UNIVERSIDAD DE GRANADA

MÁSTER UNIVERSITARIO EN GESTIÓN Y TECNOLOGÍAS DE PROCESOS DE NEGOCIO

Curso académico 2024-2025

TRABAJO FIN DE MÁSTER

Diseño e Implementación de un Clúster HPC con Automatización del Despliegue de Tareas de Cómputo Intensivo

Autor:
Sergio Ivan AQUINO BRITEZ

Tutor/es:
Dr. Gabriel José GUERRERO CONTRERAS

DETECCIÓN DEL PLAGIO

La Universidad utiliza el programa **Turnitin Feedback Studio** para comparar la originalidad del trabajo entregado por cada estudiante con millones de recursos electrónicos y detecta aquellas partes del texto copiadas y pegadas. Copiar o plagiar en un TFM es considerado una **Falta Grave**, y puede conllevar la expulsión definitiva de la Universidad.

Diseño e Implementación de un Clúster HPC con Automatización del Despliegue de Tareas de Cómputo Intensivo

Sergio Ivan AQUINO BRITEZ

Palabras clave: Automatization, Cluster, High Performance Computing, Virtualization

Resumen

En campos como la investigación científica, la docencia y diversas aplicaciones profesionales, se requieren capacidades computacionales superiores a las de estaciones de trabajo convencionales. La Computación de Altas Prestaciones (HPC, *High Performance Computing*) permite ejecutar cargas intensivas, como simulaciones numéricas, análisis de grandes volúmenes de datos y entrenamiento de modelos de inteligencia artificial. Sin embargo, su adopción presenta barreras técnicas y económicas en contextos con recursos limitados, debido a la complejidad de configuración, los requerimientos de *hardware* y la ausencia de mecanismos de automatización accesibles.

Este Trabajo Fin de Máster (TFM) presenta el diseño e implementación de un clúster HPC local virtualizado y basado en *software* libre. La propuesta está orientada a entornos con recursos limitados y busca reducir la complejidad operativa mediante una herramienta multiplataforma que facilita el uso remoto del sistema desde estaciones cliente.

La arquitectura implementada se basa en el modelo Beowulf y está compuesta por un nodo principal y tres nodos de cómputo, todos virtualizados sobre un único host físico. El sistema operativo es Rocky Linux 8.10. La infraestructura incluye Slurm para la gestión de recursos, FreeIPA para el control de identidades, NFS para el almacenamiento compartido, EasyBuild para el despliegue de aplicaciones, Ganglia para la monitorización y Open OnDemand como interfaz de acceso remoto.

La herramienta desarrollada permite automatizar tareas clave como la sincronización de archivos, la preparación de entornos y el envío y seguimiento de cargas al planificador. Esta funcionalidad busca reducir la complejidad de interacción con el clúster, proporcionando una interfaz simple y eficiente para usuarios sin experiencia previa en HPC.

Finalmente, mediante las pruebas realizadas en un entorno virtual, se comprobó que la implementación tanto de la infraestructura del clúster como de la herramienta funcionan de manera eficiente. Además, estas configuraciones se destacan por ser totalmente replicables en una infraestructura física, lo que refuerza su valor como modelo propuesto en este TFM.

Design and Implementation of an HPC Cluster with Automated Algorithm Deployment

Sergio Ivan AQUINO BRITEZ

Keywords: Automatization, Cluster, High Performance Computing, Virtualization

Abstract

In fields such as scientific research, education, and various professional applications, computational capabilities beyond those of conventional workstations are often required. High Performance Computing (HPC) enables the execution of intensive workloads, such as numerical simulations, large-scale data analysis, and the training of artificial intelligence models. However, the adoption of such infrastructures poses technical and economic challenges in resource-constrained contexts, due to the complexity of configuration, hardware requirements, and the lack of accessible automation mechanisms.

This Master's Thesis (TFM) presents the design and implementation of a local, virtualized HPC cluster based on free and open-source software. The proposal is aimed at environments with limited resources and seeks to reduce operational complexity through a cross-platform tool that facilitates remote use of the system from client workstations.

The implemented architecture follows the Beowulf model and consists of one head node and three compute nodes, all virtualized on a single physical host. The operating system is Rocky Linux 8.10. The infrastructure includes Slurm for resource management, FreeIPA for identity control, NFS for shared storage, EasyBuild for application deployment, Ganglia for monitoring, and Open OnDemand as a remote access interface.

The developed tool automates key tasks such as file synchronization, environment preparation, and job submission and monitoring to the scheduler. This functionality is intended to simplify interaction with the cluster, providing a user-friendly and efficient interface for users without prior HPC experience.

Finally, tests conducted in a virtual environment confirmed that the implementation of both the cluster infrastructure and the tool operates efficiently. Moreover, these configurations have proven to be fully replicable in a physical infrastructure, reinforcing their value as a model proposed in this TFM.

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento a Dios por concederme la salud y la oportunidad de avanzar en mi preparación, así como la fuerza y la determinación necesarias para afrontar los desafíos que surgieron a lo largo del máster.

Dedico este trabajo a mi amada esposa Laura, por el apoyo incondicional y el aliento constante que me brinda cada día para seguir alcanzando nuevas metas, tanto en lo profesional como en lo personal. A mis adoradas hijas, Lucía, Verónica y Julieta, quienes son y seguirán siendo mi mayor fuente de inspiración.

A mis padres Sergio y Herminia, por haberme apoyado en todo momento, por sus consejos, sus oraciones, sus valores, por los ejemplos de perseverancia y constancia que los caracterizan y que nos han inculcado siempre.

A mi tía Nélida, por su cariño, sus palabras de aliento y por estar siempre presente con su apoyo incondicional, brindándome fuerzas en los momentos más desafiantes.

A mis hermanos, Liz y Diego, en especial a Diego, por estar a mi lado en los momentos claves, por su ayuda constante y por impulsarme a buscar siempre mi superación personal. Gracias por recordarme que las adversidades no son obstáculos, sino oportunidades valiosas para crecer y mejorar.

A mis compañeros del Máster, con quienes compartí no solo horas de estudio y trabajo, sino también experiencias, desafíos y aprendizajes que enriquecieron profundamente esta etapa académica.

A los profesores de la Universidad de Granada, principalmente a los del Máster en Gestión y Tecnologías de Procesos de Negocio, que me apoyaron de manera incondicional desde el inicio hasta la culminación de este máster.

Por último, deseo expresar un agradecimiento especial a mi tutor, el Dr. Gabriel José Guerrero Contreras, por haberme dedicado parte de su valioso tiempo, por sus ideas y aportes excepcionales a lo largo de todo el desarrollo de este proyecto, por su constante disposición para resolver mis dudas y por brindarme su respaldo y apoyo en cada etapa.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN

La revolución de las Tecnologías de la Información y la Comunicación (TIC) de las últimas décadas ha transformado profundamente los procesos sociales, económicos y científicos. Este cambio ha dado lugar a profundos cambios en la generación, difusión y aplicación del conocimiento, permitiendo una mayor conectividad global, el acceso instantáneo a información y la automatización de una gran cantidad de procesos (Smuts & Van der Merwe, 2022). Esta situación ha provocado un cambio trascendental, permitiéndonos comunicarnos en cualquier instante y desde cualquier lugar del mundo, lo que ha provocado la generación y difusión de enormes volúmenes de datos. Por esta razón, nuestra sociedad pasa a denominarse la sociedad de la información (Lucendo-Monedero et al., 2023).

En este contexto, la utilización de los datos obtenidos puede resultar beneficiosa para generar información y de esta forma convertirse en conocimiento útil para la toma de decisiones, así como también, para el desarrollo de sistemas inteligentes, que permitan llevar a cabo operaciones de análisis de los datos o entrenar algoritmos de Inteligencia Artificial (IA, *Artificial Intelligence*) (Calzati, 2025), como los modelos de lenguaje de gran escala (LLM, *Large Language Model*), los cuales requieren grandes cantidades de texto para realizar su entrenamiento. Por ejemplo, el modelo ChatGPT, desarrollado por OpenAI, está basado en el modelo transformador generativo preentrenado 3 (GPT-3, *Generative Pretrained Transformer 3*), que fue entrenado con un conjunto de datos de aproximadamente 570 GB de texto preprocesado y filtrado de diversas fuentes (Brown et al., 2020).

Resulta importante indicar que, cuando se manejan grandes volúmenes de datos, la magnitud del procesamiento requerido suele superar la capacidad de un único nodo. Esto es debido a que un solo servidor está limitado en recursos de cómputo como memoria, procesamiento y almacenamiento, lo que finalmente restringe la capacidad de cómputo en tiempos razonables. Por ello, se recurre a la utilización de clústeres de servidores, donde múltiples nodos trabajan de manera colaborativa. Un ejemplo representativo se muestra en la Figura 1.1. De esta forma, es posible reducir significativamente los tiempos de procesamiento y permite escalar mediante la incorporación de nuevos nodos, según la necesidad de cómputo (Li et al., 2023). Este tipo de arquitectura se denomina Computación de Altas Prestaciones (HPC, *High Performance Computing*) y esta utiliza técnicas de procesamiento paralelo para ejecutar aplicaciones avanzadas de manera rápida y eficiente (T. Sterling et al., 2017).

Dado que los recursos del clúster son compartidos por múltiples usuarios o procesos, es importante contar con un mecanismo que permita organizar, gestionar y planificar la cola de trabajo estableciendo prioridades, esto permite asegurar el uso

eficiente y equitativo de los recursos disponibles (Fan, 2021). En estas colas de trabajo se pueden ejecutar diversas aplicaciones desarrolladas en distintos lenguajes de programación como Java, C/C++, Python, R o mediante bibliotecas disponibles en el clúster (T. Sterling et al., 2017). Las infraestructuras físicas de los sistemas HPC modernos están compuestas por componentes como la unidad central de procesamiento (CPU, *Central Processing Unit*), la memoria de acceso aleatorio (RAM, *Random Access Memory*), la red de interconexión entre nodos, los sistemas de almacenamiento distribuido y, opcionalmente, las unidades de procesamiento gráfico (GPU, *Graphics Processing Unit*) (Ramesh, 2024). Dicha infraestructura resulta esencial para abordar tareas computacionalmente intensivas como el entrenamiento de modelos de inteligencia artificial, simulaciones complejas o el procesamiento masivo de datos.

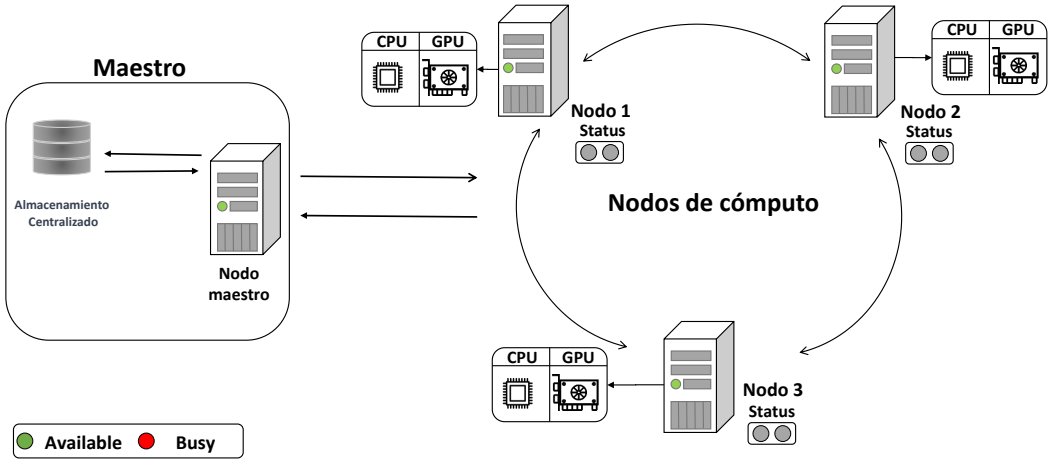

FIGURA 1.1: Arquitectura clásica de un clúster HPC, con un maestro y varios nodos de cómputo. Fuente: Elaboración propia.

1.1. Motivación

Dado el creciente uso de herramientas de inteligencia artificial, tanto en el ámbito académico como profesional, en sectores tales como las ciencias de la tierra y la predicción de desastres naturales, la industria del petróleo y gas, los servicios financieros, la detección de fraude y la fabricación e ingeniería tanto automotriz como aeroespacial, entre otros ámbitos (Hajkowicz et al., 2023).

Consecuentemente, resulta imprescindible contar con infraestructuras de cómputo HPC. El propósito de estas infraestructuras es obtener respuestas que no podrían abordarse de manera adecuada mediante técnicas empíricas o sistemas tradicionales, pero sí en modelos teóricos o simulaciones computacionales (T. Sterling et al., 2017).

En ese contexto, el diseño de un clúster HPC adquiere bastante relevancia, ya que permite escalar el procesamiento requerido para la resolución de problemas complejos. No obstante, la implementación de este tipo de infraestructuras supone un desafío técnico, el cual puede abarcar desde la planificación de la construcción de un clúster, la adquisición del *hardware*, la configuración de los componentes tanto físicos como de *software*, hasta la integración de los sistemas de gestión de tareas.

Asimismo, se identifica la necesidad de incorporar mecanismos para la automatización del despliegue de los algoritmos sobre el clúster, con el fin de reducir los errores

de las configuraciones manuales, mejorando la eficiencia operativa y permitiendo a los usuarios acceder con mayor facilidad a los recursos. Este aspecto representa una oportunidad de mejora que, además de reducir los tiempos en el despliegue de la ejecución de los algoritmos, permite adquirir un conocimiento más detallado de la infraestructura computacional sobre la cual se ejecutan los diferentes procesos.

Como se muestra en la Figura 1.2, la motivación parte de necesidades reales detectadas en el ámbito del cómputo intensivo.

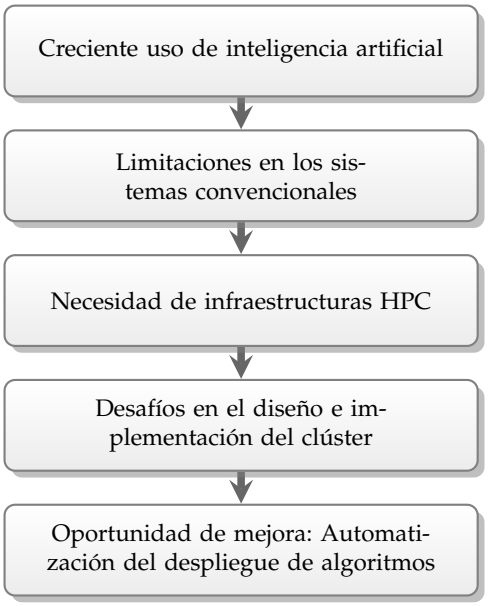

FIGURA 1.2: Esquema que resume la motivación del presente trabajo, desde el contexto general hasta la necesidad de automatización.

1.2. Objetivos

En esta sección se definen los objetivos del proyecto. Primero, en la Sección 1.2.1, se presentan los objetivos generales, que explican qué se quiere lograr en términos generales. Luego, en la Sección 1.2.2, se muestran los objetivos específicos, donde se encuentran detallados los pasos concretos que se deben seguir para cumplir con el objetivo principal.

1.2.1. Objetivo General

Diseñar e implementar un clúster de cómputo de altas prestaciones en entorno local virtualizado, basado íntegramente en software libre, que funcione tanto como referencia para futuras implementaciones similares como plataforma funcional, complementada con una herramienta que automatice el despliegue de algoritmos por parte de usuarios técnicos.

1.2.2. Objetivos Específicos

Para alcanzar los objetivos generales, se establecen los siguientes objetivos específicos:

- Identificar y comprender los componentes esenciales, tanto de *hardware* como de *software*, para la implementación de un clúster HPC en un entorno local.

- Realizar una guía paso a paso para implementar un clúster HPC local virtualizado, explicando su configuración, uso y mantenimiento como plataforma experimental.
- Diseñar una herramienta basada en *software* libre que automatice el envío y la ejecución de algoritmos en el clúster HPC, con el objetivo de simplificar su uso a usuarios técnicos.

1.3. Alcance

El presente trabajo se centra en el diseño e implementación de un clúster de cómputo de altas prestaciones en un entorno local virtualizado, utilizando exclusivamente *software* libre.

Concretamente, se abordará la instalación y configuración de los componentes esenciales del sistema, así como el desarrollo de una herramienta complementaria que automatice el envío, ejecución y monitoreo de algoritmos desde el cliente hacia el clúster.

Este trabajo se limita únicamente a entornos virtuales; si bien la infraestructura estará preparada para su despliegue futuro sobre *hardware* físico, no se abordará la adquisición de equipamiento específico ni la implementación en arquitecturas híbridas (local–nube).

1.4. Estructura de la memoria

Esta investigación se organiza en seis secciones principales, las cuales se describen en las siguientes secciones.

- **Capítulo 1: INTRODUCCIÓN**
  En este capítulo se detalla el contexto del trabajo, la motivación, los objetivos generales y específicos, el alcance, las limitaciones y la estructura de la memoria.
- **Capítulo 2: REVISIÓN DEL ESTADO DEL ARTE**
  En este capítulo se presentan los principales componentes de *hardware* y *software* de un clúster HPC, y se analizan trabajos relacionados con implementaciones similares y herramientas para el despliegue de algoritmos.
- **Capítulo 3: PLAN DEL PROYECTO**
  En este capítulo se detallan las actividades planificadas para llevar a cabo la implementación del clúster HPC local virtualizado y el desarrollo de la herramienta de automatización del despliegue de algoritmos.
- **Capítulo 4: DISEÑO Y ARQUITECTURA DEL SISTEMA HPC**
  En este capítulo se seleccionan las herramientas y componentes necesarios para construir un clúster HPC, se define su arquitectura general y se describe el proceso de instalación y configuración del sistema a nivel de *software*.
- **Capítulo 5: DISEÑO DEL SOFTWARE DE DESPLIEGUE AUTOMATIZADO DE ALGORITMOS**

En este capítulo se analiza, diseña y desarrolla una herramienta destinada a automatizar el despliegue de algoritmos en el clúster HPC.

- **Capítulo 6: CONCLUSIONES Y TRABAJO FUTURO**
  En este capítulo se presentan las conclusiones del proyecto, destacando las prestaciones logradas con la construcción del clúster HPC y el desarrollo de la herramienta de despliegue. Además, se proponen posibles líneas de trabajo futuro para continuar y mejorar lo desarrollado.

Capítulo 2

REVISIÓN DEL ESTADO DEL ARTE

2.1. Marco Actual

Para abarcar el marco actual centrado en HPC, es necesario comprender una amplia variedad de aspectos especializados relacionados tanto con la infraestructura de *hardware* como la de *software* que soportan este tipo de entorno (T. L. Sterling, 2002). Asimismo, es indispensable comprender los desafíos operativos, las buenas prácticas y las necesidades específicas. A continuación, se describen algunos de los elementos fundamentales para entender cómo se diseñan, gestionan y utilizan los clústeres HPC.

2.1.1. Fundamentos claves de la Computación de Altas Prestaciones

En la actualidad, la Computación de Altas Prestaciones es considerada como un recurso esencial para la resolución de problemas complejos (Netto et al., 2018). En esta sección se abordan los conceptos fundamentales, su relevancia en el ámbito científico y técnico, así como los distintos tipos de configuraciones de clústeres. Además, se analizan los principales clústeres utilizados en centros de datos de alto rendimiento a nivel mundial.

2.1.1.1. Concepto y características de la computación HPC

La Computación de Altas Prestaciones es un campo que se ocupa del uso de sistemas con múltiples nodos para ejecutar aplicaciones que requieren una gran cantidad de operaciones. Se basa en la coordinación de recursos computacionales distribuidos para llevar a cabo tareas que no pueden resolverse con equipos convencionales. El HPC incluye tecnologías, métodos y procesos relacionados con la ejecución de cargas de trabajo mediante supercomputadoras. Las cargas de trabajo son conjuntos de tareas que un sistema debe procesar, definidos por su volumen, tipo de operaciones y requisitos computacionales (Grama et al., 2003).

2.1.1.2. Modelos de implementación

Los sistemas de Computación de Altas Prestaciones pueden desplegarse en dos grandes entornos: en local (*on-premise*) y en la nube. En el modelo local, la infraestructura se instala y se gestiona dentro de la propia organización, lo que proporciona control total sobre los recursos, autonomía en la administración y soberanía de los

datos (Ghedira et al., 2024). Dentro del entorno en la nube, la literatura distingue tres maneras de consumir los recursos. El primero, *HPC in the Cloud*, se apoya en instancias de infraestructura como servicio (IaaS, *Infrastructure as a Service*) y deja al usuario la configuración íntegra del entorno científico. En segundo lugar, el *HPC plus Cloud* o modelo híbrido combina recursos locales con nodos en la nube para ampliar la capacidad computacional. En tercer lugar, el modelo de (HPCaaS, *High Performance Computing as a Service*) abstrae la infraestructura y ofrece la capacidad de cálculo mediante las plataformas como servicio (PaaS, *platform as a service*) o aplicaciones como servicio (SaaS, *software as a service*) accesibles a través de portales o API (Netto et al., 2018).

2.1.1.3. Supercomputadoras de la lista TOP500

La TOP500 es una lista que clasifica las principales supercomputadoras más potentes del mundo. Esta lista es publicada dos veces al año y organiza a los sistemas por su potencia de cálculo, la cual es medida en operaciones de coma flotante por segundo (FLOPS, *Floating Point Operations Per Second*). Además de identificar las computadoras más rápidas, esto es de utilidad para analizar las tendencias en HPC y la evolución tecnológica (TOP500, 2025). A continuación, se presentan las diez supercomputadoras más potentes según la última edición de la lista TOP500 de noviembre de 2024, como se puede ver en la Figura 2.1.

FIGURA 2.1: Las 10 supercomputadoras más potentes. Rendimiento medido en Rmax (PFlop/s). Imagen adaptada de (TOP500, 2025).

En la Figura 2.2, se muestra el MareNostrum 5 (Barcelona Supercomputing Center, 2025), el superordenador con mayor capacidad de cómputo en España, que se encuentra diseñado para computación de alto rendimiento y cuenta con una arquitectura heterogénea basada en particiones CPU y GPU.

FIGURA 2.2: Superordenador MareNostrum 5 (España). Imagen obtenida de (Barcelona Supercomputing Center, 2025).

2.1.1.4. Implementaciones prácticas de HPC

A continuación se muestran ejemplos de investigación aplicada apoyada en herramientas de HPC. En la figura 2.3 se muestran las áreas de aplicación donde son utilizadas las supercomputadoras del TOP500.

- **Simulación de Epidemias:** Simulación computacional de la propagación de enfermedades en grandes poblaciones para evaluar intervenciones de salud pública (Chen et al., 2025).
- **Análisis Genómico:** Estudio de miles de genomas de cáncer usando sistemas de procesamiento avanzado para descubrir alteraciones genéticas compartidas y particulares de los distintos tipos de cánceres (Weinstein et al., 2013).
- **Simulación Climática:** Simulación climática global utilizando supercomputadoras para representar cómo cambian la atmósfera, los océanos, el hielo, la tierra y los ecosistemas (Hurrell et al., 2013).

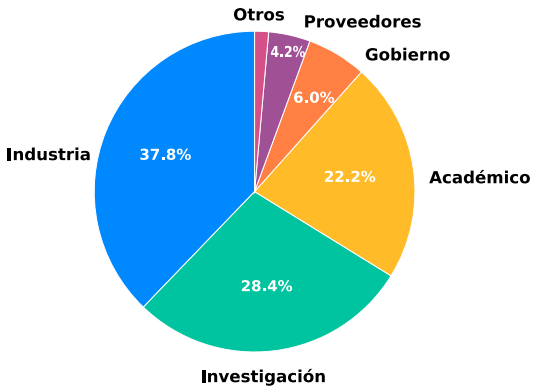

FIGURA 2.3: Distribución por áreas de aplicación del TOP500 correspondiente a noviembre de 2024. Imagen adaptada de (TOP500, 2025).

2.1.2. Arquitectura de un Clúster HPC

En esta sección se detallan los componentes clave de la arquitectura en la infraestructura para la implementación de un clúster HPC de manera local (T. Sterling et al., 2017). En la Figura 1.1 se puede observar un esquema representativo.

- **Nodo maestro:** Es el encargado de coordinar y supervisar el funcionamiento general del clúster, incluida la administración de los nodos de cómputo. Sus funciones abarcan la autenticación de usuarios, la asignación de recursos, la planificación de tareas y el acceso al sistema de almacenamiento compartido. Actúa como punto de acceso al clúster para los usuarios.
- **Nodos de cómputo:** Cada nodo de cómputo es un ordenador completo y autónomo que forma parte del clúster. Su función principal es ejecutar las tareas de procesamiento que le son asignadas por el nodo maestro.
- **Almacenamiento compartido:** Es el sistema de almacenamiento al que acceden simultáneamente los distintos nodos del clúster. Facilita la gestión de grandes volúmenes de datos, permitiendo el acceso concurrente a programas, archivos y resultados de procesamiento, así como espacios de trabajo personalizados donde cada usuario puede guardar sus scripts, datos y salidas de ejecución.
- **Red:** Es el sistema que conecta todos los ordenadores del clúster entre sí. Permite que compartan información, trabajen coordinadamente y realicen tareas en conjunto. Está formada por cables, tarjetas de conexión y dispositivos que dirigen el tráfico de datos.

2.1.3. Componentes de hardware en clústeres HPC

En esta sección se detallan los componentes clave de la infraestructura de *hardware* necesaria para la implementación local de un clúster HPC. Se distinguen dos grandes categorías: el hardware de propósito general, constituido por servidores diseñados para ofrecer un equilibrio entre rendimiento, fiabilidad y escalabilidad en una amplia gama de aplicaciones científicas y técnicas (Hewlett Packard Enterprise, 2024); y el hardware especializado o propietario, orientado a contextos de alto rendimiento mediante sistemas diseñados específicamente por el proveedor para tareas concretas, que se comercializan como soluciones integradas, preconfiguradas y cerradas, eliminando la necesidad de ensamblaje y ajuste manual (Silvano et al., 2025).

En el presente trabajo, el análisis se centra en la infraestructura basada en *hardware* de propósito general, por ser la opción más común en entornos académicos y de investigación con recursos limitados (Kumara et al., 2018).

2.1.3.1. Placa Base

Al diseñar un nodo de cómputo, la elección de la placa base influye directamente en su rendimiento, capacidad de crecimiento y adaptabilidad futura. Este componente central actúa como plataforma de conexión entre el procesador, la memoria, los dispositivos de almacenamiento y otros periféricos, incluyendo tarjetas de red o GPU (T. L. Sterling, 2002). Por ello, es crucial revisar parámetros como la cantidad de procesadores que admite (sockets), el límite de memoria que soporta, el número y tipo de ranuras PCIe disponibles para expansión, y su adecuación al chasis o bastidor del sistema (Civera, 2018). En la Figura 2.4, se muestra un ejemplo de una placa base montada en un rack.

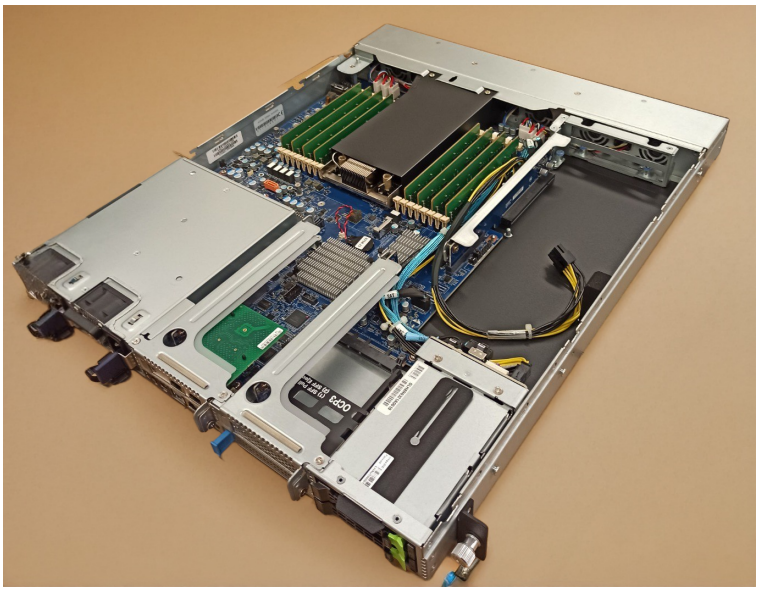

FIGURA 2.4: Placa base de servidor montada en chasis rack mostrando sockets de CPU, ranuras de memoria y slots de expansión PCIe. Fuente: Elaboración propia.

2.1.3.2. CPU

Según la lista *TOP500* de noviembre de 2024, el 94,2 % de las supercomputadoras emplean la arquitectura x86_64, predominando las implementaciones de los fabricantes *Intel* y *AMD* (TOP500, 2025). Por ello, en este trabajo se analizarán directamente estos dos fabricantes. En la Figura 2.5, se muestra un ejemplo del aspecto típico de estos procesadores.

FIGURA 2.5: Vista frontal y posterior de un procesador Intel Xeon X5570. Imagen obtenida de (Wandinger, 2017).

Entre los distintos tipos de procesadores desarrollados por *Intel*, la familia *Intel Xeon* permite abordar de manera óptima los escenarios computacionales exigentes, incluyendo inteligencia artificial, análisis de datos complejos y otras tareas propias del cómputo intensivo (I. Corporation, 2025). La familia *Intel Xeon* se clasifica en cuatro niveles: *Bronze*, *Silver*, *Gold* y *Platinum*, los cuales permiten identificar rápidamente el grado de prestaciones y escalabilidad del procesador (I. Corporation, 2024). Por otro lado, los procesadores *AMD* han ganado una presencia cada vez más significativa en el ámbito del cómputo de altas prestaciones en los últimos años y esto queda reflejado en la TOP500 de noviembre de 2024 (TOP500, 2025), lo cual se debe a su familia de procesadores *EPYC* que está orientada a servidores y centros de datos (AMD, 2024).

La familia *EPYC* se organiza en generaciones sucesivas como: *Naples*, *Rome*, *Milan* y *Genoa*, cada una introduciendo mejoras sustanciales en arquitectura, rendimiento y eficiencia energética (Advanced Micro Devices, Inc., 2023). Independientemente de si se opta por procesadores *Intel* o *AMD*, ambos fabricantes disponen de unidades que, aunque no estén diseñadas específicamente para servidores de alto rendimiento, pueden ser adecuadas para cargas de trabajo exigentes, siempre que el entorno lo permita (Kinghorn, 2025). La elección del procesador debe basarse en características clave como el recuento de núcleos, la frecuencia de operación y la compatibilidad con arquitecturas multiprocesador, aspectos que afectan directamente al rendimiento y escalabilidad del sistema (Civera, 2018).

2.1.3.3. GPU

La GPU es un procesador diseñado para ejecutar cálculos en paralelo, concebido inicialmente para acelerar el procesamiento gráfico, pero actualmente es ampliamente utilizado para tareas como inteligencia artificial y simulaciones científicas, esto es debido a su gran capacidad para gestionar miles de hilos simultáneamente (Wael & Madi, 2025).

Según el análisis publicado por *Air Street Press*, como parte de su informe anual sobre tendencias en inteligencia artificial, NVIDIA mantiene una posición dominante en la investigación académica. De acuerdo con los datos presentados en la edición correspondiente a 2024, el 91 % de los artículos académicos en IA utilizaron GPUs de esta marca. Entre los modelos más utilizados se encuentran las GPU: RTX 2080, RTX 3090, RTX 4090, K80, P100, V100, A100, H100, Titan y los chips Jetson (Press, 2024). En la Figura 2.6, se muestra un ejemplo del aspecto de un acelerador gráfico.

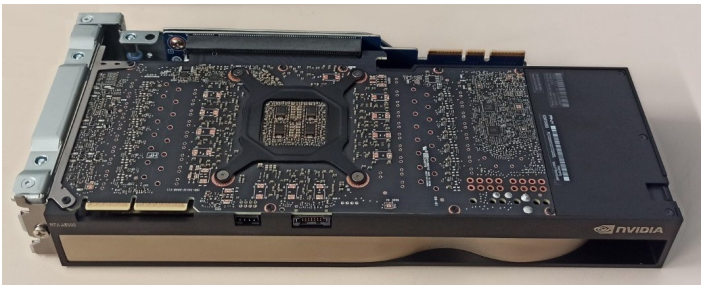

FIGURA 2.6: GPU NVIDIA RTX A5000. Fuente: Elaboración propia.

NVIDIA ofrece una amplia variedad de GPUs, que pueden emplearse en tareas como inteligencia artificial, simulaciones científicas o procesamiento de datos. Los modelos disponibles, tanto actuales como clasificados como *legacy*, se encuentran listados en los sitios mantenidos por la compañía (NVIDIA Corporation, 2024a, 2024b).

2.1.3.4. Red

En los HPC, la red es esencial para permitir la comunicación constante entre nodos durante la ejecución paralela. Dado que los procesos deben intercambiar datos de forma sincronizada, es fundamental contar con una red que minimice la latencia y maximice el ancho de banda para evitar cuellos de botella (Grama et al., 2003).

Los tipos de interconexión utilizados en sistemas HPC se organizan habitualmente en seis grandes familias, según la clasificación incluida en la edición de noviembre de 2024 de la lista TOP500 (TOP500, 2025). Estas familias son InfiniBand, Gigabit Ethernet, Omni-Path, Custom Interconnect, Proprietary Network y Ethernet. Tal como señala la literatura especializada, cada una de estas tecnologías presenta características distintas en cuanto a latencia, ancho de banda y escalabilidad, lo que condiciona su aplicación en función de las necesidades específicas. En algunos entornos, también es habitual encontrar configuraciones híbridas que combinan distintas tecnologías de interconexión (Lu et al., 2022).

2.1.4. Software base en computación HPC

La computación HPC no se define únicamente por su infraestructura física, sino también por el conjunto de herramientas de *software* que hacen posible su funcionamiento y gestión. Este conjunto incluye tanto el Sistema Operativo (SO) como el gestor de cola de trabajo que controla los nodos y las utilidades responsables de coordinar la ejecución de procesos, distribuir la carga computacional y administrar los recursos del clúster (Civera, 2018). En este contexto, uno de los modelos más reconocidos en la construcción de clústeres HPC es el enfoque Beowulf. Este tipo de arquitectura se basa en la cooperación de múltiples nodos interconectados que trabajan en paralelo para resolver tareas exigentes, utilizando *hardware* de propósito general, como servidores o estaciones de trabajo convencionales, y *software* libre o de bajo coste, como Linux como SO más empleado en este tipo de entornos (T. L. Sterling, 2002).

2.1.4.1. Sistemas Operativos: Linux y sus variantes

Según la lista *TOP500* publicada en noviembre de 2024, Linux sigue siendo el SO más adoptado en las supercomputadoras más potentes del mundo. Todos los sistemas operativos listados utilizan Linux o una variante de él, tanto en versiones libres como comerciales. Se utilizan diversas distribuciones, tales como (RHEL, *Red Hat Enterprise Linux*), (SLES, *SUSE Linux Enterprise Server*), *Ubuntu*, y entornos especializados como (CLE, *Cray Linux Environment*) y (HPE Cray OS, *HPE Cray Operating System*), entre otros. La Tabla 2.1 presenta las diez principales distribuciones de sistemas operativos, ordenadas según la cantidad de implementaciones (TOP500, 2025).

Sistema Operativo	Cantidad
Linux	190
CentOS	34
Red Hat Enterprise Linux	33
HPE Cray OS	26
Cray Linux Environment	17
Ubuntu 22.04	9
Ubuntu 22.04.3 LTS	8
Bullx SCS	8
Linux/TOSS	8
RHEL 8.6	7

TABLA 2.1: Las diez distribuciones de sistemas operativos más utilizadas en supercomputadoras. Datos adaptados de (TOP500, 2025).

2.1.5. Soluciones integradas para la gestión de clústeres HPC

Las soluciones integradas para la construcción de un clúster HPC pueden implementarse tanto en entornos locales como en plataformas en la nube. La Subsección 2.1.6 muestra algunas de las principales herramientas utilizadas en despliegues tradicionalmente instalados en entornos locales, mientras que la Subsección 2.1.7 presenta alternativas empleadas en entornos en la nube.

2.1.6. Soluciones locales

En primer lugar, entre las soluciones integradas para la gestión de clústeres HPC se encuentran alternativas libres como OpenHPC, respaldada por la Linux Foundation, que ofrece un conjunto modular de herramientas para facilitar la implementación y administración de este tipo de infraestructuras en sistemas Linux (OpenHPC Community, 2025). Otra opción es xCAT (Extreme Cloud Administration Toolkit), orientada a la gestión de grandes volúmenes de servidores, con soporte para múltiples sistemas operativos y la posibilidad de aplicar configuraciones completas en tiempos reducidos (xCAT Project, 2025). Por su parte, Warewulf permite desplegar y administrar nodos Linux en un clúster mediante un enfoque *stateless*, en el que cada nodo carga su entorno desde un servidor central directamente en memoria, sin conservar datos tras el reinicio (Warewulf Project, 2025).

En cuanto a las alternativas comerciales, NVIDIA Base Command Manager está centrada en la gestión de clústeres orientados a IA y HPC, con funciones de aprovisionamiento, monitoreo y administración de cargas de trabajo sobre plataformas NVIDIA, a través de una interfaz web que facilita el control de los recursos (NVIDIA Corporation, 2025). También cabe mencionar PBS Professional, un sistema avanzado de planificación de cargas de trabajo en entornos HPC, que permite automatizar y optimizar la ejecución de tareas a gran escala. Entre sus principales características se incluyen la planificación jerárquica, la simulación de decisiones del planificador, la gestión de créditos, soporte de seguridad multinivel, y una interfaz gráfica integrada para la administración de recursos distribuidos (Altair Engineering Inc., 2025). La apariencia de esta interfaz se muestra en la Figura 2.7.

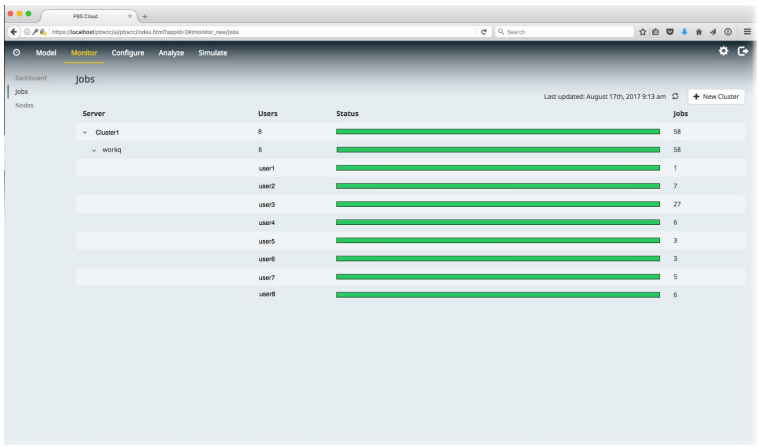

FIGURA 2.7: Monitor de trabajos en Altair PBS Professional. Imagen obtenida de (Hewlett Packard Enterprise, 2025).

2.1.7. Soluciones en la nube

En el ámbito de las soluciones basadas en la nube, algunas de las más destacadas son AWS ParallelCluster de Amazon Web Services (Amazon Web Services, 2025), Azure CycleCloud de Microsoft Azure (Microsoft Corporation, 2025a) y HPC Toolkit de Google Cloud Platform (Google Cloud, 2025). Estas herramientas permiten desplegar y gestionar clústeres HPC de forma sencilla sobre infraestructuras en la nube. Su objetivo es facilitar el aprovisionamiento dinámico de recursos, optimizar el uso de la infraestructura y proporcionar interfaces que simplifican la administración, todo ello sin necesidad de mantener entornos físicos dedicados, como se muestra en la Figura 2.8 la creación de un clúster en Azure CycleCloud.

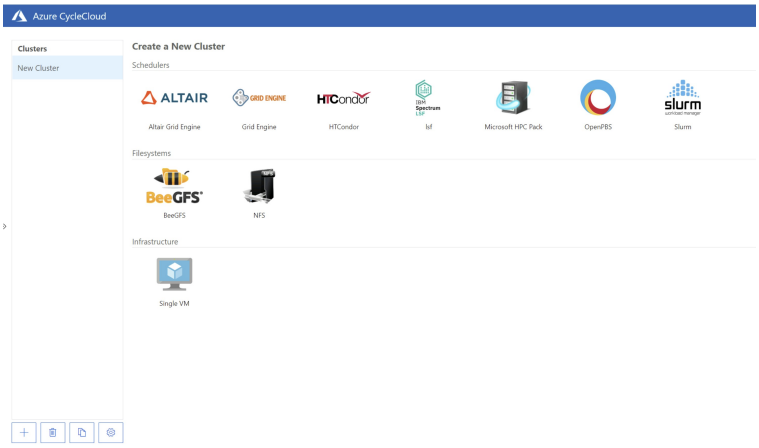

FIGURA 2.8: Vista de Azure CycleCloud para crear un nuevo clúster. Imagen obtenida de (Microsoft Corporation, 2025b).

2.1.8. Sistemas de gestión de recursos y planificación de trabajos en entornos HPC

En los entornos HPC, es fundamental contar con mecanismos que permitan organizar y distribuir adecuadamente los recursos del clúster entre los diferentes usuarios. Para ello, se emplean sistemas de gestión de recursos y planificación de trabajos, responsables de asignar los recursos físicos disponibles, según las necesidades de cada tarea, y de planificar su ejecución mediante un sistema de colas que tiene en cuenta criterios como la prioridad del usuario, las dependencias entre trabajos, los tiempos estimados de ejecución y la disponibilidad de recursos. Además, permiten supervisar el estado de los trabajos (Feitelson, 1997). En la práctica, estas funciones se gestionan mediante herramientas que organizan cómo se reparten los recursos y se ejecutan los trabajos. Hay varias opciones disponibles, y su uso varía según las necesidades del sistema. Un estudio reciente de Hyperion Research, basado en datos de 181 centros que operan más de 3.800 sistemas HPC, muestra cuáles son los gestores más utilizados en distintos sectores. La Tabla 2.2 resume los resultados más relevantes de ese estudio (Research, 2023).

Herramienta	Total (%)	Industria (%)	Gobierno (%)	Academia (%)
Slurm	50.0	40.7	70.0	61.5
OpenPBS	18.9	17.6	20.0	21.2
PBS Pro	13.9	12.0	20.0	15.4
Torque	13.3	13.9	20.0	9.6
NQS	12.2	14.8	15.0	5.8
LSF	10.6	7.4	20.0	13.5
Moab	9.4	11.1	5.0	7.7
DQS	6.1	5.6	5.0	7.7
Load Leveler	5.6	7.4	5.0	1.9
Maui	5.0	5.6	5.0	3.8
Univa/Altair Grid Engine (SGE)	5.0	5.6	–	5.8
Otro	8.3	10.2	15.0	1.9

TABLA 2.2: Porcentaje de uso de gestores de recursos HPC por sector.
Fuente: Hyperion Research (Research, 2023).

2.1.9. Gestión centralizada de identidades

En los clústeres de computación de alto rendimiento es fundamental contar con una gestión centralizada de identidades para que todos los nodos compartan la misma información de los usuarios. De esta manera, se garantiza el acceso unificado a los recursos y así simplificar las tareas administrativas por parte de los administradores del HPC (Hofert, 2024). Existen varias tecnologías que permiten implementar este tipo de solución, y cada una presenta ventajas y desventajas según las necesidades del entorno.

Entre las primeras opciones utilizadas en entornos distribuidos se encuentra el (NIS, *Network Information Service*), que organiza los datos de usuario desde un nodo central hacia el resto del clúster, permitiendo la gestión compartida de cuentas. Sin embargo, no ofrece cifrado en la comunicación ni escalabilidad adecuada para entornos más modernos (Fernández Palau, 2023).

Una opción más avanzada es (FreeIPA, *free identity, policy, and audit*), una solución integrada que centraliza la autenticación, autorización y gestión de identidades en entornos de red Linux/UNIX, almacenando y administrando información sobre usuarios, grupos y equipos para reforzar la seguridad del sistema (FreeIPA Project, 2025a). FreeIPA combina tecnologías como (LDAP, *Lightweight Directory Access Protocol*), que permite organizar y acceder de forma jerárquica a la información de usuarios y permisos. Kerberos, un mecanismo de autenticación desarrollado por el (MIT, *Massachusetts Institute of Technology*) que permite verificar la identidad de los usuarios en redes de forma cifrada y centralizada. Como también el (DNS, *Domain Name System*), el sistema que traduce nombres de dominio comprensibles por humanos en direcciones de red utilizadas por los equipos y, por último los certificados digitales que verifican la identidad de usuarios o servicios en una red mediante criptografía (Butterfield & Ngondi, 2016).

Como alternativa, se puede usar LDAP directo junto con el (NSCD, *Name Service Cache Daemon*), una solución más simple que mejora el rendimiento al guardar en caché los datos de usuario. Esta combinación demostró ser más estable y rápida en entornos HPC exigentes (Nabulsi & Hamidi, 2021).

2.1.10. Almacenamiento compartido

En la computación de alto rendimiento, el almacenamiento compartido permite que múltiples nodos accedan simultáneamente a grandes volúmenes de datos. Para ello, se emplean sistemas de archivos paralelos, como Lustre, BeeGFS, entre otros, diseñados para maximizar el rendimiento y la escalabilidad. En contraste, soluciones tradicionales como (NFS, *Network File System*) resultan inadecuadas para las cargas intensivas y altamente concurrentes típicas de estos entornos (Lüttgau et al., 2018). No obstante, NFS puede resultar útil en clústeres donde el acceso a los datos no es intensivo, o en entornos donde se prioriza la simplicidad y la compatibilidad con sistemas convencionales como en el trabajo de Kumar et al. (Kumara et al., 2018) .

2.1.11. Gestión de software en HPC

La gestión de *software* en un sistema HPC puede volverse bastante complicada. A diferencia de un entorno de escritorio tradicional, donde suele ser suficiente mantener una única versión de cada programa, en un sistema HPC es habitual instalar múltiples versiones del mismo *software*, cada una optimizada para diferentes configuraciones. Esta práctica busca principalmente garantizar la reproducibilidad de los experimentos científicos en las mismas condiciones en que fueron desarrollados (Geimer et al., 2014). Para simplificar esta tarea, existen herramientas como EasyBuild y Spack, que permiten instalar y organizar el *software* de forma más ordenada y adaptada a las necesidades específicas.

Entre las soluciones disponibles, EasyBuild es un *framework* en Python diseñado para facilitar la instalación de *software* en sistemas HPC, automatizando tareas como la compilación, la gestión de dependencias y la configuración del entorno de ejecución (Geimer et al., 2014).

Por otra parte, Spack es un gestor de paquetes para HPC que permite instalar *software* científico con configuraciones altamente personalizables. A diferencia de EasyBuild, resuelve dependencias de forma dinámica en lugar de usar recetas fijas (Gamblin et al., 2015).

2.1.12. Monitoreo y visualización del rendimiento

Monitorear un clúster HPC es clave para asegurarse de que todo funcione correctamente y los recursos se aprovechen bien. Cuando se tiene visibilidad sobre el uso de CPU, memoria, red o almacenamiento, es más fácil detectar si algún nodo está fallando, si hay sobrecarga o si el rendimiento no es el esperado. Además, contar con datos históricos permite analizar cómo evoluciona el uso del sistema y tomar decisiones más acertadas sobre mantenimiento o ampliaciones futuras. Una de las herramientas más utilizadas para estas tareas es Ganglia. Está pensada para monitorizar clústeres con muchos nodos, consume muy pocos recursos y ofrece una interfaz web clara para consultar las métricas tanto en tiempo real como el histórico. Gracias a su diseño ligero y escalable, se ha convertido en una opción muy común en centros de supercomputación (Massie et al., 2004).

2.2. Soluciones Existentes

Dentro del ámbito académico y científico se han desarrollado numerosas investigaciones relacionadas con la construcción de infraestructuras de HPC. En la Subsección 2.2.1 se detallan trabajos centrados en la implementación de clústeres HPC, mientras que en la Subsección 2.2.2 se incluyen enfoques orientados a automatizar el despliegue de algoritmos hacia el clúster.

2.2.1. Implementación Clúster HPC

Con el objetivo de analizar enfoques previos relevantes, las Tabla 2.3 y Tabla 2.4 detallan distintas implementaciones de clústeres HPC, destacando sus características técnicas, configuraciones y herramientas empleadas.

Trabajo	Descripción del trabajo	Referencia
Implementación de un clúster HPC en el Centro de Estudios de Física del Cosmos de Aragón	Se implementa un clúster HPC para el procesamiento de datos del Observatorio Astrofísico de Javalambre, incluyendo el diseño de la infraestructura, selección del *hardware* y configuración del entorno de *software*.	(Civera, 2018)
Administración de nodos de cómputo de altas prestaciones	Se instala un clúster HPC de tres nodos, cada uno con configuraciones diferenciadas según su función a realizar. Además, se automatizan tareas administrativas a través de scripts, lo cual permite validar el sistema con pruebas de rendimiento en un entorno educativo real	(García Peña et al., 2021)
High Performance Computing Cluster Setup: A Tutorial	Guía para configurar y trabajar con un clúster HPC. Incluye conexión remota, entorno de usuario, instalación de *software* y uso de Slurm para ejecutar y monitorear trabajos y además incluye consejos prácticos para usuarios nuevos	(Hofert, 2024)

TABLA 2.3: Trabajos seleccionados como estudios previos más relevantes y su ámbito de aplicación.

Trabajo	Descripción del trabajo	Referencia
Building a Shared Resource HPC Center Across University Schools and Institutes: A Case Study	Este artículo describe la creación de un centro HPC en una universidad, abordando desafíos, soluciones, planificación, modelo de costos, recursos, personal, métricas de éxito y estrategias para involucrar a los usuarios.	(MacLachlan et al., 2020)
Design and implementation of a scalable high-performance computing (HPC) cluster for omics data analysis: achievements, challenges and recommendations in LMICs	Este artículo describe la implementación de un clúster HPC destinado al análisis de datos, mostrando los principales logros, los retos técnicos enfrentados y unas recomendaciones útiles para instituciones con recursos limitados que deseen desarrollar infraestructuras similares.	(Ghedira et al., 2024)
Design and Implementation of High Performance Computing (HPC) Cluster	Se realiza la implementación de un clúster HPC sobre Ubuntu usando MPI, detallando su construcción con recursos de bajo costo y evaluando su rendimiento mediante pruebas de ejecución con distintos números de procesos.	(Kumara et al., 2018)
Migración de un clúster HPC	Se moderniza un laboratorio HPC mediante la implementación de un nuevo clúster unificado que integre nodos CPU y GPU, mejorando la gestión y seguridad con Slurm, EasyBuild, LDAP y backups remotos.	(Fernández Palau, 2023)

TABLA 2.4: Trabajos seleccionados como estudios previos más relevantes y su ámbito de aplicación.

2.2.2. Software relacionado con Automatización

En la tabla 2.5 se detallan herramientas que permiten definir y gestionar flujos de trabajo en entornos HPC. Aunque automatizan diversos aspectos del proceso, en general requieren que el usuario se conecte manualmente al clúster y ejecute los comandos.

Software	Descripción	Consideraciones
Nextflow (Nextflow Project, 2024)	Motor de ejecución para flujos reproducibles y escalables.	Muy usado en bioinformática. Requiere aprender un lenguaje propio. Se integra con muchos gestores de cola. No gestiona acceso remoto, se ejecuta dentro del clúster.
Snakemake (Snakemake Development Team, 2024)	Framework declarativo basado en Python para la construcción de pipelines científicos reproducibles y escalables.	Integración nativa con Conda y entornos HPC. Curva de aprendizaje ligera. No gestiona acceso remoto, se ejecuta dentro del clúster.
FireWorks (FireWorks Development Team, 2024)	Planificador de tareas orientado a HPC, basado en JSON y MongoDB.	Requiere base de datos MongoDB. Potente para flujos complejos, pero configuración más técnica. No gestiona acceso remoto, se ejecuta dentro del clúster.
VS Code Remote (M. Corporation, 2024)	Extensión de Visual Studio Code que abre y edita código directamente en servidores remotos a través de SSH.	La herramienta permite editar código en servidores remotos mediante SSH, pero no automatiza el flujo de trabajo HPC ni se integra con Slurm. Además, depende del uso de VS Code.
PyCharm Remote (JetBrains, 2024)	Función de la edición Professional de PyCharm para trabajar sobre servidores remotos mediante SSH.	Permite editar código y sincronizar archivos con servidores remotos vía SSH. Sin embargo, no está diseñado para entornos HPC, ya que no gestiona nodos de cómputo ni permite el envío de trabajos mediante Slurm automáticamente.

TABLA 2.5: Software relacionado con automatización y sus consideraciones.

2.3. Conclusiones del estudio realizado

Según la revisión y el estudio realizados, el campo del cómputo de altas prestaciones presenta múltiples modelos de implementación. Estos modelos han sido adoptados, en distintos ámbitos, para abordar una amplia variedad de problemas computacionales. Actualmente, existen numerosos centros de datos dedicados al HPC, y algunos de los más potentes se encuentran clasificados en la lista TOP500.

Es importante destacar que los clústeres HPC pueden ser soluciones comerciales ya construidas por fabricantes con propósito específico, o bien sistemas personalizados, donde se adquiere el *hardware* de propósito general por separado para montar una solución a medida. Este último enfoque se conoce como clúster Beowulf, el cual permite seleccionar tanto el *hardware* como el *software* que se desplegará en el entorno HPC. En este tipo de soluciones, es fundamental considerar diversos criterios para la elección del *hardware*, incluyendo los componentes clave como la CPU, GPU, RAM, placa base y la red de interconexión entre nodos.

En cuanto al *software*, existe una amplia variedad de opciones, por lo que su selección debe basarse en las necesidades específicas del entorno y los recursos disponibles. Esta elección abarca desde el SO hasta las soluciones de gestión de recursos, centralización de identidades y sistemas de almacenamiento compartido que aseguren el acceso común a los datos entre los nodos. Asimismo, resulta esencial ofrecer a los usuarios la posibilidad de seleccionar el *software* y las versiones que necesiten para sus experimentos, garantizando que no haya conflictos entre entornos.

Adicionalmente, es crucial contar con mecanismos que permitan registrar el uso de los recursos del clúster, con el objetivo de identificar posibles cuellos de botella y oportunidades de mejora en el rendimiento general del sistema.

A partir del análisis de diversas implementaciones de clústeres HPC, se observa que existe una amplia disponibilidad de documentación técnica. No obstante, son pocas las soluciones orientadas a la automatización del envío de trabajos al clúster, y aquellas que existen no automatizan el flujo de trabajo de manera completa. Este tipo de automatización sería especialmente útil para facilitar la interacción de los usuarios técnicos, evitando que deban profundizar en el funcionamiento interno de los sistemas de gestión y planificación de recursos.

Capítulo 3

PROPUESTAS

3.1. Descripción

Partiendo de los objetivos de la Sección 1.2, este proyecto plantea diseñar una infraestructura HPC funcional y documentada que sirva como guía para su implementación en entornos locales, tal como se desarrolla en el Capítulo 4, donde se analizan los componentes utilizados y se aborda su implementación a nivel de *software*. Asimismo, se pretende desarrollar una herramienta que, desde el cliente o entorno local, automatice el despliegue de algoritmos hacia el clúster, incluyendo la preparación del entorno, el envío de trabajos y su posterior monitoreo. Este desarrollo se detalla en el Capítulo 5, orientado a facilitar el trabajo del usuario técnico.

3.2. Metodología

La metodología utilizada para llevar a cabo este trabajo es el modelo en cascada iterativo. Su elección se debe a que permite la retroalimentación entre etapas, a diferencia del enfoque clásico, que sigue una secuencia rígida sin posibilidad de volver a fases anteriores. Este modelo organiza el proceso en fases bien definidas: análisis, diseño, implementación, pruebas y mantenimiento (Pressman, 2010). Aunque el modelo completo incluye una fase de mantenimiento, esta queda fuera del alcance del presente TFM.

De esta forma, la adopción del enfoque iterativo responde tanto a las características técnicas de este trabajo como a la necesidad de una estructura secuencial que permita introducir ajustes durante el desarrollo.

En la Figura 3.1, se representa este modelo, en el que las fases se suceden de manera ordenada, pero con posibilidad de retroceder a etapas anteriores cuando se detectan errores o nuevas necesidades, lo que constituye una de las principales diferencias con respecto al enfoque en cascada tradicional.

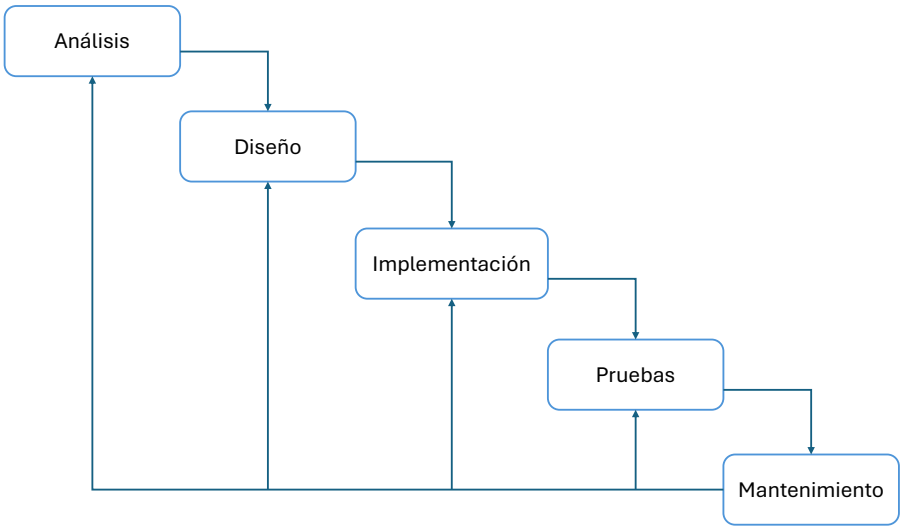

FIGURA 3.1: Modelo en cascada iterativo con retroalimentación entre fases. Imagen adaptada de (Sommerville, 2016).

3.3. Cronograma del proyecto

La planificación temporal del proyecto se organiza a través de un diagrama de Gantt, donde se reflejan las distintas tareas y fases distribuidas a lo largo del calendario. En la Tabla 3.1 se presenta un resumen de las actividades principales organizadas por etapas, mientras que en la Figura 3.2 se puede apreciar la secuencia con la duración de las actividades a lo largo del desarrollo del trabajo.

Etapa	**Actividad**	**Período de Ejecución**	
		Inicio	**Fin**
Diseño y arquitectura del sistema HPC	Análisis	05/02/2025	18/04/2025
	Diseño	17/04/2025	30/04/2025
	Implementación	28/04/2025	16/05/2025
	Pruebas	15/05/2025	19/05/2025
Diseño del software de despliegue automatizado de algoritmos	Análisis	18/04/2025	12/05/2025
	Diseño	09/05/2025	23/05/2025
	Implementación	14/05/2025	28/05/2025
	Pruebas	29/05/2025	02/06/2025
Documentaciones	Manual de administración del servidor	16/05/2025	19/05/2025
	Manual del usuario	16/05/2025	22/05/2025
	Elaboración de la memoria TFM	20/03/2025	27/06/2025

TABLA 3.1: Cronograma resumido de actividades principales.

FIGURA 3.2: Diagrama de Gantt del TFM con la distribución de tareas por fechas, elaborado con la herramienta OpenProject.

Capítulo 4

DISEÑO Y ARQUITECTURA DEL SISTEMA HPC

4.1. Análisis de Requisitos

En esta sección se especifican los requisitos del sistema, clasificados en dos categorías: los requisitos funcionales, descritos en la Subsección 4.1.1, y los requisitos no funcionales, presentados en la Subsección 4.1.2.

4.1.1. Requisitos funcionales

A continuación, se enumeran los requisitos funcionales identificados a partir del análisis de las necesidades técnicas. Cada uno de ellos describe de manera verificable el comportamiento esperado del sistema.

RF1. **Acceso y autenticación:** El sistema debe restringir el acceso al clúster únicamente a través del nodo maestro mediante protocolo SSH, mientras no existan trabajos pendiente en los nodos de computo. Los usuarios deberán autenticarse con credenciales gestionadas centralizadamente mediante FreeIPA. No se permitirá el acceso directo a los nodos de cómputo ni la escalada de privilegios con `sudo`, salvo a los usuarios pertenecientes al grupo de administración del clúster.

RF2. **Gestión de políticas de uso y asignación de recursos:** El sistema debe aplicar reglas diferenciadas según el tipo de usuario (investigadores o invitado), limitando la cuota de disco, la prioridad en la cola de trabajos y el acceso a recursos de cómputo. Todos los trabajos deberán enviarse obligatoriamente mediante el sistema de colas Slurm.

RF3. **Planificación de trabajos:** El planificador de trabajos debe ser capaz de identificar huecos entre ejecuciones largas y asignarles trabajos de corta duración, optimizando así el uso de los recursos disponibles sin afectar la política de prioridad establecida.

RF4. **Monitorización de recursos:** El sistema debe proporcionar acceso a métricas de uso de CPU, memoria, almacenamiento y estado de los nodos en tiempo real e histórico, a través de herramientas como Ganglia.

RF5. **Almacenamiento compartido:** El sistema debe montar un sistema de archivos compartido mediante NFS que permita a todos los usuarios acceder a sus directorios personales y archivos de proyecto desde cualquier nodo del clúster sin necesidad de configuración manual.

RF6. **Instalación y uso de *software* científico:** El sistema debe permitir a los usuarios cargar, listar y utilizar versiones específicas de paquetes científicos mediante *EasyBuild* y módulos de entorno, sin interferir con las configuraciones del sistema.

RF7. **Notificaciones por correo:** Al finalizar la ejecución de un trabajo, el sistema debe enviar automáticamente una notificación por correo electrónico al usuario, incluyendo el estado de finalización, duración del trabajo.

4.1.2. Requisitos no funcionales

A continuación, se enumeran los requisitos no funcionales identificados a partir del análisis de las necesidades técnicas.

RNF1. **Licenciamiento:** El sistema debe utilizar exclusivamente *software* libre en todos sus componentes principales, salvo en casos justificados donde una herramienta propietaria aporte una funcionalidad crítica no cubierta por soluciones abiertas.

RNF2. **Soporte del Sistema Operativo:** El SO debe estar en periodo de soporte oficial (activo o de mantenimiento) según las especificaciones del proveedor, con posibilidad de obtener actualizaciones de seguridad y soporte comunitario o empresarial.

RNF3. **Escalabilidad:** El clúster debe permitir la incorporación de nuevos nodos de cómputo sin reconfiguración completa del sistema, garantizando compatibilidad con hardware heterogéneo y manteniendo la integridad de la arquitectura existente.

RNF4. **Seguridad:** El sistema debe implementar un modelo de seguridad multicapa conforme a las recomendaciones del estándar ISO/IEC 27002:2022. Esto incluye autenticación centralizada mediante FreeIPA, cortafuegos por nodo, cifrado de las comunicaciones internas mediante SSH, y gestión de permisos basada en roles. Estas medidas deben garantizar la confidencialidad, integridad y disponibilidad del sistema y de los datos procesados.

RNF5. **Usabilidad:** El sistema debe ofrecer una interfaz clara e intuitiva que permita a usuarios técnicos sin experiencia previa en HPC ejecutar tareas comunes de forma autónoma, sin necesidad de conocimientos avanzados ni asistencia permanente. La solución debe estar respaldada por documentación accesible y organizada que facilite su adopción.

4.2. Casos de uso

Esta sección presenta los casos de uso que ilustran las acciones que el administrador y los distintos grupos de usuarios pueden realizar sobre el clúster HPC. En la Figura 4.1 se muestra el diagrama de casos de uso.

4.2.1. Actores del sistema

- **Usuario investigador:** Personal académico con mayor acceso a los recursos del clúster, incluyendo trabajos de larga duración (hasta 7 días) y mayor capacidad de almacenamiento.
- **Usuario invitado:** Personal con acceso limitado al clúster, restringido a trabajos de corta duración (máximo 6 horas) y menor capacidad de almacenamiento.
- **Administrador del clúster:** Personal técnico responsable de la gestión, configuración y mantenimiento del clúster HPC.

4.2.2. Diagrama de caso de uso

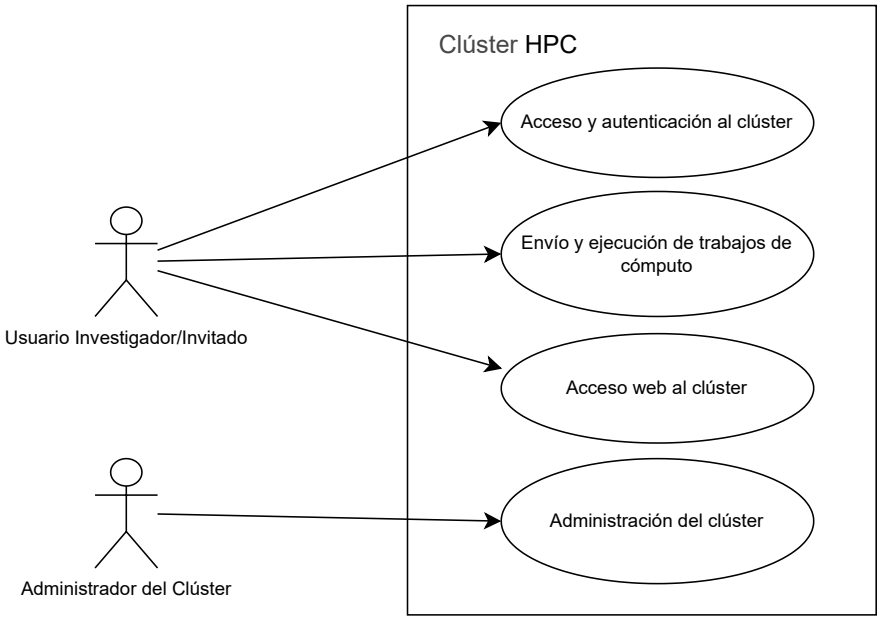

FIGURA 4.1: Diagrama de caso de uso general que ilustra las acciones de los usuarios.

4.2.3. Acceso y autenticación al clúster

Nombre	**1 - Acceso y autenticación al clúster**
Caso de Uso General	Conexión de usuarios al sistema HPC usando SSH y autenticación centralizada.
Descripción	Los usuarios se conectan al clúster a través del nodo maestro utilizando algún cliente SSH. El sistema verifica sus credenciales contra FreeIPA y les da acceso según el grupo al que pertenecen (investigadores o invitados). Una vez dentro, pueden acceder a su directorio personal y a los recursos compartidos.
Actor Principal	Usuario Investigador/Invitado
Actor Secundario	Ninguno.
Precondiciones	Tener credenciales válidas creadas en FreeIPA. El nodo maestro debe estar funcionando y accesible desde la red.
Postcondiciones	a. Usuario autenticado correctamente. b. Sesión SSH activa. c. Directorio personal montado y accesible. d. Permisos asignados según grupo de usuario.
Secuencias de Acciones	**Usuario:** a. Se conecta al nodo maestro mediante un cliente SSH. b. Introduce usuario y contraseña. c. El sistema valida credenciales en FreeIPA. d. Obtiene acceso al entorno de trabajo.
Validaciones/Restricciones	a. Solo se permite conexión directa al nodo maestro. b. Credenciales deben estar activas en FreeIPA. c. Los usuarios no tienen privilegios elevados, excepto los administradores.

TABLA 4.1: Caso de uso - Acceso y autenticación al clúster.

4.2.4. Envío y ejecución de trabajos de cómputo

Nombre	**2 - Envío y ejecución de trabajos de cómputo**
Caso de Uso General	Ejecución de algoritmos y aplicaciones en los nodos de cómputo mediante Slurm.
Descripción	El usuario prepara su código y lo envía a la cola de trabajos de Slurm. El sistema asigna recursos disponibles según las políticas configuradas y ejecuta el trabajo en los nodos de cómputo. Al finalizar, se envía una notificación por email y los resultados quedan disponibles en el directorio del usuario.
Actor Principal	Usuario Investigador/Invitado
Actor Secundario	Ninguno.
Precondiciones	Usuario autenticado en el sistema. Script de trabajo preparado. Nodos de cómputo operativos.
Postcondiciones	a. Trabajo enviado a la cola correctamente. b. Ejecución completada en nodos asignados. c. Resultados guardados en directorio personal. d. Notificación email enviada.
Secuencias de Acciones	**Usuario:** a. Prepara script con algoritmo y datos necesarios. b. Transfiere archivos al servidor mediante scp, rsync o File Browser de Open OnDemand. c. Crea archivo Slurm con configuración de recursos. d. Envía trabajo con comando al clúster. e. Monitorea estado de ejecución. f. Revisa resultados cuando termina.
Validaciones/Restricciones	a. Límites de tiempo: 6h invitados, 7 días investigadores. b. Recursos solicitados dentro de límites del clúster. c. Formato correcto del script Slurm.

TABLA 4.2: Caso de uso - Envío y ejecución de trabajos de cómputo.

4.2.5. Acceso web al clúster

Nombre	**3 - Acceso web al clúster**
Caso de Uso General	Uso del clúster mediante interfaz web para facilitar el trabajo a usuarios menos técnicos.
Descripción	Open OnDemand proporciona una interfaz web que permite trabajar con el clúster sin necesidad de conocer comandos de Linux. Los usuarios pueden navegar por archivos, editar código, enviar trabajos y ver resultados desde el navegador web. Es especialmente útil para usuarios nuevos o que prefieren interfaces gráficas.
Actor Principal	Usuario Investigador/Invitado
Actor Secundario	Ninguno.
Precondiciones	Credenciales válidas en FreeIPA. Open OnDemand configurado y funcionando. Navegador web moderno.
Postcondiciones	a. Acceso web establecido exitosamente. b. Archivos gestionados desde navegador. c. Scripts editados y guardados. d. Trabajos enviados y monitoreados vía web.
Secuencias de Acciones	**Usuario:** a. Navega a la URL de Open OnDemand. b. Inicia sesión con credenciales de FreeIPA. c. Usa explorador de archivos web. d. Edita código con editor integrado. e. Envía trabajos desde interfaz web. f. Monitorea estado de trabajos activos.
Validaciones/Restricciones	a. Autenticación exitosa via FreeIPA. b. Permisos web consistentes con sistema. c. Funcionalidades limitadas a las habilitadas.

TABLA 4.3: Caso de uso - Acceso web al clúster.

4.2.6. Administración del sistema

Nombre	**4 - Administración del sistema**
Caso de Uso General	Gestión y mantenimiento del clúster por parte del administrador.
Descripción	El administrador se encarga de todas las tareas de gestión del clúster: crear usuarios, configurar políticas, instalar *software*, monitorear el rendimiento y resolver problemas. Utiliza varias herramientas web como FreeIPA para administrar los usuarios, Ganglia para monitoreo y *EasyBuild* para gestión de *software* científico.
Actor Principal	Administrador del Sistema
Actor Secundario	Ninguno.
Precondiciones	Credenciales de administrador. Herramientas de gestión funcionando. Acceso a interfaces web administrativas.
Postcondiciones	a. Usuarios y grupos gestionados correctamente. b. Políticas y cuotas configuradas. c. *Software* científico instalado y disponible. d. Sistema monitoreado y funcionando bien.
Secuencias de Acciones	**Administrador:** a. Accede a FreeIPA para gestionar usuarios. b. Crea cuentas y asigna a grupos (investigadores/invitados). c. Configura cuotas de disco y límites. d. Instala *software* con *EasyBuild* según necesidades. e. Supervisa clúster con Ganglia y Cockpit. f. Resuelve incidencias y mantiene sistema.
Validaciones/Restricciones	a. Solo administradores pueden realizar estas tareas. b. Cambios deben seguir políticas de seguridad. c. *Software* instalado debe ser compatible.

TABLA 4.4: Caso de uso - Administración del sistema.

4.3. Diseño general del clúster HPC

La arquitectura implementada se basa en un clúster HPC virtualizado mediante VMware Workstation actualmente gratuita (VMware, Inc., 2025), instalado sobre una única máquina física. Este enfoque permite simular un entorno completo de computación de altas prestaciones sin necesidad de requerir múltiples equipos físicos. La máquina utilizada cuenta con los recursos necesarios para mantener operativos todos los nodos virtuales, tal como se muestra en la Tabla 4.5 y los recursos asignados

a las máquinas virtuales en la Tabla 4.6.

El clúster está compuesto por un nodo maestro y tres nodos de cómputo, interconectados a través de una red virtual configurada en modo de traducción de direcciones de red (NAT, *Network Address Translation*) (Broadcom Inc., 2024), como se ilustra en la Figura 4.2. El nodo maestro coordina el funcionamiento general del sistema, ya que gestiona los recursos, programa las tareas y ofrece servicios de monitorización. Además, integra Open OnDemand, lo que permite el acceso al sistema a través de una interfaz web (Ohio Supercomputer Center, 2025).

Los sistemas de archivos compartidos se distribuyen mediante NFS, siendo montados desde el nodo maestro y accesibles desde los tres nodos de cómputo (nodo01, nodo02 y nodo03). Las particiones asignadas a cada nodo se detallan en la tabla 4.7.

Es importante señalar que la instalación y configuración del clúster se llevó a cabo de forma manual, sin hacer uso de herramientas automatizadas ni soluciones integradas, la guía se puede ver en la Sección 4.7. La elección de este enfoque permitió mantener un control detallado sobre la instalación y favorecer la comprensión del proceso, lo cual resulta especialmente valioso con fines formativos y de validación técnica.

4.4. Arquitectura hardware del clúster HPC

El clúster está formado por un total de 4 servidores virtuales, todos ellos alojados en una misma máquina física. Mediante el uso de la virtualización, se logra un mejor aprovechamiento de los recursos disponibles para permitir su implementación en el entorno de prueba. En la Subsección 4.4.1 se detallan las especificaciones de la máquina física que actúa como anfitrión, mientras que en la Subsección 4.4.2 se describen los recursos asignados a cada uno de los nodos virtuales que componen el clúster.

4.4.1. Máquina Física Host

La máquina física que actúa como anfitriona del clúster está equipada con los siguientes componentes:

Componente	Especificación
Procesador	Intel Core i7-13700H (13ª Gen, 2.40 GHz)
Memoria RAM	32 GB SODIMM (5200 MT/s)
Almacenamiento principal	SSD NVMe WD PC SN560 (954 GB)
Sistema Operativo anfitrión	Windows 11 Pro

TABLA 4.5: Especificaciones técnicas de la máquina física anfitriona.

4.4.2. Recursos asignados a los nodos virtuales

Cada uno de los nodos virtuales que conforman el clúster dispone de los siguientes recursos asignados:

Características	Master	Nodo01	Nodo02	Nodo03
CPU (vCPUs)	8	4	4	4
Memoria RAM (GB)	6	4	4	4
Almacenamiento SSD (GB)	100	100	100	100
Discos secundarios SSD (GB)	300	-	-	-

TABLA 4.6: Resumen de características de *hardware* virtualizado por nodo en el clúster.

4.5. Topología de red y esquema de interconexión

La Figura 4.2 ilustra la topología de red implementada en el clúster HPC. El entorno virtual está conformado por un nodo maestro y tres nodos de cómputo (nodo01, nodo02 y nodo03), todos alojados en una misma máquina física. La interconexión entre los nodos se realiza mediante una red virtual en modo NAT, configurada dentro del entorno de VMware Workstation.

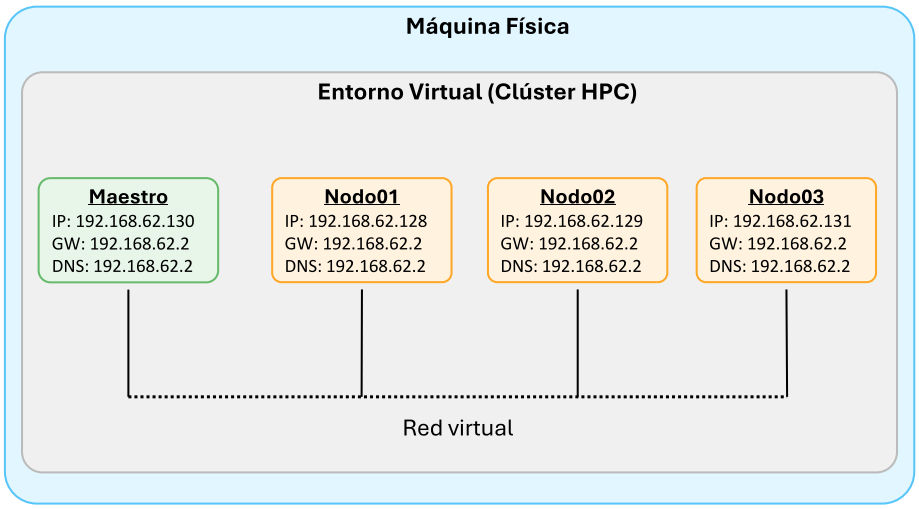

FIGURA 4.2: Esquema de interconexión de red entre los nodos virtuales del clúster. Fuente: elaboración propia

4.6. Arquitectura software del clúster HPC

La arquitectura *software* del clúster ha sido diseñada para asegurar un entorno de trabajo estable y eficiente. En esta sección se detallan las herramientas y configuraciones utilizadas, seleccionadas por su fiabilidad y su amplia adopción en entornos de computación de altas prestaciones. Algunas de estas soluciones también aparecen, en mayor o menor medida, en trabajos previos con enfoques similares, como los desarrollados por Fernández Palau (Fernández Palau, 2023), Guillén Civera (Civera, 2018) y García Peña (García Peña et al., 2021), lo que refuerza tanto la validez técnica de las decisiones adoptadas como la relevancia práctica en escenarios reales de computación de alto rendimiento.

4.6.1. Sistema operativo

El SO seleccionado para la instalación del clúster es Rocky Linux 8.10, principalmente porque, según la lista *TOP500* de noviembre de 2024, Red Hat y sus derivados son ampliamente utilizados en entornos HPC. Esta versión cuenta con soporte de seguridad hasta el 31 de mayo de 2029 (Rocky Linux, 2025a) y ofrece la posibilidad de contratar soporte comercial a través de proveedores autorizados (Rocky Linux, 2025b).

4.6.2. Particiones de disco y directorios compartidos

En la Tabla 4.7 se detallan las particiones de disco configuradas en el nodo maestro y en los nodos de cómputo. El nodo maestro incluye particiones adicionales para compartir directorios mediante NFS, facilitando así el acceso remoto y centralizado a los recursos desde los nodos de cómputo.

Punto de montaje	Tamaño	Comentario
Nodo maestro		
/ (root)	91 GB	Partición base del sistema operativo.
/boot	1 GB	Partición que contiene los archivos necesarios para el arranque del sistema.
swap	8 GB	Espacio de intercambio o memoria virtual.
/home	100 GB	Directorios personales de los usuarios, compartidos con los nodos a través de NFS para centralizar los datos.
/usr/local	200 GB	Directorio compartido de aplicaciones, usado para gestionar distintas versiones mediante módulos y accesible desde los nodos vía NFS.
Nodos de cómputo (01–03)		
/ (root)	91 GB	Partición base del sistema operativo.
/boot	1 GB	Partición que contiene los archivos necesarios para el arranque del sistema.
swap	8 GB	Espacio de intercambio o memoria virtual.

TABLA 4.7: Esquema de particiones con comentarios descriptivos. Los tamaños indicados son nominales y aproximados.

4.6.3. Sistemas de gestión de recursos y planificación de trabajos en entornos HPC

Para gestionar los recursos y planificar los trabajos en el entorno HPC, se ha optado por utilizar (Slurm, *Simple Linux Utility for Resource Management*). Según el informe de Hyperion Research (Research, 2023), Slurm es el planificador más utilizado según su estudio. En la Figura 4.3 se muestra un esquema de su arquitectura, junto con algunos de los comandos básicos que ofrece.

En este caso se utiliza una política de planificación sencilla. En ella, los trabajos se ejecutan en el orden en que se envían, esta estrategia se denomina (FIFO,*First In, First Out*), y esta se combina con una técnica de *backfilling* que permite adelantar la ejecución de trabajos más pequeños si hay recursos disponibles, siempre que no retrasen la ejecución de los trabajos que llegaron antes. Ambas opciones forman parte de las configuraciones disponibles de Slurm y se describen en su documentación oficial (SchedMD, 2024a).

En este entorno, se han definido dos grupos de trabajo, uno para el grupo de usuarios invitados, con un límite máximo de ejecución de seis horas, y otro para investigadores, que permite trabajos de hasta siete días. Esta separación facilita la gestión del clúster según el perfil y las necesidades de cada grupo.

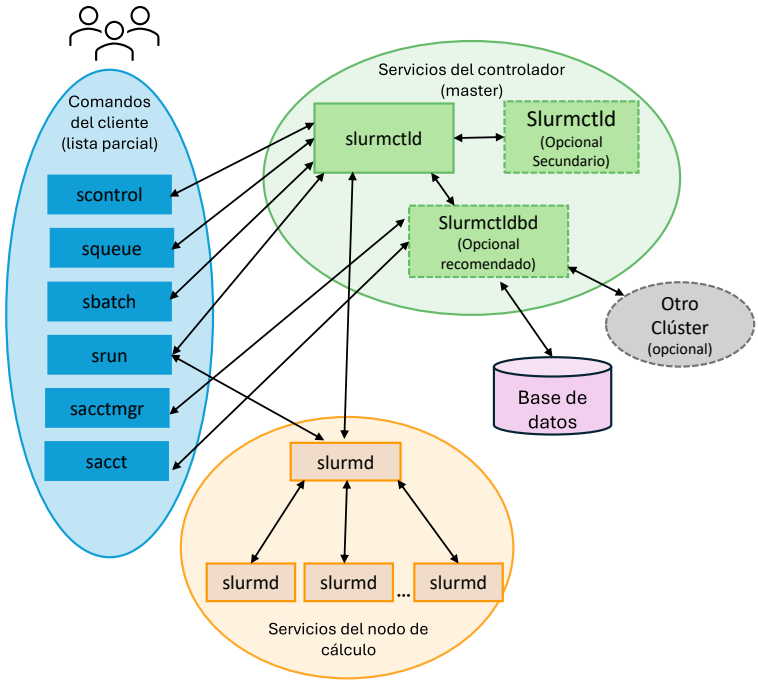

FIGURA 4.3: Componentes principales del sistema de planificación Slurm. Adaptado de (SchedMD, 2024b).

4.6.4. Gestión centralizada de identidades

Entre las distintas alternativas evaluadas, se optó por utilizar FreeIPA como solución. Esta herramienta de código abierto integra servicios como LDAP, Kerberos, DNS y certificados digitales, lo que permite una autenticación segura y homogénea en cada uno de los nodos que conforman el HPC. Además, ofrece una administración sencilla tanto por línea de comandos como desde una interfaz web.

4.6.5. Gestión de software en el HPC

En el despliegue del clúster, se seleccionó *EasyBuild* como herramienta para la instalación y gestión del *software* científico. Esta decisión se tomó debido a la necesidad de contar con un sistema que permita mantener múltiples versiones de aplicaciones sin conflictos en el entorno HPC. *EasyBuild* automatiza la compilación, la gestión de dependencias y la creación de módulos de entorno, lo que facilita considerablemente la administración del sistema. Además, al permitir la instalación en directorios

compartidos, evita replicar configuraciones en cada nodo, simplificando el mantenimiento (Fernández Palau, 2023; Geimer et al., 2014). Su integración con Lmod que es un sistema de gestión de módulos de entorno, este permite modificar dinámicamente el entorno de ejecución a los usuarios (McLay, 2024).

4.6.6. Monitoreo y visualización del rendimiento

Para la monitorización del clúster, se seleccionó Ganglia debido a su capacidad de adaptación a entornos HPC distribuidos. Su arquitectura permite recopilar métricas de múltiples nodos, manteniendo una sobrecarga mínima en CPU, memoria y ancho de banda (Massie et al., 2004). Además, ofrece una interfaz web intuitiva y visualización de los históricos, que facilita el seguimiento de indicadores como la carga de CPU, el uso de memoria y la actividad de red. En entornos científicos, Ganglia ha demostrado ser una herramienta estable y confiable, como lo demuestra su implementación en entornos HPC reales (Ghedira et al., 2024).

4.7. Implementación del clúster

En esta sección se presenta una guía paso a paso para la implementación del clúster HPC, organizada en tres partes. En primer lugar, se describen los comandos y acciones que deben ejecutarse en todos los nodos, como se indica en la Sección 4.7.1. Posteriormente, se detallan las tareas específicas del nodo maestro, disponibles en la Sección 4.7.2. Finalmente, se explican las configuraciones requeridas en los nodos de cómputo, que se encuentran en la Sección 4.7.3. La guía sigue prácticas comunes en entornos HPC, por lo que algunos comandos pueden coincidir con otros documentos. Aún así, su estructura y adaptación reflejan una experiencia propia y pensada para ser útil en un contexto real.

La elaboración de esta guía se basó en el análisis de diversas fuentes prácticas y técnicas, entre ellas los tutoriales de ServerWorld (Server-World, 2025), así como la documentación oficial de Slurm (SchedMD LLC, 2025), FreeIPA (FreeIPA Project, 2025b), Ganglia (Ganglia Project, 2025) y Munge (Chris Dunlap, 2020), entre otros manuales oficiales. La Sección 4.6 recopila la bibliografía de referencia utilizada como base metodológica y detalla implementaciones análogas. Finalmente, para complementar esta guía, se dispone de un manual de administración técnica, incluido en el Anexo B.

4.7.1. Configuración común en todos los nodos

La implementación del clúster requiere que todos los nodos compartan una configuración inicial común, independientemente de su función específica dentro de la arquitectura.

Instalación de los hosts

En principio, al realizar la instalación del SO Rocky Linux 8.10, se recomienda elegir la opción de instalación mínima. De esta forma, se evita incluir paquetes innecesarios que pueden consumir recursos y afectar el rendimiento del sistema. Además, se particionan los discos como se especifica en la sección 4.7.

Agregar repositorios

A continuación, se habilitan los repositorios de los paquetes extra para Linux empresarial (EPEL, *Extra Packages for Enterprise Linux*) y `powertools`, ya que contienen paquetes complementarios y bibliotecas de desarrollo requeridas para la instalación del *software* necesario para el clúster (Rocky Linux Documentation Team, 2024).

```
[root@host ~]$ dnf install --assumeyes epel-release
[root@host ~]$ dnf config-manager --set-enabled powertools
```

Instalar paquetes necesarios

Resulta importante indicar la conveniencia de habilitar el servicio Cockpit, ya que facilita la administración del servidor mediante una interfaz web. Asimismo, se instalan componentes adicionales necesarios para el entorno de trabajo.

```
[root@host ~]$ dnf install python3 python3.12 python3.12-pip.noarch  R tmux cockpit
python3.11-pip.noarch -y
[root@host ~]$ systemctl enable cockpit.socket --now
```

Servidor DNS local

En este apartado, se indica que para la resolución de nombres en el clúster se emplea un esquema local mediante la edición del archivo `/etc/hosts`. Esta configuración permite identificar cada nodo por su nombre dentro de la red interna, sin necesidad de desplegar un servidor DNS dedicado.

```
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.62.130 master.hpc.tfm master ood.hpc.tfm ood
192.168.62.128 nodo01.hpc.tfm nodo01
192.168.62.129 nodo02.hpc.tfm nodo02
192.168.62.131 nodo03.hpc.tfm nodo03
```

SELinux

Luego, se procede a deshabilitar SELinux mediante la modificación del archivo de configuración y el posterior reinicio del sistema, con el fin de evitar posibles conflictos con esta capa de seguridad en el entorno del clúster. Como contramedida, la seguridad se sustenta en firewalld por nodo, autenticación centralizada mediante FreeIPA y cifrado de la administración vía SSH y HTTPS.

```
[root@host ~]$ sed -i 's/^SELINUX=.*/SELINUX=disabled/' /etc/selinux/config
[root@host ~]$ reboot
[root@host ~]$ sestatus
SELinux status:                 disabled
```

NTP

La sincronización horaria es configurada en todas las máquinas del sistema mediante un servicio basado en el protocolo de tiempo de red (NTP, *Network Time Protocol*) (The NTP Project, 2024) , lo cual es esencial para el correcto funcionamiento de múltiples servicios dentro del clúster.

```
[root@host ~]$ dnf install chrony ntpstat -y
[root@host ~]$ systemctl enable --now chronyd
[root@host ~]$ chronyc sources
[root@host ~]$ chronyc -a makestep
```

Usuarios del sistema para servicios Munge, Slurm y EasyBuilder

Los usuarios del SO, Munge, EasyBuilder y Slurm son creados con identificadores fijos para garantizar la coherencia en entornos distribuidos. En caso de Munge es un servicio de autenticación diseñado para entornos HPC que permite validar credenciales entre nodos de forma segura, utiliza una clave compartida para definir un dominio de confianza entre *hosts* con usuarios y grupos comunes (The MUNGE Project, 2025).

```
[root@host ~]$ groupadd -g 1009 munge # UID específico para todos los nodos
[root@host ~]$ useradd -m -c "MUNGE Auth Service" -d /var/lib/munge -u 1009
-g munge -s /sbin/nologin munge
[root@host ~]$ groupadd -g 1010 slurm # UID específico para todos los nodos
[root@host ~]$ useradd -m -c "Slurm Workload Manager" -d /var/lib/slurm -u 1010
-g slurm -s /bin/bash slurm
[root@host ~]$ groupadd -g 1011 easybuilder # UID específico para todos los nodos
[root@host ~]$ useradd -m  easybuilder -u 1011 -g easybuilder
```

Tmux

La herramienta *Tmux* permite gestionar múltiples sesiones en una terminal y dejarlas en segundo plano y retomarlas más tarde, incluso desde otra terminal si es necesario (tmux, s.f.).

Posteriormente, se lleva a cabo el proceso para habilitar la integración con el *mouse*. Para ello, se debe editar el archivo `/etc/tmux.conf` y agregar lo siguiente.

```
set -g mouse on
```

4.7.2. Configuración en el nodo maestro

Una vez completadas las configuraciones comunes en todos los nodos que se especifican en la Sección 4.7, se procede a definir los ajustes específicos del nodo maestro.

Servidor NFS

El servidor NFS es instalado con el fin de compartir directorios entre los nodos del clúster, permitiendo el acceso centralizado tanto a los directorios personales de los usuarios como a un directorio común para aplicaciones.

```
[root@host ~]$ dnf install nfs-utils -y
```

Luego, se edita la configuración del servidor NFS en el archivo `/etc/idmapd.conf`, actualizando el valor del dominio en la línea correspondiente.

```
Domain = hpc.tfm
```

Posteriormente, se procede a configurar la exportación de los directorios `/home` y `/usr/local` desde el nodo maestro, tal como se detalla en la Tabla 4.7, con el fin de compartir los directorios personales de los usuarios y las aplicaciones mediante NFS.

```
/home 192.168.62.0/24(rw,no_root_squash)
/usr/local/ 192.168.62.0/24(rw,no_root_squash)
```

Finalmente, se procede a habilitar el servicio NFS y se configuran de manera permanente los puertos que requieren el servicio en el cortafuegos. Luego, se recargan las reglas para aplicar los cambios y permitir el intercambio de archivos entre nodos.

```
[root@master ~]$ systemctl enable --now rpcbind nfs-server
[root@master ~]$ firewall-cmd --add-service=nfs --permanent
[root@master ~]$ firewall-cmd --add-service={nfs3,mountd,rpc-bind} --permanent
[root@master ~]$ firewall-cmd --reload
```

Servidor FreeIPA

En este apartado se describen los detalles relacionados con la instalación y configuración del servidor FreeIPA, así como la creación del dominio `hpc.tfm`, utilizando todos los parámetros por defecto.

```
[root@master ~]$ dnf module install idm:DL1/server -y
[root@master ~]$ ipa-server-install
```

En primer lugar, se habilitan los puertos en el cortafuegos.

```
[root@master ~]$ firewall-cmd --add-service={freeipa-ldap,freeipa-ldaps,
kerberos,ntp} --permanent
[root@master ~]$ firewall-cmd --reload
```

A continuación, se procede a personalizar los parámetros del servidor FreeIPA.

```
[root@master ~]$ kinit admin
[root@master ~]$ ipa config-mod --defaultshell=/bin/bash
[root@master ~]$ authselect enable-feature with-mkhomedir
[root@master ~]$ systemctl enable --now oddjobd
```

Finalmente, se procede a la creación de grupos en el servidor de identidades, con el objetivo de establecer políticas de uso diferenciadas. Esta organización permitirá, posteriormente, definir las políticas de uso en el planificador de trabajos.

```
[root@master ~]# kinit admin
Password for admin@HPC.TFM:
[root@master ~]# ipa group-add invitados
-----------------------
Added group "invitados"
-----------------------
  Group name: invitados
  GID: 925400006
[root@master ~]# ipa group-add investigadores
---------------------------
Added group "investigadores"
---------------------------
  Group name: investigadores
  GID: 925400007
[root@master ~]#
```

Resulta importante indicar que es posible gestionar FreeIPA mediante una interfaz web que se puede acceder desde el navegador; este facilita la administración de usuarios, grupos y políticas. La Figura 4.4 muestra una vista general de dicha interfaz, disponible en https://master.hpc.tfm/ipa/ui/.

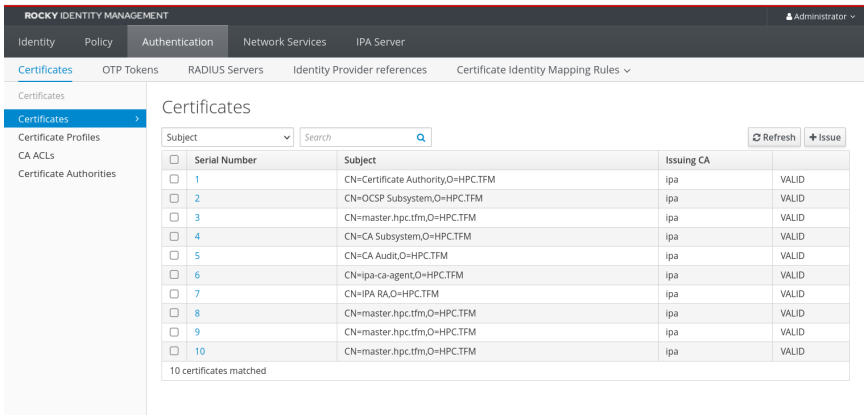

FIGURA 4.4: Interfaz de administración de FreeIPA.

Munge

En este apartado se indican los detalles acerca de la instalación de Munge y generación de la clave de autenticación utilizando una fuente segura de aleatoriedad.

```
[root@master ~]$ dnf install munge munge-libs munge-devel rng-tools -y
[root@master ~]$ rngd -r /dev/urandom
[root@master ~]$ /usr/sbin/create-munge-key -r -f
```

Luego, se crea una clave segura para Munge, se ajustan los permisos y se inicia el servicio.

```
[root@master ~]$ sh -c "dd if=/dev/urandom bs=1 count=1024 > /etc/munge/munge.key"
[root@master ~]$ chown munge:munge /etc/munge/munge.key
[root@master ~]$ chmod 400 /etc/munge/munge.key
[root@master ~]$ systemctl enable --now munge
```

A continuación, se realizan las pruebas de correcta operativa del Munge.

```
[root@master ~]$ munge -n
[root@master ~]$ munge -n | munge
[root@master ~]$ munge -n | ssh master unmunge
[root@master ~]$ remunge
```

Slurm

En este apartado se indican los detalles acerca de la instalación de los paquetes de base de Slurm para la gestión de colas de trabajos.

```
[root@master ~]$ dnf install slurm slurm-slurmctld slurm-devel -y
```

En primer lugar, se procede a instalar los paquetes adicionales de Slurm para alcanzar una funcionalidad completa.

```
[root@master ~]$ dnf install slurm-perlapi slurm-torque slurm-contribs -y
```

A continuación, se procede a la instalación de la documentación oficial de Slurm y de su herramienta gráfica, mediante el siguiente comando:

```
[root@master ~]$ dnf install slurm-doc slurm-gui -y
```

Como paso siguiente, se procede a abrir los puertos necesarios que son requeridos por el servicio.

```
[root@master ~]$ firewall-cmd --permanent --add-port=6817-6819/tcp
[root@master ~]$ firewall-cmd --permanent --add-port=6817-6819/udp
[root@master ~]$ firewall-cmd --permanent --add-port=60001-63000/tcp
[root@master ~]$ firewall-cmd --permanent --add-rich-rule='rule family="ipv4"
source address="192.168.62.131" accept'
[root@master ~]$ firewall-cmd --permanent --add-rich-rule='rule family="ipv4"
source address="192.168.62.129" accept'
[root@master ~]$ firewall-cmd --permanent --add-rich-rule='rule family="ipv4"
source address="192.168.62.128" accept'
[root@master ~]$ firewall-cmd --reload
```

Resulta importante destacar que en el sitio web oficial de Slurm se encuentra disponible un asistente interactivo para la generación del archivo de configuración, accesible en el siguiente enlace: https://slurm.schedmd.com/configurator.html. La Figura 4.5 muestra este generador, que simplifica el proceso y el *script* listo para implementar directamente en el servidor.

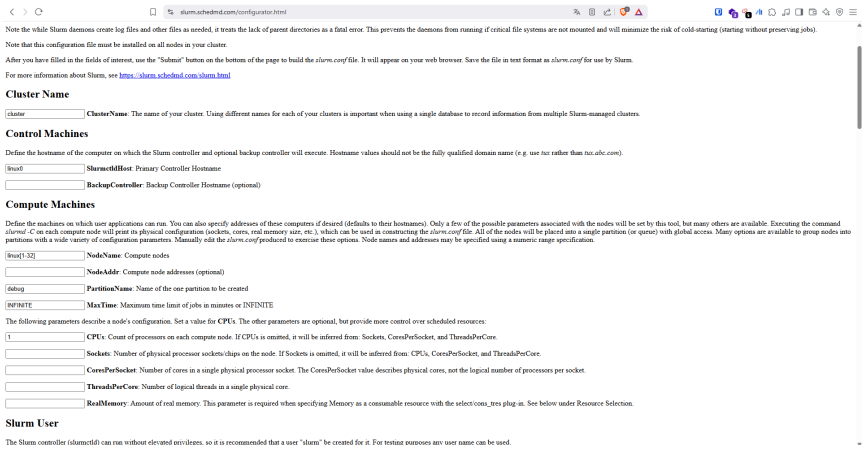

FIGURA 4.5: Asistente web de Slurm para la generación del archivo de configuración.

Posteriormente, se procede a realizar la configuración del Slurm mediante la edición del archivo `/etc/slurm/slurm.conf`, donde se definen los parámetros principales del clúster.

```
ClusterName=hpc.tfm
SlurmctldHost=master.hpc.tfm
MpiDefault=pmi2
ProctrackType=proctrack/cgroup
ReturnToService=1
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=slurm
StateSaveLocation=/var/spool/slurmctld
SwitchType=switch/none
TaskPlugin=task/affinity
InactiveLimit=0
KillWait=30
MinJobAge=300
SlurmctldTimeout=120
SlurmdTimeout=300
Waittime=0
SchedulerType=sched/backfill
SelectType=select/cons_tres
SelectTypeParameters=CR_Core
AccountingStorageType=accounting_storage/none
JobCompType=jobcomp/none
JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
SlurmctldDebug=info
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdDebug=info
SlurmdLogFile=/var/log/slurm/slurmd.log
SlurmctldParameters=slurmctld_port_range=60001-63000
SlurmdParameters=slurmd_port_range=60001-63000
NodeName=nodo[01-03] CPUs=4 RealMemory=4096 Sockets=1 CoresPerSocket=4
ThreadsPerCore=1 State=UNKNOWN
# Partición para usuarios invitados (máximo 6 horas)
PartitionName=invitados Nodes=nodo[01-03] MaxTime=06:00:00
AllowGroups=invitados State=UP
```

```
# Partición para investigadores (máximo 7 días)
PartitionName=investigadores Nodes=nodo[01-03] MaxTime=7-00:00:00
AllowGroups=investigadores State=UP
```

Luego, se crean los directorios necesarios para el funcionamiento de Slurm y se asignan los permisos adecuados al usuario del servicio.

```
[root@master ~]$ mkdir -p /var/spool/slurm/ctld /var/spool/slurmctld /var/log/slurm
/var/spool/slurmd
[root@master ~]$ chown -R slurm:slurm /var/spool/slurm /var/spool/slurmctld
/var/log/slurm /var/spool/slurmd /var/log/slurm
```

Posteriormente, se procede a modificar parámetros del servicio Slurmd.

```
[root@master ~]$ systemctl edit --full slurmctld.service
```

A continuación, se deben agregar los siguientes parámetros luego de Type=simple.

```
ExecStartPre=/bin/sleep 30
User=slurm
```

En este punto, una vez iniciado el servicio, puede comprobarse su correcto funcionamiento mediante el siguiente comando, cuya salida debería ser similar a la que se muestra a continuación.

```
[root@master ~]$ systemctl enable slurmctld --now
[root@master ~]$ systemctl status slurmctld
slurmctld.service - Slurm controller daemon
   Loaded: loaded (/usr/lib/systemd/system/slurmctld.service; enabled)
   Active: active (running) since Sun 2025-05-04 16:51:41 CEST; 9min ago
Main PID: 5515 (slurmctld)
    Tasks: 7
   Memory: 3.8M
   CGroup: /system.slice/slurmctld.service
           -515 /usr/sbin/slurmctld -D

May 04 16:51:41 master.hpc.tfm systemd[1]: Started Slurm controller daemon.
```

Finalmente, se establecen todos los nodos como disponibles y luego se muestra su estado actual.

```
[root@master ~]$ scontrol update NodeName=nodo[01-03] State=IDLE
[root@master ~]$ scontrol show nodes
```

Slurm con soporte de histórico

En este apartado se detallan las configuraciones necesarias que permitirán guardar el histórico de ejecuciones de los trabajos. En primer lugar, es necesario instalar y habilitar la base de datos relacional MariaDB (MariaDB Foundation, s.f.) junto con los componentes necesarios de Slurm.

```
[root@master ~]$ dnf install mariadb-server slurm-slurmdbd -y
[root@master ~]$ systemctl enable --now slurmdbd
[root@master ~]$ systemctl enable --now mariadb
[root@master ~]$ mysql_secure_installation
```

En primer lugar, se procede a verificar la configuración principal del clúster para establecer la conexión con la base de datos. Esta configuración se define en el archivo `/etc/slurm/slurmdbd.conf`, el cual debe ser editado para ajustar los parámetros necesarios.

```
AuthType=auth/munge
DebugLevel=4
DbdAddr=localhost
DbdHost=localhost
DbdPort=6819
LogFile=/var/log/slurm/slurmdbd.log
PidFile=/var/run/slurm/slurmdbd.pid
PurgeEventAfter=1month
PurgeJobAfter=1month
PurgeResvAfter=1month
PurgeStepAfter=1month
PurgeSuspendAfter=1month
PurgeTXNAfter=1month
PurgeUsageAfter=1month
SlurmUser=slurm
StorageType=accounting_storage/mysql
StorageHost=localhost
StoragePass= **password**
StorageUser=slurm
StorageLoc=slurm_acct_db
```

A continuación, debe actualizarse la configuración de Slurm para guardar histórico en `/etc/slurm/slurm.conf`.

```
AccountingStorageType=accounting_storage/slurmdbd
AccountingStorageHost=localhost
AccountingStoragePort=6819
JobAcctGatherType=jobacct_gather/linux
JobAcctGatherFrequency=30
```

Finalmente, se procede a inicializar y probar el registro histórico de trabajos en Slurm.

```
[root@master ~]$ chown slurm:slurm /etc/slurm/slurmdbd.conf
[root@master ~] chmod 600 /etc/slurm/slurmdbd.conf
[root@master ~]$ systemctl restart slurmctld
[root@master ~]$ sacct
```

Slurm con soporte de notificaciones de correo

En este apartado se indica el procedimiento desarrollado para la configuración del msmtp (Debian Wiki contributors, 2025) como cliente del protocolo simple de transferencia de mensajes (SMTP, `Simple Mail Transfer Protocol`) (Butterfield & Ngondi, 2016) en el nodo maestro, integrándolo con Slurm para permitir el envío automático de correos desde los scripts de trabajo.

```
[root@master ~]$ dnf install msmtp mailx -y
[root@master ~]$ su - slurm
[root@master ~]$ chmod 600 ~/.msmtprc
```

En primer lugar, se edita el archivo de configuración de msmtprc, ubicado en el directorio `/var/lib/slurm/.msmtprc` .

```
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-bundle.crt
logfile        ~/.msmtp.log

account        gmail
host           smtp.gmail.com
port           587
from           lab@gmail.com
user           lab@gmail.com
password       **password**

account default : gmail
```

A continuación, con el fin de asegurar el correcto funcionamiento, se realizan las pruebas de envío de correo.

```
[root@master ~] echo "Correo de prueba desde  msmtp" | mail -s
"Prueba SLURM" tucorreo@gmail.com
```

Finalmente, en la Figura 4.6 se muestran ejemplos de notificaciones automáticas generadas por Slurm tras el envío de un trabajo, indicando su inicio y finalización exitosa tras 30 segundos.

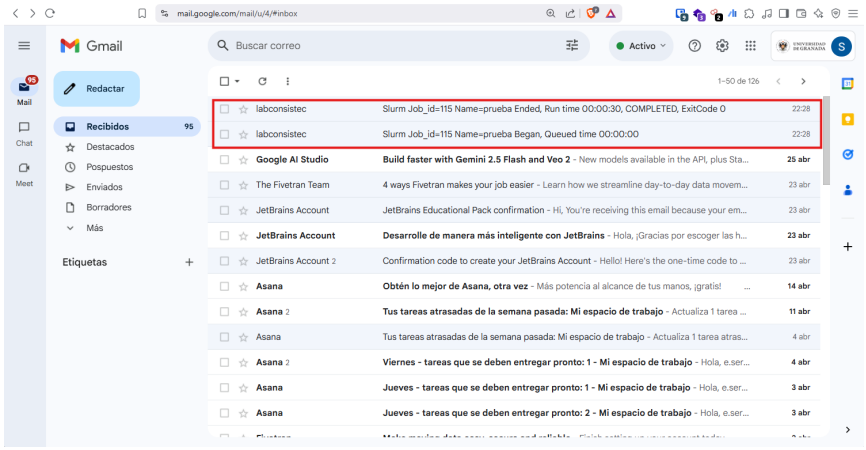

FIGURA 4.6: Notificaciones por correo electrónico generadas automáticamente por Slurm tras el envío y ejecución del trabajo

Interfaz gráfica remota

En este apartado se muestra el procedimiento de instalación del servidor gráfico virtual, lo cual resulta bastante beneficioso, ya que permite tener acceso a ciertas herramientas que serán de gran utilidad en la gestión del clúster.

En primer lugar, se procede a editar el archivo de configuración `/etc/ssh/sshd_config` y descomentar las siguientes líneas para habilitar el reenvío del entorno gráfico .

```
X11Forwarding yes
X11UseLocalhost yes
```

En este punto resulta importante reiniciar el servicio para que la nueva configuración de SSH tenga efecto.

```
[root@host ~]$ systemctl restart sshd
```

Posteriormente, se instalan los paquetes necesarios y se procede a reiniciar el entorno gráfico virtual para aplicar la configuración.

```
[root@host ~]$ yum install xorg-x11-server-Xvfb slurm-gui -y
[root@host ~]$ xvfb :1 -screen 0 1024x768x16 &
[root@host ~]$ export DISPLAY=:1
```

Finalmente, se realiza la prueba de la herramienta gráfica de Slurm iniciando sview.

```
[root@host ~]$  sview
```

La herramienta sview proporciona un panel gráfico para administrar el clúster. Cada usuario visualiza solo los recursos y funciones según sus permisos. La figura 4.7 muestra la vista con privilegios de administrador.

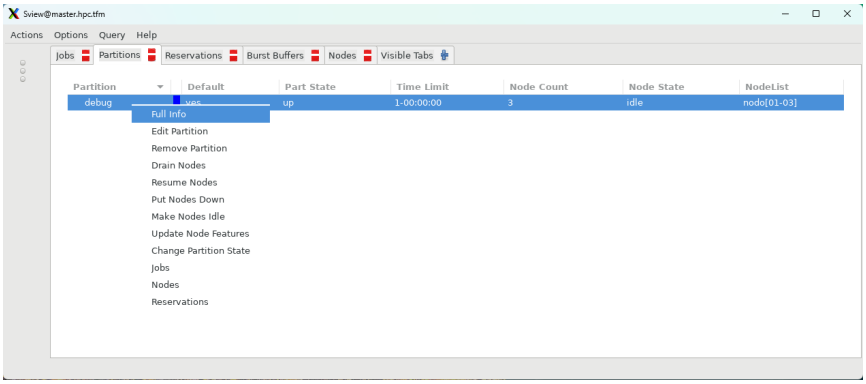

FIGURA 4.7: Interfaz principal de sview, mostrando las particiones del clúster y las opciones de gestión disponibles.

Servidor de monitoreo Ganglia

En este apartado se indica cómo se debe realizar la instalación del servidor de monitoreo Ganglia.

```
[root@master ~]$ dnf install ganglia rrdtool ganglia-gmetad ganglia-gmond
ganglia-web -y
```

En primer lugar, se procede a realizar la habilitación del acceso externo mediante la edición del archivo de configuración de Apache, el cual se encuentra ubicado en `/etc/httpd/conf.d/ganglia.conf`.

```
Alias /ganglia /usr/share/ganglia

<Location /ganglia>
  Require all granted
</Location>
```

A continuación, se procede a la edición del archivo de configuración Ganglia `/etc/ganglia/gmetad.conf`.

```
data_source "Labs" 60 192.168.62.130:8649
gridname "Labs"
```

En este punto, se procede a editar el archivo de configuración de Ganglia, ubicado en el directorio `/etc/ganglia/gmond.conf`.

```
cluster {
  name = "Labs"
  owner = "unspecified"
  latlong = "unspecified"
  url = "unspecified"
}
udp_send_channel {
#  mcast_join = 239.2.11.71 # comentar
  host = 192.168.62.130
  port = 8649
  ttl = 1
}
```

```
udp_recv_channel {
  #mcast_join = 239.2.11.71 # comentar
  port = 8649
  #bind = 239.2.11.71 # comentar
  retry_bind = true
}
```

Como paso siguiente, se procede a abrir los puertos necesarios en el cortafuegos y reiniciar los servicios para aplicar los cambios.

```
[root@host ~]$ firewall-cmd --add-port=8649/udp --permanent
[root@host ~]$ firewall-cmd --add-port=8649/tcp --permanent
[root@host ~]$ firewall-cmd --add-port=9443/tcp --permanent
[root@host ~]$ firewall-cmd --reload
[root@host ~]$ systemctl restart httpd gmetad gmond
[root@host ~]$ systemctl enable httpd gmetad gmond httpd
```

Finalmente, resulta importante destacar que Ganglia ofrece un panel web donde se puede monitorizar el estado general del clúster. Se puede acceder desde el navegador a través de la dirección https://master.hpc.tfm/ganglia/. En la Figura 4.8 se muestra cómo se ve la interfaz con acceso.

FIGURA 4.8: Interfaz principal de Ganglia.

Servidor Open OnDemand

En este apartado se detalla el proceso de instalación del servidor de gestión Open OnDemand, el cual es utilizado para proporcionar acceso web a los recursos del clúster y facilitar la ejecución remota de aplicaciones.

```
[root@master ~]$ dnf install ganglia-gmond.x86_64 -y
[root@master ~]$ dnf module enable ruby:3.3 nodejs:20
[root@master ~]$ dnf install
https://yum.osc.edu/ondemand/4.0/ondemand-release-web-4.0-1.el8.noarch.rpm
[root@master ~]$ dnf install ondemand
[root@master ~]$ dnf install -y ondemand ondemand-dex mod_ldap
```

En primer lugar, se debe generar un certificado autofirmado para habilitar acceso seguro en *Open OnDemand*.

```
[root@master ~]$ mkdir -p /etc/pki/tls/private
[root@master ~]$ mkdir -p /etc/pki/tls/certs
[root@master ~]$ openssl req -x509 -nodes -days 3650 \
  -newkey rsa:2048 \
  -keyout /etc/pki/tls/private/ood.key \
  -out /etc/pki/tls/certs/ood.crt \
  -subj "/C=ES/ST=YourState/L=YourCity/O=HPC-TFM/OU=IT/CN=ood.hpc.tfm"
```

Luego, se procede a la edición del archivo de configuración ubicado en el directorio `/etc/ood/config/ood_portal.yml` para habilitar el acceso seguro a Open OnDemand y la autenticación LDAP de FreeIPA.

```
servername: 'ood.hpc.tfm'
ssl:
  - SSLCertificateFile /etc/pki/tls/certs/ood.crt
  - SSLCertificateKeyFile /etc/pki/tls/private/ood.key
oidc_uri: "/oidc"
oidc_opts:
  OIDCSSLValidateServer: Off
dex:
  ssl: true
  https_port: "5554"
  client_id: "ood.hpc.tfm"
  client_name: "HPC OnDemand2"
  client_secret: "/etc/ood/dex/ondemand.secret"
  connectors:
    - type: ldap
      id: ldap
      name: "FreeIPA LDAP"
      config:
        host: "master.hpc.tfm:636"
        insecureSkipVerify: true
        bindDN: "uid=admin,cn=users,cn=accounts,dc=hpc,dc=tfm"
        bindPW: "**password**"
        userSearch:
          baseDN: "cn=users,cn=accounts,dc=hpc,dc=tfm"
          filter: "(objectClass=posixAccount)"
          username: uid
          idAttr: uid
          emailAttr: mail
          nameAttr: displayName
          preferredUsernameAttr: uid
        groupSearch:
          baseDN: "cn=groups,cn=accounts,dc=hpc,dc=tfm"
          filter: "(objectClass=posixGroup)"
          userMatchers:
            - userAttr: DN
              groupAttr: member
              nameAttr: cn
auth:
  - 'AuthType openid-connect'
  - 'Require valid-user'
```

A continuación, se procede a configurar los parámetros ubicados en el directorio `/etc/ood/config/clusters.d/my_cluster.yml`.

```
---
v2:
```

```
  metadata:
    title: "HPC.TFM"
  login:
    host: "127.0.0.1"
  job:
    adapter: "slurm"
```

Finalmente, se deben reiniciar los servicios y aplicar la configuración del portal.

```
[root@master ~]$ /opt/ood/ood-portal-generator/sbin/update_ood_portal
[root@master ~]$ systemctl enable --now ondemand-dex.service
[root@master ~]$ systemctl restart httpd
[root@master ~]$ firewall-cmd --add-port=5554/tcp --permanent
[root@master ~]$ firewall-cmd --reload
```

Open OnDemand permite el acceso al entorno HPC a través de una interfaz web, accesible mediante navegador en la dirección https://ood.hpc.tfm/. En la Figura 4.9 se muestra una vista general de la plataforma tras el inicio de sesión.

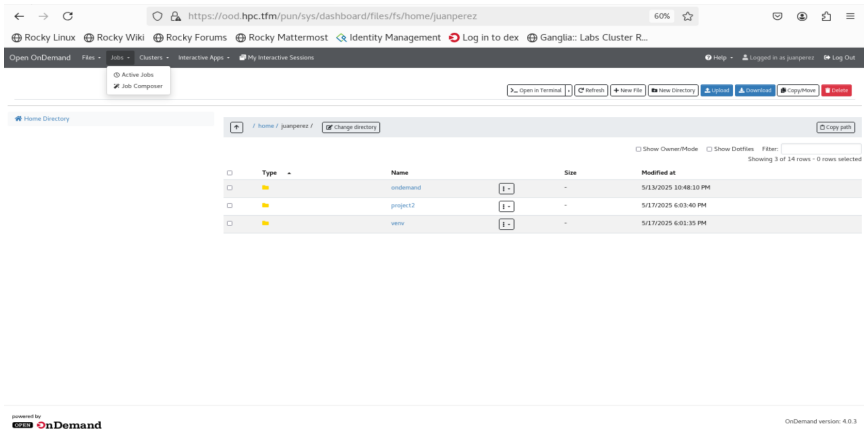

FIGURA 4.9: Vista del directorio personal en Open OnDemand

EasyBuild

En este apartado se indica el procedimiento de instalación de los paquetes necesarios para la utilización de *EasyBuild*.

```
[root@master ~]$ dnf groupinstall -y "Development Tools"
[root@master ~]$ dnf install readline-devel environment-modules tcl-devel -y
[root@master ~]$ dnf install libibverbs libibverbs-devel rdma-core-devel -y
```

A continuación, se ejecuta la instalación de *EasyBuild*.

```
[root@master ~]$ su - easybuilder
[root@master ~]$ mkdir -p /usr/local/easybuild/{sources,builds,modules/all,
containers,packages,ebfiles_repo,easyconfigs}
[root@master ~]$ chown -R easybuilder:easybuilder /usr/local/easybuild
[root@master ~]$ pip3 install --prefix /usr/local easybuild
```

Posteriormente, se deben configurar las variables de entorno necesarias para *EasyBuild* y el sistema de módulos.

```
[root@master ~]$ cp /usr/share/lmod/lmod/init/profile /etc/profile.d/z00_lmod.sh
[root@master ~]$ bash -c 'cat > /etc/profile.d/zzeasybuild.sh <<EOF
export EASYBUILD_PREFIX=/usr/local/easybuild/ &&
export EASYBUILD_SOURCEPATH=/usr/local/easybuild/sources &&
export EASYBUILD_BUILDPATH=/usr/local/easybuild/builds &&
module use /usr/local/easybuild/modules/all
EOF'
```

4.7.3. Configuración en el nodo de cálculo

Una vez completadas las configuraciones comunes en todos los nodos que se especifican en la Sección 4.7, se procede a definir los ajustes específicos de los nodos de cómputo.

Cliente NFS

En este apartado, se indica el proceso de instalación del cliente NFS que permitirá acceder a los directorios compartidos del *Home* del servidor y autofs que permiten que estos se monten automáticamente.

```
[root@host ~]$ dnf install nfs-utils autofs -y
[root@host ~]$ systemctl enable --now autofs
[root@host ~]$ systemctl enable --now nfs-server
```

En primer lugar, se procede a editar archivo de configuración `/etc/idmapd.conf`, descomentar y escribir nombre de dominio.

```
Domain = hpc.tfm
```

Luego, se procede a editar el archivo de configuración `/etc/auto.master`, agregando en las últimas líneas.

```
/home /etc/auto.home
/usr/local/ /etc/auto.usr.local
```

Seguidamente, se realiza la edición del archivo de configuración /etc/auto.home .

```
* -fstype=nfs4  master.hpc.tfm:/home/&
```

Finalmente, se procede a agregar el punto de montaje `/usr/local`.

```
[root@host ~]$ echo 'master.hpc.tfm:/usr/local  /usr/local  nfs4
rw,bg,hard,intr,_netdev  0  0' >> /etc/fstab
[root@host ~]$ systemctl daemon-reload
[root@host ~]$ mount -a
```

Cliente FreeIPA

En este apartado, se describe el proceso de instalación del cliente FreeIPA y su incorporación al dominio `hpc.tfm`, gestionado por el servidor `master.hpc.tfm`.

```
[root@host ~]$ dnf module install idm:DL1/client -y
[root@host ~]$ ipa-client-install --server=master.hpc.tfm --domain hpc.tfm
```

A continuación, se habilita la creación automática de directorios personales al momento del inicio de sesión, y se procede a activar el servicio correspondiente.

```
[root@host ~]$ authselect enable-feature with-mkhomedir
[root@host ~]$ systemctl enable --now oddjobd.service
```

Munge

En este apartado, se describe el proceso de instalación de Munge en el nodo de cálculo.

```
[root@host ~]$ dnf install munge munge-libs munge-devel -y
```

A continuación, se distribuye y configura la clave de Munge, luego se ajustan los permisos necesarios y se habilita el servicio para verificar su operación correcta mediante una prueba local.

```
[root@host ~]$ scp root@master:/etc/munge/munge.key /etc/munge
[root@host ~]$ chown -R munge:munge /etc/munge/ /var/log/munge/
[root@host ~]$ chmod 0700 /etc/munge/ /var/log/munge/
[root@host ~]$ systemctl enable --now munge
[root@host ~]$ munge -n | unmunge | grep STATUS
```

Slurm

Seguidamente, se procede a instalar los paquetes necesarios para el gestor de colas Slurm.

```
[root@master ~]$ dnf install slurm slurm-slurmd slurm-pam slurm-devel -y
```

Adicionalmente, se lleva a cabo la instalación de los paquetes complementarios de Slurm para garantizar su funcionalidad completa.

```
[root@master ~]$ dnf install slurm-perlapi slurm-torque slurm-contribs slurm-doc -y
```

Por otra parte, es necesario abrir los puertos correspondientes para permitir la comunicación entre los nodos del clúster, según los servicios utilizados, como se detalla a continuación.

```
[root@host ~]$ firewall-cmd --permanent --add-port=6817-6819/tcp
[root@host ~]$ firewall-cmd --permanent --add-port=6817-6819/udp
[root@host ~]$ firewall-cmd --reload
```

Asimismo, se debe copiar la configuración desde el servidor principal, ya que el archivo `slurm.conf` debe ser idéntico tanto en el nodo maestro como en los nodos de cómputo.

```
[root@host ~]$ scp root@master:/etc/slurm/slurm.conf /etc/slurm/slurm.conf
```

Como paso siguiente, se crean los directorios necesarios para el funcionamiento de Slurm y se asignan los permisos correspondientes al usuario del servicio.

```
[root@host ~]$ mkdir -p /var/spool/slurm/ctld /var/spool/slurmctld /var/log/slurm
/var/spool/slurmd
[root@host ~]$ chown -R slurm:slurm /var/spool/slurm /var/spool/slurmctld
/var/log/slurm /var/spool/slurmd
/var/log/slurm
```

Finalmente, se procede a iniciar los servicios necesarios y a verificar el correcto funcionamiento de los mismos.

```
[root@host ~]$ systemctl enable slurmd --now
[root@host ~]$ systemctl status slurmd
slurmd.service - Slurm node daemon
Loaded: loaded (/etc/systemd/system/slurmd.service;enabled; vendor preset: disabled)
Active: active (running) since Mon 2025-05-05 10:07:51 CEST; 6h ago
Main PID: 1523 (slurmd)
    Tasks: 1
   Memory: 6.5M
   CGroup: /system.slice/slurmd.service
           -1523 /usr/sbin/slurmd -D

May 05 10:07:51 nodo01.hpc.tfm systemd[1]: Started Slurm node daemon.
May 05 15:47:55 nodo01.hpc.tfm slurmd[1523]: slurmd: Launching batch job 113 for
UID 925400003
```

Limitación de acceso a los nodos de cómputo

En este apartado se indica la configuración necesaria que debe realizarse con el objetivo de que los usuarios no puedan conectarse directamente a los nodos de cómputo. Para ello, se procede a editar el archivo `/etc/pam.d/sshd` y agregar la siguiente línea.

```
#%PAM-1.0
auth       substack     password-auth
auth       include      postlogin
account    required     pam_sepermit.so
account    required     pam_nologin.so
account    required     pam_slurm.so  #Agregar
```

Cliente de monitoreo Ganglia

En este apartado se indica el procedimiento de instalación del cliente de monitoreo Ganglia que permite el envío de los datos al servidor central.

```
[root@master ~]$ dnf install ganglia-gmond -y
```

A continuación, se procede a editar el archivo de configuración Ganglia `/etc/ganglia/gmond.conf`. Modificar y agregar estos parámetros.

```
udp_send_channel {
#  mcast_join = 239.2.11.71 # comentar
  host = 192.168.62.130
  port = 8649
  ttl = 1
}
```

Finalmente, se procede a abrir los puertos del cortafuegos y reiniciar los servicios.

```
[root@host ~]$ firewall-cmd --add-port=8649/udp --permanent
[root@host ~]$ firewall-cmd --add-port=8649/tcp --permanent
[root@host ~]$ firewall-cmd --reload
[root@host ~]$ systemctl enable gmond --now
```

EasyBuilder

En este apartado se muestra el procedimiento para configurar las variables de entorno necesarias para el funcionamiento de *EasyBuild* en los clientes y del sistema de módulos Lmod.

```
[root@master ~]$ cp  /usr/share/lmod/lmod/init/profile /etc/profile.d/z00_lmod.sh
[root@master ~]$ bash -c 'cat > /etc/profile.d/zzeasybuild.sh <<EOF
export EASYBUILD_PREFIX=/usr/local/easybuild/
export EASYBUILD_SOURCEPATH=/usr/local/easybuild/sources
export EASYBUILD_BUILDPATH=/usr/local/easybuild/builds
module use /usr/local/easybuild/modules/all
EOF'
```

4.8. Pruebas

En esta sección, se presentan las pruebas realizadas para asegurar el adecuado funcionamiento del clúster HPC y comprobar el cumplimiento de los requisitos establecidos en la Sección 4.1, las cuales se organizan en las siguientes categorías:

4.8.1. Verificación de acceso y autenticación

En relación con el requisito funcional **RF1**, se verificó que los usuarios únicamente pudieran conectarse al nodo maestro, restringiéndose el acceso a los nodos de cómputo en ausencia de tareas en ejecución. Asimismo, se comprobó que la asignación al grupo correspondiente se realizara correctamente.

Pruebas realizadas

En primer lugar, se verifica que los usuarios han sido creados de manera centralizada en el servidor FreeIPA.

```
[root@master ~]# ipa user-find --raw |grep "uid:"
  uid: admin
  uid: juanperez
```

```
  uid: lauramartinez
  uid: luisperez
```

A continuación, se procede a la verificación de los usuarios asignados a los distintos grupos.

```
[root@master ~]# kinit admin
Password for admin@HPC.TFM:
[root@master ~]# ipa group-show investigadores --all --raw
  dn: cn=investigadores,cn=groups,cn=accounts,dc=hpc,dc=tfm
  cn: investigadores
  gidnumber: 925400007
  member: uid=juanperez,cn=users,cn=accounts,dc=hpc,dc=tfm
[root@master ~]# ipa group-show invitados --all --raw
  dn: cn=invitados,cn=groups,cn=accounts,dc=hpc,dc=tfm
  cn: invitados
  gidnumber: 925400006
  member: uid=lauramartinez,cn=users,cn=accounts,dc=hpc,dc=tfm
```

Posteriormente, se procede a realizar un intento de conexión SSH al nodo maestro con un usuario inexistente.

```
login as: miguel
Keyboard-interactive authentication prompts from server:
| Password:
End of keyboard-interactive prompts from server
Access denied
Keyboard-interactive authentication prompts from server:
| Password:
```

Seguidamente, se procede a verificar que el nodo maestro permite el acceso mediante autenticación correcta.

```
login as: juanperez
Keyboard-interactive authentication prompts from server:
| Password:

Web console: https://master.hpc.tfm:9090/ or https://192.168.62.130:9090/

Last login: Thu Jul  3 23:41:58 2025 from 192.168.62.1
[juanperez@master ~]$
```

Finalmente, se procede a constatar que el acceso mediante cliente SSH, al nodo01 o a cualquiera de ellos, se encuentra restringido. El acceso es permitido únicamente cuando hay un trabajo en curso ejecutándose por el usuario en cuestión.

```
login as: juanperez
Keyboard-interactive authentication prompts from server:
| Password:
| Access denied: user juanperez (uid=1000) has no active jobs on this node.
End of keyboard-interactive prompts from server
Access denied
juanperez@192.168.62.128's password:
```

4.8.2. Validación del sistema de colas Slurm

En relación con los requisitos funcionales **RF2** y **RF3**, se configuró Slurm para aplicar políticas diferenciadas según el tipo de usuario, incluyendo limitaciones de recursos y prioridad en cola. Asimismo, se verificó que el planificador asigna trabajos de corta duración en huecos entre tareas largas, optimizando el uso de los recursos sin afectar la prioridad definida.

Pruebas realizadas

La Figura 4.10 muestra la configuración de particiones en Slurm, con colas separadas para investigadores e invitados, cada una con su propio límite de tiempo de ejecución y acceso a los mismos nodos del clúster.

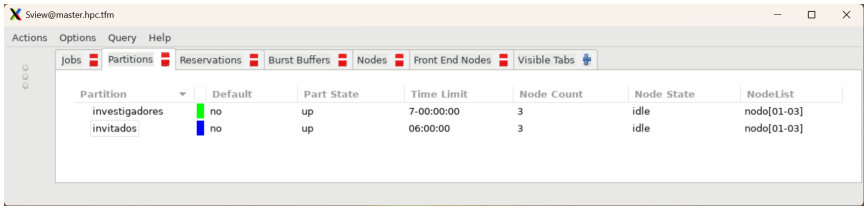

FIGURA 4.10: Vista de la configuración de particiones en Slurm mediante la herramienta sview, confirmando las dos colas configuradas con sus respectivos límites de tiempo y acceso compartido a los nodos del clúster.

Seguidamente, con el objetivo de validar el *backfilling*, en primer lugar, se lanzó un trabajo cuya duración es de dos minutos, utilizando cuatro vCPU, después uno que requería las doce vCPU y quedó en cola, y finalmente un trabajo de diez segundos con una vCPU; al liberarse un nodo, SLURM inició de inmediato el corto antes del que necesita más recursos.

```
sbatch -p investigadores --job-name=largo \
      --nodes=1 --ntasks-per-node=4 --time=00:02:00 \
      --wrap="sleep 120"

sbatch -p investigadores --job-name=grande \
      --nodes=3 --ntasks-per-node=4 --time=00:04:00 \
      --wrap="sleep 240"

sbatch -p investigadores --job-name=corto \
      --ntasks=1 --time=00:00:10 \
      --wrap="sleep 10"
```

4.8.3. Monitorización de recursos

En relación con el requisito funcional **RF4**, se habilitó la herramienta Ganglia para el monitoreo de recursos, permitiendo visualizar tanto el uso histórico como el uso en tiempo real de los nodos del clúster.

Pruebas realizadas

A continuación, se muestra un ejemplo representativo de las métricas disponibles. En la Figura 4.11 se muestra el monitoreo del uso de memoria del clúster, tal como se visualiza a través de la herramienta Ganglia.

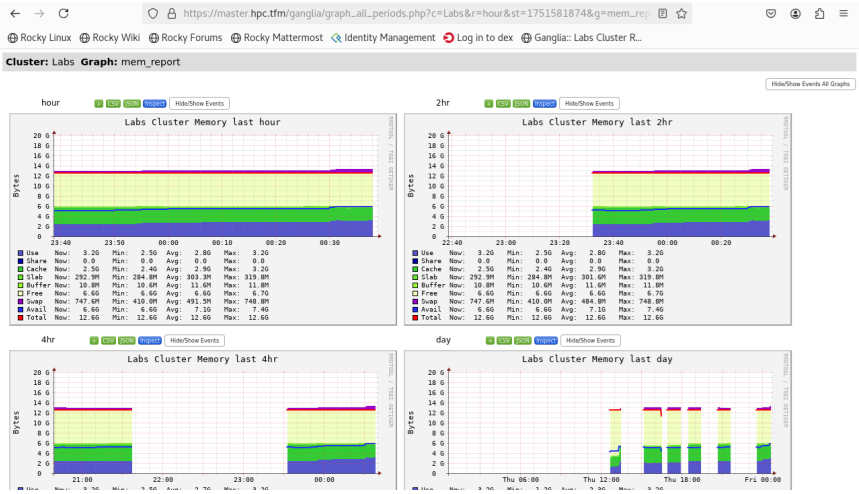

FIGURA 4.11: Monitoreo de memoria del clúster con Ganglia.

4.8.4. Almacenamiento compartido NFS

En relación con el requisito funcional **RF5** se procedió a la verificación de su cumplimiento. Para ello, se procedió a realizar la configuración de un sistema de archivos compartido mediante NFS desde el nodo maestro, exportando los directorios `/home` y `/usr/local` a los nodos del clúster. Se validó el acceso transparente a los archivos personales y de proyecto desde cualquier nodo, sin necesidad de configuración manual.

Pruebas realizadas

En primer lugar, se procedió a la verificación de los recursos compartidos desde el nodo maestro hacia los distintos nodos del clúster.

```
[root@master ~]# exportfs -v
/home 192.168.62.0/24.
(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,no_root_squash,no_all_squash)
/usr/local 192.168.62.0/24.
(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,no_root_squash,no_all_squash)
```

A continuación, se verificó el montaje de directorios compartidos desde nodo de cálculo, accediendo con permisos de administrador.

```
[root@nodo01 ~]# su juanperez
[juanperez@nodo01 root]$ mount | grep nfs | awk '{print $1, "->", $3}'
master.hpc.tfm:/usr/local -> /usr/local
master.hpc.tfm:/home/juanperez -> /home/juanperez
```

4.8.5. Gestión de Software con EasyBuild

En relación con el requisito funcional **RF6**, se procedió a comprobar que los usuarios pueden usar los módulos de entorno para cargar, listar y trabajar con versiones específicas de *software* científico que se encuentran disponibles en el servidor. Adicionalmente, resulta importante indicar que *EasyBuild* permite instalar nuevos paquetes bajo demanda.

Pruebas realizadas

En primer lugar, se procedió a verificar todas las versiones disponibles de los módulos que se encuentran disponibles para su utilización.

```
[usuario@master ~]$ module avail

--------------------- /usr/local/easybuild/modules/all ----------------------
   Autoconf/2.69-GCCcore-10.2.0           M4/1.4.19-GCCcore-13.3.0
   gettext/0.22.5                   (D)   ncurses/6.2-GCCcore-11.2.0
   Automake/1.16.2-GCCcore-10.2.0         Miniconda3/23.10.0-1
   help2man/1.47.16-GCCcore-10.2.0        ncurses/6.5-GCCcore-13.3.0
   Automake/1.16.5                  (D)   Miniconda3/24.7.1-0           (D)
   Autotools/20200321-GCCcore-10.2.0      OpenMPI/4.1.2-GCC-10.2.0
   Bison/3.8.2                      (D)   Perl/5.32.0-GCCcore-10.2.0
   DB/18.1.40-GCCcore-10.2.0              Perl/5.38.0                   (D)
   GCC/10.2.0                             Python/2.7.18-GCCcore-11.2.0-bare
   libreadline/8.1-GCCcore-11.2.0         zlib/1.2.11-GCCcore-11.2.0

....
```

A continuación, se procede a verificar la factibilidad de la carga de una versión específica de los módulos listados.

```
[usr@master ~]$ module load Miniconda3/23.10.0-1
```

Luego, se verifica que para la instalación de un *software* específico, el administrador procede a utilizar *EasyBuild*, como se muestra a continuación.

```
[root@master ~]$ su - easybuilder
[easybuilder@master ~]$ eb
/usr/local/easybuild/easyconfigs/m/Miniconda3/Miniconda3-24.7.1.eb --robot
```

4.8.6. Sistema de notificaciones por email

En relación con el requisito funcional **RF7**, se integró un sistema de notificación por correo electrónico que envía automáticamente un aviso a los usuarios al finalizar la ejecución de sus trabajos.

Pruebas realizadas

En primer lugar, se presenta un ejemplo representativo de las notificaciones. En la Figura 4.12 se muestra la bandeja de entrada del correo.

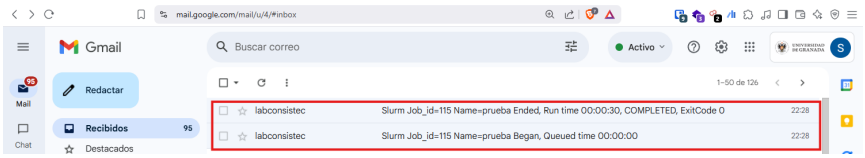

FIGURA 4.12: Notificaciones por correo electrónico del Slurm

4.8.7. Licenciamiento

En relación con el requisito no funcional **RNF1**, se verificó que el sistema utiliza únicamente *software* libre en todos sus componentes principales. Para más detalles sobre licencias libres y de código abierto, pueden consultarse la Free Software Foundation (Free Software Foundation, 2025) y la Open Source Initiative (Open Source Initiative, 2025).

Pruebas realizadas

En primer lugar, en la Tabla 4.8 se presentan los componentes principales del sistema, sus licencias y los enlaces oficiales que confirman su carácter de *software* libre.

Software	Licencia	Enlace de verificación
Rocky Linux	BSD 3-Clause	https://rockylinux.org/es-ES/legal/licensing
SLURM	GPL v2	https://github.com/SchedMD/slurm
FreeIPA	GPL v3	https://github.com/freeipa/freeipa
Ganglia	BSD License	https://github.com/ganglia/monitor-core
Open OnDemand	MIT License	https://github.com/OSC/ondemand
EasyBuild	GPL v2	https://github.com/easybuilders/easybuild
NFS Utils	GPL v2	https://github.com/andreas-gruenbacher/nfs-utils

TABLA 4.8: Tipos de licencias de software utilizado.

4.8.8. Escalabilidad

En relación con el requisito funcional **RF3**, se comprobó que el sistema permite la incorporación de nodos de cómputo a demanda, facilitando la escalabilidad del clúster según las necesidades de uso.

Pruebas realizadas

En primer lugar, es importante indicar que actualmente, Slurm gestiona tres nodos de cómputo, siendo posible agregar mayor cantidad de nodos al clúster, previa instalación y configuración de Slurm en los nuevos nodos, y actualizando el archivo de configuración en los distintos nodos.

```
[root@master ~]# grep NodeName /etc/slurm/slurm.conf
NodeName=nodo[01-03] CPUs=4 RealMemory=4096 Sockets=1 CoresPerSocket=4
ThreadsPerCore=1 State=UNKNOWN
```

A continuación, se procedió a realizar la verificación de los recursos compartidos, los cuales se encuentran configurados para ser accesibles desde cualquiera de los nodos que se encuentran activos, como aquellos que se vayan incorporando al clúster.

```
[root@master ~]# cat /etc/exports
/home 192.168.62.0/24(rw,no_root_squash)
/usr/local/ 192.168.62.0/24(rw,no_root_squash)
```

Luego, se procedió a verificar la arquitectura actual del clúster en términos de cantidad de nodos de cómputo.

```
[root@master ~]# sinfo -o "%D %C"
NODES CPUS(A/I/O/T)
3 0/12/0/12

# 3 nodos, 12 CPUs totales
```

4.8.9. Seguridad

En relación con el requisito no funcional **RNF4**, se implementó un modelo de seguridad multicapa basado en ISO/IEC 27002:2022, que incluye autenticación centralizada con FreeIPA, cortafuegos por nodo, cifrado mediante SSH y control de accesos por roles, garantizando la confidencialidad, integridad y disponibilidad del sistema.

Pruebas realizadas

En primer lugar, se realizó la verificación de los puertos abiertos en el nodo maestro para asegurar que solo se encuentren habilitados los necesarios, evitando servicios innecesarios expuestos.

```
[root@master ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources: 192.168.62.0/24
  services: cockpit dhcpv6-client freeipa-ldap freeipa-ldaps kerberos
  mountd nfs nfs3 ntp rpc-bind ssh
  ports: 944-951/tcp 944-951/udp 6817-6819/tcp 6817-6819/udp
  60001-63000/tcp 8649/udp 9443/tcp 8649/tcp 5554/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

```
        rule family="ipv4" source address="192.168.62.131" accept
        rule family="ipv4" source address="192.168.62.128" accept
        rule family="ipv4" source address="192.168.62.129" accept
```

A continuación, se procedió a realizar la verificación de los puertos abiertos en los nodos de cómputo para asegurar que solo se encuentren habilitados los necesarios, evitando la exposición de servicios innecesarios.

```
[root@nodo01 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens160
  sources:
  services: cockpit dhcpv6-client ssh
  ports: 6817-6819/tcp 6817-6819/udp 8649/udp 8649/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Finalmente, se procedió a verificar que el nodo maestro mantenga abiertos exclusivamente los puertos seguros necesarios para los servicios principales del clúster, como SSH, HTTPS y LDAPS.

```
[root@master ~]# ss -tlnp | grep -E "(22|443|636)"
LISTEN 0  32 192.168.122.1:53   0.0.0.0:* users:(("dnsmasq",pid=2343,fd=6))
LISTEN 0 128       0.0.0.0:22   0.0.0.0:* users:(("sshd",pid=1352,fd=3))
LISTEN 0 128          [::]:22      [::]:* users:(("sshd",pid=1352,fd=4))
LISTEN 0 100             *:8443       *:* users:(("java",pid=3246,fd=93))
LISTEN 0 511             *:443        *:* users:(("httpd",pid=109878,fd=7),
LISTEN 0 128             *:636        *:* users:(("ns-slapd",pid=2466,fd=8))
```

4.8.10. Usabilidad

En relación con el requisito no funcional **RNF5**, se habilitó Open OnDemand para ofrecer una interfaz web intuitiva que permita a los usuarios sin experiencia previa en HPC ejecutar tareas comunes. Adicionalmente, se proporcionó documentación accesible para facilitar su uso.

Pruebas realizadas

En primer lugar, la Figura 4.13 muestra la interfaz de Open OnDemand, donde los usuarios pueden crear, editar y enviar trabajos al clúster de forma sencilla, sin usar la línea de comandos.

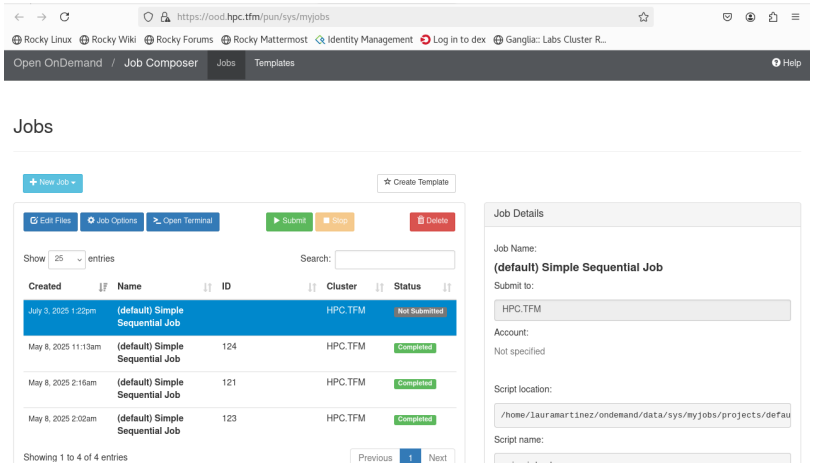

FIGURA 4.13: Interfaz de Open OnDemand en el módulo compositor de trabajos

Capítulo 5

DISEÑO DEL SOFTWARE DE DESPLIEGUE AUTOMATIZADO DE ALGORITMOS

5.1. Análisis de Requisitos

En esta sección se especifican los requisitos del sistema, los cuales se dividen en dos categorías: los requisitos funcionales, descritos en la Subsección 5.1.1, y los requisitos no funcionales, presentados en la Subsección 5.1.2.

5.1.1. Requisitos funcionales

A continuación, se enumeran los requisitos funcionales identificados a partir del análisis de las necesidades técnicas.

RF1. **Sincronización automática de carpetas:** El sistema debe sincronizar automáticamente los archivos del entorno local del usuario con el clúster, manteniendo la coherencia entre ambos entornos sin intervención manual.

RF2. **Replicación automática del entorno Python:** El sistema debe capturar las dependencias del entorno Python local del usuario y generar automáticamente un entorno equivalente en el clúster mediante entornos virtuales.

RF3. **Gestión de tareas de usuario:** El sistema debe permitir a los usuarios listar el estado de sus trabajos en Slurm, visualizar detalles de ejecución y cancelar tareas activas si fuera necesario.

RF4. **Visualización de resultados:** El sistema debe proporcionar acceso a la salida estándar y archivos de resultado generados por los algoritmos, sin requerir acceso manual al sistema remoto.

RF5. **Automatización del envío de trabajos:** El sistema debe generar automáticamente los scripts de ejecución compatibles con Slurm, transferirlos al clúster y lanzar los trabajos correspondientes desde el nodo maestro.

5.1.2. Requisitos no funcionales

A continuación, se enumeran los requisitos no funcionales identificados a partir del análisis de las necesidades técnicas.

RNF1. **Compatibilidad multiplataforma:** El sistema debe ejecutarse correctamente en Windows y Linux, permitiendo a los usuarios interactuar con el clúster desde ambos entornos sin necesidad de realizar adaptaciones técnicas adicionales.

RNF2. **Facilidad de instalación:** El sistema debe contar con un procedimiento de instalación simple, documentado y guiado, que no requiera conocimientos avanzados en administración de sistemas ni intervención manual compleja.

RNF3. **Usabilidad:** El sistema debe estar diseñado para ser utilizado por personal técnico con conocimientos básicos en programación científica, sin necesidad de formación especializada en sistemas HPC. Debe incluir documentación clara sobre su funcionamiento.

RNF4. **Licenciamiento:** Todos los componentes utilizados en el desarrollo del sistema deben estar licenciados bajo esquemas de software libre, garantizando su libre distribución, modificación y reutilización.

5.2. Casos de uso

Esta sección presenta los casos de uso que ilustran las acciones que el usuario puede realizar para ejecutar los procesos definidos por la solución de despliegue. En la Figura 5.1 se muestra el diagrama de casos de uso.

5.2.1. Actores del sistema

- **Investigador o usuario técnico:** Utiliza el clúster HPC para desplegar aplicaciones, lanzar y supervisar sus trabajos y preparar los entornos de ejecución.

5.2.2. Diagrama de caso de uso

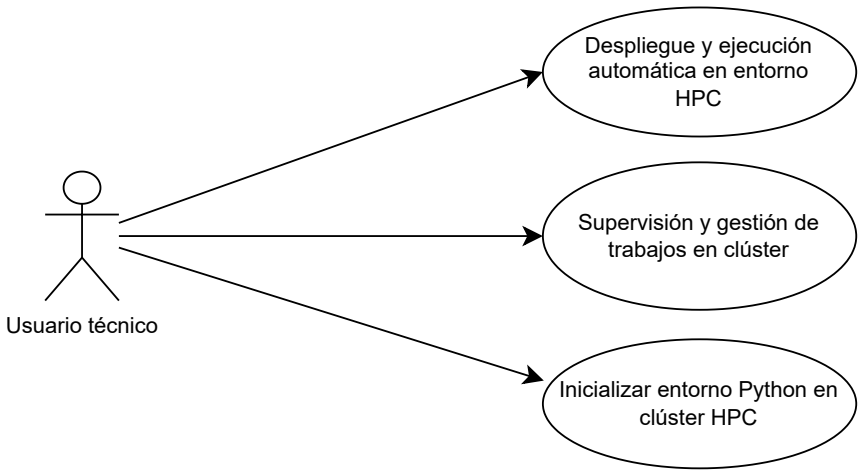

FIGURA 5.1: Diagrama de caso de uso general que ilustra las acciones del usuario técnico en un entorno HPC.

5.2.3. Despliegue y ejecución automática en entorno HPC

Nombre	**1 - Despliegue y ejecución automática en entorno HPC**
Caso de Uso General	Despliegue automatizado de experimentos en HPC.
Descripción	Este caso de uso permite al usuario ejecutar automáticamente un algoritmo en el entorno HPC. El sistema envía de forma remota todos los archivos locales necesarios al nodo maestro, crea el entorno de ejecución, lanza el trabajo en el planificador de recursos y proporciona una interfaz para monitorear en tiempo real los logs de ejecución y estado del trabajo en la cola.
Actor Principal	Investigador o usuario técnico.
Actor Secundario	Ninguno.
Precondiciones	El usuario debe contar con credenciales válidas. También se debe disponer de un algoritmo a ejecutar, desarrollado en un lenguaje compatible con el servidor.
Postcondiciones	El algoritmo es enviado y comienza la ejecución.
Secuencias de Acciones	**Investigador:** a. Descargar implementación dentro del proyecto. b. Configurar parámetros de conexión al clúster. c. Ejecutar aplicación de envío de código al servidor. d. Monitorear la salida o log generado de la ejecución.
Validaciones/Restricciones	a. Los parámetros definidos deben ser válidos. b. El usuario debe tener permisos en el clúster.

TABLA 5.1: Caso de uso - Despliegue y ejecución automática en entorno HPC.

5.2.4. Supervisión y gestión de trabajos en clúster.

Nombre	**2 - Supervisión y gestión de trabajos en clúster**
Caso de Uso General	Herramientas para la cancelación, consulta e inspección detallada de trabajos en ejecución o finalizados.
Descripción	Este caso de uso proporciona al usuario la capacidad de gestionar y supervisar sus trabajos dentro del entorno HPC. Incluye comandos para cancelar todos los trabajos activos del usuario, obtener información del estado del clúster, y acceder al log del último trabajo ejecutado. Estas permiten un control del flujo de ejecución mediante las utilidades proporcionadas por el sistema de planificación de recursos.
Actor Principal	Investigador o usuario técnico
Actor Secundario	Ninguno
Precondiciones	El usuario debe contar con credenciales válidas, y haber ejecutado previamente trabajos mediante el gestor de colas.
Postcondiciones	Los trabajos activos son cancelados si así se solicita, se muestra el estado general de los recursos y se accede al log del último trabajo.
Secuencias de Acciones	**Investigador:** a. Descargar implementación dentro del proyecto. b. Configurar parámetros de conexión al clúster. c. Ejecutar tareas de control: cancelar trabajos activos, consultar el estado del sistema, listar trabajos recientes e inspeccionar el log del último trabajo.
Validaciones/Restricciones	a. Los parámetros definidos deben ser válidos. b. El usuario debe tener permisos en el clúster. c. El usuario sólo puede consultar y cancelar sus propios trabajos.

TABLA 5.2: Caso de uso - Supervisión y gestión avanzada de trabajos en clúster.

5.2.5. Inicializar entorno Python en clúster HPC

Nombre	**3 - Inicializar entorno Python en clúster HPC**
Caso de Uso General	Automatización del proceso de creación, activación y configuración de un entorno virtual Python para ejecución remota en un clúster HPC.
Descripción	Este caso de uso permite al usuario inicializar un entorno de trabajo Python completamente funcional en un clúster remoto, incluyendo la creación de un entorno virtual, instalación de dependencias, generación del archivo de dependencias y activación final del entorno para continuar trabajando de forma interactiva.
Actor Principal	Investigador o usuario técnico
Actor Secundario	Ninguno
Precondiciones	Contar con credenciales válidas. También se debe disponer de un algoritmo a ejecutar en Python.
Postcondiciones	a. Se ha creado un entorno virtual Python llamado según la configuración. b. Se han instalado todas las dependencias necesarias. c. El entorno queda listo para ejecutar tareas en el clúster.
Secuencias de Acciones	**Investigador:** a. Descargar implementación dentro del proyecto. b. Configurar parámetros de conexión al clúster. c. Ejecutar aplicación para la creación del entorno. d. Crear el entorno virtual con `python` utilizando las dependencias del código.
Validaciones/Restricciones	a. El código debe ser desarrollado en Python.

TABLA 5.3: Caso de uso - Inicialización de entorno Python en clúster HPC.

5.3. Diseño

En esta sección se detalla el diseño del sistema, organizado en dos apartados principales: el diseño de interfaces, descrito en la Subsección 5.4, y el diseño arquitectónico, presentado en la Subsección 5.4.2.

5.4. Diseño de Interfaces

Esta sección presenta las interfaces disponibles, las cuales permiten al usuario interactuar con la herramienta a través del terminal.

5.4.1. Automatización de ejecución en clúster

En la Figura 5.2 se muestra una sesión en el terminal dentro del clúster HPC, donde el usuario supervisa el estado de los nodos y trabajos en ejecución, y realiza el seguimiento de errores en tiempo real. Además, se observa la salida de un script en Python y un resumen del uso de disco por parte del usuario actual.

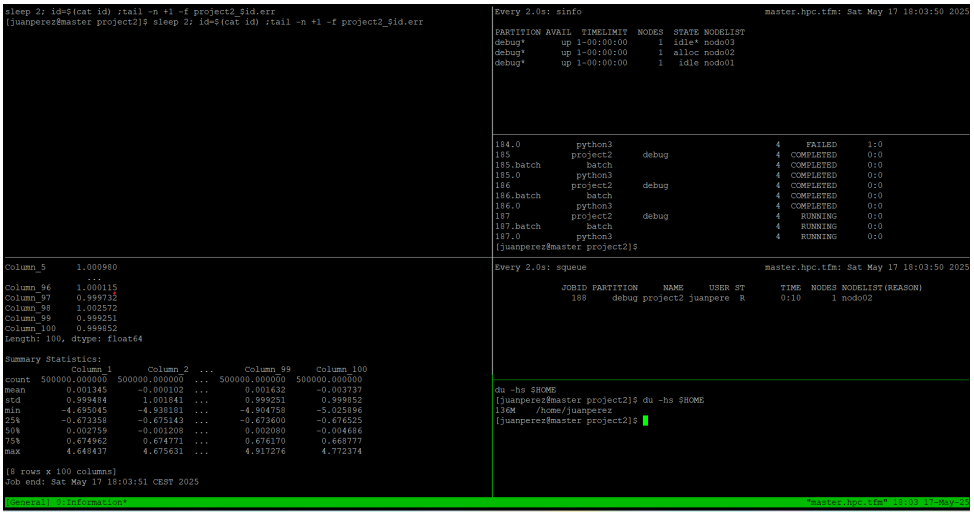

FIGURA 5.2: Automatización de la ejecución de trabajos en clúster mediante Slurm y multiplexación de terminal con tmux.

5.4.1.1. Supervisión y gestión de trabajos en clúster

En esta sección, se resumen las principales acciones de supervisión y gestión, como la consulta de recursos, la revisión de trabajos, la cancelación de procesos y la inspección de registros de salida, tal como se ilustra en las Figuras 5.3 y 5.4.

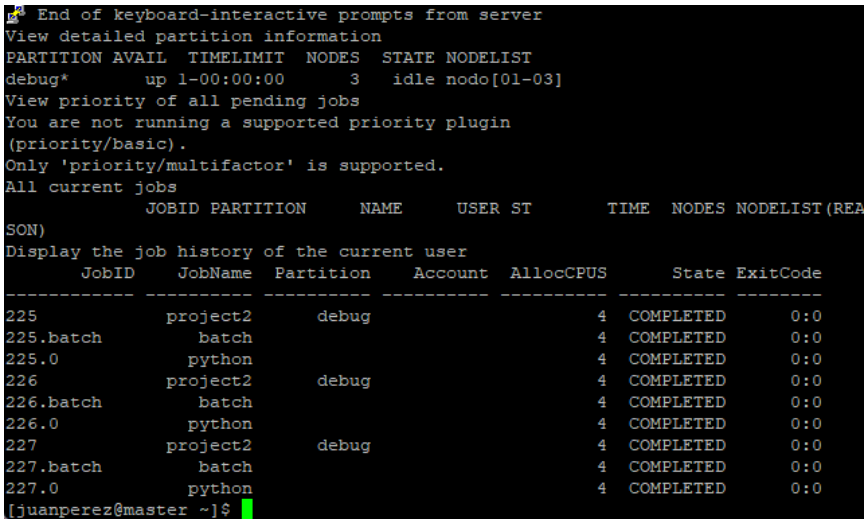

FIGURA 5.3: Se supervisan los recursos del clúster y se consulta el historial de trabajos ejecutados por el usuario, mostrando recursos disponibles, estado de los nodos y estado de las ejecuciones.

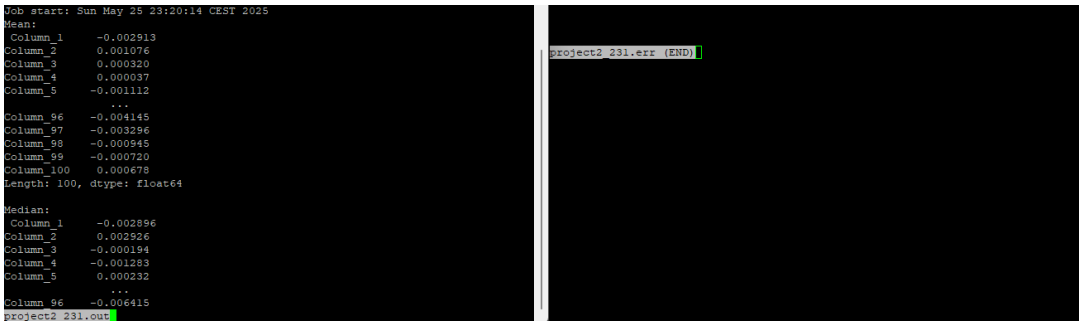

FIGURA 5.4: Inspección de registros de salida de la ejecución

5.4.1.2. Inicialización del entorno Python en el clúster

En la Figura 5.5 se muestra el proceso de inicialización de un entorno virtual Python en el clúster, incluyendo la detección de dependencias locales, la generación del archivo de requisitos y la creación del entorno en el nodo maestro.

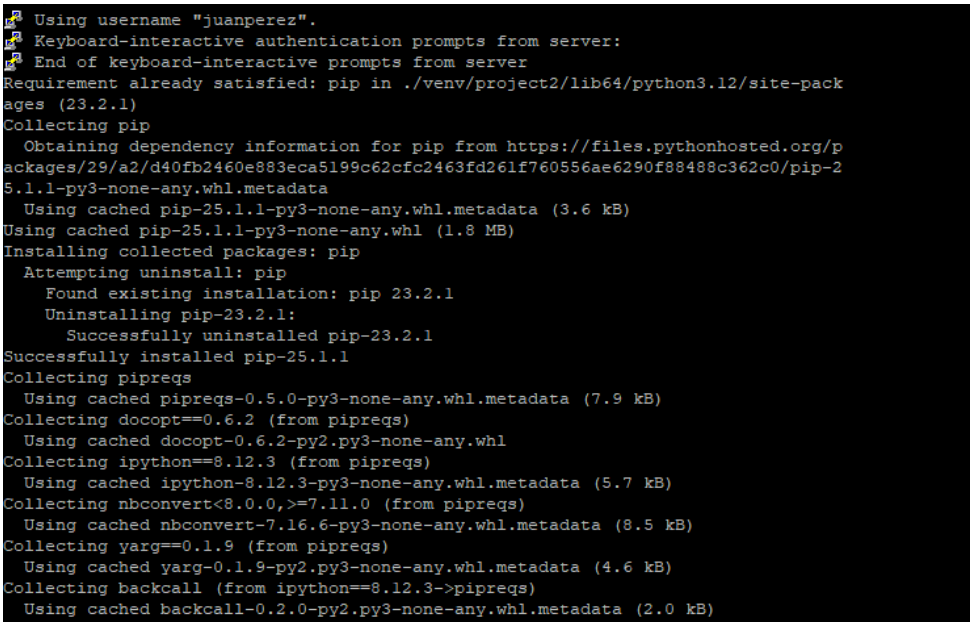

FIGURA 5.5: Creación de entorno virtual Python e instalación de dependencias en el clúster

5.4.2. Arquitectura

La arquitectura general del sistema de despliegue automatizado propuesto es posible observar en la Figura 5.6. Esta arquitectura está diseñada para simplificar la ejecución de algoritmos en entornos HPC desde un entorno local, minimizando la intervención del usuario. Entre sus funciones principales se encuentran el envío directo de trabajos al nodo maestro, la inicialización del entorno Python en el clúster, y la supervisión y gestión de los trabajos ejecutados. Todo ello se realiza tras una configuración inicial sencilla.

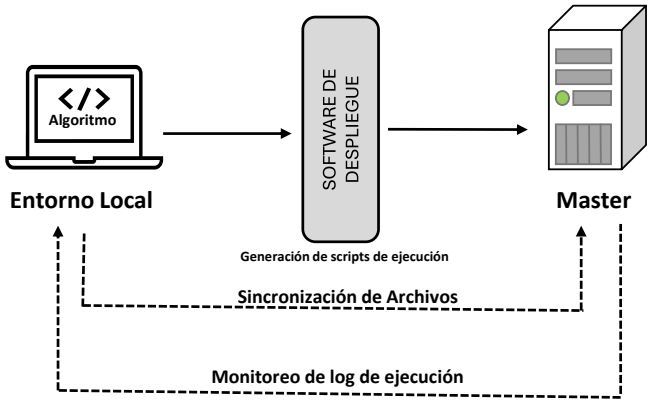

FIGURA 5.6: Flujo de trabajo del sistema de despliegue automatizado.
Fuente: elaboración propia

5.5. Implementación

La implementación de la arquitectura requirió de diversas herramientas y tecnologías, las cuales fueron seleccionadas tanto por su compatibilidad con entornos HPC como por su bajo número de dependencias y su adecuación a los requisitos funcionales y no funcionales definidos previamente.

Con el fin de facilitar su ejecución desde SO Windows, se utiliza la herramienta *Cygwin*, que proporciona un entorno similar a Unix sobre *Windows* (Red Hat, Inc., 2025), permitiendo así realizar operaciones nativas de *Linux* desde dicho sistema. Para establecer conexiones remotas seguras mediante SSH desde *Windows*, se incluye el uso de PuTTY (PuTTY Developers, 2024). Para sincronizar los archivos del entorno local al servidor, se utiliza la herramienta *Rsync* (Tridgell, 1996), que permite mantener sincronizados los archivos entre el sistema local y el clúster. Para la ejecución del *software* se utiliza Python con la menor cantidad posible de dependencias, evitando así que el usuario deba realizar operaciones complementarias. Además, se emplea la librería **pipreqs** (Kravcenko, 2024), que permite generar automáticamente el archivo `requirements.txt` en función del código fuente del proyecto.

Resulta importante indicar que el único requisito del lado del servidor es contar con Tmux, una herramienta que permite continuar la ejecución de procesos sin conexión activa y gestionar múltiples sesiones en una sola terminal.

A continuación, la Figura 5.7 presenta la estructura del proyecto y sus componentes principales.

Estructura del Proyecto

Proyecto/ *(Directorio raíz del sistema desarrollado)*
- Slurm/ *(Contiene la implementación completa de la herramienta)*
  - cygwin64/ *(entorno Cygwin para entornos Windows)*
  - log/ *(logs de ejecución)*
  - putty/ *(Cliente SSH para Windows)*
  - _config.ini *(Parámetros de configuración del sistema)*
  - _secret.ini *(credenciales)*
  - access_server.py *(conexión SSH al nodo maestro)*
  - add_ssh_known_host.py *(añadir host SSH confiable)*
  - cancel_all_job.py *(cancelación de trabajos)*
  - create_environment.py *(Creación de entorno Python)*
  - information.py *(Estado del clúster)*
  - last_job_log.py *(Último log)*
  - send_job.py *(Envío de trabajo a cola slurm)*

FIGURA 5.7: Estructura interna del sistema de automatización para despliegue de algoritmos en clúster HPC.

5.5.1. Configuración del entorno de ejecución

El proceso de automatización permite que el sistema genere de forma dinámica el archivo `sbatch`, lo que habilita el envío de trabajos al clúster mediante Slurm. En este sentido, la configuración necesaria para personalizar este proceso se define a través de un archivo editable, que incluye los parámetros de conexión SSH, sincronización de archivos y la planificación de tareas. Los valores admitidos y su uso se encuentran documentados en el manual de usuario en el Anexo C.

5.5.2. Disponibilidad del código fuente

El código fuente completo del sistema se encuentra disponible en el siguiente repositorio: https://github.com/seriab1/Slurm

5.6. Pruebas

En esta sección, se presentan las pruebas realizadas para asegurar el adecuado funcionamiento del *software* desarrollado para utilizar en el HPC y comprobar el cumplimiento de los requisitos establecidos en la Sección 5.1, se llevaron a cabo pruebas organizadas en las siguientes categorías:

5.6.1. Sincronización automática de carpetas

En relación con el requisito funcional **RF1**, que establece que el sistema debe sincronizar automáticamente los archivos del entorno local del usuario con el clúster, manteniendo la coherencia entre ambos entornos sin intervención manual, se llevaron a cabo pruebas para validar la correcta transferencia de archivos mediante el uso de la herramienta *rsync*.

Pruebas realizadas

En primer lugar, se procede a verificar que durante el envío, los archivos se transfieren al servidor mediante rsync, como parte del proceso automatizado ejecutado al correr el *script* `send_job.py`. A continuación, se muestra el registro del proceso, que incluye la creación del directorio de destino, la copia de archivos y las estadísticas finales.

```
Windows
sending incremental file list
created directory /home/juanperez/project10
./
example.py

587 100%    0.00kB/s    0:00:00
587 100%    0.00kB/s    0:00:00 (xfr#1, to-chk=1/3)
model_python.sh

470 100%  458.98kB/s    0:00:00
470 100%  458.98kB/s    0:00:00 (xfr#2, to-chk=0/3)

sent 781 bytes  received 105 bytes  590.67 bytes/sec
total size is 1,057  speedup is 1.19
Folder sent successfully.

Process finished with exit code 0
```

Posteriormente, se procede a desplegar la lista de archivos sincronizados en el servidor.

```
[juanperez@master ~]$ cd project10
[juanperez@master project10]$ ls -l
total 20
-rwx------ 1 juanperez juanperez  587 May 12 23:17 example.py
-rw-rw-r-- 1 juanperez juanperez    4 Jul  5 16:17 id
-rw-rw-r-- 1 juanperez juanperez   24 Jul  5 16:17 jobid.txt
-rwx------ 1 juanperez juanperez  470 Jul  5 16:17 model_python.sh
-rw-rw-r-- 1 juanperez juanperez    0 Jul  5 16:17 project10_359.err
-rw-rw-r-- 1 juanperez juanperez 1647 Jul  5 16:17 project10_359.out
```

5.6.2. Replicación automática del entorno Python

En relación con el requisito funcional **RF2**, que indica que el sistema debe capturar automáticamente las dependencias del entorno Python local y recrear un entorno equivalente en el clúster utilizando entornos virtuales, se llevaron a cabo pruebas para validar esta funcionalidad.

Pruebas realizadas

En primer lugar, para la realización de las pruebas se ejecutó el *script* `send_job.py`, el cual se encarga de conectarse al servidor, crear un entorno virtual y generar automáticamente el archivo requirements.txt a partir del código del proyecto. Luego, instala todas las dependencias necesarias dentro de ese entorno, replicando así el entorno Python local en el clúster de forma automática y sin necesidad de intervención manual.

```
(project10) cat requirements.txt
numpy==2.3.1
pandas==2.3.0
(project10) pip list
Package                   Version
------------------------- -----------
asttokens                 3.0.0
attrs                     25.3.0
backcall                  0.2.0
nbformat                  5.10.4
numpy                     2.3.1
packaging                 25.0
pandas                    2.3.0
pandocfilters             1.5.1
parso                     0.8.4
pexpect                   4.9.0
pickleshare               0.7.5
pip                       25.1.1
pipreqs                   0.5.0
platformdirs              4.3.8

...
```

5.6.3. Gestión de tareas de usuario

En relación con el requisito funcional **RF3**, que establece que el sistema debe permitir a los usuarios listar el estado de sus trabajos en Slurm, visualizar detalles de ejecución y cancelar tareas activas si fuera necesario, se llevaron a cabo pruebas para verificar el correcto funcionamiento de estas operaciones.

Pruebas realizadas

En primer lugar, con el objetivo de realizar las pruebas se procedió a la ejecución del *script* `cancel_all_job.py`, encargado de conectarse al servidor, listar los trabajos activos del usuario en Slurm y cancelarlos automáticamente. La prueba permitió validar correctamente la cancelación de un trabajo en estado *PENDING*, demostrando así el funcionamiento del sistema de gestión de tareas sin necesidad de intervención manual.

```
Using username "juanperez".
Keyboard-interactive authentication prompts from server:
End of keyboard-interactive prompts from server
All current jobs
Sat Jul 05 17:37:10 2025
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NO
               360 investiga project1 juanpere  PENDING       0:00   6:00:00
Cancel all jobs
All current jobs
```

```
Sat Jul 05 17:37:10 2025
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NO
[juanperez@master ~]$
```

A continuación, se procede a visualizar información de los trabajos mediante la ejecución del *script* `information.py`.

```
Using username "juanperez".
Keyboard-interactive authentication prompts from server:
End of keyboard-interactive prompts from server
View detailed partition information
PARTITION      AVAIL  TIMELIMIT  NODES  STATE NODELIST
investigadores    up 7-00:00:00      3   idle nodo[01-03]
View priority of all pending jobs
You are not running a supported priority plugin
(priority/basic).
Only 'priority/multifactor' is supported.
All current jobs
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
Display the job history of the current user
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
320            project2 investiga+                     4  COMPLETED      0:0
320.batch         batch                                4  COMPLETED      0:0
320.0            python                                4  COMPLETED      0:0
321            project2 investiga+                     4  COMPLETED      0:0
321.batch         batch                                4  COMPLETED      0:0
321.0            python                                4  COMPLETED      0:0
322            project2 investiga+                     4  COMPLETED      0:0
322.batch         batch                                4  COMPLETED      0:0
```

5.6.4. Visualización de resultados

En relación con el requisito funcional **RF4**, el sistema cuenta con una terminal que muestra automáticamente la salida estándar y archivos de resultado de los algoritmos ejecutados remotamente. El acceso al sistema remoto se gestiona de forma automática sin intervención del usuario.

Pruebas realizadas

A continuación, se procede a visualizar la información relacionada con la ejecución del último trabajo en cola, lo cual se realiza mediante la ejecución del *script* `last_job_log.py`.

```
Using username "juanperez".
Keyboard-interactive authentication prompts from server:
End of keyboard-interactive prompts from server
                ...
Column_96    -0.000756
Column_97     0.002548
Column_98     0.000906
Column_99     0.001269
Column_100   -0.000753
Length: 100, dtype: float64
```

```
Standard Deviation:
Column_1      1.000927
Column_2      1.000721
Column_3      0.998352
Column_4      1.002014
Column_5      1.001525
                ...
[8 rows x 100 columns]
Job end: Sat Jul  5 17:45:33 CEST 2025
(END)
```

5.6.5. Automatización del envío de trabajos

En relación con el requisito funcional **RF5**, el sistema cuenta con funcionalidad que genera automáticamente los *scripts* de Slurm requeridos para cada trabajo. Los *scripts* se transfieren al clúster y se lanzan desde el nodo maestro sin intervención manual del usuario.

Pruebas realizadas

A continuación, se procede a visualizar la información relacionada con la ejecución del último trabajo en cola, lo cual se realiza mediante la ejecución del *script* `send_job.py`.

```
#!/bin/bash
#------- SLURM OPTIONS
#SBATCH -J project10
#SBATCH -p investigadores
#SBATCH -n 1
#SBATCH -c 4
#SBATCH --mem=0
#SBATCH -o project10_%j.out
#SBATCH -e project10_%j.err
#SBATCH --mail-user=correo@gmail.com
#SBATCH --mail-type=END,FAIL
#SBATCH --time=0-06:00:00

#------- LOAD ENVIRONMENT
source ../venv/project10/bin/activate
#------- LOAD MODULES
#
#------- USER COMMANDS

echo "Job start: $(date)"

srun python -u example.py

echo "Job end: $(date)"
```

5.6.6. Compatibilidad multiplataforma

En relación con el requisito funcional **RNF1**, el sistema demuestra su compatibilidad multiplataforma mediante las pruebas realizadas exitosamente en entornos Linux. El trabajo fue enviado correctamente desde el sistema operativo Linux.

Pruebas realizadas

A continuación, se valida el envío correcto de un trabajo desde un entorno Linux como se muestra en la Figura 5.8, donde se observa la supervisión en tiempo real del estado del trabajo, el uso de recursos y los archivos de salida generados en el clúster.

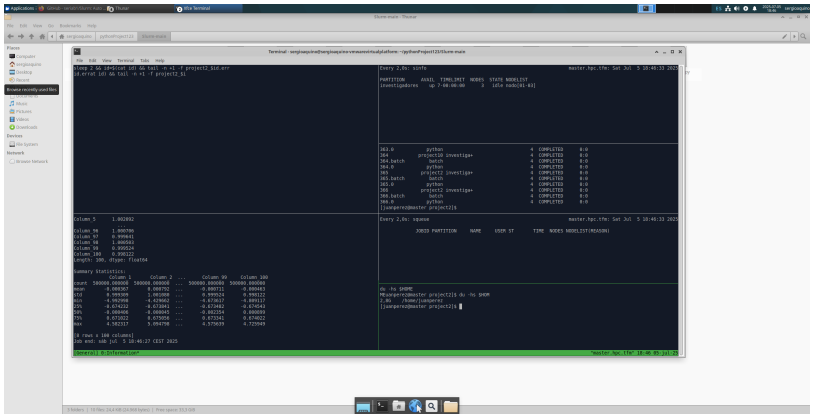

FIGURA 5.8: Monitoreo de trabajos en el clúster desde Linux

5.6.7. Licenciamiento

En relación con el requisito no funcional **RNF4**, se verificó que el sistema utiliza únicamente *software* libre en todos sus componentes principales.

Pruebas realizadas

A continuación, en la Tabla 5.4 se detallan los principales componentes de *software*, sus licencias y los enlaces oficiales que confirman su carácter de *software* libre.

Software	Licencia	Enlace de verificación
PuTTY	MIT	https://www.chiark.greenend.org.uk/~sgtatham/putty/licence.html
Cygwin64	LGPL v3	https://cygwin.com/licensing.html
Python	PSF-2.0	https://docs.python.org/3/license.html

TABLA 5.4: Tipos de licencias de software utilizado en complemento.

Capítulo 6

CONCLUSIONES Y TRABAJO FUTURO

6.1. Objetivos cumplidos

La utilización de servidores HPC es crucial para la resolución de problemas complejos que no pueden ser abordados eficientemente con sistemas convencionales. Esta necesidad se ha vuelto aún más evidente con el creciente uso de algoritmos de inteligencia artificial en ámbitos tanto académicos como profesionales, donde se requieren grandes capacidades de procesamiento para realizar simulaciones, entrenamientos de modelos o análisis de datos a gran escala.

La implementación de un clúster HPC presenta desafíos importantes, tanto en la selección del *hardware* como en la configuración del conjunto de *software*. Además, la capa de acceso a los recursos puede representar una barrera para su uso, especialmente para usuarios con conocimientos limitados en entornos de computación HPC.

En este contexto, este TFM presenta el desarrollo del diseño e implementación de un clúster HPC con recursos limitados, junto con una herramienta de *software* que permite el despliegue automatizado de algoritmos y facilita el uso del entorno HPC.

En relación con los objetivos específicos planteados inicialmente, se logró identificar y comprender los componentes fundamentales, tanto de *hardware* como de *software*, necesarios para la implementación de un clúster HPC en un entorno local. Este análisis permitió definir una arquitectura coherente y funcional.

Adicionalmente, también se ha cumplido con la elaboración de una guía práctica que describe detalladamente el proceso de instalación, configuración y mantenimiento del clúster. Esta documentación técnica facilita que otros usuarios puedan replicar el sistema.

Otro objetivo alcanzado fue el desarrollo de una herramienta que automatiza el envío y la ejecución de algoritmos en el entorno HPC. Esta herramienta reduce significativamente la complejidad operativa y mejora la accesibilidad del sistema para usuarios técnicos sin experiencia específica en administración de clústeres.

Finalmente, se garantizó la compatibilidad multiplataforma de la herramienta, permitiendo su uso en distintos Sistemas Operativos. Esto amplía su aplicabilidad en diversos entornos académicos y profesionales.

A pesar de los logros obtenidos, este trabajo presenta ciertas limitaciones que deben tenerse en cuenta. En primer lugar, no se abordan aspectos relacionados con la infraestructura física de un centro de datos, como el espacio disponible, la refrigeración o el montaje en rack, ya que el enfoque del proyecto se ha centrado exclusivamente en el diseño y configuración funcional de los nodos. Asimismo, la puesta en marcha del entorno HPC se ha realizado en máquinas virtuales, no en *hardware* físico real, aunque esto ha sido útil para validar el diseño. Tampoco se ha contemplado el uso de GPU, debido a la falta de disponibilidad de este *hardware*, aunque los objetivos definidos han podido alcanzarse sin su incorporación. Finalmente, no se llevaron a cabo pruebas con cargas de trabajo reales, ya que la validación se ha centrado en asegurar la funcionalidad del sistema y no en su comportamiento bajo condiciones intensivas de uso.

En resumen, este trabajo demuestra que es posible construir una infraestructura HPC funcional, accesible y automatizada utilizando recursos limitados y *software* libre. La solución desarrollada cumple con los objetivos propuestos y establece una base sólida para futuras implementaciones y mejoras, respondiendo a la creciente demanda de capacidades computacionales en los ámbitos académico y profesional.

6.2. Líneas futuras de trabajo

En el marco del trabajo desarrollado, y considerando tanto las limitaciones actuales como las oportunidades de mejora, se proponen las siguientes líneas de desarrollo futuro para consolidar y ampliar el alcance del proyecto.

En primer lugar, se considera prioritario migrar la infraestructura actual, basada en máquinas virtuales, a nodos físicos. Esta transición permitirá obtener métricas de rendimiento más representativas en un entorno HPC real. Asimismo, se prevé la incorporación de aceleradores GPU, cuya integración con el planificador Slurm permitirá evaluar la gestión eficiente de recursos heterogéneos y ampliar la variedad de cargas de trabajo soportadas.

También se plantea automatizar la instalación del servidor y sus servicios, con el objetivo de agilizar el despliegue, reducir posibles errores de configuración y facilitar la replicación del entorno en otros sistemas. Por otro lado, se contempla la implementación de contenedores en los nodos del clúster HPC, con el objetivo de garantizar entornos más reproducibles y facilitar la portabilidad de las aplicaciones.

Adicionalmente, se propone la incorporación de un sistema de redundancia que refuerce la disponibilidad del entorno de ejecución de Slurm, así como la instalación de un sistema de almacenamiento distribuido que responda a los requisitos de escalabilidad y alta disponibilidad del entorno.

En cuanto a la herramienta de despliegue automatizado, se contempla su evolución hacia una arquitectura más modular y extensible, junto con el desarrollo de una interfaz de usuario que simplifique aún más su utilización. Además, se plantea incorporar un mecanismo de descarga automática de los resultados generados en el entorno HPC hacia el entorno local del usuario.

Finalmente, como parte de los trabajos futuros, se prevé la preparación de un artículo científico para su presentación en un congreso. Este artículo tendrá como objetivo difundir los resultados obtenidos en el TFM, con especial énfasis en la innovación

introducida mediante la herramienta de automatización desarrollada, destacando su aplicabilidad, escalabilidad y contribución a la gestión eficiente de entornos HPC.

Declaración sobre el uso de tecnologías de IA generativa y asistencia por IA en el proceso de redacción

Durante la elaboración de este trabajo, el autor ha empleado modelos de lenguaje de gran tamaño (LLM), como ChatGPT, así como herramientas basadas en inteligencia artificial, como DeepL Write, con el objetivo de mejorar la claridad y la legibilidad del texto. Todas las sugerencias generadas por estas herramientas fueron revisadas críticamente y adaptadas por el autor, quien asume plena responsabilidad por el contenido final presentado.

Bibliografía

Advanced Micro Devices, Inc. (2023). 4th Gen AMD EPYC™ Processors Architecture [Accedido el 4 de junio de 2025]. *AMD*. Consultado el 4 de junio de 2025, desde https://www.amd.com/content/dam/amd/en/documents/products/epyc/4th-gen-epyc-processor-architecture-white-paper.pdf

Altair Engineering Inc. (2025). Altair PBS Professional [Accedido el 22 de mayo de 2025]. https://altair.com.es/pbs-professional/

Amazon Web Services. (2025). AWS ParallelCluster [Accedido el 22 de mayo de 2025]. https://aws.amazon.com/es/hpc/parallelcluster/

AMD. (2024). Descripción general del procesador AMD EPYC [Accedido el 4 de junio de 2025]. Consultado el 4 de junio de 2025, desde https://www.amd.com/es/partner/articles/epyc-processor-overview.html

Barcelona Supercomputing Center. (2025). Arranca MareNostrum 5: el nuevo supercomputador europeo instalado en el BSC [Accedido el 28 de junio de 2025].

Broadcom Inc. (2024). *Understanding the NAT Device* [Accedido el 29 de mayo de 2025]. Broadcom. https://techdocs.broadcom.com/us/en/vmware-cis/desktop-hypervisors/workstation-pro/17-0/using-vmware-workstation-pro/configuring-network-connections/configuring-network-address-translation-1/features-and-limitations-of-nat-configurations/understanding-the-nat-device.html

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. https://arxiv.org/abs/2005.14165

Butterfield, A., & Ngondi, G. E. (2016). *A dictionary of computer science*. Oxford University Press.

Calzati, S. (2025). An ecosystemic view on information, data, and knowledge: insights on agential AI and relational ethics. *AI and Ethics*, 1-14.

Chen, J., Hoops, S., Mortveit, H. S., Lewis, B. L., Machi, D., Bhattacharya, P., Venkatramanan, S., Wilson, M. L., Barrett, C. L., & Marathe, M. V. (2025). Epihiper—A high performance computational modeling framework to support epidemic science. *PNAS nexus*, *4*(1), pgae557.

Chris Dunlap. (2020). MUNGE (MUNGE Uid 'N' Gid Emporium) – Home [Accedido el 27 de junio de 2025]. https://github.com/dun/munge/wiki

Civera, L. G. (2018, noviembre). Implementación de un clúster HPC en el Centro de Estudios de Física del Cosmos de Aragón [Publicado en el sitio web del autor]. https://www.luisguillen.com/posts/2018/11/implementacion-cluster-hpc-ubuntu/resources/tfc-luisguillenc.pdf

Corporation, I. (2024). *Understanding Intel Xeon Scalable Processors: Numbers and Suffixes* [Accedido el 2 de mayo de 2025]. https://www.intel.com/content/

www/us/en/support/articles/000059657/processors/intel-xeon-processors.html
Corporation, I. (2025). *Procesadores Intel® – Información y Comparativas* [Accedido el 2 de mayo de 2025]. https://www.intel.la/content/www/xl/es/products/details/processors.html
Corporation, M. (2024). Remote - SSH [Accedido el 12 de junio de 2025]. https://code.visualstudio.com/docs/remote/ssh
Debian Wiki contributors. (2025). msmtp - Debian Wiki [Accedido el 30 de mayo de 2025]. https://wiki.debian.org/msmtp
Fan, Y. (2021). Job Scheduling in High Performance Computing. https://arxiv.org/abs/2109.09269
Feitelson, D. G. (1997). Job scheduling in multiprogrammed parallel systems.
Fernández Palau, M. (2023). *Migración de un clúster HPC* [B.S. thesis]. Universitat Politècnica de Catalunya.
FireWorks Development Team. (2024). *FireWorks* [Accedido el 1 de junio de 2025]. https://materialsproject.github.io/fireworks/
Free Software Foundation. (2025). Licenses - Free Software Foundation [Accedido el 27 de junio de 2025]. https://www.gnu.org/licenses/license-list.html
FreeIPA Project. (2025a). About FreeIPA [Accedido el 23 de mayo de 2025]. https://www.freeipa.org/About.html
FreeIPA Project. (2025b). FreeIPA Documentation [Accedido el 27 de junio de 2025]. https://www.freeipa.org/page/Documentation.html
Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., De Supinski, B. R., & Futral, S. (2015). The Spack package manager: bringing order to HPC software chaos. *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 1-12.
Ganglia Project. (2025). Ganglia Monitor Core Wiki [Accedido el 27 de junio de 2025]. https://github.com/ganglia/monitor-core/wiki
García Peña, K. E., et al. (2021). Administración de nodos de cómputo de altas prestaciones.
Geimer, M., Hoste, K., & McLay, R. (2014). Modern scientific software management using easybuild and lmod. *2014 First International Workshop on HPC User Support Tools*, 41-51.
Ghedira, K., Khamessi, O., Hkimi, C., Kamoun, S., Dhamer, N., Daassi, K., Ben Salah, W., Othman, H., Belhadj, W., & Ghorbal, Y. (2024). Design and implementation of a scalable high-performance computing (HPC) cluster for omics data analysis: achievements, challenges and recommendations in LMICs. *GigaScience*, *13*, giae060.
Google Cloud. (2025). Cluster Toolkit: Overview [Accedido el 22 de mayo de 2025]. https://cloud.google.com/cluster-toolkit/docs/overview
Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2003). *Introduction to Parallel Computing, Second Edition*. Addison-Wesley Professional.
Hajkowicz, S., Sanderson, C., Karimi, S., Bratanova, A., & Naughtin, C. (2023). Artificial intelligence adoption in the physical sciences, natural sciences, life sciences, social sciences and the arts and humanities: A bibliometric analysis of research publications from 1960-2021. *Technology in Society*, *74*, 102260.
Hewlett Packard Enterprise. (2024). ¿Qué es la computación de alto rendimiento (HPC)? https://www.hpe.com/es/es/what-is/high-performance-computing.html
Hewlett Packard Enterprise. (2025). Altair PBS Professional [Accedido el 27 de junio de 2025]. Consultado el 27 de junio de 2025, desde https://buy.hpe.com/

es/es/software/third-party-software/third-party-software/third-party-software/altair-pbs-professional/p/1010144164

Hofert, M. (2024). High Performance Computing Cluster Setup: A Tutorial. *Journal of Data Science*, 1-22. https://doi.org/10.6339/24-JDS1159

Hurrell, J. W., Holland, M. M., Gent, P. R., Ghan, S., Kay, J. E., Kushner, P. J., Lamarque, J.-F., Large, W. G., Lawrence, D., Lindsay, K., et al. (2013). The community earth system model: a framework for collaborative research. *Bulletin of the American Meteorological Society*, *94*(9), 1339-1360.

JetBrains. (2024). Remote Development with PyCharm [Accedido el 12 de junio de 2025]. https://www.jetbrains.com/help/pycharm/remote.html

Kinghorn, D. (2025). *Hardware Recommendations for Scientific Computing* [Accedido el 10 de junio de 2025]. Puget Systems. https://www.pugetsystems.com/solutions/ai-and-hpc-workstations/scientific-computing/hardware-recommendations/

Kravcenko, V. (2024). *pipreqs: Generate pip requirements file based on imports in your project* [Accedido el 26 de mayo de 2025]. https://pypi.org/project/pipreqs/

Kumara, D., Theboa, L. A., & Memona, S. (2018). Design and Implementation of High Performance Computing (HPC) Cluster. *Academic Journal of Management Sciences ISSN*, *2305*, 2864.

Li, J., Michelogiannakis, G., Cook, B., Cooray, D., & Chen, Y. (2023). Analyzing Resource Utilization in an HPC System: A Case Study of NERSC Perlmutter. https://arxiv.org/abs/2301.05145

Lu, P.-J., Lai, M.-C., & Chang, J.-S. (2022). A survey of high-performance interconnection networks in high-performance computer systems. *Electronics*, *11*(9), 1369.

Lucendo-Monedero, Á. L., Ruiz-Rodríguez, F., & González-Relaño, R. (2023). The information society and socio-economic sustainability in european regions. Spatio-temporal changes between 2011 and 2020. *Technology in Society*, *75*, 102337. https://doi.org/https://doi.org/10.1016/j.techsoc.2023.102337

Lüttgau, J., Kuhn, M., Duwe, K., Alforov, Y., Betke, E., Kunkel, J., & Ludwig, T. (2018). Survey of storage systems for high-performance computing. *Supercomputing Frontiers and Innovations*, *5*(1).

MacLachlan, G., Hurlburt, J., Suarez, M., Wong, K. L., Burke, W., Lewis, T., Gallo, A., Flidr, J., Gabiam, R., Nicholas, J., & Ensor, B. (2020). Building a Shared Resource HPC Center Across University Schools and Institutes: A Case Study. https://arxiv.org/abs/2003.13629

MariaDB Foundation. (s.f.). MariaDB: Open source database [Accedido el 30 de mayo de 2025].

Massie, M. L., Chun, B. N., & Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, *30*(7), 817-840.

McLay, R. (2024). Lmod: A New Environment Module System [Accedido el 02 de mayo de 2025].

Microsoft Corporation. (2025a). Azure CycleCloud: Descripción general [Accedido el 22 de mayo de 2025]. https://learn.microsoft.com/es-es/azure/cyclecloud/overview?view=cyclecloud-8

Microsoft Corporation. (2025b). AzureTRE CycleCloud Shared-Service Template [Accedido el 27 de junio de 2025]. Consultado el 27 de junio de 2025, desde https://microsoft.github.io/AzureTRE/v0.20.0/tre-templates/shared-services/cyclecloud/

Nabulsi, N., & Hamidi, A. (2021). Towards modern authentication in a large-scale compute Installation [ISSN 2348-120X (online)]. *International Journal of Computer Science and Information Technology Research*.

Netto, M. A. S., Calheiros, R. N., Rodrigues, E. R., Cunha, R. L. F., & Buyya, R. (2018). HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges. *ACM Computing Surveys*, *51*(1), 1-29. https://doi.org/10.1145/3150224

Nextflow Project. (2024). *Nextflow - Data-driven computational pipelines* [Accedido el 1 de junio de 2025]. https://www.nextflow.io

NVIDIA Corporation. (2024a). *CUDA GPUs* [Accedido el 7 de junio de 2025]. https://developer.nvidia.com/cuda-gpus

NVIDIA Corporation. (2024b). *CUDA Legacy GPUs* [Accedido el 7 de junio de 2025]. https://developer.nvidia.com/cuda-legacy-gpus

NVIDIA Corporation. (2025). *NVIDIA Base Command Manager Documentation* [Accedido el 22 de mayo de 2025]. https://docs.nvidia.com/base-command-manager/

Ohio Supercomputer Center. (2025). *Administer Open OnDemand* [Accedido el 29 de mayo de 2025]. Ohio Supercomputer Center. https://openondemand.org/administer-open-ondemand

Open Source Initiative. (2025). Licenses by Name - Open Source Initiative [Accedido el 27 de junio de 2025]. https://opensource.org/licenses/

OpenHPC Community. (2025). OpenHPC: Community building blocks for HPC systems [Accedido el 22 de mayo de 2025]. https://openhpc.community/

Press, A. S. (2024, abril). *91 % of AI papers used NVIDIA in 2024* [Accedido el 7 de junio de 2025]. https://press.airstreet.com/p/compute-index-2024

Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach* (7.ª ed.). McGraw-Hill.

PuTTY Developers. (2024). *PuTTY: A Free SSH and Telnet Client* [Accedido el 31 de mayo de 2025]. https://www.putty.org/

Ramesh, R. S. (2024). Scalable Systems and Software Architectures for High-Performance Computing on cloud platforms. *arXiv preprint arXiv:2408.10281*.

Red Hat, Inc. (2025). Cygwin: A large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows [Accedido el 25 de mayo de 2025].

Research, H. (2023). Slurm Remains Top Resource Manager [Accedido el 23 de mayo de 2025]. https://hyperionresearch.com/product/slurm-remains-top-resource-manager/

Rocky Linux. (2025a). *Current Supported Releases* [Accedido el 25 de mayo de 2025]. https://wiki.rockylinux.org/rocky/version/#current-supported-releases

Rocky Linux. (2025b). Support Providers [Accedido el 25 de mayo de 2025].

Rocky Linux Documentation Team. (2024). Repository Layout [Accedido el 29 de mayo de 2025]. https://wiki.rockylinux.org/rocky/repo/

SchedMD. (2024a). Slurm Workload Manager - Scheduler Parameters [Accedido el 02 de mayo de 2025].

SchedMD. (2024b). *Slurm Workload Manager Overview* [Accedido el 23 de mayo de 2025]. https://slurm.schedmd.com/overview.html

SchedMD LLC. (2025). Slurm Workload Manager – Authentication Plugins [Accedido el 27 de junio de 2025]. https://slurm.schedmd.com/authentication.html

Server-World. (2025). Linux & Open Source HowTo – Server-World [Accedido el 27 de junio de 2025]. https://www.server-world.info/

Silvano, C., Ielmini, D., Ferrandi, F., Fiorin, L., Curzel, S., Benini, L., Conti, F., Garofalo, A., Zambelli, C., Calore, E., Schifano, S., Palesi, M., Ascia, G., Patti, D., Petra, N., De Caro, D., Lavagno, L., Urso, T., Cardellini, V., ... Perri, S. (2025). A Survey on Deep Learning Hardware Accelerators for Heterogeneous HPC Platforms. *ACM Comput. Surv.*, *57*(11). https://doi.org/10.1145/3729215

Smuts, H., & Van der Merwe, A. (2022). Knowledge Management in Society 5.0: A Sustainability Perspective. *Sustainability*, *14*(11). https://doi.org/10.3390/su14116878

Snakemake Development Team. (2024). *Snakemake Workflow Management System* [Accedido el 1 de junio de 2025]. https://snakemake.readthedocs.io

Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education Limited.

Sterling, T., Brodowicz, M., & Anderson, M. (2017). *High performance computing: modern systems and practices*. Morgan Kaufmann Publishers Inc.

Sterling, T. L. (2002). *Beowulf cluster computing with Linux*. MIT press.

The MUNGE Project. (2025). MUNGE Uid 'N' Gid Emporium [Accedido el 30 de mayo de 2025]. https://dun.github.io/munge/

The NTP Project. (2024). Network Time Protocol (NTP) [Accedido el 29 de mayo de 2025]. https://www.ntp.org

tmux. (s.f.). *tmux wiki* [Accedido el 28 de mayo de 2025]. https://github.com/tmux/tmux/wiki

TOP500. (2025). Top500 List of the World's Fastest Supercomputers [Accedido el 26 de abril de 2025].

Tridgell, A. (1996). rsync - a fast, versatile, remote (and local) file-copying tool [Accedido el 25 de mayo de 2025].

VMware, Inc. (2025). *Fusion and Workstation* [Accedido el 25 de mayo de 2025]. https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

Wael, A., & Madi, A. (2025). Accelerating Artificial Intelligence: The Role of GPUs in Deep Learning and Computational Advancements. *East Journal of Engineering*, *1*(1), 31-46.

Wandinger, M. (2017). Intel Xeon X5570 Prozessor [Licenciado bajo CC BY-SA 3.0. Accedido el 28 de junio de 2025]. https://commons.wikimedia.org/wiki/File:Xeon-x5570.jpg

Warewulf Project. (2025). Warewulf User Guide v4.6.x [Accedido el 22 de mayo de 2025]. https://warewulf.org/docs/v4.6.x/

Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., & Stuart, J. M. (2013). The cancer genome atlas pan-cancer analysis project. *Nature genetics*, *45*(10), 1113-1120.

xCAT Project. (2025). xCAT: Extreme Cloud Administration Toolkit [Accedido el 22 de mayo de 2025]. https://xcat-docs.readthedocs.io/en/stable/

Apéndice A

LISTA DE SIGLAS Y ACRÓNIMOS

Esta sección presenta las siglas y acrónimos utilizados a lo largo del TFM, relacionados con el desarrollo del trabajo. Su definición facilita la comprensión del contenido técnico.

CLE Cray Linux Environment

CPU Central Processing Unit (Unidad Central de Procesamiento)

DNS Domain Name System (Sistema de Nombres de Dominio)

FIFO First In, First Out (Primero en entrar, primero en salir)

FLOPS Floating Point Operations Per Second (Operaciones de Punto Flotante por Segundo)

FreeIPA Free Identity, Policy and Audit

GPT-3 Generative Pretrained Transformer 3 (Transformador Generativo Preentrenado 3)

GPU Graphics Processing Unit (Unidad de Procesamiento Gráfico)

HPC High Performance Computing (Computación de Altas Prestaciones)

HPCaaS High Performance Computing as a Service (Computación de Altas Prestaciones como Servicio)

HPE Cray OS HPE Cray Operating System

IA Inteligencia Artificial (Artificial Intelligence)

IaaS Infrastructure as a Service (Infraestructura como Servicio)

IDE Integrated Development Environment (Entorno de Desarrollo Integrado)

LDAP Lightweight Directory Access Protocol

LLM Large Language Model (Modelo de Lenguaje de Gran Escala)

MIT Massachusetts Institute of Technology (Instituto Tecnológico de Massachusetts)

NAT Network Address Translation (traducción de direcciones de red)

NFS Network File System

NIS Network Information Service (Servicio de Información de Red)

NSCD Name Service Cache Daemon

NTP Network Time Protocol (protocolo de tiempo de red)

PaaS Platform as a Service (Plataforma como Servicio)

PBS Pro Portable Batch System Professional (Sistema Profesional de Lotes Portátil)

PCIe Peripheral Component Interconnect Express (Interconexión de Componentes Periféricos Express)

RAM Random Access Memory (Memoria de Acceso Aleatorio)

RHEL Red Hat Enterprise Linux

SaaS Software as a Service (Software como Servicio)

SLES SUSE Linux Enterprise Server

Slurm Simple Linux Utility for Resource Management (Utilidad de Linux para la Gestión de Recursos)

SMTP Simple Mail Transfer Protocol

SO Sistema Operativo

TIC Tecnologías de la Información y la Comunicación

VS Code Visual Studio Code

xCAT Extreme Cloud Administration Toolkit

Apéndice B

MANUAL DE ADMINISTRACIÓN DEL SERVIDOR

B.1. Información general

En este documento encontrará información relevante relacionada con la administración básica del servidor.

B.2. Administración de los recursos

Para facilitar la administración se encuentran habilitadas algunas herramientas de administración web.

B.2.1. Cockpit

Para la gestión general de los servidores, se puede acceder a una interfaz gráfica web para administrar servidores Linux a través de los siguientes enlaces.

Para administrar el nodo maestro desde el enlace https://master.hpc.tfm:9090/
Para administrar el nodo01 desde el enlace https://nodo01.hpc.tfm:9090/
Para administrar el nodo02 desde el enlace https://nodo02.hpc.tfm:9090/
Para administrar el nodo03 desde el enlace https://nodo03.hpc.tfm:9090/

B.2.2. FreeIPA

Para la gestión general de los servidores, el servidor FreeIPA proporciona una interfaz gráfica web que permite administrar usuarios, grupos, políticas de autenticación, certificados, control de acceso y servicios relacionados con la identidad en sistemas Linux. Esta interfaz es accesible a través del siguiente enlace: https://master.hpc.tfm/ipa/ui/

B.2.3. Ganglia

Para la monitorización del clúster, el sistema **Ganglia** proporciona una interfaz gráfica web que permite visualizar en tiempo real el estado de los nodos, el uso de

CPU, memoria, red y otras métricas relevantes de rendimiento. Esta herramienta facilita el seguimiento del comportamiento del sistema y la detección temprana de posibles cuellos de botella. La interfaz está disponible en el siguiente enlace: https://master.hpc.tfm/ganglia/

B.3. Procesos comunes de administración

Esta sección describe las tareas administrativas más frecuentes en el clúster. Todas las acciones requieren privilegios de administrador y deben ejecutarse en el nodo maestro.

B.3.1. Creación de usuario

Para crear un nuevo usuario dentro de FreeIPA, primero se debe autenticar como administrador. A continuación, se muestra un ejemplo del procedimiento.

```
[root@master ~]$ kinit admin
[root@master ~]$ipa user-add luisperez --first=Luis --last=Perez --password
Password:
Enter Password again to verify:
----------------------
Added user "luisperez"
----------------------
  User login: luisperez
  First name: Luis
  Last name: Perez
  Full name: Luis Perez
  Display name: Luis Perez
  Initials: LP
  Home directory: /home/luisperez
  GECOS: Luis Perez
  Login shell: /bin/bash
  Principal name: luisperez@HPC.TFM
  Principal alias: luisperez@HPC.TFM
  User password expiration: 20250526221554Z
  Email address: luisperez@hpc.tfm
  UID: 925400005
  GID: 925400005
  Password: True
  Member of groups: ipausers
  Kerberos keys available: True
```

B.3.2. Cambio de contraseña

Para cambiar la contraseña de un usuario dentro de FreeIPA, primero se debe autenticar como administrador. A continuación, se muestra un ejemplo del procedimiento.

```
[root@master ~]$ kinit admin
[root@master ~]$ ipa passwd luisperez
New Password:
Enter New Password again to verify:
---------------------------------------
```

```
Changed password for "luisperez@HPC.TFM"
----------------------------------------
```

B.3.3. Asignar grupo a usuario

Para asociar a un usuario con un grupo específico del clúster, como `investigadores` o `invitados`, se utiliza el siguiente comando. La pertenencia a un grupo determina las particiones de Slurm a las que puede acceder y las restricciones asociadas (como el tiempo máximo de ejecución o la prioridad de los trabajos).

```
[root@master ~]$ ipa group-add-member investigadores --users=juanperez
```

B.3.4. Asignar espacio cuota de disco a usuario

Para gestionar el espacio asignado a un usuario en su carpeta personal, ubicada dentro del directorio `/home`, se utiliza el siguiente comando, que permite establecer un límite de uso del disco.

```
[root@master ~]$xfs_quota -x -c 'limit bsoft=25g bhard=30g luisperez' /home
```

B.3.5. Bloqueo o desbloqueo de cuentas

Para bloquear o desbloquear una cuenta de usuario en FreeIPA, primero se debe autenticar como administrador. A continuación, se muestra un ejemplo para bloquear una cuenta.

```
[root@master ~]$ kinit admin
[root@master ~]$ ipa user-disable luisperez
```

A continuación, se muestra un ejemplo para desbloquear una cuenta.

```
[root@master ~]$ kinit admin
[root@master ~]$ ipa user-enable luisperez
```

B.4. Gestión de software con EasyBuild

Para buscar recetas disponibles en los repositorios gestionados por EasyBuild, se utiliza el siguiente comando. A continuación, se muestra un ejemplo del procedimiento.

```
[root@master ~]$ su - easybuilder
[easybuilder@master ~]$ eb -S minicon
== found valid index for /usr/local/easybuild/easyconfigs, so using it...
CFGS1=/usr/local/easybuild/easyconfigs/m/Miniconda3
 * $CFGS1/Miniconda3-4.8.3.eb
 * $CFGS1/Miniconda3-4.9.2.eb
 * $CFGS1/Miniconda3-4.12.0.eb
 * $CFGS1/Miniconda3-22.11.1-1.eb
 * $CFGS1/Miniconda3-23.5.2-0.eb
```

```
 * $CFGS1/Miniconda3-23.9.0-0.eb
 * $CFGS1/Miniconda3-23.10.0-1.eb
 * $CFGS1/Miniconda3-24.7.1-0.eb
[easybuilder@master ~]$
```

Para instalar un software específico mediante EasyBuild, se utiliza la ruta a su receta junto con la opción `-robot`, que permite resolver automáticamente las dependencias. A continuación, se muestra un ejemplo del procedimiento.

```
[root@master ~]$ su - easybuilder
[easybuilder@master ~]$ eb
/usr/local/easybuild/easyconfigs/m/Miniconda3/Miniconda3-24.7.1-0.eb --robot
```

B.5. Gestión administrativa de Slurm

Visualizar particiones del clúster configuradas en el Slurm

```
[root@master ~]$ scontrol show partition
```

Mostrar configuración activa del Slurm

```
[root@master ~]$ scontrol show config
```

Marcar un nodo como no disponible

```
[root@master ~]$ scontrol update NodeName=nodo01 State=DOWN
Reason="Mantenimiento"
```

Habilitar un nodo

```
[root@master ~]$ scontrol update NodeName=node01 State=RESUM
```

Apéndice C

MANUAL DEL USUARIO DE UTILIZACIÓN DEL CLÚSTER

C.1. Información general

En este documento encontrará información relevante relacionada con la utilización básica del servidor HPC.

C.2. Recursos disponibles

El clúster cuenta con un total de 4 servidores con las siguientes características:

Características	Maestro	Nodo01	Nodo02	Nodo03
CPU (vCPUs)	8	4	4	4
Memoria RAM (GB)	6	4	4	4
Almacenamiento SSD (GB)	200	100	100	100
Discos secundarios SSD (GB)	300	-	-	-

TABLA C.1: Resumen de características de hardware virtualizado por nodo en el clúster.

Todos los servidores operan con Rocky Linux 8.10.

C.3. Cómo conectarse al clúster

Existen distintas formas de acceder al clúster, según el sistema operativo y la herramienta utilizada. A continuación, se detallan los pasos necesarios.

C.3.1. Acceso al clúster mediante SSH

Para ingresar a los recursos, es necesario poseer un cliente SSH.

Nota

Es importante sustituir la contraseña temporal por una personalizada para proteger adecuadamente su cuenta. Para ello, utilice el siguiente comando.

```
passwd usuario
```

Precaución

Solo está autorizada la realización de tareas administrativas en el servidor maestro, empleando comandos como `ls`, `cd`, `pwd`, `rm` o aquellos asociados a `Slurm`. Ejecutar cálculos en este servidor podría perjudicar el trabajo de los demás usuarios.

C.3.1.1. Acceso desde sistemas Windows

- **MobaXterm:** Es un cliente SSH que facilita la conexión remota a otros sistemas. Puede descargarlo o consultar más información en la página oficial de MobaXterm.
- **Windows Terminal:** Es una aplicación de terminal moderna para usuarios que permite ejecutar comandos de una forma rápida y cómoda. Puede consultar más información en la página oficial de Windows.
- **PuTTY:** Es un cliente SSH y telnet gratuito que permite establecer conexiones remotas de forma segura. Puede descargarlo o consultar más información en la página oficial de PuTTY.
- **Windows Subsystem for Linux (WSL):** Es un recurso que permite utilizar distribuciones Linux directamente en Windows. Puede descargarlo o consultar más información en la página oficial de Windows.

C.3.1.2. Acceso desde sistemas Linux y MacOS

- **Terminal:** Es una utilidad nativa de Linux y MacOS que brinda la posibilidad de ejecutar comandos directamente mediante una interfaz de línea de comandos.

```
ssh -X nombre_apellido@direccion_ip -p 22
```

C.3.2. Acceso al clúster mediante Open OnDemand

Los usuarios también disponen de la herramienta Open OnDemand, que proporciona una interfaz web desde la cual es posible acceder al clúster y utilizar múltiples funcionalidades de forma sencilla. Esta plataforma facilita tareas que serían más complejas de realizar desde la terminal. El acceso a la plataforma se realiza a través del siguiente enlace: https://ood.hpc.tfm/. En la Figura C.1 se muestra la pantalla de inicio de sesión de Open OnDemand.

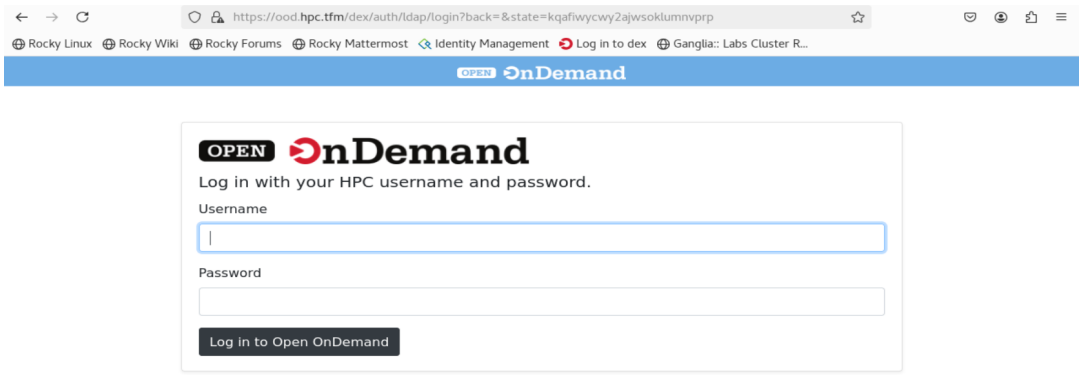

FIGURA C.1: Inicio de sesión en Open OnDemand.

A continuación en la tabla C.2, se describen las principales aplicaciones disponibles en la plataforma Open OnDemand.

Aplicación	Descripción funcional
Dashboard	Página principal que actúa como centro de control para acceder y navegar entre las diferentes aplicaciones del entorno de trabajo.
File Browser	Herramienta de navegación que permite explorar la estructura de directorios, cargar y descargar archivos, además de realizar operaciones de organización y renombrado.
File Editor	Editor de texto avanzado que incluye diferentes temas visuales, combinaciones de teclas personalizables y resaltado de sintaxis para facilitar la programación.
Terminal	Interfaz de línea de comandos que establece una sesión SSH directa con el nodo de login, permitiendo el acceso remoto desde el navegador web.
My Jobs	Módulo dedicado a la creación, configuración y envío de trabajos computacionales al sistema de colas del clúster.
Active Jobs	Panel de monitoreo que muestra el estado actual de los trabajos en la cola, incluyendo aquellos que están ejecutándose y los que están en espera.

TABLA C.2: Aplicaciones disponibles en Open OnDemand y su funcionalidad principal.

C.4. Manual de Comandos de Linux

Nota

En caso de tener el conocimiento básico de Linux, puede omitir esta sección.

C.4.1. Navegación en el Sistema de Archivos

`ls`: Muestra el contenido de un directorio.
Uso: `ls [opciones] [directorio]`
Ejemplo: `ls -l`

`cd`: Cambia de directorio.
Uso: `cd [directorio]`
Ejemplo: `cd /home/user`

`cd -`: Cambia al directorio posicionado anteriormente.
Uso: `cd -`

`cd ..`: Navega un nivel hacia arriba en el árbol de directorios.
Uso: `cd ..`

C.4.2. Visualización y Edición de Archivos

`cat`: Permite visualizar el contenido completo de un archivo.
Uso: `cat [archivo]`
Ejemplo: `cat /etc/fstab`

`tail`: Muestra las últimas líneas de un archivo.
Uso: `tail [opciones] [archivo]`
Ejemplo: `tail -n 10 /var/log/syslog` (últimas 10 líneas)
Opción útil: `-f` recupera el archivo en tiempo real.

`less`: Visualiza un archivo de forma interactiva.
Uso: `less [archivo]`
Ejemplo: `less /var/log/syslog`

`nano`: Editor de texto sencillo.
Uso: `nano [archivo]`
Ejemplo: `nano /etc/hosts`

`vim`: Editor de texto avanzado basado en *vi*.
Uso: `vim [archivo]`
Ejemplo: `vim /etc/hosts`

C.4.3. Gestión de Archivos y Directorios

`tar`: Archiva y comprime archivos.
Crear archivo: `tar -cvf archivo.tar directorio/`
Extraer archivo: `tar -xvf archivo.tar`
Comprimir con gzip: `tar -czvf archivo.tar.gz directorio/`
Extraer archivo comprimido: `tar -xzvf archivo.tar.gz`
Ejemplo: `tar -czvf backup.tar.gz /home/user`

`du`: Muestra el uso de espacio en disco.
Uso: `du [opciones] [archivo/directorio]`
Ejemplo: `du -sh /home/user`

`scp`: Copia archivos entre hosts de forma segura.
Uso: `scp [opciones] origen destino`
Ejemplos:
Copiar archivo a host remoto:
`scp archivo.txt usuario@servidor:/ruta/destino`
Copiar archivo desde host remoto:
`scp usuario@servidor:/ruta/origen/archivo.txt /ruta/local/destino`
Copiar directorio recursivamente:
`scp -r directorio usuario@servidor:/ruta/destino`

C.4.4. Multiplexación de Terminales

`tmux`: Permite gestionar múltiples sesiones de terminal.
Uso: `tmux [comando]`
Ejemplos:
Iniciar nueva sesión: `tmux`
Reanudar sesión: `tmux attach-session -t nombre_sesion`
Separar sesión: `Ctrl-b d`

C.4.5. Ejecución y Supervisión de Comandos

`watch`: Ejecuta un comando periódicamente y muestra su salida.
Uso: `watch [opciones] comando`
Ejemplo: `watch -n 5 ls -l` (ejecuta `ls -l` cada 5 segundos)

C.5. Utilización de módulos de EasyBuild

Para que los usuarios puedan utilizar los paquetes de software disponibles, tienen las siguientes opciones:

Mostrar vista resumida de los módulos disponibles, mostrando cuántas versiones existen para cada uno.

```
[usuario@master ~]$ module overview

--------------------- /usr/local/easybuild/modules/all --------------------
Autoconf  (2)   DB      (1)   Miniconda3 (2)   Perl   (3)   UCX
Automake  (2)   GCC     (1)   OpenMPI    (1)   Python (2)   UnZip
Autotools (2)   GCCcore (3)   OpenSSL    (2)   SQLite (2)   XZ
Bison     (4)   M4      (5)   PMIx       (1)   Tcl    (2)   binutils

------------------------- /usr/share/modulefiles ------------------------
pmi (1)

------------------ /usr/share/lmod/lmod/modulefiles/Core ------------------
lmod (1)   settarg (1)
```

Se visualizarán todas las versiones disponibles de los módulos que puedes cargar en el entorno.

```
[usuario@master ~]$ module avail

--------------------- /usr/local/easybuild/modules/all ---------------------
   Autoconf/2.69-GCCcore-10.2.0           M4/1.4.19-GCCcore-13.3.0
   gettext/0.22.5                    (D)  ncurses/6.2-GCCcore-11.2.0
   Automake/1.16.2-GCCcore-10.2.0         Miniconda3/23.10.0-1
   help2man/1.47.16-GCCcore-10.2.0        ncurses/6.5-GCCcore-13.3.0
   Automake/1.16.5                   (D)  Miniconda3/24.7.1-0           (D)
   Autotools/20200321-GCCcore-10.2.0      OpenMPI/4.1.2-GCC-10.2.0
   Bison/3.8.2                       (D)  Perl/5.32.0-GCCcore-10.2.0
   DB/18.1.40-GCCcore-10.2.0              Perl/5.38.0                   (D)
   GCC/10.2.0                             Python/2.7.18-GCCcore-11.2.0-bare
   libreadline/8.1-GCCcore-11.2.0         zlib/1.2.11-GCCcore-11.2.0
   GCCcore/10.2.0                         Python/3.12.3-GCCcore-13.3.0  (D)
   libreadline/8.2-GCCcore-13.3.0    (D)  zlib/1.2.11
   GCCcore/11.2.0                         SQLite/3.36-GCCcore-11.2.0
   libtool/2.4.6-GCCcore-10.2.0           zlib/1.2.13
   GCCcore/13.3.0                    (D)  SQLite/3.45.3-GCCcore-13.3.0  (D)

....
```

Para cargar una versión específica de los módulos listados.

```
[usr@master ~]$ module load Miniconda3/23.10.0-1 SQLite/3.36-GCCcore-11.2.0
```

Para mostrar los módulos que se tienen cargados actualmente.

```
[usuario@master ~]$ module list

Currently Loaded Modules:
  1) Miniconda3/23.10.0-1         2) GCCcore/11.2.0
  3) ncurses/6.2-GCCcore-11.2.0   4) libreadline/8.1-GCCcore-11.2.0
  5) zlib/1.2.11-GCCcore-11.2.0   6) Tcl/8.6.11-GCCcore-11.2.0
  7) SQLite/3.36-GCCcore-11.2.0
```

Para limpiar el entorno de ejecución.

```
[usuario@master ~]$ module purge
[usuario@master ~]$ module list
No modules loaded
[usuario@master ~]$
```

C.6. Gestor de trabajos

En el clúster HPC, los usuarios deben enviar sus trabajos a una cola de trabajo que organiza su ejecución en los nodos disponibles. La gestión de estos trabajos se realiza mediante Slurm, el sistema de planificación utilizado. Todas las solicitudes son recibidas por el nodo maestro, que se encarga de asignar los trabajos a los nodos de cómputo de acuerdo con los recursos disponibles.

C.6.1. Slurm

Los comandos disponibles para gestionar trabajos en el clúster son:

`squeue`: Comando que se ha utilizado con el fin de permitir la visualización del estado actual de los trabajos en el planificador, incluyendo los que se encuentran en ejecución.

```
[usuario@master ~]$ squeue
 JOBID PARTITION          NAME     USER     ST TIME  NODES NODELIST(REASON)
   443     investigadores   job     usuario  R       0:10      1 node01
   444     investigadores   job2    usuario  R       0:04      1 node03
```

`scancel`: Comando que se ha utilizado con el fin de permitir la finalización anticipada de un trabajo que se encuentra en proceso de ejecución. Por ejemplo, seguidamente se muestra la cancelación del trabajo con el identificador 443.

```
[usuario@master ~]$ scancel 443
```

`sinfo`: Comando que se ha utilizado con el fin de detallar información sobre el estado actual de los nodos, en relación con la disponibilidad y el estado operativo de los mismos.

```
[usuario@master ~]$ sinfo
PARTITION        AVAIL  TIMELIMIT  NODES  STATE NODELIST
investigadores*    up   7-00:00:00     1  down* nodo02
investigadores*    up   7-00:00:00     1  alloc nodo01
investigadores*    up   7-00:00:00     1   idle nodo03
```

`scontrol`: Comando que se ha utilizado con el fin de permitir la administración del sistema de planificación de trabajos, incluyendo la consulta detallada de trabajos activos o en ejecución.

```
[usuario@master ~]$ scontrol show job 443
```

`sacct`: Comando que se ha utilizado con el fin de visualizar información histórica de los trabajos que hayan culminado. El detalle incluye el estado, duración y recursos utilizados por cada uno de los trabajos.

```
[usuario@master ~]$ sacct -u usuario
       JobID    JobName  Partition    Account  AllocCPUS     State ExitCode
------------ ---------- --------- ---------- ---------- ---------- --------
443             jobnam  investigadores           4  COMPLETED     0:0
443.batch        batch                           4  COMPLETED     0:0
443.0           python                           4  COMPLETED     0:0
444             jobnam  investigadores           4  COMPLETED     0:0
```

```
444.batch         batch                          4  COMPLETED      0:0
444.0             python                         4  COMPLETED      0:0
445               jobnam  investigadores         4  COMPLETED      0:0
445.batch         batch                          4  COMPLETED      0:0
445.0             python                         4  COMPLETED      0:0
```

`sview`: Comando que se ha utilizado con el fin de desplegar una interfaz gráfica que permita la monitorización y la gestión de trabajos de Slurm, incluyendo opciones para visualizar detalles, cancelar o reanudar tareas.

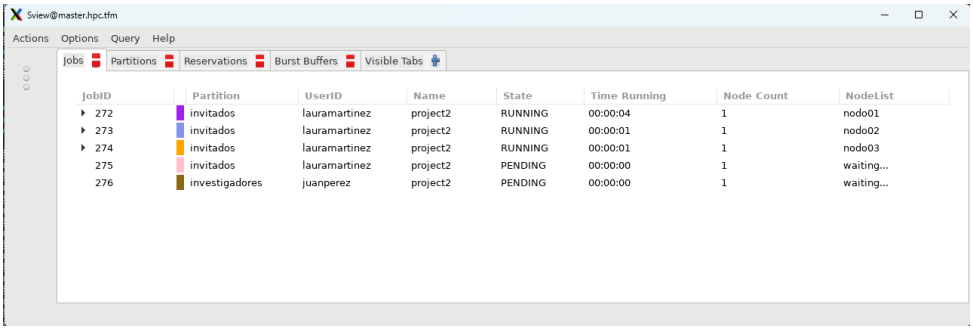

FIGURA C.2: Visualización del estado de los trabajos en ejecución en el clúster mediante la herramienta `sview`.

C.6.2. Ejemplo de script de trabajo

Un script de trabajo es un fichero que contiene las instrucciones necesarias para ejecutar un trabajo en el clúster. A continuación, se muestra un ejemplo de un script de trabajo en Slurm:

```
#!/bin/bash
#------- SLURM OPTIONS
#SBATCH -J jobname             # Nombre del trabajo a lanzar
#SBATCH -p invitados           # Nombre de la partición o cola a utilizar
#SBATCH -n 1                   # Número de tareas (tasks) a ejecutar
#SBATCH -c 4                   # Número de CPUs por tarea
#SBATCH --mem=1G               # Asignar espacio de memoria ram
#SBATCH -o jobname_%j.out      # Archivo de salida estándar
#SBATCH -e jobname_%j.err      # Archivo de salida de error estándar
#SBATCH --mail-user=user@example.com  # Correo para notificaciones
#SBATCH --mail-type=END,FAIL   # Tipo de notificaciones
#SBATCH --time=0-02:00:00      # Límite de tiempo para el trabajo

#------- LOAD MODULES
#------- LOAD ENVIRONMENT
#------- USER COMMANDS

echo "Job start: $(date)"

srun ./Hello_world

echo "Job end: $(date)"
```

Para ejecutar la tarea. `sbatch`: Enviar un trabajo por lotes.

```
[usuario@master ~]$ sbatch model.sh
```

C.7. Entorno de cómputo

Nota

La cuota asignada por usuario es de 30GB.

Tip

Es posible ver la salida de manera dinámica durante la ejecución con el comando `tail -f jobname_470.out` o `tail -f jobname_470.err`.

C.7.1. Python

Código de ejemplo

Ejemplo de código en Python: `example.py`

```
import pandas as pd
import numpy as np

rows, cols = 500000, 10
data = np.random.randn(rows, cols)
df = pd.DataFrame(data, columns=[f'Column_{i+1}' for i in range(cols)])

mean_values = df.mean()
median_values = df.median()
std_dev_values = df.std()
summary_statistics = df.describe()

# Mostrar resultados.
print("Mean:\n", mean_values)
print("\nMedian:\n", median_values)
print("\nStandard Deviation:\n", std_dev_values)
print("\nSummary Statistics:\n", summary_statistics)
```

Código Slurm

Cree un script de trabajo para configurar el slurm y ejecutar un programa. Por ejemplo, el siguiente script ejecuta un programa llamado `example.py`. Los principales parámetros que se deben ajustar son: `-J,-mail-user, -time`.

Ejemplo script slurm: `model_python.sh`

```
#!/bin/bash
#------- SLURM OPTIONS
#SBATCH -J jobname
```

```
#SBATCH -p invitados
#SBATCH -n 1
#SBATCH -c 4
#SBATCH --mem=0
#SBATCH -o jobname_%j.out
#SBATCH -e jobname_%j.err
#SBATCH --mail-user=user@example.com
#SBATCH --mail-type=END,FAIL
#SBATCH --time=0-06:00:00

#------- LOAD MODULES
#------- LOAD ENVIRONMENT
source /opt/python_envs/MLPytorch/bin/activate
#------- USER COMMANDS

echo "Job start: $(date)"

srun python example.py

echo "Job end: $(date)"
```

Ejecución

```
[usuario@master ~]$ sbatch model_python.sh
Submitted batch job 449
```

Tip

Es posible ver la salida de manera dinámica durante la ejecución, con el comando `tail -f jobname_449.out` o `tail -f jobname_449.err`.

C.7.2. R

R es un lenguaje de programación y software libre para análisis estadístico y visualización de datos, ampliamente utilizado por estadísticos y científicos de datos.

Código de ejemplo

Ejemplo codigo en R: `example.R`

```
# Establecer un directorio de biblioteca personalizado y crearlo si es
# necesario
user_lib <- "~/R/library"
dir.create(user_lib, recursive = TRUE)

# Instala y carga el paquete 'dplyr' en el directorio de librerías
# personalizadas
if (!require(dplyr, lib.loc = user_lib)) {
install.packages("dplyr", lib=user_lib, repos="https://cran.r-project.org")
library(dplyr, lib.loc = user_lib)
}

# Definir el tamaño de las matrices y realizar la multiplicación de
```

```
# matrices
n <- 10000
product <- matrix(runif(n * n), n, n) %*% matrix(runif(n * n), n, n)

# Mostrar la suma de todos los elementos del producto
print(sum(product))
```

Código Slurm

Cree un script de trabajo para configurar el slurm y ejecutar un programa. Por ejemplo, el siguiente script ejecuta un programa llamado `example.R`. Los principales parámetros que se deben ajustar son: `-J,-mail-user, -time`.

Ejemplo script slurm: `model_r.sh`

```
#!/bin/bash
#------- SLURM OPTIONS
#SBATCH -J jobname
#SBATCH -p debug
#SBATCH -n 1
#SBATCH -c 4
#SBATCH --mem=1G
#SBATCH -o jobname_%j.out
#SBATCH -e jobname_%j.err
#SBATCH --mail-user=user@example.com
#SBATCH --mail-type=END,FAIL
#SBATCH --time=0-06:00:00

#------- LOAD MODULES
#------- LOAD ENVIRONMENT
#------- USER COMMANDS

echo "Job start: $(date)"

srun Rscript example.R

echo "Job end: $(date)"
```

Ejecución

```
[usuario@master ~]$ sbatch model_r.sh
Submitted batch job 679
```

Tip

Es posible ver la salida de manera dinámica durante la ejecución, con el comando `tail -f jobname_679.out` o `tail -f jobname_679.err`.

C.8. Software de despliegue automatizado de algoritmos

Para utilizar el software desarrollado, es necesario descargar el repositorio disponible en https://github.com/seriab1/Slurm y ubicarlo dentro de la carpeta del programa desarrollado. La estructura resultante del proyecto debe quedar de manera similar con la que se muestra en la Figura 5.7.
A continuación es necesario ajustar los parámetros de configuración del archivo `./Slurm/_config.ini`

```
[SLURM]
user = juanperez               # Usuario SSH
host = 192.168.62.130          # Dirección del host SSH
port = 22                      # Puerto SSH
folder_dest = project2         # Carpeta de destino
task_name = project2           # Nombre de la tarea
exclude_folders = Slurm,.idea  # Carpetas que se excluirán del envío
source_folder = "../"          # Carpeta de origen
partition = investigadores     # Partición del clúster a utilizar
nodes = 1                      # Número de nodos
ntasks = 4                     # Número de tareas
mail = aquinobritezz@gmail.com # Correo para notificaciones
mail_type = END,FAIL           # Tipos de notificación (fin y fallos)
time = 0-06:00:00              # Tiempo máximo de ejecución (6 horas)
name_venv = project2           # Nombre del entorno virtual
environment = ../venv/project2/bin/activate # Ruta al entorno virtual
application = python           # Aplicación a ejecutar
script = example.py            # Script que se va a ejecutar
log_debug = 1                  # 1: Muestra información de depuración
module = ""                    # Módulo que se desee cargar en el entorno
```

Para culminar la configuración, es necesario establecer las credenciales `./Slurm/_secret.ini` .

C.8.1. Funcionalidades

A continuación, se describen los distintos scripts desarrollados para automatizar tareas comunes dentro del entorno HPC gestionado con Slurm.

C.8.1.1. Acceso al servidor

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python access_server.py
```

C.8.1.2. Agregar certificado

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python add_ssh_known_host.py
```

C.8.1.3. Cancelar todos los trabajos

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python cancel_all_job.py
```

C.8.1.4. Crear entorno virtual Python

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python create_environment.py
```

C.8.1.5. Información de Slurm

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python information.py
```

C.8.1.6. Información de salida log de la última ejecución

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python last_job_log.py
```

C.8.1.7. Envío al servidor

```
[usuario@local ~]$ cd Slurm
[usuario@local ~]$ python send_job.py
```


mgtpn
máster de
gestión y
tecnologías de procesos
de negocio


UNIVERSIDAD
DE GRANADA
MÁSTERES
ugr