



**Escuela Superior
de Ingeniería y Tecnología**
Universidad de La Laguna

Trabajo de Fin de Grado

Administración de nodos de cómputo de altas prestaciones

High-performance compute node management

Kevin Eliezer García Peña

La Laguna, 8 de septiembre de 2021

D. **Eduardo Manuel Segredo González**, con N.I.F. 78.564.242-Z, Profesor Ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Vicente Blanco Pérez**, con N.I.F. 42.171.808-C, Profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Administración de nodos de cómputo de altas prestaciones"

ha sido realizada bajo su dirección por D. **Kevin Eliezer García Peña**, con N.I.F. 54.113.567-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de septiembre de 2021

Agradecimientos

En primer lugar, agradecer a todos los profesores por la formación recibida durante estos últimos 5 años y hacer especial mención a Eduardo Manuel Segredo González y Vicente Blanco Pérez por la tutorización y guía de este proyecto final.

En segundo lugar, agradecer a todas las personas que he conocido durante mis estudios y que, de alguna manera u otra, han aportado en mi desarrollo profesional: organizadores, participantes y ponentes del CESINF y las Jornadas Técnicas de Ingeniería, compañeros de delegación y de clase.

Por último, agradecer a mi familia por el apoyo incondicional y confianza en mí. Agradecer también a todas las personas de mi entorno más cercano que, de alguna manera u otra, me han ayudado durante estos últimos meses, que han sido más complicados de lo habitual.

La Laguna, 8 de septiembre de 2021

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.

Resumen

El objetivo de este Trabajo Fin de Grado es implementar un cluster para computación de altas prestaciones (HPC, High Performance Computing por sus siglas en inglés) aplicando buenas prácticas para implementar seguridad y facilitar las labores posteriores de gestión y mantenimiento.

Para ello, el grupo PAL (Parallel Algorithms and Languages) de la ULL cuenta con tres máquinas: Bólido, Rayo y Centella. Utilizando herramientas de código abierto (Open Source), configuramos un cluster en el que Bólido es la máquina de entrada (Front-End), mientras que Rayo y Centella son nodos de cómputo. Finalmente, realizamos pruebas para medir el rendimiento y comprobar el correcto funcionamiento del cluster.

Palabras clave: HPC, cluster, linux, computación, paralelismo

Abstract

The main aim of this work is to implement a cluster for high performance computing, using the best practices for systems security and subsequent management and maintenance.

To be able to do it, the group Parallel Algorithms and Languages from University La Laguna has three machines: Bolido, Rayo and Centella. Thanks to open source tools and these machines, a high performance computing cluster was set up, in which Bolido works as the front-end machine and Rayo and Centella are just computing nodes. Finally, some tests were made to measure the cluster's performance and the right operation.

Keywords: HPC, cluster, linux, computing, parallelism

Índice general

1. Introducción	1
1.1. Computación de altas prestaciones	1
1.2. Programación paralela	2
2. Recursos disponibles	4
2.1. Estado del arte	4
2.2. Buenas prácticas	5
2.3. Análisis de herramientas	5
2.3.1. Slurm	5
2.3.2. Munge	6
2.3.3. OpenSSH	6
2.3.4. OpenLDAP	6
2.3.5. NTP	6
2.3.6. NFS	6
3. Configuración de cluster HPC	7
3.1. Preparación inicial	7
3.1.1. Actualización de los sistemas	7
3.1.2. DNS	8
3.1.3. SSH	8
3.1.4. NTP	9
3.1.5. Munge	10
3.1.6. Creación de usuarios locales	11
3.2. Configuración del controlador	12
3.2.1. NFS	12
3.2.2. LDAP	13
3.2.3. Slurm	15
3.2.4. Lmod	17
3.2.5. Quota	18
3.2.6. Script de gestión de usuarios	19
3.3. Configuración de los nodos	19
3.3.1. NFS	19
3.3.2. LDAP	20
3.3.3. Slurm	20
3.3.4. Bloqueo de autenticación de usuarios	21
3.4. Instalación de herramientas	21
3.4.1. OpenMPI	21
3.4.2. OpenBLAS	22

3.4.3. GSL	22
3.4.4. Intel Performance Tools	23
3.5. Problemas	24
3.5.1. Configuración de Bolido en Raid 1	24
3.5.2. Instalación de slurm con compatibilidad con OpenMPI	24
3.5.3. Errores al configurar el cluster	24
3.5.4. Error espacio al actualizar de Debian 9 a Debian 10 (Centella)	25
4. Análisis de resultados	26
4.1. Análisis del rendimiento	26
4.1.1. Métricas	26
4.2. Cálculo de PI	28
4.3. NAS Parallel Benchmarks	28
4.3.1. Generación de pruebas y filtración	29
4.4. Resultados	31
4.4.1. Gráficas IS	33
4.4.2. Gráficas EP	34
4.4.3. Gráficas CG	35
4.4.4. Gráficas MG	36
4.4.5. Gráficas FT	37
4.4.6. Gráficas BT	38
4.4.7. Gráficas SP	39
4.4.8. Gráficas LU	40
5. Conclusiones y líneas futuras	41
5.1. Conclusiones	41
5.2. Líneas futuras	41
6. Summary and Conclusions	43
7. Presupuesto	44
7.1. Presupuesto	44
A. Scripts	45
A.1. Script administración de usuarios	45
A.2. Script generador CSV	47
A.3. Generador de graficas	47

Índice de Figuras

3.1. Comprobación de que el sistema se actualizó correctamente	8
3.2. Comprobación de la sincronización del reloj	10
3.3. Comprobación codificación y decodificación de claves de Munge en el controlador	11
3.4. Comprobación codificación de clave de Munge en el nodo y decodificación en el controlador	12
3.5. Componentes de slurm	16
4.1. SpeedUp	27
4.2. Jerarquía de memoria	31
4.3. Speedup IS	33
4.4. Eficiencia IS	33
4.5. Utilización IS	33
4.6. Redundancia IS	33
4.7. Calidad Paralelismo IS	33
4.8. Tiempo Ejecución IS	33
4.9. Speedup EP	34
4.10Eficiencia EP	34
4.11Utilización EP	34
4.12Redundancia EP	34
4.13Calidad Paralelismo EP	34
4.14Tiempo Ejecución EP	34
4.15Speedup CG	35
4.16Eficiencia CG	35
4.17Utilización CG	35
4.18Redundancia CG	35
4.19Calidad Paralelismo CG	35
4.20Tiempo Ejecución CG	35
4.21Speedup MG	36
4.22Eficiencia MG	36
4.23Utilización MG	36
4.24Redundancia MG	36
4.25Calidad Paralelismo MG	36
4.26Tiempo Ejecución MG	36
4.27Speedup FT	37
4.28Eficiencia FT	37
4.29Utilización FT	37
4.30Redundancia FT	37
4.31Calidad Paralelismo FT	37

4.32	Tiempo Ejecución FT	37
4.33	Speedup BT	38
4.34	Eficiencia BT	38
4.35	Utilización BT	38
4.36	Redundancia BT	38
4.37	Calidad Paralelismo BT	38
4.38	Tiempo Ejecución BT	38
4.39	Speedup SP	39
4.40	Eficiencia SP	39
4.41	Utilización SP	39
4.42	Redundancia SP	39
4.43	Calidad Paralelismo SP	39
4.44	Tiempo Ejecución SP	39
4.45	Speedup LU	40
4.46	Eficiencia LU	40
4.47	Utilización LU	40
4.48	Redundancia LU	40
4.49	Calidad Paralelismo LU	40
4.50	Tiempo Ejecución LU	40

Índice de Listados

3.1. Fichero /etc/hosts	8
3.2. Fichero /etc/ntp.conf de Bolido	9
3.3. Fichero /etc/ntp.conf de Bolido	10
3.4. Fichero /etc/exports de Bolido	12
3.5. Fichero /etc/nsswitch.conf	13
3.6. Fichero /etc/ldap/ldap.conf	14
3.7. fichero userandgroups.ldif	14
3.8. fichero /etc/cpu/cpu.conf	14
3.9. Fichero /etc/slurm-llnl/slurm.conf	17
3.10 Configuración /etc/fstab en Bolido para Quotas	18
3.11 Fichero /etc/nsswitch.conf	20
3.12 Modulefile de OpenMPI	22
3.13 Modulefile de OpenBLAS	22
3.14 Modulefile de GSL	23
4.1. Script scripts/generate.sh	29
4.2. Script run.sh	30
4.3. Script generador_csv.sh	30

Índice de Tablas

7.1. Presupuesto del material 44
7.2. Presupuesto del personal 44

Capítulo 1

Introducción

1.1. Computación de altas prestaciones

La computación de altas prestaciones (HPC, High Performance Computing por sus siglas en inglés) es una forma de procesar grandes volúmenes de datos a velocidades muy altas utilizando varios ordenadores y dispositivos de almacenamiento [1]. Este campo nace de la necesidad de resolver problemas computacionales muy complejos, generalmente de la ciencia y de la ingeniería.

Numerosas son las aplicaciones que tiene [2] y los beneficios que aporta a los ciudadanos, a la industria y a la ciencia [3].

En los beneficios para los ciudadanos, la supercomputación es clave para el avance de la medicina: permite descubrir nuevos medicamentos, desarrollar terapias médicas que cubran necesidades individuales y estudiar enfermedades genéticas poco comunes. Ha permitido en la actualidad desarrollar los tratamientos para el COVID-19. Adicionalmente, permite monitorizar los efectos del cambio climático así como mejorar nuestro conocimiento de los procesos geofísicos y realizar simulaciones que predigan la evolución del tiempo a partir de sus patrones.

Para la industria (automóvil, aeroespacial, energías renovables), los supercomputadores permiten innovar, ser más productivos y escalar con productos de mayor valor y servicio, reduciendo los ciclos de diseño y producción y acelerando el diseño de nuevos materiales, gracias a que minimiza los costes, incrementa la eficiencia de recursos y acorta y optimiza los procesos de decisión.

Por último, en la ciencia, permite avanzar los conocimientos que tenemos sobre la materia y explorar el universo, así como modelar la atmósfera y el fenómeno oceánico a nivel planetario.

Un ejemplo de sistema de computación de altas prestaciones existentes podría ser el Teide-HPC [4], perteneciente al ITER (Instituto Tecnológico y de Energías Renovables) y que se encuentra en Tenerife. Está compuesto por 1100 servidores de cómputo Fujitsu, con un total de 17800 cores de cómputo y 36 TB de memoria, siendo así el segundo más potente de España y, hasta el 2015, uno de los top 500 supercomputadores más potentes del mundo [5].

En España se ubica también el supercomputador Marenostrum [6], que es considerado el sexagésimo tercer computador más potente del mundo en la última revisión de junio

de 2021 [7]. La última versión del Marenostrom tiene un rendimiento pico de 13.9 flops y se compone de dos bloques: uno de propósito general, que tiene 165.888 procesadores en total y una memoria central de 390 terabytes, y otro de tecnologías emergentes, que está formado por tres clusters de tecnologías que se están desarrollando actualmente en Estados Unidos y Japón para la nueva generación de supercomputadores pre-exascale.

A nivel Europeo, el sector de la computación de altas prestaciones no para de crecer. Actualmente, existe la iniciativa Eurohpc [8] que, con un presupuesto de mil millones de euros, se están construyendo 7 supercomputadores en toda Europa, se espera que dos de ellos estén dentro del top 5 mundial: LUMI, localizado en Finlandia, con un rendimiento máximo de 552 petaflops y Leonardo, en Italia, con un rendimiento máximo de 322.6 petaflops.

En la computación de altas prestaciones no importa únicamente la cantidad de núcleos y la potencia que tiene el hardware, sino que también debe tenerse en cuenta cómo se desarrolla el software que va a ejecutarse sobre la máquina, para maximizar el rendimiento. Para ello, se emplea la programación paralela.

1.2. Programación paralela

La programación paralela es una técnica en la que muchas instrucciones se ejecutan de forma simultánea. Se basa en el principio de "Divide y vencerás": los problemas grandes se pueden dividir en partes más pequeñas que pueden resolverse de forma concurrente. Existen varios tipos de computación paralela [9]:

- **Paralelismo a nivel de bit:** cuando se aumenta el tamaño de la palabra del procesador, se habla de paralelismo a nivel de bit.

Por ejemplo, al operar con dos números de 16 bits sobre un procesador de 8 bits, se requeriría de dos instrucciones, mientras que si el procesador fuera de 16 bits, esta se podría realizar en una sola instrucción.

- **Paralelismo a nivel de instrucción:** consiste en cambiar el orden de las instrucciones de un programa y juntarlas en grupos, siempre sin alterar el resultado final del programa.

Esto se puede hacer gracias a que los procesadores modernos tienen N etapas por los que pasan las instrucciones, por lo que ese procesador puede ejecutar hasta N instrucciones diferentes simultáneamente con la ordenación y agrupación adecuada.

- **Paralelismo de datos:** cada procesador realiza la misma tarea sobre un subconjunto independiente de datos.

Es muy común utilizarlo en big data, especialmente cuando se opera con una gran cantidad de datos que no es posible cargar en memoria.

- **Paralelismo de tareas:** cada hilo realiza una tarea distinta e independiente al resto.

Un ejemplo podría ser una aplicación de mensajería instantánea, donde un hilo se encuentra pendiente para recibir información y otro está pendiente para la entrada de texto del cliente.

En los últimos años, el interés por la computación paralela aplicada en la computación de altas prestaciones ha aumentado debido a las restricciones físicas que impiden el escalado en frecuencia del hardware. Los ordenadores paralelos se pueden clasificar según el nivel de paralelismo que admite su hardware [10]:

- **Procesamiento multinúcleo.** La máquina tiene más de una unidad de CPU, por lo que puede procesar múltiples instrucciones por ciclo de secuencias. Si el núcleo es superescalar, adicionalmente, en cada ciclo puede ejecutar múltiples instrucciones de un flujo de instrucciones.
- **Multiprocesamiento simétrico.** Se trata de un sistema computacional con múltiples procesadores iguales que comparten memoria y están conectados a través de un bus.
- **Computación en cluster.** Se trata de un grupo de ordenadores independientes que trabajan en colaboración por red. El más típico es el cluster Beowulf [11], que implementa múltiples ordenadores idénticos conectados a una red local por Ethernet.
- **Computación distribuida.** Se hace uso de ordenadores que se comunican a través de Internet para trabajar en un problema dado. Normalmente, se refiere a problemas vergonzosamente paralelos, ya que la latencia es demasiado alta y el ancho de banda es bajo.

Los dos paradigmas principales de la programación paralela son el paso de mensajes y la memoria compartida [12].

El paradigma de paso de mensajes es ampliamente utilizado en el software moderno para ejecutar tareas sincronizadas entre varios ordenadores.

Para ello, se intercambia información de forma síncrona o asíncrona. La manera síncrona es sencilla de implementar, pero su inconveniente es que es poco flexible, ya que bloquea el proceso hasta que se realiza el intercambio de mensajes entre procesos. El intercambio de mensajes de forma asíncrona no produce bloqueos en el código, sin embargo, puede generar fallos de seguridad cuando se intercambian y manipulan variables.

En el paradigma de memoria compartida, el intercambio de datos entre programas se realiza reservando un área de RAM en la que otro proceso puede acceder. Así, cuando se modifiquen valores dentro de este área, será visible para los demás procesos.

El desarrollo de programas informáticos paralelos es más complejo que los secuenciales.

La concurrencia introduce nuevos tipos de errores de software, como las condiciones de carrera y añade nuevos problemas como la sincronización entre diferentes subtareas.

Además, no todas las paralelizaciones conllevan una aceleración. Generalmente, mientras una tarea se divide en cada vez más hilos, estos hilos pasan una porción cada vez mayor de su tiempo comunicándose entre sí. Eventualmente, la sobrecarga de comunicación domina el tiempo empleado para resolver el problema, y la paralelización adicional aumenta la cantidad de tiempo requerido para terminar.

Capítulo 2

Recursos disponibles

En este capítulo se describe el estado del arte tanto antes de empezar el proyecto como después, así como los recursos disponibles, el análisis de posibles herramientas y las herramientas utilizadas para llevar a cabo la configuración del *cluster* para computación de altas prestaciones.

2.1. Estado del arte

Antes de comenzar el proyecto, el grupo de investigación PAL (Parallel Algorithms and Languages) contaba ya con tres máquinas con las que realizar computación de altas prestaciones: Bolido, Rayo y Centella.

Cada una de estas máquinas tiene un procesador AMD con 48 núcleos y 64 gigas de RAM. Bolido tiene dos discos duros de 500GB: uno de tipo SAS y otro SATA. Rayo también cuenta con dos discos duros, aunque ambos son de tipo SAS y Centella cuenta con un único disco duro de 500GB de tipo SAS.

Estas máquinas se utilizaban de manera individual y no tenían ningún mantenimiento ni ninguna gestión. Los usuarios accedían a cualquiera de ellas cuando necesitaban realizar alguna prueba, compilaban su código y lo dejaban ejecutándose con un número de procesadores determinados. Con el paso de los años, el sistema operativo de estas máquinas quedó también obsoleto.

Con el afán de poner en funcionamiento de nuevo estas máquinas y organizar mejor los recursos disponibles, se decidió actualizar y configurar de nuevo los sistemas y crear un cluster, en el que se centralicen los usuarios y la gestión de las máquinas disponibles.

Para la actualización de los sistemas, se tuvo que ir físicamente al STIC (Servicio TIC de la Universidad de La Laguna) porque la interfaz remota iDRAC [13] del servidor era demasiado lenta. En concreto, Bolido y Centella se tuvieron que instalar desde cero, para configurar el sistema con LVM (gestor de volúmenes lógicos) y Debian 10.

Por último, se realizó la configuración del cluster con distintas herramientas open source y se realizó un análisis de rendimiento.

2.2. Buenas prácticas

Es conveniente que al implementar un *cluster* se utilicen una serie de buenas prácticas, con el fin de facilitar tanto el mantenimiento como la administración del sistema en el futuro y mantener seguros los sistemas. Para este caso, me he apoyado en un paper sobre las mejores prácticas antes de implementar el sistema [14] y de elegir las herramientas.

Para nuestro caso, lo que más nos interesa en nuestro sistema es minimizar el trabajo de administración y gestión de sistemas. Esto lo logramos con:

- **Centralización:** conviene que todo lo que sea común en todas las máquinas se encuentre centralizado: usuarios, librerías, software, autenticación, manejo de recursos.
- **Escalabilidad:** hacer que las modificaciones del *cluster* sean lo más sencillas posibles, tanto para añadir como para eliminar nodos.
- **Administración de sistemas mínimo:** trazabilidad sencilla y sistemas robustos. Administrar muchas máquinas puede ser complejo, por lo que es conveniente que los sistemas sean lo más minimalistas posibles, especialmente los nodos.
- **Administración flexible del entorno de usuario:** permitir al usuario modificar el entorno de software fácilmente y ajustarlo a sus necesidades.

2.3. Análisis de herramientas

Existen diversas herramientas para configurar un *cluster* para realizar programación paralela. En este apartado, se definen las herramientas utilizadas para la implementación de nuestro sistema y las alternativas que se han encontrado y descartado.

2.3.1. Slurm

Para el sistema HPC, necesitábamos un software que permitiera aceptar trabajos, planificarlos y monitorizarlos. Básicamente, un software de planificación de tareas. Para ello, se decidió utilizar Slurm [15].

Slurm es un planificador de tareas de código abierto [16] para sistemas basados en Linux que es tolerable a fallos y que permite escalar la gestión de sistemas distribuidos.

Con Slurm, se puede asignar usuarios a nodos de cómputo, con acceso no exclusivo, exclusivo, con recursos compartidos o con recursos limitados. Permite también iniciar, realizar y monitorizar los trabajos sobre los nodos y gestionar las colas de trabajos pendientes.

Adicionalmente, se puede extender con diferentes plugins, para realizar autenticación, registro de tareas completadas, gestión de energía, etc.

Como alternativa, se consideró TORQUE y Kubernetes.

TORQUE (Terascale Open-source Resource and Queue manager) [17] es un planificador de trabajos similar a Slurm que no llegó a mantenerse en el tiempo, por lo que no era conveniente utilizarlo.

Kubernetes [18] es un orquestador de código abierto para cargas de trabajo basadas en contenedores. Permite manejar cargas de trabajo similar a un cluster HPC, aunque no ofrece todas las capacidades de gestión de trabajos que slurm.

Kubernetes se basa en cluster de nodos que se controlan por un master. Cada nodo tiene una serie de contenedores que comparten recursos y existen en la red local, por lo que pueden comunicarse entre sí. Se descartó debido a que su uso es idóneo para escalar tecnologías *cloud* y contenedores basados en microservicios, mientras que slurm permite lanzar aplicaciones en escala.

2.3.2. Munge

Munge [19] es un servicio de autenticación para crear y validar credenciales, diseñado para ser escalable en sistemas distribuidos.

Es un software completamente necesario para el funcionamiento correcto de slurm, ya que lo utiliza para autenticar los trabajos.

2.3.3. OpenSSH

Para la conexión remota cifrada entre máquinas, se utilizó OpenSSH [20], que es la implementación libre utilizada generalmente en sistemas linux.

2.3.4. OpenLDAP

Para el acceso unificado a la información de red y autenticación centralizada, se utilizó OpenLDAP [21].

En concreto, se comparten en todos los sistemas los ficheros `/etc/passwd` y `/etc/group` globales para la autenticación de los usuarios.

2.3.5. NTP

Se utiliza la implementación de NTP para sincronizar los relojes del cluster, completamente necesario para evitar problemas

Como alternativa, existe chrony [22].

Para nuestro caso de uso, es preferible utilizar el demonio NTP porque nuestros sistemas siempre están conectados por red. Chrony se considera mejor para sistemas que se desconectan de una red o se suspenden de forma frecuente, ya que tiene una sincronización más rápida y mejor respuesta a los cambios en la frecuencia del reloj [23].

2.3.6. NFS

Por último, se utilizó NFS [24] (Network File System) para centralizar el software disponible y las carpetas de los usuarios. En cambio, autofs [25] se utilizó para montar dichos directorios en los clientes (nodos de cómputo).

Capítulo 3

Configuración de cluster HPC

3.1. Preparación inicial

Antes de configurar el cluster HPC, debe realizarse primero una configuración inicial en todas las máquinas.

3.1.1. Actualización de los sistemas

Al inicio, Bolido y Centella se encontraban con un sistema Debian 8, por lo que se realizó la migración a Debian 10. La actualización se realizó manualmente, cambiando en el fichero `/etc/apt/sources.list` los repositorios a los correspondientes de la versión siguiente y forzando después una actualización.

Para actualizar de Debian 8 a Debian 9 [26], el fichero debe apuntar a los siguientes repositorios:

```
deb http://htpredir.debian.org/debian stretch main contrib non-free
deb http://htpredir.debian.org/debian stretch-updates main contrib non-free
deb http://security.debian.org stretch/updates main contrib non-free
```

Similar es la migración de Debian 9 a Debian 10 [27], donde el fichero debe apuntar a los siguientes repositorios:

```
deb http://deb.debian.org/debian debian buster main
deb http://deb.debian.org/debian buster-updates main
deb http://deb.debian.org/debian buster/updates main
```

Finalmente, se fuerza la actualización con:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo reboot
```

Por último, con los siguientes comandos se comprueba si el sistema se actualizó correctamente (figura 3.1) y se elimina las dependencias obsoletas que ya no son necesarias en el sistema.

```
$ hostnamectl
$ sudo apt --purge autoremove
```

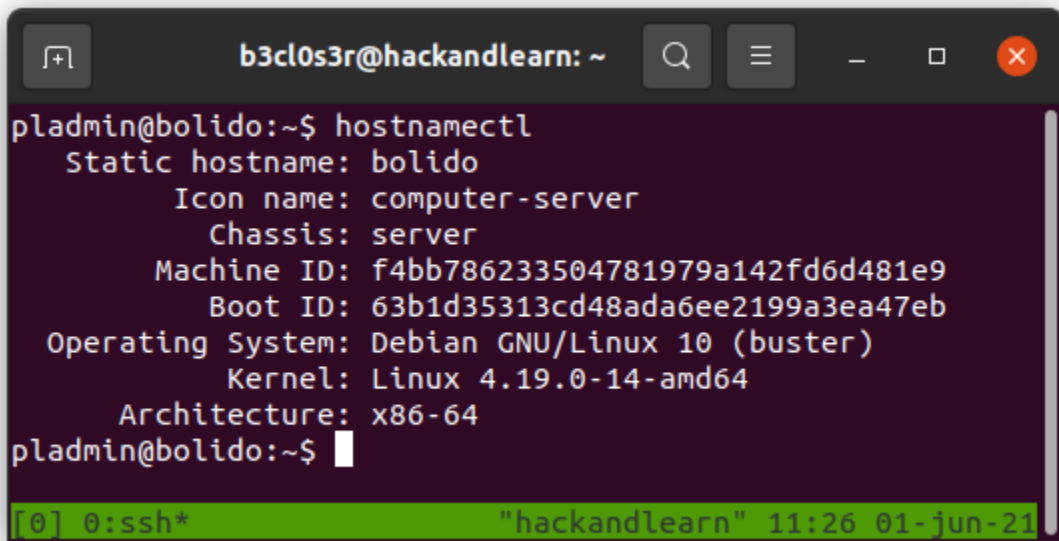
A terminal window titled 'b3cl0s3r@hackandlearn: ~' showing the output of the 'hostnamectl' command. The output lists system information: Static hostname: bolido, Icon name: computer-server, Chassis: server, Machine ID: f4bb786233504781979a142fd6d481e9, Boot ID: 63b1d35313cd48ada6ee2199a3ea47eb, Operating System: Debian GNU/Linux 10 (buster), Kernel: Linux 4.19.0-14-amd64, Architecture: x86-64. The prompt is 'pladmin@bolido:~\$'. At the bottom, a green status bar shows '[0] 0:ssh*' and '"hackandlearn" 11:26 01-jun-21'.

Figura 3.1: Comprobación de que el sistema se actualizó correctamente

3.1.2. DNS

Antes de configurar el *cluster*, es conveniente que configuremos el DNS entre las tres máquinas. Para ello, solo hay que modificar el fichero `/etc/hosts` de cada máquina y añadir en cada línea dos pares separados por un espacio: dirección ip y los nombres que le asociaremos.

La configuración del fichero para las tres máquinas es como el siguiente:

Listado 3.1: Fichero `/etc/hosts`

```
127.0.0.1      localhost
192.168.1.2   bolido bolido.pcg.ull.es
192.168.1.3   rayo rayo.pcg.ull.es
192.168.1.4   centella centella.pcg.ull.es
```

3.1.3. SSH

SSH (Secure Shell) es un protocolo ampliamente utilizado para la gestión remota de máquinas. En nuestro caso, es necesario para poder acceder a nuestros ordenadores.

Para la instalación del cliente y servidor, se ejecutan los siguientes comandos respectivamente:

```
$ apt-get install openssh-client openssh-server
```

Otro requisito para la creación del *cluster*, es poder conectarse entre nodos realizando la autenticación sin contraseña. Para ello, como las carpetas de usuario se encuentran compartidas por NFS, se puede utilizar la autenticación mediante claves.

Para ello, se genera el par de claves automáticamente y se añade autoriza la clave pública:

```
$ ssh-keygen -t rsa
$ cp ~/.ssh/id_rsa.pub authorized_keys
```

3.1.4. NTP

Uno de los requisitos para poder configurar Slurm correctamente es que todos los ordenadores del *cluster* tengan el reloj sincronizado. Para ello, hay que configurar NTP (Network Protocol Time), el protocolo que permite sincronizar los relojes de los ordenadores.

Se instala NTP en cada una de las máquinas con el siguiente comando:

```
$ apt-get install ntp
```

Tanto para configurar el cliente como el servidor, hay que modificar el fichero `/etc/ntp.conf`.

En Bolido, se ha sincronizado su reloj con el pool de servidores oficial de Debian. Adicionalmente, se ha bloqueado acceso al servicio a todas las máquinas excepto a las pertenecientes al *cluster* y un servidor de backup que mantenga la hora del sistema en caso de que la sincronización con los servidores externos falle.

Listado 3.2: Fichero `/etc/ntp.conf` de Bolido

```
# Backup server if official NTP servers fail
server 127.127.1.0
fudge 127.127.1.0 stratum 10

# Time servers
pool 0.debian.pool.ntp.org iburst
pool 1.debian.pool.ntp.org iburst
pool 2.debian.pool.ntp.org iburst
pool 3.debian.pool.ntp.org iburst

# Ignore all traffic
restrict default ignore
restrict -6 default ignore

# Allow localhost to manage ntpd
restrict 127.0.0.1
restrict ::1
```

```
# Allow servers to reply to our queries
restrict source nomodify noquery notrap
```

```
# Allow nodes to access ntp
restrict rayo nomodify
restrict centella nomodify
```

En Rayo y Centella, se ha añadido a Bolido como servidor de referencia para sincronizar el reloj y se han comentado todos aquellos que hacían referencia a los oficiales de Debian.

Listado 3.3: Fichero /etc/ntp.conf de Bolido

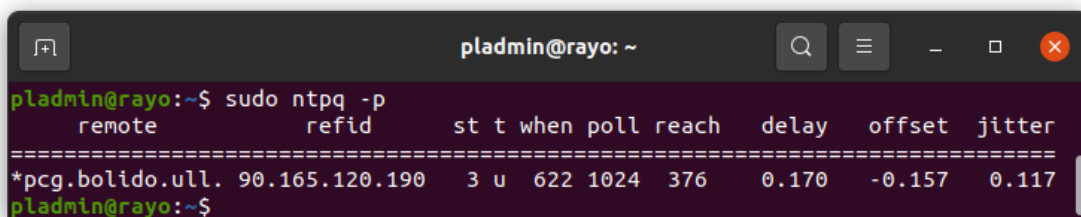
```
# ...

# Sincronizacion de reloj con Bolido
server bolido iburst

# ...

# Comentamos los servidores de debian
# para que no se sincronice con ellos
# pool 0.debian.pool.ntp.org iburst
# pool 1.debian.pool.ntp.org iburst
# pool 2.debian.pool.ntp.org iburst
# pool 3.debian.pool.ntp.org iburst
```

Con el comando `ntpq` se puede comprobar que los nodos tienen la hora sincronizada con bolido (figura 3.2).



```
pladmin@rayo: ~
pladmin@rayo:~$ sudo ntpq -p
      remote           refid      st t when poll reach  delay  offset jitter
-----
*pcg.bolido.ull. 90.165.120.190  3 u 622 1024 376   0.170  -0.157  0.117
pladmin@rayo:~$
```

Figura 3.2: Comprobación de la sincronización del reloj

3.1.5. Munge

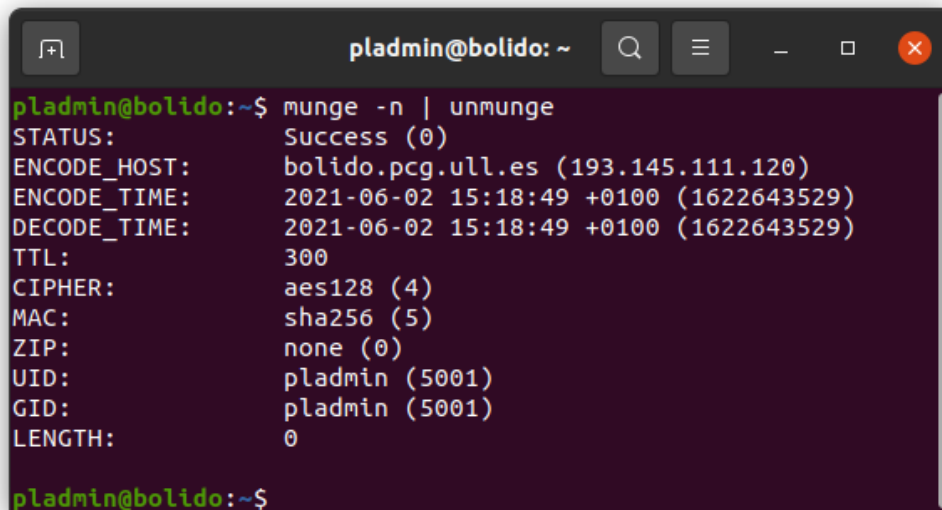
Munge es un servicio de autenticación y validación de credenciales. Se utilizará con Slurm para validar sus procesos.

Para instalar y habilitar el servicio, se ejecuta:

```
$ apt-get install libmunge-dev libmunge2 munge
```

```
$ systemctl enable munge
$ systemctl start munge
```

En la figura 3.3 se comprueba que munge funciona correctamente, codificando y decodificando claves desde el controlador (Bolido).

A terminal window titled 'pladmin@bolido: ~' showing the execution of the 'munge -n | unmunge' command. The output displays various parameters: STATUS: Success (0), ENCODE_HOST: bolido.pcg.ull.es (193.145.111.120), ENCODE_TIME: 2021-06-02 15:18:49 +0100 (1622643529), DECODE_TIME: 2021-06-02 15:18:49 +0100 (1622643529), TTL: 300, CIPHER: aes128 (4), MAC: sha256 (5), ZIP: none (0), UID: pladmin (5001), GID: pladmin (5001), and LENGTH: 0.

```
pladmin@bolido:~$ munge -n | unmunge
STATUS:          Success (0)
ENCODE_HOST:     bolido.pcg.ull.es (193.145.111.120)
ENCODE_TIME:     2021-06-02 15:18:49 +0100 (1622643529)
DECODE_TIME:     2021-06-02 15:18:49 +0100 (1622643529)
TTL:             300
CIPHER:          aes128 (4)
MAC:             sha256 (5)
ZIP:             none (0)
UID:             pladmin (5001)
GID:             pladmin (5001)
LENGTH:         0
pladmin@bolido:~$
```

Figura 3.3: Comprobación codificación y decodificación de claves de Munge en el controlador

A continuación, en los nodos (Rayo y Centella) se copia la clave de la máquina principal (Bolido) y se guarda en /etc/munge con el usuario munge de propietario y con permiso de lectura únicamente para este usuario.

```
$ scp bolido :/etc/munge/munge.key /etc/munge/
$ chown munge:munge /etc/munge/munge.key
$ chmod 400 /etc/munge/munge.key
```

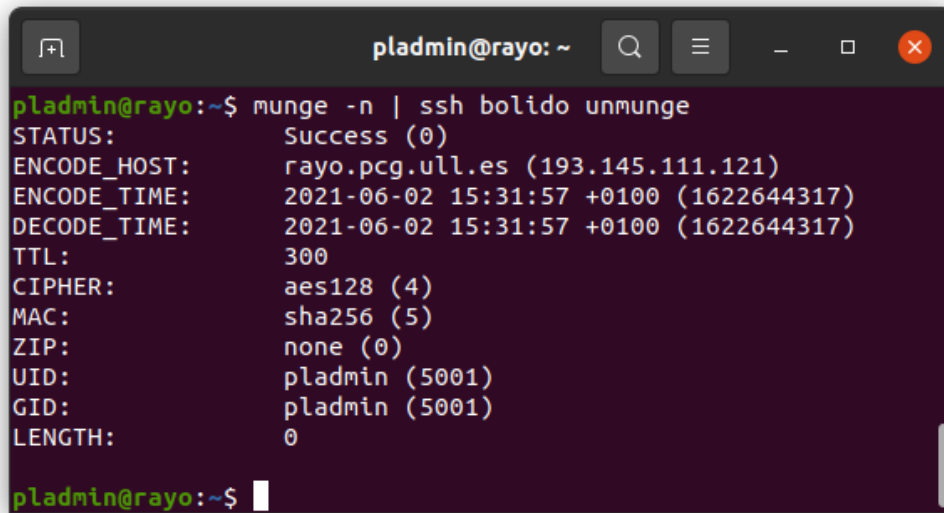
En la figura 3.4, se comprueba que munge funciona correctamente entre un nodo y el controlador: codificando desde un nodo y decodificando el hash generado desde el controlador.

3.1.6. Creación de usuarios locales

En todas las máquinas, se creó un usuario local común como administrador y un usuario llamado slurm con el mismo UID y GUID para el servicio de Slurm.

Se empleó los siguientes comandos:

```
groupadd -g 5000 slurm
useradd slurm -u 5000 -g 5000
```



```
pladmin@rayo:~$ munge -n | ssh bolido unmunge
STATUS:          Success (0)
ENCODE_HOST:    rayo.pcg.ull.es (193.145.111.121)
ENCODE_TIME:    2021-06-02 15:31:57 +0100 (1622644317)
DECODE_TIME:    2021-06-02 15:31:57 +0100 (1622644317)
TTL:            300
CIPHER:         aes128 (4)
MAC:            sha256 (5)
ZIP:            none (0)
UID:            pladmin (5001)
GID:            pladmin (5001)
LENGTH:        0
pladmin@rayo:~$
```

Figura 3.4: Comprobación codificación de clave de Munge en el nodo y decodificación en el controlador

3.2. Configuración del controlador

El controlador (Bolido) es la máquina de entrada del *cluster*. Se encarga de administrar, gestionar los recursos, ejecutar tareas sobre los nodos y proveer de autenticación centralizada y de software y herramientas a los nodos.

3.2.1. NFS

NFS [24] (Network File System) es un protocolo de nivel de aplicación utilizado para compartir sistemas de archivos entre ordenadores. En nuestro *cluster*, permite centralizar el software que proveeremos y las carpetas personales de los usuarios en todo el sistema.

El paquete se instala con:

```
$ apt-get install nfs-kernel-server
```

A continuación, se modifica el fichero `/etc/exports` para compartir los directorios de `/home` y `/opt` con permisos de escritura y mapeo de root independientemente del sistema (`no_root_squash`).

Listado 3.4: Fichero `/etc/exports` de Bolido

```
/home    rayo(rw, sync, no_root_squash, no_subtree_check)
         centella(rw, sync, no_root_squash, no_subtree_check)

/opt     rayo(rw, sync, no_root_squash, no_subtree_check)
         centella(rw, sync, no_root_squash, no_subtree_check)
```


3.2.2. LDAP

LDAP (Lightweight Directory Access Protocol) [28] es un protocolo de nivel de aplicación que permite el acceso a servicio de directorio de forma ordenada y distribuida. En concreto, se utilizará para compartir a los nodos los usuarios y grupos que existen en nuestro sistema.

Se instala LDAP como servidor con la siguiente instrucción:

```
$ apt-get install slapd
```

A continuación, se reconfigura LDAP:

```
$ sudo dpkg-reconfigure slapd
```

En él, se establece como nombre distinguido (DN) base `dc=bolido,dc=pcg,dc=ull,dc=es` y se crea una nueva base de datos con su usuario de administrador.

Para que el sistema local autentique los usuarios contra LDAP, se debe instalar los siguientes paquetes:

```
$ sudo apt-get install libnss-ldapd nscd nslcd
```

Durante la instalación, aparecerán una serie de pantallas para configurar LDAP. Se debe establecer en ella la dirección IP del LDAP del servidor (Bolido), el nombre distinguido para la base de las búsquedas (`dc=bolido, dc=pcg, dc=ull, dc=es`), la versión de LDAP (3) y el usuario administrador de LDAP. Esto configurará `nscd` y `nslcd` con sus respectivos ficheros de configuración: `libnss-ldap.conf` y `nslcd.conf`.

Es importante que se instale `libnss-ldapd` en lugar de `libnss-ldap`. Esta versión es una mejora que ofrece mejor depuración, mejor rendimiento y separación del código de NSS, LDAP y PAM [29].

Por último, hay que indicar en el fichero `/etc/nsswitch.conf` que emplee LDAP como servicio de autenticación.

Listado 3.5: Fichero `/etc/nsswitch.conf`

```
passwd:      files ldap systemd
group:       files ldap systemd
shadow:      files ldap
gshadow:     files ldap
```

Es importante que LDAP se encuentre antes que `systemd` para que al realizar la autenticación, el sistema intente rescatar primero la información de las bases de datos, después de LDAP y, en caso de no lograrlo, pregunte a los demonios del sistema.

Para gestionar LDAP, se puede utilizar `ldap-utils`, `ldapvi` y `cpu`. `Ldap-utils` provee de herramientas para manipular las bases de datos, `LDAPvi` permite modificar LDAP como si se utilizara un editor de texto y `CPU` facilita las tareas de añadir nuevos usuarios y grupos al sistema.

```
$ apt-get install ldap-utils cpu ldapvi
```

En primer lugar, es conveniente configurar el fichero `ldap.conf`, el fichero de configuración para las herramientas de cliente de LDAP. Este debe tener permisos de lectura y escritura solo para `root`, para evitar fallos de seguridad.

Listado 3.6: Fichero `/etc/ldap/ldap.conf`

```
uri ldap://0.0.0.0
base dc=bolido ,dc=pcg ,dc=ull ,dc=es
ldap_version 3
ssl start_tls
binddn cn=admin ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
bindpw asafepasswordgoeshere
TLS_CACERT /etc/ssl/certs/ca-certificates.crt
```

A continuación, con el siguiente comando y fichero en formato `ldif`, se crean las unidades organizativas en las que se encontrarán los usuarios y grupos del *cluster*.

```
$ ldapadd -x -D "cn=admin ,dc=bolido ,dc=pcg ,dc=ull ,dc=es"
-W -H ldap:// -f userandgroups.ldif
```

Listado 3.7: fichero `userandgroups.ldif`

```
dn: ou=People ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
objectClass: organizationalUnit
description: bolido users and groups
ou: People

dn: ou=Users ,ou=People ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
objectClass: organizationalUnit
description: bolido users
ou: Users

dn: ou=Groups ,ou=People ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
objectClass: organizationalUnit
description: bolido groups
ou: Groups
```

Ahora, se configura la herramienta `CPU` [30] para que cree de una forma más sencilla los usuarios en LDAP. Para ello, primero se debe configurar el fichero `/etc/cpu/cpu.conf` y establecer la opción de `SHADOWLASTCHANGE` a 0 para forzar que cuando el usuario se conecte deba que cambiar la contraseña.

Listado 3.8: fichero `/etc/cpu/cpu.conf`

```
# ...
LDAP_URI      = ldap://localhost
BIND_DN       = cn=admin ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
BIND_PASS     = password
USER_BASE     = ou=Users ,ou=People ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
GROUP_BASE    = ou=Groups ,ou=People ,dc=bolido ,dc=pcg ,dc=ull ,dc=es
```

```
# ...  
SHADOWLASTCHANGE = 0
```

Ahora, es posible crear y eliminar usuarios dentro de nuestro *cluster* de una forma muy sencilla:

```
$ cpu useradd $USERNAME -m -p$PASSWORD  
$ cpu userdel -r $USERNAME  
$ cpu groupdel $USERNAME
```

Con la opción `-m`, CPU crea el directorio home del usuario. Para eliminar usuarios, se utiliza `-r` para eliminar también su directorio *home*. Por defecto, CPU crea el grupo del usuario pero no lo elimina, por lo que hay que hacerlo manualmente.

Con LDAPvi [31], se puede modificar LDAP con un editor de texto utilizando el siguiente comando:

```
$ ldapvi -D "cn=admin,dc=bolido,dc=pcg,dc=ull,dc=es"
```

3.2.3. Slurm

Slurm [15] es un sistema de gestión de sistemas y de clústeres, y está compuesto por distintas herramientas y servicios.

Dentro de los servicios, existen:

- **Slurmctld:** controlador del *cluster*.
- **Slurmd:** demonio de los nodos del *cluster*.
- **Slurmdbd:** interfaz para interactuar con la base de datos de Slurm. No se configurará porque para el uso de este *cluster* no aporta ninguna funcionalidad útil.

El cliente puede hacer uso de los siguientes comandos:

- **sacct:** informa de trabajos activos o completados.
- **salloc:** reserva recursos para un trabajo.
- **sttach:** permite vincularse a las entradas y salidas de un trabajo en ejecución.
- **sbatch:** utilidad para enviar un script como trabajo.
- **sbcast:** transfiere un fichero desde el disco local a un nodo.
- **scancel:** cancelar tareas pendientes o en ejecución.
- **scontrol:** es una herramienta administrativa utilizada para ver o modificar el estado de Slurm.
- **sinfo:** muestra el estado de las particiones y nodos gestionados por slurm.

- **sprio:** muestra información detallada de los componentes que afectan a la prioridad de un trabajo.
- **squeue:** muestra información sobre el estado de los trabajos.
- **srun:** permite lanzar trabajos para ejecutar.
- **sshare:** muestra información detallada sobre el uso del *cluster*.
- **sstat:** obtiene información sobre los recursos utilizados por un trabajo.
- **strigger:** permite establecer, obtener o observar cuándo ocurre un evento, como trabajos que han alcanzado el tiempo límite.
- **sview:** interfaz gráfica que obtiene información del estado para trabajos, particiones y nodos gestionados por Slurm.

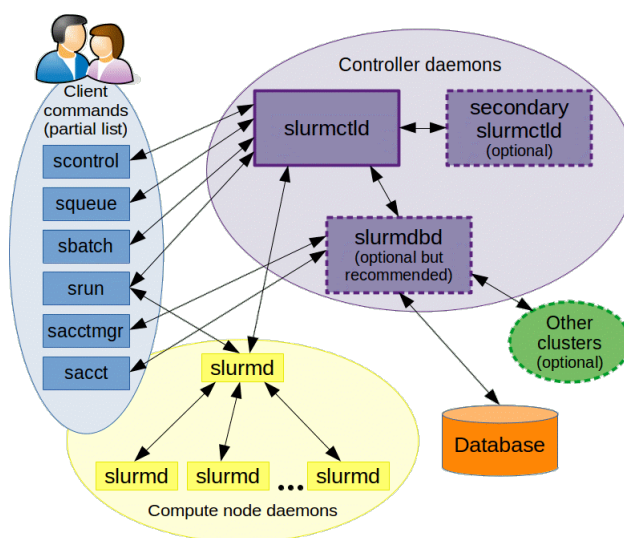


Figura 3.5: Componentes de slurm

Instalamos slurm en el servidor con:

```
$ apt-get install slurm-wlm slurm-client
```

Como slurmd no es necesario en el controlador, se puede desactivar el demonio con:

```
$ systemctl stop slurmd
$ systemctl disable slurmd
```

Para configurar Slurm (/etc/slurm-llnl/slurm.conf), se ha generado el fichero de configuración desde el generador de la documentación oficial [32].

Al final de este fichero, se ha incluido la información de los nodos que componen el cluster y se han creado 3 colas para lanzar trabajos.

Listado 3.9: Fichero /etc/slurm-llnl/slurm.conf

```
NodeName=rayo CPUs=48 Boards=1 SocketsPerBoard=4 CoresPerSocket=12
ThreadsPerCore=1 RealMemory=64408
NodeName=centella CPUs=48 Boards=1 SocketsPerBoard=8 CoresPerSocket=6
ThreadsPerCore=1 RealMemory=64392

PartitionName=rayo Nodes=rayo MaxTime=INFINITE State=UP
PartitionName=centella Nodes=centella MaxTime=INFINITE State=UP
PartitionName=short Nodes=rayo,centella Default=YES MaxTime=INFINITE State=UP
```

3.2.4. Lmod

Lmod [33] es una herramienta que permite personalizar las variables de entorno del usuario. Dentro del cluster, será útil para poder cargar distintas herramientas y poder utilizar distintas versiones de una misma, evitando problemas a la hora de trabajar con dependencias y facilitando la gestión del entorno.

Primero hay que instalar Lua [33]. Para ello, se descarga la última versión y se almacena en /opt/storage/downloads.

```
$ wget https://sourceforge.net/projects/lmod/files/lua-5.1.4.9.tar.bz2
```

A continuación, se instala lua en /opt/soft con su versión correspondiente y se debe crear el enlace simbólico al binario en el sistema.

```
$ tar xf lua-5.1.4.9.tar.bz2
$ cd lua-X.Y.Z
$ ./configure --prefix=/opt/soft/lua/5.1.4.9
$ make; make install
$ cd /opt/soft/lua; ln -s 5.1.4.9 lua
$ mkdir /usr/local/bin; ln -s /opt/soft/lua/5.1.4.9/lua/bin/lua /usr/local/bin
```

Para instalar lmod, se debe crear la carpeta /opt/soft/lmod/, clonar el repositorio oficial y compilarlo:

```
$ git clone https://github.com/TACC/Lmod
$ cd lmod
$ ./configure --prefix=/opt/soft
$ make install
```

A continuación, hay que crear un enlace simbólico con el script generado por lmod y vincularlo con el /etc/profile.d del sistema, para que cuando un usuario ejecute una terminal se inicialicen las variables de entorno de lmod.

```
ln -s /opt/apps/lmod/lmod/init/profile /etc/profile.d/z00_lmod.sh
```

Finalmente, se puede comprobar si lmod está correctamente instalando comprobando su variable de entorno:

```
$ echo $MODULEPATH
```

Para proveer de herramientas a los usuarios, se deben crear unos ficheros llamados modulefiles, en los que se indican como debe modificarse las variables del entorno del sistema (frecuentemente, \$PATH y \$LD_LIBRARY_PATH).

La organización para los programas que se ha utilizado es la siguiente. Primero, se ha creado el directorio /opt/soft/modulefiles/Linux e, internamente, una carpeta para cada herramienta. Dentro de cada herramienta, se crea un fichero .lua cuyo nombre es la versión que carga de esa herramienta, permitiendo así tener distintas versiones de un mismo programa o librería.

Algunos comandos básicos para utilizar lmod son los siguientes:

- Mostrar el software disponible:

```
$ module spider
```

- Cargar en el entorno software concreto:

```
$ module load name/version
```

- Mostrar modulefile de un software concreto:

```
$ module show name/version
```

3.2.5. Quota

Para evitar el consumo excesivo de espacio de disco por parte de los usuarios, se ha limitado utilizando quota.

```
$ apt-get install quota
```

Hay que modificar el fichero /etc/fstab, añadiendo que se van a utilizar las cuotas de usuario sobre la partición /home.

Listado 3.10: Configuración /etc/fstab en Bolido para Quotas

```
# ...  
/dev/mapper/bolido—home-home /home ext4 defaults,usrquota 0 2  
# ...
```

A continuación, se activan las cuotas de usuario en /home con el siguiente comando:

```
$ quotaon -uv /home
```

Para facilitar la creación de usuarios y la asignación de cuotas, se creó un script para el administrador de sistemas.

3.2.6. Script de gestión de usuarios

Para facilitar la gestión de usuarios dentro del *cluster*, se programó una pequeña herramienta en bash con la que crear los usuarios en LDAP y asignarles automáticamente las cuotas (ver apéndice 1).

La herramienta se encuentra guardada en `/usr/local/sbin` para que solo esté disponible para root bajo el nombre de `cluster-user-management`.

Ejemplos de uso:

Crear usuario test con contraseña testbolido y asignar 25G de cuota de disco:

```
$ ./cluster-user-management -a test -q 25G
```

Modificar las cuotas de disco del usuario test a 500 megas:

```
$ ./cluster-user-management -m test -q 500M
```

Eliminar usuario test:

```
$ ./cluster-user-management -d test
```

3.3. Configuración de los nodos

3.3.1. NFS

Autofs permite montar sistemas de archivos bajo demanda. Esto es especialmente útil para los nodos del cluster.

Se instala autofs con:

```
$ apt-get install autofs
```

Después, hay que crear el fichero `/etc/auto.master`, en el que añade al final del fichero la ruta local de montaje y el fichero de referencia:

```
/home    /etc/auto.home  
/opt     /etc/auto.opt
```

A continuación, se crea el fichero `/etc/auto.home` con:

```
* -fstype=nfs4 bolido :/home/&
```

Y un fichero `/etc/auto.opt` con:

```
* -fstype=nfs4 bolido :/opt/&
```

En ambos casos, se especifica que se monte utilizando `nfs4` todo el contenido de las carpetas `/home` y `/opt`, incluyendo subdirectorios de estos.

3.3.2. LDAP

Se instala LDAP en el cliente [34] con:

```
$ sudo apt-get install libnss-ldapd nscd nslcd
```

Se realiza la misma configuración que en el servidor, configurando nscd y nslcd.

Es importante que se instale libnss-ldapd en lugar de libnss-ldap. Esta versión es una mejora que ofrece mejor depuración, mejor rendimiento y separación del código de NSS, LDAP y PAM [29].

Por último, se debe indicar en el fichero /etc/nsswitch.conf [35] (fichero de configuración de la base de datos del sistema) para que utilice LDAP como servicio de autenticación.

Listado 3.11: Fichero /etc/nsswitch.conf

```
passwd:      files ldap systemd
group:       files ldap systemd
shadow:      files ldap
gshadow:     files ldap
```

Es importante que LDAP se coloque antes que systemd para que al realizar la autenticación, el sistema intente primero rescatar la información de las bases de datos, después de LDAP y, en caso de no lograrlo, pregunte a los demonios del sistema.

Se puede comprobar que el servicio está funcionando correctamente si el siguiente comando nos devuelve la información del /etc/passwd de LDAP:

```
$ getent passwd
```

3.3.3. Slurm

Se instala slurm en el cliente con:

```
$ apt-get install slurm-wlm
```

Se debe copiar el fichero de configuración que se encuentra en Bolido y guardarlo en /etc/slurm-llnl bajo el nombre de slurm.conf. También se debe generar una clave privada y un certificado para el nodo.

```
$ openssl genrsa -out /etc/slurm-llnl/slurm.key 1024
$ openssl rsa -in /etc/slurm-llnl/slurm.key -pubout
  -out /etc/slurm-llnl/slurm.cert
```

Con el siguiente comando, se puede generar la información sobre el nodo para incluirlo en el fichero de configuración de slurm:

```
$ slurmd -C
```


3.3.4. Bloqueo de autenticación de usuarios

Se puede evitar que los usuarios no puedan conectarse por SSH a los nodos, manteniendo la posibilidad de que lancen trabajos con slurm. Para ello, hay que instalar libpam-slurm y modificar cómo se realiza la autenticación de SSH a través de las PAM.

```
$ apt-get install libpam-slurm
```

Por último, se modificó el fichero /etc/pam.d/sshd. Se añadió la siguiente línea para permitir autenticación local para el usuario de administración:

```
$ account sufficient pam_localuser.so
```

De acuerdo a la documentación [36], se añadió también esta línea como última opción de las cuentas, para bloquear a todos los usuarios que no tengan una instancia de slurm en la máquina:

```
$ account required pam_slurm_adopt.so
```

3.4. Instalación de herramientas

Todas las herramientas que se proveen dentro del cluster se encuentran compartidas por NFS en el directorio /opt/soft y los usuarios del cluster pueden cargarlas y seleccionar entre distintas versiones disponibles gracias a lmod.

3.4.1. OpenMPI

OpenMPI es una implementación del estándar MPI [37] (Interfaz de Paso Mensajes), que define la sintaxis y la semántica de las funciones que deben encontrarse en una librería de pasos de mensajes para ser utilizada por programas que aprovechan la existencia de distintos procesadores.

Para instalar OpenMPI, primero hay que descargarlo del directorio oficial:

```
$ wget https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.6.tar.gz
```

A continuación, se crean los directorios donde se almacenará OpenMPI, se instala en el sistema varias librerías necesarias para que openmpi y slurm funcionen correctamente, y se configura y compila OpenMPI.

```
$ mkdir -p /opt/soft/openmpi/3.1.6
$ sudo apt-get install -y libpmi2-0-dev libpmi2-0
$ ./configure --prefix=/opt/soft/openmpi/3.1.6 --with-slurm --with-pmi
--with-pmi-libdir=/usr/lib/x86_64-linux-gnu
$ make install all
```

Por último, se crea el modulefile para OpenMPI, en el que se incluyen los binarios a la variable PATH del sistema y las librerías dinámicas a la variable LD_LIBRARY_PATH.

Listado 3.12: Modulefile de OpenMPI

```
help([  
For detailed instructions , go to:  
    https://www.open-mpi.org/  
])  
whatis("Version:_3.1.6")  
whatis("Keywords:_Compiler_OpenMPI")  
  
setenv("OPENMPIPATH", "/opt/soft/openmpi/3.1.6/bin")  
prepend_path("PATH", "/opt/soft/openmpi/3.1.6/bin")  
prepend_path("LD_LIBRARY_PATH", "/opt/soft/openmpi/3.1.6/lib")
```

Se realiza el proceso equivalente para instalar la versión 4.1.1.

3.4.2. OpenBLAS

OpenBLAS [38] es una implementación open source de las API de BLAS (Basic Linear Algebra Subprograms) y LAPACK con distintas optimizaciones realizadas manualmente para distintos tipos de procesador.

Se descarga y compila OpenBLAS con:

```
$ git clone https://github.com/xianyi/OpenBLAS.git  
$ mv OpenBLAS /opt/soft/openblas/3.14  
$ make
```

OpenBLAS solo se puede utilizar como librería estática o dinámica, por lo que solo se necesita poner a disposición del usuario los ficheros generados que se encuentran en lib mediante la variable de entorno LD_LIBRARY_PATH.

Listado 3.13: Modulefile de OpenBLAS

```
help([  
For detailed instructions , go to:  
    https://github.com/xianyi/OpenBLAS  
])  
whatis("Version:_3.14")  
whatis("Keywords:_OpenBLAS")  
  
prepend_path("LD_LIBRARY_PATH", "/opt/soft/openblas/3.14/lib")
```

3.4.3. GSL

GNU Scientific Library [39] (GSL) es una librería numérica para programadores de C / C++.

Para instalar GSL, se descarga el paquete, se configura la carpeta de salida y se compila con las siguientes instrucciones:

```

$ wget https://mirrors.up.pt/pub/gnu/gsl/gsl-latest.tar.gz
$ tar xf gsl-latest.tar.gz
$ ./configure --prefix=/opt/soft/gsl/2.7
$ make install

```

Por último, se genera el modulefile para poner a disposición de los usuarios la librería:

Listado 3.14: Modulefile de GSL

```

help([
  For detailed instructions, go to:
    https://www.gnu.org/software/gsl/
])
whatis("Version:_2.7")
whatis("Keywords:_Compiler_GSL")

prepend_path( "PATH", "/opt/soft/gsl/2.7/bin")
prepend_path( "LD_LIBRARY_PATH", "/opt/soft/gsl/2.7/lib")

```

3.4.4. Intel Performance Tools

Intel provee de un conjunto de herramientas para computación de alto rendimiento: compiladores, librerías y herramientas para análisis, depuración y tuneado.

Para instalar OneApi, se añadió el repositorio de intel a apt y se descargó el paquete de OneApi [40] y, con todos los paquetes descargados, se instalaron individualmente en el directorio /opt/soft.

```

$ wget https://apt.repos.intel.com/intel-gpg-keys/
  GPG-PUB-KEY-INTEL-SW-PRODUCTS.PUB -O - |
  sudo apt-key add -
$ echo "deb_https://apt.repos.intel.com/oneapi_all_main" |
  sudo tee /etc/apt/sources.list.d/oneAPI.list
$ sudo apt install intel-basekit
$ cd /var/cache/apt/archives
$ packages = $(ls intel*.deb)
$ for i in ${packages[@]}; do
  dpkg-deb -x $i /opt/soft/intel/oneapi/;
done

```

Las herramientas de intel proveen de un script que genera automáticamente los modulefiles configurados.

Simplemente, se ejecutó el script y se copiaron todos los ficheros generados en nuestro directorio de modulefiles.

```

$ ./modulefiles-setup.sh
$ mv modulefiles /opt/soft/modulefiles/Linux/oneapi

```

3.5. Problemas

En este apartado se comentan algunas de las dificultades encontradas a la hora de configurar el cluster.

3.5.1. Configuración de Bolido en Raid 1

En una aproximación inicial, se intentó configurar la máquina de Bolido con los discos en Raid 1 via hardware. Aunque los discos fueran del mismo tamaño (500gb), no se pudo realizar esta configuración porque el hardware no permite configurar RAID 1 con discos de distinto tipo: SATA y SAS.

Como solución final, se montó el sistema en LVM, utilizando el disco SAS para el sistema y el SATA como almacenamiento.

3.5.2. Instalación de slurm con compatibilidad con OpenMPI

La instalación correcta de OpenMPI para que funcionara correctamente con Slurm fue bastante problemática, especialmente porque surgían problemas a la hora de lanzar tareas a los nodos con MPI.

El problema fue que, tanto slurm como OpenMPI, debían depender de la misma librería y compartirla: libpmi.

Para solucionarlo, se instaló en el sistema libpmi y libpmi2 y se le indicó en la compilación de OpenMPI que tomara el libpmi instalado en el sistema.

3.5.3. Errores al configurar el cluster

Múltiples veces la configuración del controlador o de los nodos fallaba por distintas razones, como tener ficheros de configuración distinto o que el propio usuario del *cluster* no tuviera el mismo UID. Para encontrar estos problemas y solucionarlos, ejecutaba el controlador de slurm o el demonio del nodo con los siguientes comandos:

```
$ slurmctld -c -vvvv -D
$ slurmd -c -vvvv -D
```

También se encontró problemas para configurar bien la autenticación con LDAP. Para lograrlo, se tuvo que emplear los servicios afectados (nslcd) en modo de depuración para poder detectar problemas, como por ejemplo intento de conexiones a LDAP por SSL:

```
$ nslcd -d
```

En este caso en específico, el siguiente comando fue de utilidad para encontrar problemas en los ficheros de configuración. De esta manera, con expresiones regulares, se muestra la información del fichero eliminando todos los comentarios que suelen traer los ficheros de configuración de Linux.

```
$ cat /etc/nslcd.conf | grep -v ^# | sed '/^$/d'
```

También fue de utilidad los siguientes comandos para depurar el funcionamiento correcto del sistema:

```
$ id <ldap_user>  
$ getent passwd  
$ getent shadow
```

Con `getent`, se comprueba que el servicio NSS está obteniendo correctamente las bases de datos de LDAP y con `id` se comprueba la existencia del usuario. En caso de consultar un usuario de LDAP y que este no aparezca, implica que existe algún error en el que el sistema no puede autenticar al usuario con LDAP a través de las PAM [41] (Pluggable Authentication Modules).

3.5.4. Error espacio al actualizar de Debian 9 a Debian 10 (Centella)

La partición raíz de Centella no era lo suficientemente grande como para almacenar la actualización del sistema. Debido a que el particionado no era el adecuado y que la partición contigua a la raíz era necesaria para el funcionamiento correcto del sistema, se procedió a instalar el sistema desde cero.

Capítulo 4

Análisis de resultados

4.1. Análisis del rendimiento

Tras configurar el cluster, se realizaron algunos experimentos con el fin de comprobar el funcionamiento correcto y realizar un análisis del rendimiento de las máquinas.

Para hacer pruebas, se utilizó un programa sencillo que calcula el valor de pi encontrado en un repositorio de github [42]. Este programa se utilizó simplemente para comprobar que todo funcionaba y familiarizarse con slurm.

Después, para realizar un análisis de rendimiento, se utilizó NPB: un *benchmark* perteneciente a la NASA, interesante especialmente porque las distintas pruebas de paralelización utilizan la memoria de distintas formas.

4.1.1. Métricas

El objetivo principal del análisis de rendimiento es conseguir los mejores resultados en el menor tiempo posible y consumiendo el menor número de recursos posible.

A la hora de determinar el rendimiento de una arquitectura paralela, es común medir la eficiencia mediante estas dos medidas [43]:

- **Tiempo de ejecución o respuesta** de una aplicación.
- **Productividad** (*throughput*), el número de aplicaciones que es capaz de procesar por unidad de tiempo.

Sin embargo, aunque ofrecen información útil, son medidas demasiado simples, por lo que se introducen otras métricas como el *speedup*, la eficiencia, la utilización, la redundancia y la calidad del paralelismo.

Aceleración

El *speedup* mide la relación entre el tiempo de ejecución de un algoritmo en paralelo con respecto a lo que tarda el mismo algoritmo en resolver el problema de forma secuencial.

$$S(n) = \frac{T(1)}{T(n)}$$

Generalmente, speedup tiene un valor entre 0 y p (número de procesadores), donde lo ideal sería que fuera igual a p (Figura 4.2). Sin embargo, también cabe la posibilidad que el valor sea mayor que p, causando así un speedup superlineal. Cuando esto ocurre, suele deberse a la gestión interna de la memoria en el caso del algoritmo paralelo, en el que cada procesador trabaja con menos datos que en el algoritmo secuencial.

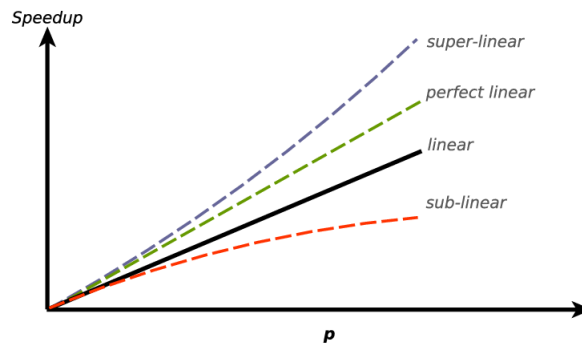


Figura 4.1: SpeedUp

Eficiencia

La eficiencia mide la fracción de tiempo durante la cual se utiliza de manera útil un procesador. Se obtiene dividiendo el speedup entre el número de procesadores.

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n * T(n)}$$

Esta medida tendrá un valor entre 0 y 1, siendo 1 el valor ideal. Como ocurre con el speedup, también podemos obtener una eficiencia superior, si el speedup es superlineal.

Redundancia

La redundancia en un cálculo paralelo se define como el número total de operaciones ejecutadas con n procesadores dividido por el número de operaciones ejecutadas cuando utilizamos un único procesador.

$$R(n) = \frac{O(n)}{O(1)}$$

Utilización

La utilización de un sistema indica el porcentaje de recursos que se utilizan durante la ejecución de un programa paralelo.

$$U(n) = R(n)E(n) = \frac{O(n)}{n * T(n)}$$

Calidad del paralelismo

La calidad de un cálculo paralelo es directamente proporcional al speedup y la eficiencia, inversamente proporcional a la redundancia.

$$Q(n) = \frac{S(n) * E(n)}{R(n)} = \frac{T^3(1)}{n * T^2(n) * O(n)}$$

Al combinar el speedup, la eficiencia y la redundancia en una expresión, con la calidad medimos el mérito relativo de un cálculo paralelo en un sistema.

4.2. Cálculo de PI

Se utilizó un programa de ejemplo del calculo de PI con OpenMPI de un repositorio de Github [44] para hacer pruebas rápidas y ver que el cluster se encontraba funcionando correctamente. Fue especialmente útil para interactuar con el sistema como si fuera un usuario para detectar errores y aspectos a mejorar, así como familiarizarse con slurm y comprender su funcionamiento.

Por ejemplo, con los siguientes comandos, se carga el modulo de OpenMPI, se reservan 16 CPUs de los 2 nodos que hay en el sistema y se lanza un trabajo que ejecutará el cálculo de Pi.

```
$ module load openmpi
$ salloc -N 2 -n 16
$ srun -N 2 -n 16 --mpi=pmi2 ./pi
```

4.3. NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB) es un conjunto de programas que fueron diseñados para evaluar el rendimiento de ordenadores paralelos [45]. El banco de pruebas está compuesto distintos tipos de pruebas:

- **IS:** Integer Sort, acceso aleatorio a memoria.
- **EP:** Embarrassingly Parallel: problemas en los que no hay apenas dificultades en separarlos en tareas paralelas.
- **CG:** Conjugado del gradiente. Acceso irregular a memoria.
- **MG:** Multi Grid en una secuencia de mallas. Utiliza memoria de forma intensiva.
- **FT:** transformada de Fourier en 3 dimensiones. Intensivo en comunicaciones
- **BT:** Resuelve matrices tridiagonales por bloques.
- **SP:** Resuelve matrices pentadiagonales.
- **LU:** Resuelve sistemas de ecuaciones lineales mediante Gauss-Seidel.

NAS provee de distintas clases, con la que permite aumentar el tamaño del problema a solucionar:

- **S y W:** pruebas rápidas.

- **A, B, C:** pruebas de tamaño estándar, en el que el tamaño del problema es aproximadamente 4 veces mayor de una clase a la siguiente.
- **D, E, F:** pruebas grandes, en los que el tamaño es, aproximadamente, 16 veces mayor que la clase anterior.

Para realizar el análisis de rendimiento, se compilaron todas las pruebas para un tamaño de problema de tipo C: lo suficiente para obtener resultados con los analizar el rendimiento pero que no llevaran un tiempo de cálculo excesivamente elevado.

```
$ make <prueba> CLASS=C
```

4.3.1. Generación de pruebas y filtración

Para generar las pruebas, se crearon dos scripts: uno que ejecuta iterativamente todos los programas de NPB, y otro que lanza como trabajos veinte ejecuciones de este script con sbatch, variando el número de procesadores, el nodo y el nombre del fichero de salida.

Se utilizaron los siguientes procesadores para las pruebas:

Número de procesadores	1	2	4	8	16	25	32	36	48
------------------------	---	---	---	---	----	----	----	----	----

Los valores que no son múltiplo de 2 (25 y 36) se utilizaron porque algunos programas de NPB piden como requisito que el número de procesadores empleado sea un número al cuadrado. El caso de 48 núcleos se utilizó porque es el máximo permitido en las máquinas.

Listado 4.1: Script scripts/generate.sh

```
#!/bin/bash

for i in $(ls /home/pladmin/pruebas/scripts/bin)
do
    echo "EJECUCION_DE_${i}"
    mpirun /home/pladmin/pruebas/scripts/bin/${i}
done
```

Finalmente, lanzamos todos los trabajos ejecutando el siguiente script:

Listado 4.2: Script run.sh

```
#!/bin/bash

for iteracion in {1..20}
do
  for procesador in {1,2,4,8,16,25,32,36,48}
  do
    for nodo in "rayo" "centella"
    do
      sbatch -o "scripts/output/"$nodo"-n"$procesador_"$iteracion".log"
      -p $nodo -n $procesador
      -J $nodo"-n"$procesador_"$iteracion "scripts/generate.sh"
    done
  done
done
```

Cada trabajo guarda las ocho ejecuciones de los distintos programas en un fichero .log formado por el nombre de los nodos, de los procesadores y el número de la iteración.

También se generaron dos scripts adicionales, uno con el que se genera un nuevo fichero .log que contiene todas las ejecuciones con distintos procesadores para cada tipo de programa y otro que, a partir de este último, crea un fichero CSV con todos los datos para procesarlos fácilmente en Python y poder generar las estadísticas que veremos a continuación.

Listado 4.3: Script generador_csv.sh

```
#!/bin/bash

files=$(ls original)

# Create CSV
for file in $files;
do
  seconds=$(cat original/$file | grep "Time_in_seconds" | tr -d "_" |
    cut -d "=" -f 2))
  mops_total=$(cat original/$file | grep "Mop/s_total" | tr -d "_" |
    cut -d "=" -f 2))
  mops_process=$(cat original/$file | grep "Mop/s/process" | tr -d "_" |
    cut -d "=" -f 2))
  iterations=$(cat original/$file | grep "Iterations_=====" | tr -d "_" |
    cut -d "=" -f 2))
  node=$(echo $file | cut -d "-" -f 1))
  process=$(cat original/$file | grep "EJECUCION_DE" | cut -d "_" -f 3 |
    awk -F "-" '{print_$2}' | cut -d "_" -f 1))

  echo "Time_in_seconds,MOPS_Total,_MOPS_Process,Iterations,Node,Process"
  >> "csv/"$file".csv"
```

```

for i in $(seq 0 "${#seconds[@]}");
do
    echo ${seconds[i]},${mops_total[i]},${mops_process[i]},${iterations[i]},
    $node,${process[i]} >> "csv/" $file ".csv"
done
done

```

4.4. Resultados

Para comprender bien los resultados, primero conviene explicar cómo funciona la jerarquía de memoria, el tamaño de problema con el que se ha trabajado y la capacidad de nuestro hardware.



Figura 4.2: Jerarquía de memoria

En la jerarquía de memoria de un computador, los componentes más rápidos son también los que menos capacidad tienen y los más caros. Cuando un programa se ejecuta, dependiendo de la cantidad de almacenamiento que necesite, utilizan los distintos recursos de la jerarquía según cuánto necesite almacenar y según la disponibilidad de los mismos.

Rayo y Centella cuentan con las siguientes características:

Máquina	L1	L2	L3	RAM
Rayo	1536KiB	6MiB	10MiB	64GB
Centella	576KiB	12MiB	12MiB	64GB

En todas las gráficas generadas se obtienen una serie de resultados en común.

- En las gráficas de tiempo de ejecución, se puede observar que Centella realiza los cálculos más rápido que Rayo. Sin embargo, cuanto más aumenta el número de procesadores, ambas máquinas tienden a tener un tiempo de ejecución similar.

- En las gráficas de la aceleración, la eficiencia, la utilización, la redundancia y la calidad del paralelismo, Rayo aparece generalmente por encima de Centella. Esto no significa que Rayo sea más potente ni más rápido, sino que Rayo se ve mucho más beneficiado de la ejecución paralela que Centella.
- Tanto para Rayo como para Centella ocurre speedup superlineal en la mayoría de algoritmos. Esto es así porque, tras utilizar un número determinado de procesadores, el tamaño del problema cabe en caché, que implica evitar el acceso a memoria RAM y, por tanto, acelerar muchísimo más los cálculos.

Los tamaños de problema para cada algoritmo son:

Algoritmo	Tamaño de problema C
IS	2^{27} claves con valor máximo de 2^{23}
EP	2^{32} pares de números aleatorios
CG	150000 filas, 75 iteraciones, 15 no ceros
MG	Matriz 512 x 512 x 512, 20 iteraciones
FT	Matriz 512 x 512 x 512, 20 iteraciones
BT	Matriz 162 x 162 x 162, 200 iteraciones
SP	Matriz 162 x 162 x 162, 400 iteraciones
LU	Matriz 480 x 320 x 28, 0.8GB de memoria necesaria

De todos los algoritmos, el que menos tiempo tardó en ejecutarse de media fue el de IS (Figura 4.8) y el que más el algoritmo de BT, matrices tridiagonales en bloque (Figura 4.4.6).

Una de las gráficas que llama más la atención es la de la Calidad del Paralelismo (Figura 4.19), especialmente Centella. En este caso, Centella presenta un buen speed up (Figura 4.4.3) y una buena eficiencia (Figura 4.16). Sin embargo, la redundancia es muy alta (Figura 4.18, por lo que la calidad del paralelismo es mala).

4.4.1. Gráficas IS

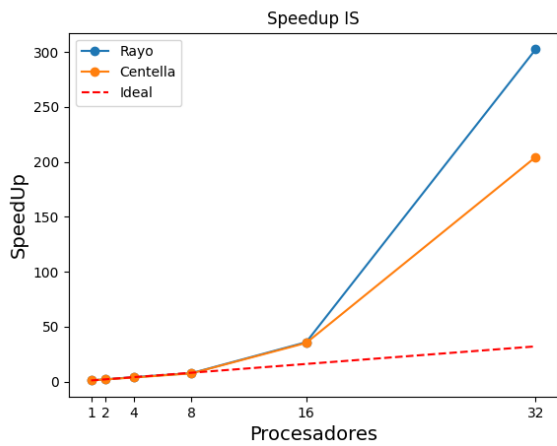


Figura 4.3: Speedup IS

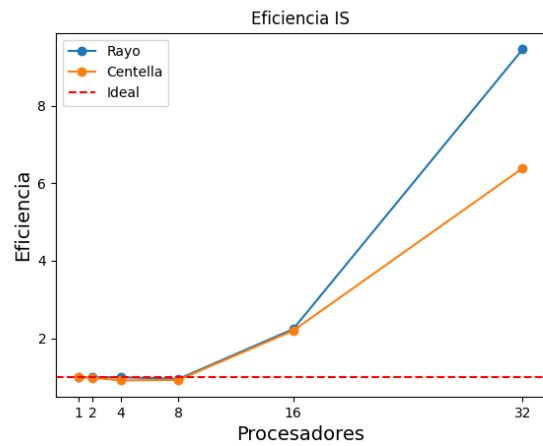


Figura 4.4: Eficiencia IS

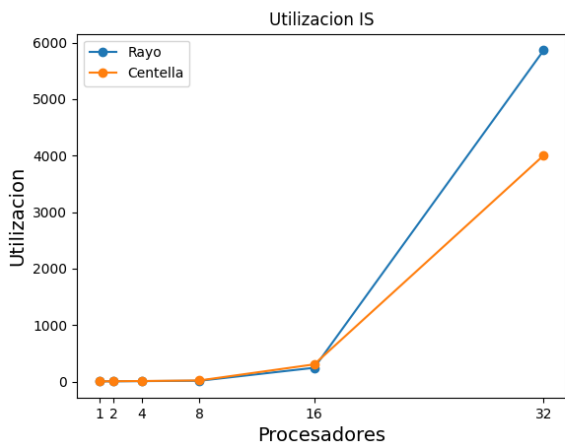


Figura 4.5: Utilización IS

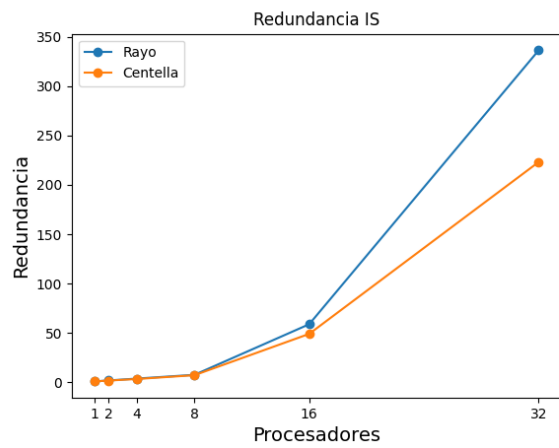


Figura 4.6: Redundancia IS

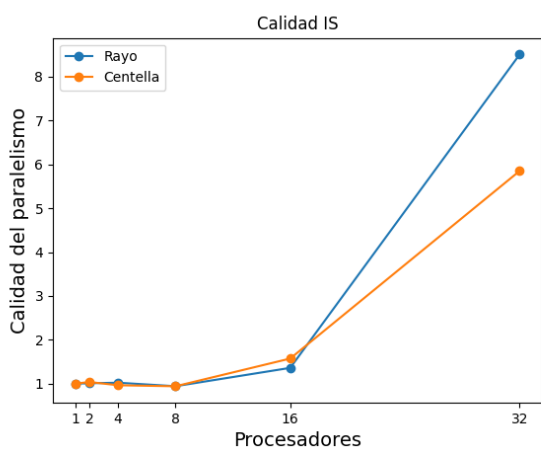


Figura 4.7: Calidad Paralelismo IS

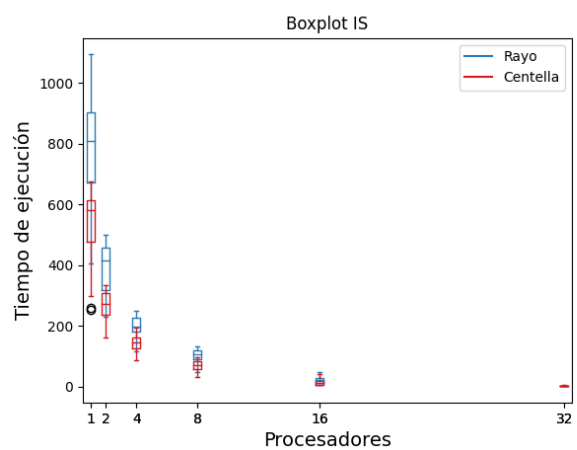


Figura 4.8: Tiempo Ejecución IS

4.4.2. Gráficas EP

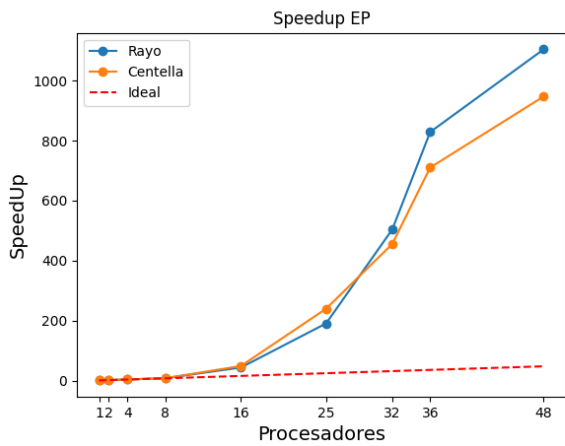


Figura 4.9: Speedup EP

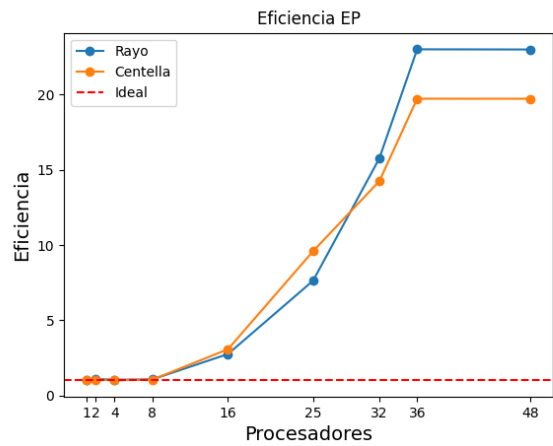


Figura 4.10: Eficiencia EP

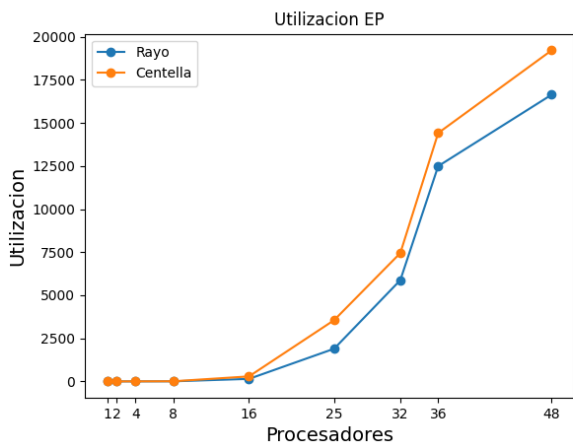


Figura 4.11: Utilización EP

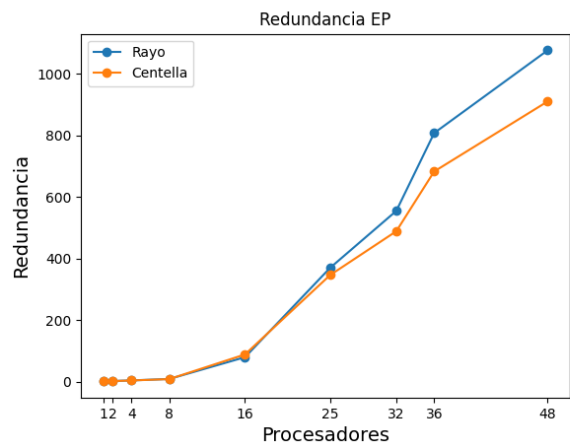


Figura 4.12: Redundancia EP

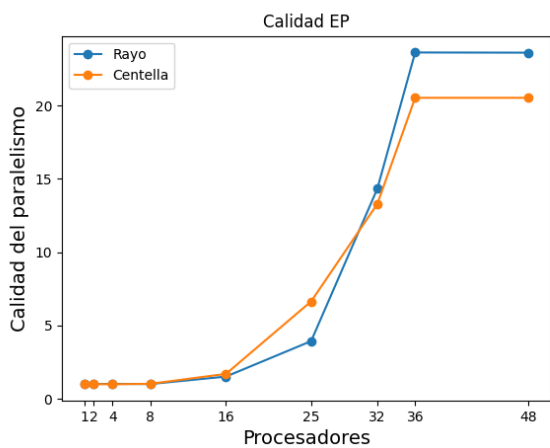


Figura 4.13: Calidad Paralelismo EP

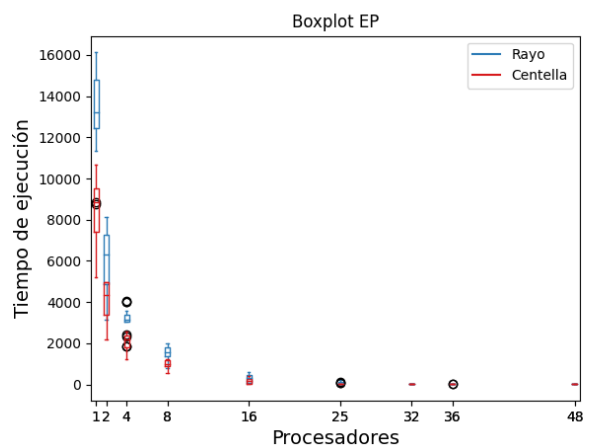


Figura 4.14: Tiempo Ejecución EP

4.4.3. Gráficas CG

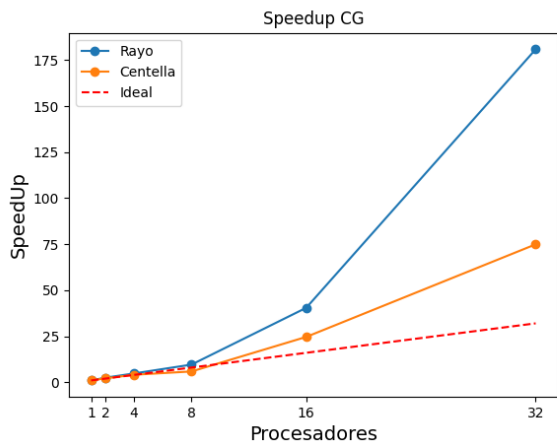


Figura 4.15: Speedup CG

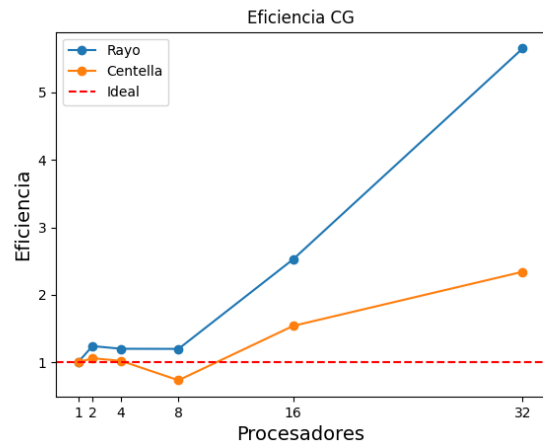


Figura 4.16: Eficiencia CG

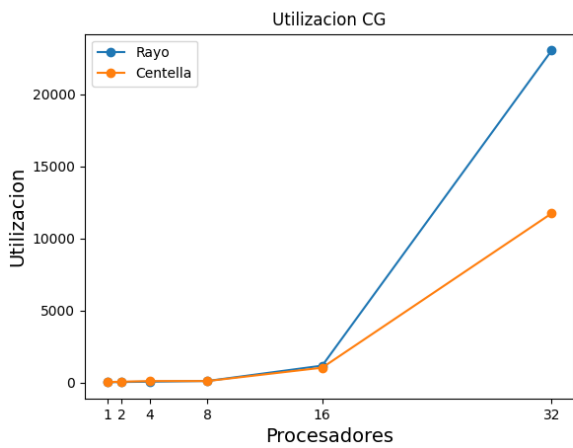


Figura 4.17: Utilización CG

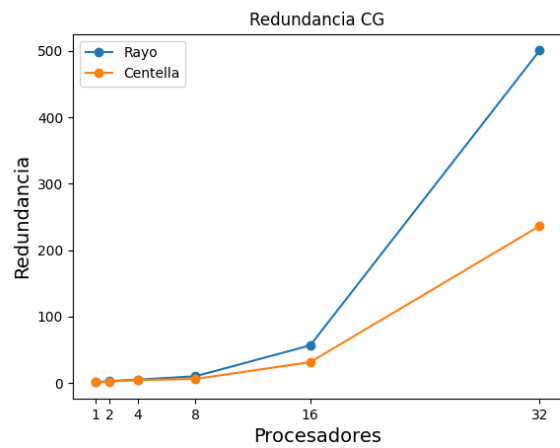


Figura 4.18: Redundancia CG

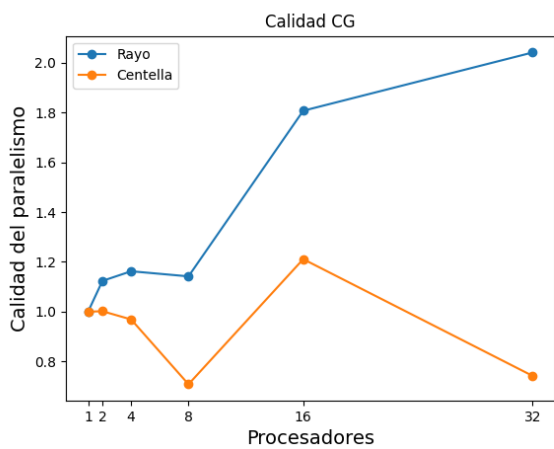


Figura 4.19: Calidad Paralelismo CG

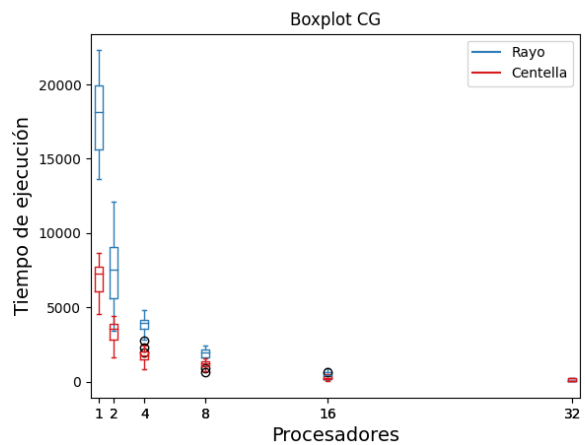


Figura 4.20: Tiempo Ejecución CG

4.4.4. Gráficas MG

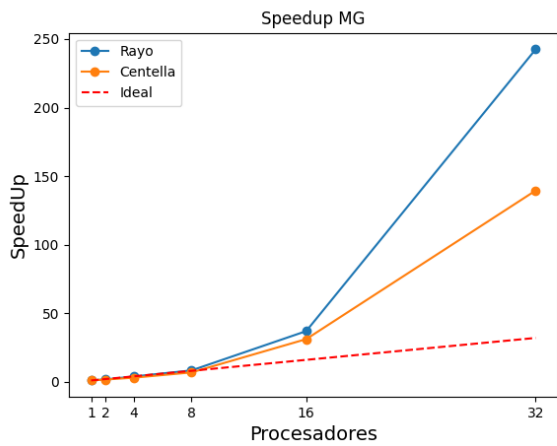


Figura 4.21: Speedup MG

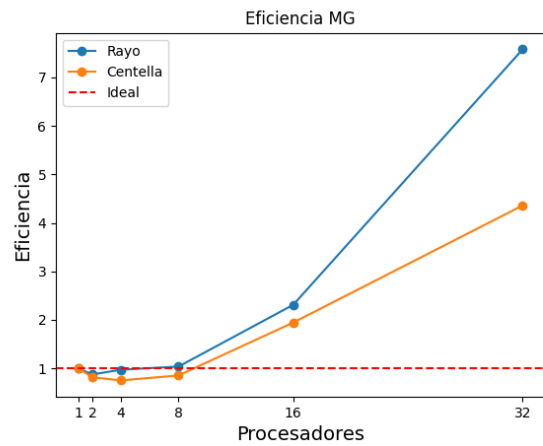


Figura 4.22: Eficiencia MG

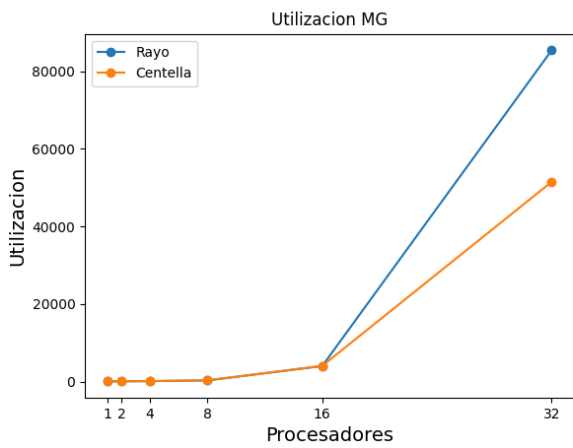


Figura 4.23: Utilización MG

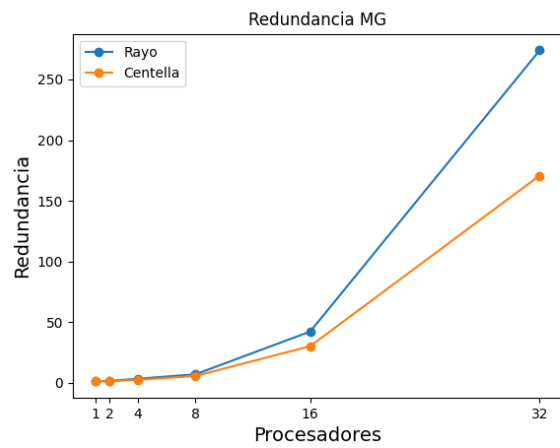


Figura 4.24: Redundancia MG

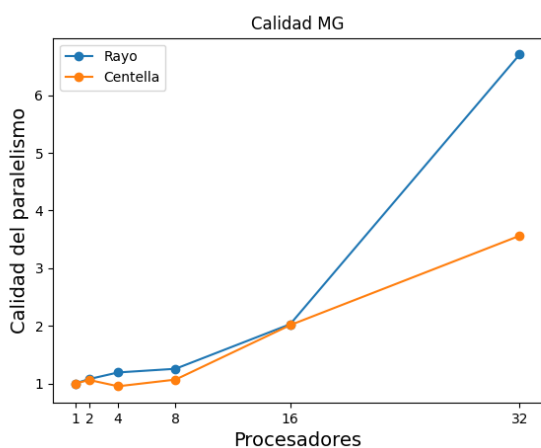


Figura 4.25: Calidad Paralelismo MG

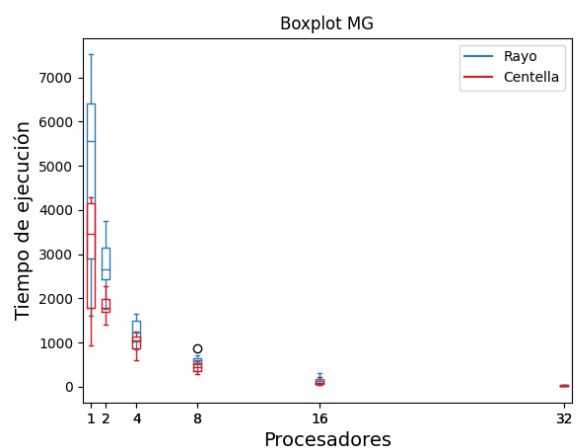


Figura 4.26: Tiempo Ejecución MG

4.4.5. Gráficas FT

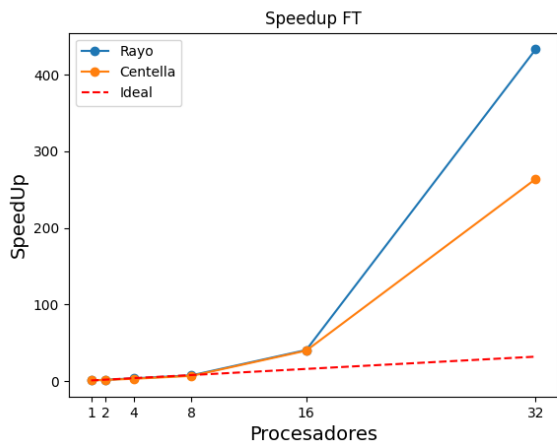


Figura 4.27: Speedup FT

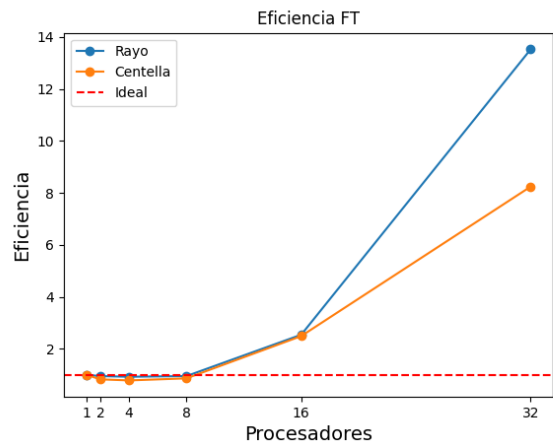


Figura 4.28: Eficiencia FT

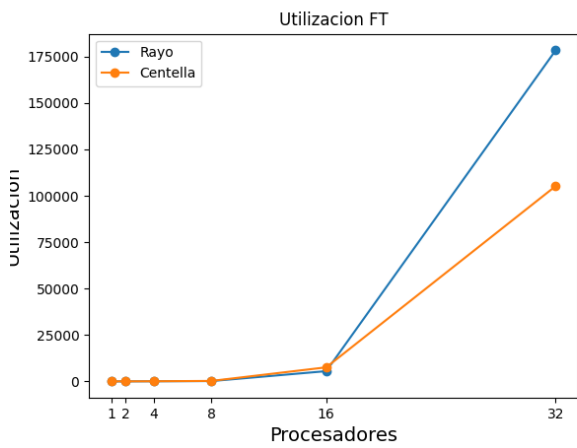


Figura 4.29: Utilización FT

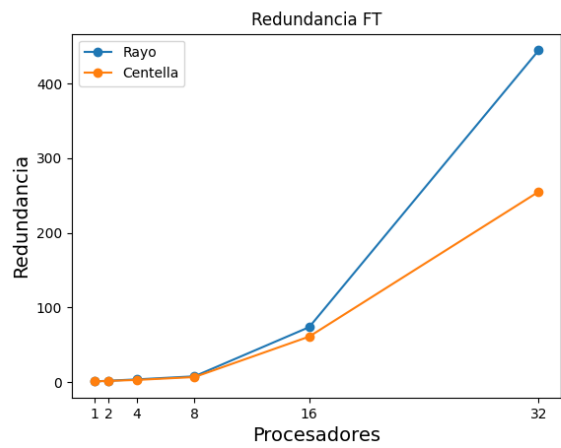


Figura 4.30: Redundancia FT

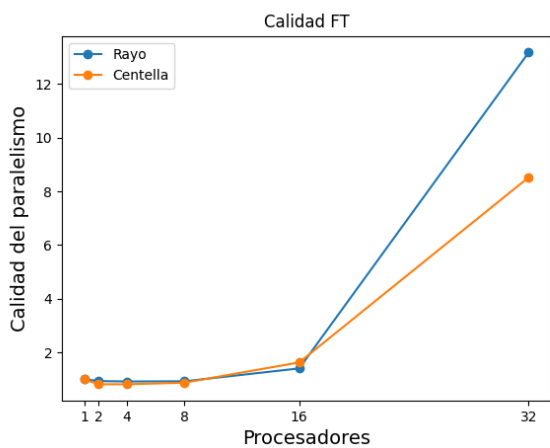


Figura 4.31: Calidad Paralelismo FT

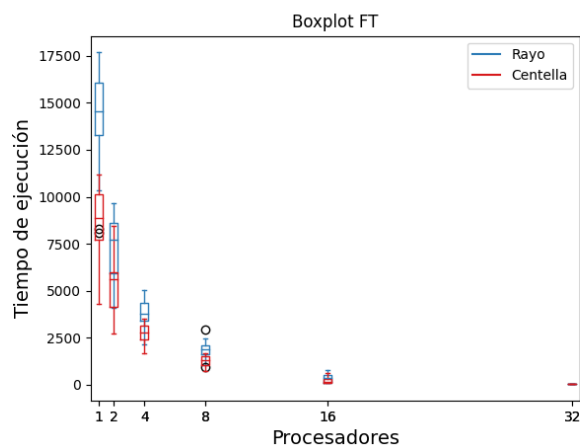


Figura 4.32: Tiempo Ejecución FT

4.4.6. Gráficas BT

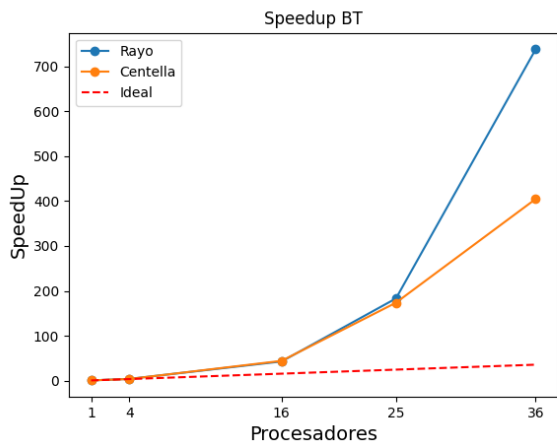


Figura 4.33: Speedup BT

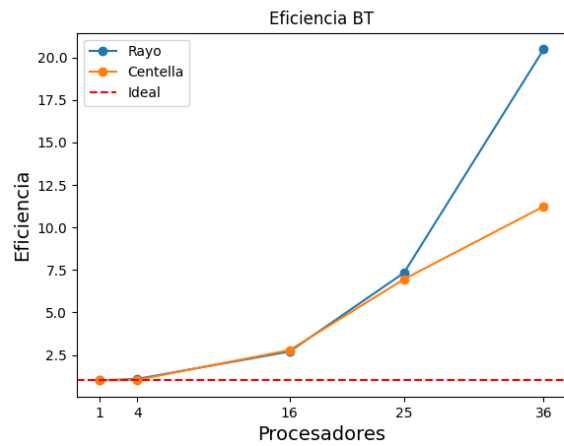


Figura 4.34: Eficiencia BT

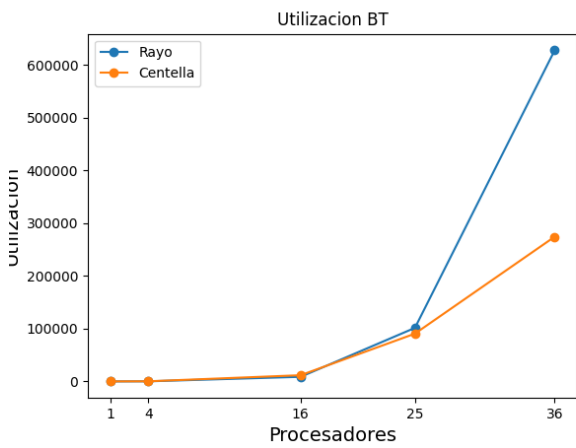


Figura 4.35: Utilización BT

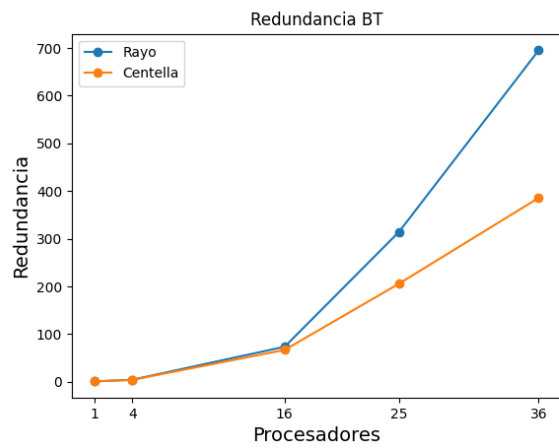


Figura 4.36: Redundancia BT

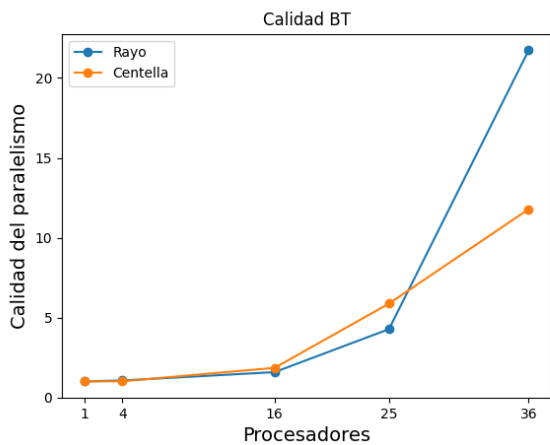


Figura 4.37: Calidad Paralelismo BT

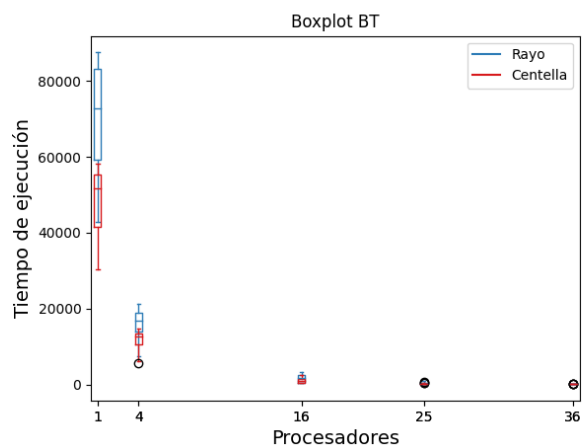


Figura 4.38: Tiempo Ejecución BT

4.4.7. Gráficas SP

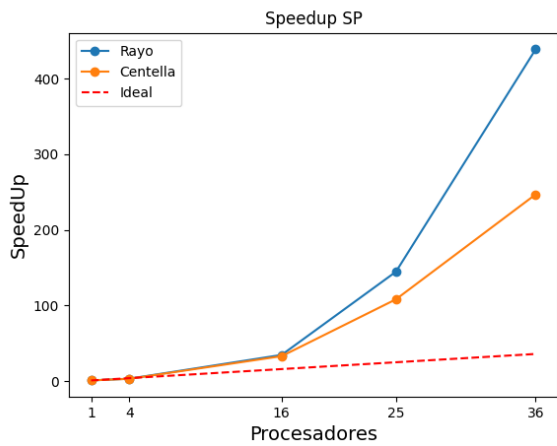


Figura 4.39: Speedup SP

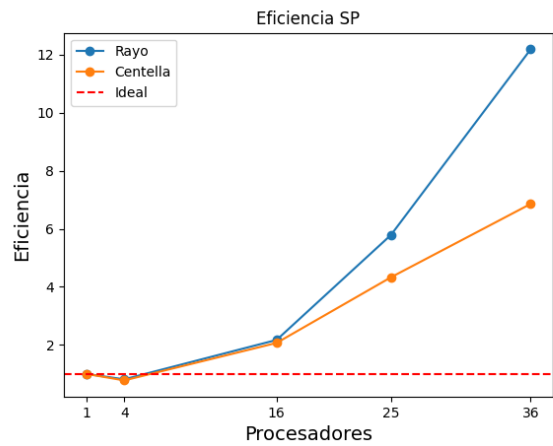


Figura 4.40: Eficiencia SP

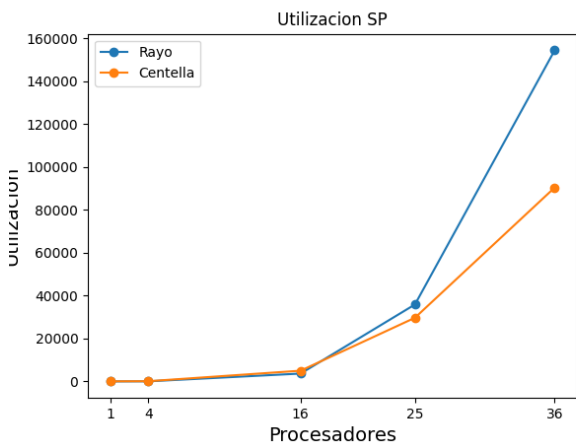


Figura 4.41: Utilización SP

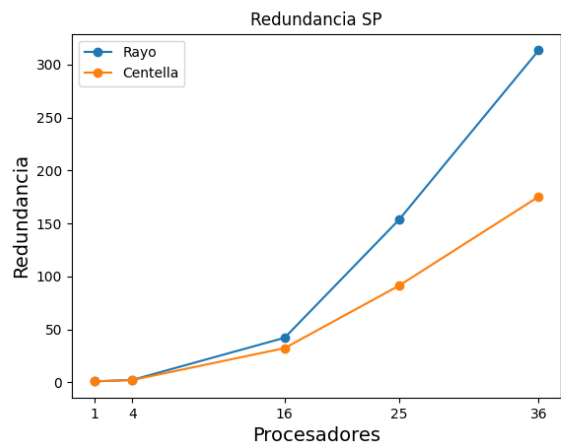


Figura 4.42: Redundancia SP

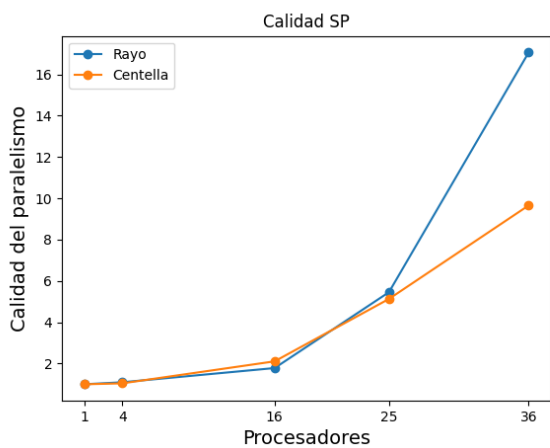


Figura 4.43: Calidad Paralelismo SP

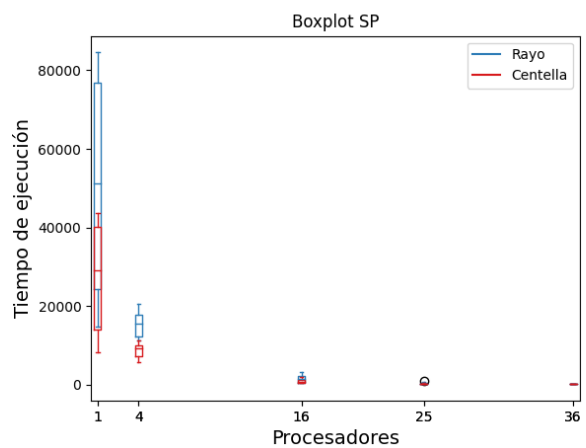


Figura 4.44: Tiempo Ejecución SP

4.4.8. Gráficas LU

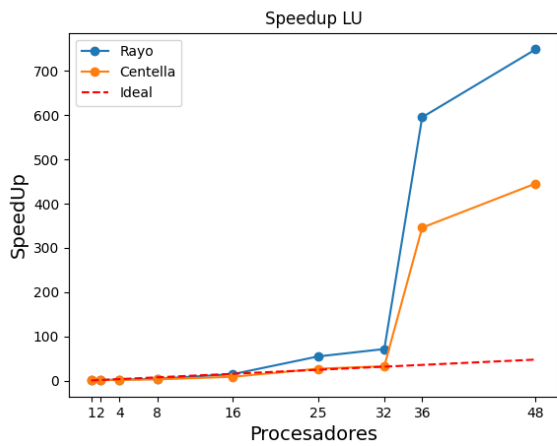


Figura 4.45: Speedup LU

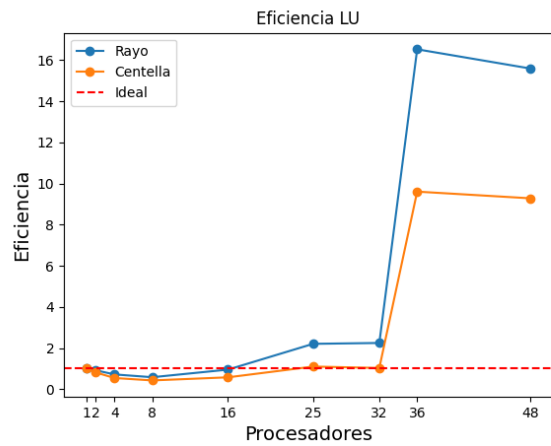


Figura 4.46: Eficiencia LU

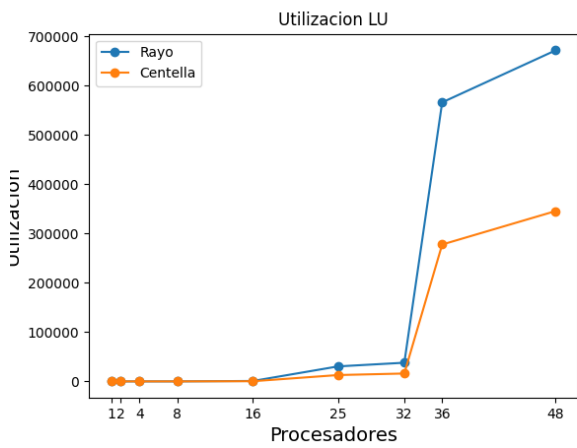


Figura 4.47: Utilización LU

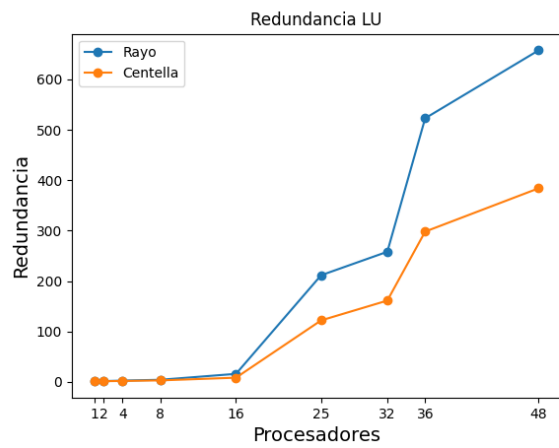


Figura 4.48: Redundancia LU

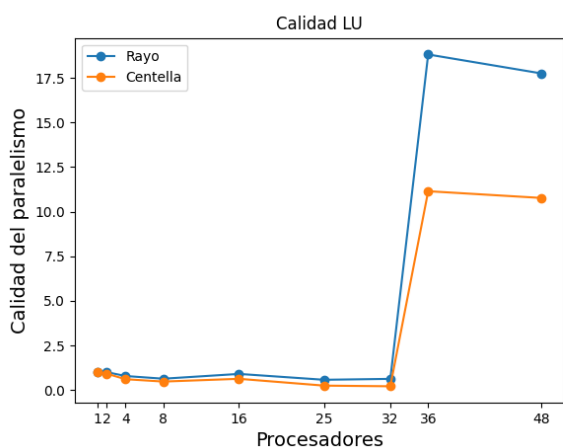


Figura 4.49: Calidad Paralelismo LU

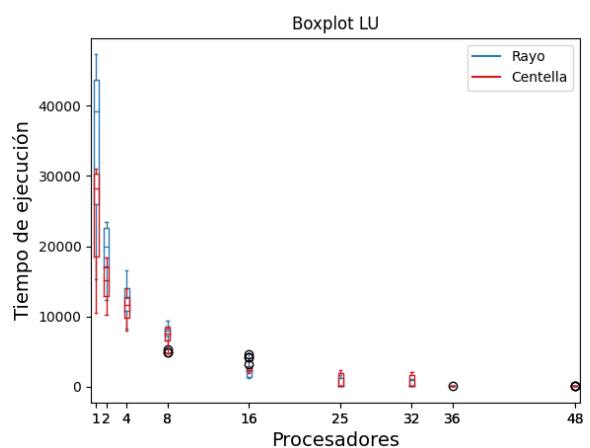


Figura 4.50: Tiempo Ejecución LU

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

La computación de altas prestaciones está a la última orden del día. La tendencia del mundo actual es seguir mejorando y exprimiendo al máximo el potencial de la tecnología, con el fin de avanzar en la ciencia.

Dos tendencias actuales lo confirman: el desarrollo de ordenadores cuánticos e instalación de más supercomputadores por el mundo y, quizás la más desapercibida, los desarrollos de vacunas del coronavirus, descubrimientos y estudios continuos en astronomía y el modelado del tiempo para su predicción.

Implementar un sistema de computación de altas prestaciones es relativamente sencillo. Aunque existen diversas soluciones empresariales, es posible crearse un sistema propio combinando distintas herramientas gratuitas y de código libre, a cambio del tiempo que cuesta implementar y depurar y el precio a pagar por las máquinas y su mantenimiento energético mensual.

Como mantener múltiples sistemas puede ser una labor tediosa, es conveniente seguir una serie de buenas prácticas. Esencialmente, se puede resumir en que las tareas relacionadas con sistemas solo se tengan que hacer una única vez, independientemente cuánto de grande sea el sistema. Por eso, es conveniente tener todos los recursos centralizados y compartidos con los distintos nodos, así como disponer en cada nodo únicamente lo necesario para ejercer su función dentro del *cluster*.

5.2. Líneas futuras

El trabajo se puede continuar por diversas líneas.

En primer lugar, se puede ampliar el sistema con más nodos de cómputo que, incluso, podrían incluir alguna tarjeta gráfica para calcular modelos de inteligencia artificial o realizar grandes operaciones con un gran volumen de datos, así como ampliar el software proporcionado en el *cluster*.

Por otro lado, se puede utilizar Ansible [46] y preparar con él algún script para actualizar todos los sistemas a su última versión y automatizar la configuración de un nuevo nodo.

Por último, se podría incorporar Slurm Web [47], para visualizar de manera gráfica el estado y la disponibilidad de los recursos, los trabajos que se encuentran en cola y aquellos que están ejecutándose.

Capítulo 6

Summary and Conclusions

High-performance computing is the last order of the day. The trend in today's world is to keep improving and making the most of the potential of technology, in order to make science advance.

Two current trends confirm this: the development of quantum computers and the installation of more supercomputers around the world and, perhaps the most unnoticed, the developments of coronavirus vaccines, discoveries and continuous studies in astronomy and time modeling for its prediction.

Implementing a high-performance computing system is relatively straightforward. Although there are various business solutions, it is possible to create your own system combining different free and open source tools, in exchange for the time it takes to implement and debug it, and the price to pay for the machines and their monthly energy maintenance.

Since maintaining multiple systems can be a tedious task, a number of good practices should be followed. Essentially, it can be summed up as system-related tasks must be done only once, regardless of how big the system is. For this reason, it is convenient to have all the resources centralized and shared with the different nodes, as well as to have in each node only what is necessary to exercise its function within the cluster.

Capítulo 7

Presupuesto

En este capítulo se realiza un presupuesto del trabajo realizado. Todas las herramientas utilizadas son gratuitas, por lo que el presupuesto necesario para este proyecto está definido por el material necesario y el sueldo del administrador de sistemas durante 4 meses (tiempo de duración de la asignatura).

Para el presupuesto de los servidores, se tomó de referencia una factura de la compra realizada por el departamento en 2014. Cabe a destacar que el precio del mismo modelo de servidor en el año de la realización de este trabajo (2021) es considerablemente más barato.

Los programas desarrollados se encuentran alojados en github [48].

7.1. Presupuesto

Material	Cantidad	Presupuesto
Servidor Dell PowerEdge R815	3	7.021€
Portatil de trabajo	1	800€
Total:		21.863€

Tabla 7.1: Presupuesto del material

Personal	Meses	Presupuesto
Ing. Informático	4	1.200€
Total:		4.800€

Tabla 7.2: Presupuesto del personal

Para realizar este proyecto hicieron falta 4 meses y un precio final de 26.663€.

Apéndice A

Scripts

A.1. Script administración de usuarios

```
#!/bin/sh

show_help()
{
    # Display Help
    echo "Bash utility for cluster user management"
    echo
    echo "Syntax: cluster-user-management [a|m|d|u:]"
    echo "options:"
    echo " -h          Print this Help."
    echo " -a <user>    Create given user."
    echo " -m <user>    Modify given user."
    echo " -d <user>    Delete given user."
    echo " -q <quota>   Set user quota space. [20G default]"
    echo
    echo "Examples: "
    echo " - New user: -a test -q 25G"
    echo " - Modify user: -m test -q 500M"
    echo " - Delete user: -d test"
}

# $1 : username
# $2 : quota size (20G default)
add_user()
{
    if id -u "${1}" >/dev/null 2>&1; then
        echo "[!] Ya existe el usuario ${1}."
        return
    fi

    nscd --invalidate=passwd
    cpu useradd $1 -m -p"${1}bolido"
    setquota -u $1 $2 $2 0 0 /home
    mkdir "/home/${1}/.ssh"
    ssh-keygen -t rsa -f "/home/${1}/.ssh/id_rsa" -q -N ""
    cp "/home/${1}/.ssh/id_rsa.pub" "/home/${1}/.ssh/authorized_keys"
    chown -R $1 "/home/${1}/.ssh" && chgrp -R $1 "/home/${1}/.ssh"
    echo "The password for user ${1} is: ${1}bolido"
}
}
```

```

# $1 : username
delete_user()
{
    cpu userdel -r $1
    cpu groupdel $1
    nscd --invalidate=passwd
}

# $1 : username
# $2 : quota size (20G default)
modify_user()
{
    setquota -u $1 $2 $2 0 0 /home
}

# A POSIX variable
OPTIND=1          # Reset in case getopts has been used previously in the shell.
OPERATION=
QUOTA="20G"
USERNAME=

if [ $# -eq 0 ]; then
    show_help
    exit 0
fi

while getopts "h?a:d:m:q:" opt; do
    case "$opt" in
        h|\?)
            show_help
            exit 0
            ;;
        a) [ -n "$OPERATION" ] && show_help && exit || OPERATION="ADD_USER" && USERNAME=$OPTARG
            ;;
        d) [ -n "$OPERATION" ] && show_help && exit || OPERATION="DELETE_USER" && USERNAME=$OPTARG
            ;;
        m) [ -n "$OPERATION" ] && show_help && exit || OPERATION="MODIFY_USER" && USERNAME=$OPTARG
            ;;
        q) QUOTA=$OPTARG
            ;;
    esac
done

shift $((OPTIND-1))

[ "${1:-}" = "--" ] && shift

[ -z "$USERNAME" ] && show_help && exit

case $OPERATION in
    ADD_USER) add_user $USERNAME $QUOTA;;
    DELETE_USER) delete_user $USERNAME;;
    MODIFY_USER) modify_user $USERNAME $QUOTA;;
esac

```

A.2. Script generador CSV

```
#!/bin/bash

LOGS_DIR="/home/pladmin/pruebas/scripts/output/"
PARSED_LOGS_OUTPUT="/home/pladmin/pruebas/parsed-output/"
LOGS_FILES=$(ls $LOGS_DIR)
FILE_DELIMITER="EJECUCION DE"
file=$(echo "$LOGS_FILES" | head -n 1)
FILE_NAMES=$(cat $LOGS_DIR/$file | grep "$FILE_DELIMITER" | cut -d " " -f 3))

for file in $LOGS_FILES;
do
delimiters=$(cat ${LOGS_DIR}${file} | grep -n "EJECUCION DE" | cut -d ":" -f 1))
node=$(echo $file | cut -d "-" -f 1)
echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[0]}
    sed -n "$(( ${delimiters[0]}+1 )) , $(( ${delimiters[1]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[0]}
        echo '$(( ${delimiters[0]}+1 )) , $(( ${delimiters[1]}-1 ))p' "${LOGS_DIR}${file}"
echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[1]}
    sed -n "$(( ${delimiters[1]}+1 )) , $(( ${delimiters[2]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[1]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[2]}
    sed -n "$(( ${delimiters[2]}+1 )) , $(( ${delimiters[3]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[2]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[3]}
    sed -n "$(( ${delimiters[3]}+1 )) , $(( ${delimiters[4]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[3]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[4]}
    sed -n "$(( ${delimiters[4]}+1 )) , $(( ${delimiters[5]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[4]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[5]}
    sed -n "$(( ${delimiters[5]}+1 )) , $(( ${delimiters[6]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[5]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[6]}
    sed -n "$(( ${delimiters[6]}+1 )) , $(( ${delimiters[7]}-1 ))p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[6]}
        echo "EJECUCION DE ${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[7]}
    sed -n "$(( ${delimiters[7]}+1 )) , $ p" "${LOGS_DIR}${file}" >> ${PARSED_LOGS_OUTPUT}$node-${FILE_NAMES[7]}
done
```

A.3. Generador de graficas

```
import csv
import matplotlib.pyplot as plt
import numpy as np
from colorama import Fore, Style
import pandas as pd
import os

CSV_PATH="/Users/b3cl0s3r/Desktop/final-output/csv"
ITERATIONS=20

results = dict()
nodes = dict()

def generateSpeedUpPlot(processors_int, speedup, operation):
    plt.plot(processors_int, speedup["rayo"], marker='o', label="Rayo")
    plt.autoscale(True, axis='x')
    plt.xticks(processors_int)
    plt.plot(processors_int, speedup["centella"], marker='o', label="Centella")
    plt.title("Speedup "+operation.upper())
```

```

plt.xlabel('Procesadores', fontsize=14)
plt.ylabel('SpeedUp', fontsize=14)
plt.plot(processors_int, processors_int, '--', color="red", label="Ideal")
plt.legend()
plt.savefig("plots/speed-up-"+operation+".png")
plt.cla()
plt.clf()

def generateEfficiencyPlot(processors_int, efficiency, operation):
    plt.plot(processors_int, efficiency["rayo"], marker='o', label="Rayo")
    plt.autoscale(True, axis='x')
    plt.xticks(processors_int)
    plt.plot(processors_int, efficiency["centella"], marker='o', label="Centella")
    plt.title("Eficiencia "+operation.upper())
    plt.xlabel('Procesadores', fontsize=14)
    plt.ylabel('Eficiencia', fontsize=14)
    plt.axhline(y=1, color='r', linestyle='--', label="Ideal")
    plt.legend()
    plt.savefig("plots/efficiency-"+operation+".png")
    plt.cla()
    plt.clf()

def generateRedundancyPlot(processors_int, redundancy, operation):
    plt.plot(processors_int, redundancy["rayo"], marker='o', label="Rayo")
    plt.autoscale(True, axis='x')
    plt.xticks(processors_int)
    plt.plot(processors_int, redundancy["centella"], marker='o', label="Centella")
    plt.title("Redundancia "+operation.upper())
    plt.xlabel('Procesadores', fontsize=14)
    plt.ylabel('Redundancia', fontsize=14)
    plt.legend()
    plt.savefig("plots/redundancy-"+operation+".png")
    plt.cla()
    plt.clf()

def generateUtilizationPlot(processors_int, utilization, operation):
    plt.plot(processors_int, utilization["rayo"], marker='o', label="Rayo")
    plt.autoscale(True, axis='x')
    plt.xticks(processors_int)
    plt.plot(processors_int, utilization["centella"], marker='o', label="Centella")
    plt.title("Utilizacion "+operation.upper())
    plt.xlabel('Procesadores', fontsize=14)
    plt.ylabel('Utilizacion', fontsize=14)
    plt.legend()
    plt.savefig("plots/utilizacion-"+operation+".png")
    plt.cla()
    plt.clf()

def generateQualityPlot(processors_int, quality, operation):
    plt.plot(processors_int, quality["rayo"], marker='o', label="Rayo")
    plt.autoscale(True, axis='x')
    plt.xticks(processors_int)
    plt.plot(processors_int, quality["centella"], marker='o', label="Centella")
    plt.title("Calidad "+operation.upper())
    plt.xlabel('Procesadores', fontsize=14)
    plt.ylabel('Calidad del paralelismo', fontsize=14)
    plt.legend()
    plt.savefig("plots/calidad-"+operation+".png")

```

```

plt.cla()
plt.clf()

def generateBoxPlot(processors_int, time_array, operation):
    rayo = plt.boxplot(time_array["rayo"], positions=processors_int)
    centella = plt.boxplot(time_array["centella"], positions=processors_int)
    set_box_color(rayo, '#D7191C')
    set_box_color(centella, '#2C7BB6')
    plt.autoscale(True, axis='x')
    plt.xlabel('Procesadores', fontsize=14)
    plt.ylabel('Tiempo de ejecución', fontsize=14)
    plt.title("Boxplot "+operation.upper())
    plt.plot([], c='#D7191C', label='Rayo')
    plt.plot([], c='#2C7BB6', label='Centella')
    plt.legend()
    plt.savefig("plots/boxplot-"+operation+".png")
    plt.cla()
    plt.clf()

def set_box_color(bp, color):
    plt.setp(bp['boxes'], color=color)
    plt.setp(bp['whiskers'], color=color)
    plt.setp(bp['caps'], color=color)
    plt.setp(bp['medians'], color=color)
    plt.setp(bp['fliers'], color=color)
    plt.setp(bp['means'], color=color)

for filename in os.listdir(CSVPATH):
    operation=filename.split("-")[1].split(".")[0]

    if operation not in results:
        results[operation] = dict()

    node=filename.split("-")[0]
    nodes[node] = ""
    results[operation][node] = dict()

    with open('../csv/'+filename, newline='') as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='|')
        next(reader, None)
        csvcontent = list(reader)

        # Mapping data
        for row in csvcontent:
            processors=row[5]
            if processors not in results[operation][node]:
                results[operation][node][processors] = dict()
                results[operation][node][processors]["time"] = 0.0
                results[operation][node][processors]["mops"] = 0.0
                results[operation][node][processors]["time_array"] = []

            time=row[0]
            mops=row[1]
            results[operation][node][processors]["time"] += float(time)
            results[operation][node][processors]["mops"] += float(mops)
            results[operation][node][processors]["time_array"].append(float(time))

```

```

processors=list(results[operation][node].keys())
for processor in processors:
    results[operation][node][processor]["time"] /= ITERATIONS
    results[operation][node][processor]["mops"] /= ITERATIONS
    # Tiempo de ejecucion 1 nodo entre Tiempo de ejecucion N nodos
    results[operation][node][processor]["speedup"] = results[operation][node]["1"]["time"] / res
    results[operation][node][processor]["efficiency"] = results[operation][node][processor]["spe
    results[operation][node][processor]["redundancy"] = results[operation][node][processor]["mop
    results[operation][node][processor]["utilization"] = results[operation][node][processor]["mo
    results[operation][node][processor]["quality"] = results[operation][node][processor]["speedu

# Generate statistics
nodes=list(nodes.keys())
operations=list(results.keys())

for operation in operations:

    speedup=dict()
    efficiency=dict()
    redundancy=dict()
    utilization=dict()
    quality=dict()
    time_array=dict()

    for node in nodes:
        # Initialize arrays
        processors=list(results[operation][node].keys())
        processors_int=[ int(i) for i in list(results[operation][node].keys()) ]
        speedup[node]=[]
        efficiency[node]=[]
        redundancy[node]=[]
        utilization[node]=[]
        quality[node]=[]
        time_array[node]=[]

        for processor in processors:
            speedup[node].append(results[operation][node][processor]["speedup"])
            efficiency[node].append(results[operation][node][processor]["efficiency"])
            redundancy[node].append(results[operation][node][processor]["redundancy"])
            utilization[node].append(results[operation][node][processor]["utilization"])
            quality[node].append(results[operation][node][processor]["quality"])
            time_array[node].append(results[operation][node][processor]["time_array"])

# Generate plots
generateSpeedUpPlot(processors_int, speedup, operation)
generateEfficiencyPlot(processors_int, efficiency, operation)
generateRedundancyPlot(processors_int, redundancy, operation)
generateUtilizationPlot(processors_int, utilization, operation)
generateQualityPlot(processors_int, quality, operation)
generateBoxPlot(processors_int, time_array, operation)

```

Bibliografía

- [1] insideHPC, “What is high performance computing?” [Online]. Available: <https://insidehpc.com/hpc-basic-training/what-is-hpc/>
- [2] S. Technologies, “Business use cases for high-performance computing.” [Online]. Available: <https://www.sifytechnologies.com/blog/business-use-cases-high-performance-computing/>
- [3] EuroHPC, “Benefits of supercomputing.” [Online]. Available: <https://eurohpc-ju.europa.eu/discover-eurohpc-ju#ecl-inpage-14>
- [4] Iter, “Teide hpc.” [Online]. Available: <https://teidehpc.iter.es/>
- [5] Top500, “Teide hpc - ranking.” [Online]. Available: <https://www.top500.org/system/178254/>
- [6] B. S. Centre, “Marenostrum.” [Online]. Available: <https://www.bsc.es/marenostrum/marenostrum>
- [7] Top500, “Marenostrum - ranking.” [Online]. Available: <https://www.top500.org/system/179067/>
- [8] “Eurohpc.” [Online]. Available: <https://eurohpc-ju.europa.eu/discover-eurohpc-ju>
- [9] “Programación paralela.” [Online]. Available: http://ferestrepoqa.github.io/paradigmas-de-programacion/paralela/paralela_teoría/index.html
- [10] “Ordenadores paralelos.” [Online]. Available: https://es.wikipedia.org/wiki/Computacion_paralela#Clases_de_computadoras_paralelas
- [11] “Cluster beowulf.” [Online]. Available: https://es.wikipedia.org/wiki/Cluster_Beowulf
- [12] “Parallel paradigms.” [Online]. Available: <https://pdc-support.github.io/introduction-to-mpi/05-parallel-paradigms/index.html>
- [13] Dell, “Tecnología idrac.” [Online]. Available: <https://www.delltechnologies.com/es-es/solutions/openmanage/idrac.htm>
- [14] R. McLay, K. W. Schulz, W. L. Barth, and T. Minyard, “Best practices for the deployment and management of production hpc clusters,” in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–11.
- [15] SchedMD, “Slurm - quick start.” [Online]. Available: <https://slurm.schedmd.com/quickstart.html>

- [16] —, “Slurm github repository.” [Online]. Available: <https://github.com/SchedMD/slurm>
- [17] “Torque github repository.” [Online]. Available: <https://github.com/adaptivecomputing/torque>
- [18] “Kubernetes github repository.” [Online]. Available: <https://github.com/kubernetes/kubernetes>
- [19] “Munge.” [Online]. Available: <https://dun.github.io/munge/>
- [20] “Open ssh.” [Online]. Available: <https://www.openssh.com/>
- [21] “Openldap.” [Online]. Available: <https://www.openldap.org>
- [22] “Chrony.” [Online]. Available: <https://chrony.tuxfamily.org/>
- [23] S. Mishra, “Centos / rhel 7 : Chronyd v/s ntp service.” [Online]. Available: <https://medium.com/@codingmaths/centos-rhel-7-chronyd-v-s-ntp-service-5d65765fdc5f>
- [24] “Network file system.” [Online]. Available: https://es.wikipedia.org/wiki/Network_File_System
- [25] A. Libre, “Nfs client.” [Online]. Available: <https://www.alcancelibre.org/staticpages/index.php/configuracion-autofs>
- [26] PhoenixNap, “How to upgrade debian 8 jessie to debian linux 9 stretch.” [Online]. Available: <https://phoenixnap.com/kb/how-to-upgrade-debian-8-jessie-to-debian-9-stretch>
- [27] —, “How to upgrade debian 9 stretch to linux debian 10 buster.” [Online]. Available: <https://phoenixnap.com/kb/how-to-upgrade-debian-9-stretch-to-debian-10-buster>
- [28] “Ldap.” [Online]. Available: https://es.wikipedia.org/wiki/Protocolo_ligero_de_acceso_a_directorios
- [29] “Libnss-pamd.” [Online]. Available: <https://arthurdejong.org/nss-pam-ldap/>
- [30] “Change password utility.” [Online]. Available: <http://cpu.sourceforge.net/>
- [31] “Ldapvi tool.” [Online]. Available: <http://www.lichteblau.com/ldapvi/manual/>
- [32] “Herramienta de configuración de slurm.” [Online]. Available: <https://slurm.schedmd.com/configurator.html>
- [33] R. McLay, “Lmod instalation.” [Online]. Available: https://lmod.readthedocs.io/en/latest/030_installing.html
- [34] DevConnected, “How to setup openldap server on debian 10.” [Online]. Available: <https://devconnected.com/how-to-setup-openldap-server-on-debian-10/>
- [35] “nswitch.conf documentation.” [Online]. Available: <http://manpages.ubuntu.com/manpages/bionic/es/man5/nswitch.conf.5.html>
- [36] “Slurm pam.” [Online]. Available: https://slurm.schedmd.com/pam_slurm_adapt.html

- [37] Wikipedia, "Mpi." [Online]. Available: https://es.wikipedia.org/wiki/Interfaz_de_Paso_de_Mensajes
- [38] "Openblas." [Online]. Available: <https://www.openblas.net/>
- [39] "Gnu scientific library." [Online]. Available: <https://www.gnu.org/software/gsl/>
- [40] "Intel one api." [Online]. Available: <https://software.intel.com/content/www/us/en/develop/documentation/installation-guide-for-intel-oneapi-toolkits-linux/top/installation/install-using-package-managers/apt.html>
- [41] "Linux pam documentation." [Online]. Available: <http://www.linux-pam.org/>
- [42] TahaDz, "Pi calculation repository." [Online]. Available: <https://github.com/TahaDz/pi-calculation-openmpi>
- [43] U. d. V. Vicente Arnau Llombart, "Introducción al paralelismo y rendimiento." [Online]. Available: https://www.uv.es/varnau/OC_T4.pdf
- [44] "Pi calculation openmpi." [Online]. Available: <https://github.com/TahaDz/pi-calculation-openmpi>
- [45] NASA, "Nas benchmark." [Online]. Available: <https://www.nas.nasa.gov/software/npb.html>
- [46] "Ansible." [Online]. Available: <https://www.ansible.com/>
- [47] "Slurm web." [Online]. Available: <https://edf-hpc.github.io/slurm-web/introduction.html>
- [48] K. E. G. Peña, "Administración de nodos de cómputo de altas prestaciones." [Online]. Available: <https://github.com/ULL-ESIT-INF-TFG-2021/administracion-nodos-computo-altas-prestaciones-b3cl0s3r>