el sistema de ecuaciones pentadiagonal para resolver superficie libre, presente en modelos semi-implícitos, es resuelto usando un método directo, el cual es un método computacionalmente muy costoso pero con una solución exacta. Los errores producidos por el uso de una implementación anidada son debidos por un lado a la naturaleza aproximada del método iterativo usado en Si3D para resolver el sistema de ecuaciones pentadiagonal y, por otro lado, al uso de interpolación espacial y las diferencias entre batimetrías (ambas consecuencia por la diferencia de resolución entre el modelo completo de baja y el modelo anidado de alta).
    - La información que debe ser transmitida desde el modelo completo de baja resolución al modelo anidado de alta resolución, con el objetivo de construir un procedimiento de anidamiento correcto, largamente depende de la forma en que se implemente. El método utilizado es importante para determinar qué dependencias espaciales existen en la correcta

construcción de las ecuaciones de conservación discretizadas del modelo anidado. La calidad de los resultados del modelo anidado se ven severamente afectados si no se transmiten todas las variables necesarias. Esto se demuestra, en particular, realizando un estudio con uno de los componentes de velocidad (velocidad tangencial) que se debe transmitir a lo largo de la frontera desde el modelo completo de baja resolución al modelo anidado de alta resolución, el cual sin embargo no es transmitido en muchas aplicaciones presentes en la literatura. Este componente, aun no siendo necesario para garantizar la conservación de masa en el modelo anidado, puede afectar a la calidad de los resultados cuando existen corrientes de fuerte intensidad, circulación lateral y/o vórtices cerca de la frontera anidada. En estos casos los errores se deben al error en el cálculo del término advectivo en las ecuaciones de momentum cerca de la frontera en el modelo anidado.

- Se demuestra la necesidad de usar grids de alta resolución para simular correctamente patrones de circulación de pequeña escala, imposibles de capturar con una resolución menor en dos casos reales: el lago Tahoe y el río Sacramento. Se observa que la recirculación horizontal en una bahía del lago Tahoe y la recirculación vertical en un meandro del río Sacramento no son correctamente simulados en modelos de menor resolución. Estos patrones, sin embargo, sí son característicos del tipo de área donde se presentan (como se indica en la literatura) y son reproducidos de forma similar por modelos completos de alta resolución.
- La implementación del procedimiento de anidamiento presentada permite la ejecución de los modelos de alta y baja resolución en paralelo. Para ello se aplica una estructura pipeline para acoplar ambos modelos de forma online, siendo el tiempo de ejecución final equivalente al del modelo más lento, el cual dependiendo de la aplicación, será el modelo completo baja resolución o el anidado de alta resolución. Además, las comunicaciones se encuentran implementadas de forma eficiente al ser solapadas con cálculo. Este acoplamiento online evita el uso de ficheros para la transferencia de información entre el modelo de baja y el anidado de alta y hace posible transferencias incluso a cada paso de tiempo, evitando errores por interpolación temporal. Un fichero con la

información que se transfiere en la implementación propuesta tendría un tamaño prohibitivo (de hasta 600GB para el lago Tahoe con alta resolución).

- Implementación paralela para pequeños *cluster* de computadores, P-Si3D:
    - La implementación paralela P-Si3D es probada con éxito y con un tiempo de ejecución aceptable en modelos de media y baja resolución del lago Tahoe, obteniendo una relación de 5:30 usando los 3 nodos de ACII (12 hebras) para el caso de media resolución.
    - Diversas configuraciones de la arquitectura utilizada combinando distintos números de procesos y hebras fueron evaluados. Los resultados demuestran que el rendimiento es peor cuando las opciones por defecto de la BIOS y del sistema operativo son utilizadas (el tiempo de ejecución aumenta entre un 40% y un 60% dependiendo del número de hebras y procesos usados). Esto se debe a la distribución de hebras utilizada por defecto por el sistema operativo entre los 8 cores lógicos de un nodo cuando se activa Hyper-Threading. El programa se aprovecha solo de forma leve del prefetching hardware (reduciendo el tiempo de ejecución entre un 5% y 8%) y de Intel® Turbo Mode Technology, reduciendo el tiempo de ejecución levemente, entre un 5% y 8% en el primer caso y entre un 3% y 7% en el segundo caso.
    - El procesamiento por bloques reduce el tiempo de ejecución de forma muy moderada (un 4% de mejora aproximadamente).
    - La comparación de dos tipos de corte en la descomposición de dominio (corte-estrecho y corte-largo), tanto con cálculo redundante como sin él, demuestra que aunque el número de comunicaciones es el mismo en ambos casos, la cantidad de datos a calcular (de forma redundante) y a comunicar varía. Los resultados demuestran que la mejor distribución es aquella donde se presente fronteras entre subdominios de menor longitud.

- Aplicación de trazadores usando P-Si3D y N-Si3D, P/N-Si3D:
    - La combinación del procedimiento de anidamiento N-Si3D, usado para reducir el área de interés en alta resolución a la zona litoral de un lago, y P-Si3D, usado para dividir la carga de trabajo usando descomposición de dominio, ha permitido obtener resultados de todo el perímetro del lago Tahoe en alta resolución en pequeños clusters. Los resultados obtenidos con este modelo son usados para ilustrar que las bahías de un lago no pueden ser estudiadas de forma individual y están conectadas con otras bahías vecinas a través de corrientes que se producen a lo largo de la costa. Las bahías y otras irregularidades en la línea de costa pueden atrapar material transportado por corrientes a lo largo de la costa como consecuencia de remolinos producidos en estas bahías. Estos procesos parecen ser más fuertes durante periodos de viento de mayor intensidad.

- Implementación paralela escalable para *clusters* de bajo coste, SP-Si3D:
    - Se demuestra que a la hora de desarrollar una implementación paralela de modelos hidrodinámicos 3D, la resolución de un sistema de ecuaciones pentadiagonal, presente en modelos semi-implícitos, no es una tarea trivial y requiere costosas técnicas de reordenamiento y comunicaciones, las cuales serán de uno-a-muchos y muchos-a-uno si se resuelve el sistema en secuencial o de muchos-a-uno y entre subdominios vecinos si se resuelve en paralelo. Los resultados demuestran que una implementación secuencial de la resolución del sistema de ecuaciones pentadiagonal haciendo uso de las comunicaciones de uno-a-muchos y muchos-a-uno, provoca una reducción inaceptable de la escalabilidad del modelo.
    - Una estructura de procesamiento en dos niveles (usando una estructura pipeline y descomposición de dominio) es propuesta para poder llevar a cabo la simulación de zonas anidadas en alta resolución demasiado extensas, como puede ser toda la zona litoral de grandes lagos. Con este método, la zona anidada se divide en múltiples subdominios anidados, permitiendo así que las ecuaciones del modelo puedan ser resueltas de forma independiente en cada subdominio, usando para ello las velocidades del modelo completo de baja resolución como condiciones

de frontera. Con este procedimiento, cada subdominio anidado ensambla y resuelve un subsistema de ecuaciones pentadiagonal independiente, sin añadir comunicaciones o reordenamiento en su cálculo. Con este tipo de procesamiento en dos niveles el tiempo total de ejecución se reduce al tiempo de ejecución de un subdominio anidado de alta resolución.

- La estructura de procesamiento en dos niveles presentada permite reducir el tiempo de ejecución del modelo anidado en alta resolución casi de forma lineal con el número de ordenadores usados en clusters de gama baja. SP-Si3D logra obtener un *SpeeUp* de 7.91 dividiendo la zona litoral del lago Tahoe en 8 subdivisiones. Con esta escalabilidad se logra simular toda la zona litoral del lago Tahoe en alta resolución obteniendo una relación de 6:30 en ARCHIMEDES.
- Las diferencias (NRMSE) al comparar los resultados de SP-Si3D con los resultados de un modelo completo de alta resolución son en todos los casos muy pequeñas (menores al 4%). Se demuestra que SP-Si3D es capaz de simular correctamente fenómenos de pequeña escala aunque el límite entre subdominios atraviese uno de ellos (con un error menor al 2,23% en el caso estudiado).

- Implementación paralela optimizada para plataformas de altas prestaciones OP-Si3D.
    - En una implementación paralela de un modelo hidrodinámico 3D, el tipo de arquitectura que se utilice, la adaptación a la arquitectura utilizada y la mejora de todas las etapas afectarán considerablemente a la eficiencia y escalabilidad del modelo paralelo. En relación con este apartado se obtienen las siguientes conclusiones:
        - El uso de afinidad mejora las prestaciones en las plataformas con memoria compartida y memoria distribuida utilizadas, con una mejora de hasta un 45%. Los resultados demuestran que el mapa de afinidad más adecuado varía según la cantidad de memoria disponible de la arquitectura utilizada.
        - El uso de puntos de sincronización que, en el caso de memoria compartida, se adapten a la arquitectura NUMA y a la descomposición de dominio utilizada, mejora los tiempos

obtenidos comparando con el uso de las barreras de OpenMP hasta un 11,67%. Esto se consigue haciendo uso de barreras con una estructura jerarquizada y una sincronización específica para los intercambios entre subdominios vecinos.

- Optimizaciones básicas como desenrollado de bucles mejora el tiempo de ejecución de algoritmos típicamente utilizados en algebra lineal como es el Algoritmo de Thomas para la resolución de sistemas tridiagonales hasta un 4%.
- El tipo de almacenamiento usado para aprovechar la localidad de los datos puede no ser eficiente si en el caso de memoria compartida, dos hebras compiten por el acceso a información situada en la zona de solapamiento de variables compartidas. Los resultados demuestran que es más eficiente el uso variables locales a cada hebra, permitiendo su acceso de forma más frecuente desde memoria caché y evitando penalizaciones por acceso a memoria principal. El uso de variables locales para determinados cálculos permite obtener una mejora de hasta un 6,16% con respecto al uso de variables compartidas.
- Llevando a cabo el cómputo de las columnas de agua situadas al borde de cada subdominio antes que el resto de columnas internas, tanto durante el cálculo del PCG como en las etapas 3 y 4 de Si3D, es posible solapar todas las comunicaciones con cálculo con una mejora del tiempo de hasta un 63%. Con esta alternativa se obtienen tiempo similares usando Infiniband y Gigabit Ethernet, con diferencias menores a un 2,51% entre ambos.
- En el caso de bucles con poco cálculo y cuyos resultados son necesarios por subdominios vecinos, los resultados demuestran que es más eficiente solventar las dependencias reduciendo los puntos de sincronización/comunicación y añadiendo cálculo redundante. El cálculo redundante mejora el tiempo de ejecución hasta un 3,2% en el caso de sincronización usando hebras y hasta un 15,5% en el caso de comunicaciones usando procesos.

- El desarrollo de una implementación paralela y optimizada de la etapa donde se resuelve el sistema de ecuaciones pentadiagonal permite mejorar los tiempos y la escalabilidad en plataformas de altas prestaciones con un número de nodos mayor que 10. Gracias a estas optimizaciones, OP-Si3D consigue simular un modelo completo de alta resolución del lago Tahoe con una relación de 2:30 usando 256 cores en ALHAMBRA y un *SpeedUp* de 154.
    - En la propuesta presentada aquí se ha conseguido adaptar el PCG a la arquitectura utilizada y a la descomposición de dominio usada en el resto de etapas de Si3D, sin añadir ningún tipo de coste computacional adicional como reordenamiento o comunicaciones extras.
    - La elección del precondicionador utilizado por el PCG en la resolución del sistema de ecuaciones pentadiagonal afecta de forma considerable a la eficiencia de una implementación paralela debido al *overhead* añadido por factorizaciones, nuevo reordenamiento y/o comunicaciones. La alternativa del precondicionador MIC presentada aquí (MMIC) es implementada sin añadir ningún tipo de cómputo o comunicación/sincronización adicional en su implementación, a costa de reducir su eficiencia a medida que aumentan el número de subdominios. Sin embargo, esta reducción se ve considerablemente mejorada gracias a la alternativa presentada aquí y al uso de un parámetro cuyo valor varía según el número de subdominios. Los resultados demuestran que esta alternativa obtiene mejores resultados que otras alternativas evaluadas tanto en secuencial como en paralelo.

General Conclusions

Several optimization approaches have been proposed in this work to improve the efficiency of three-dimensional 3D hydrodynamic models when applied to simulate circulation, transport and mixing processes in inland waters. These approaches were successfully implemented in a semi-implicit finite difference code (Si3D), but they are general enough so that they can also be implemented in similar models. Using the optimized version of Si3D, we were able to conduct simulations with very high-resolution grids with reasonable run-times, in small commodity clusters. The optimized model has also been applied or is currently being used, outside this dissertation, in several projects. For example, it has been used to conduct very-high resolution simulations of medium- or small-scale river confluences, river bends, and river inflows in reservoirs. It has also been used in very-high resolution simulations of near-shore processes in lakes and reservoirs (http://terc.ucdavis.edu/research/modeling/three-d-tahoe.html), or in the simulation of the large-scale circulation of large and very large lakes, such as Lake Tahoe and Lake Tanganyika in north of Africa (http://terc.ucdavis.edu/publications/ newsletters1/winter2014-15.pdf, http://terc.ucdavis.edu/research/world/lake-tanganyika.html). In addition, several manuscripts have been published in international JRC journals and various PhD thesis of the Civil Engineering Department at the University of Granada University have been completed, in which use has been made of optimized versions of Si3D proposed here.

As an example of the improvements, the complete simulation of Lake Tahoe, using a sequential model of Si3D and a high resolution grid with 20 x 20m grid cells, has a ratio of 3000:30 compared to the original version of Si3D (including O2) and a ratio of 360:30 adding various basic optimizations. This means that it would take about 3000 or 360 days respectively to simulate 30 days. However, with our parallel implementation following the strategies proposed in this work, that ratio can be reduced to 6:30, using 64 cores in a distributed memory architecture and 2:30 using 256 cores.

Specific conclusions of each chapter are enumerated next:

- Nested Implementation N-Si3D:
    - A one-way nested implementation was developed to simulate localized circulation features in large-scale models. The nested implementation was validated by comparing the results of simulations in a small region (sub-domain) of a lake or river model, calculated both by the nested- or inner-model and the complete or outer-model. In the validation experiments, the results of the high resolution nested model differs in all cases less than 4% from the results of a complete model using high resolution too. These errors are even reduced to 0 when both models (inner and outer model) use the same grid resolution and the pentadiagonal matrix for water surface elevation built in the semi-implicit model was solved using a direct method, which is computationally demanding but exact. The errors produced in a nesting implementation are due to the approximate nature of the iterative matrix solvers used in Si3D to solve the pentadiagonal matrix, and to spatial interpolation and bathymetry mismatch (both consequence of the different inner and outer grid resolutions).
    - The information that needs to be transmitted from the outer to the inner model to construct a seamless nesting implementation largely depends on the computational stencil, which dictates the spatial dependencies in the construction of the correct discretized conservations equations. If not all the variables are transmitted the quality of the results of the nested model may be severely affected. The velocity components along the boundary between the inner- and the outer- model (tangential velocities), in particular, are seldom transmitted across nested boundaries in many applications. This approach, while unimportant to guarantee mass conservation in the inner model, may be wrong when very strong currents, lateral circulation and/or vortices exist near the nested boundary. The errors in that case result from miscalculations of the magnitude of the advection terms in the momentum equations of the inner model near the boundary.

- The usefulness of the nested approach is illustrated in two test cases. In one of them we simulate the lateral circulation that develops a river bend in Sacramento River. In the other, the nested model is used to simulate the development of separation eddies in a small bay of a large lake (Lake Tahoe, CA), which occurs in response to changes in wind forcing and large-scale circulation. These features can only be represented with sufficiently high spatial resolution.
- The outer- and the inner- models are run in parallel. A pipeline structure for coupling both models online is used, being the final execution time equivalent to the slowest model, which, depending on the particular implementation, it can be either the low-resolution or the nested high-resolution model). In addition, communications are implemented efficiently by being overlapped with calculation. This online coupling prevents the use of files for transferring information between low and high resolution models and makes possible the communication even each time-step, avoiding errors by temporal interpolation. A file with the information necessary to be transferred in the proposed implementation would have a prohibitive size (up to 600GB in the case of Lake Tahoe using a high resolution nested grid).

- Parallel Implementation for small commodity clusters P-Si3D
    - The parallel implementation of Si3D (P-Si3D) was used to build a mid-resolution model of Lake Tahoe, with cartesian grid cells that were 50 m wide. This model was run in ACII, a small commodity cluster of three nodes (12 threads), with ratios between computational to real time of 5:30.
    - Different platform configurations with varying number of threads and processes were tested. The results show that performance is worse if the default BIOS and operating system configuration is used (time increases by between 40% to 60% depending on the number of processes and threads used). This is due to the thread distribution of the operating system among the eight logical cores of a node when Hyper-Threading is enabled. The program makes a weak use of the pre-fetching hardware (pre-fetching decreases execution time by between 5% to 8%) and Intel®

Turbo Mode Technology decreases slightly the execution time (by between 3% to 7%).

- Block-driven processing reduces execution time too slightly (4% improvement approximately).
- The comparison of wide-direction or narrow-direction distribution in a parallel implementation, with MPI communications and with or without redundant calculation, shows that though the number of communications is the same, the quantity of data to calculate (with redundancy) or communicate varies. The results prove that the distribution approach more efficient is the one with a lower border length.

- Tracers application using P-Si3D and N-Si3D, P/N-Si3D
  - The nesting procedure, used to reduce the high resolution area of interest to the littoral zone of a lake, and the parallelized version of the code P-Si3D, used to divide the workload using domain decomposition, have been combined to conduct high-resolution (20 x 20 m Cartesian grid cells in the horizontal) simulations of the near-shore perimeter of Lake Tahoe using a small commodity cluster. The model results are used to illustrate that the physical and chemical environments in neighboring littoral embayments are tightly linked as a result of pulsating long-shore currents that develop within the coastal boundary layers. Bays and other shoreline irregularities may trap material transported by long-shore currents, as a result of the development of separation eddies. Long-shore currents and local bay-scale eddies appear to become more energetic in response to stronger wind events.

- Scalable parallel implementation in small commodity clusters, SP-Si3D
  - The solution of a pentadiagonal matrix for the free-surface elevation in semi-implicit models in parallel implementations of the model can be done either sequentially or in parallel. In both cases it requires expensive reordering techniques and communications between processes/threads. These communications are one-to-many and many-to-one if the system is solved sequentially or many-to-one and between neighbor sub-domains

if it is solved in parallel. The sequential implementation using one-to-many and many-to-one communications of the matrix solution seriously jeopardizes the scalability of the parallel implementation.

  - A processing structure on two levels (using pipeline structure and domain decomposition) is proposed to conduct scalable approaches to simulate extensive near-shore domains with high spatial resolution. In this approach, the near-shore region is simulated using multiple nested sub-domains. The equations are solved independently in each sub-domain, using the velocities from the outer-model as boundary conditions. Hence, in each nested sub-domain a pentadiagonal matrix is assembled and solved independently, without any reorder and communications added. With this two-level processing, the total execution time is reduced to the execution time of one high-resolution sub-domain.
  - The two–level processing structure scales almost linearly with the number of computers used in small commodity clusters. A SpeedUp of 7.91 was achieved when the high-resolution (20x20 horizontal grid cells) nested model of the littoral zone of Lake Tahoe is solved with 8 nodes. In Archimedes, a cluster existing in the Water Research Institute, the ratio of computational to real time in these computations where 6:30.
  - The differences (NRMSE) when comparing the results of SP-Si3D with the results of a complete high-resolution model are in all cases very small (less than 4%). It is demonstrated that SP-Si3D can correctly simulate small-scale patterns although the boundary between nested subdomains crosses one of this pattern (with an error less than 2.23% in the case studied).

- Optimized parallel implementation for high performance platforms OP-Si3D.
  - In a parallel implementation of a 3D hydrodynamic model, the type of architecture used, the adaptation to the architecture used and the improvement of all stages, significantly affect the efficiency and scalability of the code. In connection with this points the following conclusions are obtained:

- The use of affinity improves performance on platforms with shared memory and distributed memory, with an improvement up to 45%. The most appropriate affinity map varies in relation to the amount of memory available in the architecture used.
- The use of synchronization points that, in the case of shared memory, are adapted to the NUMA architecture and to the domain decomposition used, improves the execution time obtained compared to the use of OpenMP barriers up to 11.67%. To achieve this, barriers have been implemented with a hierarchical structure and a specific synchronization for exchanges between neighboring subdomains.
- Basic optimizations, like loop unrolling, improve the runtime of algorithms typically used in linear algebra such as the Thomas Algorithm for solving tridiagonal systems up to 4%.
- The type of storage used to exploit data locality cannot be efficient if, in the case of shared memory, two threads compete for the access to information located in the overlapping area of shared variables. The results show that it is more efficient the use a local variable for each thread, allowing the access to these variables more frequently from cache memory and avoiding penalties in the access to main memory. By using local variables for specific calculations one can achieve an improvement of 6.16% with respect to the use of shared variables.
- Performing the computation of the water columns located on the border of each sub-domain before the rest of internal columns, both during the calculation of PCG and in S3 and S4 of Si3D, it is possible to overlap all communication with computation, obtaining an improvement of the execution time up to 63%. With this alternative similar execution times are obtain using Infiniband and Gigabit Ethernet with differences lower than 2.51%.
- For loops with little calculation and whose results are needed by neighboring sub-domains, the results show that it is more efficient to solve the dependencies reducing the points of

synchronization/communication by adding redundant computation. The redundant computation improves execution time up to 3.2% in the case of synchronization using threads and up to 15.5% in the case of communications using processes.

- The development of a parallel and optimized implementation of the pentadiagonal matrix solution, improves the execution time and scalability when high performance platforms with a number greater than 10 nodes are used. Using these optimizations, OP-Si3D is able to simulate a complete model of Lake Tahoe using high resolution with a relation of 2:30 using 256 cores in ALHAMBRA and a SpeedUp of 154.
    - The PCG was adapted to the architecture and the domain decomposition used in the other stages of Si3D, without adding any additional computational cost such as reordering or extra communications.
    - The choice of preconditioner significantly affects the efficiency of a parallel PCG solver due to the added overhead by new factorization, reordering and/or communications. The alternative preconditioner MIC presented here (MMIC) is implemented without adding any type of calculation or communication/synchronization in its implementation, at the cost of reducing their efficiency when the number of sub-domains used increase. However, this reduction is greatly enhanced by the alternative presented here and the use of a parameter whose value varies according to the number of subdomains used. This alternative performs better than other sequential and parallel alternatives.

Scientific Production and Activity

Publications

Acosta MC, Anguita M, Fernández-Baldonero FJ, Ramón CL, Schladow SG, Rueda FJ. Evaluation of a nested-grid implementation for 3D finite-difference semi-implicit hydrodynamic models. Environmental Modelling and Software 2015; **64**: 241-261. doi:10.1016/j.envsoft.2014.10.015. Impact Factor: 4.538.

Anguita M, **Acosta MC**, Fernández-Baldonero FJ, Rueda FJ. Scalable Parallel implementation for 3D semi-implicit hydrodynamic models. Environmental Modelling and Software 2015, (under review). Impact Factor: 4.538.

International Conferences

Title: Parallel Implementation of a Semi-implicit 3D Lake Hydrodynamic Model.

Authors: **Acosta MC**, Anguita M, Fernández-Baldonero FJ, Rueda FJ.

Name of the Conference: International on Computational and Mathematical Methods in Science and Engineering.

Location: Almería, Spain.

Date: 07/2010

Title: Spatial Expansion of an Invasive by Wind-Driven Currents.

Authors: Hoyer AB, **Acosta MC**, Anguita M, Fernandez-Baldomero J, Schladow G, Rueda FJ.

Name of the Conference: Symposium for European Freshwater Science (SEFS).

Location: Girona, Spain.

Date: 04/2011

Title: Mixing of Density Currents Inflowing a Meditarrean Stratified Reservoir (Spain).

Authors: Cortés A, **Acosta MC**, Rueda FJ.

Name of the Conference: Annual Conference of the International Association for Great Lakes Research (IAGLR).

Location: Ontario, Canada.

Date: 05/2012

Title: Heterogeneus Parallel Implementation of a Semi-implicit 3D Hydrodynamic Model.

Authors: **Acosta MC**, Anguita M, Fernández-Baldonero FJ, Rueda FJ.

Name of the Conference: Computational Methods in Water Resources, XIX International Conference.

Location: Urbana-Champaign, Illinois, USA.

Date: 06/2012

Title: N1SI3D: A One-way nested grid procedure for a 3D Hydrodynamic Model.

Authors: **Acosta MC**, Anguita M, Fernández-Baldonero FJ, Rueda FJ.

Name of the Conference: International Conference on Approximation Methods and Numerical Modeling in Environment and Natural Resources: MAMERN'13.

Location: Granada, Spain.

Date: 03/2013

Title: Prediction of the pathways of river water entering a Mediterranean stratified reservoir (Beznar, Spain).

Authors: Cortés A, **Acosta MC**, Rueda FJ.

Name of the Conference: International Conference on Approximation Methods and Numerical Modeling in Environment and Natural Resources: MAMERN'13.

Location: Granada, Spain.

Date: 03/2013

National Conferences

Title: Evaluación de la paralelización de un modelo hidrodinámico 3D.

Authors: **Acosta MC**, Anguita M, Fernández-Baldonero FJ, Rueda FJ.

Name of the Conference: XXII Jornadas de Paralelismo.

Location: Tenerife, Spain.

Date: 06/2011

Research Visit

Host University: University of California, Davis.

Tutor: S. Geoff Schladow

Departament: Civil and Environmental Engineering.

Location: Davis, California, USA.

Dates: 06/2013 to 09/2013

Co-Direction of Final Degree Projects

Student: Felipe Torres González

Principal Director: Mancia Anguita

Degree: Computer Engineering

University: University of Granada (Spain)

End of project: In progress (June 2015).

Student: Francisco

Principal Director: Francisco J. Rueda

Degree: Civil Engineering

University: University of Granada (Spain)

End of project: In progress (June 2015)

Grants

Name: Researcher Worker

Objective: Pre-doctoral

Organization: University of Granada

Dates: 15/02/2010 to 14/02/2011

Name: FPU Plan Propio (UGR)

Objective: Pre-doctoral

Organization: University of Granada

Dates: 01/03/2011 to 28/02/2015

Name: Plan Propio de Movilidad Internacional

Objective: Research Visit

Organization: University of Granada

Dates: 06/2013 to 09/2013

References

Acosta MC, Anguita M, Fernández-Baldonero FJ, Ramón CL, Schladow SG, Rueda FJ. Evaluation of a nested-grid implementation for 3D finite-difference semi-implicit hydrodynamic models. Environmental Modelling and Software 2015; **64**: 241-261.

Acosta MC, Anguita M, Rueda FJ, Fernandez FJ. Parallel Implementation of a Semi-Implicit 3-D Lake Hydrodynamic Model. Proc. of the 2010 Int. Conf. on Comp. and Math. Methods in Science and Eng. (CMMSE) 2010**; IV**: 1026-1037. Almería (Spain).

Al-Khalissi H, Berekovic M, Marongiu A. On the Relevance of Architectural Awareness for Efficient Fork/Join Support on Cluster-Based Manycores. Proceedings of international Workshop on Manycore Embedded Systems 2014. DOI:10.1145/2613908.2613911.

Al-Khalissi H. Efficient Barrier Synchronization for OpenMP-like Parallelism on the Intel SCC. Parallel and Distributed Systems (ICPADS) 2013. DOI:10.1109/ICPADS.2013.15

Amestoy PR, Duff IS, Excellent JY. Multifrontal Parallel Distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Engrg*. 2000; **184**: 501-520.

Amritkar A, Tafti D, Liu R, Kufrin R, Chapman B. OpenMP Parallelism for fluid and fluid-particulate systems. *Parallel Computing* 2012; **38**: 501-517.

Anderson E, Schwab D, Lang G. Real-Time Hydraulic and Hydrodynamic Model of the St. Clair River, Lake St. Clair, Detroit River System. *J. Hydraul. Eng.* 2010: **136**(8), 507–518.

Anderson MA, Stewart MH, Yates MV Gerva CV. Modeling the impact of Body-contact Recreation of Pathogen Concentrations in a Source Drinking Water Reservoir. *Water Research* 1998. 32; **11**: 3293-3306.

Anguita M, Diaz J, Ros E, Fernandez-Baldomero FJ. Optimization strategies for high-performance computing of optical-flow in general-purpose processors. *Circuits and Systems for Video Technology, IEEE Transactions* 2009; **10**: 1475-88.

Anguita M, Fernandez-Baldomero FJ. Software optimization for improving student motivation in a computer architecture course. *IEEE Transactions on Education* 2007; **4**: 373-378.

Anguita M, Martinez-Lechado JM. MP3 optimization exploiting processor architecture and using better algorithms. *IEEE Micro* 2005; **3**: 81-92.

Appt J, Imberger J, Kobus H. Basin-scale motion in stratified upper lake constance. Limnology and Oceanography 2004; 49(4):919–933.

Armengol J, Toja J, Vidal A. Seasonal rhythm and secular changes in Spanish reservoirs. In: Limnology Now: A Paradigm of Planetary Problems. *(Ed.) R. Margalef. Ed.Elsevier* 1994; 237-253.

Asunci M, Mantas J and Castro M. Simulation of one-layer shallow water systems on multicore and CUDA architectures. *The Journal of Supercomputing* 2010; online 10 March 2010.

Balaji V. The FMS Manual: A developer's guide to the GFDL Flexible Modeling System. Geophysical Fluid Dynamics Laboratory 2002.

Barnes SL. Applications of the Barnes objective analysis scheme. Part I: effects of undersampling, wave position, and station randomness. *J. of Atmos. and Oceanic Tech.* 1994; **11**: 1433-1448.

Barnes SL. Mesoscale objective analysis using weighted time-series Observations. NOAA Technical Memorandum 1964. National Severe Storms laboratory.

Barrett R, Berry M, Chan T, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R et al. Templates for the solution of linear systems: Building blocks for iterative methods (SIAM) 2004. Philadelphia, PA (EEUU).

Barth A, Alvera-Azcárate A, Rixen M, Beckers J. Two-way nested model of mesoscale circulation features in the Ligurian Sea. *Progress in Oceanography* 2005; **66:** 171-189. DOI: http://dx.doi.org/10.1016/j.pocean.2004.07.017.

Beare MI, Stevens DP. Optimisation of a parallel ocean general circulation model. *Annales Geophysicae* 1997; **15**: 1369-1377. DOI: 10.1007/s00585-997-1369-3.

Benzi M. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics* 2002; **182**: 418-477.

Bhattacharjee S, Patidar S. Narayanan PJ. Real-time rendering and manipulation of large terrains. Computer vision, graphics & image processing, 2008. ICVGIP '08. sixth indian conference on.

Bjorstad PE, Moe R, Skogen M. Parallel domain decomposition and iterative refinement algorithms. *Notes on Numerical Fluid Mechanics* 1993; 28-46.

Blayo E, Debreu L. Nesting ocean models. In: E. Chassignet and J. Verron (Ed.). *An Integrated View of Oceanography: Ocean Weather Forecasting in the 21st Century*. Kluwer, 2006.

Blayo E, Debreu L. Revisiting open boundary conditions from the point of view of characteristic variables. *Ocean Modelling* 2005; **9**: 231-252. DOI: http://dx.doi.org/10.1016/j.ocemod.2004.07.001.

Bliznak M, Dulik T Jasek R. Performance Analysis of Built-in Parallel Reduction`s Implementation in OpenMP C/C++ Language Extension. *Advances in Intelligent Systems and Computing* 2014; **285**: 607-617.

Blumberg AF, Herring HJ. Circulation modeling using orthogonal curvilinear coordinates, in Nihoul, JCJ, and Jamart, BM, eds. Three-dimensional models of marine and estuarine dynamics: Amsterdam, Netherlands, Elsevier 1987; 55−88.

Blumberg AF, Mellor GL. A Description of a Three-Dimensional Coastal Ocean Circulation Model. Three-Dimensional Coastal Ocean Models. American Geophysical Union, 1987: 1-16. http://www.stevens-tech.edu/ses/ceoe/fileadmin/ceoe/pdf/alan_publications/AFB032.pdf.

Blumberg AF. Turbulent mixing processes in lakes, reservoirs and impoundments. *Physics-based modelling of lakes, reservoir and impondments* 1986: 79-104. New York (EEUU)

Bonaventura L, Rosatti G. A cascadic conjugate gradient algorithm for mass conservative, semi-implicit discretization of the shallow water equations on locally refined structured grids. *International Journal for Numerical Methods in Fluids* 2002; **40**: 217-230. DOI: 10.1002/fld.274.

Brooks DA, Baca MW, Lo YT. Tidal Circulation and Residence Time in a Macrotidal Estuary: Cobscook Bay, Maine. Estuarine, Coastal and Shelf Science 1999; **49**: 647-665. DOI: http://dx.doi.org/10.1006/ecss.1999.0544.

Brown R, Sharapov I. High-scalability parallelization of a molecular modeling application: performance and productivity comparison between OpenMP and MPI implementations. *International Journal of Parallel Programming* 2007; **35**, 441-458.

Bryan K. A numerical method for the study of the circulation of the world ocean: *Journal of Computational Physics* 1969; **4**: 347−376.

Cailleau S, Fedorenko V, Barnier B, Blayo E, Debreu L. Comparison of different numerical methods used to handle the open boundary of a regional ocean circulation model of the Bay of Biscay. *Ocean Modelling* 2008; **25**: 1-16. DOI: http://dx.doi.org/10.1016/j.ocemod.2008.05.009.

Carpenter SR et al. Understanding regional change: A comparison of two lake districts. *BioScience* 2007; **57**: 323–335.

Castro MJ, García-Rodríguez JA, González-Vida JM and Parés C. A parallel 2d finite volume scheme for solving systems of balance laws with nonconservative products: Application to shallow flows. *Comput. Methods Appl. Mech. Eng.* 2006; **195**: 2788-2815.

Castro MJ, García-Rodríguez JA, González-Vida JM and Parés C. Solving shallow-water systems in 2D domains using Finite Volume methods and multimedia SSE instructions *J. Comput. Appl. Math.* 2008; **221**: 16-32.

Casulli V, Cattani E. Stability, accuracy and efficiency of a semi-implicit method for three-dimensional shallow water flow. Computers and Mathematics with Applications 1994; 27(4):99–112.

Casulli V, Cheng RT. Semi-implicit finite difference methods for three-dimensional shallow water flow. *International Journal for Numerical Methods in Fluids* 1992; **15**: 629-648. DOI: 10.1002/fld.1650150602.

Casulli V, Walters RT. An unstructured grid, three-dimensional model based on the shallow water equations. *Int. J. Numer. Meth. Fluids* 2000; **32**: 331–348.

Chan T, Glowinski R, Periaux J, Widlund OB. eds. *Third Internat. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM) 1990. SIAM,* Philadelphia (EEUU).

Chan TF Mathew TP. Domain Decomposition Algorithms. *Cambritge University Press* 2008. DOI: 10.1017/S0962492900002427.

Chapman DC. Numerical Treatment of Cross-Shelf Open Boundaries in a Barotropic Coastal Ocean Model. *Journal of Physical Oceanography* 1985; **15**: 1060-1075. DOI: 10.1175/1520-0485(1985)015<1060:NTOCSO>2.0.CO;2.

Chapman RS, Johnson WH, Vemulakonda SR. User's guide for the sigma stretched version of CH3D-WES—A three-dimensional numerical hydrodynamic, salinity, and temperature model: U.S. Army Corps of Engineers, Waterways Experiment Station, Technical report HL-96-21 1996; 28 p. plus appendixes.

Chen CH Liu R. An Unstructured Grid, Finite-Volume, Three-Dimensional, Primitive Equations Ocean Model: Application to Coastal Ocean and Estuaries. *Journal of Atmospheric and Oceanic Technology* 2003; **20**: 159-186.

Cushman-Roisin B. Introduction to Geophysical Fluid Dynamics. Prentice-Hall, Inc. 1994. Englewood Cliffs, New Jersey (EEUU).

Daoud AH, Rakha KA, Abul-azm AG. A two-dimensional finite volume hydrodynamic model for coastal areas: Model development and validation. *Ocean Engineering* 2008; **35**: 150-164.

Davies AM. Application of the Dufort-Frankel and Saul'ev methods with time splitting to the formulation of a three dimensional hydrodynamic sea model: International. *Journal for Numerical Methods in Fluids* 1985; **5**: 405−425.

Davies HC. A lateral boundary formulation for multi-level prediction models. *Quarterly Journal of the Royal Meteorological Society* 1976; **102**: 405-418. DOI: 10.1002/qj.49710243210.

De Goede ED. A time-splitting method for the three-dimensional shallow water equations: *International Journal for Numerical Methods in Fluids* 1991; **13**: 519−534.

De Vicente IL, Cruz-Pizarro L, Rueda FJ. Sediment resuspension in two adjacent coastal shallow lakes: controlling factors and consequences on phosphate dynamics. *Aquatic Sciences* 2010. DOI: 10.1007/s00027-009-0107-1.

Debreu L, Blayo E. Two-way embedding algorithms: a review. *Ocean Dynamics* 2008; **58**: 415-428. DOI: 10.1007/s10236-008-0150-9.

Debreu L, Marchesiello P, Penven P, Cambon G. Two-way nesting in split-explicit ocean models: Algorithms, implementation and validation. *Ocean Modelling* 2012; **49–50**: 1-21. DOI: http://dx.doi.org/10.1016/j.ocemod.2012.03.003.

Debreu L, Vouland C, Blayo E. AGRIF: Adaptive grid refinement in Fortran. *Computers & Geosciences* 2008; **34**: 8-13. DOI: http://dx.doi.org/10.1016/j.cageo.2007.01.009.

Delis AI, Mathioudakis EN. A finite volume method parallelization for the simulation of free surface shallow water flows. *Mathematics and Computers in Simulation* 2009; **79**: 3339–3359.

Dinehart RL, Burau JR. Averaged indicators of secondary flow in repeated acoustic Doppler current profiler crossings of bends. *Water Resources Research* 2005; **41**: - W09405. DOI: 10.1029/2005WR004050.

Dongarra JJ, Duff IS, Sorensen DC, Van der Vorst HA. Numerical linear Algebra on High-Perdormance Computers (SIAM) 1998.

Dukowicz JK, Smith RD. Implicit free-surface method for the Bryan-Cox-Semtner ocean model. *Journal of Geophysical Research: Oceans* 1994; **99**: 7991-8014. DOI: 10.1029/93JC03455.

Dukowicz JK, Smith RD. Implicit free-surface method for the Bryan-Cox-Semtner ocean model: American Geophysical Union. *Journal of Geophysical Research (Oceans)* 1994; **99**: 7991−8014.

Dyk D, Geveler M, Mallach S, Ribbrock D, Göddeke D and Gutwenger C. HONEI: A collection of libraries for numerical computations targeting multiple processor architectures. *Comput. Phys. Commun.*2009; **180**: 2534-2543.

Ecer A, Akay HU, Kemle WB, Wang H, Ercoskun D. Parallel computation of fluid dynamics problems. *Comput. Methods Appl. Mech. Eng*. 1999; 174, 433.

Eijkhout V. Beware of unperturbed modified incomplete factorizations, in Iterative *Methods in Linear Algebra*, R. Beauwens and P. de Groen, eds. 1992; 583–591.

Enhua W, Liu Y, Liu X. An improved study of real-time fluid simulation on GPU: Research articles. *Comput.Animat.Virtual Worlds* 2004; **15**(3-4): 139-46.

EPA. User's Manual for Environmental Fluid Dynamics Code. Hydro version (EFDC-Hydro). Release 1.00. U.S. Environmental Protection Agency (EPA) 2002.

Ferrarin C, Umgiesser G, Cucco A, Hsu T-W, Roland A, Amos CL. Development and validation of a finite element morphological model for shallow wáter basins. *Coastal Enineering* 2008; **55**: 716-731.

Fischer PV, Patera AT. Parallel simulation of viscous incompressible flows. *Ann. Rev. Fluid Mech* 1994; 483-527

Fox AD, Maskell SJ. Two-Way Interactive Nesting of Primitive Equation Ocean Models with Topography. *Journal of Physical Oceanography* 1995; **25**: 2977-2996. DOI: 10.1175/1520-0485(1995)025<2977:TWINOP>2.0.CO;2.

Fringer OB, Gerritsen M, Street RL. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean. simulator. *Ocean Modelling* 2006; 14:139–173.

Galperin B, Kantha SH, Hassid S, Rosati A. A quasi-equilibrium turbulent energy model for geophysical flows. *J. Atmos. Sci.* 1988; **45**: 55-62

Gardner JV, Mayer LA, Hughes-Clarke J. Cruise report RV inland Surveyer Cruise IS-98; the bathymetry of Lake Tahoe, California-Nevada, August 2 through August 17, 1998, Lake Tahoe, California and Nevada. Open-File Report 98-509 1998. US Geological Survey, California (EEUU).

Giunta G, Mariani P, Montella R, Riccio A. pPOM: A nested, scalable, parallel and Fortran 90 implementation of the Princeton Ocean Model. *Environmental Modelling and Software* 2007; **22**: 117-122. DOI: http://dx.doi.org/10.1016/j.envsoft.2006.05.024.

Goforth RR, Carman SM. Multiscale relationships between Great Lakes nearshore fish communities and anthropogenic shoreline factors. *Journal of Great Lakes Research* 2009. **35**, 215–223.

Golub GH, Loan CF. Matrix Computations. The John Hopkins University Press 1996, Baltimore (EEUU).

Griffies SM, Boning C, Bryan FO, Chassignet EP, Gerdes R, Hasumi H, Hirst A, Treguier AM, Webb D. Developments in ocean climate modelling. *Ocean Modelling* 2000; **2**: 123-192. DOI: doi:10.1016/S1463-5003(00)00014-7".

Griffies SM, Harrison MJ, Pacanowski RC, Rosati A. A Technical guide to MOM4. GFDL Ocean Group Technical Report 2008 **; 5**.

Gropp WD et al. High-performance parallel implicit CFD. *Parallel Comput*.; 2001: **27**, 337–362.

Guobin S, Guang-Ping G, Shipeng L, Heung-Yeung S, Zhang Y. Accelerate video decoding with generic GPU. *Circuits and Systems for Video Technology, IEEE Transactions* 2005; **15**(5): 685-93.

Hackbusch W. Parallel algorithms for partial differential equations. *Notes on Numerical Fluid Mechanics* 1991; **31**.

Halliwell G,Jr, Barth A, Weisberg R, Hogan P, Smedstad O, Cummings J. Impact of GODAE products on nested HYCOM simulations of the West Florida Shelf. *Ocean Dynamics* 2009; **59**: 139-155. DOI: 10.1007/s10236-008-0173-2.

Hamrick JM. *A three-dimensional environmental fluid dynamics computer code : theoretical and computational aspects*. Virginia Institute of Marine Science, College of William and Mary: Gloucester Point, VA, 1992.

Harris GP. Phytoplankton Ecology: structure, function and fluctuation. Chapman & Hall 1986. London (UK).

Harris LM, Durran DR. An Idealized Comparison of One-Way and Two-Way Grid Nesting. *Monthly Weather Review* 2010; **138**: 2174-2187. DOI: 10.1175/2010MWR3080.1.

Heggelund Y, Berntsen J. A method for analysing nesting techniques for the linearized shallow water equations. *International Journal for Numerical Methods in Fluids* 2002; **38**: 163-185. DOI: 10.1002/fld.215.

Herzfeld M, Andrewartha JR. A simple, stable and accurate Dirichlet open boundary condition for ocean model downscaling. *Ocean Modelling* 2012; **43–44**: 1-21. DOI: http://dx.doi.org/10.1016/j.ocemod.2011.11.010.

Herzfeld M. An alternative coordinate system for solving finite difference ocean models. *Ocean Modelling* 2006; **14**: 174-196. DOI: http://dx.doi.org/10.1016/j.ocemod.2006.04.002.

Herzfeld M. Improving stability of regional numerical ocean models. *Ocean Dynamics* 2009; **59**: 21-46. DOI: 10.1007/s10236-008-0158-1.

Herzfeld M. The role of numerical implementation on open boundary behaviour in limited area ocean models. *Ocean Modelling* 2009; **27**: 18-32. DOI: http://dx.doi.org/10.1016/j.ocemod.2008.10.008.

Hestenes MR, Stiefel EL. Method of Conjugate Gradient for Solving Linear Systems. *J. Res. Nat. Bureau Stand* 1952; **49**:459.

Hill MC. Solving groundwater flow problems by conjugate-gradient methods and the strongly implicit procedure: American Geophysical Union. *Water Resources Research* 1990; **26**: 1961−1969.

Hinterberger C, Fröhlich J, Rodi W. Three-Dimensional and Depth-Averaged Large-Eddy Simulations of Some Shallow Water Flows. *Journal of Hydraulic Engineering-asce* 2007; **133**: 857-872. DOI: 10.1061/(ASCE)0733-9429(2007)133:8(857).

Hodges BR, Imberger J, Saggio A, Winters KB. Modeling basin-scale internal waves in a stratified lake. *Limnol. Oceanogr* 2000; **45**(7): 1603-1620.

Hodges BR, Rueda FJ. Linear Accuracy and Stability of Semi-Implicit Two-Level Predictor-Corrector Methods for Hydrostatic Barotropic/Baroclinic Flows. *International Journal of Computational Fluid Dynamics* 2008; **22**(9): 593–607.

Hodges BR. Hydrodynamical Modeling. *Encyclopedia of Inland Waters* 2009; 1: 613-627, Oxford: Elsevier.

Hoffman KH, Zou J. Parallel Efficiency of domain decomposition methods. *Parallel Computing* 1993; **19**: 1375-1391.

Hoyer AB, Schladow SG, Rueda FJ. A hydrodynamic-based approach to evaluating the risk of waterborne pathogens entering drinking water intakes in a large stratified lake. In revision 2015.

Hoyer AB, Wittmann ME, Chandra S, Schladow SG, Rueda FJ. A 3D individual-based aquatic transport model for the assessment of the potential dispersal of planktonic larvae of an invasive bivalve. *Journal of Environmental Management* 2014; **145**: 330-340.

Hu Y, Huang X, Wang X, Fu H, Xu S, Ruan H, Xue W, Yang G. A Scalable Barotropic Mode Solver for the Parallel Ocean Program. In: Wolf F, Mohr B, an Mey D (Eds.). Springer 2013 **8097**: 739-750. Berlin (Germany) http://dx.doi.org/10.1007/978-3-642-40047-6_74.

Huang A, Rao YR, Lu Y, Zhao J. Hydrodynamic modeling of Lake Ontario: An intercomparison of three models. *Journal of Geophysical Research: Oceans* 2010; **115**: - C12076. DOI: 10.1029/2010JC006269.

Huang JCK, Sloss PW. Simulation and verification of Lake Ontario’s mean state: American Meteorological Society. *Journal of Physical Oceanography* 1981; **11**: 1548−1566.

Husbands P, Yelick K. Multi-Threading and One-Sided Communication in Parallel LU Factorization. *Association for Computing Machinery* 2007; 10-16. DOI: 978-1-59593-764-3/07/001.

Imberger J, Parker G. The Diurnal Mixed Layer. *Limnol. Oeanog* 1985; **30**: 737-770.

Imberger J. Flux paths in a stratified lake: A review, in Physical Processes in Lakes and Oceans, Coastal Estuarine Stud. Edited by J. Imberger 1998; 54: 1–17. AGU, Washington (EEUU).

Ino F, Kotani Y, Hagihara K. Harnessing the power of idle GPUs for acceleration of biological sequence alignment. IEEE international symposium on Parallel & distributed processing (IPDPS) 2009.

Jensen TG. Open boundary conditions in stratified ocean models. *Journal of Marine Systems* 1998; **16**: 297-322.

Jiaquan G, Li B, He G. Modified Incomplete Cholesky Preconditioned Conjugate Gradient Algorithm on GPU for the 3D Parabolic Equation. C.-H. Hsu et al. (Eds.): NPC. 2013; LNCS **8147**: 298–307.

Jin X-Y. Quasi-three-dimensional numerical modeling of flow and dispersion in shallow water: Delft University of Technology, Department of Civil Engineering, Communication on Hydraulic and Geotechnical Engineering, Report 93-3 1993; 174 p.

Jin, KR, Hamrick JH Tisdale T. Application of Three-Dimensional Hydrodynamic Model for Lake Okeechobee. *J. Hydraul. Eng.* 2000; **126**(10): 758-771.

Jordi A, Wang D. sbPOM: A parallel implementation of Princenton Ocean Model. *Environmental Modelling and Software* 2012; **38**: 59-61. DOI: http://dx.doi.org/10.1016/j.envsoft.2012.05.013.

Kalff J. Limnology: Inland Water Ecosystems. Prentice-Hall 2001.

Kantha LH, Claysin CA. An improved mixed layer model gor geophysical applications. *Journal og Geophysical Research* 1994; **99**: 235-266

Karypis G, Schloegel K, Kumar V. PARMETIS project URL 2000. Available from: <http:// www-users.cs.umn.edu/~karypis/metis/main.shtml>.

Keyes DE, Gropp WD. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *J. Sci. Sta. Comput.* 1987; **8**: 166-201.

Killworth PD, Stainforthd D, Webb DJ, Paterson SM. The development of a free-surface Bryan-Cox-Semtner ocean model: American Meteorological Society. *Journal of Physical Oceanography* 1991; **21**: 1333−1348.

Kim SY, Im YT. Parallel Processing of 3D rigid-viscoplastic finite element analysis using domain decomposition and modified block Jacobi preconditioning technique. *Journal of Material Processing Technology* 2003; 134: 254-264.

Kincaid DR, Oppe TC Joubert WD. An introduction to the NSPCG software package. International Journal of Numerical Methods in Engineering 1989; **27**(3): 589-608.

Koch SE, DesJardins M. Kocin P. An interactive Barnes Objective Map Analysis Scheme for Use with Satelite and Conventional Data, *Journal of Climate and Applied Meteorology* 1983.

Kolerski T, Shen HT, Knack IM. A Nested Model for River Ice Dynamics. Proceedings, 20th IAHR Ice Symposium 2010 Lahti, Finland.

Kourafalou V, Peng G, Kang H, Hogan P, Smedstad O, Weisberg R. Evaluation of Global Ocean Data Assimilation Experiment products on South Florida nested simulations with the Hybrid Coordinate Ocean Model. *Ocean Dynamics* 2009; **59**: 47-66. DOI: 10.1007/s10236-008-0160-7.

Kowalik Z, Murty TS. Numerical modelling of ocean dynamics. BULK Series on Ocean Engineering. WORLD SCIENTIFIC 1993. http://dx.doi.org/10.1142/9789812795991_0001.

Laval B, Imberger J, Hodges BR, Stocker R. Modeling circulation in lakes: spatial and temporal variations. *Limnology and Oceanography* 2003; 48(3):983–994.

Lavelle JW, Thacker WC. A pretty good sponge: Dealing with open boundaries in limited-area ocean models. *Ocean Modelling* 2008; **20**: 270-292. DOI: http://dx.doi.org/10.1016/j.ocemod.2007.10.002.

Lawrence G, Ashley KI, Yonemitsu N, Ellis JR. Natural dispersion in a small lake 1995; **40**: 1519-1526.

Lee M, Chun CH, Hong S. Financial derivatives modeling using GPU's. Scalable computing and communications; eighth international conference on embedded computing 2009.

Leendertse JJ, Liu SK. A three-dimensional model for estuaries and coastal seas—Volume II, aspects of computation: Rand Corporation, Santa Monica Calif., Report R-1764-OWRT 1975; 123 p.

Leendertse JJ, Liu SK. A three-dimensional model for estuaries and coastal seas—Volume IV, turbulent energy computation: Rand Corporation, Santa Monica, Calif., Report R-2187-OWRT 1977; 59 p.

Leendertse JJ. Aspects of SIMSYS2D—A system for two-dimensional flow computation: Rand Corporation, Santa Monica, Calif., Report R-3572-USGS. Prepared for the U.S. Geological Survey 1987; 80 p.

Leendertse, JJ, Alexander RC, Liu SK. A three-dimensional model for estuaries and coastal seas—Volume I, principles of computation: Rand Corporation, Santa Monica, Calif., Report R-1417-OWRR 1973; 57 p.

Leendertse, JJ. A new approach to three-dimensional free-surface flow modeling: Rand Corporation, Santa Monica, Calif., Memorandum R-3712-NETH/RC, Prepared for The Netherlands Rijkswaterstaat 1989; 51 p.

Leon LF, Smith REH, Malkin SY, Depew D, Hipsey MR, Antenucci JP, Higgins SN, Hecky RE, Rao RY. Nested 3D modeling of the spatial dynamics of nutrients and phytoplankton in a Lake Ontario nearshore zone. *Journal of Great Lakes Research* 2012; **38**, Supplement 4: 171-183. DOI: http://dx.doi.org/10.1016/j.jglr.2012.02.006.

Liggett JA. Unsteady circulation in shallow, homogeneous lakes: Proceedings of the American Society of Civil Engineers, *Journal of the Hydraulics Division* 1969; **95**: 1273−1288.

Lodge DM, Barko JW, Strayer D et al. Spatial heterogeneity and habitat interactions in lake communities. – In: Carpenter, SR (ed.), Complex interactions in lake communities. Springer-Verlag 1988; 181–208.

Luecke GR, Wei-Hua L. Scalability and performance of OpenMP and MPI on a 128-processor SGI Origin 2000, Concurrency and Computation: Practice and Experience 2001; **13**: 905–928.

Magoulu M, Van der Vorst HA. ParIC: A Family of Parallel Incomplete Cholesky Preconditioners. *High Performance Computing and Networking Lecture Notes in Computer Science* 2000; **1823**: 89-98.

Mamadou HN, Nanri T, Murakami K. Collective Communication Cost Analysis over Gigabit Ethernet and Infiniband. *High Performance Computing (HiPC)* 2006; 547-559.

Manzini G, Stolcis L. Distributed parallel strategies for industrial CFD solvers: a case study and analysis of performances. *J. Parallel Distrib. Comput*. 1999; **57**: 334.

Marchesiello P, McWilliams JC, Shchepetkin AF. Open boundary conditions for long-term integration of regional oceanic models. *Ocean Modelling* 2001; **3**: 1-20. DOI: http://dx.doi.org/10.1016/S1463-5003(00)00013-5.

Martin PJ. Simulation of the mixed layer at OWS November and Papa with several models. *J. Geophysical Research* 1985; **90**: 903-916

Martinsen EA, Engedahl H. Implementation and testing of a lateral boundary scheme as an open boundary condition in a barotropic ocean model. *Coastal Engineering* 1987; **11**: 603-627. DOI: http://dx.doi.org/10.1016/0378-3839(87)90028-7.

Mason E, Molemaker J, Shchepetkin AF, Colas F, McWilliams JC, Sangrà P. Procedures for offline grid nesting in regional ocean models. *Ocean Modelling* 2010; **35**: 1-15. DOI: http://dx.doi.org/10.1016/j.ocemod.2010.05.007.

Message Passing Interface Forum, MPI Documents.

Messina P, Culler D, Pfeiffer W, Martin W, Oden JT and Smith G. Architecture," *Commun ACM* 1998; **41**: 36-44.

Meyer PD, Valocchi AJ, Ashby SF, Saylor PE. A numerical investigation of the conjugate gradient method as applied to three-dimensional groundwater flow problems in randomly heterogeneous porous media: American Geophysical Union. Water Resources Research 1989; **25**: 1440−1446.

Michalakes J. RSL: A Parallel Runtime System Library for Regional Atmospheric Models with Nesting. In: Baden S, Chrisochoides N, Gannon D, Norman M (Eds.). Springer 2000 **117**: 59-74. New York(EEUU) http://dx.doi.org/10.1007/978-1-4612-1252-2_4.

Myungho L, Chun CH, Hong S. Financial derivatives modeling using GPU's. Scalable computing and communications; eighth international conference on embedded computing (SCALCOM-EMBEDDEDCOM) 2009.

Naff RL, Wilson JD. A comparison of preconditioning techniques for parallelized PCG solvers for the cell-centered finite-difference problem. *XVI International Conference on Computational Methods in Water Resources* 2006; 18-22.

Naik NH. Parallelization of a class of implicit finite difference schemes in computational fluid dynamics. International Journal of High Speed Computing 1993; **5**: 1-50.

Nesterov O. A simple parallelization technique with MPI for ocean circulation models. *Journal of Parallel and Distributed Computing* 2010; **70**: 35-44. DOI: http://dx.doi.org/10.1016/j.jpdc.2009.09.005.

Nishimoto MM, Washburn L. Patterns of coastal eddy circulation and abundance of pelagic juvenile fish in the Santa Barbara Channel, California, USA. *Marine Ecology Progress Series* 2002; **241**: 183-199.

Norden M, Holmgren S, Thune M. OpenMP versus MPI for PDE solvers based on regular sparse numerical operators. *Future Generation Computer System* 2006; **22**: 194-203.

Nycander J, Döös K. Open boundary conditions for barotropic waves. *Journal of Geophysical Research: Oceans* 2003; **108**: - 3168. DOI: 10.1029/2002JC001529.

Oddo P, Pinardi N. Lateral open boundary conditions for nested limited area models: A scale selective approach. *Ocean Modelling* 2008; **20**: 134-156. DOI: http://dx.doi.org/10.1016/j.ocemod.2007.08.001.

O'Donncha F, Ragnoli E, Suits F. Parallelisation study of a three-dimensional environmental flow model. *Computers & Geosciences* 2014; **64**: 96-103. DOI: http://dx.doi.org/10.1016/j.cageo.2013.12.006.

OpenMP Architecture Review Board, OpenMP Specifications.

Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC. GPU computing. Proceedings of the IEEE 2008; **96**(5): 879-99.

Paglieri L, Ambrosi D, Formaggia L, Quarteroni A and Scheinine AL. Parallel computation for shallow water flow: A domain decomposition approach. *Parallel Computing* 1997; **23**: 1261-1277.

Pairaud IL, Gatti J, Bensoussan N, Verney R Garreau P. Hydrology and circulation in a coastal area off Marseille: Validation of a nested 3D model with observations. *Journal of Marine System* 2011; **88**: 20-33.

Palma ED, Matano RP. Dynamical impacts associated with radiation boundary conditions. *Journal of Sea Research* 2001; **46**: 117-132. DOI: http://dx.doi.org/10.1016/S1385-1101(01)00076-4.

Palma ED, Matano RP. On the implementation of passive open boundary conditions for a general circulation model: The barotropic mode. *Journal of Geophysical Research: Oceans* 1998; **103**: 1319-1341. DOI: 10.1029/97JC02721.

Panetta J, Teixeira T, Paulo Filho RP, Finho C, Sotelo D,. Roxo da Motta FM, Pinheiro SS. Accelerating kirchhoff migration by CPU and GPU cooperation. International symposium on Computer architecture and high performance computing (SBAC-PAD) 2009.

Paris CB, Helgers J, van Sebille E, Srinivasan A. Connectivity Modeling System: A probabilistic modeling tool for the multi-scale tracking of biotic and abiotic variability in the ocean. *Environmental Modelling and Software* 2013; **42**: 47-54. DOI: http://dx.doi.org/10.1016/j.envsoft.2012.12.006.

Passoni GP, Cremonesi B, Giancarlo A. Analysis and implementation of a parallelization strategy on a Navier-Stokes solver for shear flow simulations. *Parallel Computing* 2001; **27**: 1665-1685.

Pellissetti MF, Ghanem RG. Iterative solution of systems of linear equations arising in the context of stochastic finite elements. *Advances in Engineering Software* 2000; **31**: 607–616.

Penven P, Debreu L, Marchesiello P, McWilliams JC. Evaluation and application of the ROMS 1-way embedding procedure to the central california upwelling system. *Ocean Modelling* 2006; **12**: 157-187. DOI: http://dx.doi.org/10.1016/j.ocemod.2005.05.002.

Perry RW, Skalski JR, Brandes PL, Sandstrom PT, Klimley AP, Ammann A, MacFarlane B. Estimating Survival and Migration Route Probabilities of Juvenile Chinook Salmon in the Sacramento–San Joaquin River Delta. *North American Journal of Fisheries Management* 2010; **30**: 142-156. DOI: 10.1577/M08-200.1.

Pjesivac-Grbovic J, Angskun T, Bosilca G, Fagg G, Gabriel E, Dongarra J. Performance analysis of MPI collective operations. *Cluster Computing* 2007; **10**: 127-143. DOI: 10.1007/s10586-007-0012-0.

Ramón CL, Armengol J, Dolz J, Prats J, Rueda FJ. Mixing dynamics at the confluence of two large rivers undergoing weak density variations. *Journal of Geophysical Research – Oceans* 2014; **119**. DOI:10.1002/2013JC009488.

Rao P. A parallel hydrodynamic model for shallow water equations. *Applied Mathematics and Computation* 2004; 150: 291-302.

Rao YR, Schwab DJ. Transport and Mixing Between the Coastal and Offshore Waters in the Great Lakes: a Review. *J. Great Lakes Res* 2007; **33**: 202–218.

Resch M, Bjorn S, Isabel L. A comparison of OpenMP and MPI for the parallel CFD test case. *Proceedings of the First European Workshop on OpenMP* 1999.

Rueda FJ, Cowen EA. The residence time of a freshwater embayment connected to a large lake. Limnol. Oceanogr. 2005; **50**: 1638-1653.

Rueda FJ, MacIntyre S. Modelling the fate and transport of storm-river-water in small multi-basin lakes. *Environmental Modeling and Software* 2009; **25**: 146–157

Rueda FJ, MacIntyre S. Flow and spatial heterogeneity of stream inflows in a small multibasin lake. *Limnol. Oceanogr*. 2009; **54**(6): 2041–2057.

Rueda FJ, Sanmiguel-Rojas E, Hodges BR. Baroclinic stability of two-level semi-implicit numerical methods for the 3D shallow water equations. *International Journal for Numerical Methods in Fluids* 2006. DOI: 10.1002/fld.1391.

Rueda FJ, Schladow SG, Palmarson SO. Basin-scale internal wave dynamics during a winter cooling period in a large lake. *Journal Geophysical Research (Oceans)* 2003; **108**(C3). DOI 10.1029/2001JC000942

Rueda FJ, Schladow SG. Mixing and stratification in lakes of varying horizontal length scales: Scaling arguments and energy partitioning. *Limnology and Oceanography Limnology and Oceanography* 2009; **54**(6): 2003–2017.

Rueda FJ, Schladow SG. The internal dynamics of a shallow polymictic lake. Part II: numerical simulations. ASCE. *Journal of Hydraulic Engineering* 2003; **129**(2): 92-101.

Rueda FJ, Vidal J, Schladow SG. Modeling the effect of size reduction on the stratification of a large wind-driven lake using an uncertainty based approach. Water Resour. Res. 2009; **45**:W03411. DOI:10.1029/2008WR006988

Rueda FJ, Vidal J. Currents in the Upper Mixed Layer and in Unstratified Water Bodies. In: Likens GE (Ed.). *Encyclopedia of Inland Waters*. Academic Press: Oxford, 2009: 568-582.

Rueda FJ. A three-dimensional hydrodynamic and transport model for lake environments. PhD thesis 2001, University of California, Davis (EEUU).

Schindler DE Scheuerell MD. Habitat coupling in lake ecosystems. Oikos 2002; **98**: 177–189.

Schwab DJ, Bedford KW. Initial implementation of the Great Lakes Forecasting System: A real-time system for predicting lake circulation and thermal structure, *Water Pollut. Res. J. Can* 1994; **29**: 203-220.

Schwab DJ, Beletsky D, DePinto J, Dolan DM. A hydrodynamic approach to modeling phosphorus distribution in Lake Erie. *Journal of Great Lakes Research* 2009; **35**: 50-60. DOI: http://dx.doi.org/10.1016/j.jglr.2008.09.003.

Semtner AJ, Chervin RM. A simulation of the global ocean circulation with resolved eddies, J. *Geophys. Res.*1988; 15502-15222.

Semtner AJ. Modeling ocean circulation: Science 1995; **269**: 1379−1385.

Shchepetkin AF, McWilliams JC. The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. Ocean Modelling 2005; 9: 347-404. DOI: http://dx.doi.org/10.1016/j.ocemod.2004.08.002.

Sheng J, Wright DG and Greatbatch RJ. CANDIE: a new version of the DieCAST ocean circulation Model. *J Atm Oceanic Tech* 1998; **15**: 1414–1432.

Sheng YP, Lick, Wilbert, Gedney RT, Molls FB. Numerical computation of three-dimensional circulation in Lake Erie—A comparison of a free-surface model and a rigid-lid model: American Meteorological Society. *Journal of Physical Oceanography* 1978; **8**: 713−727.

Sheng YP. Finite-difference models for hydrodynamics of lakes and shallow seas, chap. 6 of Gray, W.G., ed., Physics-based modeling of lakes, reservoirs, and impoundments: American Society of Civil Engineers 1986; 146−228.

Sheng YP. Mathematical modeling of three-dimensional coastal currents and sediment dispersion—model development and applications: U.S. Army Engineer Waterways Experiment Station, Technical report no. CERC-83-2 1983; 288 p.

Silva-Moura L, Buyya R. Parallel Programming Models and Paradigms. In: Buyya R (Ed.). High Performance Cluster Computing 2. Prentice Hall 1998: 4-27. http://www.buyya.com/cluster/v2chap1.pdf.

Smith PE. A semi-implicit, three-dimensional model of estuarine circulation. United States Geological Survey. Open-File Report 2006-1004: Sacramento, California, 2006. http://pubs.usgs.gov/of/2006/1004/pdf/ofr2006-1004.pdf.

Smith R, Jones P, Briegleb B, Bryan F, Danabasoglu G, Dennis J, Dukowicz J, Eden C, Fox-Kemper B, Gent P, Hecht H, Jayne S, Jochum M, Large W, Lindsay K, Maltrud M, Norton N, Peacock S, Vertenstein M, Yeager S. The parallel Ocean Program (POP) reference manual. Ocean Component of the Community Climate System Model (CCSM) and Community Earth System Model (CESM) 2010.

Smith RD, Dukowicz JK, Malone RC. Parallel ocean general circulation modelling, *Physica D* 1992.

Song S, Lynett PJ Kim D. Nested and multi-physics modeling of tsunami evolution from generation to inundation. *Ocean Modelling* 2011; **38**: 96-113.

Stelling GS. On the construction of computational methods for shallow water flow problems: Rijkswaterstaat Communication, The Hague, Netherlands 1984; **35**: 226 p.

Stone SH. An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations. *Journal of the ACM (JACM)* 1973; **20**: 27-38

Tai CH, Zhao Y, Liew KM. Parallel-multigrid computation of unsteady incompressible viscous flows using matrix-free implicit method and high-resolution characteristics-based scheme. *Comput. Methods Appl. Mech. Engrg.* 2005; **194**: 3949–3983

Taylor AH. Identifying forest reference conditions on early cut-over lands, Lake Tahoe basin, USA. *Ecological Applications* 2004; **14**: 1903-1920. DOI: 10.1890/02-5257.

Teranishi K, Raghavan P. A hybrid parallel preconditioner using incomplete Cholesky factorization and sparse approximate inversion. Domain Decomposition Methods in Science and Engineering XVI, *Lecture Notes in Computational Science and Engineering* 2007; **55**: 755–762.

Thomas LH. Eliptic Problems in Linear Difference Equations over a Network. Wastom Sci. Comput. Lab. Rept, Columbia University, New York (EEUU) 1949.

Tölke J, Krafczyk M. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *International Journal of Computational Fluid Dynamics* 2008; **22**(7): 443.

Tubbs KR, Tsai FT. Multilayer shallow water flow using lattice Blotzman method with high performance computing. *Adv. Water Resour* 2009; **32**: 1767-1776.

Uittenbogaard RE, van Kester JA, Stelling GS. Implementation of three turbulence models in TRISULA for rectangular horizontal grids: Delft Hydraulics, Delft, Netherlands, Prepared for the Tidal Waters Division of the Dutch Ministry of Transport, Public Works and Water Management 1992; 169 p.

Vadeboncoeur Y, McIntyre PB, Vander MJ. Borders of Biodiversity: Life at the Edge of the World's Large Lakes. *BioScience* 2011; **61**: 526–537.

Wang P, Song YT, Chao Y, Zhang H. Parallel Computation of the Regional Ocean Modeling System. *International Journal of High Performance Computing Applications* 2005; **19**: 375-385. DOI: http://dx.doi.org/10.1177/1094342005059115.

Weller H, Lock S, Wood N. Runge–Kutta IMEX schemes for the Horizontally Explicit/Vertically Implicit (HEVI) solution of wave equations. *Journal of Computational Physics* 2013; **252**: 365-381. DOI: http://dx.doi.org/10.1016/j.jcp.2013.06.025.

Westerink JJ, Luetich RA, Feyen JC, Atkinson JH, Dawson C, Roberts HJ, Powell MD et al. A Basin- to Channel-Scale Unstructured Grid Hurricane Storm Surge Model Applied to Southern Loisiana. *Mon. Wea. Rev.* 2008; **136**: 833–864. DOI: http://dx.doi.org/10.1175/2007MWR1946.1

Wilders P, van Stijn TL, Stelling GS, Fokkema GA. A fully implicit splitting method for accurate tidal computations. *International Journal for Numerical Methods in Engineering* 1988; **26**: 2707−2721.

Yakubov S, Cankurt B, Abdel-Maksoud M, Rung T. Hybrid MPI/OpenMP parallelization of an Euler–Lagrange approach to cavitation modelling. *Computers & Fluids* 2012; **45**: 185-191.

Yu D. Parallelization of a two-dimensional flood inundation model based on domain decomposition. *Environmental Modelling and Software* 2010; **25**: 935-945

Zamani K, Bombardelli F, Wuertz S, Smith P. Toward a 3-Dimensional Numerical Modeling of Tidal Currents in San Francisco Bay. *American Society of Civil Engineers* 2010; 1385-1394. http://dx.doi.org/10.1061/41114(371)147.

Zao J, Rao YR, Wassenaar LI. Erratum to"Numerical modeling of hydrodynamics and tracer dispersion during ice-free period in Lake Winnipeg". *Journal of Great Lakes Research* 2012; **38**: 584.

Zavatarelli M, Pinardi N. The Adriatic Sea modelling system: a nested approach. *Annales Geophysicae* 2003; **21**: 345-364. DOI: 10.5194/angeo-21-345-2003.

Zhai L, Sheng J, Greatbatch RJ. Application of a nested-grid ocean circulation model to Lunenburg Bay of Nova Scotia: Verification against observations. *Journal of Geophysical Research: Oceans* 2008; **113**: - C02024. DOI: 10.1029/2007JC004230.

Zhang B, Oosterlee CW. *Option pricing with COS method on graphics processing units*. IEEE international symposium on Parallel & distributed processing (IDPS) 2009.

Zhang W, Hong H, Shang S, Chen D, Chai F. A two-way nested coupled tide-surge model for the Taiwan Strait. *Continental Shelf Research* 2007; **27**: 1548-1567. DOI: 10.1016/j.csr.2007.01.018.