Escuela
Politécnica
Superior

# Despliegue de Clusters GPU de Bajo Coste

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Iván Rodríguez Ferrández

Tutor/es:
José García Rodríguez y Alberto García García

Universitat d'Alacant
Universidad de Alicante

Junio 2018

UNIVERSIDAD DE ALICANTE

Grado en ingeniería informática

Trabajo Fin de Grado

# Despliegue de Clusters GPU de Bajo Coste

*Autor*

Iván Rodríguez Ferrández

*Tutores*

José García Rodríguez y Alberto García García

Departamento de Tecnología Informática y Computación
(DTIC)
Alicante, Junio 5, 2018

« *The right man in the wrong place can make all the difference* »

**G-Man**

# Resumen

En la actualidad, la demanda de procesamiento se incrementa cada día. La sociedad genera una cantidad monstruosa de datos al día y la necesidad de poder procesar todos ellos nos lleva a crear nuevos centros de datos y de computación donde tratarlos. Como parte negativa, estos centros son costosos de crear y todavía más de mantener.

Por ello, el objetivo de este Trabajo Final de Grado es el desarrollo y la implementación de un clúster de bajo coste utilizando equipamientos disponibles en centros de enseñanza públicos. Para ello se ha desarrollado un conjunto de software para gestión, instalación y puesta en funcionamiento de un clúster de GPUs de bajo coste así como para su exposición para el uso por parte de un público general poco especializado. Para ello, en este proyecto se ha desarrollado y se implementa una API Full Rest para el lanzamiento de los trabajos dentro de un clúster a través de una interfaz web.

Como prueba de concepto, se ha desplegado el sistema propuesto en el laboratorio L14 de la Escuela Politécnica Superior de la Universidad de Alicante. De esta forma, se ha configura un laboratorio híbrido capaz de servir tanto para las clases cotidianas como para computación de altas prestaciones. Para finalizar, se establecieron una serie de directivas para comparar el rendimiento del clúster creado con otras soluciones comerciales que están disponibles en la propia Universidad de Alicante.

# Agradecimientos

Este proyecto, para mí, tiene una connotación especial, ya que me ha permitido conocer a gente estupenda a lo largo de su desarrollo. Además, sin proponerlo, el proyecto ha hecho que pueda terminar la carrera, ya que ha sido mi motivación constante para seguir adelante, permitiéndome, de este modo, poner a prueba mis conocimientos y capacidades informáticas. Hay tantas personas que agradecer por ese trabajo que no sé por dónde empezar. Yo creo que los primeros son mis padres y mi pareja que me han estado apoyando incondicionalmente en todo momento para que siga adelante. También a mi compañero y amigo Andrés Carpena Latour, quien ha estado sufriendo conmigo inmensas calamidades y quebraderos de cabeza en la realización de este trabajo final de grado.

No pueden faltar los técnicos de la EPSA: Mamen, Xavi, Juan Antonio, Adolfo, Zubi, Cande, Juan Jose, ... Sin su aguante a todos mis problemas y sin su enorme apoyo, este proyecto no hubiera funcionado.

También a los dos alumnos en prácticas: Samuel Aldegunde López y Miguel Rico Sánchez, por realizar el mantenimiento de todos los ordenadores del L14, consiguiendo así aumentar su rendimiento y vida útil.

Quisiera dar también una mención especial en los agradecimientos a mi tutor José García Rodríguez que nos encaminó y aporto todos los esfuerzos para apoyarnos en todo lo que pudo y a Albert García García por aguantarme con mis preguntas y ayudarme a continuar con mi proyecto de la mejor manera posible.

Gracias a todos por hacer que este proyecto sea posible y devolverme la ilusión a seguir mejorando mis conocimientos y aptitudes. Muchas gracias a todos.

# Índice general

# Índice de figuras

# Índice de Código

# Lista de Algoritmos

# Capítulo 1

# Introducción

En este capítulo se realizará un breve repaso histórico a la computación y los avances en los procesadores (Sección 1.1). Después se revisan una serie de trabajos relacionados con el campo de la supercomputación. Por su relación con este proyecto, se observan diferentes aspectos de la supercomputación (Sección 1.2), así como trabajos relacionados con el diseño y empleo de clústers CPU, GPU y CPU/GPU (Sección 1.3). Para finalizar se plantea la propuesta del trabajo (Sección 1.4) y se define la estructura del documento (Sección 1.5).

## 1.1. Historia de la computación

Con el comienzo de la computación moderna, el reto de crear un computador cada vez más rápido se extendió por la comunidad tecnológica de la época en la década de los 60s. Así, tal fue el impacto, que Gordon Earl Moore proponía en su publicación más relevante, la famosa ley de Moore [1], que establece que el número de transistores por unidad de superficie en un procesador se duplica cada año. Con este postulado, Gordon E. Moore junto a Robert Norton Noyce, crearon la empresa Intel. Sobre el año 1970, a esta empresa poco conocida, se le propuso realizar el primer microprocesador de la historia y el 15 de diciembre de 1971 lo lanzaron al mercado, figura 1.1. Este microprocesador sentó las bases sobre las que se aplicaría la ley de Moore.

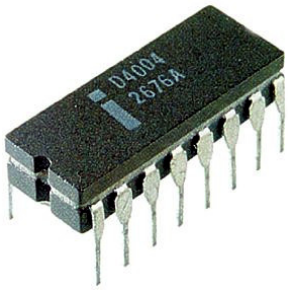



Figura 1.1: Microprocesador Intel 4004

En 1975, Moore actualizó su ley rebajando las expectativas de duplicación del número de transistores por unidad de superficie a 24 meses. Esta modificación se aplicó debido a las dificultades surgidas en la reducción de tamaño de los transistores, que quedaron reflejadas en la salida del procesador de Intel 8080. Y se ha estado cumpliendo desde entonces hasta 2004 con la salida del procesador Pentium 4.

Por otra parte, en el año 2000 una pequeña empresa conocida como Nvidia lanzó al mercado una tarjeta gráfica que disponía de una Unidad de Procesamiento Gráfico (más conocida como GPU en inglés), figura 1.2. Dicho avance dio pie a que el procesador pudiera centrarse en las tareas del cálculo más puro, relegando así a este segundo "procesador" las tareas de mostrar gráficos por pantalla. Al disponer de un hardware específico para el procesamiento de gráficos, las interfaces de usuario UI, comenzaron a hacerse más y más ricas en contenido exigiendo cada vez más rendimiento a estas nuevas GPU's.

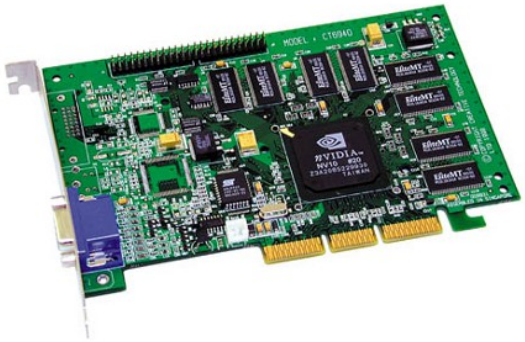


Figura 1.2: GeForce 256 Primera GPU

Sobre los años 2001 a 2003 se observó que se podía aprovechar la potencia de estos nuevos procesadores gráficos en tareas de cálculo en coma flotante y cálculo de sistemas de matrices con gran eficiencia. En 2001 la comunidad

científica consiguió utilizar algunas de las GPU para el cálculo de multiplicación de matrices, una tarea paralela que los procesadores de la época tenían dificultades en procesar. En 2005 se consiguió implementar la factorización LU aplicando esta metodología. Con estos pioneros se empezó a acuñar el término *General-purpose computing on graphics processing units* o GPGPU, el cual se refería a la utilización de las GPU como un procesador de propósito general para tareas paralelas, y no solo para procesamiento de gráficos.

Sin embargo, no se disponía de una API de alto nivel para poder realizar estas tareas, por lo que para ejecutar estos primeros algoritmos paralelos se tenia que "engañar" a las tarjetas gráficas pasando los datos como una imagen y utilizando los *shaders* para pasar las operaciones que tenían que realizarse en cada celda de datos. Esto provocaba que solo el personal que tuviera alto conocimiento de la arquitectura y de su programación a bajo nivel, fuera capaz de desarrollar software.

En 2004, con la salida del procesador Intel Pentium 4, se constató que no se podía seguir aumentando la velocidad de reloj del procesador. Esto se debía a que el aumento de la frecuencia de reloj provocaba que la temperatura de los microprocesadores aumentara de forma exponencial. Por lo que, en 2005 Intel presentó los primeros procesadores comerciales multinúcleo, dando paso así a los sistemas de procesamiento paralelos en la CPU.

En 2007, Nvidia puso a disposición de la comunidad CUDA, acrónimo de *Compute Unified Device Architecture*. Este hecho, supuso un hito que popularizó el uso de las GPU como procesadores de propósito general (GPGPU), permitiendo que estas tarjetas gráficas pudieran ser programadas utilizando el lenguaje C++. Al ser un lenguaje extensamente conocido por la comunidad, y gracias a la extensa documentación y ayuda de la comunidad de desarrolladores, permitió que se produjeran más programas que explotan el paralelismo que ofrecen las GPU's.

Con estos avances en la computación paralela, se abre el camino a la creación de arquitecturas y estructuras de procesamiento paralelo distribuidas, los clúster. Estas estructuras aúnan muchos recursos hardware en un entorno paralelo distribuido que permite explotar al máximo las capacidades computacionales de los procesadores y así lograr que algoritmos que antes se consideraban como intratables a nivel computacional puedan ser computados en un tiempo asumible gracias a la división de trabajo.

## 1.2. Motivación y Contexto

Desde hace unas décadas, los sistemas de altas prestaciones, HPC y clúster están presentes en nuestra sociedad. Gracias al avance de este tipo de

infraestructuras, ha sido posible resolver problemas de gran coste computacional como, por ejemplo, la predicción de la meteorología con una precisión suficiente para que sus resultados sean relevantes, así como su uso para la simulación de fármacos, procesos químicos, predicción de catástrofes, etc... Por estos motivos, el crecimiento en tamaño, complejidad y número de estos superordenadores se hace patente. Cada vez, más países disponen de uno o varios de estos clúster para poder satisfacer la demanda de procesamiento, convirtiéndose en un recurso indispensable para que la comunidad científica pueda seguir avanzando en diversos campos. Aunque estos equipos proliferan, no suelen estar al alcance de estudiantes o para pequeños proyectos. Esto es debido a que el coste de estos sistemas es muy alto, ya que, para su construcción, se necesita una gran inversión económica destinada al equipo informático y a los sistemas auxiliares. Además, hay que sumar el coste del mantenimiento de toda su infraestructura, así como el gasto por consumo eléctrico que suele ser elevado. Por todo ello este tipo de instalaciones siguen siendo escasas y de difícil acceso.

En el caso de la Universidad de Alicante, el Instituto Universitario de Investigación Informática IUII, dispone de un clúster de cálculo llamado Euler orientado a proyectos de investigación en todas las disciplinas que necesiten potencia de cálculo para sus proyectos. Este servicio queda fuera del alcance de los alumnos, ya que su uso se limita a proyectos de investigación. Sin embargo, en la Escuela Politécnica Superior de Alicante EPSA, se dispone de un laboratorio de altas prestaciones equipado con tarjetas gráficas GTX 480, figura 1.3, cedidas por NVIDIA. Gracias a la relación de NVIDIA con la UA, siendo esta GPU Educaction Center y GPU Research Center, avalados por NVIDIA.

Figura 1.3: GTX 480 Cedidas por NVIDIA

Gracias a la donación recibida, este laboratorio dispone de una gran can-

tidad de recursos computacionales, y puede explotarse como un recurso en bruto de gran interés. Por ello se planteó la creación de un sistema para aprovechar el laboratorio en las horas que no tuviera un uso docente. Con la guía de los profesores José García (director del GPU Research Center) y Juan Antonio Gil (director de tecnologías de la EPS), se creó el proyecto de *Cluster as a service*, que desarrollé durante el curso académico 2016-2017 en prácticas en empresa I y II junto a mi compañero Andrés Carpena. En el proyecto se implementó un sistema para el aprovechamiento de este recurso, para lo que se diseñó y se creó un clúster que fuera compatible con el uso normal del aula.

La motivación personal para la realización de este proyecto viene dada por los años en los que he trabajado en montaje de redes de ordenadores, así como el gran apoyo constante de mi padre (Juan Rodríguez López- Ingeniero en Informática) que me facilitó desde bien pequeño el introducirme en el mundo de los ordenadores. Por todo ello, este proyecto ha servido como motivación personal para mejorar mis capacidades como futuro ingeniero y además supone un orgullo que mi proyecto no quede solo en la teoría, sino que, además, he podido ver como se ha materializado y se ofrece actualmente como recurso en la Escuela Politécnica Superior de la Universidad de Alicante (aunque todavía en pruebas por razones ajenas a mi trabajo).

El valor de este Proyecto de final de Grado no se limita a superar una asignatura obligatoria del grado de Informática, sino que se ha materializado en un recurso de utilidad a la comunidad, pudiendo conseguir que sea utilizado por los alumnos de la UA que necesiten recursos de altas prestaciones o recursos remotos para las aplicaciones. Además, gracias a este proyecto, he logrado abrir la puerta a los supercomputadores a un público más amplio.

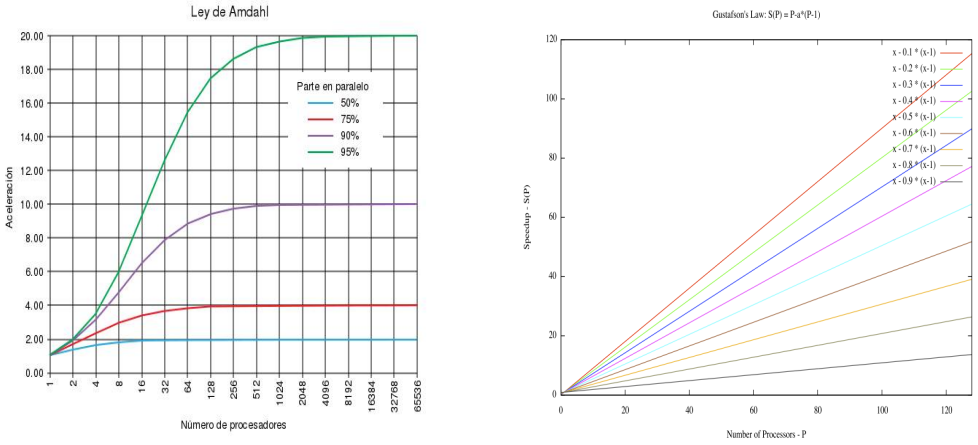
## 1.3. Estado del Arte

En esta sección se repasan, como paso previo a la definición de una propuesta concreta, trabajos y conceptos relacionados directamente con el problema planteado.

### 1.3.1. Paralelismo

En los entornos de computación de altas prestaciones, el hardware que incorporan los equipos no suele diferir mucho del que un equipo domestico puede utilizar. La clave de estos sistemas es la capacidad de aunar muchos de estos equipos básicos para crear un entorno paralelo. Esto hace que la potencia de un clúster no dependa únicamente de lo rápido que funcione uno

de los equipos de forma individual, sino del rendimiento obtenido con su uso conjunto en paralelo. Esta velocidad o capacidad de cómputo, conseguida gracias la paralelización, viene definida por la ley de Amdahl[2], figura 1.4a, la cual establece que la mejora en la velocidad de ejecución de un programa como resultado de la paralelización está limitada por la porción del programa que no puede ser paralelizada. Además, la ley de Gustafson[3], figura 1.4b, complementa la ley de Amdahl en el sentido de que esta ley establece la mejora en relación con el número de procesadores de que dispone el sistema paralelo.




(a) Ley de Amdahl (b) ley de Gustafson

Figura 1.4: Representación de la ley de Amdahl (a) y la ley de Gustafson (b)

Con esta base teórica se puede concluir que la línea a seguir para mejorar el rendimiento de un programa es la paralelización, lo que permitirá aumentar su rendimiento en comparación a su versión secuencial. Con la introducción de la paralelización aparecieron una nueva serie de problemas, uno de ellos es el riesgo de dependencia de datos. Debido a que múltiples procesadores pueden acceder a un mismo dato, por lo que se pueden generar problemas, dado el riesgo de escritura antes de lectura (Riesgo WAR) o de escritura en la misma variable (Riesgo WAW). Por ello, esta dependencia de datos es un factor fundamental para el desarrollo de programas y algoritmos paralelos. Es muy importante definir que partes de un programa son independientes para que su paralelización sea más sencilla. Por ello Bernstein [4] creó una serie de condiciones que describen cuando dos segmentos de código son independientes y por tanto se pueden ejecutar en paralelo.

Dentro de este nuevo paradigma de programación existen diferentes niveles de paralelización. En este proyecto, la paralelización que emplea el clúster

pertenece a la tipología de nivel de tareas. Así, cada proceso se encarga de calcular una parte del problema que se intenta resolver.

### 1.3.2. Sistemas Multinúcleo

*Desde 2003, la industria de los semiconductores se ha posicionado en dos trayectorias principales para el diseño de microprocesadores. La trayectoria de los multinúcleos busca mantener la velocidad de ejecución de los programas secuenciales, mientras que se desarrollan para trabajar en múltiples cores. Los multicores comenzaron como dual core processors, con el número de núcleos doblándose cada generación de semiconductores. [...]. En contraste, la aproximación de many-core se centra más en la ejecución de aplicaciones paralelas. Los many-core comenzaron como un gran número de procesadores mucho más pequeños, [...]donde cada uno de ellos dispone de un altísimo número de hilos de ejecución*[5].

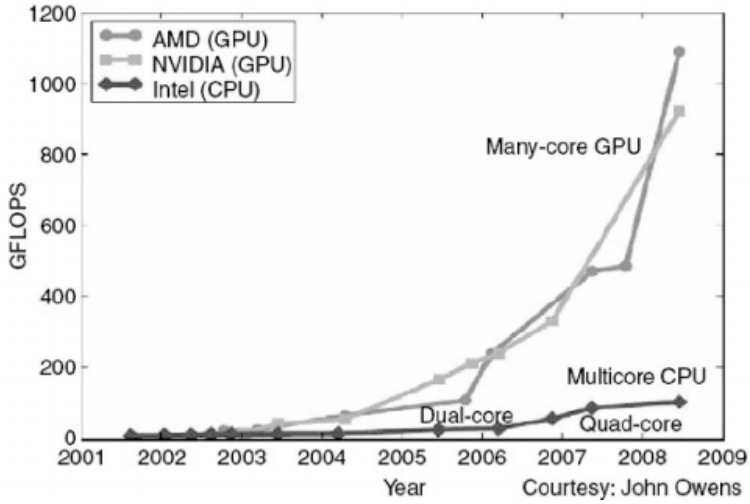



Figura 1.5: Aproximacion many-core vs multicore

Los sistemas multinúcleo fueron la primera aproximación al paralelismo a nivel de tarea dentro de un mismo componente de computación. Esta solución viene condicionada por la imposibilidad de incrementar la velocidad de los procesadores mono-núcleo. Esto se hizo presente cuando, a finales de los 90, este aumento en la velocidad provocaba un aumento exponencial de la temperatura. Por lo que una refrigeración por aire resultaba imposible. Por ello la industria decidió apostar por los procesadores multinúcleo[6]. Esto fue también motivado por la vigencia de la ley de Moore que llevaba años prediciendo con buen acierto el movimiento de la industria de los semiconductores.

Esta aproximación al multinúcleo tenía una serie de ventajas importantes. Al disponer de varios core en un mismo *DIE* se evitaban muchos errores por degradado de la señal por el hecho de que los core están muy próximos entre si. Esto a su vez se traducía en que se podía utilizar una señal de más calidad que podía transportar más cantidad de datos. Todo ello, provocó una revolución en la computación ya que se podía trabajar con varios procesos de forma simultánea. Fue tal la repercusión que desde el comienzo de la producción de los procesadores multinúcleo las empresas han ido añadiendo más y más núcleos dentro de un mismo *DIE*, tanto es así que pasamos de los procesadores de dos núcleos a los 8 o 16, en procesadores domésticos y hasta los 72 núcleos en los servidores de la mano de los procesadores Xeon Phi de Intel.

Esta tendencia a desarrollar sistemas multinúcleo por parte de las empresas desarrolladoras de hardware, provoca que la comunidad científica se centre en formas de optimizar y/o mejorar las operaciones en sistemas con más de un núcleo. Un ejemplo de ello es el trabajo de Chenchen Fu et al., [7] orientado a optimizar el consumo eléctrico de la memoria compartida en los sistemas multinúcleo. Para ello, analizan la carga del procesador para comprobar si el procesador se encuentra en estado *idle* para proceder a poner a la memoria en un estado de *sleep*. Según sus resultados, incluyendo el coste energético extra de pasar de estados la memoria, consiguen una mejora de entre un 7,25 % a un 11,71 %. Esto hace que sistemas multinúcleo en dispositivos móviles puedan ahorrar energía cuando no están en uso.

La necesidad de adoptar los sistemas multinúcleo es tan necesaria para la evolución de la computación que muchos investigadores dedican mucho esfuerzo a convertir algoritmos puramente secuenciales a algoritmos paralelos. Es por ello que multitud de trabajos, como el de Michael D. McCool[8], estudien los sistemas *many-cores* y como adaptar los algoritmos puramente secuenciales a estos procesadores y además como se podrían exportar estos mismos algoritmos a futuros procesadores multinúcleo. En est mismo trabajo se presentan diferentes arquitecturas *many-cores*; así como modelos de procesamiento para tareas paralelas.

### 1.3.3. Clusters y Supercomputadores

Cuando la demanda de capacidad de cómputo comienza a ser cada vez más alta, se hace patente que las maquinas convencionales no son lo suficientemente potentes como para cubrir dicha demanda. Como se ha comentado con anterioridad, el paralelismo parece ser la solución cuando un problema es altamente complejo y pesado. Con el objeto de aportar soluciones nacieron también los supercomputadores y los clústers.

La idea principal de estos sistemas es la disponibilidad de una mayor cantidad de procesadores lo que se traduce en un incremento en FLOPS con respecto a un computador convencional, así como mayor disponibilidad y redundancia dado que el sistema está separado en diferentes maquinas, por lo que la caída o el fallo de un ordenador no inhabilita el sistema, solo ralentiza el sistema o ciertas operaciones no pueden ser ejecutadas. Esto cobra más sentido durante los años 70 u 80 cuando los ordenadores apenas habían comenzado sus andanzas y eran bastante propensos a la avería.

Los clústers son muy importantes para el desarrollo del conocimiento, siendo utilizados para múltiples tareas que serían intratables en en ordenadores convencionales. Algunas de estas tareas con alto coste computacional que se benefician del paralelismo son, por ejemplo, la predicción de la meteorología, el análisis molecular, la investigación del clima o la Mecanica Quantica.

Otro interesante campo de aplicación es el análisis del terreno. Utilizando la potencia de los HPC para procesar gran cantidad de información podemos entender como el terreno cambia y se comporta, pudiendo conocer mejor nuestro entorno, así como poder explotarlo de la forma más eficiente y ecológica. Un ejemplo de este este uso se puede encontrar en el trabajo de Bryan C. Pijanowski [9], el cual utiliza sistemas de inteligencia artificial unidos con GIS tradicionales para extraer un mayor conocimiento del entorno. Todo este procesamiento se lleva a cabo en supercomputadores, ya que el objetivo es tener unas resoluciones cercanas a los 30 metros en mapas de nivel de continentes, esto representa el proceso de carga computacional que solo puede ser realizado en los clústers y supercomputadores. Este hecho representa un hito, ya que permite disponer de mapas de gran tamaño y de alta resolución, lo que abre la posibilidad de analizar tanto comportamientos del terreno, como datos socioeconómicos desde la perspectiva de un continente, país o ciudad. Para ello se utilizan GIS comerciales para extraer datos concretos del mapa y después, utilizando los datos socioeconómicos disponibles, crear un modelo de más escala utilizando redes neuronales para la extracción de características.

La medicina de uno de los campos en el que el empleo de la supercomputación puede aportar mayores beneficios. Aunque siempre ha sido un campo muy interesante para la computación, ha sido difícil poder procesar la gran cantidad de datos que se generan, así como la falta de estructura de dichos datos. Por ello, el uso de los HPC ha supuesto un gran salto para este campo. Un ejemplo de uso puede encontrarse en el trabajo de Björn Gmeiner [10] en el que se describe como optimizar un algoritmo de creación de modelos 3D para su ejecución en supercomputadores. Dicho algoritmo está orientado a transformar las imágenes de resonancia magnética a un modelo aproxima-

do en 3D de la cabeza humana. En el artículo también se expone que el algoritmo es altamente escalable y se puede ejecutar en diferentes superordenadores. En este caso las pruebas se realizan en el clúster de la universidad de Erlangen-Nuremberg, concretamente el BlueGene/P que dispone de unos 300,000 núcleos de procesamiento.

Estos ejemplos muestran como el uso de los supercomputadores y clúster es ampliamente empleado para múltiples disciplinas. Gracias estos sistemas podemos obtener respuestas a problemas que antes era imposible calcular en un solo ordenador.

### 1.3.4. Clústers Híbridos

En secciones anteriores se ha mostrado como los clúster y superordenadores explotan el paralelismo a nivel de CPU. En cambio, este esquema clásico está siendo sustituido por el esquema del clúster Híbrido. Este tipo de clúster utiliza no solo el paralelismo CPU, si no que aprovecha el uso de procesadores que explotan el paralelismo a nivel interno, más concretamente nos referimos a las GPU's. Las GPU's originalmente se ha utilizado para el entretenimiento de los usuarios, pero con los años se han hecho un hueco en los centros de datos y centros de investigación. Esto se debe a que estos procesadores están diseñados y optimizados para realizar en paralelo muchos cálculos simples y con el tiempo también operaciones en coma flotante. Este tipo de cálculos son muy útiles para ciertas aplicaciones científicas, procesamiento de imágenes, análisis de nubes de puntos, etc,. Por esta serie de características los centros de supercomputación cada vez se están equipando con un mayor número de estas GPU's.

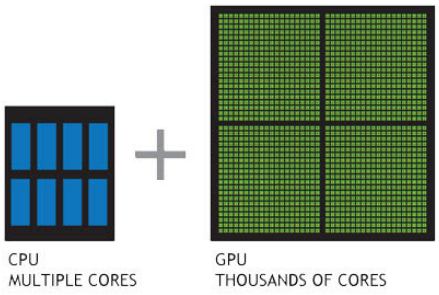



Figura 1.6: Representación de la unión del paralelismo en CPU y GPU

El problema principal que presenta el uso de estos dispositivos es que en si son arquitecturas paralelas, por lo que abstraerlas en un sistema paralelo dentro de un clúster representa un problema complejo. Por ello es importante

el proceso de sincronización, paso de parámetros, y paso de memoria entre las diferentes GPU's para poder aprovechar todo el potencial de cálculo de estos procesadores. Es por ello que los diseños de las redes de interconexión entre cada uno de los componentes del clúster tienen que ser de un alto ancho de banda y baja latencia.

Un ejemplo de uso muy común del clúster híbrido es la resolución de sistemas lineales. En el trabajo de Lilia Ziane Khodja [11] utilizan un clúster híbrido para resolver sistemas lineales dispersos. Para ello utilizan CUDA con la unión de MPI para paralelizar el problema en el clúster. Para la resolución de este problema utilizaban el método GMRES para la ejecución iterativa del método, así como para minimizar la comunicación. El clúster está compuesto por seis maquinas conectadas por una red InfiniBand de 20 GB/s. Cada nodo está compuesto de procesadores de cuatro núcleos y una GPU Tesla C1060. Gracias al uso de las GPU's se obtiene una fracción de mejora de aproximadamente 6 con respecto a CPU.

Como se ha comentado, dentro de los clústers híbridos la comunicación y paso de mensajes es importante. Por ello un buen algoritmo de comunicación que minimice la comunicación entre nodos es algo esencial. En el caso anterior se ha comentado el método GMRES, es un método muy utilizado en la computación de altas prestaciones, por ello en el artículo de Ichitaro Yamazaki[12] propone un método para la mejora en los tiempos de comunicación de los nodos, ya que argumenta que estamos llegando al punto donde los ordenadores modernos tardan menos en realizar operaciones aritméticas que en el paso de mensajes. Es por ello que propone dos soluciones: una primera que es una mejora al GMRES a la que denomina CA-GMRES, que utiliza métodos de Krylov para la reducción de la comunicación, con ello puede llegar a conseguir una fracción de mejora de 2,5. Además, presenta un mejora al método anterior utilizando métodos de Krylov con un sistema de precondiciones y se consigue una fracción de mejora de hasta un 7,4. Este trabajo refleja cómo la comunicación entre nodos es un tema importante y como estas mejoras puede hacer variar en gran medida el rendimiento de un clúster.

### 1.3.5. RCUDA

A medida que el clúster híbrido ha ido suscitando interés dentro de la comunidad científica ,gracias a la gran potencia de cálculo que ofrecen las tarjetas gráficas, cada vez aparecen más y más herramientas software para poder aprovechar mejor el potencial de estos procesadores gráficos masivamente paralelos. Una herramienta recientemente creada para su uso en clústers híbridos es RCUDA. RCUDA es una herramienta creada por inves-

tigadores de la Universidad Politécnica de Valencia (UPV) que permite la gestión de estos procesadores gráficos mediante un nuevo enfoque.

La forma clásica de trabajar con tarjetas gráficas dentro de un clúster es utilizar una librería para paralelización que sea compatible con CUDA como por ejemplo MPI o OpenMPI. El problema de esto es el hecho de que el código y el algoritmo a procesar tienen que poder expresarse utilizando estas librerías, lo que supone que la persona que desarrolle para estos sistemas tiene que conocer muy bien el uso de las librerías. Pero gracias a RCUDA o *Remote CUDA*, ya no es necesario el uso de esas librerías para lograr el paralelismo.

Desarrollado por el Grupo de Arquitecturas Paralelas de la UPV, RCUDA es un proyecto desarrollado para clústers que permite la virtualización de tarjetas gráficas dentro del mismo. La idea de la herramienta es evitar la necesidad de que un programador que conozca CUDA tenga que utilizar librerías paralelas, en vez de ello el programador se centra en poder ejecutar su código CUDA en múltiples tarjetas gráficas. RCUDA se encarga de virtualizar las tarjetas gráficas que se encuentran en un clúster para que se muestren como tarjetas conectadas directamente a la maquina donde se ejecuta el software en cuestión. En este marco, RCUDA es el encargado de la comunicación con estas tarjetas que están alojadas en los nodos del clúster. También facilita que un ordenador que no disponga de una tarjeta con CUDA y esté conectado en la red del clúster, puede solicitar tarjetas gráficas a los nodos de este para ejecutar su código CUDA en local, dejando la parte de comunicación directamente a RCUDA. Dentro de este marco se han realizado algunas investigaciones y artículos. Uno de los primeros proyectos que se realizaron, es el artículo de Javier Prades, Carlos Reaño y Federico Silla [13] en el que se describe como utilizar RCUDA para máquinas virtuales para poder dotar al sistema de acceso a tarjetas gráficas. El entorno que utilizaban era un clúster de que utilizaba Xen para la virtualización y una red InfiniBandpara la interconexión de estos, la idea es verificar como RCUDA se utiliza para acelerar aplicaciones científicas dentro de las máquinas virtuales. La virtualización de GPU's con RCUDA dentro de los clústeres, tiene un gran potencial para aplicaciones científicas o de aprendizaje profundo ya que se pueden crear nodos virtuales con la cantidad de GPU's necesarias para un proyecto.

## 1.4. Propuesta

Una vez analizado el estado actual de los sistemas de Computación de Altas Prestaciones (High Performance Computing o HPC en inglés), nuestra propuesta es crear un clúster híbrido que utilice el laboratorio L14 de la Escuela Politécnica Superior de la Universidad de Alicante (EPSA) localizado en el edificio EPS 1. La idea es aprovechar el potencial que ofrecen las tarjetas gráficas que fueron cedidas a la universidad por NVIDIA y los equipos que hay disponibles durante las horas sin docencia, fines de semana y vacaciones. La propuesta se divide en tres objetivos principales. El primero, la investigación y puesta en marcha de un clúster híbrido que pueda funcionar como computador común para las clases ordinarias y pueda cambiar a clúster cuando se desea. El segundo, garantizar la escalabilidad del despliegue y la portabilidad a otros laboratorios. Y finalmente, el tercero consiste en la creación y puesta en funcionamiento de una interfaz web que permita a los alumnos poder utilizar los recursos del clúster sin necesidad de conocer su funcionamiento, que sea completamente transparente.

## 1.5. Estructura del documento

Este documento estructura de la siguiente forma: Durante este pasado Capítulo 1 se revisa el estado del arte actual en lo referente a la computación de altas prestaciones, también se presenta la motivación principal para la realización de este proyecto así como las metas propuestas. En el Capítulo 2 se presentan los componentes hardware y software que han sido utilizados para el desarrollo de este proyecto, tanto en los sistemas utilizados en el proyecto como los equipos auxiliares utilizados para validar los resultados. En el Capítulo 3 se describe el funcionamiento e implementación de la API para el uso del clúster. En el Capítulo 4 se describen tanto el sistema creado para este proyecto así como todo el software implementado para el funcionamiento del sistema. Seguidamente, en el Capítulo 5, se presenta la experimentación realizada para validar el sistema creado, utilizando una batería de pruebas para conocer los beneficios y las carencias del sistema. Por último, en el Capítulo 6, se presentan las conclusiones obtenidas durante la realización de este proyecto así como las líneas futuras del mismo.

# Capítulo 2

# Materiales y Métodos

En este capítulo se describen los diferentes sistemas, tanto hardware como software, así como la metodología aplicada para comparar diferentes HPCs con el sistema propuesto. Para ello, se han seleccionado dos equipos: Euler, un clúster del Instituto Universitario de Investigación Informática (IUII), y el servidor de cómputo GPU del Departamento de Tecnología Informática y Computación (DTIC) y del 3DPLab, Clarke.

## 2.1. Planificación

El proyecto en cuestión ha tomado una gran cantidad de tiempo ya que en realidad comenzó a gestarse tres años atrás, desde el primer año se planteó de manera voluntario la realización de este proyecto, el segundo año, como prácticas en empresa realizadas en la EPSA y el DTIC y el tercer año como proyecto final de grado.

El proyecto se ha divido en cuatro grandes partes. La primera parte que consistió en una toma de contacto con el equipo de la EPSA y decisión del sistema operativo. La segunda parte estuvo más orientada a la instalación y puesta en marcha del clúster. La tercera fue dedicada en la creación de la API para enviar los trabajos y los *scripts* necesarios para ello. Por último, una cuarta parte dedicada el *benchkmarking* del clúster y su comparación con otros equipos disponibles. Todo el proyecto queda reflejado en el siguiente esquema de plantificación temporal y su representación en un diagrama de Gantt (ver Figuras 2.1 y 2.2).

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|---|
| 1 | | **Investigación del sistema operativo** | **36 días** | **lun 02/11/15** | **lun 21/12/15** |
| 5 | | **Analisis y extracion de de datos del L14** | **5 días** | **lun 01/02/16** | **vie 05/02/16** |
| 9 | | **Instalación del master** | **80 días** | **lun 08/02/16** | **vie 27/05/16** |
| 13 | | **Pruebas de instalación con el modelo** | **20 días** | **lun 30/05/16** | **vie 24/06/16** |
| 16 | | Instalación de los nodos | 11 días | lun 12/09/16 | lun 26/09/16 |
| 17 | | Testeo del sistema | 43 días | mar 27/09/16 | jue 24/11/16 |
| 18 | | **Creacion de la api** | **112 días** | **vie 25/11/16** | **lun 01/05/17** |
| 22 | | Benkmarking | 7 días | mar 02/05/17 | mié 10/05/17 |
| 23 | | Integracion con los servicios de la ua | 9 días | jue 11/05/17 | mar 23/05/17 |
| 24 | | Testeo del sistema web | 5 días | mié 24/05/17 | mar 30/05/17 |
| 25 | | Benchmarking GPU | 49 días | vie 19/01/18 | mié 28/03/18 |
| 26 | | Procesado de datos y generación de estadísticas | 25 días | lun 09/04/18 | vie 11/05/18 |

Figura 2.1: Esquema de fases del proyecto.

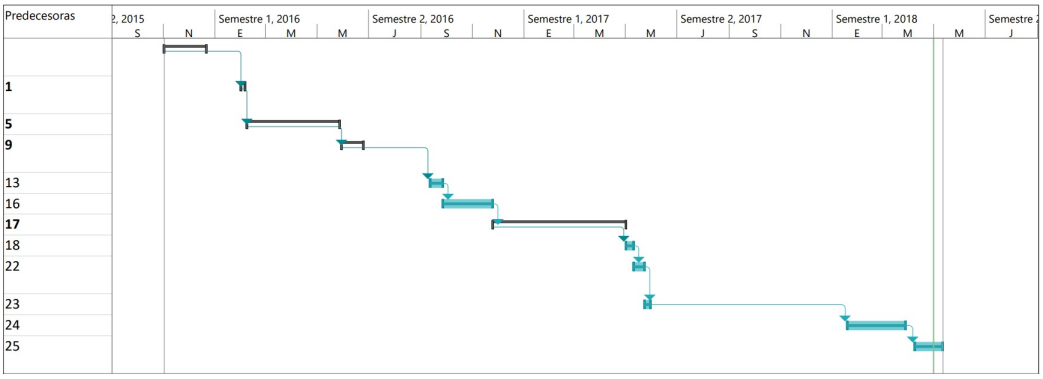



Figura 2.2: Diagrama de Gantt.

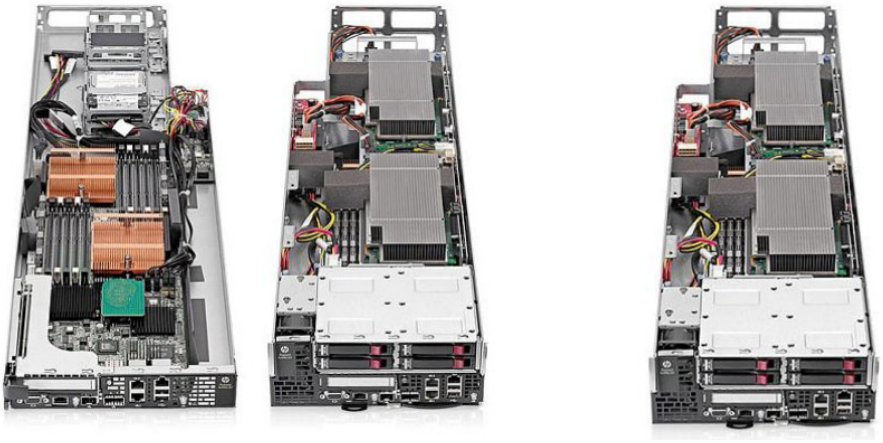
## 2.2. Hardware

Dentro de esta sección se describirán todos los componentes hardware que han sido utilizados durante el desarrollo de este proyecto. En particular se especificaran los componentes hardware de los dos superordenadores que se utilizarán para comparar los resultados. De la misma manera se proporcionará la descripción de los componentes del proyecto en cuestión.

### 2.2.1. Euler

Euler es un clúster que pertenece al Instituto Universitario de Investigación en Informática de la UA (IUII). Es un clúster adquirido por el grupo de computación de altas prestaciones y paralelismo de la Universidad de Alicante, está cofinanciado por el proyecto construcción y optimización automáticas

de bibliotecas paralelas de computación científica-UA, del Ministerio de Ciencia e Innovación. En cuanto al hardware, está conformado por 26 nodos de cálculo, los cuales se dividen en dos grupos: los nodos de cálculo CPU y un nodo de cálculo GPU tal y como se muestra en la Figura 2.3.



(a) (b)

Figura 2.3: a Nodo de cálculo CPU de Euler y b Nodo de cálculo GPU.

#### 2.2.1.1. Nodos de Cálculo CPU

Los nodos de cálculo son servidores HP Proliant SL 390 Generación 7. Equipados con dos procesadores Xeon X5660, los cuales disponen de las siguientes características:

- 6 cores a 2,80 Ghz, Max Turbo 3,2 Ghz
- 12 MB cache Level 2
- 64 bit, SSE 4.2
- 32 nm , Max TPD 95 w
- Intel QPI Speed 6,4 GT/s, 2 Links
- Soporte Hyperthreading (desactivado)

Además de las características CPU, el servidor presenta estas especificaciones hardware:

- Front Bus 800-1333 MHz
- Memoria 48 GB DDR3-1333
- 1 disco SFF SATA 7,2 k, 500GB
- Dual port 1GbE NIC
- 10GbE SFP+ (via ConnectX2)

- QDR IB QSFP (via ConnectX2)

Estos nodos están centrados en el cálculo, usando librerías paralelas como MPICH o OpenMPI para poder utilizar el potencial paralelo del clúster. Euler presenta 26 de estos nodos con los que en pleno funcionamiento ofrece un total de 312 núcleos de procesamiento CPU con un total de 1.21 TiB de memoria RAM.

#### 2.2.1.2. Nodo de Cálculo GPU

El nodo de cálculo GPU es un servidor HP Proliant SL 390 Generación 7. Equipado con dos procesadores Xeon X5660 los cuales mantienen las especificaciones comentadas anteriormente al igual que las características hardware también comentadas.

Lo que diferencia este nodo del resto es que dispone de tres tarjetas gráficas, concretamente Tesla M2050 (ver Figura 2.4) que presentan las siguientes características:

- Arquitectura: Fermi.
- Memoria: 3 GB GDDR5 ECC.
- núcleos: 448 *Thread Processos*.
- Rendimiento en operaciones en coma flotante de doble precisión: 515 GFLOPS.
- Rendimiento en operaciones en coma flotante de simple precisión: 1.03 TFLOPS.
- Velocidad de la memoria: 1.55 GHz.
- Interfaz de memoria: 384-bits.
- Ancho de banda de la memoria: 148 GB/s.

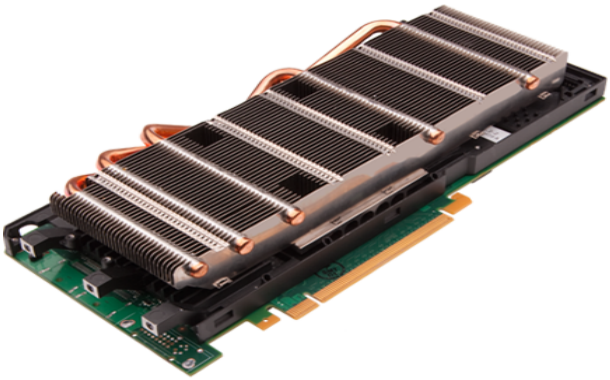


Figura 2.4: Tarjeta gráfica Tesla M2050 instalada en Euler

Este nodo se utiliza principalmente para el uso del paralelismo que se puede extraer de las GPUs. Al disponer de tres GPUs, este nodo presenta un procesamiento bruto de 1.5 TFLOPS en doble precisión y 3.09 TFLOPS en simple precisión. Con un total de 9 GiB de memoria de vídeo.

#### 2.2.1.3. Red de Interconexión

Euler es un clúster orientado al paralelismo y la red de interconexión es muy importante. En concreto, presenta tres redes de interconexión diferentes:

- Red de administración a 1Gbit.
- Red de interconexión a 1Gbit.
- Red de interconexión de baja latencia por InfiniBand 4X QDR.

Esta estructura es un esquema clásico de interconexión de nodos en un clúster, figura 2.5. La red de administración esta orientada al mantenimiento individual de cada uno de los nodos mientras que la red de interconexión de 1Gbit se dedica al control de las consolas y al acceso a los nodos por parte de los programas. Pero la parte principal del clúster es la red de baja latencia. Esta es la parte más importante dado que son necesarios un gran ancho de banda y una baja latencia para minimizar el tiempo de paso de mensajes dentro de un clúster, ya que este puede ser el factor más limitante en el rendimiento total del sistema. Esta red está creada con un switch de InfiniBand x4 con el protocolo QDR, lo que proporciona un ancho de banda de 40 Gbps brutos y 32 Gbps efectivos y que permite que cada nodo tenga una conexión parecida a la que tendrían si estuvieran integrados en un mismo hardware.

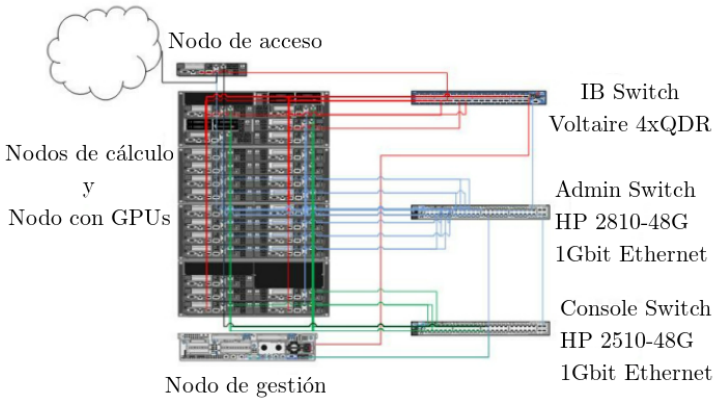



Figura 2.5: Esquema de interconexión de nodos de Euler.

### 2.2.2. Clarke

Clarke es un superordenador que pertenece al DTIC y esta alojado en laboratorio de I+D 2 en la segunda planta de la Politécnica III. Este es un ordenador orientado al uso de GPUs en concreto para el entrenamiento de redes neuronales de aprendizaje profundo.

Sus características hardware son las siguientes:

- CPU

  En el aspecto de la CPU presenta i7-6800K con estas funcionalidades:

  - 6 cores a 3,40 GHz, Max Turbo 3,80 GHz.
  - 15 MB Smart cache.
  - 64 bit, SSE 4.2,AVX2.
  - 14 nm , Max TPD 140 w.
  - Soporte Hyperthreading.

  Además de las características CPU, Clarke presenta estas especificaciones hardware:

  - Front Bus 1333-2133.
  - Memoria 16 GB DDR4-2666.
  - 2 discos SATA 7,2 k, 3TB (RAID1).
  - 1 disco SATA SSD, 500GB.

  Pero las partes hardware más importantes son las tres tarjetas gráficas que dispone. Dos de ellas son las que se utilizan para las operaciones de inteligencia artificial y operaciones de cálculo y la tercera está dedicada para salida de vídeo para tareas de mantenimiento. Las especificaciones son las siguientes:
- NVIDIA Quadro 2000

Figura 2.6: Tarjeta NVIDIA Quadro 2000

- Arquitectura: Fermi
- Memoria: 1024 MB DDR5
- núcleos: 192 *CUDA Cores*
- Interfaz de memoria: 128-bits
- Ancho de banda de la memoria: 41,6 GB/s

▪ NVIDIA Tesla K40c


Figura 2.7: Tarjeta Tesla K40c

- Arquitectura: Kepler
- Memoria: 12 GB GDDR5
- núcleos: 2880 *CUDA Cores*
- Interfaz de memoria: 384-bits
- Ancho de banda de la memoria: 288 GB/s

La primera que se muestra en el listado es la tarjeta orientada a salida de vídeo para operaciones de depuración del ordenador o fallos puntuales. Aparte de la tarjeta para visualizar, Clarke dispone de dos K40c, estas tarjetas están orientadas a centros de datos y superordenadores y por ello no disponen de salida gráfica, la potencia de su GPU esta orientada únicamente a cálculos paralelos. Esta tarjetas presentan una potencia de 4.29 TFLOPS. Como se ha comentado este es un superordenador orientado a la explotación del paralelismo dentro de la propia tarjeta, más que el uso del paralelismo de CPU como el ejemplo anterior de Euler.

### 2.2.3. Ordis

Ordis hace referencia al clúster creado en el laboratorio L14 en el edificio EPS 1 perteneciente a la EPSA. Ordis está conformado por 31 nodos de cálculo con características idénticas y un nodo maestro que opera en un entorno virtual localizado en la sala de servidores de ese mismo edificio. Este clúster está orientado a proporcionar un entorno de pruebas para alumnos y profesores, para la programación en paralelo, así como para aprovechar los recursos hardware de ese laboratorio cuando no se esté utilizando para fines docentes. Uno de los objetivos principales es que los equipos disponibles en el L14 puedan ser aprovechados tanto como un uso normal de clase como un sistema de altas prestaciones.

#### 2.2.3.1. Master Node

Ordis mantiene un esquema clásico de clúster, donde se encuentra un nodo maestro encargado de distribuir las tareas sobre los diferentes nodos de cálculo. Este nodo maestro en nuestro caso corre sobre un entorno virtual. Esto es así para facilitar que pueda ser mantenido y distribuido entre diferentes máquinas. La máquina virtual presenta estas especificaciones:

- Procesador de doble núcleo Genuine Intel KVM a 3Ghz
- 2GB de memoria RAM
- Disco duro HDD de 100GB

El máster no procesa ningún tipo de cálculo solo es encargado de operaciones de coordinación de los nodos y coordinación de trabajos. Por ello las especificaciones hardware no son de extremada importancia en el Master Node.

#### 2.2.3.2. Nodos de Cálculo

Los nodos de cálculo que utiliza el sistema son los ordenadores que hay disponibles dentro del edificio 1 de la EPSA, en concreto el laboratorio L14. Estos son los ordenadores que se utilizan para las clases normales de diferentes asignaturas que se estudian en la carrera de Ingeniería Informática así como asignaturas de otros grados. Por lo que estos ordenadores no están pensados para ser usados como un clúster híbrido. Pero a pesar de este hecho, presentan unas serie de componentes hardware interesantes para su explotación. En concreto cuentan con estas especificaciones:

El procesador que utiliza es el Pentium G840 que presenta estas características:

- 2 cores a 2,80 GHz.
- 3 MB Smart cache.
- 64 bit, SSE 4.2.
- 32 nm , Max TPD 65 w.
- Sin Soporte Hyperthreading.

Además de la CPU, los nodos presentas estas características hardware:

- Front Bus 1066-1333.
- Memoria 8 GB DDR3-1333.
- 2 discos SATA 4,4 k, 500GB.
- Tarjeta de red Realtek 1Gb.

También cada nodo dispone de una GPU. En concreto las GTX 480 cedidas por NVIDIA Figura 1.3. Estas presentan las siguientes características:

- Arquitectura: Fermi.
- Memoria: 1,5 GB GDDR5.
- núcleos: 480 *CUDA Cores*.
- Interfaz de memoria: 384-bits.
- Ancho de banda de la memoria: 177.4 GB/s.

#### 2.2.3.3. Esquema de Red

El L14 dispone de una red de interconexión Ethernet a 1 Gb/s. Esta red es la única de que dispone el clúster para el paso de mensajes y el intercambio de archivos. Los nodos se dividen en tres bancadas dentro del laboratorio, todos los nodos y el master están conectados a un mismo switch, ver Figura 2.8.

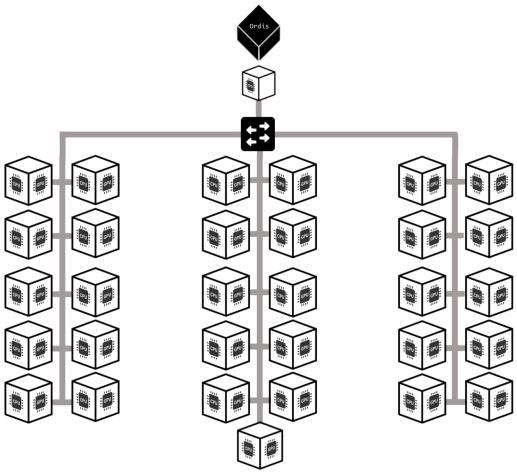


Figura 2.8: Distribución de ordis, con un master node y 31 nodos de procesamiento.

## 2.3. Software

En esta sección se va a describir el software que utiliza cada uno de los sistemas que ya se han comentado. En concreto el software que disponen para el lanzamiento de tareas en cada uno de los superordenadores comentados anteriormente así como librerías y funcionalidades propias de cada sistema.

### 2.3.1. Euler

Euler presenta en sus nodos el sistema operativo conocido como Centos 7, un sistema operativo orientado a entornos empresariales el cual es un *fork* de RedHat y es de uso gratuito. Además, presenta una característica muy interesante para entornos de producción: cada versión dispone de soporte de actualizaciones durante diez años. Para el sistema de coordinación de lanzamiento de tareas, utiliza SGE o *Sun Grid Engine*, el cual es el encargado de gestionar los trabajos, así como las prioridades y las colas de los mismos. Sobre este software se profundizará con más detalle en la parte de software de Ordis (Sección 2.3.3), ya que se utiliza en el proyecto para la gestión de los trabajos. Así mismo, incorpora una serie de software que se lista a continuación:

- Compiladores.
    - Compilador de C/C++ con optimizaciones para procesadores Intel

    - Compilador de Fortran con optimizaciones para procesadores Intel
    - Compilador de C/C++ de GNU
    - Compilador de Fortran 95 de GNU
    - NVIDIA Toolkit con compilador nvcc
- Librerías paralelas
    - openmpi-gcc para los compiladores GNU
    - openmpi-intel para los compiladores de Intel
    - mvapich-gcc para los compiadores de GNU
    - mvapich-intel para los compiladores de Intel
    - mvapich2-gcc para los compilaodres GNU
    - mvapich2-intel para los compiladores Intel
    - impi Implementacion de MPI por intel
    - Spark 1.3.1
- Librerías científicas
    - Intel MKL
    - CUDA
    - PCL
    - OpenCV
    - Armadillo
    - PETSC
    - R
    - Matlab
    - JDK
    - Python 2.7.5 y 3.5.4
    - NAMD Molecular Dynamics Software
    - FDS - Fire Dynamics Simulator

Como se puede constatar, se dispone de una cantidad ingente librerías y entornos paralelos, algunos de ellos son incompatibles entre si. Debido a ello, dentro del sistema operativo existe un sistema conocido como *modules*, encargado de cargar las diferentes librerías que se necesiten para cada usuario concreto. Así se evitan problemas de incompatibilidades dentro del clúster ya que cada uno controla las librerías que necesita su proyecto.

### 2.3.2. Clarke

Clarke dispone como sistema operativo Ubuntu 16.04 LTS. Dentro de este sistema, como no dispone de más nodos, no es necesaria la utilización de un controlador de trabajos como SGE comoe en Euler. Dentro de Clarke, para el control de las librerías y los proyectos de los usuarios se emplea

un sistema de contenedores virtuales, en concreto utiliza *Docker*. Con este sistema se permite aislar cada trabajo de forma independiente al resto, de manera similar al uso de un sistema de máquinas virtuales pero más flexible y con menos consumo de espacio y recursos. El proceso es el siguiente: un usuario crea un contenedor de docker donde el usuario instala todo el software que necesite, después en esa instancia de docker correrá su aplicación con las librerías que descargue o prepare de forma aislada a otros contenedores.

### 2.3.3. Ordis

Cuando se presentó el proyecto de creación del clúster, una de las decisiones de mayor importancia fue la elección del sistema operativo, dado que el objetivo era la utilización del clúster como un clúster híbrido para su uso con CPU y GPU, se debía encontrar un SO que soportase las tarjetas gráficas NVIDIA. Tras una búsqueda encontramos el artículo [14] en el foro de NVIDIA que explicaba de forma superficial el montaje de un clúster híbrido. En el artículo utilizaban Rocks. Rocks es un sitema operativo enfocado a su uso en clústers, en concreto utilizamos la versión 6.2. Rocks utiliza como sistema operativo base Centos 6.6, que como se comentó presenta la característica de que provee actualizaciones durante diez años desde el lanzamiento de la versión. En el Anexo A se explica con detalle la instalación completa de Rocks para el caso concreto del L14 de la EPSA.

Es importante recalcar en la instalación el software base que instala en el *master node*. Del software que nos permite disponer el instalador se seleccionaron los siguientes paquetes:

- Area 51. Incluye dos programas para el control de seguridad del clúster. El primero es Tripwire, un software encargado de detectar cambios dentro del kernel del sistema y dar un aviso al administrador del sistema, y el segundo es chkrootkit que se utiliza para evitar la toma de control del sistema utilizando rootkit.
- Base. Incluye el código base del SO necesario para su funcionamiento.
- Fingerprint. Incorpora el software fingerprint encargado de leer una lista arbitraria de ficheros binarios y guardar sus dependencias creando un fichero llamado Swirl, este puede servir para ver si una aplicación puede funcionar en otro sistema. Se utiliza principalmente para la instalación de software dentro de los nodos en Rocks.
- ganglia. Importa el software ganglia, con el que es posible la monitorización de los nodos del clúster y extraer información sobre carga de los nodos, consumo de red, nodos activos, nodos caídos, etc.

- Hpc.Trae una serie de librerías necesarias para los entornos paralelos en concreto MPI en tres de sus versiones, OpenMPI, MPICH, MPICH2. También dentro del paquete se incluye una serie de benchmarks para medir el rendimiento de la red interna del clúster, en concreto Iperf, stream, IOzone.
- Java. Instala la maquina *Java virtual machine* para la ejecución de programas Java.
- Kernel. Incluye el kernel del SO, básico para el funcionamiento del sistema.
- OS. Incorpora el SO, en concreto Centos 6.6.
- Python. Trae el interprete de python en su versión 2.7.
- SGE. Instala y configura *Sun Grid Engine* en el clúster.
- Web-server. Incorpora un servidor web, necesario para el funcionamiento de ganglia.

Dentro de todo este conjunto de software, que se incorpora al sistema durante la instalación del *master node*, incidiremos en el funcionamiento y uso de dos de ellos: el paquete de Ganglia y SGE, ya que estos desarrollan un rol muy importante para el desrrollo de este clúster.

#### 2.3.3.1. Ganglia

Ganglia es un sistema escalable de monitorización de supercomputadores, clústers y redes de computadores. Este software nos permite ver estadísticas, en directo o en diferido, de los diferentes nodos que lo componen. Dentro de las métricas que incorpora se encuentra uso de CPU, memoria RAM, almacenamiento y consumo de red. Algunas compañías que lo utilizan a nivel empresarial son Cray, MIT, NASA y Twitter.

Ganglia se compone de tres partes fundamentales: El Ganglia Monitor Daemon o gmond, el Ganglia Meta daemon o gmetad y el PHP-web Front o gweb (ver Figura 2.9). Utiliza un protocolo multicast para el paso de información. Para la representación de los datos utiliza un esquema XML unido con un XDR para poder trasmitir la información entre los nodos de forma eficiente y que soporte diferentes tipos de máquinas dentro del mismo área de monitorización. Esa información es transmitida a RRDtool que es la encargada del almacenamiento de la información.

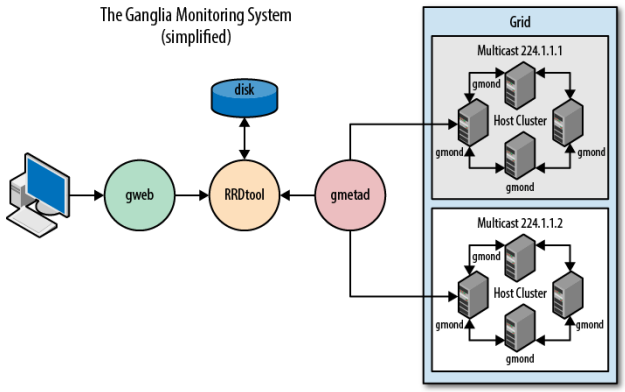



Figura 2.9: Funcionamiento de Ganglia [15].

- GMOND.
  Gmond es un daemon multihilo que se ejecuta en cada uno de los nodos del sistema. Es el encargado de comunicarse con el SO para extraer información sobre el estado de la CPU, memoria, disco, etc. A diferencia de otros sistemas de monitorización, gmond no espera ningún tipo de señal para mandar los mensajes, manda la información a través de broadcast a la red cada cierto tiempo. Los datos son recogidos y enviados utilizando el protocolo XDR a la red para ser recogidos por otras herramientas. En la Figura 2.9 se aprecia cómo se recolecta la información dentro del clúster y en concreto en la Figura 2.10 se puede ver en detalle el funcionamiento de gmond extrayendo los datos para ser enviados.

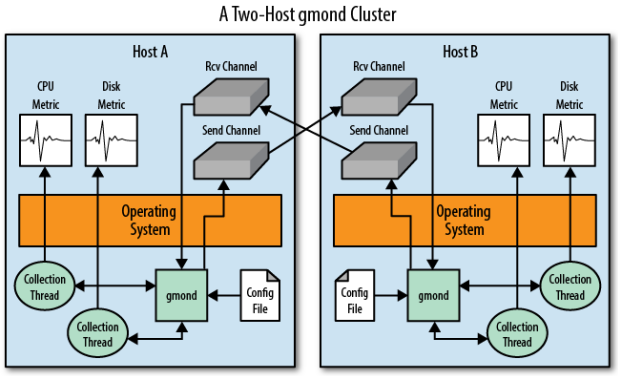



Figura 2.10: Funcionamiento de gmond en dos nodos.

- GMETAD. Gmetad esta alojado dentro del *master node* y es un programa encargado de extraer la información enviada por gmond dentro del clúster y añadirla a RRDtool para su almacenamiento y posterior uso.

Además tiene la posibilidad de extraer información de otros gmetad para crear una estructura más jerárquica.

- GWEB.
  GWeb esta también alojado en el *master node* y es el encargado de recuperar la información almacenada en RRDtool y mostrarla a través de una interfaz web. Esta nos muestra en detalle la información en vivo del sistema que se está monitorizando así como los registros de uso del mismo durante diferentes espacios temporales.

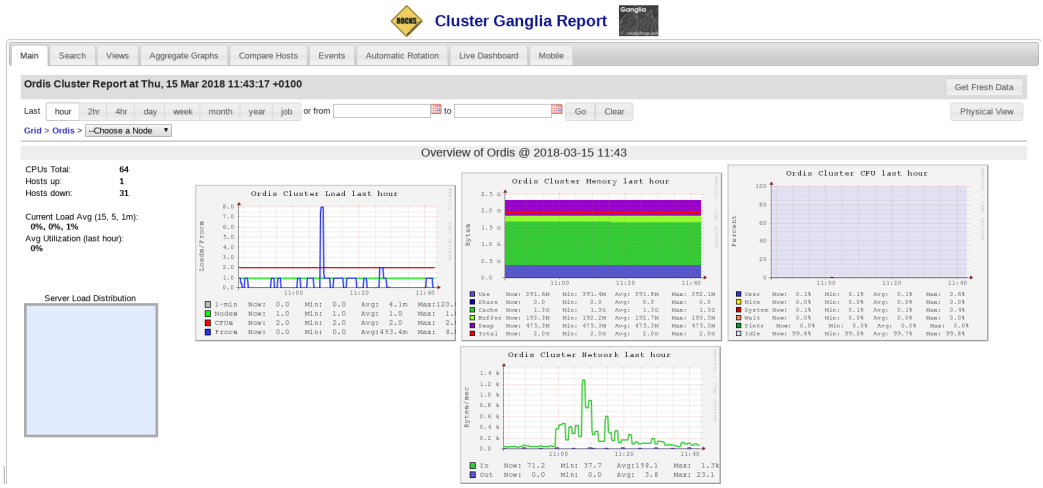



Figura 2.11: Ganglia funcionando en Ordis

Con estas tres partes, ganglia nos permite monitorizar el estado del clúster durante la ejecución de procesos, en tiempo real con un mínimo impacto en la red. Para más información sobre ganglia se puede consultar el siguiente libro [16].

### 2.3.3.2. Sun Grid Engine

Sun Grid Engine o SGE es un gestor de trabajos para entornos de clúster, creado originalmente para los sistemas propietarios SUN pero tras la compra de de Oracle a SUN, el proyecto se declaró *opensource*, por lo que las versiones han sido mantenidas por la comunidad. El funcionamiento del SGE es como el de un director de orquesta, es el encargado de gestionar los trabajos que son lanzados en el clúster. Para ello, todos los trabajos son gestionados en un nodo maestro, donde está en funcionamiento el núcleo de SGE. Desde dicho núcleo, SGE llama a los demonios de ejecución en cada uno de los nodos (ver Figura 2.12).

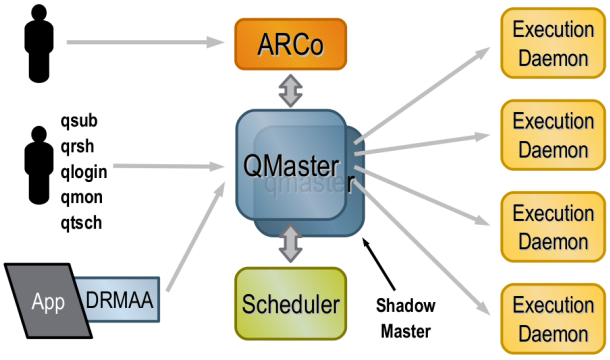



Figura 2.12: Esquema de funcionamiento de SGE.

Para la gestión de los múltiples trabajos en el clúster, SGE utiliza una serie de colas configurables, tanto para la urgencia de los trabajos, como para un hardware en concreto. Además, cada usuario dispone de su prioridad en cuanto a que su trabajo entre dentro de los recursos. SGE lanza los trabajos con el propio usuario a través de los *execution daemons* por lo que evitamos que algún proceso pueda acceder a partes de la memoria que no le pertenecen. Para el uso del SGE, tanto usuarios como administradores tiene que utilizar una serie de comandos para tanto, lanzar, monitorizar, o parar los trabajos lanzados. A continuación describimos los más importantes.

- QSUB.
  Este comando es el encargado de insertar los trabajos dentro del clúster. Lo común con el lanzamiento de este comando es lanzarlo pasándole por parámetro un *script* que contiene todos los parámetros de lanzamiento así como el programa o los programas a lanzar dentro de ese trabajo. Un ejemplo de un *script* para qsub es el mostrado en el Código 2.1.

```
#!/bin/bash/
#$ -S /bin/bash
#$ -V
#$ -cwd
#$ -q gpu.q
#$ -N cudaTest5
#$ -m be
#$ -pe mpi 25
mpirun -np 25  ./jacobi_cuda_aware_mpi -t 5 5 -d  3686
```

Código 2.1: Script para lazar un trabajo con Qsub.

  Como podemos observar, en la línea 1 marcamos el fichero como un *script* en bash y a continuación en las líneas 2 - 8 se especifican los comandos de qsub. Qsub reconocerá todo lo que se encuentre delante la estructura #$ como un parámetro propio. Unos de los más importantes

son *-pe* , para la selección del entorno paralelo, *-N* para especificar el nombre del trabajo y *-q* para la selección de la cola en la que depositar nuestro trabajo. Para ver en detalle todos los parámetros se pueden consultar en el manual web de qsub. Después de la especificación de los parámetros todo lo que se especifique después de eso sera lo que se ejecutará en el clúster.

- QSTAT.

  Este comando es utilizado para monitorizar los trabajos en el clúster así como comprobar el estado de las colas. Por defecto, cuando se lanza sin ningún parámetro muestra todos los trabajos que están dentro de SGE con el usuario que está utilizando durante el lanzamiento del comando. Suele tener este aspecto mostrado en el Código 2.2 al ser lanzado.

```
job-ID  prior   name       user     state submit/start at
    queue
-----------------------------------------------------------------------

2325 0.50000 cudaTest5  jaquer        qw    03/29/2018 17:59:27
```

Código 2.2: lanzamiento de Qstat.

  Como se puede observar, nos proporciona mucha información con respecto al trabajo lanzado. Como por ejemplo el ID del trabajo, la prioridad, el nombre que le hemos asignado, el estado y cuándo empezó. Para una lista completa de comandos y estados, recomendamos visitar la página del Euler de la IUII

- QMON

  Cuando se desea configurar los aspectos de SGE, existen dos formas de cambiar su comportamiento. Bien directamente accediendo a los ficheros de configuración o bien podemos utilizar qmon que nos muestra una interfaz GUI que nos permite tanto configurar todos los aspectos de SGE como de ver el estado del clúster así como los trabajos, ejecutados, en ejecución o en espera tal y como se muestra en la Figura 2.13.

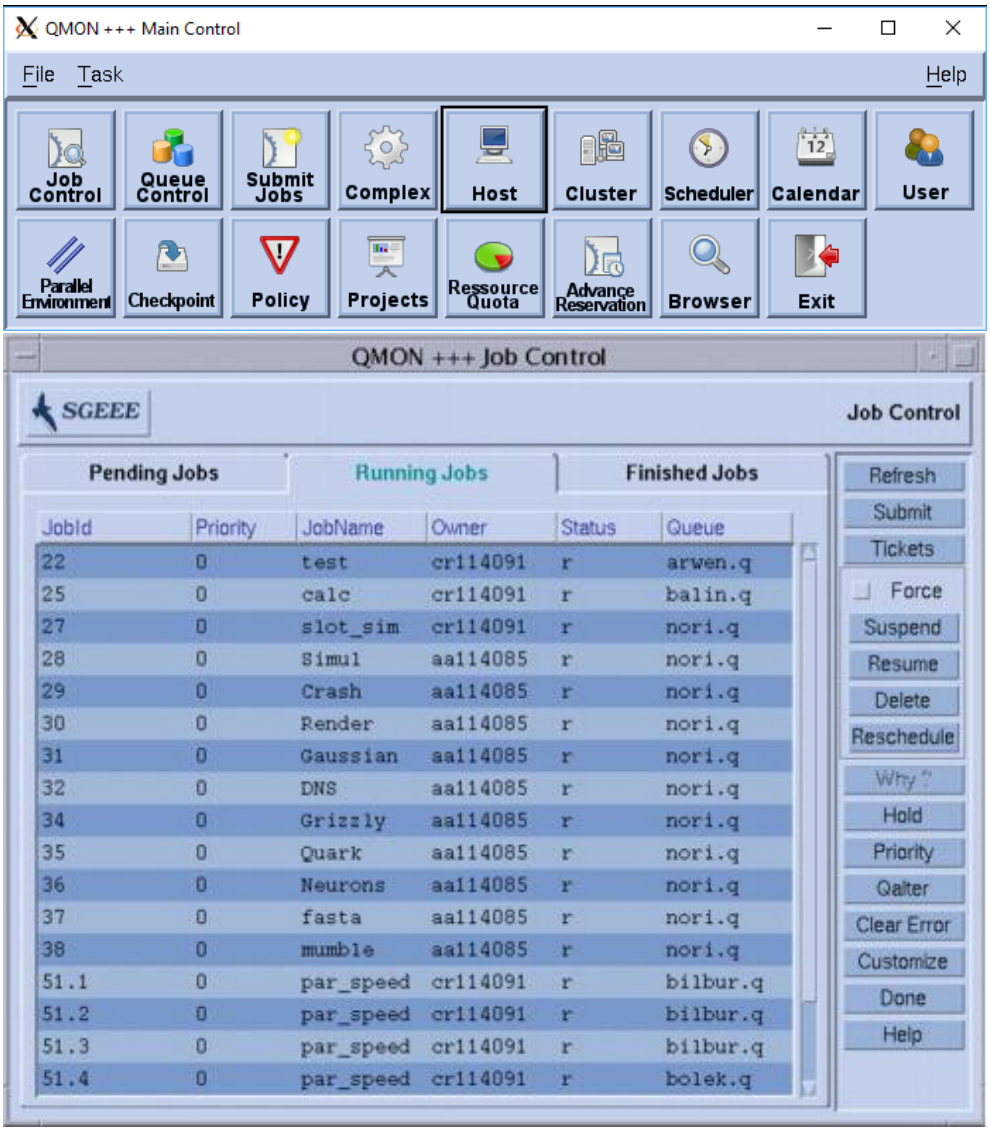



Figura 2.13: Visualización de Qmon

# Capítulo 3

# Sistema: Front End

En este capítulo describiremos la parte del sistema propuesto dedicada a los usuarios, es decir, el *front end*. En primer lugar, expondremos las necesidades de esta parte de la propuesta en la Sección 3.1. Seguidamente, describiremos las particularidades de la interfaz web en la Sección 3.2. De la misma manera, detallaremos la API REST que orquestra todo el proceso en la Sección 3.3. Por último, trataremos una serie de *scripts* útiles para esta parte del sistema en la Sección 3.4.

## 3.1. Introducción

Como hemos mencionado durante el transcurso de este documento, la configuración, puesta en marcha y uso de un clúster es un proceso bastante complejo. Generalmente, la configuración y puesta en marcha solo es necesario que se realice una vez en el vida útil de clúster. En cambio, dado que son los usuarios quienes más lo van a utilizar, una parte fundamental en el desarrollo de este proyecto fue la creación y puesta en marcha de una interfaz sencilla para que los mismos puedan subir y lanzar proyectos al clúster.

Para implementar esta solución se requieren una serie de condiciones. En primer lugar, es requisito fundamental que pueda integrarse con los sistemas existentes en la EPSA. Segundo, los usuarios no tienen acceso al clúster de forma directa en ningún momento. Tercero, tiene que ser accesible siempre, aunque el clúster no este operativo. Cuarto, tiene que ser fácil de ampliar. Y quinto y último, tiene que ser sencillo de utilizar.

Teniendo en cuenta los requerimientos que se nos propusieron, se decidió el uso de una aplicación web con *API REST*. Esta aplicación *FULL REST* recibe las peticiones desde un servidor de la EPSA hasta el nodo *master* y la aplicación ejecuta una serie de *scripts* para recibir, preparar, lanzar y recoger

los trabajos que se van a procesar en el clúster. Esta aplicación seguirá este esquema básico que se muestra en la Figura 3.1, el cual desarrollaremos en las siguientes secciones.

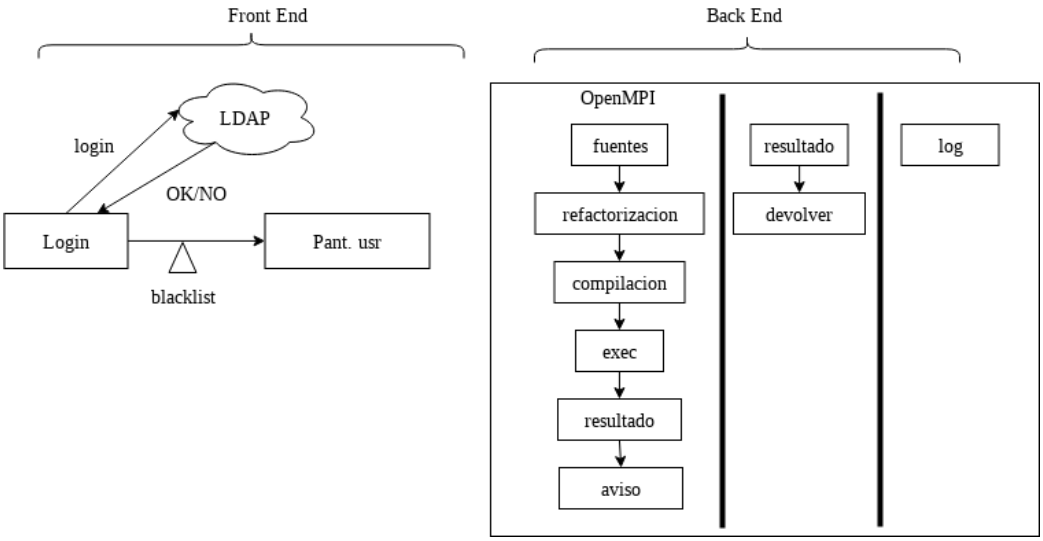



Figura 3.1: Esquema general de la aplicación.

## 3.2. Interfaz Web

Para comenzar con esta parte del proyecto, describiremos el proceso que sigue un usuario típico para lanzar un trabajo con nuestra aplicación propuesta. En primer lugar, el usuario entra en la interfaz web de la aplicación. Antes de entrar se le preguntan sus credenciales de la EPSA. Con ello, la aplicación garantiza que conocemos el usuario y que está identificado dentro de la EPSA, además extraemos del registro LDAP información referente al usuario, como por ejemplo su email, que será el recipiente al que se le enviarán los correos de la aplicación.


Figura 3.2: Interfaz web de Ordis cortesía de los técnicos de la EPSA.

Desde esta interfaz se nos indica que subamos un fichero comprimido en formato ZIP (tal y como se muestra en la Figura 3.2). Este archivo contendrá nuestro trabajo que se va a lanzar en el clúster. Para ser considerado válido para la aplicación, el fichero debe contener una estructura que describimos a continuación.

Tal y como se indica en la Figura 3.3, tiene que contener las carpetas include, lib y src, junto al archivo makefile que compilará el proyecto y un archivo opcional param.txt con los parámetros de lanzamiento de la aplicación. Si este archivo contiene más de una línea, se generará un trabajo diferente por cada línea del archivo.

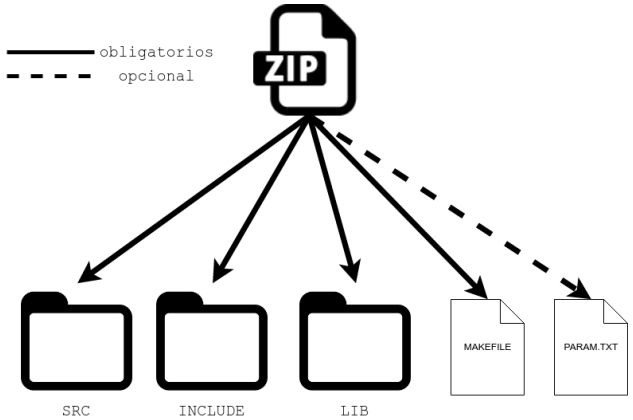



Figura 3.3: Esquema de la guía de estilo del fichero ZIP.

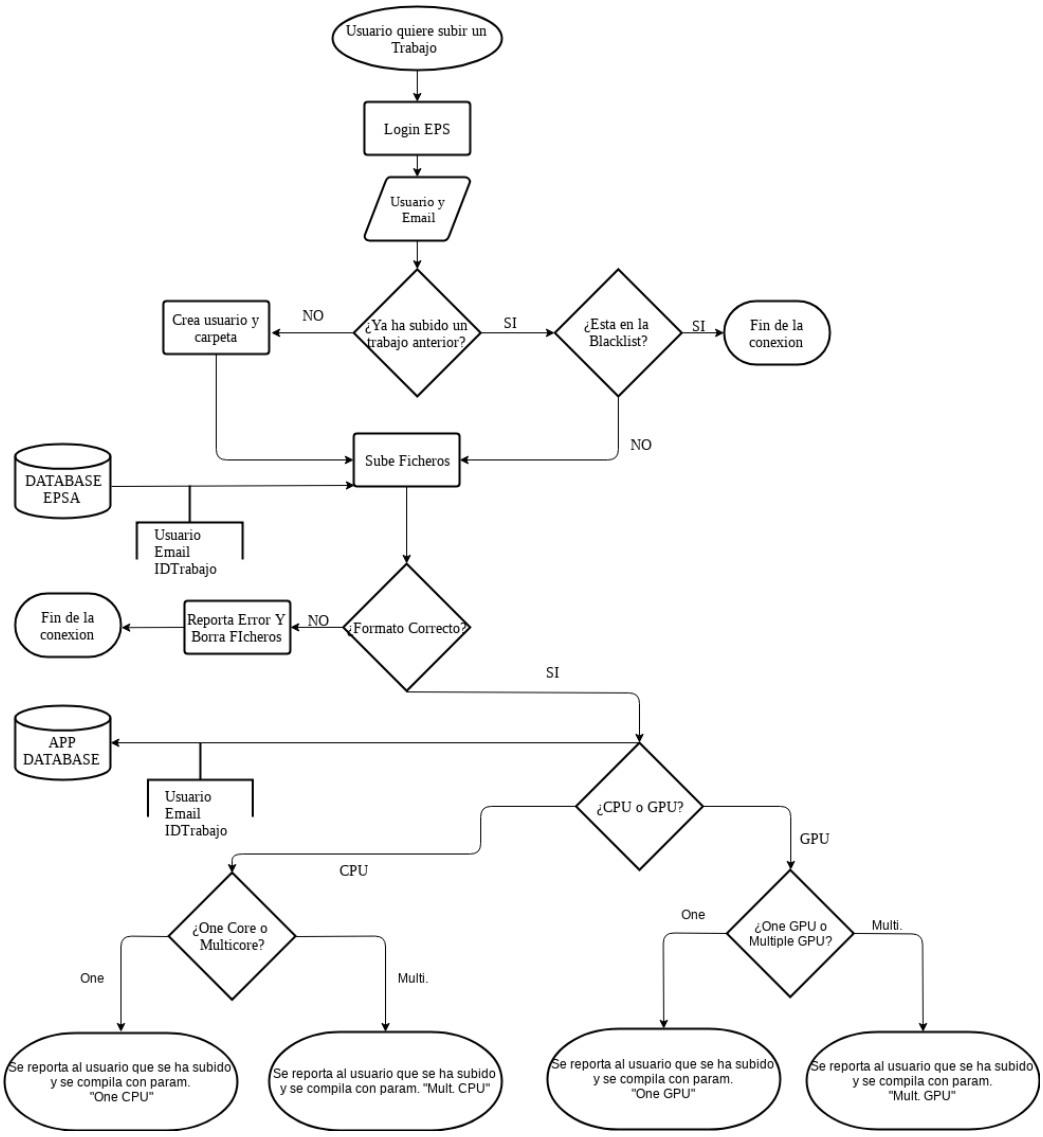



Figura 3.4: Diagrama de flujo de la aplicación web

Tras la subida del fichero se nos muestra un desplegable con las diferentes opciones de posibles trabajos que se pueden lanzar dentro del clúster. Aquí se incluyen las opciones de lanzamiento con MPI, CUDA, MPI + CUDA, Otro paralelismo y No parallel. Cada una de estas opciones genera una configuración diferente (explicado de forma más detallada en la Sección 3.4). Por último, la interfaz nos presenta el campo número de nodos, donde el usuario selecciona el número de nodos que se van a reservar para su trabajo. Este número no puede ser superior a 31 en CUDA + MPI ni a 62 con MPI. Esta barrera se debe a la limitación de nodos de clúster. Una vez todos los

parámetros del trabajo son seleccionados por el usuario, ya puede proceder a pulsar el botón *upload*. Con ello la web llamará a un método *POST* de la *API REST* que reside en Ordis. Si cumple todos los requisitos comentados anteriormente, el trabajo se habrá subido correctamente al clúster, con lo que la aplicación utilizará una serie de *scripts* para comunicarse con el sistema operativo y con Sun Grid Engine para que el trabajo sea lanzado. La Figura 3.4 muestra un diagrama de flujo explicativo del proceso anterior.

## 3.3. API REST

Una vez los datos han sido recopilados por la interfaz web, estos son enviados a la *API REST* alojada en Ordis. Su función es la de organización de todo el proceso de envío un trabajo al clúster. La API está escrita en lenguaje PHP utilizando el framework Epiphany. Tanto el lenguaje como el framework fue elegido por recomendaciones directas del grupo de técnicos de la EPSA, ya que casi todos sus servicios de *API REST* están creados en este lenguaje.

Es importante destacar que dicha API necesita una serie de dependencias externas para su funcionamiento, las cuales se muestran en la Figura 3.5. Estas dependencias son necesarias para que la API pueda gestionar correctamente los trabajos así como mantener a los usuarios y administradores conscientes de todo lo que ocurre en el clúster.

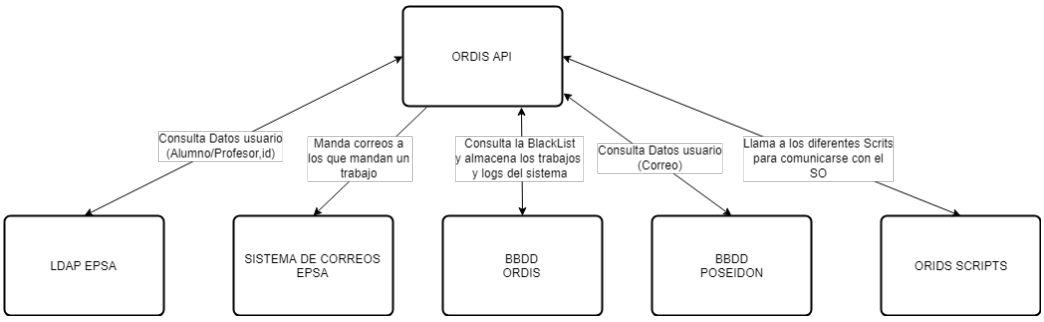



Figura 3.5: Dependencias de API REST.

A continuación expondremos los diferentes métodos que acepta la *API REST* así como su funcionamiento interno.

El primer método a tratar es aquél que más va ser utilizado durante la vida útil de la aplicación. Este es el método *POST, /operation/code/$user*. Dicho método es el encargado de subir un nuevo trabajo al clúster para que sea procesado. Como se especifica en el diagrama de la Figura 3.4, el primer

paso, después de que un usuario se haya autentificado correctamente en los servidores de la EPSA y haya rellenado el formulario con el trabajo que se desea realizar, es el procesamiento de los parámetros que han llegado a la API. Primero se comprueba de si el usuario existe en el registro de LDAP de EPSA y si además no esta en nuestra lista de usuarios bloqueados por algún mal uso de la aplicación tal y como mostramos en el Código 3.1.

```
$ldap = new LDAPConnection();
            //search in LDAP registry
      $usrData = $ldap->buscar($user);
      if($usrData != "ERROR: No such user" && UsrConnection::
   notBanned($usrData['uid']))

////////////////////////////////////////////////////
// Buscar un usuario en LDAP EPSA
////////////////////////////////////////////////////


public function buscar($user) {
      if($this->conn) {
        $ldapbind = ldap_bind($this->conn, $this->username, $this->
   password);
        if($ldapbind) {
          $ldaptree="uid=".$user.",ou=people,ou=eps,o=ua,c=es";
          $result=ldap_search($this->conn, $ldaptree, "(cn=*)");
          if (ldap_error($this->conn) == "Success") {
              // PROCESS DATA
          }

      }
    }
  }
////////////////////////////////////////////////////////////
// Buscar si el usuario no esta autorizado
// a utilizar el cluster
////////////////////////////////////////////////////////////


public static function notBanned($user) {
      $notBanned = false;
      $aux=0;
      $conexion = new Connection();
      $conn = $conexion->getConnection();
      $consulta = "SELECT count(*) FROM CAAS_BANNED_USERS WHERE USER
   =?";
      if($sentencia = $conn->prepare($consulta)) {
          // PROCESS DATA
        }

      }
  }
```

Código 3.1: Extracción de usuario y verificación de acceso al clúster.

Después este proceso que comprueba que el usuario existe como alumno de la EPSA y además no está dentro de la lista negra del clúster, se procede a subir el fichero ZIP proporcionado por el usuario y a comprobar que cumple con los criterios obligatorios. Primero se comprueba de que el fichero proporcionado es un fichero ZIP. Los tipos mime aceptados son:

- application/zip
- application/x-zip-compressed
- multipart/x-zip
- application/x-compressed

Esto nos garantiza que el fichero que contiene el trabajo podrá ser procesado por nuestra API. Después se comprueba que el ZIP cumple con la estructura anteriormente comentada y mostrada en la Figura 3.3. A continuación, la aplicación llama a un *script* para comprobar si el usuario ya está creado en el clúster, si no es así lo crea, estos *scripts* serán explicados en profundidad en la Sección 3.4. La API continúa con la comprobación del número de nodos solicitados. Si se supera el número de nodos disponibles por el clúster, la API devuelve un error de número de nodos. Después se produce la comprobación del tipo mime y la descompresión del ZIP proporcionado en el fichero *home* del propio usuario en el sistema de ficheros del clúster. Esto está expresado en el Código 3.2 de la API.

```
if($usrData != "ERROR: No such user" && UsrConnection::notBanned(
    $usrData['uid'])) { //No ha habido error
     //checks if user exist if not create user in the system
     shell_exec("/export/scripts/maintenance/checkUser.sh ".$usrData[
    "uid"]);
     $success = false;
     //get type and n nodes
     $tipo = $_POST['tipo_operacion'];
     $n_nodos = $_POST['n_nodos'];

     $MAXIMO_NODOS_AUX= self::$MAXIMO_NODOS_CPU;
     if($tipo === "cuda" || $tipo === "mpi-cuda")
         $MAXIMO_NODOS_AUX= self::$MAXIMO_NODOS_GPU;
     if($n_nodos <  self::$MINIMO_NODOS || $n_nodos >
    $MAXIMO_NODOS_AUX) {
         echo "ERROR: Numero de nodos incorrecto. Ha de estar entre "
    .self::$MINIMO_NODOS." y ".$MAXI$
         return;
     }

     //Trying to upload and unzip the file
     if($_FILES["zip_file"]["name"]) {
     $filename = $_FILES["zip_file"]["name"];
     $source = $_FILES["zip_file"]["tmp_name"];
     $type = $_FILES["zip_file"]["type"];
```

```
    $name = explode(".", $filename);
    if(!self::checkFileUploaded($filename, $type, $name)) {
        echo "The file you are trying to upload is not a .zip file.
   Please try again.";
        //console($message, 1, 5);
    }
    else {
        $target_path = "uploads/".$filename;
        $extract_path = "/export/home/".$user."/"."projects/".$name
   [0];
        if(self::extractFile($filename, $name, $user, $source,
   $target_path, $extract_path))$
```

Código 3.2: Creación del usuario y extracción de los ficheros.

Cuando el trabajo ha pasado ya ha pasado por este proceso, se considera aceptado por el sistema y se procede a la llamada del *script* principal: el encargado de la compilación y lanzamiento del trabajo. Para el funcionamiento correcto de este *script* es necesario que la API recoja una serie de datos del usuario y se lo proporcione al *script*. A partir del momento que la API llama al *script* que toma control absoluto sobre el trabajo y la aplicación registra en el log del sistema que el trabajo se ha subido correctamente y que usuario lo ha proporcionado. A partir de ahí queda a la espera del siguiente usuario. Esto se representa en el Código 3.3 de la API de esta forma.

```
[...]/* Escribimos fichero con los datos del cliente*/
//file_put_contents
$parsedUserData = $usrData["uid"].":".$usrData["gidnumber"]." mailto
    --> ".$usrData["mail"];
$dataFilePath = $extract_path."/datos.txt";
//Get priority from BD
/* Comprobacion de tipo */
[...]
//Here adding operation to BD
$codeConn->addCodeOperation($user, $opID, "subido", '', '',
    $fecha_subida, $extract_path);

//Ask for usr priority
$usrConn = new UsrConnection();
$priority = $usrConn->getUserPriority($usrData["uid"]);
echo $usrData["uid"].":".$usrData["gidnumber"]." prority --> ".
    $priority."<br/>";

//Recibe el usuario (sistema), nombre_carpeta_proyecto, email_usr,
    tipo_code (cpu/gpu), tipo_paralelismo (mpi/otro), prioridad, n
    nodos
$output=[];
switch($tipo) {
  case "mpi":
    echo "Compiling MPI code";
    //Script de MPI
```

```
    shell_exec("/export/scripts/compile/compileStdCode.sh "
    .$usrData['uid']
    ." ".$name[0]
    ." ".$usrData['mail']
    ." 'cpu'"
    ." 'mpi'"
    ." ".$priority
    ." ".$n_nodos
    ." &> /dev/null &");
[...]/*Resto de tipos*/
}

//Add to log
$res = Log::addLog($usrData['uid'], "Se ha subido el trabajo $name
    [0]", 1);
```

Código 3.3: Recolección de datos de usuario y lanzamiento de la compilación.

La siguiente operación que dispone la API es la de actualizar un código ya existente. Este método utiliza una metodología muy similar a la que se utiliza para subir un trabajo al *clúster* con la diferencia de que se comprueba si el código que se indica está en una fase de espera, antes de que esté en ejecución o después de la ejecución errónea. Si es afirmativo la API se encarga de subir el código nuevo proporcionado por usuario. Este Código 3.4 seguirá el mismo proceso que se sigue cuando se sube un trabajo nuevo.

```
$ldap = new LDAPConnection();
      $usrData = $ldap->buscar($user);

      if($usrData != "ERROR: No such user") { //No ha habido error

        //Comprobamos que la operacion existe y es updateable
        $codeConn = new CodeConnection();
        $state = $codeConn->getOperationState($user, $operation);
        $path = $codeConn->getOperationPath($user, $operation);

        if($state === "subido" || $state === "make_error" || $state
    === "make" || $state === "finished_with_errors")
```

Código 3.4: Comprobación del estado de un trabajo para su actualización.

Como última operación que puede realizar la API, existe la posibilidad de que un usuario borre un trabajo. Para ello, como ocurre con la operación de actualización de un trabajo, el trabajo tiene que estar en fase de espera. Este *script* el trabajo de la cola y borrar todos los archivos contenidos en la carpeta del usuario referentes al trabajo en cuestión. El código completo de esta parte de la API se encuentra en el Anexo B.

## 3.4. Scripts API

Para que la aplicación de los trabajos funcione correctamente fue necesario la creación de varios *scripts*: un *script* para la compilación y lanzamiento de los trabajos, así como uno para la monitorización de la finalización de los procesos y para envío del resultado del trabajo al usuario que lo mandó. Estos *scripts* son necesarios para poder unir la interfaz web con el esquema clásico de trabajo en un clúster con SGE.

### 3.4.1. Script de Compilación

Cuando la API ya ha cumplido su parte de recibir los datos del usuario y extraer todos los datos necesarios, corre por parte del *script* de compilación la responsabilidad de la generación del ejecutable final y el lanzamiento del trabajo del usuario en el clúster.

El proceso de compilación sigue varias fases marcadas. La primera es comprobar que la llamada con la que se invoca al *script* desde la API tiene todos los parámetros necesarios. Además, comprueba que el usuario y la carpeta existen en el sistema. Una vez que se ha comprobado que el proyecto tiene todos los archivos que necesita se procede a la fase de compilación del mismo. Utilizando el Makefile, que es obligatorio para el proyecto, se procede a la compilación. Dado que múltiples usuarios pueden lanzar un proyecto al clúster, y que la compilación se ejecuta en el nodo *master* el cual dispone de recursos limitados, se optó por la creación de un sistema basado en semáforos para controlar esta situación.

El sistema funciona de la siguiente manera: una vez el proyecto ha pasado por la comprobación de los ficheros se crea un fichero de texto nombrado como *.lock* que contiene el usuario y el proyecto al que corresponde dicho y se llama a la función de lock que gestionará el semáforo (ver Código 3.5).

```
#Checks makefile
  if [ ! -f "/export/home/$user/projects/$proyect/makefile" ]; then
                echo "ERROR MAKEFILE DON'T EXIST"
                exit
        fi
#Create the .lock file for this build
  echo "$user$proyect" > .lock
#check if locket
  lock
```

Código 3.5: Comprobación del Makefile y creación del *.lock*.

Esta función lock lo primero que intenta es copiar el fichero en la carpeta *home* de todos los usuarios, utilizando el comando cp con el parámetro

-n para evitar que pueda sobrescribir el archivo. Una vez que intenta eso, comprueba si la copia del fichero ha sido satisfactoria, comprobando que el contenido del fichero en *home* forma parte del proyecto que está ejecutando el *script*. Si lo ha conseguido, sale del bucle y empieza la compilación del proyecto. Por el contrario, si no ha sido posible la escritura del fichero, el *script* entra en un *leep* donde cada dos minutos comprueba si ese archivo *.lock* existe, si otro proceso lo borra tras terminar la compilación, sale del bucle y se repite el proceso de sobreescritura del fichero y comprobación de si ha sido posible la reserva. Este proceso se encuentra expresado en el Código 3.6.

```
function waitLock {
while [ -f "/export/home/.lock" ]
        do
  # echo "sleeping"
          sleep 2m
        done
}
#function check if I lock
function lock {
locket=0
while [ $locket -eq 0 ]
  do
  #trys to copy
    cp /export/home/"$user"/projects/"$proyect"/.lock /export/home/
    -n
  #checks if I sucess lock
  if [  "$(cat /export/home/.lock | grep "$user""$proyect")" != ""
    ]; then
    locket=1
  else
    waitLock
  fi
  done
}
```

Código 3.6: Sistema de semáforo para la compilación.

Después de pasar por el semáforo, el flujo se encamina hacia la compilación. Este proceso es lineal por lo que sigue una serie de pasos preestablecidos. Primero se cargan las librerías del sistema, tales como OpenMPI y CUDA, así como el compilador nvcc de CUDA. Después se lanza el comando *make* para empezar el proceso de compilación redirigiendo la salida estándar y la salida de error a un fichero de resultado. Una vez hecho eso, se invoca a un *wait* del sistema de Linux hasta que el proceso de compilación concluya. Cuando este proceso ha concluido, se comprueba si en el fichero resultado hay algún mensaje de error, si se encuentra se elimina el *.lock* y se envía un correo al usuario avisando del error, dando a su vez por finalizado el *script*. Si por el contrario

no hay ningún error, se busca en la carpeta del *Makefile* el fichero ejecutable generado y se almacena en una variable, ya que será necesario más tarde para la creación del *run.sh*. A continuación se eliminan los ficheros resultado y todos los ficheros .o generados. Este proceso se muestra en el Código 3.7.

```
#Compile
#set enviromental variables
#LD_lib and mpi
export PATH=$PATH:/opt/gridengine/lib/linux-x64:/opt/openmpi/lib:/
    export/library/:/opt/python/lib:/opt/openmpi/bin
# CUDA
export PATH=$PATH:/usr/local/cuda-8.0/include/:/usr/local/cuda-8.0/
    bin/
#Get bin name
  make &> result &
#waits until make getiing the PID finish
  wait "$(ps | grep 'make' |awk '{print $1}')"
#Check errors makefile
  if [ "$(cat result | grep error)" == "" ] && [ "$(cat result |
    grep Error)" == "" ]; then
     #filename=$(gawk -F' ' '{ print $3 }' ./result | grep -v  "\-o")
     filename=$(find . -exec file {} \; | grep executable |cut -d: -
    f1)
     rm ./result ./*.o -f
  else
     echo "ERROR IN COMPILATION"
     #Send Error
     #eliminate lock
     rm /export/home/.lock
  exit
  fi
#elimitate the lock
```

Código 3.7: Compilación del proyecto.

Una vez se dispone del proyecto compilado y listo para su ejecución se procede a crear el archivo *run.sh* que será el que finalmente SGE ejecutará. Antes de la creación de dicho archivo se comprueba si existe el archivo *param.txt*, el cual es el encargado de contener los parámetros de lanzamiento de la aplicación compilada. Si dispone de una línea de parámetros se creará el fichero *run.sh* con los parámetros proporcionados. Si por el contrario el archivo contiene más de una línea de parámetros, se procederá a crear un fichero *run.sh* por cada línea que contenga *param.txt*. Este proceso está codificado en el Código 3.8.

```
# check if param.txt exits
if [ -a /export/home/"$user"/projects/"$proyect"/param.txt ]; then
  # file exists count number of lines
```

```
  lines=$(cat /export/home/"$user"/projects/"$proyect"/param.txt |
    wc -l)
  # loop for all param lines
  for ((c=1; c<=lines; c++));
  do
    # call to create run.sh file
    param=$(sed "$c q;d" /export/home/"$user"/projects/"$proyect"/
    param.txt)
    createRunfile "$param" "run_$c.sh"
    #encole job
    sudo /export/scripts/compile/encoleSGEJob.sh "$user" "$proyect"
    "run_$c.sh"
  done
else
  # call to create run.sh file without param
  createRunfile "" "run.sh"
  #encole job
  sudo /export/scripts/compile/encoleSGEJob.sh "$user" "$proyect" "
    run.sh"
fi
```

Código 3.8: Comprobación de param.txt para creación de run.sh.

Tanto tengan parámetros o no se llama a la función createRunfile. Esta función recopila todos los datos proporcionados por parámetro además del nombre del fichero a ejecutar para crear el archivo *run.sh*. En la parte superior del fichero se indican los parámetros de configuración de Sun Grid Engine (los cuales dependen de los datos proporcionados al *script*). Adicionalmente, se añade el ejecutable del proyecto con los parámetros si son necesarios. Además, se añaden al *run.sh* una serie de comandos para almacenar en ficheros predefinidos la fecha y hora de inicio, finalización del proyecto así como la fecha y hora a la que acaba el proyecto en milisegundos. Estos ficheros serán necesarios para otros *scripts* que se encargarán de informar al usuario cuando su proyecto ha comenzado, para cuando el proyecto haya finalizado o ya este disponible para descargar, después de que se comprima y se migre al almacenamiento final del proyecto (ver Código 3.9).

```
  namefile=$2
  #Create run.sh

  echo "#!/bin/bash" > "$namefile"
  #Add default config for SGE

  echo "#$ -V" >> "$namefile"
  echo "#$ -N $user$proyect" >> "$namefile"
  echo "#$ -wd /export/home/$user/projects/$proyect/" >> "$namefile"
  echo "#$ -j y" >> "$namefile"
  echo "#$ -S /bin/bash" >> "$namefile"
  echo "#$ -p $priority" >> "$namefile"
```

```
  #Switch between mpi or other

  if [ "$typep" == "mpi" ]; then
    echo "#$ -pe orte $nodes" >> "$namefile"
  fi

  #Switch between cpu or gpu

  if [ "$type" == "gpu" ]; then
    echo "#$ -q gpu.q" >> "$namefile"
    echo "#$ -hard -l gpu=1" >> "$namefile"
  fi

  #if is mpi change script to run other default
  #Add starfile for know when starts the job

  echo "echo \$(date +%k_%M_%m_%d_%y) > start " >> "$namefile"

  # add to the top of the exit file the pararm options

  echo "echo \"Param options: $1\"" >> "$namefile"

  #check types
   if [ "$typep" == "mpi" ]; then
    echo "mpirun -np $nodes  $filename $1" >> "$namefile"
  else
                echo "$filename $1" >> "$namefile"

  fi

  #Add end file for know when finish

  echo "echo \$(date +%k_%M_%m_%d_%y) > finish " >> "$namefile"

  #Add time in milisecods for use of the sha256

  echo "echo \$(date +%s%N) > end " >> "$namefile"
  chown "$user":projects ./"$filename"
  chown "$user":projects ./"$namefile"
  chmod 775 ./"$namefile"

}
```

Código 3.9: Creación de run.sh.

Para finalizar el *script* llama a *encoleSGEJob.sh* con permisos de sudo para poder encolar el trabajo. En la Sección 3.4.2 comentaremos este *script* y sus pormenores en detalle. Todo el proceso descrito anteriormente se encuentra esquematizado en la Figura 3.6.

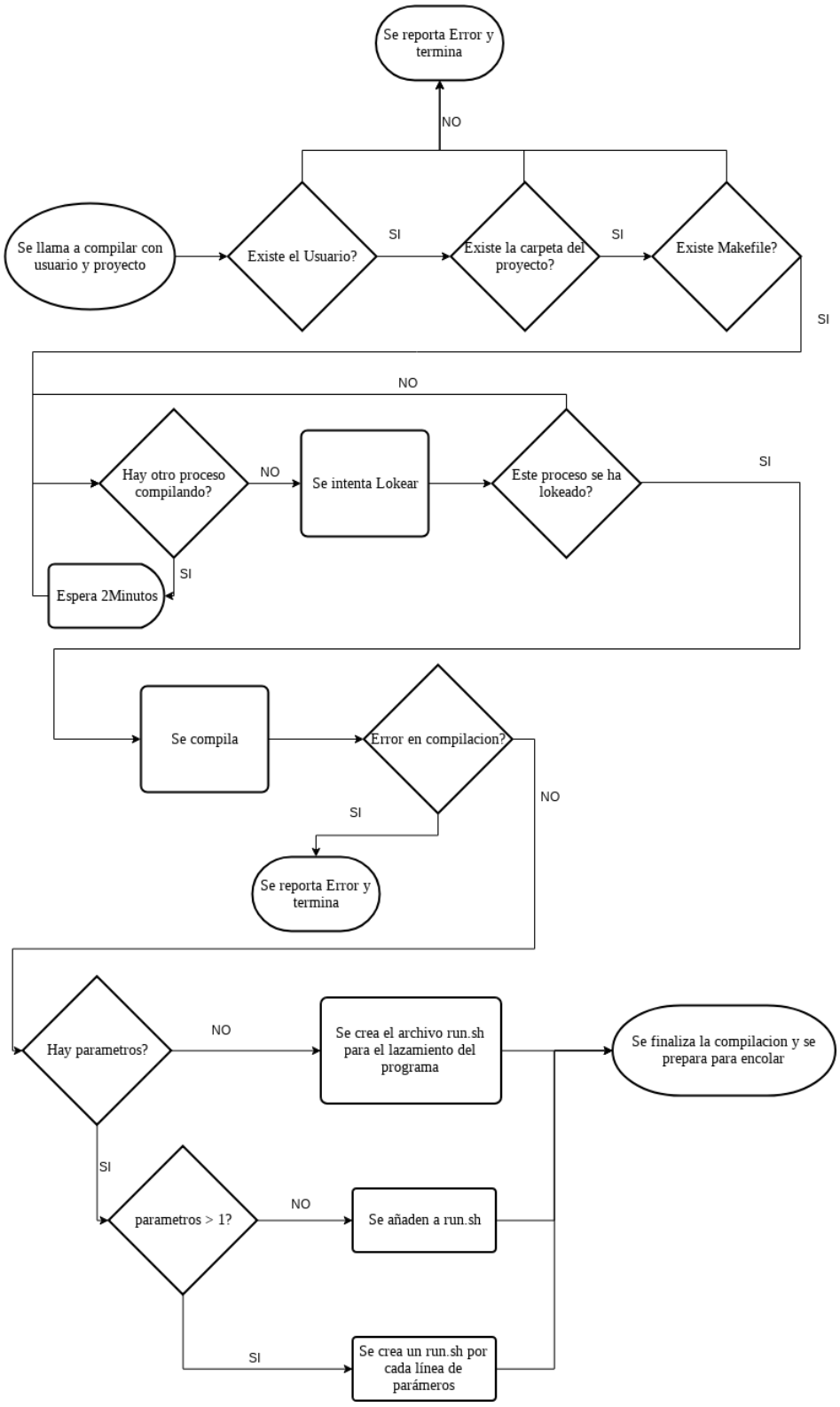



Figura 3.6: Diagrama de flujo del proceso de compilación

### 3.4.2. Script de Lanzamiento de Trabajo

Cuando el trabajo del usuario está ya compilado y los *run.sh* listos, la última parte consiste en lanzarlo a Sun Grid Engine para que sea él quien gestione las colas, la prioridad y la asignación de recursos. Como se comentó con anterioridad en la parte de Sun Grid Engine 2.3.3.2, es necesario llamar al comando *qsub* para encolar el trabajo. No obstante, para que el proceso pueda acceder a los archivos el trabajo que se alojan en la carpeta del usuario, es necesario que ese comando se lance desde la shell del propio usuario que solicita el proyecto. Dado que la cadena de procesos la inicia la API, el usuario que lanza los *scripts* es el usuario apache. Por defecto este usuario tiene pocos permisos por motivos de seguridad por lo que cualquier permiso concedido tiene que estar controlado. Si queremos que el usuario apache pueda lanzar comandos como otro usuario (tal y como se muestra en el Código 3.10), este tendría que tener permisos de sudo para poder ejecutar un comando similar a este.

```
su - $user -c "/opt/gridengine/bin/linux-x64/qsub /export/home/$user
    /projects/$proyect/$filename"
```

Código 3.10: lanzar un comando como otro usuario.

Esto representaría una gran brecha de seguridad, dado que cualquier atacante con acceso al usuario apache tendría control absoluto del sistema. Siendo como es apache un servicio que recibe llamadas desde el exterior esto no puede aplicarse. Por ello, en el *sudoers file* de CentOS tenemos que añadir el Código 3.11 para autorizar a apache pueda lanzar ciertos comandos como sudo sin contraseña.

```
## Rocks alias specification
Cmnd_Alias ROCKS = /opt/rocks/bin/rocks
## User add for apache specification
Cmnd_Alias USERADD = /usr/sbin/useradd
## Script for launch job as an user
Cmnd_Alias QSUBUSER = /export/scripts/compile/encoleSGEJob.sh

apache ALL=(%projects) NOPASSWD: USERADD, ROCKS, QSUBUSER
```

Código 3.11: Autorización a apache de comandos sudo

En concreto se le limita a lanzar tres comandos: lanzar comando para añadir usuarios, lanzar comandos de Rocks para sincronización y el comando para encolar un trabajo. Con ello, aunque tengamos un usuario sin contraseña con permisos de sudo, limitamos las acciones que es capaz de realizar. El código completo del *script* para lanzar comando se encuentra en el Anexo B.

### 3.4.3. Script de Control del Sistema

Una vez nuestro proyecto está encolado, es necesaria una forma de detectar cuándo ha comenzado a procesarse y cuándo ha finalizado, tanto para que el sistema sea consciente como para informar del estado al usuario. Para realizar esta función usaremos un *script* llamado watchDog que se ejecuta cada cinco minutos encargado detectar esos cambios en los trabajos.

Cuando el *cron* del sistema operativo llama a este *script*, recorre todos los ficheros de proyectos de los usuarios extrayendo datos como el usuario, proyecto o el email. Una vez que dispone de estos datos comprueba si en las carpetas de proyecto existe un fichero *start*. Este fichero es uno de los que crea el *run.sh script* para poder controlar el estado de ejecución del trabajo. Al encontrarlo, el *script* manda un correo electrónico utilizando el sistema de correos de la EPSA para informar al usuario de que su proyecto ha comenzado. El formato y envío del email corre a cargo de los servidores de correo de la EPSA.

Además de controlar el inicio de un trabajo la parte más importante es la detección de la finalización de un proyecto. Para ello, de la misma forma que al inicio, el *script* busca un archivo llamado *finish* para detectar la finalización del trabajo. Al detectar este fichero el trabajo ha finalizado, por lo que se dispone a comprimir los resultados de la ejecución. Para ello inserta todos los ficheros del usuario en un *TGZ* a excepción de los archivos utilizados para el control de la aplicación (*start, finish, ...*). Además del archivo *finish*, el *script* de ejecución del trabajo crea un último fichero nombrado como *end* el cual contiene la fecha en la que concluyó el trabajo en milisegundos, esto se utiliza para generar el nombre del archivo *TGZ*.

El nombre del *TGZ* se crea utilizando el nombre del usuario y el nombre del proyecto concatenado con la fecha en milisegundos con la que finalizó el proyecto. Con toda esa cadena generamos un sha256 que será el nombre del fichero. Esto se representa en el *script* tal y como se muestra en el Código 3.12.

```
    #Calculate Hash
      resu=$(echo -n $user$project$(cat $f/end)|sha256sum)
    #Eliminate - and the two spaces
      resu=${resu%???}
    #tar -czf $f/$resu.tar.gz -C $f .
    #Create Tar with all data in the folder
      tar --exclude="start.send*" --exclude="finish.send*" --
exclude="end*" --exclude="run.sh*" --exclude="datos.txt*"  -czf
$f/$resu.tar.gz -C $f .
```

Código 3.12: Función de creación del TGZ.

Una vez creado se copia el fichero en el servidor de archivos de la EPSA y

se informa al usuario de que su proyecto ha finalizado y está disponible para ser descargado, proporcionándole la URL de descarga del mismo.

La razón de la creación del nombre del *TGZ* con esa codificación tan peculiar no es otra que por la necesidad de poder crear un nombre único para el fichero y que sea difícil poder conocerlo. Esto es necesario ya que el servidor de archivos de la EPSA no comprueba qué usuarios pueden descargar qué ficheros, por ello la generación de ese nombre nos garantiza que solo el usuario que reciba el correo es capaz de descargar el archivo. El código completo del *script* así como de su archivo de configuración se encuentran en el Anexo B.

### 3.4.4. Script de Limpieza de Trabajos

Además de comprobar si un trabajo ha comenzado o concluido, es importante la gestión del espacio. En nuestro clúster la capacidad de almacenamiento de trabajos es baja. Al ser un sistema abierto a un público relativamente general se espera un tránsito de ficheros considerable. Es por ello que fue necesario implementar un *script* para la gestión de la eliminación de los trabajos, tanto la carpeta de proyectos como los comprimidos. El *script* realiza dos pasadas: la primera por todas las carpetas de los proyectos y la segunda por la carpeta montada con los proyectos en *TGZ*. El *script* realiza la misma acción en cada una de ellas. Primero localiza los ficheros y extrae la fecha de modificación de los mismos. A continuación la compara con el parámetro del tiempo máximo y si lo supera borra el proyecto en cuestión. Este comportamiento queda reflejado en el Código 3.13.E l código completo se encuentra disponible en el Anexo B.

```
maxTime=1209600
maxTimeTGZ=1209600
#see all projects in ordis
for f in /export/home/*/projects/* ;do
        #extract data for time actual and less from last modfy of
    file datos.txt
        if [ -r $f/datos.txt ]; then
                time=$(($(date +%s) - $(stat  -c %Y $f/datos.txt)))
                #echo "$time"
                #if the time is more or equal to two weeks delete
    folder
                if [ "$time" -ge "$maxTime" ]; then
                        echo "borrado $f"
                        rm -rf $f
                fi
        fi
done
```

Código 3.13: Proceso de borrado de proyectos antiguos

# Capítulo 4

# Sistema: Back End

En este capítulo se describen los cambios realizados sobre el sistema operativo base Rocks 6.2 para que fuera operativo en el laboratorio L14 que y que fuese posible integrarlo con los sistemas de la EPSA.

Como ya se comentó en la Sección 2.3.3, el sistema operativo elegido fue Rocks 6.2 dada su orientación a la creación de clústers. Aunque *off the shelf* Rocks tiene la capacidad de crear clústers con solo las configuraciones mínimas del mismo, estas no eran válidas por el entorno en el que iba a ser desplegado. Nuestro caso es particular, necesitamos que el sistema sea capaz de funcionar tanto como una clase normal los días lectivos, y como un clúster los días no lectivos. Por ello, era necesario cambiar aspectos de funcionamiento, desde el sistema de instalaciones automáticas, como los sistemas de monitorización y gestión del propio clúster.

A lo largo de este capítulo tratamos por una parte, en la Sección 4.1, el sistema de instalaciones automáticas. Por otro lado, en la Sección 4.2 describimos el sistema de gestión y monitorización. Por último, en la Sección 4.3, tratamos los diversos scripts programados para el funcionamiento y mantenimiento del clúster.

## 4.1. Sistema de Instalaciones automáticas

Como se describe en el Anexo A, la instalación del nodo maestro no requiere ningún cambio importante, en cambio el resto de nodos necesitan una configuración más específica. Rocks utiliza un sistema de instalación mediante PXE y TFTP para la instalación de los nodos de cálculo. Por defecto, en el momento en el que instala los nodos, Rocks instala también su propio sistema de arranque y fuerza a que dicha instalación sea la única. No obstante, para que el sistema funcione con los sistemas operativos que

ya existen en el laboratorio L14, es necesario cambiar la forma en la que se instalan los nodos mediante el sistema de PXE.

Antes de ello necesitamos conocer, en detalle, los sistemas operativos presentes en los computadores del L14. En sus ordenadores están instalados los sistemas operativos Windows 10 y Ubuntu 16.04, ambos gestionados por un sistema de arranque LiLo. Estos dos sistemas se encuentran alojados en el primer disco duro de 500 GB que disponen los equipos. Como el sistema que proponemos debe ser capaz de funcionar respetando los sistemas ya creados, utilizamos el segundo disco duro del que disponen los equipos para instalar el sistema operativo y así aislar los ya instalados con el nuestro. Además, es necesario que los nodos de Rocks utilicen LiLo en vez del sistema de arranque por defecto, GNU GRUB.

Con esto en mente, procedemos a modificar el XML que utiliza el PXE para la instalación. Este XML se encuentra en la ruta */export/rocks/install/site-profiles/6.2/nodes/* con el nombre *replace-partition.xml*. En él indicaremos los cambios necesarios para que los nodos se instalen con nuestras necesidades específicas. La primera parte es indicar al instalador la ruta en la que instalar el sistema operativo. Para ello procederemos a indicarlo tal y como se muestra en el Código 4.1.

```
    echo "
    partition / --onpart=/dev/sdb4
    partition swap --onpart=/dev/sda3"&gt; /tmp/
user_partition_info
```

Código 4.1: Especificación del disco de instalación PXE.

En estas líneas definimos que no queremos realizar ningún cambio en el disco duro, después indicamos que queremos que la raíz del sistema / se instale en el disco 2 partición 4. Por último, definimos que nuestra partición de SWAP está en el primer disco partición 3. Con estos cambios permitimos que nuestra instalación sea compatible con las instalaciones ya existentes en el laboratorio L14.

Dado que utilizaremos el cargador LiLo, es necesario que el archivo de imagen del kernel esté en el directorio de archivos de arranque. Para ello, se extrae el archivo en cuestión se extrae de uno de los nodos y se coloca en el sistema de archivos de LiLo. La instalación de esta imagen en cada uno de los ordenadores del L14 corre a cargo del grupo de técnicos de la EPSA mediante su propio sistema de instalaciones.

Aprovechando que utilizamos el archivo para gestionar la localización en la que se instala nuestro sistema operativo, añadimos para la parte final de la instalación una serie de paquetes que serán necesarios para el lanzamiento de pruebas. Estos se dividen en dos grupos principalmente: (1) Los orientados

a librerías y compilación como g++, gcc, Fortran, etc.. y (2) los orientados al funcionamiento de las tarjetas gráficas, driver, CUDA, etc.

La instalación de la paquetería de librerías y compilación no representa ningún problema, solo es necesaria la instalación de los propios paquetes y sus dependencias. En cambio, toda la paquetería orientada a las tarjetas gráficas necesita una atención especial. En primer lugar, es necesario instalar una serie de librerías para cubrir las dependencias de CUDA. Después de ello, se debe instalar el driver de las tarjetas gráficas. En este sentido, CentOS instala por defecto el driver *noveau* (la versión libre para los drivers de las tarjetas gráficas NVIDIA). Por ello, es necesario deshabilitarlo para instalar el driver propietario, esto se realiza después de la instalación del sistema tal y como se indica en el Código 4.2.

```
<!-- desactivamos nouveau el driver de video en centos -->
echo 'blacklist nouveau' >> /etc/modprobe.d/blacklist.conf
mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.
    bak
dracut -v /boot/initramfs-$(uname -r).img $(uname -r)
<!-- instalacion del driver de la GTX 480 -->
./driver480.run -s
```

Código 4.2: Instalación del driver de CUDA.

A continuación se procede a la instalación del driver ya descargado con el parámetro -s para no requerir entrada del usuario. El único problema es que para la instalación de CUDA es necesario que *noveau* esté desactivado por completo, por ello no se puede realizar durante la instalación y hay que realizarlo más tarde, tal y como se muestra en la Sección 4.3.

## 4.2. SGE

El gestor Sun Grid Engine se encuentra preinstalado y listo para usar con sus funciones básicas dentro del propio Rocks. No obstante, para nuestro caso es necesario realizar una serie de cambios para poder operar con las tarjetas gráficas.

El primero de todos es la actualización de la librería OpenMPI, ya que en la versión que trae el sistema operativo por defecto no es compatible con CUDA. Para ello se descargó la ultima versión disponible y se procedió a compilarla en el nodo maestro. Una vez todo el proceso de compilación terminó, se procedió a copiar la librería en todos los nodos de cálculo del clúster.

Uno vez hecho esto, deberemos especificar a SGE que nuestro clúster dispone de otro recurso adicional como es la GPU. Esto se realiza modificando el archivo de configuración de valores complejos de Sun Grid Engine, en el

cual simplemente hay que añadir la variable GPU tal y como se muestra en el Código 4.3.

```
name    shortcut  type    relop requestable consumable
gpu      gpu       BOOL     ==       YES          NO
```

Código 4.3: Añadido de la variable GPU a SGE.

Dado que vamos a tener trabajos heterogéneos tanto solo CPU como CPU-GPU es necesario especificar a SGE que existen dos tipos de colas de para los mismos: (1) la cola de trabajo que únicamente va a reservar un número de CPUs para ejecución exclusiva en procesadores convencionales y (2) la cola que va a reservar un número de GPUs para ejecución CPU-GPU. El problema radica en que para poder utilizar la GPU, necesitamos reservar una CPU también. Por ello, si aplicamos un esquema clásico de colas dispondríamos de 31 procesadores (también denominados *slots*) para la cola CPU y 31 *slots* de CPU-GPU. Esto hecho generaría cierto desaprovechamiento de recursos ya que por ejemplo un trabajo que necesite 60 *slots* no se podría lanzar aunque no haya ningún trabajo utilizando GPUs. Es por ello que al crear las colas de trabajo de Sun Grid Engine se especifico que los trabajos que van a la cola CPU pueden utilizar *slots* de la cola GPU, pero los lanzados en la cola GPU única y exclusivamente pueden utilizar recursos de esa cola. Para ello, se modificó la cola CPU para que tuviera de subordinada a la cola GPU. De esta forma, en caso de que los *slots* de la cola CPU se llenen, comenzará a utilizar recursos de la cola GPU mientras esté disponible. Los archivos de configuración completos se encuentran en el Anexo B.

## 4.3. Scripts

En esta sección se describen los diferentes *scripts* utilizados para el funcionamiento y mantenimiento del clúster.

### 4.3.1. Instalación de CUDA

Para el funcionamiento completo del sistema también fue necesario la implementación de una serie de *scripts* para ejecutar ciertas acciones en el clúster de una forma sencilla. El primero de este repertorio de *scripts* es el encargado de la instalación de CUDA en los nodos del clúster. Esto es necesario ya que como se comentó en el apartado de instalaciones automáticas (ver Sección 4.1) es necesario un reinicio del sistema para poder instalar CUDA. Este *script* consta de dos partes diferenciadas. La primera consiste en la copia de los paquetes de instalación en la carpeta temporal de los nodos de forma

iterativa (ver Código 4.4), esto se realiza así para evitar congestionar la red ya que los paquetes de instalación son pesados. La segunda parte consiste en instalar CUDA de forma paralela en los nodos de cálculo tal y como se muestra en el Código 4.5.

```
rocks iterate host compute "scp -r /export/rocks/install/contrib
    /6.2/x86_64/cudasdk8.0.run %:/tmp"
rocks iterate host compute "scp -r /export/rocks/install/contrib
    /6.2/x86_64/cudasdk8.0_patch.run %:/tmp"
```

Código 4.4: Copia del runfile de CUDA de forma iterativa.

```
rocks run host "/tmp/cudasdk8.0.run --silent --toolkit --driver --
    samples"
rocks run host "/tmp/cudasdk8.0_patch.run --silent --accept-eula"
```

Código 4.5: Instalación de CUDA de forma paralela.

Además de este *script*, para instalar CUDA en todos los nodos del clúster se implementó una variante del mismo que únicame instala el paquete CUDA en la lista de nodos que se le pase por parámetros. Esto fue necesario ya que por cualquier tipo de problemas, uno o más nodos pueden necesitar ser reinstalados y por ello es conveniente instalar CUDA de forma selectiva solo en ese grupo de nodos. Estos dos *scripts* se encuentran de forma completa en el Anexo B.

### 4.3.2. Inicio y apagado del clúster

Una tarea importante del clúster que era necesario implementar fue la gestión del arranque y parada del mismo. Esto es necesario por el hecho de que nuestro clúster va a tener que iniciarse y apagarse en repetidas ocasiones durante su vida útil. Esto difiere de un clúster tradicional en el cual solo se inicia una vez y se apaga por completo cuando se va a retirar del funcionamiento. Es por ello que se implementaron una serie de *scripts* para gestionar el apagado y el encendido del clúster. La gestión del arranque tiene dos partes: la primera parte es la encargada de gestionar el arranque por Wake On Lan y la segunda es la responsable de configurar los nodos para aceptar trabajos.

Durante la parte de arranque de los nodos, el *script* de inicio lanza el comando Wake On Lan a todos los nodos del clúster para comenzar su arranque. Una vez lanzado espera cinco minutos y lanza un comando *dummy* para determinar qué nodos se han despertado. Si algún nodo no ha tenido tiempo de iniciarse o no puede iniciarse por algún motivo, se vuelve a lanzar el Wake On Lan por segunda vez y el *script* espera de nuevo cinco minutos. Al pasar

este tiempo se lanza de nuevo el comando *dummy* para volver a verificar qué nodos están activos. Si sigue habiendo nodos que no se han iniciado, se consideran como nodos caídos del sistema y se apuntan en el log del *script*. Simultáneamente se envía un correo al administrador del clúster para notificar los nodos afectados por la caída para una posible revisión si se repite el fallo de estos nodos a lo largo del tiempo. Esta primera parte de arranque se encuentra reflejada en el Código 4.6.

```
function wait {
  sleep 5m
}

#wake on lan
/export/scripts/start/startNodesAux.sh
#wait 5 minutes
wait
#check nodes
#launche test an save result in ./salida.txt
T1="$(/opt/rocks/bin/rocks run host compute "./" | grep "down" >
    salida.txt)"
#checking up nodes
fail=0
while read p; do
  if [ "$p" != "" ]; then
    fail=$((fail+1))
  fi
done < salida.txt
if [ "$fail"  == "0"  ]; then
  #true
  echo "ok" >> "/export/logs/log$(date +%k_%M_%m_%d_%y).txt"
else
  #false
  #wake on lan again
  /export/home/jaquer/scripts/start/startNodes.sh
  wait
  #check again
  $T1
  #fail to 0
  fail=0
  #read file
  while read p; do
    if [ "$p" != "" ]; then
            fail=$((fail+1))
    fi
  done < salida.txt
  if [ "$fail"  != "0"  ]; then
    #still down
    echo "error" >> "/export/logs/log$(date +%k_%M_%m_%d_%y).txt"
          echo "Nodos Afectados:">> "/export/logs/log$(date +%k_%M_%
   m_%d_%y).txt"
          echo $fail >> "/export/logs/log$(date +%k_%M_%m_%d_%y).txt
```

```
    "
    cat salida.txt >> "/export/logs/log$(date +%k_%M_%m_%d_%y).txt"
    #send email
    cat "/export/logs/log$(date +%k_%M_%m_%d_%y).txt" |mail -s "
    error" "ivanrodriguezferrandez@gmail.com" #change for final
    rm ./salida.txt
  else
    echo "ok" >> "/export/logs/log$(date +%k_%M_%m_%d_%y).txt"
    rm ./salida.txt
  fi

fi
```

Código 4.6: Arranque de los nodos del clúster.

Después de este proceso de inicio de los nodos, se procede a la configuración de los mismos para procesar trabajos. En primer lugar, se exporta la carpeta *home* a todos los nodos del clúster para que dispongan de acceso a los archivos de cada uno de los trabajos. A continuación, se ejecuta un comando en todos los nodos para cambiar las tarjetas a modo *Exclusive Process* con la finalidad de garantizar que solo un proceso puede entrar a la gráfica a la vez. Para finalizar, se desbloquean los procesos de la cola de trabajos. Esto es necesario para evitar que durante el proceso de encendido Sun Grid Engine comience a lanzar trabajos cuando el clúster aún no se ha iniciado y configurado por completo. Todo este proceso se expresa en el Código 4.7.

```
#start NFS
/export/home/jaquer/scripts/start/exportFolder.sh
#change state of grafic card
/opt/rocks/bin/rocks run host compute 'nvidia-smi -c 1' &>/dev/null
#first un hold all jobs
/opt/gridengine/bin/linux-x64/qrls -u "*"
```

Código 4.7: Configuración de los nodos al arranque.

Cuando es necesario apagar el clúster, la primera acción a realizar es el bloqueo de los trabajos que todavía no están en ejecución. Este proceso ocurre media hora antes del apagado del clúster (se detalla en la parte de la gestión del *cron* en la Sección 4.3.3). Después del bloqueo de los trabajos se procede al apagado de los nodos de forma iterativa con el comando *shutdown*, tal y como se muestra en el *script* del Código 4.8.

```
#!/bin/bash
/opt/rocks/bin/rocks run host compute "shutdown -h now"
```

Código 4.8: Apagado de los nodos.

Todos los *scripts* encargados de la gestión del apagado y encendido del clúster se encuentran en el Anexo B.

### 4.3.3. Gestión temporal de los *scripts*

En consecuencia de las características particulares de este clúster, algunos de los *scripts* que se han explicado en este documento deben lanzarse durante períodos de tiempo concretos para el correcto funcionamiento del clúster. Para ello, utilizando la herramienta *cron*, se programaron los diferentes *scripts* que tienen que ser lanzados durante ciertos intervalos. El primero es el encargado de llamar al *script* start.sh cada viernes a las 11:00. El siguiente en la cronología se trata de holdLastWorks.sh a las 11:30 del domingo. Por último, el apagado con shutdownNodes.sh a las 11:55 para gestionar el apagado del clúster (ver Anexo B para ver el código completo).

Además de los *scripts* anteriormente comentados, que se encargan del arranque y parada de los nodos, dentro del *cron* se especifican dos *scripts* que se ejecutan de forma periódica todos los días. El primero es el encargado del control de los proyectos, para determinar cuándo comienzan y cuándo finalizan (ver Sección 3.4.3). Dicho *script* es lanzado cada cinco minutos por el *cron*. El último en la lista del *cron* es el encargado de la limpieza de los trabajos dentro del clúster (ver Sección 3.4.4). En este caso, se lanza todos los días a la 1:00 para limpiar todos los trabajos con más de dos semanas de antigüedad. La configuración completa del *cron* está descrita en el Anexo B.

# Capítulo 5

# Experimentación

Para evaluar el rendimiento del clúster, se plantearon una serie de *benchkmarks* para medir sus prestaciones tanto a nivel de CPU como de GPU. Para ello, se realizaron dos conjuntos de pruebas: (1) pruebas CPU en las cuales se comparó el rendimiento entre Ordis y Euler y (2) pruebas GPU en las que se contrastó el rendimiento entre Ordis y un servidor de computación de altas prestaciones (Clarke). En este capítulo exponemos en primer lugar en la Sección 5.1 las pruebas CPU y seguidamente en la Sección 5.2 las correspondientes a la GPU.

## 5.1. Pruebas CPU

Para la selección de las pruebas del conjunto CPU se decidió, tras una investigación preliminar, utilizar los conjuntos de pruebas más estandarizados posibles tanto para clústers como para supercomputadores. Finalmente, se optó por pruebas que midiesen el rendimiento utilizando librerías paralelas tales como OpenMPI. Con estas precondiciones se barajó el uso de LINPACK, el cual es el más extendido para la medición de la potencia de un clúster (además es requisito indispensable pasar este *benchmark* para poder entrar en la lista del Top 500). Otro *benchmark* dentro de una categoría similar a la de LINPACK es HPCG. Este último fue diseñado para medir de forma más realista el comportamiento de un superordenador bajo carga variada.

Analizando estos *benckmarks* con detenimiento apreciamos que prueban de forma excesivamente genérica el rendimiento de un clúster. Dado que el clúster que proponemos no entra dentro de la categoría de un clúster convencional, consideramos que estas pruebas no podrían medir los puntos fuertes y débiles del sistema. Por ello, tras una investigación más en profundidad, seleccionamos un *benchmark* más adecuado para nuestro propó-

sito: NAS (Numerical Aerodynamic Simulations). Se trata de un conjunto de *benckmarks* creados por la NASA [17], en concreto la división de supercomputación avanzada. Estos *benckmarks* no solo prueban la potencia, como LINPACK o HPCG, sino que además testean determinados puntos críticos, como por ejemplo, la transferencia de información entre los nodos, el acceso a memoria de forma aleatoria o el tiempo de acceso a memoria en nodos lejanos o cercanos. Las ecuaciones y fórmulas han sido extraídas del documento original de la propia NASA que describe y establece dicho *benchmark* [17].

De todos los benckmarks que dispone esta *suite* de la NASA, seleccionamos los siguientes para medir las capacidades CPU del clúster:

- **CG (Conjugate Gradient)**: comunicación y accesos a memoria irregulares.
- **MG (Multi-Grid on a sequence of meshes)**: comunicación de corta y larga distancia, uso intensivo de memoria.
- **FT (Discrete 3D Fast Fourier Transform)**: comunicación todos con todos.
- **LU (Lower-Upper Gauss-Seidel solver)**.

La selección de estos *benckmarks* en concreto responde a su capacidad de medir tanto el rendimiento de red en diferentes niveles como la potencia de cálculo. Estas pruebas disponen de diferentes rangos, cada uno para un tamaño de equipo concreto, estos se subdividen en cuatro grupos:

- **S**: el tamaño más pequeño, diseñado para tests.
- **W**: para workstations (de los años 90).
- **A,B,C**: para pruebas estándar (se multiplica por 4 el tamaño del problema con cada clase).
- **D,E,F**: para grandes pruebas (se multiplica por 16 el tamaño del problema con cada clase).

Cada prueba se realizó utilizando el rango más alto disponible para la misma, siendo D para CG, MG y LU. En cambio, para FT se utilizó el tamaño C dado que este es el tamaño máximo que dispone la *suite* con MPI. Para evaluar el rendimiento de nuestro clúster, realizamos la comparativa con el clúster del IUII, Euler. Para medir el rendimiento de forma comparativa, se realizaron pruebas escalares en las que se incrementa el valor de número de nodos. En cada prueba se abarcó el máximo y mínimo de nodos que se pueden lanzar tanto en Ordis como en Euler. Cada una de las pruebas fue lanzada con los parámetros indicados en la Tabla 5.1.

| Benchmark | Clase | Parámetro | Valor de los parámetros |
|---|---|---|---|
| CG | D | Nº de columnas<br>Nº de noceros<br>Nº de iteraciones<br>eigenvalue shift | 1,500,000<br>21<br>100<br>500 |
| FT | C | Tamaño de grid | 512 x 512 x 512 |
| | | Nº de iteraciones | 20 |
| MG | D | Tamaño de grid<br>Nº de iteraciones | 1024 x 1024 x 1024<br>50 |
| LU | D | Tamaño de grid | 408 x 408 x 408 |
| | | Nº de iteraciones | 300 |
| | | Tiempo entre steps | 1.0 |

Cuadro 5.1: Tabla de valores aplicados a los *benchkmarks*.

En todas las pruebas se extrajeron los siguientes datos: El tiempo absoluto necesario para realizar la prueba, los MFLOPS/s totales y los MFLOPS/s por proceso. Todos los datos generados se encuentran en el anexo C.1

### 5.1.1. MG: Simple 3D Multigrid

MG es una versión simplificada de un multigrid kernel. Este *benchkmark* testea tanto las comunicaciones entre nodos tanto a corta como a larga distancia. El problema a resolver se trata de cuatro iteraciones del algoritmo *V-cycle multigrid* que se explicará a continuación. Esta ecuación se utiliza para obtener una solución aproximada al problema discreto de Poisson:

$$\nabla^2 u = v$$

Dentro de una matriz de $256 \times 256 \times 256$ con *periodic boundary conditions*.

#### 5.1.1.1. Descripción del Algoritmo

Se establece $v = 0$ excepto en los 20 puntos listados en la Tabla 5.2, en los cuales se establece $v = 1$. Estos puntos se determinaron como las ubicaciones de los diez números pseudoaleatorios más grandes y los diez más pequeños. Empezando con la solución iterativa con $u = 0$, cada una de las cuatro iteraciones consisten en los siguientes dos pasos (en los cuales $k = 8 = \log_2(256)$):

$$r = v - Au \quad \text{(Evaluación del valor residual)}$$

$$u = u + M^k r \quad \text{(Aplicación de la corrección)}$$

$M^k$ representa el operador *V-cycle multigrid*, definido en el Algoritmo 1.

| $v_{i,j,k}$k | (i,j,k) | | | | |
|---|---|---|---|---|---|
| -1.0 | 211, 154, 98<br>199, 7, 203 | 102, 138, 112<br>250, 170, 157 | 101, 156, 59<br>82, 184, 255 | 17, 205, 32<br>154, 162, 36 | 92, 63, 205<br>223, 42, 240 |
| +1.0 | 57, 120, 167<br>115, 123, 207 | 5, 118, 175<br>202, 83, 209 | 176, 246, 164<br>203, 18, 198 | 45, 194, 234<br>243, 172, 14 | 212, 7, 248<br>54, 209, 40 |

Cuadro 5.2: Tabla de los puntos con valores diferentes de 0 en la matriz.

**Algoritmo 1** Operador *V-cycle multigrid*

$z_k = M^k r_k$ :
**if** k >1 **then**
  $r_{k-1} = P r_k$ (restricción del residuo)
  $z_{k-1} = M^{k-1} r_{k-1}$ (solución recursiva)
  $z_k = Q z_{k-1}$ (prolongación)
  $r_k = r_k - A z_k$ (evaluación del residuo)
  $z_k = z_k + S r_k$ (aplicación se suavizado)
**else**
  $z_1 = S r_1$. (aplicación de suavizado)
**end if**

En esta definición, $A$ representa el elemento trilineal finito de la discretización laplaciana $\nabla^2$ normalizado como se indica en la Tabla 5.3 en la que los coeficientes de $P, Q$, y $S$ también están listados. En esta tabla, $c_0$ denota el coeficiente central del operador del punto 27, donde estos coeficientes están ordenados como un cubo $3 \times 3 \times 3$. $c_0$ es el coeficiente que multiplica el valor de la matriz $(i, j, k)$, mientras que $c_1$ multiplica los seis valores en la matriz de puntos que difieren exactamente en uno en un solo un índice, $c_2$ multiplica los doce valores siguientes más próximos, los cuales difieren exactamente en uno en dos índices, $c_3$ multiplica los ocho valores localizados en posiciones de la malla que difieren en uno en cada uno de los tres índices. La restricción del operador $P$ dada en el algoritmo 1 es la proyección trilinear del operador para la teoría de los elementos finitos, normalizados de tal forma que los coeficientes de todos los operandos son diferentes en cada nivel.

| C | $c_0$ | $c_1$ | $c_2$ | $c_0$ |
|---|---|---|---|---|
| A | -8.0/3.0 | 0.0 | 1.0/6.0 | 1.0/12.0 |
| P | 1.0/2.0 | 1.0/4.0 | 1.0/8.0 | 1.0/16.0 |
| Q | 1.0 | 1.0/2.0 | 1.0/4.0 | 1.0/8.0 |
| S(a) | -3.0/8.0 | +1.0/32.0 | -1.0/64.0 | 0.0 |
| S(b) | -3.0/17.0 | +1.0/32.0 | -1.0/61.0 | 0.0 |

Cuadro 5.3: Coeficientes para el elemento finito trilineal discretizado.

#### 5.1.1.2. Medición del Tiempo del Algoritmo

La medición del tiempo se inicializa antes de evaluar el valor residual la primera vez y antes de inicializar $u$ y $v$. La finalización de la medición ocurre antes de evaluar el último valor residual, pero después de imprimir su valor.

#### 5.1.1.3. Experimentación

Este *benchmark* es encargado de medir la comunicación entre nodos accediendo tanto a nodos cercanos como a nodos lejanos. Después de las pruebas se obtuvieron los resultados en la Figura 5.1.

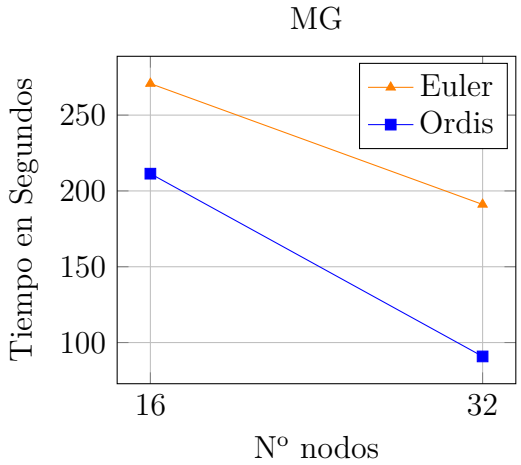



Figura 5.1: Gráfica de las pruebas MG sobre Euler y Ordis que muestra la evolución del tiempo de ejecución en función del número de nodos utilizado.

Como se muestra en el gráfico, con un mismo número de nodos Ordis es capaz de resolver el *benchmark* en menos tiempo que Euler. Dado que se trata comunicaciones con una carga de datos baja (y el número de nodos es un valor pequeño) el tráfico de red es bajo, lo cual beneficia a Ordis dado que

el benchmark se ejecuta de forma exclusiva en él. En cambio, la red de Euler se ve relativamente ralentizada dado su propósito abierto y la presencia de otros trabajos siendo lanzados de forma simultánea en otros nodos que hacen uso de la red.

Es por ello que estas pruebas en las que se mide el inicio de la comunicación más que el intercambio grueso de datos, Ordis gana ventaja sobre Euler. En cambio en pruebas como FT 5.1.3, donde el intercambio de datos es mayor, Euler presenta ventaja sobre Ordis como veremos a continuación.

### 5.1.2. CG: Conjugate Gradient

Este método es utilizado para calcular la aproximación al valor más pequeño propio de una matriz grande, dispersa y simétricamente positiva. Este kernel es típico en redes no estructuradas en las cuales se prueban las comunicaciones entre nodos utilizando multiplicaciones de vectores de matrices no estructuradas.

#### 5.1.2.1. Descripción del Algoritmo

El Algoritmo 2 muestra el pseudocódigo de este método. $A$ denota la matriz dispersa de orden $n$, las letras romanas minúsculas denotan vectores, $x_j$ es el componente $j^{th}$ del vector $x$, y el superíndice $T$ indica el operador transpuesta habitual. Las letras griegas en minúsculas son escalares. Se denota con $||x||$ el operador norma Euclídeo del vector $x$, $||x|| = \sqrt{(x^T x)}$. Todas las cantidades son reales. Los valores para el tamaño del sistema $n$, número de iteraciones *niter*, y el *shift* de $\lambda$ para el tamaño de problema que se usa se encuentran especificados en la Tabla 5.1 en el apartado de *benchmark* CG. La solución de $z$ al sistema lineal de ecuaciones $Az = x$ es aproximada utilizando el método del gradiente conjugado(CG). El cual se implementa con el pseudocódigo 2. El método de la potencia inversa se implementa como el pseudocódigo 3.

#### 5.1.2.2. Medición del Tiempo del Algoritmo

El tiempo medido coincide con el requerido para computar todas las *niter* iteraciones e imprimir los resultados, después de que la matriz generada sea descargada en cada uno de los nodos y después de que se halla procedido a la inicialización del vector inicial $x$.

**Algoritmo 2** Algoritmo del gradiente conjugado.

$z = 0$
$r = x$
$p = r^T r$
$p = r$
$i = 1$
**while** $i > 25$ **do**
  $q = Ap$
  $\alpha = \rho/(\rho^T q)$
  $z = z + \alpha p$
  $\rho_0 = \rho$
  $r = r - \alpha q$
  $\rho = r^T r$
  $\beta = \rho/\rho_0$
  $p = r + \beta p$
  $i = i + 1$
**end while**
calculamos el residuo $||r|| = ||x - Az||$

**Algoritmo 3** Algoritmo de la potencia inversa.

$x = [1, 1, ..., 1]^T$;
(Comenzar con la medición del tiempo)
**while** $it = 1, niter$ **do**
  Resolver el sistema $Az = x$ y devolver $||r||$, de la siguente forma:
  $\zeta = \lambda + 1/(x^T z$
  Imprimo it, $||r||$, y $\zeta$
  $x = z/||z||$
**end while**
(finalizamos la medición del tiempo)

#### 5.1.2.3. Experimentación

Este *benchkmark* se centra de medir la comunicación entre nodos de forma no estructurada, accediendo a posiciones de memoria aleatorias entre nodos lejanos. Después de las pruebas se obtuvieron los resultados que quedan reflejados en la Figura 5.2.

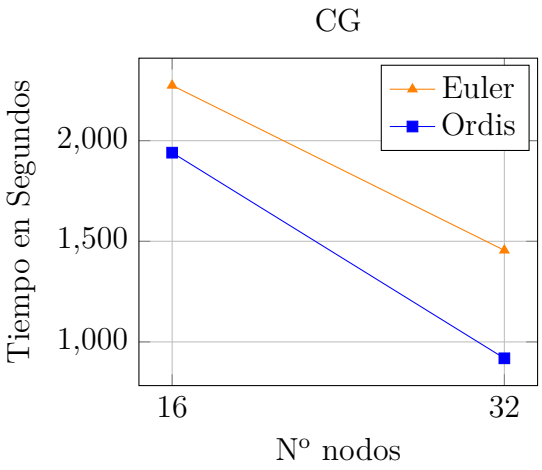



Figura 5.2: Gráfica de las pruebas CG sobre Euler y Ordis que muestra cómo evoluciona el tiempo de ejecución a medida que se incrementa el número de nodos.

Como se muestra en el gráfico, con un mismo número de nodos Ordis es más rápido que Euler. Dado que el benchmark se compone principalmente de comunicaciones con una carga de datos baja (y acceso aleatorio), pequeñas cantidades de datos van a ser accedidas en momentos aleatorios. Como se comentó en la prueba anterior, al ser una red libre de trabajo, Ordis tiene ventaja sobre ese tipo de comunicaciones.

### 5.1.3. FT: 3D Fast Fourier Transform

FT es una solución de una ecuación en derivadas parciales en 3D utilizando la transformada rápida de Fourier (FFT). Este kernel ejecuta la esencia de muchos programas, realizando una gran cantidad de comunicaciones de larga distancia dentro del clúster. Para la resolución de estas ecuaciones se utilizan las FFTs tanto directas como inversas.

#### 5.1.3.1. Descripción del Algoritmo

Considerando la ecuación en derivadas parciales:

$$\frac{\partial(x,t)}{\partial t} = \alpha\nabla^2 u(x,t)$$

En la que $x$ es una posición en un espacio tridimensional. Cuando la transformada de Fourier se aplica a cada lado de la ecuación se obtiene la siguiente expresión:

$$\frac{\partial v(z,t)}{\partial t} = -4\alpha\pi^2|z|^2 v(z,t)$$

Donde $v(z,t)$ es la transformada de Fourier de $u(x,t)$. Esta tiene la solución que se muestra a continuación:

$$v(z,t) = e^{-4\alpha\pi^2|z|^2 t} v(z,0)$$

Ahora se considera la versión discreta de la función de derivadas parciales. Siguiendo los pasos anteriores, se puede resolver calculando la trasformación discreta de Fourier de forma directa del estado original del vector $u(x,0)$, multiplicando los resultados por ciertos exponentes y a continuación aplicando la inversa del algoritmo de la trasformada de Fourier en 3D. Las versiones directa e inversa de la transformada de Fourier de $n_1 \times n_2 \times n_3$ del vector $x$ quedarían definidas como.

$$F_{q,r,s}(u) = \sum_{l=0}^{n3-1}\sum_{k=0}^{n2-1}\sum_{j=0}^{n1-1} u_{j,k,l} e^{-2\pi ijq/n1} e^{-2\pi ikr/n2} e^{-2\pi ils/n3}$$

$$F^{-1}_{q,r,s}(u) = \frac{1}{n_1 n_2 n_3}\sum_{l=0}^{n3-1}\sum_{k=0}^{n2-1}\sum_{j=0}^{n1-1} u_{j,k,l} e^{-2\pi ijq/n1} e^{-2\pi ikr/n2} e^{-2\pi ils/n3}$$

Para el caso concreto FT de tamaño C se aplicaría el Algoritmo 4.

#### 5.1.3.2. Medición del Tiempo del Algoritmo

La medición del tiempo comprende desde la generación de los números aleatorios hasta la impresión del último *checksum*.

#### 5.1.3.3. Experimentación

Este *benchmark* es encargado de medir la comunicación entre nodos de forma conjunta (todos a uno y uno a todos) con grandes cantidades de datos. Después de las pruebas se obtuvieron los resultados reflejados en la Figura 5.3.

**Algoritmo 4** Algoritmo de FT para tamaño C.

Set $n_1 = 512, n_2 = 512, n_3 = 512$
Set $T = 1$
Set $N = 20$
Set $\alpha = 10^{-6}$
Set $Seed = 314159265$
Se crean $2n_1n_2n_3$ elementos con $\rightarrow RandomGenerator64bit(Seed)$.
$U_{j,k,l}, 0 \leq j < n_1, 0 \leq k < n_2, 0 \leq l < n_3$
**while** $T > N$ **do**
  Calculamos la transformada de Fourier directa sobre U $\rightarrow V$
  Calculamos $W_{j,k,l} = e^{-4\alpha\pi^2(\overline{j}^2+\overline{k}^2+\overline{l}^2)t}V_{j,k,l}$
  $\overline{j}$ es $j$ para $0 \leq j < n_1/2$ y $j - n_1$ para $n1/2 \leq j < n_1$.
  $\overline{j}$ y $\overline{k}$ son iguales a $\overline{j}$
  Calculamos la inversa de la trasformada de Fourier sobre $W \rightarrow X$.
  Calculamos el checksum como $\sum j = 0^{1023} X_{q,r,s}$
  Donde $q = j(modn_1)$,$r = 3j(modn_2)$ y $s = 5j(modn_3)$
  Print checksum
  $T = T + 1$
**end while**

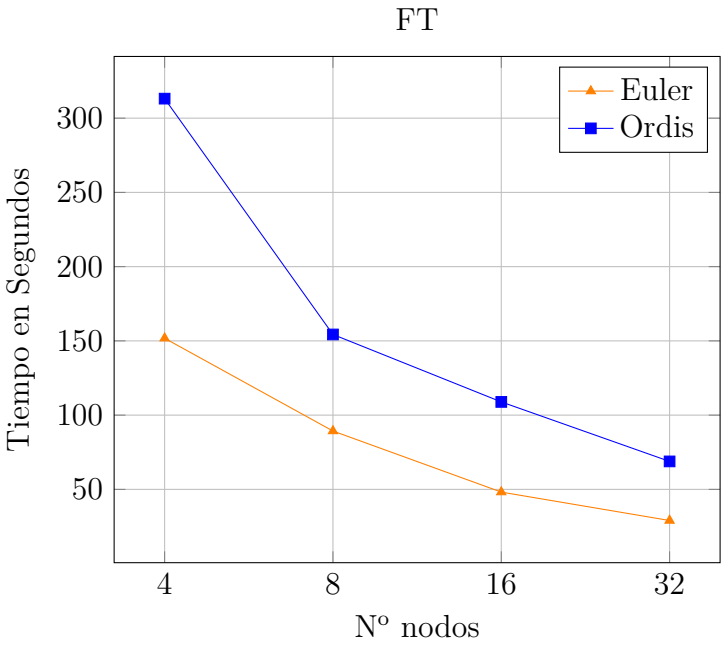



Figura 5.3: Gráfica de las pruebas FT sobre Euler y Ordis.

Como se observa en la gráfica, Euler es mejor que Ordis en todo momento.

A diferencia de las pruebas anteriores, en este caso se requiere un ancho de banda superior que el proporcionado por Ethernet (utilizado por Ordis), lo que hace que Euler (que se comunica mediante Infinitiband) necesite menos tiempo en completar el *benchmark*.

### 5.1.4. LU solver

Este *benchmark* utiliza la factorización *lower upper* o LU, en concreto esta prueba está orientada a medir la potencia de procesamiento de los nodos. Este *benchkmark* es una aplicación completa, a diferencia de los anteriores que eran *kernels* por lo que el objetivo de la prueba es medir el clúster ante un problema real. En concreto para esta factorización LU utiliza el algoritmo *Symmetric successive over-relaxation*.

#### 5.1.4.1. Descripción del Algoritmo

Dado el sistema lineal definido como:

$$[\boldsymbol{K}^n][\Delta \boldsymbol{U}^n] = [\boldsymbol{RHS}^n]$$

donde

$$\boldsymbol{K}^n = \{\boldsymbol{I} - \delta_\tau [D_\xi \boldsymbol{A}^n + D_\xi^2 \boldsymbol{N}^n + D_\eta \boldsymbol{B}^n + D_\eta^2 \boldsymbol{Q}^n + D_\zeta \boldsymbol{C}^n + D_\zeta^2 \boldsymbol{S}^n]\} \Delta \boldsymbol{U}^n, for(i,j,k) \in D_h$$

Se especifica que las incógnitas se ordenen correspondiéndose con las puntos de la matriz de forma que queden ordenados lexicográficamente, tal que el índice en la dimensión $\xi$ sea más rápida en ser accedido, seguida por la dimensión$\eta$ y por último la dimensión $\zeta$. La matriz de discretización de diferencia finita $\boldsymbol{K}$, que resulta de tal orden, tiene una estructura de bandas regular. Hay un total de siete bloques diagonales, cada uno con un tamaño de $(5 \times 5)$. La matriz $\boldsymbol{K}$ se puede escribir como la suma de tres matrices, la matriz $\boldsymbol{D}$, $\boldsymbol{Y}$ y $\boldsymbol{Z}$.

$$\boldsymbol{K}^n = \boldsymbol{D}^n + \boldsymbol{Y}^n + \boldsymbol{Z}^n$$

donde

$\boldsymbol{D}^n$ = Es el bloque principal, la diagonal de $\boldsymbol{K}^n$
$\boldsymbol{Y}^n$ = Son las tres diagonales superiores de $\boldsymbol{K}^n$
$\boldsymbol{Z}^n$ = Son las tres diagonales inferiores de $\boldsymbol{K}^n$

Por lo tanto, $\boldsymbol{D}$ es una matriz de bloques diagonales, mientras que $\boldsymbol{Y}$ y $\boldsymbol{Z}$ representan la triangular superior y la triagular inferior de las matrices. Entonces el algoritmo de *Symmetric successive over-relaxation* sobre un punto se puede denotar como:

$$[\boldsymbol{X}^n][\Delta \boldsymbol{U}^n] = [\boldsymbol{RHS}]$$

donde

$$\boldsymbol{X}^n = \omega(2. - \omega)(\boldsymbol{D}^n + \omega \boldsymbol{Y}^n)(\boldsymbol{D}^n)^{-1}(\boldsymbol{D}^n + \omega \boldsymbol{Z}^n)$$

$$= \omega(2. - \omega)(\boldsymbol{D}^n + \omega \boldsymbol{Y}^n)(\boldsymbol{I} + \omega(\boldsymbol{D}^n)^{-1}\boldsymbol{Z}^n)$$

y $\omega \in (0., 2.)$ que representa la constante de *over-relaxation*. Basándonos en este hecho, el *Symmetric successive over-relaxation* o SSOR se puede implementar de siguiente forma:

**Inicialización:** Se establecen los valores del límite de $\boldsymbol{U}_{i,j,k}$ de acuerdo a como se establece en la siguiente ecuación.
$u^{(m)} = f^{(m)}(\xi, \eta, \zeta)$ para $(\tau, \xi, \eta, \zeta) \in D_\tau \times \partial D$ siendo m = 1,2, ..., 5
Ahora inicializamos los valores de $\boldsymbol{U}^0_{i,j,k}$ de acuerdo a la siguiente función.

$$P^{(m)}_\xi = (1 - \xi)u^{(m)}(0, \eta, \zeta) + \xi u^{(m)}(1, \eta, \zeta)$$
$$P^{(m)}_\eta = (1 - \eta)u^{(m)}(\xi, 0, \zeta) + \eta u^{(m)}(\xi, 1, \zeta)$$
$$P^{(m)}_\zeta = (1 - \zeta)u^{(m)}(\xi, \eta, 0) + \zeta u^{(m)}(\xi, \eta, 1)$$

A continuación se computa el vector de la función de fuerzas,$\boldsymbol{H}^*_{i,j,k}$ para $(i, j, k) \in D_h$.

**Paso 1:** Se calcula $[\boldsymbol{RHS}^n_{i,j,k}]$ para $(i, j, k) \in D_h$.

**Paso 2:** Se forma y se resuelve el siguiente sistema regular, disperso, de la triangular inferior para obtener $[\Delta \boldsymbol{U}_1]$:

$$(D^n + \omega \boldsymbol{Y}^n)[\delta \boldsymbol{U}_1] = [\boldsymbol{RHS}]$$

**Paso 3:** Se forma y se resuelve el siguiente sistema regular, disperso, de la triangular inferior para obtener $[\Delta \boldsymbol{U}^n]$:

$$(\boldsymbol{I} + \omega(\boldsymbol{D}^n)^{-1}\boldsymbol{Z}^n)[\Delta \boldsymbol{U}^n] = [\Delta \boldsymbol{U}_1]$$

**Paso 4:** Actualizamos la solución:

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + [1/\omega(2. - \omega)]\delta \boldsymbol{U}^n$$

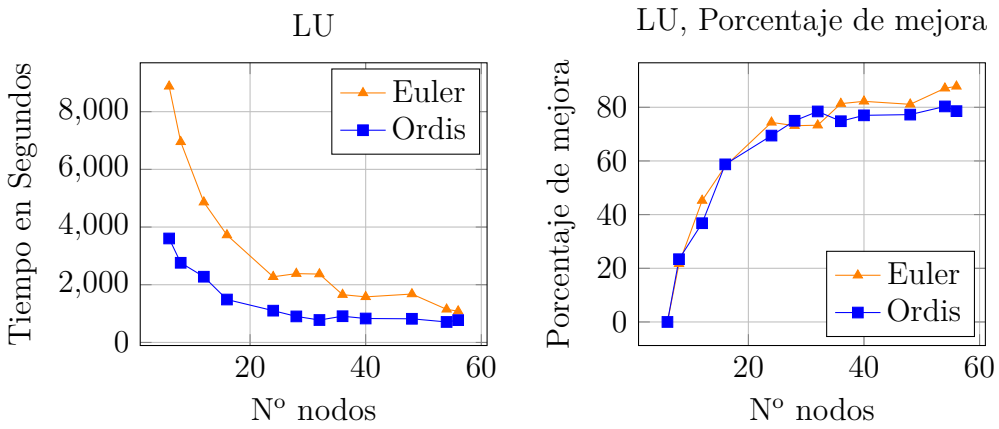
Los pasos del uno al cuatro constituyen una iteración del algoritmo SSOR, por lo que estos pasos se repiten $N$ veces siendo $N$ el parámetro del tamaño de la prueba a realizar. En el caso del *benchmark* de tamaño $D$ serían 300 iteraciones del algoritmo con los siguientes parámetros de entrada:
$N_\xi = 408$; $N_\eta = 408$; $N_\zeta = 408$; $\Delta\tau = 1{,}0$; $\omega = 1{,}2$

#### 5.1.4.2. Medición del Tiempo del Algoritmo

Para la medición del algoritmo. El contador comienza cuando se inicializa la matriz y mide el tiempo que transcurre hasta que se realizan $N$ veces los pasos del uno al cuarto.

#### 5.1.4.3. Experimentación

Este *benchmark* mide la potencia del clúster con una aplicación completa, es decir la potencia de cálculo pura del clúster. Después de las pruebas se obtuvieron los resultados que se muestran en los gráficos de la Figura 5.4.




(a) Lanzamiento de LU (b) Porcentaje de mejora

Figura 5.4: Gráfica de pruebas LU sobre Euler y Ordis 5.4a y Gráfica de porcentaje de mejora con respecto a la ejecución con 6 nodos 5.4b.

Como se puede observar, Ordis es más rápido que Euler en esta prueba (ver Figura 5.4a). Esto es debido a que Ordis presenta unas CPUs más rápidas que Euler. Además, como se denota en el gráfico, la diferencia entre ambos se reduce cada vez más hasta que Ordis y Euler tienen una diferencia marginal. Esto es debido a que cuando comienzan las pruebas la red de Ordis puede manejar el paso de información entre diferentes nodos. Si visualizamos los porcentajes de mejora en una gráfica (ver Figura 5.4b), se puede comprobar

que con 24 nodos Ordis se estanca con la mejora. Esto es debido a que en ese momento llegamos al punto de saturación de la red Ethernet de Ordis por lo que la mejora apenas cambia incluso aumentando el número de nodos. En cambio Euler sigue incrementado su porcentaje de mejora dado que utiliza la red Infinitiband con un mayor ancho de banda.

## 5.2. Pruebas GPU

Una de las partes principales dentro del clúster que se ha desplegado es la computación sobre GPUs, como se ha comentado con anterioridad, Ordis dispone de 31 tarjetas gráficas GTX 480 (ver Figura 1.3). Por este motivo, medir el rendimiento de las mismas dentro del clúster se erige como una parte fundamental de la experimentación en este trabajo. Para ello se precisa de un *benchmark* capaz de medir la potencia de las tarjetas gráficas que disponemos lanzando cálculos sobre las mismas. Tras un análisis en profundidad, seleccionamos una prueba implementada opr Jiri Kraus (Senior Developer de NVIDIA) para la resolución de la ecuación de Poisson utilizando el método de Jacobi como un ejemplo de *CUDA-aware MPI*[18], es decir, CUDA combinado con MPI. Con este experimento pretendemos por una parte medir el rendimiento del clúster que utiliza hardware gráfico común en su construcción como compararlo con Clarke, un servidor de altas prestaciones de un único nodo con tarjetas de gama profesional de última generación. Las imágenes y ecuaciones han sido extraídas del articulo original [18]. Todos los datos extraídos se encuentran en el anexo C.2

### 5.2.1. Descripción del Algoritmo

Este *benchmark* utiliza el algoritmo de Jacobi para la resolución de la ecuación de Poisson en un rectángulo con *Dirichlet boundary conditions* (ver Figura 5.5). Jacobi es un método iterativo utilizado para resolver sistemas de ecuaciones lineales, la idea de este método es construir una solución convergente definida de forma iterativa, donde el límite de la función representa la solución de la ecuación. Para este *benchmark* utilizaremos este método para resolver el problema de Dirichlet, el cual se fundamenta en encontrar una función armónica sobre un dominio $\Omega$. Esto se utiliza en el ámbito científico, por ejemplo, para calcular el campo magnético que genera una cavidad metálica con potencial constante.

$$\Delta u(x,y) = 0 \forall (x,y) \in \Omega / \delta\Omega$$
$$u(x,y) = f(x,y) \in \delta\Omega$$

Según esta formulación, el problema de Dirichlet se puede convertir en un problema de Possion dadas unas condiciones particulares. Bajo esas condiciones, podemos aplicar Jacobi para solventar un problema de Dirichlet convertido a problema de Poisson.

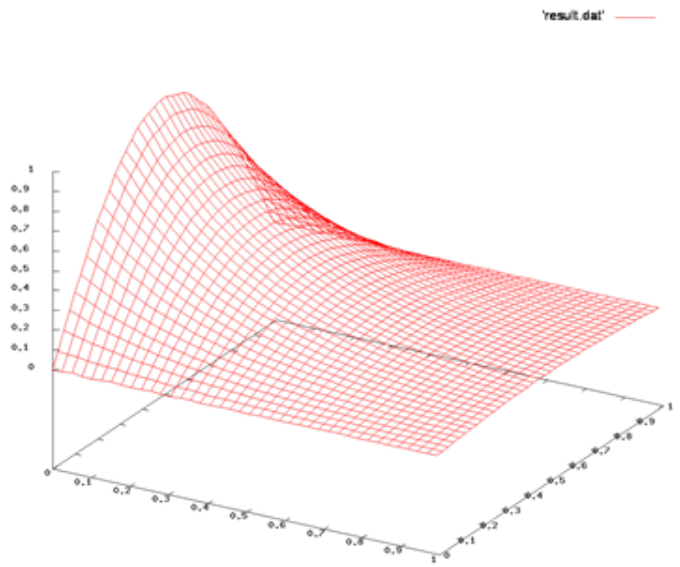



Figura 5.5: Representación de la solución de la ecuación de Poisson con *boundary conditions*.

Primero es necesario aproximar el operador laplaciano $\Delta$, para ello utilizamos diferencias centrales de segundo orden para nuestra matriz concreta. El kernel de Jacobi que utilizamos aplica el Algoritmo 5 a cada uno de los puntos de nuestro domino.

**Algoritmo 5** Algoritmo de aproximación de Laplacian para Jacobi, Jacobi-Kernel

A $\rightarrow$ matriz de puntos actual
Anew $\rightarrow$ matriz de puntos siguiente
$Anew[i][j] = 0{,}25 \times (A[j][i+1] + A[j-1][i] + A[j+1][i] + A[j][i-1])$

Para poder paralelizar el problema aplicamos una descomposición 2D del domino con $n \times k$ subdominios (ver Figura 5.6). Se utiliza una descomposición 2D para reducir la cantidad de datos transferidos entre procesos. Si lo realizásemos con una descomposición 1D, la cantidad de datos a transferir

se vería incrementada sobremanera con lo que el tiempo de comunicación excedería el tiempo de cálculo.

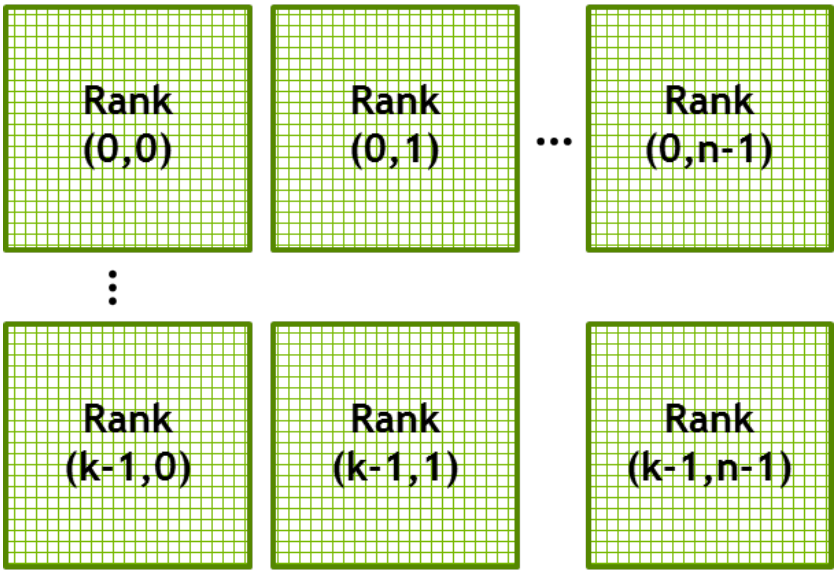



Figura 5.6: Representación de la descomposición del domino para paralelizar el problema.

Cuando aplicamos esta paralelización es necesario intercambiar datos entre las GPUs para que de forma cooperativa resuelvan el problema. Por ello, es necesario crear un *halo* para cada proceso (ver Figura 5.7) que es actualizado cada iteración. De forma resumida, el algoritmo que resuelve el problema de Poisson aplicando el método iterativo de Jacobi puede ser expresado con el pseudocódigo del Algoritmo 6.

**Algoritmo 6** Algoritmo para aplicar Jacobi para resolver la ecuación de Poisson.

```
while iterations <1000 and residue >1.0E-5F do
    residue = CallJacobiKernel
    SwapSolution();                    ▷ Intercambiamos A por Anew
    TransferAllHalos();                ▷ enviamos y recibimos los halos
    if iterations % 100 == 0 then
        Print residue y iterations
    end if
    iterations = iterations + 1
end while
```

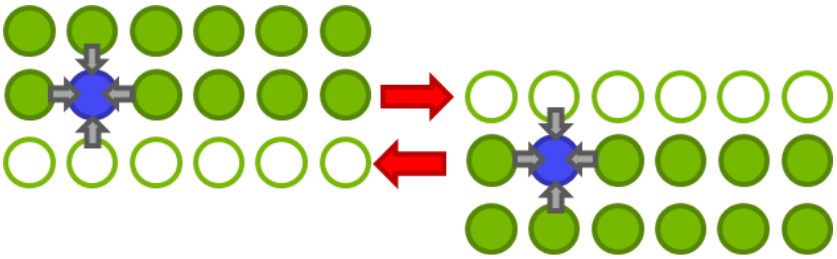


Figura 5.7: Representación del intercambio de información entre tarjetas en los límites del dominio.

### 5.2.2. Medición del Tiempo del Algoritmo

Para la medición del tiempo utilizamos la propia API de MPI, la cual proporciona el tiempo transcurrido con extrema exactitud para programas que utilizan la librería, en concreto se utiliza la función *Wtime*. La medición comienza cuando se inicializan las variables de MPI y no termina hasta que el ultimo nodo devuelve el resultado y se imprime. Para cerciorarnos de que todos los nodos han terminado, creamos una barrera para sincronizar los nodos y no avanzar en el programa hasta que la última llamada se ha completado, para ello utilizamos la función de la librería MPI *Barrier*.

### 5.2.3. Experimentación

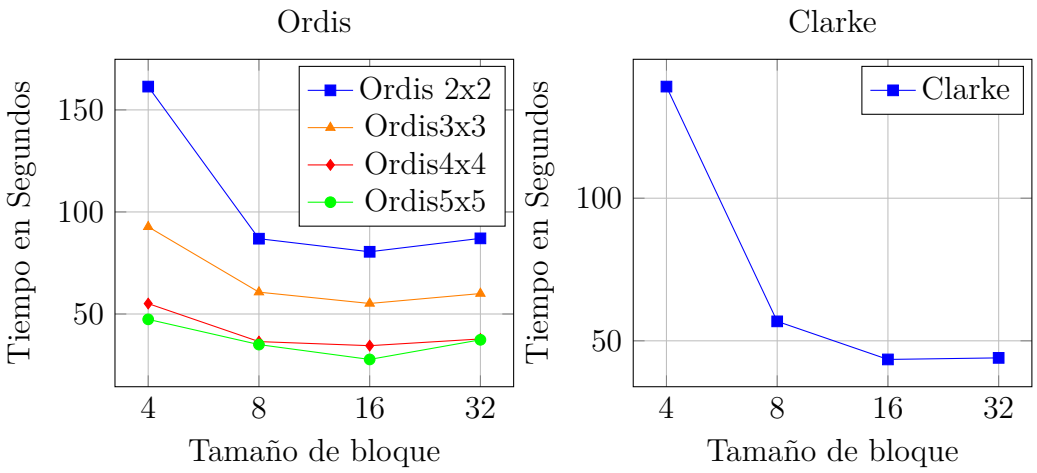
Para realizar las pruebas GPU, además de simplemente medir el rendimiento de Ordis, se utilizó Clarke como supercomputador para la comparativa. Dado que Clarke solo dispone de dos tarjetas gráficas, es necesario definir de antemano el tamaño del problema a resolver. En nuestro caso, se utilizó un tamaño de $18432 \times 18432$, dado que dividido entre cuatro tarjetas el tamaño del problema queda reducido a $9216 \times 9216$ (el tamaño máximo admitido por las tarjetas gráficas instaladas en Ordis). Como el tamaño del problema a resolver es fijo, la prueba de escalabilidad ha consistido en incrementar el número de nodos en Ordis manteniendo el tamaño global. El objetivo era determinar el punto de corte entre los dos sistemas, es decir, cuántos nodos/tarjetas de Ordis son necesarias para igualar la potencia de Clarke resolviendo el mismo problema.

Sabiendo que el número máximo de tarjetas gráficas que dispone Ordis es de 31, se definió como máximo para las pruebas 25 nodos, que construyen una matriz de $5 \times 5$ tarjetas, la matriz cuadrada más grande que se puede crear en Ordis disponiendo de solo 31 tarjetas gráficas. Sabiendo el máximo número de tarjetas disponibles y el mínimo con el que van a realizar las pruebas, se

determinaron los puntos en los cuales se van a tomar datos comparar como se comporta el *benchmark* con tamaños incrementales de nodos, siempre siendo estos matrices cuadradas. En consecuencia los puntos donde se van a ejecutar el *benchmark* quedan definidos de la siguiente manera.

- Matriz de $2 \times 2$ usando 4 nodos con tamaño por tarjeta 9216
- Matriz de $3 \times 3$ usando 9 nodos con tamaño por tarjeta 6144
- Matriz de $4 \times 4$ usando 16 nodos con tamaño por tarjeta 4608
- Matriz de $5 \times 5$ usando 25 nodos con tamaño por tarjeta 3686

Antes de lanzar el *benchmark* sobre los dos sistemas a probar, es también necesario determinar el tamaño de bloque óptimo para cada GPU en nuestro problema (con el objetivo de obtener una comparativa justa e igualada entre Ordis y Clarke). Para ello, se ejecutaron las pruebas con los tamaños de bloque 4, 8, 16 y 32 tanto sobre Ordis como sobre Clarke. Como se puede observar en la Figura 5.8, el tamaño óptimo es 16 en ambos sistemas por lo que las pruebas de escalabilidad se realizarán con dichas dimensiones.




(a) (b)

Figura 5.8: Gráfica de pruebas GPU con diferentes tamaños de bloque en Ordis 5.8a y Clarke 5.8b. En ambos casos se muestra la evolución del tiempo de ejecución al variar el tamaño de bloque.

Teniendo en cuenta este parámetro determinado empíricamente, al realizar las pruebas de escalabilidad con el tamaño de bloque 16 se obtuvieron los resultados mostrados en la Figura 5.9 sobre el tiempo de ejecución del *benchmark*, además de los GFLOPS medidos en la misma. En dichas gráficas, la línea horizontal representa las pruebas realizadas en Clarke (en las cuales

se mantuvo siempre el mismo número de nodos, ya que solamente se dispone de uno). En cambio en Ordis se fue incrementando el tamaño de una matriz de 2x2 GPUS, hasta 5x5, incrementando el ancho y el alto en 1 en cada paso.

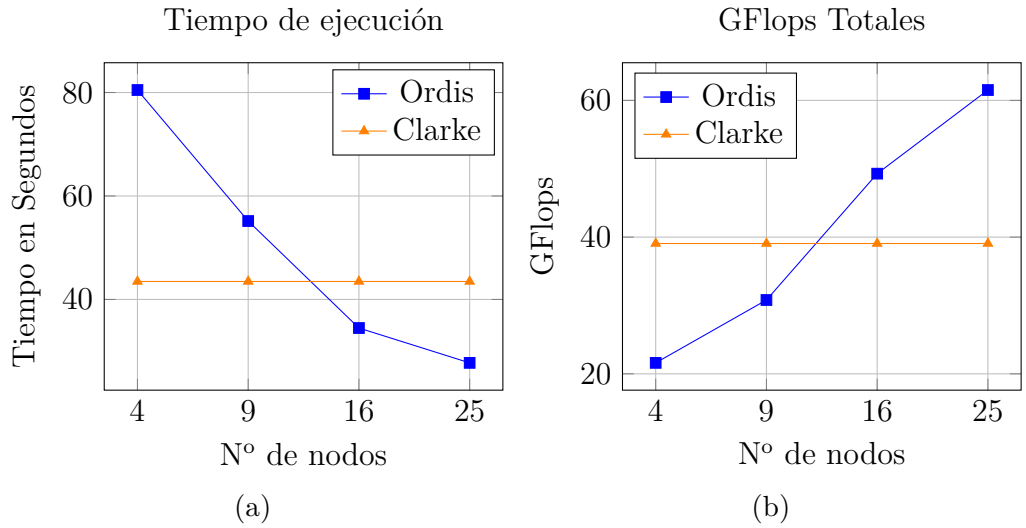



Figura 5.9: Gráfica comparativa del tiempo de ejecución del *benchmark* 5.9a y GFLOPS obtenidos 5.9b. Se muestra la evolución de ambas mediciones a medida que incrementamos el número de nodos. En el caso de Clarke, se muestra una línea horizontal ya que no existe posibilidad de escalar el número de nodos.

Como se extrae de las gráficas, cuando Ordis alcanza las 16 tarjetas gráficas (matriz de 4x4 nodos) reduce su tiempo por debajo de la marca de Clarke. Esto nos indica que a partir de ese tamaño compensará su lanzamiento en el clúster en lugar de en el servidor. Sin embargo, es necesario observar también el tiempo de comunicación.

El tiempo de comunicación va a ser el factor limitante es este problema. Como se muestra en el gráfico 5.10 el tiempo de comunicación crece conforme aumentamos el número de nodos utilizados, además se puede observar que el tiempo de Clarke para comunicar las tarjetas es ínfimo, debido a que las tarjetas se encuentran conectadas entre ellas por el bus PCI-E de Clarke, el cual dispone de un ancho de banda muy superior al Ethernet.

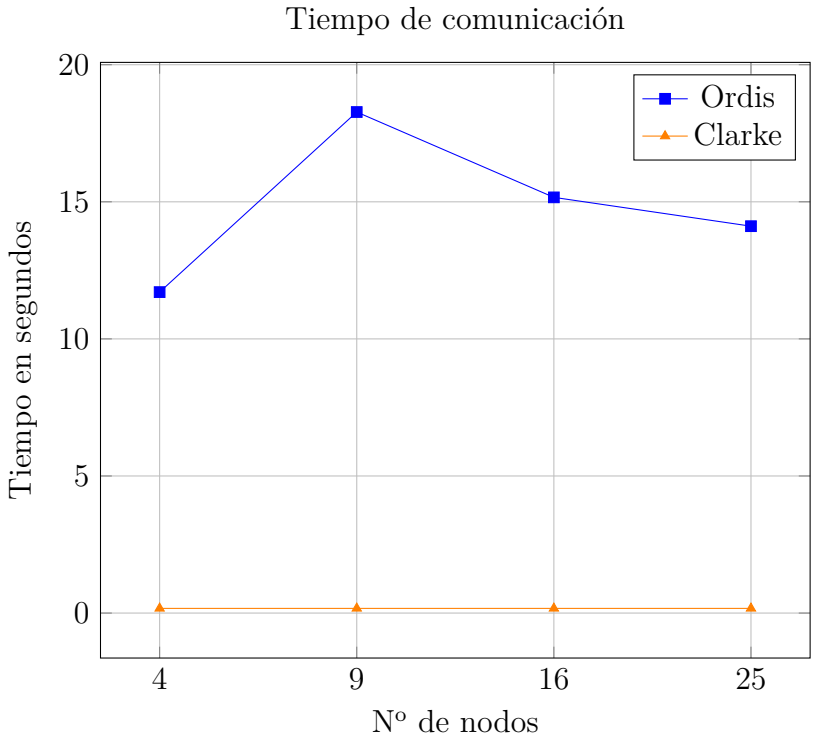



Figura 5.10: Gráfica de tiempo de comunicación entre procesos.

Un hecho a destacar en el gráfico del tiempo de comunicación se encuentra cuando una matriz de nodos de $3\times3$ representa el punto más alto de tiempo de comunicación. Esto es debido que al subdividir el problema entre más nodos del clúster, la comunicación entre nodos es mayor de forma obvia dada la mayor cantidad, pero el problema sigue siendo lo suficientemente grande por nodo como para requererir la transferencia de una gran cantidad de datos, por ello, la matriz $3 \times 3$ produce el mayor tiempo de comunicación. Con tamaños más grandes comenzamos a subdividir más el problema por lo que la cantidad de datos a transferir es menor y esto contrarresta el incremento de comunicaciones.

De la misma forma, al subdividir el problema en más nodos, el tiempo que cada uno necesita para resolver el problema es menor. Si llegamos a casos extremos esto puede conducir al hecho de que el tiempo de comunicación supere al tiempo de cómputo para la tarea, lo que haría que la paralelización no sea óptima para ciertos tamaños de problemas. Una representación de esta tendencia en Ordis se muestra en la Figura 5.11.

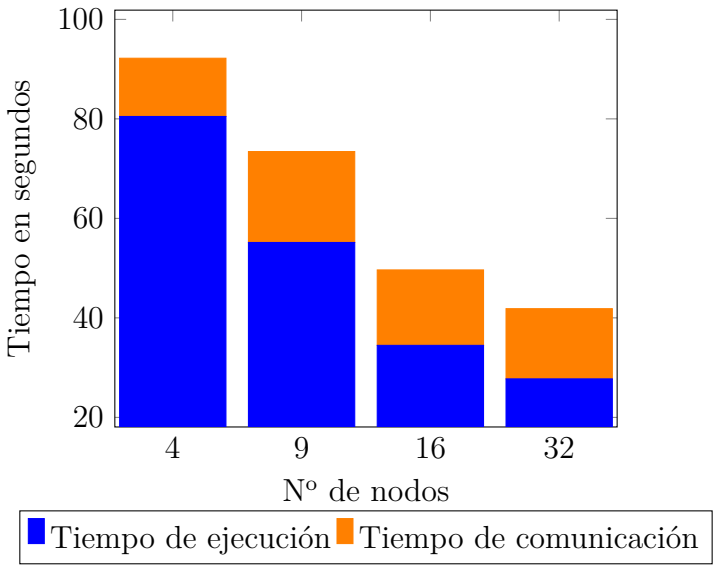



Figura 5.11: Gráfica comparativa del tiempo de computación y del tiempo de comunicación en Ordis.

Esto nos demuestra la importancia de la red de comunicación dentro de un clúster, así como la importancia de minimizar la transferencia de comunicación cuando se paraleliza un problema dado.

# Capítulo 6

# Conclusión

En este capítulo discutiremos las conclusiones extraídas de la realización de este trabajo. En la Sección 6.1 hablaremos sobre las conclusiones finales a las que hemos llegado con este Trabajo de Final de Grado. La Sección 6.2 resume los puntos fuertes del sistema propuesto así como los hitos alcanzados. Finalmente, en la Sección 6.3 se presentarán mejoras al sistema actual y posibles cambios futuros.

## 6.1. Conclusiones

En este trabajo se ha desarrollado por completo un clúster, desde la selección del hardware, pasando por su instalación y adaptación al entorno, hasta su puesta en marcha y funcionamiento. Además de ello, se ha creado un sistema capaz de simplificar el uso del mismo para que sea mas accesible y abierto de cara a usuarios no experimentados.

Adicionalmente, durante la fase de experimentación se ha comparado el sistema propuesto con soluciones comerciales existentes (Euler, el clúster del Instituto Universitario de Investigación Informática y Clarke, el servidor de supercomputación del Departamento de Tecnología Informática y Computación). En esta comparativa se han analizado los puntos fuertes y débiles de cada uno de ellos y de la solución presentada.

Durante este proyecto hemos podido aprovechar los conocimientos adquiridos a lo largo del Grado, desde la parte de planificación de un proyecto, pasado por la gestión de una red de computadores, hasta el diseño e implementación de una API para gestionar los recursos del clúster. Además de ser un proyecto final de grado, se ha implementado un sistema que será usado por alumnos de la EPSA, lo cual supone un gran valor añadido a este trabajo.

## 6.2. Aspectos destacados

Los apectos destacados del proyecto son los siguientes:

- Clúster:
  - Clúster de despliegue rápido (instalación y funcionamiento en menos de 2 horas).
  - Permite que otros sistemas operativos estén presentes en los nodos.
  - Escalable a múltimples laboratorios.
  - Master Node virtual, por lo que se puede integrar en cualquier sistema.
- API:
  - Interfaz Sencilla.
  - API FULL Rest, lo que permite que se incorpore fácilmente.
  - Facil de incrementar su funcionalidad.
  - Gestión de errores y notificaciones automáticas.
  - Gestión total del proceso de una trabajo sin intervención.

## 6.3. Trabajos futuros

Dada la limitación temporal impuesta para el proyecto, una serie de mejoras y posibles ampliaciones fueron dejadas de lado o pospuestas para una fase adicional del mismo. A continuación enumeraremos algunas de ellas.

La principal mejora es la actualización del sistema operativo base Rocks, el cual fue actualizado debidamente durante la fase final de este proyecto con los parches necesarios para mitigar los problemas de *Spectre* y *Meltdown*, vulnerabilidades que están presentes en el sistema actual. Además de la actualización del sistema operativo es necesario la actualización de las librerías y programas instalados, ya que durante el transcurso de este proyecto se han liberado versiones mejoradas para la mayoría de ellas.

Otra posible vía de mejora es la integración de los *scripts* de mantenimiento con la base de datos de la EPSA. Actualmente, los informes de error que genera el sistema se se almacenan de forma interna en texto plano dentro del master node de Ordis. Además de eso algo que se podría implementar es que el arrancado y apagado del clúster pudiese ser dinámico, o sea que utilizando la base de datos de la EPSA se puedan extraer los huecos disponibles en un laboratorio y si supera un umbral de tiempo arrancase el clúster para ejecutar trabajos.

También desde la concepción de este proyecto, la idea fue integrar sistemas de Deep Learning al clúster, pero debido a las limitaciones hardware de

las tarjetas gráficas, esto fue imposible. La idea es que con el cambio de los equipos para el curso académico 2018-2019, estos dispongan de la capacidad de lanzar proyectos de Deep Learning, con ello el interés del clúster aumentaría ya que no solo podríamos resolver problemas de paralelización tanto a CPU como a GPU, si no hacer Deep Learning y así permitir que alumnos que no disponen del hardware realizar proyectos durante los fines de semana y vacaciones.

Otra posible vía que fue relegada a pruebas fue la integración en el clúster con RCUDA, con ello sería posible que un nodo pueda disponer de forma simultánea de múltiples tarjetas gráficas de forma transparente para el sistema operativo, por ello se podrían paralelizar redes neuronales para que se ejecuten en múltiples tarjetas gráficas.

# Bibliografía


[1] Gordon E Moore. «Cramming more components onto integrated circuits». En: *Proceedings of the IEEE* 86.1 (1998), págs. 82-85.

[2] Gene M Amdahl. «Validity of the single processor approach to achieving large scale computing capabilities». En: *Proceedings of the April 18-20, 1967, spring joint computer conference*. ACM. 1967, págs. 483-485.

[3] John L Gustafson. «Reevaluating Amdahl's law». En: *Communications of the ACM* 31.5 (1988), págs. 532-533.

[4] Arthur J Bernstein. «Analysis of programs for parallel processing». En: *IEEE Transactions on Electronic Computers* 5 (1966), págs. 757-763.

[5] David B y W Hwu Wen-Mei. *Programming massively parallel processors: a hands-on approach*. Morgan kaufmann, 2010, págs. 2-3.

[6] David Geer. «Chip makers turn to multicore processors». En: *Computer* 38.5 (2005), págs. 11-13.

[7] Chenchen Fu y col. «Maximizing Common Idle Time on Multicore Processors With Shared Memory». En: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2017).

[8] Michael D McCool. «Scalable programming models for massively multicore processors». En: *Proceedings of the IEEE* 96.5 (2008), págs. 816-831.

[9] Bryan C Pijanowski y col. «A big data urban growth simulation at a national scale: configuring the GIS and neural network based land transformation model to run in a high performance computing (HPC) environment». En: *Environmental Modelling & Software* 51 (2014), págs. 250-268.

[10] Björn Gmeiner y col. «Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters». En: *Concurrency and Computation: Practice and Experience* 26.1 (2014), págs. 217-240.

[11] Lilia Ziane Khodja y col. «Parallel sparse linear solver with GMRES method using minimization techniques of communications for GPU clusters». En: *The journal of Supercomputing* 69.1 (2014), págs. 200-224.

[12] Ichitaro Yamazaki y col. «Domain decomposition preconditioners for communication-avoiding Krylov methods on a hybrid CPU/GPU cluster». En: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press. 2014, págs. 933-944.

[13] Javier Prades, Carlos Reaño y Federico Silla. «CUDA acceleration for Xen virtual machines in infiniband clusters with rCUDA». En: *ACM SIGPLAN Notices* 51.8 (2016), pág. 35.

[14] Pradeep Gupta. «How to Build a GPU-Accelerated Research Cluster». En: (2013). URL: https://devblogs.nvidia.com/how-build-gpu-accelerated-research-cluster/.

[15] Brad Nicholes Matt Massie Bernard Li y Vladimir Vuksan. *Monitoring with Ganglia*. O'Reilly Media, Inc., 2012, pág. 5.

[16] Brad Nicholes Matt Massie Bernard Li y Vladimir Vuksan. *Monitoring with Ganglia*. O'Reilly Media, Inc., 2012, págs. 4-8.

[17] D. Bailey y col. «THE NAS PARALLEL BENCHMARKS». En: (1994). URL: https://www.nas.nasa.gov/assets/pdf/techreports/1994/rnr-94-007.pdf.

[18] Jiri Kraus. «Benchmarking CUDA-Aware MPI». En: (2013). URL: https://devblogs.nvidia.com/benchmarking-cuda-aware-mpi/.

[19] Zhe Fan y col. «GPU cluster for high performance computing». En: *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*. IEEE. 2004, págs. 47-47.

[20] Javier Prades y Federico Silla. «Turning GPUs into Floating Devices over the Cluster: The Beauty of GPU Migration». En: *Parallel Processing Workshops (ICPPW), 2017 46th International Conference on*. IEEE. 2017, págs. 129-136.

# Apéndice A

# Instalación de ROCKS

La instalación de este sistema operativo se realiza utilizando la ISO de instalación de Rocks clúster. La instalación en sí misma es relativamente simple ya que basta con seguir las instrucciones de Anaconda para la instalación. Las únicas partes sensibles para la instalación son las que se describen a continuación.

Primero, la elección de los paquetes básicos de Rocks, algunos de ellos son incompatibles con otros por lo que hay ser cuidadosos con la elección y además, muchos de ellos no se pueden instalar una vez terminada la instalación. En nuestro caso elegimos los siguientes:

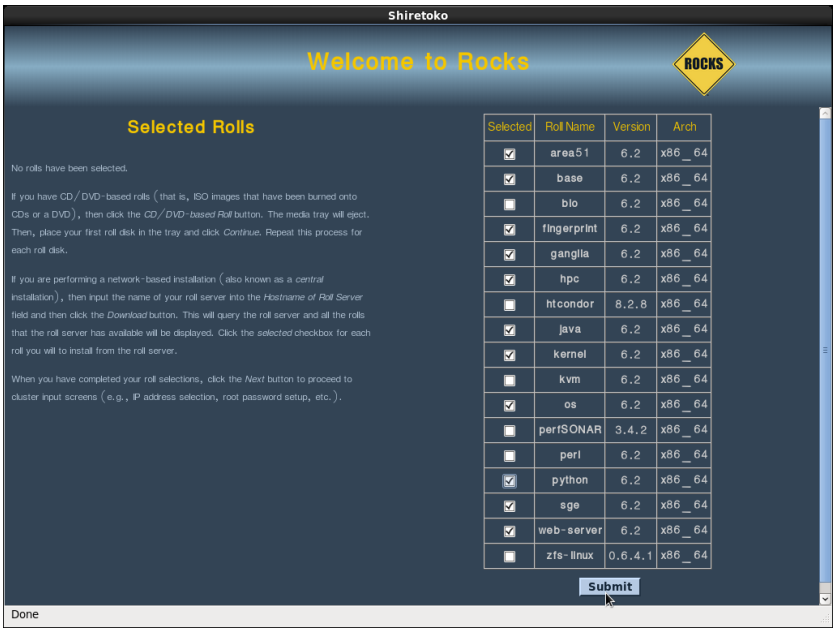



Figura A.1: Selección de paquetes en Rocks para Ordis.

Los básicos, e imprescindibles para el funcionamiento de Rocks son, base, kernel y os. El resto son mejoras de paquetes para el sistema. Para nuestro uso concreto hemos utilizado:

- Area51 :Envío de informes de sistema
- Fingerprint:Detección de dependencias
- Ganglia: Para monitorización de los nodos
- HPC:Incluye entornos paralelos como OpenMPI, Mpich, Mpich2.
- Java: Utilidad básica de java
- Python: Utilidad básica de python
- SGE: Gestor de colas de trabajo
- Web-server: Web server básico

La siguiente parte es especificar los datos del clúster, como nombre dirección, localización etc.:

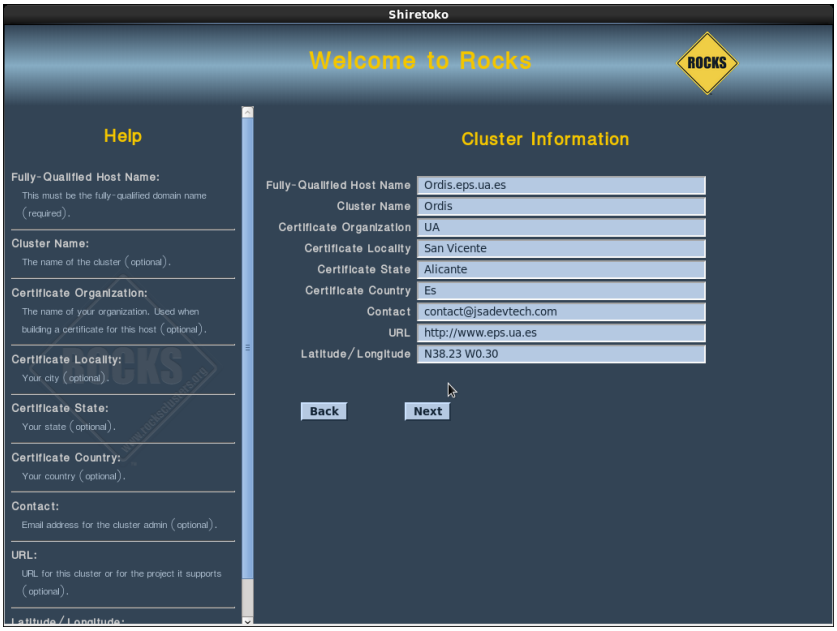



Figura A.2: Especificación de datos del clúster.

La Figura A.2 muestra los datos introducidos para Ordis.

La siguiente parte es la especificación de la red pública y la red privada del clúster. Este paso es de gran importancia, sobre todo la especificación de la red privada, ya que si se define mal, es prácticamente imposible cambiarla sin afectar al correcto funcionamiento del clúster, por lo que la IP que se le especifique en la instalación tiene que ser la definitiva. Con la IP pública también se puede provocar algún problema pero no de tanta severidad como

con la privada. La pauta a seguir es que ambas IPs especificadas sean las finales del sistema para evitar problemas.

En primer lugar, la especificación de la red pública se trata simplemente de especificar la IP y la máscara y seleccionar la interfaz de red que proporcionará la red pública tal y como se muestra en la Figura A.3.

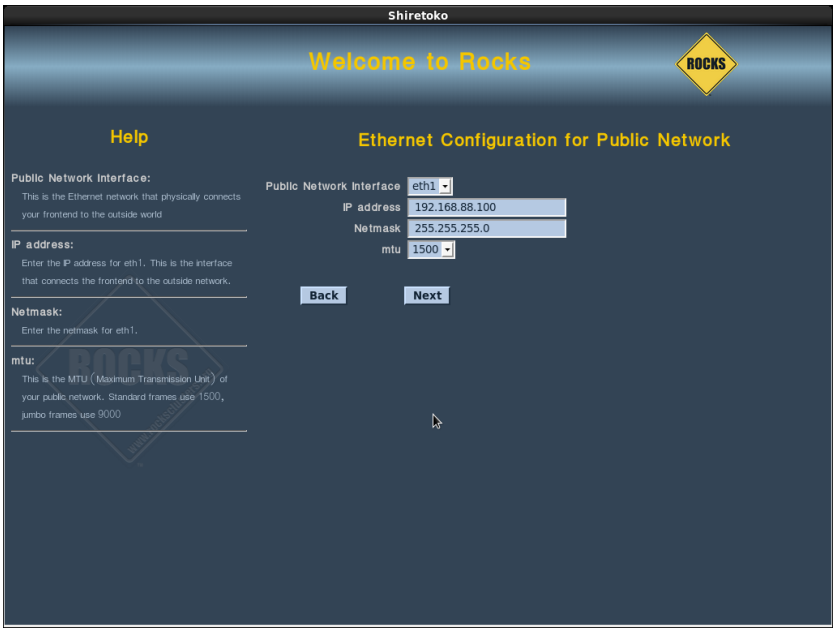



Figura A.3: Especificación de interfaz de red pública.

Seguidamente, la interfaz privada. al igual que la pública especificamos la IP y la máscara y la interfaz de red que proveerá la red interna del clúster. Es necesario ser cuidadosos porque nos permitirá elegir otra vez la misma interfaz de red que en la pública lo que puede provocar muchos problemas más adelante de la configuración (ver Figura A.4.

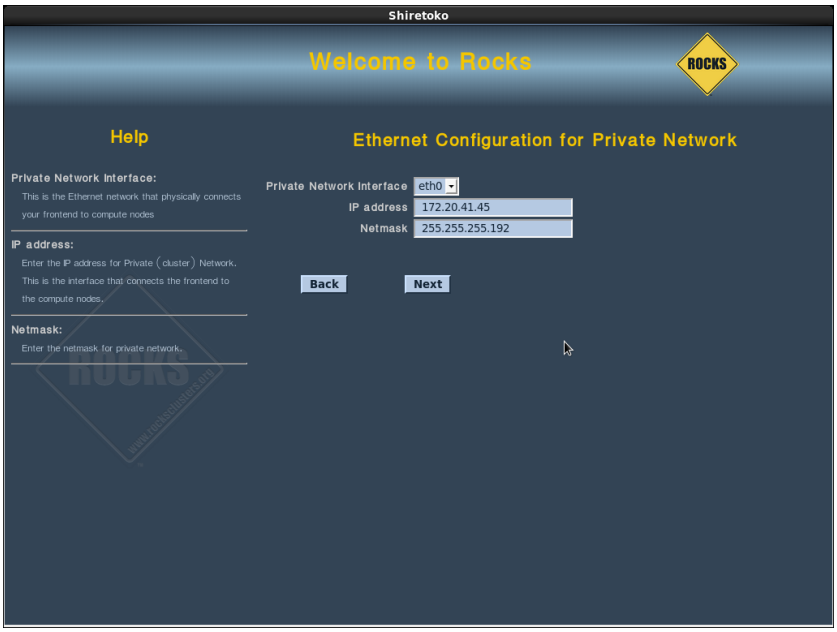



Figura A.4: Especificación de interfaz de red privada.

Después de estas configuraciones comenzará la instalación del Master Node. Tras ello nuestro sistema ya está listo para aplicarle las modificaciones descritas en el documento para que cumpla nuestras necesidades.

# Apéndice B

# Código

Todo el código se encuentra disponible en el siguiente GitHub mediante licencia GPL v3.0. https://github.com/jaquerinte/OrdisCode

# Apéndice C

# Datos Extraídos

## C.1. CPU benchmarks

### C.1.1. benchmark MG

#### C.1.1.1. Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 237,09 | 13133,63 | 820,85 |
| 16 | D | 231,33 | 13460,79 | 841,3 |
| 16 | D | 228,83 | 13607,49 | 850,47 |
| 16 | D | 311,96 | 9981,64 | 623,85 |
| 16 | D | 215,44 | 14453,66 | 903,35 |
| 16 | D | 426,03 | 7308,92 | 456,81 |
| 16 | D | 232,95 | 13366,98 | 835,44 |
| 16 | D | 432,02 | 7207,66 | 450,48 |
| 16 | D | 414,35 | 7515,1 | 469,69 |
| 16 | D | 318,89 | 9764,51 | 610,28 |
| 16 | D | 369,68 | 8423,1 | 526,44 |
| 16 | D | 256,87 | 12122,33 | 757,65 |
| 16 | D | 271,01 | 11489,95 | 718,12 |
| 16 | D | 264,25 | 11783,72 | 736,48 |
| 16 | D | 210,58 | 14787,02 | 924,19 |
| 16 | D | 227,96 | 13659,81 | 853,74 |
| 16 | D | 287,85 | 10817,57 | 676,1 |
| 16 | D | 279,88 | 11125,84 | 695,36 |
| 16 | D | 240,96 | 12922,75 | 807,67 |
| 16 | D | 284,31 | 10952,46 | 684,53 |

| | | | | |
|---|---|---|---|---|
| 16 | D | 275,98 | 11283,07 | 705,19 |
| 16 | D | 246,55 | 12629,57 | 789,35 |
| 16 | D | 336,1 | 9264,71 | 579,04 |
| 16 | D | 294,32 | 10579,98 | 661,25 |
| 16 | D | 190,66 | 16332,18 | 1020,76 |
| 16 | D | 274,93 | 11325,93 | 707,87 |
| 16 | D | 194,74 | 15989,51 | 999,34 |
| 16 | D | 275,45 | 11304,69 | 706,54 |
| 16 | D | 194,73 | 15990,6 | 999,41 |
| 16 | D | 240,32 | 12957,2 | 809,83 |
| 16 | D | 258,7 | 12036,33 | 752,27 |
| 16 | D | 239,74 | 12988,72 | 811,8 |
| 16 | D | 261,78 | 11894,87 | 743,43 |
| 16 | D | 262,85 | 11846,69 | 740,42 |
| 16 | D | 263,37 | 11823,13 | 738,95 |
| 16 | D | 288,61 | 10789,22 | 674,33 |
| 16 | D | 209,84 | 14839,46 | 927,47 |
| 16 | D | 228,88 | 13604,75 | 850,3 |
| 16 | D | 223,15 | 13954,07 | 872,13 |
| 16 | D | 329,03 | 9463,76 | 591,49 |

Cuadro C.1: MG 16 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 126,27 | 24659,5 | 770,61 |
| 32 | D | 219,49 | 14186,84 | 443,34 |
| 32 | D | 181,43 | 17162,41 | 536,33 |
| 32 | D | 218,52 | 14249,84 | 445,31 |
| 32 | D | 182,3 | 17081,24 | 533,79 |
| 32 | D | 125,75 | 24762,84 | 773,84 |
| 32 | D | 186,18 | 16724,59 | 522,64 |
| 32 | D | 225,82 | 13789,34 | 430,92 |
| 32 | D | 187,82 | 16578,85 | 518,09 |
| 32 | D | 182,63 | 17050,15 | 532,82 |
| 32 | D | 147,72 | 21079,98 | 658,75 |
| 32 | D | 217,51 | 14315,73 | 447,37 |
| 32 | D | 218,18 | 14271,61 | 445,99 |
| 32 | D | 219,96 | 14156,48 | 442,39 |
| 32 | D | 180,39 | 17261,99 | 539,44 |
| 32 | D | 135,26 | 23021,9 | 719,43 |
| 32 | D | 198,66 | 15674,18 | 489,82 |

| | | | | |
|---|---|---|---|---|
| 32 | D | 560,01 | 5560,4 | 173,76 |
| 32 | D | 155,27 | 20053,82 | 626,68 |
| 32 | D | 187,7 | 16589,21 | 518,41 |
| 32 | D | 151,5 | 20553,03 | 642,28 |
| 32 | D | 162,14 | 19204,77 | 600,15 |
| 32 | D | 217,07 | 14344,63 | 448,27 |
| 32 | D | 124,83 | 24944,03 | 779,5 |
| 32 | D | 138,6 | 22466,56 | 702,08 |
| 32 | D | 182,56 | 17056,15 | 533 |
| 32 | D | 220,18 | 14142,27 | 441,95 |
| 32 | D | 181,02 | 17201,97 | 537,56 |
| 32 | D | 175,65 | 17727,93 | 554 |
| 32 | D | 221,98 | 14027,71 | 438,37 |
| 32 | D | 182,51 | 17061,65 | 533,18 |
| 32 | D | 183,22 | 16994,91 | 531,09 |
| 32 | D | 195,63 | 15917,11 | 497,41 |
| 32 | D | 184,88 | 16842,5 | 526,33 |
| 32 | D | 145,72 | 21369,32 | 667,79 |
| 32 | D | 169,81 | 18336,92 | 573,03 |
| 32 | D | 218,52 | 14249,62 | 445,3 |
| 32 | D | 178,25 | 17469,11 | 545,91 |
| 32 | D | 182,57 | 17055,88 | 533 |
| 32 | D | 169,8 | 18338,82 | 573,09 |

Cuadro C.2: MG 32 Euler

#### C.1.1.2. Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 164,99 | 18873,25 | 1179,58 |
| 16 | D | 164,48 | 18931,03 | 1183,19 |
| 16 | D | 166,34 | 18719,81 | 1169,99 |
| 16 | D | 169,73 | 18345,7 | 1146,61 |
| 16 | D | 486,71 | 6397,8 | 399,86 |
| 16 | D | 166,44 | 18708,48 | 1169,28 |
| 16 | D | 160,93 | 19349,48 | 1209,34 |

Cuadro C.3: MG 16 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 141,46 | 22011,5 | 687,86 |
| 32 | D | 136,15 | 22871,24 | 714,73 |
| 32 | D | 136,38 | 22832,57 | 713,52 |

Cuadro C.4: MG 32 Ordis

## C.1.2. benchmark CG

### C.1.2.1. Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 1950,66 | 1867,52 | 116,72 |
| 16 | D | 3320,11 | 1097,22 | 68,58 |
| 16 | D | 2640,36 | 1379,7 | 86,23 |
| 16 | D | 2495,96 | 1459,52 | 91,22 |
| 16 | D | 3549,98 | 1026,17 | 64,14 |
| 16 | D | 3631,17 | 1003,23 | 62,7 |
| 16 | D | 2950,12 | 1234,83 | 77,18 |
| 16 | D | 2620,56 | 1390,12 | 86,88 |
| 16 | D | 2774,39 | 1313,05 | 82,07 |
| 16 | D | 2249,48 | 1619,44 | 101,21 |
| 16 | D | 2707,08 | 1345,69 | 84,11 |
| 16 | D | 1910,92 | 1906,36 | 119,15 |
| 16 | D | 2012,14 | 1810,46 | 113,15 |
| 16 | D | 3014,32 | 1208,53 | 75,53 |
| 16 | D | 2784,77 | 1308,15 | 81,76 |
| 16 | D | 2974,18 | 1224,84 | 76,55 |
| 16 | D | 2404,84 | 1514,82 | 94,68 |
| 16 | D | 2200,16 | 1655,74 | 103,48 |
| 16 | D | 3112,88 | 1170,27 | 73,14 |
| 16 | D | 1680,03 | 2168,35 | 135,52 |
| 16 | D | 1793,74 | 2030,9 | 126,93 |
| 16 | D | 2263,54 | 1609,38 | 100,59 |
| 16 | D | 1725,71 | 2110,96 | 131,93 |
| 16 | D | 2067,87 | 1761,67 | 110,1 |
| 16 | D | 2148,94 | 1695,2 | 105,95 |
| 16 | D | 1777,77 | 2049,14 | 128,07 |
| 16 | D | 2493,65 | 1460,87 | 91,3 |
| 16 | D | 2437,43 | 1494,56 | 93,41 |

| | | | | |
|---|---|---|---|---|
| 16 | D | 1746,67 | 2085,62 | 130,35 |
| 16 | D | 1749,07 | 2082,76 | 130,17 |
| 16 | D | 1870,12 | 1947,95 | 121,75 |
| 16 | D | 1968,33 | 1850,76 | 115,67 |
| 16 | D | 2500,31 | 1456,98 | 91,06 |
| 16 | D | 1851,04 | 1968,03 | 123 |
| 16 | D | 1708,03 | 2132,8 | 133,3 |
| 16 | D | 2320,06 | 1570,18 | 98,14 |
| 16 | D | 1848,33 | 1970,92 | 123,18 |
| 16 | D | 1749,87 | 2081,81 | 130,11 |
| 16 | D | 1877,3 | 1940,5 | 121,28 |
| 16 | D | 1990,98 | 1829,7 | 114,36 |
| 16 | D | 1847,83 | 1971,44 | 123,22 |
| 16 | D | 1705,98 | 2135,37 | 133,46 |
| 16 | D | 1807,82 | 2015,08 | 125,94 |
| 16 | D | 2368,82 | 1537,86 | 96,12 |
| 16 | D | 1920,46 | 1896,89 | 118,56 |
| 16 | D | 1931,25 | 1886,29 | 117,89 |
| 16 | D | 2741,95 | 1328,58 | 83,04 |
| 16 | D | 2159,15 | 1687,19 | 105,45 |
| 16 | D | 2763,68 | 1318,13 | 82,38 |
| 16 | D | 1597,83 | 2279,91 | 142,49 |

Cuadro C.5: CG 16 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 1121,97 | 3246,88 | 101,47 |
| 32 | D | 745,81 | 4884,49 | 152,64 |
| 32 | D | 706,02 | 5159,76 | 161,24 |
| 32 | D | 727,52 | 5007,28 | 156,48 |
| 32 | D | 587,63 | 6199,26 | 193,73 |
| 32 | D | 3204,74 | 1136,72 | 35,52 |
| 32 | D | 544,71 | 6687,73 | 208,99 |
| 32 | D | 582,65 | 6252,32 | 195,39 |
| 32 | D | 2641,73 | 1378,98 | 43,09 |
| 32 | D | 2580,8 | 1411,54 | 44,11 |
| 32 | D | 2760,39 | 1319,7 | 41,24 |
| 32 | D | 2871,61 | 1268,59 | 39,64 |
| 32 | D | 2350,95 | 1549,55 | 48,42 |
| 32 | D | 3603,48 | 1010,94 | 31,59 |
| 32 | D | 443,88 | 8206,94 | 256,47 |

| | | | | |
|---|---|---|---|---|
| 32 | D | 1520,17 | 2396,37 | 74,89 |
| 32 | D | 1820,3 | 2001,26 | 62,54 |
| 32 | D | 777,83 | 4683,39 | 146,36 |
| 32 | D | 2173,79 | 1675,83 | 52,37 |
| 32 | D | 1370,43 | 2658,22 | 83,07 |
| 32 | D | 575,4 | 6331,09 | 197,85 |
| 32 | D | 1269,4 | 2869,77 | 89,68 |
| 32 | D | 762,55 | 4777,26 | 149,29 |
| 32 | D | 1335,73 | 2727,26 | 85,23 |
| 32 | D | 1742,97 | 2090,05 | 65,31 |
| 32 | D | 1338,25 | 2722,14 | 85,07 |
| 32 | D | 1323,49 | 2752,51 | 86,02 |
| 32 | D | 1401,79 | 2598,75 | 81,21 |
| 32 | D | 581,33 | 6266,47 | 195,83 |
| 32 | D | 780,04 | 4670,17 | 145,94 |
| 32 | D | 1417,36 | 2570,2 | 80,32 |
| 32 | D | 1403,04 | 2596,43 | 81,14 |
| 32 | D | 564,93 | 6448,41 | 201,51 |
| 32 | D | 1767,52 | 2061,03 | 64,41 |
| 32 | D | 2190,97 | 1662,69 | 51,96 |
| 32 | D | 791,56 | 4602,2 | 143,82 |

Cuadro C.6: CG 32 Euler

#### C.1.2.2. Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 2829,28 | 1287,57 | 80,47 |
| 16 | D | 1649,7 | 2208,21 | 138,01 |
| 16 | D | 1725,57 | 2111,13 | 131,95 |
| 16 | D | 1787,92 | 2037,51 | 127,34 |
| 16 | D | 1709,6 | 2130,85 | 133,18 |

Cuadro C.7: CG 16 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 901,93 | 4039,01 | 126,22 |
| 32 | D | 906,25 | 4019,73 | 125,62 |
| 32 | D | 947,46 | 3844,91 | 120,15 |

Cuadro C.8: CG 32 Ordis

## C.1.3. benchmark FT

### C.1.3.1. Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 4 | C | 126,68 | 3129,03 | 782,26 |
| 4 | C | 155,65 | 2546,72 | 636,68 |
| 4 | C | 153,32 | 2585,34 | 646,34 |
| 4 | C | 164,16 | 2414,72 | 603,68 |
| 4 | C | 128,86 | 3076,17 | 769,04 |
| 4 | C | 124,32 | 3188,34 | 797,08 |
| 4 | C | 179,19 | 2212,15 | 553,04 |
| 4 | C | 179,59 | 2207,21 | 551,8 |
| 4 | C | 179,6 | 2207,04 | 551,76 |
| 4 | C | 149,98 | 2643,02 | 660,76 |
| 4 | C | 150,83 | 2628,12 | 657,03 |
| 4 | C | 159,49 | 2485,33 | 621,33 |
| 4 | C | 149,9 | 2644,33 | 661,08 |
| 4 | C | 163,12 | 2430,03 | 607,51 |
| 4 | C | 162,46 | 2439,87 | 609,97 |
| 4 | C | 163,75 | 2420,71 | 605,18 |
| 4 | C | 177,4 | 2234,39 | 558,6 |
| 4 | C | 178,53 | 2220,35 | 555,09 |
| 4 | C | 150,18 | 2639,47 | 659,87 |
| 4 | C | 154,43 | 2566,83 | 641,71 |
| 4 | C | 154 | 2573,93 | 643,48 |
| 4 | C | 148,44 | 2670,38 | 667,59 |
| 4 | C | 147,74 | 2683,1 | 670,77 |
| 4 | C | 144,93 | 2734,95 | 683,74 |
| 4 | C | 159,1 | 2491,44 | 622,86 |
| 4 | C | 158,7 | 2497,76 | 624,44 |
| 4 | C | 159,05 | 2492,28 | 623,07 |
| 4 | C | 144,57 | 2741,84 | 685,46 |
| 4 | C | 144,22 | 2748,53 | 687,13 |
| 4 | C | 144,67 | 2739,94 | 684,99 |
| 4 | C | 149,18 | 2657,11 | 664,28 |
| 4 | C | 147,01 | 2696,33 | 674,08 |
| 4 | C | 151,9 | 2609,61 | 652,4 |
| 4 | C | 161,22 | 2458,62 | 614,65 |
| 4 | C | 152,44 | 2600,23 | 650,06 |
| 4 | C | 158,15 | 2506,44 | 626,61 |

| | | | | |
|---|---|---|---|---|
| 4 | C | 115,72 | 3425,51 | 856,38 |
| 4 | C | 116,46 | 3403,5 | 850,88 |
| 4 | C | 110,07 | 3601,09 | 900,27 |

Cuadro C.9: FT 4 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 8 | C | 78,78 | 5031,74 | 628,97 |
| 8 | C | 73,09 | 5422,93 | 677,87 |
| 8 | C | 76,24 | 5199,25 | 649,91 |
| 8 | C | 76,13 | 5206,52 | 650,82 |
| 8 | C | 86,94 | 4559,44 | 569,93 |
| 8 | C | 80,16 | 4945,22 | 618,15 |
| 8 | C | 72,81 | 5443,96 | 680,49 |
| 8 | C | 73,25 | 5411,33 | 676,42 |
| 8 | C | 72,54 | 5464,72 | 683,09 |
| 8 | C | 79,38 | 4993,72 | 624,21 |
| 8 | C | 79,86 | 4963,48 | 620,43 |
| 8 | C | 72,39 | 5475,84 | 684,48 |
| 8 | C | 82,29 | 4817,05 | 602,13 |
| 8 | C | 87,17 | 4547,41 | 568,43 |
| 8 | C | 85,33 | 4645,56 | 580,7 |
| 8 | C | 74,16 | 5345,37 | 668,17 |
| 8 | C | 554,15 | 715,31 | 89,41 |
| 8 | C | 71,52 | 5542,39 | 692,8 |
| 8 | C | 80,76 | 4908,36 | 613,55 |
| 8 | C | 73,35 | 5403,76 | 675,47 |
| 8 | C | 85,45 | 4638,93 | 579,87 |
| 8 | C | 86,1 | 4603,7 | 575,46 |
| 8 | C | 70,73 | 5604,24 | 700,53 |
| 8 | C | 71,66 | 5531,44 | 691,43 |
| 8 | C | 72,81 | 5443,98 | 680,5 |
| 8 | C | 81,7 | 4851,92 | 606,49 |
| 8 | C | 79,67 | 4975,59 | 621,95 |
| 8 | C | 75,42 | 5255,51 | 656,94 |
| 8 | C | 80,69 | 4912,19 | 614,02 |
| 8 | C | 91,2 | 4346,51 | 543,31 |
| 8 | C | 91,58 | 4328,56 | 541,07 |
| 8 | C | 76,05 | 5212,51 | 651,56 |
| 8 | C | 87,2 | 4545,79 | 568,22 |
| 8 | C | 86,11 | 4603,44 | 575,43 |

| | | | | |
|---|---|---|---|---|
| 8 | C | 89,69 | 4419,46 | 552,43 |
| 8 | C | 94,46 | 4196,54 | 524,57 |
| 8 | C | 90,55 | 4377,65 | 547,21 |
| 8 | C | 89,86 | 4410,93 | 551,37 |
| 8 | C | 92,26 | 4296,32 | 537,04 |
| 8 | C | 89,47 | 4430,36 | 553,8 |
| 8 | C | 69,37 | 5714,15 | 714,27 |
| 8 | C | 89,69 | 4419,37 | 552,42 |
| 8 | C | 68,9 | 5752,97 | 719,12 |
| 8 | C | 69,05 | 5740,55 | 717,57 |
| 8 | C | 71,26 | 5562,21 | 695,28 |
| 8 | C | 85,74 | 4623,39 | 577,92 |
| 8 | C | 96,61 | 4103,14 | 512,89 |
| 8 | C | 77,58 | 5109,33 | 638,67 |
| 8 | C | 75,9 | 5222,69 | 652,84 |
| 8 | C | 70,75 | 5603,05 | 700,38 |
| 8 | C | 66,64 | 5947,97 | 743,5 |

Cuadro C.10: FT 8 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | C | 57,79 | 6859,64 | 428,73 |
| 16 | C | 50,2 | 7895,53 | 493,47 |
| 16 | C | 50,31 | 7878,28 | 492,39 |
| 16 | C | 50,51 | 7848,24 | 490,52 |
| 16 | C | 37,9 | 10458,72 | 653,67 |
| 16 | C | 39,95 | 9922,78 | 620,17 |
| 16 | C | 48,07 | 8245,58 | 515,35 |
| 16 | C | 49,9 | 7943,85 | 496,49 |
| 16 | C | 46,62 | 8503,18 | 531,45 |
| 16 | C | 51,76 | 7658,5 | 478,66 |
| 16 | C | 51,44 | 7706,03 | 481,63 |
| 16 | C | 43,56 | 9099,32 | 568,71 |
| 16 | C | 47,63 | 8321,71 | 520,11 |
| 16 | C | 48,91 | 8104,26 | 506,52 |
| 16 | C | 48,7 | 8139,3 | 508,71 |
| 16 | C | 47,02 | 8429,91 | 526,87 |
| 16 | C | 46,97 | 8439,75 | 527,48 |
| 16 | C | 57,67 | 6873,57 | 429,6 |
| 16 | C | 46,42 | 8539,12 | 533,7 |
| 16 | C | 41,47 | 9559,35 | 597,46 |

| | | | | |
|---|---|---|---|---|
| 16 | C | 48,27 | 8212,02 | 513,25 |
| 16 | C | 47,64 | 8320,77 | 520,05 |
| 16 | C | 47,1 | 8415,36 | 525,96 |
| 16 | C | 45,54 | 8704,77 | 544,05 |
| 16 | C | 48,02 | 8255,15 | 515,95 |
| 16 | C | 48,08 | 8244,82 | 515,3 |
| 16 | C | 41,95 | 9449,65 | 590,6 |
| 16 | C | 42,51 | 9324,71 | 582,79 |
| 16 | C | 43,52 | 9108,76 | 569,3 |
| 16 | C | 41,43 | 9567,3 | 597,96 |
| 16 | C | 38,34 | 10337,98 | 646,12 |
| 16 | C | 43,47 | 9118,87 | 569,93 |
| 16 | C | 42,55 | 9315,7 | 582,23 |
| 16 | C | 42,74 | 9274,63 | 579,66 |
| 16 | C | 43,03 | 9211,9 | 575,74 |
| 16 | C | 42,73 | 9277,07 | 579,82 |
| 16 | C | 46,84 | 8462,29 | 528,89 |
| 16 | C | 49,99 | 7929,53 | 495,6 |
| 16 | C | 44,97 | 8814,45 | 550,9 |
| 16 | C | 44,94 | 8820,06 | 551,25 |
| 16 | C | 46,34 | 8554,77 | 534,67 |
| 16 | C | 49,17 | 8061,18 | 503,82 |
| 16 | C | 49,8 | 7959,67 | 497,48 |
| 16 | C | 68,16 | 5815,39 | 363,46 |
| 16 | C | 60,18 | 6586,53 | 411,66 |
| 16 | C | 55,27 | 7172,17 | 448,26 |
| 16 | C | 49,47 | 8013,01 | 500,81 |
| 16 | C | 52,5 | 7550,9 | 471,93 |
| 16 | C | 59,45 | 6667,05 | 416,69 |
| 16 | C | 51,13 | 7753,07 | 484,57 |
| 16 | C | 58,9 | 6729,33 | 420,58 |

Cuadro C.11: FT 16 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | C | 27,57 | 14377,35 | 449,29 |
| 32 | C | 24,33 | 16295,3 | 509,23 |
| 32 | C | 26,64 | 14880,77 | 465,02 |
| 32 | C | 23,29 | 17017,54 | 531,8 |
| 32 | C | 28,25 | 14033,2 | 438,54 |
| 32 | C | 24,37 | 16264,61 | 508,27 |

| | | | | |
|---|---|---|---|---|
| 32 | C | 29,61 | 13385,71 | 418,3 |
| 32 | C | 31,67 | 12514,55 | 391,08 |
| 32 | C | 26,99 | 14686,01 | 458,94 |
| 32 | C | 26,95 | 14708,94 | 459,65 |
| 32 | C | 25,57 | 15500,77 | 484,4 |
| 32 | C | 25,68 | 15437,06 | 482,41 |
| 32 | C | 25,13 | 15773,5 | 492,92 |
| 32 | C | 23,59 | 16805,91 | 525,18 |
| 32 | C | 26,52 | 14946,33 | 467,07 |
| 32 | C | 23,23 | 17066,7 | 533,33 |
| 32 | C | 31,45 | 12605,75 | 393,93 |
| 32 | C | 31,87 | 12439,33 | 388,73 |
| 32 | C | 25,5 | 15546,89 | 485,84 |
| 32 | C | 25,14 | 15765,67 | 492,68 |
| 32 | C | 101,57 | 3902,59 | 121,96 |
| 32 | C | 25,37 | 15627,03 | 488,34 |
| 32 | C | 83,47 | 4749,04 | 148,41 |
| 32 | C | 28,2 | 14055,71 | 439,24 |
| 32 | C | 24,64 | 16086,86 | 502,71 |
| 32 | C | 24,83 | 15967,17 | 498,97 |
| 32 | C | 25,31 | 15659,93 | 489,37 |
| 32 | C | 23,94 | 16557,15 | 517,41 |
| 32 | C | 26,81 | 14783,76 | 461,99 |
| 32 | C | 23,57 | 16816,3 | 525,51 |
| 32 | C | 25,75 | 15393,8 | 481,06 |
| 32 | C | 25,39 | 15611,6 | 487,86 |
| 32 | C | 25,74 | 15397,59 | 481,17 |
| 32 | C | 24,56 | 16140,01 | 504,38 |
| 32 | C | 25,62 | 15469,71 | 483,43 |
| 32 | C | 24,48 | 16194,43 | 506,08 |
| 32 | C | 27,22 | 14561,92 | 455,06 |
| 32 | C | 28,01 | 14150,23 | 442,19 |
| 32 | C | 29,02 | 13660,1 | 426,88 |
| 32 | C | 24,53 | 16156,49 | 504,89 |
| 32 | C | 23,47 | 16885,7 | 527,68 |
| 32 | C | 24,1 | 16450,61 | 514,08 |
| 32 | C | 24,52 | 16165,79 | 505,18 |
| 32 | C | 23,87 | 16609,45 | 519,05 |

Cuadro C.12: FT 32 Euler

### C.1.3.2. Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 4 | C | 189,49 | 2091,92 | 522,98 |
| 4 | C | 190,42 | 2081,64 | 520,41 |
| 4 | C | 189,62 | 2090,47 | 522,62 |
| 4 | C | 1174,45 | 337,51 | 84,38 |
| 4 | C | 196,03 | 2022,03 | 505,51 |
| 4 | C | 186,38 | 2126,72 | 531,68 |
| 4 | C | 186,29 | 2127,76 | 531,94 |
| 4 | C | 192,41 | 2060,17 | 515,04 |

Cuadro C.13: FT 4 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 8 | C | 140,23 | 2826,79 | 353,35 |
| 8 | C | 143,06 | 2770,84 | 346,35 |
| 8 | C | 172,26 | 2301,07 | 287,63 |
| 8 | C | 142,05 | 2790,48 | 348,81 |
| 8 | C | 174,75 | 2268,26 | 283,53 |
| 8 | C | 140,14 | 2828,53 | 353,57 |
| 8 | C | 143,44 | 2763,53 | 345,44 |
| 8 | C | 178,62 | 2219,21 | 277,4 |

Cuadro C.14: FT 8 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | C | 141,28 | 2805,64 | 175,35 |
| 16 | C | 104,63 | 3788,57 | 236,79 |
| 16 | C | 86,68 | 4572,85 | 285,8 |
| 16 | C | 103,01 | 3847,92 | 240,5 |

Cuadro C.15: FT 16 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | C | 67,56 | 5867,05 | 183,35 |
| 32 | C | 65,68 | 6035,1 | 188,6 |
| 32 | C | 66,82 | 5932,46 | 185,39 |
| 32 | C | 66,28 | 5980,52 | 186,89 |

| | | | | |
|---|---|---|---|---|
| 32 | C | 77,81 | 5094,49 | 159,2 |

Cuadro C.16: FT 32 Ordis

## C.1.4. benchmark LU

### C.1.4.1. Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 6 | D | 8251,59 | 4835,18 | 805,86 |
| 6 | D | 8819,73 | 4523,71 | 753,95 |
| 6 | D | 8982,15 | 4441,92 | 740,32 |
| 6 | D | 9980,12 | 3997,74 | 666,29 |
| 6 | D | 7213,35 | 5531,12 | 921,85 |
| 6 | D | 10272,11 | 3884,1 | 647,35 |
| 6 | D | 11263,58 | 3542,21 | 590,37 |
| 6 | D | 6676,74 | 5975,66 | 995,94 |
| 6 | D | 10765,34 | 3706,15 | 617,69 |
| 6 | D | 7775,15 | 5131,47 | 855,25 |
| 6 | D | 10121,09 | 3942,06 | 657,01 |
| 6 | D | 11615,22 | 3434,97 | 572,49 |
| 6 | D | 11150,41 | 3578,16 | 596,36 |
| 6 | D | 10807,16 | 3691,81 | 615,3 |
| 6 | D | 8224,45 | 4851,14 | 808,52 |
| 6 | D | 7433,82 | 5367,09 | 894,51 |
| 6 | D | 7932,41 | 5029,74 | 838,29 |
| 6 | D | 7111 | 5610,73 | 935,12 |
| 6 | D | 11215,59 | 3557,36 | 592,89 |
| 6 | D | 7527,94 | 5299,98 | 883,33 |
| 6 | D | 7977,95 | 5001,03 | 833,5 |
| 6 | D | 11288,63 | 3534,35 | 589,06 |
| 6 | D | 10407,84 | 3833,45 | 638,91 |
| 6 | D | 8353,31 | 4776,3 | 796,05 |
| 6 | D | 8868,1 | 4499,04 | 749,84 |
| 6 | D | 7499,22 | 5320,28 | 886,71 |
| 6 | D | 9870,69 | 4042,06 | 673,68 |
| 6 | D | 10596,43 | 3765,23 | 627,54 |
| 6 | D | 7149,11 | 5580,82 | 930,14 |
| 6 | D | 11815,72 | 3376,68 | 562,78 |
| 6 | D | 10282,66 | 3880,12 | 646,69 |
| 6 | D | 9938,32 | 4014,56 | 669,09 |

| | | | | |
|---|---|---|---|---|
| 6 | D | 11271,21 | 3539,81 | 589,97 |
| 6 | D | 5816,11 | 6859,91 | 1143,32 |
| 6 | D | 10058,49 | 3966,59 | 661,1 |
| 6 | D | 8863,13 | 4501,56 | 750,26 |
| 6 | D | 9692,48 | 4116,38 | 686,06 |
| 6 | D | 10973,23 | 3635,93 | 605,99 |
| 6 | D | 6924,14 | 5762,15 | 960,36 |
| 6 | D | 8310,3 | 4801,02 | 800,17 |
| 6 | D | 7396,77 | 5393,96 | 898,99 |
| 6 | D | 8828,19 | 4519,38 | 753,23 |
| 6 | D | 7216,23 | 5528,92 | 921,49 |
| 6 | D | 8166,18 | 4885,75 | 814,29 |
| 6 | D | 10267,48 | 3885,86 | 647,64 |
| 6 | D | 8345,89 | 4780,55 | 796,76 |
| 6 | D | 10567,38 | 3775,58 | 629,26 |
| 6 | D | 7961,39 | 5011,43 | 835,24 |
| 6 | D | 7962,01 | 5011,04 | 835,17 |
| 6 | D | 6967,56 | 5726,24 | 954,37 |

Cuadro C.17: LU 6 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 8 | D | 6492,15 | 6145,57 | 768,2 |
| 8 | D | 5187,61 | 7691 | 961,38 |
| 8 | D | 6352,77 | 6280,4 | 785,05 |
| 8 | D | 7356,2 | 5423,71 | 677,96 |
| 8 | D | 10580,25 | 3770,98 | 471,37 |
| 8 | D | 6070,99 | 6571,9 | 821,49 |
| 8 | D | 7429,03 | 5370,55 | 671,32 |
| 8 | D | 5791,81 | 6888,68 | 861,09 |
| 8 | D | 5665,39 | 7042,4 | 880,3 |
| 8 | D | 6376,54 | 6256,98 | 782,12 |
| 8 | D | 5517,7 | 7230,9 | 903,86 |
| 8 | D | 7229,43 | 5518,82 | 689,85 |
| 8 | D | 5688,14 | 7014,23 | 876,78 |
| 8 | D | 7620,25 | 5235,77 | 654,47 |
| 8 | D | 5712,19 | 6984,69 | 873,09 |
| 8 | D | 8791,46 | 4538,26 | 567,28 |
| 8 | D | 5827,01 | 6847,06 | 855,88 |
| 8 | D | 5913,44 | 6746,99 | 843,37 |
| 8 | D | 6078,75 | 6563,5 | 820,44 |

| | | | | |
|---|---|---|---|---|
| 8 | D | 7660,41 | 5208,33 | 651,04 |
| 8 | D | 6131,21 | 6507,35 | 813,42 |
| 8 | D | 5502,31 | 7251,13 | 906,39 |
| 8 | D | 7427,03 | 5371,99 | 671,5 |
| 8 | D | 8972,82 | 4446,53 | 555,82 |
| 8 | D | 5630,51 | 7086,02 | 885,75 |
| 8 | D | 8096,86 | 4927,58 | 615,95 |
| 8 | D | 9965,23 | 4003,71 | 500,46 |
| 8 | D | 5105,68 | 7814,42 | 976,8 |
| 8 | D | 5382,1 | 7413,08 | 926,63 |
| 8 | D | 10288,85 | 3877,78 | 484,72 |
| 8 | D | 6370,32 | 6263,1 | 782,89 |
| 8 | D | 7320,51 | 5450,16 | 681,27 |
| 8 | D | 5239,85 | 7614,33 | 951,79 |
| 8 | D | 7833,79 | 5093,06 | 636,63 |
| 8 | D | 7709,06 | 5175,46 | 646,93 |
| 8 | D | 7878,85 | 5063,93 | 632,99 |
| 8 | D | 7611,61 | 5241,72 | 655,21 |
| 8 | D | 5634,23 | 7081,35 | 885,17 |
| 8 | D | 5470,88 | 7292,78 | 911,6 |
| 8 | D | 5879,69 | 6785,72 | 848,22 |
| 8 | D | 6004,74 | 6644,4 | 830,55 |
| 8 | D | 7275,57 | 5483,82 | 685,48 |
| 8 | D | 5609,44 | 7112,64 | 889,08 |
| 8 | D | 8966,58 | 4449,63 | 556,2 |
| 8 | D | 6436,47 | 6198,73 | 774,84 |
| 8 | D | 5852,09 | 6817,72 | 852,22 |
| 8 | D | 8714,3 | 4578,44 | 572,31 |
| 8 | D | 9088,94 | 4389,72 | 548,72 |
| 8 | D | 9214,94 | 4329,7 | 541,21 |
| 8 | D | 9103,07 | 4382,91 | 547,86 |
| 8 | D | 5561,65 | 7173,76 | 896,72 |

Cuadro C.18: LU 8 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 12 | D | 5627,37 | 7089,98 | 590,83 |
| 12 | D | 4139,74 | 9637,79 | 803,15 |
| 12 | D | 3866,68 | 10318,39 | 859,87 |
| 12 | D | 4179,82 | 9545,38 | 795,45 |
| 12 | D | 4199,32 | 9501,04 | 791,75 |

| | | | | |
|---|---|---|---|---|
| 12 | D | 6108,11 | 6531,96 | 544,33 |
| 12 | D | 3800,23 | 10498,82 | 874,9 |
| 12 | D | 6088,04 | 6553,49 | 546,12 |
| 12 | D | 5186,33 | 7692,9 | 641,07 |
| 12 | D | 3793,8 | 10516,62 | 876,38 |
| 12 | D | 6983,85 | 5712,88 | 476,07 |
| 12 | D | 4170,48 | 9566,75 | 797,23 |
| 12 | D | 5687,44 | 7015,1 | 584,59 |
| 12 | D | 4026,65 | 9908,46 | 825,71 |
| 12 | D | 4315,6 | 9245,05 | 770,42 |
| 12 | D | 3763,44 | 10601,45 | 883,45 |
| 12 | D | 3571,62 | 11170,84 | 930,9 |
| 12 | D | 3816,52 | 10454,02 | 871,17 |
| 12 | D | 4201,86 | 9495,29 | 791,27 |
| 12 | D | 5160,78 | 7730,99 | 644,25 |
| 12 | D | 3760,69 | 10609,21 | 884,1 |
| 12 | D | 4455,93 | 8953,89 | 746,16 |
| 12 | D | 7385,79 | 5401,98 | 450,17 |
| 12 | D | 3813,51 | 10462,27 | 871,86 |
| 12 | D | 3920,57 | 10176,56 | 848,05 |
| 12 | D | 4210,64 | 9475,49 | 789,62 |
| 12 | D | 3733,56 | 10686,29 | 890,52 |
| 12 | D | 4016,72 | 9932,97 | 827,75 |
| 12 | D | 3839,02 | 10392,73 | 866,06 |
| 12 | D | 6329,73 | 6303,26 | 525,27 |
| 12 | D | 7183,34 | 5554,23 | 462,85 |
| 12 | D | 4176,65 | 9552,62 | 796,05 |
| 12 | D | 6473,89 | 6162,9 | 513,57 |
| 12 | D | 4040,88 | 9873,58 | 822,8 |
| 12 | D | 6348,01 | 6285,11 | 523,76 |
| 12 | D | 8331,4 | 4788,86 | 399,07 |
| 12 | D | 5026,53 | 7937,47 | 661,46 |
| 12 | D | 3837,29 | 10397,41 | 866,45 |
| 12 | D | 4174,4 | 9557,76 | 796,48 |
| 12 | D | 6088,46 | 6553,04 | 546,09 |
| 12 | D | 3858,91 | 10339,16 | 861,6 |
| 12 | D | 4078,65 | 9782,14 | 815,18 |
| 12 | D | 5018,89 | 7949,55 | 662,46 |
| 12 | D | 3959,2 | 10077,28 | 839,77 |
| 12 | D | 4609,24 | 8656,08 | 721,34 |
| 12 | D | 5651,99 | 7059,09 | 588,26 |

| | | | | |
|---|---|---|---|---|
| 12 | D | 7260,7 | 5495,05 | 457,92 |
| 12 | D | 4093,08 | 9747,64 | 812,3 |
| 12 | D | 6924,22 | 5762,08 | 480,17 |
| 12 | D | 3773,32 | 10573,7 | 881,14 |

Cuadro C.19: LU 12 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 3686,65 | 10822,28 | 676,39 |
| 16 | D | 2918,62 | 13670,13 | 854,38 |
| 16 | D | 2915,06 | 13686,84 | 855,43 |
| 16 | D | 3960,05 | 10075,11 | 629,69 |
| 16 | D | 6939,55 | 5749,35 | 359,33 |
| 16 | D | 4454,23 | 8957,32 | 559,83 |
| 16 | D | 2888,25 | 13813,88 | 863,37 |
| 16 | D | 3795,59 | 10511,66 | 656,98 |
| 16 | D | 2808,36 | 14206,85 | 887,93 |
| 16 | D | 4319,48 | 9236,74 | 577,3 |
| 16 | D | 3767,07 | 10591,25 | 661,95 |
| 16 | D | 3772,56 | 10575,83 | 660,99 |
| 16 | D | 3847,23 | 10370,55 | 648,16 |
| 16 | D | 3438,71 | 11602,59 | 725,16 |
| 16 | D | 3146,76 | 12679,07 | 792,44 |
| 16 | D | 3295,64 | 12106,27 | 756,64 |
| 16 | D | 2826,47 | 14115,84 | 882,24 |
| 16 | D | 4335,47 | 9202,68 | 575,17 |
| 16 | D | 3592,09 | 11107,18 | 694,2 |
| 16 | D | 4255,02 | 9376,68 | 586,04 |
| 16 | D | 4262,11 | 9361,08 | 585,07 |
| 16 | D | 4410,52 | 9046,08 | 565,38 |
| 16 | D | 4097,83 | 9736,35 | 608,52 |
| 16 | D | 4026,56 | 9908,7 | 619,29 |
| 16 | D | 3156,96 | 12638,1 | 789,88 |
| 16 | D | 2444,89 | 16318,94 | 1019,93 |
| 16 | D | 2822,43 | 14136,01 | 883,5 |
| 16 | D | 3031,93 | 13159,26 | 822,45 |
| 16 | D | 6963,34 | 5729,71 | 358,11 |
| 16 | D | 3427,69 | 11639,9 | 727,49 |
| 16 | D | 3384,29 | 11789,15 | 736,82 |
| 16 | D | 4926,55 | 8098,55 | 506,16 |
| 16 | D | 4926,06 | 8099,36 | 506,21 |

| | | | | |
|---|---|---|---|---|
| 16 | D | 3088,87 | 12916,66 | 807,29 |
| 16 | D | 2567,32 | 15540,69 | 971,29 |
| 16 | D | 3089,68 | 12913,28 | 807,08 |
| 16 | D | 4629,73 | 8617,77 | 538,61 |
| 16 | D | 3693,81 | 10801,31 | 675,08 |
| 16 | D | 2897,03 | 13772,03 | 860,75 |
| 16 | D | 3062,86 | 13026,35 | 814,15 |
| 16 | D | 3078,52 | 12960,11 | 810,01 |
| 16 | D | 2748,19 | 14517,88 | 907,37 |
| 16 | D | 3358,99 | 11877,96 | 742,37 |
| 16 | D | 2585,06 | 15434,07 | 964,63 |
| 16 | D | 2433,07 | 16398,17 | 1024,89 |
| 16 | D | 6145,4 | 6492,32 | 405,77 |
| 16 | D | 3108,81 | 12833,82 | 802,11 |
| 16 | D | 2497,08 | 15977,83 | 998,61 |
| 16 | D | 4655,39 | 8570,27 | 535,64 |
| 16 | D | 3425,45 | 11647,48 | 727,97 |
| 16 | D | 5956,67 | 6698,02 | 418,63 |

Cuadro C.20: LU 16 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 24 | D | 2189,48 | 18222,56 | 759,27 |
| 24 | D | 1848,16 | 21587,91 | 899,5 |
| 24 | D | 2368,44 | 16845,68 | 701,9 |
| 24 | D | 2720,74 | 14664,36 | 611,01 |
| 24 | D | 2409,84 | 16556,23 | 689,84 |
| 24 | D | 2856,89 | 13965,53 | 581,9 |
| 24 | D | 2306,64 | 17297,02 | 720,71 |
| 24 | D | 2800,56 | 14246,39 | 593,6 |
| 24 | D | 2782,04 | 14341,27 | 597,55 |
| 24 | D | 1816,83 | 21960,19 | 915,01 |
| 24 | D | 1820,29 | 21918,39 | 913,27 |
| 24 | D | 1863,58 | 21409,3 | 892,05 |
| 24 | D | 1904,47 | 20949,65 | 872,9 |
| 24 | D | 2781,96 | 14341,67 | 597,57 |
| 24 | D | 2207,36 | 18074,99 | 753,12 |
| 24 | D | 1668,57 | 23911,42 | 996,31 |
| 24 | D | 2014,43 | 19806,03 | 825,25 |
| 24 | D | 1923,01 | 20747,63 | 864,48 |
| 24 | D | 1862,72 | 21419,14 | 892,46 |

| | | | | |
|---|---|---|---|---|
| 24 | D | 2916,99 | 13677,75 | 569,91 |
| 24 | D | 1856,37 | 21492,43 | 895,52 |
| 24 | D | 1947,21 | 20489,78 | 853,74 |
| 24 | D | 1872,08 | 21312,1 | 888 |
| 24 | D | 2095,58 | 19039,11 | 793,3 |
| 24 | D | 1864,79 | 21395,36 | 891,47 |
| 24 | D | 2866,79 | 13917,28 | 579,89 |
| 24 | D | 2713,81 | 14701,84 | 612,58 |
| 24 | D | 1857,5 | 21479,37 | 894,97 |
| 24 | D | 2583,1 | 15445,76 | 643,57 |
| 24 | D | 2596,37 | 15366,81 | 640,28 |
| 24 | D | 1940,32 | 20562,55 | 856,77 |
| 24 | D | 2824,23 | 14127,03 | 588,63 |
| 24 | D | 1858,01 | 21473,48 | 894,73 |
| 24 | D | 2600,17 | 15344,36 | 639,35 |
| 24 | D | 1961,25 | 20343,08 | 847,63 |
| 24 | D | 2474,78 | 16121,82 | 671,74 |
| 24 | D | 2780,39 | 14349,78 | 597,91 |
| 24 | D | 2526,25 | 15793,36 | 658,06 |
| 24 | D | 1883,11 | 21187,3 | 882,8 |
| 24 | D | 2677,94 | 14898,72 | 620,78 |
| 24 | D | 1835,93 | 21731,78 | 905,49 |
| 24 | D | 3875,57 | 10294,72 | 428,95 |
| 24 | D | 1683,07 | 23705,46 | 987,73 |
| 24 | D | 2790,18 | 14299,43 | 595,81 |
| 24 | D | 2259,29 | 17659,53 | 735,81 |
| 24 | D | 2675,77 | 14910,83 | 621,28 |
| 24 | D | 1863,43 | 21410,97 | 892,12 |
| 24 | D | 1904,79 | 20946,05 | 872,75 |
| 24 | D | 1755,94 | 22721,65 | 946,74 |

Cuadro C.21: LU 24 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 28 | D | 2380,62 | 16759,47 | 598,55 |
| 28 | D | 2747,35 | 14522,32 | 518,65 |
| 28 | D | 1564,9 | 25495,53 | 910,55 |
| 28 | D | 3101,16 | 12865,5 | 459,48 |
| 28 | D | 1375,37 | 29008,87 | 1036,03 |
| 28 | D | 1561,26 | 25554,98 | 912,68 |
| 28 | D | 1503,63 | 26534,47 | 947,66 |

| | | | | |
|---|---|---|---|---|
| 28 | D | 4304 | 9269,96 | 331,07 |
| 28 | D | 5525,57 | 7220,6 | 257,88 |
| 28 | D | 1583,09 | 25202,55 | 900,09 |
| 28 | D | 2885,07 | 13829,12 | 493,9 |
| 28 | D | 1535,72 | 25979,95 | 927,86 |
| 28 | D | 2687,56 | 14845,44 | 530,19 |
| 28 | D | 2565,79 | 15549,96 | 555,36 |
| 28 | D | 1588,12 | 25122,75 | 897,24 |
| 28 | D | 2330,91 | 17116,87 | 611,32 |
| 28 | D | 2532,72 | 15752,98 | 562,61 |
| 28 | D | 3613,49 | 11041,39 | 394,34 |
| 28 | D | 2635,3 | 15139,8 | 540,71 |
| 28 | D | 1602,54 | 24896,63 | 889,17 |
| 28 | D | 1538,09 | 25939,98 | 926,43 |
| 28 | D | 2207,08 | 18077,22 | 645,62 |
| 28 | D | 4393,9 | 9080,3 | 324,3 |
| 28 | D | 1745,98 | 22851,26 | 816,12 |
| 28 | D | 2401,55 | 16613,42 | 593,34 |
| 28 | D | 1952,48 | 20434,53 | 729,8 |
| 28 | D | 1851,13 | 21553,24 | 769,76 |
| 28 | D | 1977,25 | 20178,54 | 720,66 |
| 28 | D | 2395,07 | 16658,34 | 594,94 |
| 28 | D | 2251 | 17724,52 | 633,02 |
| 28 | D | 1761,06 | 22655,68 | 809,13 |
| 28 | D | 2433,28 | 16396,75 | 585,6 |
| 28 | D | 2730,02 | 14614,53 | 521,95 |
| 28 | D | 2267,98 | 17591,86 | 628,28 |
| 28 | D | 1661,7 | 24010,3 | 857,51 |
| 28 | D | 1999,62 | 19952,73 | 712,6 |
| 28 | D | 1604,24 | 24870,25 | 888,22 |
| 28 | D | 2769,15 | 14407,99 | 514,57 |
| 28 | D | 2329,99 | 17123,63 | 611,56 |
| 28 | D | 2432,02 | 16405,29 | 585,9 |
| 28 | D | 1966,83 | 20285,42 | 724,48 |
| 28 | D | 2897,66 | 13769,02 | 491,75 |
| 28 | D | 2602,25 | 15332,09 | 547,57 |
| 28 | D | 2437,81 | 16366,29 | 584,51 |
| 28 | D | 1995,34 | 19995,56 | 714,13 |
| 28 | D | 2778,89 | 14357,5 | 512,77 |
| 28 | D | 2393 | 16672,74 | 595,45 |
| 28 | D | 3071,16 | 12991,14 | 463,97 |

Cuadro C.22: LU 28 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 2182,9 | 18277,47 | 571,17 |
| 32 | D | 2384,77 | 16730,29 | 522,82 |
| 32 | D | 2498,71 | 15967,38 | 498,98 |
| 32 | D | 1680,35 | 23743,83 | 741,99 |
| 32 | D | 2196,31 | 18165,91 | 567,68 |
| 32 | D | 1819,11 | 21932,67 | 685,4 |
| 32 | D | 3448,99 | 11568,02 | 361,5 |
| 32 | D | 1390,29 | 28697,58 | 896,8 |
| 32 | D | 1945,28 | 20510,09 | 640,94 |
| 32 | D | 3048,24 | 13088,86 | 409,03 |
| 32 | D | 1703,93 | 23415,26 | 731,73 |
| 32 | D | 2383,01 | 16742,68 | 523,21 |
| 32 | D | 2656,4 | 15019,53 | 469,36 |
| 32 | D | 2331,78 | 17110,5 | 534,7 |
| 32 | D | 3094,45 | 12893,39 | 402,92 |
| 32 | D | 4429,84 | 9006,63 | 281,46 |
| 32 | D | 3848,59 | 10366,89 | 323,97 |
| 32 | D | 3271,77 | 12194,59 | 381,08 |
| 32 | D | 1341,86 | 29733,24 | 929,16 |
| 32 | D | 2379,67 | 16766,14 | 523,94 |
| 32 | D | 2283,74 | 17470,44 | 545,95 |
| 32 | D | 2206,73 | 18080,13 | 565 |
| 32 | D | 1358 | 29379,95 | 918,12 |
| 32 | D | 2157,18 | 18495,45 | 577,98 |
| 32 | D | 2387,58 | 16710,6 | 522,21 |
| 32 | D | 2076,68 | 19212,35 | 600,39 |
| 32 | D | 2278,57 | 17510,06 | 547,19 |
| 32 | D | 3831,7 | 10412,58 | 325,39 |
| 32 | D | 2096,67 | 19029,2 | 594,66 |
| 32 | D | 2301,82 | 17333,21 | 541,66 |
| 32 | D | 1655,13 | 24105,64 | 753,3 |
| 32 | D | 2369,61 | 16837,36 | 526,17 |
| 32 | D | 1677,95 | 23777,76 | 743,05 |
| 32 | D | 2016,51 | 19785,68 | 618,3 |
| 32 | D | 2393,9 | 16666,48 | 520,83 |
| 32 | D | 2184,18 | 18266,77 | 570,84 |

Cuadro C.23: LU 32 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 36 | D | 1289,1 | 30950,23 | 859,73 |
| 36 | D | 1503,63 | 26534,37 | 737,07 |
| 36 | D | 1220,04 | 32702,02 | 908,39 |
| 36 | D | 1184,23 | 33690,94 | 935,86 |
| 36 | D | 1561,93 | 25544,02 | 709,56 |
| 36 | D | 1347,2 | 29615,54 | 822,65 |
| 36 | D | 1160,14 | 34390,73 | 955,3 |
| 36 | D | 1949,79 | 20462,64 | 568,41 |
| 36 | D | 2396,61 | 16647,68 | 462,44 |
| 36 | D | 1728,54 | 23081,92 | 641,16 |
| 36 | D | 1807,18 | 22077,5 | 613,26 |
| 36 | D | 2410,83 | 16549,43 | 459,71 |
| 36 | D | 2032,58 | 19629,19 | 545,26 |
| 36 | D | 1879,96 | 21222,71 | 589,52 |
| 36 | D | 1920,97 | 20769,67 | 576,94 |
| 36 | D | 1779,18 | 22424,94 | 622,91 |
| 36 | D | 1378,8 | 28936,61 | 803,79 |
| 36 | D | 1977,82 | 20172,65 | 560,35 |
| 36 | D | 2002,42 | 19924,89 | 553,47 |
| 36 | D | 1174,37 | 33973,78 | 943,72 |
| 36 | D | 1480,9 | 26941,72 | 748,38 |
| 36 | D | 1889,51 | 21115,47 | 586,54 |
| 36 | D | 1197,65 | 33313,59 | 925,38 |
| 36 | D | 1935,12 | 20617,77 | 572,72 |
| 36 | D | 1918,17 | 20799,95 | 577,78 |
| 36 | D | 1742,04 | 22903,06 | 636,2 |
| 36 | D | 1349,14 | 29572,84 | 821,47 |
| 36 | D | 1786,47 | 22333,34 | 620,37 |
| 36 | D | 1287,42 | 30990,59 | 860,85 |
| 36 | D | 1936,34 | 20604,82 | 572,36 |
| 36 | D | 1166,04 | 34216,48 | 950,46 |
| 36 | D | 2000,17 | 19947,26 | 554,09 |
| 36 | D | 1252,66 | 31850,58 | 884,74 |

Cuadro C.24: LU 36 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 40 | D | 1865,47 | 21387,58 | 534,69 |
| 40 | D | 1613,88 | 24721,7 | 618,04 |
| 40 | D | 1733,4 | 23017,11 | 575,43 |
| 40 | D | 1518,12 | 26281,2 | 657,03 |
| 40 | D | 2563,47 | 15564,04 | 389,1 |
| 40 | D | 1582,95 | 25204,78 | 630,12 |
| 40 | D | 1696,87 | 23512,71 | 587,82 |
| 40 | D | 1389,27 | 28718,57 | 717,96 |
| 40 | D | 1047,95 | 38072,34 | 951,81 |
| 40 | D | 2410,81 | 16549,58 | 413,74 |
| 40 | D | 1630,76 | 24465,87 | 611,65 |
| 40 | D | 1566,06 | 25476,67 | 636,92 |
| 40 | D | 1157,12 | 34480,26 | 862,01 |
| 40 | D | 1307,33 | 30518,62 | 762,97 |
| 40 | D | 1667,84 | 23921,87 | 598,05 |
| 40 | D | 1607,93 | 24813,29 | 620,33 |
| 40 | D | 1732,87 | 23024,2 | 575,6 |
| 40 | D | 1061,8 | 37575,72 | 939,39 |
| 40 | D | 2578,04 | 15476,07 | 386,9 |
| 40 | D | 1606,05 | 24842,28 | 621,06 |
| 40 | D | 1053,79 | 37861,41 | 946,54 |
| 40 | D | 1773,53 | 22496,3 | 562,41 |
| 40 | D | 1058 | 37710,6 | 942,76 |
| 40 | D | 1599,05 | 24950,99 | 623,77 |
| 40 | D | 1231,93 | 32386,52 | 809,66 |
| 40 | D | 1239,89 | 32178,64 | 804,47 |
| 40 | D | 1305,33 | 30565,45 | 764,14 |
| 40 | D | 1085,95 | 36740,18 | 918,5 |
| 40 | D | 1271,48 | 31379,08 | 784,48 |
| 40 | D | 1335,84 | 29867,24 | 746,68 |
| 40 | D | 3013,57 | 13239,41 | 330,99 |
| 40 | D | 1485,97 | 26849,67 | 671,24 |
| 40 | D | 1388,74 | 28729,53 | 718,24 |
| 40 | D | 1535,21 | 25988,55 | 649,71 |
| 40 | D | 1693,44 | 23560,34 | 589,01 |
| 40 | D | 1642,49 | 24291,06 | 607,28 |
| 40 | D | 1502,18 | 26560,1 | 664 |
| 40 | D | 1616,97 | 24674,53 | 616,86 |
| 40 | D | 1668,93 | 23906,34 | 597,66 |

| | | | | |
|---|---|---|---|---|
| 40 | D | 1322,86 | 30160,35 | 754,01 |

Cuadro C.25: LU 40 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 48 | D | 21797,55 | 1830,39 | 38,13 |
| 48 | D | 1261,68 | 31622,86 | 658,81 |
| 48 | D | 1543,26 | 25853 | 538,6 |
| 48 | D | 1354,72 | 29451,06 | 613,56 |
| 48 | D | 1723,98 | 23142,92 | 482,14 |
| 48 | D | 1075 | 37114,22 | 773,21 |
| 48 | D | 1783,47 | 22370,94 | 466,06 |
| 48 | D | 1959,81 | 20358,07 | 424,13 |
| 48 | D | 1122,03 | 35558,74 | 740,81 |
| 48 | D | 1352,25 | 29504,76 | 614,68 |
| 48 | D | 1504,25 | 26523,42 | 552,57 |
| 48 | D | 961,96 | 41475,78 | 864,08 |
| 48 | D | 998,71 | 39949,64 | 832,28 |
| 48 | D | 844,94 | 47220,1 | 983,75 |
| 48 | D | 1168,89 | 34133,21 | 711,11 |
| 48 | D | 964,01 | 41387,57 | 862,24 |
| 48 | D | 1356,02 | 29422,88 | 612,98 |
| 48 | D | 844,35 | 47252,58 | 984,43 |
| 48 | D | 955,21 | 41768,72 | 870,18 |
| 48 | D | 1090,22 | 36596,22 | 762,42 |
| 48 | D | 1197,4 | 33320,58 | 694,18 |
| 48 | D | 847,07 | 47101,17 | 981,27 |
| 48 | D | 861,88 | 46291,67 | 964,41 |
| 48 | D | 1165,06 | 34245,4 | 713,45 |
| 48 | D | 1300,86 | 30670,37 | 638,97 |
| 48 | D | 836,39 | 47702,68 | 993,81 |
| 48 | D | 1146,2 | 34808,88 | 725,18 |
| 48 | D | 977,39 | 40821,07 | 850,44 |
| 48 | D | 975,32 | 40907,42 | 852,24 |
| 48 | D | 1181,81 | 33759,95 | 703,33 |
| 48 | D | 833,01 | 47896,2 | 997,84 |
| 48 | D | 1216,7 | 32791,8 | 683,16 |
| 48 | D | 830,41 | 48046,15 | 1000,96 |
| 48 | D | 1185,16 | 33664,54 | 701,34 |
| 48 | D | 837,87 | 47618,34 | 992,05 |
| 48 | D | 1191,72 | 33479,23 | 697,48 |

| | | | | |
|---|---|---|---|---|
| 48 | D | 945,55 | 42195,36 | 879,07 |
| 48 | D | 1157,79 | 34460,37 | 717,92 |
| 48 | D | 1180,4 | 33800,22 | 704,17 |
| 48 | D | 1583,43 | 25197,09 | 524,94 |

Cuadro C.26: LU 48 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 54 | D | 786,46 | 50731,04 | 939,46 |
| 54 | D | 766,09 | 52079,95 | 964,44 |
| 54 | D | 946,12 | 42169,94 | 780,92 |
| 54 | D | 1067,79 | 37364,93 | 691,94 |
| 54 | D | 1069,85 | 37292,89 | 690,61 |
| 54 | D | 1057,12 | 37741,96 | 698,93 |
| 54 | D | 1063,42 | 37518,59 | 694,79 |
| 54 | D | 1093,65 | 36481,55 | 675,58 |
| 54 | D | 1072,46 | 37202,2 | 688,93 |
| 54 | D | 1597,41 | 24976,68 | 462,53 |
| 54 | D | 1146,84 | 34789,6 | 644,25 |
| 54 | D | 1191,66 | 33480,91 | 620,02 |
| 54 | D | 1422,64 | 28044,9 | 519,35 |
| 54 | D | 1266,56 | 31501,05 | 583,35 |
| 54 | D | 1096,91 | 36373,13 | 673,58 |
| 54 | D | 934,86 | 42677,92 | 790,33 |
| 54 | D | 681,04 | 58583,72 | 1084,88 |
| 54 | D | 944,22 | 42254,74 | 782,5 |
| 54 | D | 1535,34 | 25986,32 | 481,23 |
| 54 | D | 1294,18 | 30828,84 | 570,9 |
| 54 | D | 1503,66 | 26533,96 | 491,37 |
| 54 | D | 1426,33 | 27972,53 | 518,01 |
| 54 | D | 1906,09 | 20931,77 | 387,63 |
| 54 | D | 1993,99 | 20009,09 | 370,54 |
| 54 | D | 950,79 | 41962,74 | 777,09 |
| 54 | D | 1735,56 | 22988,51 | 425,71 |
| 54 | D | 909,9 | 43848,57 | 812,01 |
| 54 | D | 912,66 | 43715,95 | 809,55 |
| 54 | D | 975,58 | 40896,54 | 757,34 |
| 54 | D | 961,37 | 41500,95 | 768,54 |
| 54 | D | 1070,34 | 37275,92 | 690,29 |
| 54 | D | 1083,02 | 36839,58 | 682,21 |
| 54 | D | 781,26 | 51068,96 | 945,72 |

| | | | | |
|---|---|---|---|---|
| 54 | D | 948,43 | 42067,27 | 779,02 |
| 54 | D | 1117,61 | 35699,29 | 661,1 |
| 54 | D | 972,83 | 41012,35 | 759,49 |

Cuadro C.27: LU 54 Euler

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 56 | D | 684,88 | 58255,59 | 1040,28 |
| 56 | D | 811,15 | 49186,62 | 878,33 |
| 56 | D | 1217,28 | 32776,34 | 585,29 |
| 56 | D | 938,9 | 42494,54 | 758,83 |
| 56 | D | 935,91 | 42630,13 | 761,25 |
| 56 | D | 1746,72 | 22841,63 | 407,89 |
| 56 | D | 1008,7 | 39553,99 | 706,32 |
| 56 | D | 1022,92 | 39003,81 | 696,5 |
| 56 | D | 647,45 | 61623 | 1100,41 |
| 56 | D | 1004,37 | 39724,36 | 709,36 |
| 56 | D | 1016,6 | 39246,28 | 700,83 |
| 56 | D | 1010,93 | 39466,37 | 704,76 |
| 56 | D | 645,09 | 61848,32 | 1104,43 |
| 56 | D | 647,3 | 61637,17 | 1100,66 |
| 56 | D | 942,19 | 42345,87 | 756,18 |
| 56 | D | 915,72 | 43570,2 | 778,04 |
| 56 | D | 1609,5 | 24789,05 | 442,66 |
| 56 | D | 2211,6 | 18040,3 | 322,15 |
| 56 | D | 805,92 | 49506,27 | 884,04 |
| 56 | D | 853,7 | 46735,33 | 834,56 |
| 56 | D | 915,88 | 43562,37 | 777,9 |
| 56 | D | 786,62 | 50720,86 | 905,73 |
| 56 | D | 1056,51 | 37764,03 | 674,36 |
| 56 | D | 1058,07 | 37708,2 | 673,36 |
| 56 | D | 1428,06 | 27938,52 | 498,9 |
| 56 | D | 940,33 | 42429,93 | 757,68 |
| 56 | D | 1342 | 29730,22 | 530,9 |
| 56 | D | 1042 | 38289,58 | 683,74 |
| 56 | D | 1058,3 | 37700,09 | 673,22 |
| 56 | D | 1620,14 | 24626,23 | 439,75 |
| 56 | D | 1050,24 | 37989,35 | 678,38 |
| 56 | D | 1226,72 | 32524,08 | 580,79 |
| 56 | D | 1553,18 | 25687,98 | 458,71 |

Cuadro C.28: LU 56 Euler

### C.1.4.2. Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 6 | D | 3508,61 | 11371,45 | 1895,24 |
| 6 | D | 3422,03 | 11659,15 | 1943,19 |
| 6 | D | 3707,05 | 10762,71 | 1793,78 |
| 6 | D | 3544,24 | 11257,12 | 1876,19 |
| 6 | D | 3554,5 | 11224,63 | 1870,77 |
| 6 | D | 3713,92 | 10742,8 | 1790,47 |
| 6 | D | 3669,7 | 10872,27 | 1812,05 |
| 6 | D | 3711,13 | 10750,9 | 1791,82 |

Cuadro C.29: LU 6 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 8 | D | 2759,64 | 14457,68 | 1807,21 |
| 8 | D | 3022,81 | 13198,93 | 1649,87 |
| 8 | D | 2701,69 | 14767,75 | 1845,97 |
| 8 | D | 2611,32 | 15278,84 | 1909,85 |
| 8 | D | 2710,09 | 14721,97 | 1840,25 |

Cuadro C.30: LU 8 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 12 | D | 3668,64 | 10875,4 | 906,28 |
| 12 | D | 2295,07 | 17384,22 | 1448,69 |
| 12 | D | 1848,01 | 21589,72 | 1799,14 |
| 12 | D | 1760,39 | 22664,19 | 1888,68 |
| 12 | D | 2282,85 | 17477,22 | 1456,44 |
| 12 | D | 1811,5 | 22024,77 | 1835,4 |

Cuadro C.31: LU 12 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 16 | D | 1436,64 | 27771,79 | 1735,74 |
| 16 | D | 1439,48 | 27716,82 | 1732,3 |
| 16 | D | 1438,81 | 27729,83 | 1733,11 |

| | | | | |
|---|---|---|---|---|
| 16 | D | 1412,12 | 28253,99 | 1765,87 |
| 16 | D | 1424,82 | 28002,16 | 1750,13 |
| 16 | D | 1874,08 | 21289,38 | 1330,59 |
| 16 | D | 1440,67 | 27693,94 | 1730,87 |
| 16 | D | 1431,7 | 27867,48 | 1741,72 |

Cuadro C.32: LU 16 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 24 | D | 990,79 | 40268,78 | 1677,87 |
| 24 | D | 971,53 | 41067,25 | 1711,14 |
| 24 | D | 958,12 | 41641,76 | 1735,07 |
| 24 | D | 1159,51 | 34409,3 | 1433,72 |
| 24 | D | 1359,36 | 29350,45 | 1222,94 |
| 24 | D | 1014,26 | 39336,87 | 1639,04 |
| 24 | D | 959,71 | 41572,75 | 1732,2 |
| 24 | D | 1394,65 | 28607,8 | 1191,99 |

Cuadro C.33: LU 24 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 28 | D | 870,28 | 45844,74 | 1637,31 |
| 28 | D | 860,17 | 46383,98 | 1656,57 |
| 28 | D | 1020,15 | 39109,9 | 1396,78 |
| 28 | D | 857,09 | 46550,31 | 1662,51 |

Cuadro C.34: LU 28 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 32 | D | 763,93 | 52227,47 | 1632,11 |
| 32 | D | 776,37 | 51390,1 | 1605,94 |
| 32 | D | 790,2 | 50490,62 | 1577,83 |
| 32 | D | 776,14 | 51405,59 | 1606,42 |

Cuadro C.35: LU 32 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 36 | D | 833,82 | 47849,78 | 1329,16 |
| 36 | D | 1024,5 | 38943,92 | 1081,78 |

| | | | | |
|---|---|---|---|---|
| 36 | D | 866,05 | 46069,01 | 1279,69 |

Cuadro C.36: LU 36 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 40 | D | 1019,77 | 39124,39 | 978,11 |
| 40 | D | 943,92 | 42268,29 | 1056,71 |
| 40 | D | 896,35 | 44511,6 | 1112,79 |
| 40 | D | 639,65 | 62374,93 | 1559,37 |
| 40 | D | 623,03 | 64038,71 | 1600,97 |
| 40 | D | 1039,81 | 38370,42 | 959,26 |
| 40 | D | 640,63 | 62278,87 | 1556,97 |

Cuadro C.37: LU 40 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 48 | D | 847,04 | 47102,68 | 981,31 |
| 48 | D | 815,09 | 48948,92 | 1019,77 |
| 48 | D | 816,76 | 48848,98 | 1017,69 |
| 48 | D | 807,32 | 49420,05 | 1029,58 |
| 48 | D | 809,61 | 49280,39 | 1026,67 |

Cuadro C.38: LU 48 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 54 | D | 763,94 | 52226,49 | 967,16 |
| 54 | D | 834,64 | 47802,41 | 885,23 |
| 54 | D | 715,86 | 55734,33 | 1032,12 |
| 54 | D | 845,83 | 47170,04 | 873,52 |
| 54 | D | 864,93 | 46128,41 | 854,23 |
| 54 | D | 841,87 | 47391,93 | 877,63 |
| 54 | D | 861,81 | 46295,33 | 857,32 |

Cuadro C.39: LU 54 Ordis

| Total processes | Class | Time in seconds | Mop/s total | Mop/s process |
|---|---|---|---|---|
| 56 | D | 717,07 | 55640,36 | 993,58 |
| 56 | D | 756,62 | 52731,74 | 941,64 |
| 56 | D | 844,26 | 47257,97 | 843,89 |

Cuadro C.40: Ordis LU 56

## C.2. GPU benchmarks

### C.2.1. Clarke

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 138,9685 | 0,2408 | 12,22 |
| 1 | 139,1287 | 0,2536 | 12,21 |
| 2 | 139,1259 | 0,2526 | 12,21 |
| 3 | 139,133 | 0,2595 | 12,21 |
| 4 | 139,1429 | 0,2691 | 12,21 |
| 5 | 139,145 | 0,2717 | 12,21 |
| 6 | 139,1343 | 0,2718 | 12,21 |
| 7 | 139,1328 | 0,2622 | 12,21 |
| 8 | 139,1443 | 0,2656 | 12,21 |
| 9 | 139,1425 | 0,2616 | 12,21 |
| 10 | 139,1488 | 0,2733 | 12,21 |
| 11 | 139,0674 | 0,2701 | 12,21 |
| 12 | 139,1554 | 0,2751 | 12,2 |
| 13 | 139,1354 | 0,2539 | 12,21 |
| 14 | 139,1418 | 0,263 | 12,21 |
| 15 | 139,1327 | 0,2661 | 12,21 |
| 16 | 139,1362 | 0,2709 | 12,21 |
| 17 | 139,1429 | 0,2662 | 12,21 |
| 18 | 139,1411 | 0,2535 | 12,21 |
| 19 | 139,14 | 0,2591 | 12,21 |
| 20 | 139,1408 | 0,2579 | 12,21 |
| 21 | 139,1471 | 0,2686 | 12,21 |
| 22 | 139,1125 | 0,2603 | 12,21 |
| 23 | 139,1328 | 0,2559 | 12,21 |
| 24 | 139,1362 | 0,2624 | 12,21 |
| 25 | 139,1464 | 0,2671 | 12,21 |
| 26 | 139,1407 | 0,272 | 12,21 |
| 27 | 139,1383 | 0,261 | 12,21 |
| 28 | 139,1313 | 0,255 | 12,21 |
| 29 | 139,1416 | 0,2614 | 12,21 |
| 30 | 139,1401 | 0,2604 | 12,21 |
| 31 | 139,1365 | 0,2686 | 12,21 |

| | | | |
|---|---|---|---|
| 32 | 139,1336 | 0,2551 | 12,21 |
| 33 | 139,119 | 0,2621 | 12,21 |
| 34 | 139,1356 | 0,2681 | 12,21 |
| 35 | 139,1378 | 0,2604 | 12,21 |
| 36 | 139,1467 | 0,2707 | 12,21 |
| 37 | 139,1415 | 0,2751 | 12,21 |
| 38 | 139,1421 | 0,2715 | 12,21 |
| 39 | 139,1471 | 0,2639 | 12,21 |
| 40 | 139,1371 | 0,261 | 12,21 |
| 41 | 139,1429 | 0,2581 | 12,21 |
| 42 | 139,1392 | 0,2649 | 12,21 |
| 43 | 139,1378 | 0,2714 | 12,21 |
| 44 | 139,1365 | 0,27 | 12,21 |
| 45 | 139,1322 | 0,2768 | 12,21 |
| 46 | 139,1291 | 0,2624 | 12,21 |
| 47 | 139,1432 | 0,2688 | 12,21 |
| 48 | 139,142 | 0,2676 | 12,21 |
| 49 | 139,1389 | 0,27 | 12,21 |
| 50 | 139,1372 | 0,2653 | 12,21 |
| 51 | 139,1502 | 0,2654 | 12,2 |
| 52 | 139,1489 | 0,2784 | 12,21 |
| 53 | 139,1466 | 0,2688 | 12,21 |
| 54 | 139,1524 | 0,2723 | 12,2 |
| 55 | 139,1294 | 0,2654 | 12,21 |
| 56 | 139,1484 | 0,2593 | 12,21 |
| 57 | 139,1517 | 0,2756 | 12,2 |
| 58 | 139,1441 | 0,27 | 12,21 |
| 59 | 139,149 | 0,2796 | 12,21 |
| 60 | 139,1513 | 0,2715 | 12,2 |
| 61 | 139,1474 | 0,2665 | 12,21 |
| 62 | 139,1368 | 0,2658 | 12,21 |
| 63 | 139,1361 | 0,256 | 12,21 |
| 64 | 139,1335 | 0,2539 | 12,21 |
| 65 | 139,1379 | 0,2709 | 12,21 |
| 66 | 139,1327 | 0,2601 | 12,21 |
| 67 | 139,1384 | 0,2555 | 12,21 |
| 68 | 139,1359 | 0,258 | 12,21 |
| 69 | 139,1341 | 0,2538 | 12,21 |
| 70 | 139,1345 | 0,2609 | 12,21 |
| 71 | 139,126 | 0,2497 | 12,21 |
| 72 | 139,129 | 0,2494 | 12,21 |

| | | | |
|---|---|---|---|
| 73 | 139,1238 | 0,2459 | 12,21 |
| 74 | 139,1326 | 0,2595 | 12,21 |
| 75 | 139,1298 | 0,2636 | 12,21 |
| 76 | 139,1291 | 0,2581 | 12,21 |
| 77 | 139,1343 | 0,2592 | 12,21 |
| 78 | 139,1393 | 0,2571 | 12,21 |
| 79 | 139,1391 | 0,2539 | 12,21 |
| 80 | 139,1381 | 0,2465 | 12,21 |
| 81 | 139,143 | 0,2514 | 12,21 |
| 82 | 139,1381 | 0,2594 | 12,21 |
| 83 | 139,1402 | 0,2643 | 12,21 |
| 84 | 139,1309 | 0,2542 | 12,21 |
| 85 | 139,1348 | 0,2522 | 12,21 |
| 86 | 139,1403 | 0,2598 | 12,21 |
| 87 | 139,1318 | 0,2568 | 12,21 |
| 88 | 139,1278 | 0,2458 | 12,21 |
| 89 | 139,1335 | 0,2549 | 12,21 |
| 90 | 139,1352 | 0,2597 | 12,21 |
| 91 | 139,1404 | 0,2547 | 12,21 |
| 92 | 139,1459 | 0,2536 | 12,21 |
| 93 | 139,1397 | 0,2572 | 12,21 |
| 94 | 139,1425 | 0,2684 | 12,21 |
| 95 | 139,1538 | 0,2664 | 12,2 |
| 96 | 139,1454 | 0,2588 | 12,21 |
| 97 | 139,1434 | 0,2542 | 12,21 |
| 98 | 139,1467 | 0,2674 | 12,21 |
| 99 | 139,1303 | 0,2535 | 12,21 |

Cuadro C.41: Clarke con tamaño de bloque 4

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 56,7996 | 0,2513 | 29,9 |
| 1 | 56,8206 | 0,2468 | 29,89 |
| 2 | 56,8275 | 0,2454 | 29,89 |
| 3 | 56,8257 | 0,2451 | 29,89 |
| 4 | 56,8276 | 0,2467 | 29,89 |
| 5 | 56,8289 | 0,2474 | 29,88 |
| 6 | 56,8297 | 0,2452 | 29,88 |
| 7 | 56,832 | 0,2513 | 29,88 |
| 8 | 56,8258 | 0,249 | 29,89 |
| 9 | 56,8285 | 0,2528 | 29,89 |

| | | | |
|---|---|---|---|
| 10 | 56,8298 | 0,2509 | 29,88 |
| 11 | 56,8154 | 0,255 | 29,89 |
| 12 | 56,8296 | 0,2485 | 29,88 |
| 13 | 56,8268 | 0,245 | 29,89 |
| 14 | 56,8268 | 0,2485 | 29,89 |
| 15 | 56,8257 | 0,2491 | 29,89 |
| 16 | 56,8261 | 0,2474 | 29,89 |
| 17 | 56,8283 | 0,2521 | 29,89 |
| 18 | 56,8252 | 0,2482 | 29,89 |
| 19 | 56,8206 | 0,245 | 29,89 |
| 20 | 56,8199 | 0,2423 | 29,89 |
| 21 | 56,8236 | 0,2443 | 29,89 |
| 22 | 56,817 | 0,247 | 29,89 |
| 23 | 56,8202 | 0,2436 | 29,89 |
| 24 | 56,8264 | 0,2483 | 29,89 |
| 25 | 56,8262 | 0,2482 | 29,89 |
| 26 | 56,8288 | 0,2526 | 29,88 |
| 27 | 56,8238 | 0,2481 | 29,89 |
| 28 | 56,8272 | 0,2495 | 29,89 |
| 29 | 56,8239 | 0,2517 | 29,89 |
| 30 | 56,832 | 0,2561 | 29,88 |
| 31 | 56,83 | 0,2501 | 29,88 |
| 32 | 56,8324 | 0,2502 | 29,88 |
| 33 | 56,8155 | 0,2517 | 29,89 |
| 34 | 56,8285 | 0,2518 | 29,89 |
| 35 | 56,8268 | 0,253 | 29,89 |
| 36 | 56,8252 | 0,2467 | 29,89 |
| 37 | 56,8332 | 0,2522 | 29,88 |
| 38 | 56,8255 | 0,248 | 29,89 |
| 39 | 56,8303 | 0,2538 | 29,88 |
| 40 | 56,8268 | 0,2523 | 29,89 |
| 41 | 56,828 | 0,2465 | 29,89 |
| 42 | 56,8292 | 0,2508 | 29,88 |
| 43 | 56,8297 | 0,2507 | 29,88 |
| 44 | 56,8181 | 0,2501 | 29,89 |
| 45 | 56,8263 | 0,2547 | 29,89 |
| 46 | 56,8224 | 0,2472 | 29,89 |
| 47 | 56,8275 | 0,2534 | 29,89 |
| 48 | 56,8268 | 0,2496 | 29,89 |
| 49 | 56,8202 | 0,2484 | 29,89 |
| 50 | 56,8225 | 0,2475 | 29,89 |

| | | | |
|---|---|---|---|
| 51 | 56,8244 | 0,2484 | 29,89 |
| 52 | 56,829 | 0,2503 | 29,88 |
| 53 | 56,8267 | 0,2491 | 29,89 |
| 54 | 56,8334 | 0,2507 | 29,88 |
| 55 | 56,8232 | 0,2525 | 29,89 |
| 56 | 56,8262 | 0,2469 | 29,89 |
| 57 | 56,8275 | 0,246 | 29,89 |
| 58 | 56,8267 | 0,2455 | 29,89 |
| 59 | 56,8291 | 0,2459 | 29,88 |
| 60 | 56,8286 | 0,2474 | 29,89 |
| 61 | 56,8251 | 0,2461 | 29,89 |
| 62 | 56,8204 | 0,2475 | 29,89 |
| 63 | 56,819 | 0,2425 | 29,89 |
| 64 | 56,8209 | 0,2432 | 29,89 |
| 65 | 56,8236 | 0,2483 | 29,89 |
| 66 | 56,825 | 0,2522 | 29,89 |
| 67 | 56,823 | 0,2464 | 29,89 |
| 68 | 56,8231 | 0,2474 | 29,89 |
| 69 | 56,8187 | 0,247 | 29,89 |
| 70 | 56,8229 | 0,2488 | 29,89 |
| 71 | 56,8177 | 0,2463 | 29,89 |
| 72 | 56,826 | 0,2488 | 29,89 |
| 73 | 56,8216 | 0,2493 | 29,89 |
| 74 | 56,825 | 0,2509 | 29,89 |
| 75 | 56,8235 | 0,2466 | 29,89 |
| 76 | 56,825 | 0,2498 | 29,89 |
| 77 | 56,8251 | 0,254 | 29,89 |
| 78 | 56,829 | 0,2554 | 29,88 |
| 79 | 56,8281 | 0,2525 | 29,89 |
| 80 | 56,824 | 0,2459 | 29,89 |
| 81 | 56,8302 | 0,2556 | 29,88 |
| 82 | 56,8285 | 0,2504 | 29,89 |
| 83 | 56,8291 | 0,2535 | 29,88 |
| 84 | 56,8269 | 0,2468 | 29,89 |
| 85 | 56,8259 | 0,243 | 29,89 |
| 86 | 56,8241 | 0,2446 | 29,89 |
| 87 | 56,8287 | 0,2489 | 29,88 |
| 88 | 56,8226 | 0,2477 | 29,89 |
| 89 | 56,8238 | 0,2458 | 29,89 |
| 90 | 56,8289 | 0,2494 | 29,88 |
| 91 | 56,8281 | 0,252 | 29,89 |

| | | | |
|---|---|---|---|
| 92 | 56,8291 | 0,2472 | 29,88 |
| 93 | 56,8287 | 0,2452 | 29,88 |
| 94 | 56,8272 | 0,2442 | 29,89 |
| 95 | 56,825 | 0,244 | 29,89 |
| 96 | 56,8274 | 0,2479 | 29,89 |
| 97 | 56,8245 | 0,2457 | 29,89 |
| 98 | 56,8264 | 0,2495 | 29,89 |
| 99 | 56,8248 | 0,2491 | 29,89 |

Cuadro C.42: Clarke con tamaño de bloque 8

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 43,4435 | 0,1637 | 39,09 |
| 1 | 43,4756 | 0,1723 | 39,06 |
| 2 | 43,4789 | 0,1738 | 39,06 |
| 3 | 43,472 | 0,1741 | 39,07 |
| 4 | 43,4772 | 0,1725 | 39,06 |
| 5 | 43,4725 | 0,1676 | 39,07 |
| 6 | 43,4782 | 0,1719 | 39,06 |
| 7 | 43,4733 | 0,17 | 39,07 |
| 8 | 43,4775 | 0,1713 | 39,06 |
| 9 | 43,4742 | 0,173 | 39,07 |
| 10 | 43,4783 | 0,172 | 39,06 |
| 11 | 43,4538 | 0,1691 | 39,08 |
| 12 | 43,4748 | 0,1709 | 39,06 |
| 13 | 43,4809 | 0,1732 | 39,06 |
| 14 | 43,4759 | 0,1699 | 39,06 |
| 15 | 43,4779 | 0,1725 | 39,06 |
| 16 | 43,4758 | 0,1685 | 39,06 |
| 17 | 43,477 | 0,1707 | 39,06 |
| 18 | 43,4732 | 0,1679 | 39,07 |
| 19 | 43,4772 | 0,1713 | 39,06 |
| 20 | 43,4732 | 0,1689 | 39,07 |
| 21 | 43,4787 | 0,1719 | 39,06 |
| 22 | 43,4606 | 0,1719 | 39,08 |
| 23 | 43,477 | 0,1701 | 39,06 |
| 24 | 43,4782 | 0,1708 | 39,06 |
| 25 | 43,4749 | 0,1686 | 39,06 |
| 26 | 43,4768 | 0,1684 | 39,06 |
| 27 | 43,4759 | 0,1674 | 39,06 |
| 28 | 43,4769 | 0,1697 | 39,06 |

| | | | |
|---|---|---|---|
| 29 | 43,4737 | 0,1691 | 39,07 |
| 30 | 43,4756 | 0,1701 | 39,06 |
| 31 | 43,4715 | 0,1716 | 39,07 |
| 32 | 43,4771 | 0,1713 | 39,06 |
| 33 | 43,4662 | 0,17 | 39,07 |
| 34 | 43,4753 | 0,1739 | 39,06 |
| 35 | 43,4749 | 0,1701 | 39,06 |
| 36 | 43,4737 | 0,169 | 39,07 |
| 37 | 43,4765 | 0,171 | 39,06 |
| 38 | 43,4711 | 0,1669 | 39,07 |
| 39 | 43,4782 | 0,1719 | 39,06 |
| 40 | 43,4756 | 0,1739 | 39,06 |
| 41 | 43,4769 | 0,1706 | 39,06 |
| 42 | 43,4738 | 0,1693 | 39,07 |
| 43 | 43,481 | 0,1771 | 39,06 |
| 44 | 43,4695 | 0,1748 | 39,07 |
| 45 | 43,4735 | 0,1686 | 39,07 |
| 46 | 43,4773 | 0,1709 | 39,06 |
| 47 | 43,4739 | 0,1677 | 39,07 |
| 48 | 43,4781 | 0,1749 | 39,06 |
| 49 | 43,4738 | 0,1677 | 39,07 |
| 50 | 43,4794 | 0,1716 | 39,06 |
| 51 | 43,4762 | 0,1697 | 39,06 |
| 52 | 43,4784 | 0,1694 | 39,06 |
| 53 | 43,4732 | 0,1673 | 39,07 |
| 54 | 43,4773 | 0,1722 | 39,06 |
| 55 | 43,4701 | 0,1717 | 39,07 |
| 56 | 43,4752 | 0,1689 | 39,06 |
| 57 | 43,4807 | 0,1733 | 39,06 |
| 58 | 43,4798 | 0,172 | 39,06 |
| 59 | 43,4798 | 0,1735 | 39,06 |
| 60 | 43,4755 | 0,1696 | 39,06 |
| 61 | 43,4781 | 0,1705 | 39,06 |
| 62 | 43,4738 | 0,169 | 39,07 |
| 63 | 43,4768 | 0,1742 | 39,06 |
| 64 | 43,4737 | 0,1694 | 39,07 |
| 65 | 43,4778 | 0,1705 | 39,06 |
| 66 | 43,4669 | 0,1683 | 39,07 |
| 67 | 43,475 | 0,1692 | 39,06 |
| 68 | 43,4761 | 0,1694 | 39,06 |
| 69 | 43,4741 | 0,1683 | 39,07 |

| | | | |
|---|---|---|---|
| 70 | 43,4782 | 0,1701 | 39,06 |
| 71 | 43,4748 | 0,1684 | 39,06 |
| 72 | 43,4757 | 0,1681 | 39,06 |
| 73 | 43,4763 | 0,1699 | 39,06 |
| 74 | 43,4777 | 0,1699 | 39,06 |
| 75 | 43,4759 | 0,1677 | 39,06 |
| 76 | 43,4786 | 0,1745 | 39,06 |
| 77 | 43,4749 | 0,1717 | 39,06 |
| 78 | 43,4758 | 0,169 | 39,06 |
| 79 | 43,4746 | 0,1691 | 39,06 |
| 80 | 43,4738 | 0,1679 | 39,07 |
| 81 | 43,4749 | 0,173 | 39,06 |
| 82 | 43,4735 | 0,1679 | 39,07 |
| 83 | 43,4779 | 0,1701 | 39,06 |
| 84 | 43,4731 | 0,1653 | 39,07 |
| 85 | 43,4762 | 0,1686 | 39,06 |
| 86 | 43,4735 | 0,1671 | 39,07 |
| 87 | 43,4777 | 0,1703 | 39,06 |
| 88 | 43,4755 | 0,1727 | 39,06 |
| 89 | 43,4731 | 0,1697 | 39,07 |
| 90 | 43,4747 | 0,1688 | 39,06 |
| 91 | 43,4765 | 0,1692 | 39,06 |
| 92 | 43,4797 | 0,1702 | 39,06 |
| 93 | 43,4735 | 0,1669 | 39,07 |
| 94 | 43,4782 | 0,1723 | 39,06 |
| 95 | 43,4756 | 0,1706 | 39,06 |
| 96 | 43,4813 | 0,1734 | 39,06 |
| 97 | 43,4772 | 0,1744 | 39,06 |
| 98 | 43,4777 | 0,1737 | 39,06 |
| 99 | 43,4787 | 0,1743 | 39,06 |

Cuadro C.43: Clarke con tamaño de bloque 16

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 43,9842 | 0,2042 | 38,61 |
| 1 | 44,029 | 0,2155 | 38,57 |
| 2 | 44,0218 | 0,2103 | 38,58 |
| 3 | 44,0232 | 0,2132 | 38,58 |
| 4 | 44,025 | 0,2135 | 38,58 |
| 5 | 44,0194 | 0,2095 | 38,58 |
| 6 | 44,0214 | 0,2113 | 38,58 |

| | | | |
|---|---|---|---|
| 7 | 44,0196 | 0,2132 | 38,58 |
| 8 | 44,0177 | 0,2092 | 38,58 |
| 9 | 44,0176 | 0,2092 | 38,58 |
| 10 | 44,0152 | 0,2131 | 38,58 |
| 11 | 43,9926 | 0,2096 | 38,6 |
| 12 | 44,0175 | 0,2092 | 38,58 |
| 13 | 44,0181 | 0,209 | 38,58 |
| 14 | 44,0235 | 0,2129 | 38,58 |
| 15 | 44,0237 | 0,2137 | 38,58 |
| 16 | 44,0246 | 0,2133 | 38,58 |
| 17 | 44,0259 | 0,2175 | 38,58 |
| 18 | 44,025 | 0,2146 | 38,58 |
| 19 | 44,0178 | 0,2103 | 38,58 |
| 20 | 44,0198 | 0,2146 | 38,58 |
| 21 | 44,0197 | 0,2134 | 38,58 |
| 22 | 44,0029 | 0,2105 | 38,6 |
| 23 | 44,0248 | 0,2174 | 38,58 |
| 24 | 44,0215 | 0,2182 | 38,58 |
| 25 | 44,022 | 0,216 | 38,58 |
| 26 | 44,0244 | 0,2178 | 38,58 |
| 27 | 44,0218 | 0,2144 | 38,58 |
| 28 | 44,0236 | 0,2145 | 38,58 |
| 29 | 44,0231 | 0,2133 | 38,58 |
| 30 | 44,0198 | 0,2122 | 38,58 |
| 31 | 44,0242 | 0,2147 | 38,58 |
| 32 | 44,0216 | 0,2128 | 38,58 |
| 33 | 44,0087 | 0,2103 | 38,59 |
| 34 | 44,0245 | 0,2148 | 38,58 |
| 35 | 44,0201 | 0,2146 | 38,58 |
| 36 | 44,0229 | 0,21 | 38,58 |
| 37 | 44,0195 | 0,2093 | 38,58 |
| 38 | 44,0232 | 0,2132 | 38,58 |
| 39 | 44,0243 | 0,2149 | 38,58 |
| 40 | 44,0246 | 0,2147 | 38,58 |
| 41 | 44,0239 | 0,2124 | 38,58 |
| 42 | 44,025 | 0,2131 | 38,58 |
| 43 | 44,0213 | 0,2115 | 38,58 |
| 44 | 44,0146 | 0,2131 | 38,59 |
| 45 | 44,022 | 0,213 | 38,58 |
| 46 | 44,0188 | 0,2107 | 38,58 |
| 47 | 44,0182 | 0,2096 | 38,58 |

| | | | |
|---|---|---|---|
| 48 | 44,0208 | 0,2137 | 38,58 |
| 49 | 44,028 | 0,2201 | 38,57 |
| 50 | 44,0208 | 0,2119 | 38,58 |
| 51 | 44,0224 | 0,2128 | 38,58 |
| 52 | 44,0184 | 0,21 | 38,58 |
| 53 | 44,0264 | 0,2154 | 38,58 |
| 54 | 44,0206 | 0,2125 | 38,58 |
| 55 | 44,016 | 0,2123 | 38,58 |
| 56 | 44,021 | 0,2118 | 38,58 |
| 57 | 44,0187 | 0,2107 | 38,58 |
| 58 | 44,0199 | 0,2155 | 38,58 |
| 59 | 44,023 | 0,2158 | 38,58 |
| 60 | 44,028 | 0,2209 | 38,57 |
| 61 | 44,0239 | 0,2133 | 38,58 |
| 62 | 44,0186 | 0,2096 | 38,58 |
| 63 | 44,02 | 0,2118 | 38,58 |
| 64 | 44,0239 | 0,2113 | 38,58 |
| 65 | 44,0193 | 0,2086 | 38,58 |
| 66 | 44,0226 | 0,2202 | 38,58 |
| 67 | 44,0262 | 0,2146 | 38,58 |
| 68 | 44,0246 | 0,2193 | 38,58 |
| 69 | 44,0226 | 0,2142 | 38,58 |
| 70 | 44,0207 | 0,2126 | 38,58 |
| 71 | 44,0236 | 0,2153 | 38,58 |
| 72 | 44,0249 | 0,2162 | 38,58 |
| 73 | 44,0293 | 0,2163 | 38,57 |
| 74 | 44,0201 | 0,2106 | 38,58 |
| 75 | 44,0259 | 0,2136 | 38,58 |
| 76 | 44,0275 | 0,2151 | 38,57 |
| 77 | 44,0185 | 0,2124 | 38,58 |
| 78 | 44,027 | 0,2133 | 38,57 |
| 79 | 44,021 | 0,2083 | 38,58 |
| 80 | 44,0227 | 0,2124 | 38,58 |
| 81 | 44,0198 | 0,2079 | 38,58 |
| 82 | 44,0216 | 0,2125 | 38,58 |
| 83 | 44,0164 | 0,2097 | 38,58 |
| 84 | 44,0222 | 0,2125 | 38,58 |
| 85 | 44,0247 | 0,2144 | 38,58 |
| 86 | 44,0211 | 0,2099 | 38,58 |
| 87 | 44,0183 | 0,2068 | 38,58 |
| 88 | 44,0251 | 0,2161 | 38,58 |

| | | | |
|---|---|---|---|
| 89 | 44,0202 | 0,211 | 38,58 |
| 90 | 44,021 | 0,2106 | 38,58 |
| 91 | 44,0217 | 0,2123 | 38,58 |
| 92 | 44,0207 | 0,2113 | 38,58 |
| 93 | 44,0239 | 0,212 | 38,58 |
| 94 | 44,022 | 0,2114 | 38,58 |
| 95 | 44,0202 | 0,2096 | 38,58 |
| 96 | 44,0172 | 0,2086 | 38,58 |
| 97 | 44,024 | 0,2126 | 38,58 |
| 98 | 44,0178 | 0,2108 | 38,58 |
| 99 | 44,0253 | 0,2167 | 38,58 |

Cuadro C.44: Clarke con tamaño de bloque 32

### C.2.2. Ordis

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 163,089 | 2,4066 | 10,41 |
| 1 | 163,6881 | 2,7293 | 10,38 |
| 2 | 163,0566 | 2,3435 | 10,42 |
| 3 | ERROR | ERROR | ERROR |
| 4 | 163,0916 | 2,3945 | 10,41 |
| 5 | 163,0627 | 2,3504 | 10,42 |
| 6 | 163,0216 | 2,3658 | 10,42 |
| 7 | 163,4123 | 2,6142 | 10,39 |
| 8 | 163,1623 | 2,4448 | 10,41 |
| 9 | ERROR | ERROR | ERROR |
| 10 | 163,1603 | 2,4437 | 10,41 |
| 11 | 162,9682 | 2,293 | 10,42 |
| 12 | 163,075 | 2,4094 | 10,41 |
| 13 | 163,1395 | 2,4087 | 10,41 |
| 14 | 163,589 | 2,6704 | 10,38 |
| 15 | ERROR | ERROR | ERROR |
| 16 | 163,0527 | 2,3835 | 10,42 |
| 17 | 162,9884 | 2,3305 | 10,42 |
| 18 | 163,0662 | 2,4093 | 10,41 |
| 19 | 163,2747 | 2,4665 | 10,4 |
| 20 | 163,1687 | 2,4293 | 10,41 |
| 21 | ERROR | ERROR | ERROR |
| 22 | 163,1499 | 2,4203 | 10,41 |
| 23 | 163,1167 | 2,3753 | 10,41 |

| | | | |
|---|---|---|---|
| 24 | 163,1502 | 2,4577 | 10,41 |
| 25 | 163,3278 | 2,6005 | 10,4 |
| 26 | 163,5101 | 2,6382 | 10,39 |
| 27 | ERROR | ERROR | ERROR |
| 28 | 163,1852 | 2,4235 | 10,41 |
| 29 | 163,0802 | 2,3307 | 10,41 |
| 30 | 163,6154 | 2,8866 | 10,38 |
| 31 | 163,159 | 2,4654 | 10,41 |
| 32 | 163,0513 | 2,3494 | 10,42 |
| 33 | ERROR | ERROR | ERROR |
| 34 | 163,0585 | 2,3996 | 10,42 |
| 35 | 163,5596 | 2,6196 | 10,38 |
| 36 | 163,1482 | 2,4119 | 10,41 |
| 37 | 163,3187 | 2,4178 | 10,4 |
| 38 | 163,1091 | 2,4066 | 10,41 |
| 39 | ERROR | ERROR | ERROR |
| 40 | 163,2197 | 2,4885 | 10,41 |
| 41 | 163,4684 | 2,5787 | 10,39 |
| 42 | 163,2909 | 2,5612 | 10,4 |
| 43 | 163,19 | 2,4379 | 10,41 |
| 44 | 163,1932 | 2,4464 | 10,41 |
| 45 | ERROR | ERROR | ERROR |
| 46 | 163,0691 | 2,4032 | 10,41 |
| 47 | 163,131 | 2,4072 | 10,41 |
| 48 | 163,2407 | 2,5319 | 10,4 |
| 49 | 163,1557 | 2,4672 | 10,41 |
| 50 | 163,1737 | 2,3922 | 10,41 |
| 51 | ERROR | ERROR | ERROR |
| 52 | 163,2032 | 2,4804 | 10,41 |
| 53 | 163,0651 | 2,3722 | 10,42 |
| 54 | 163,5151 | 2,6658 | 10,39 |
| 55 | 163,1968 | 2,4443 | 10,41 |
| 56 | 163,1824 | 2,4326 | 10,41 |
| 57 | ERROR | ERROR | ERROR |
| 58 | 163,3224 | 2,5595 | 10,4 |
| 59 | 163,0783 | 2,422 | 10,41 |
| 60 | 163,0332 | 2,408 | 10,42 |
| 61 | 163,1983 | 2,4682 | 10,41 |
| 62 | 163,0615 | 2,3403 | 10,42 |
| 63 | ERROR | ERROR | ERROR |
| 64 | 163,0865 | 2,3932 | 10,41 |

| | | | |
|---|---|---|---|
| 65 | 162,9996 | 2,3212 | 10,42 |
| 66 | 163,2994 | 2,5773 | 10,4 |
| 67 | 163,3061 | 2,4515 | 10,4 |
| 68 | 163,0184 | 2,3352 | 10,42 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 163,2144 | 2,4601 | 10,41 |
| 71 | 163,0647 | 2,3822 | 10,42 |
| 72 | 163,4565 | 2,6309 | 10,39 |
| 73 | 163,3472 | 2,4444 | 10,4 |
| 74 | 163,0424 | 2,2787 | 10,42 |
| 75 | ERROR | ERROR | ERROR |
| 76 | 163,4209 | 2,5845 | 10,39 |
| 77 | 163,0682 | 2,3791 | 10,41 |
| 78 | 163,08 | 2,4125 | 10,41 |
| 79 | 163,236 | 2,4377 | 10,4 |
| 80 | ERROR | ERROR | ERROR |
| 81 | ERROR | ERROR | ERROR |
| 82 | 163,0159 | 2,2901 | 10,42 |
| 83 | 163,193 | 2,458 | 10,41 |
| 84 | 163,1107 | 2,3736 | 10,41 |
| 85 | 163,2446 | 2,4707 | 10,4 |
| 86 | ERROR | ERROR | ERROR |
| 87 | ERROR | ERROR | ERROR |
| 88 | 163,1503 | 2,4226 | 10,41 |
| 89 | 163,2151 | 2,5022 | 10,41 |
| 90 | 163,0988 | 2,4238 | 10,41 |
| 91 | 163,4535 | 2,6701 | 10,39 |
| 92 | ERROR | ERROR | ERROR |
| 93 | 163,196 | 2,4377 | 10,41 |
| 94 | 163,5884 | 2,5934 | 10,38 |
| 95 | 163,0279 | 2,3378 | 10,42 |
| 96 | 163,1548 | 2,4464 | 10,41 |
| 97 | 163,1551 | 2,412 | 10,41 |
| 98 | ERROR | ERROR | ERROR |

Cuadro C.45: Ordis usando 4 nodos con tamaño de bloque 4

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 87,7247 | 2,462 | 19,36 |
| 1 | 87,6864 | 2,474 | 19,37 |
| 2 | 87,6371 | 2,426 | 19,38 |

| | | | |
|---|---|---|---|
| 3 | 87,6809 | 2,4511 | 19,37 |
| 4 | ERROR | ERROR | ERROR |
| 5 | ERROR | ERROR | ERROR |
| 6 | 87,6757 | 2,4435 | 19,37 |
| 7 | 87,9407 | 2,5102 | 19,31 |
| 8 | 87,7857 | 2,5137 | 19,35 |
| 9 | 87,6852 | 2,4565 | 19,37 |
| 10 | ERROR | ERROR | ERROR |
| 11 | 87,6576 | 2,4178 | 19,37 |
| 12 | 87,6552 | 2,4473 | 19,38 |
| 13 | 87,6377 | 2,4243 | 19,38 |
| 14 | 87,7726 | 2,4993 | 19,35 |
| 15 | 87,8151 | 2,5471 | 19,34 |
| 16 | 87,7068 | 2,4562 | 19,36 |
| 17 | ERROR | ERROR | ERROR |
| 18 | 87,6589 | 2,4128 | 19,37 |
| 19 | 87,7387 | 2,4952 | 19,36 |
| 20 | 87,7967 | 2,5332 | 19,34 |
| 21 | 87,6796 | 2,439 | 19,37 |
| 22 | ERROR | ERROR | ERROR |
| 23 | 87,7271 | 2,4526 | 19,36 |
| 24 | 88,2044 | 2,6143 | 19,25 |
| 25 | 87,7348 | 2,4657 | 19,36 |
| 26 | 87,6614 | 2,4329 | 19,37 |
| 27 | ERROR | ERROR | ERROR |
| 28 | 87,7121 | 2,4285 | 19,36 |
| 29 | 87,7423 | 2,4933 | 19,36 |
| 30 | 87,8689 | 2,4712 | 19,33 |
| 31 | 87,7576 | 2,4807 | 19,35 |
| 32 | ERROR | ERROR | ERROR |
| 33 | 87,6178 | 2,403 | 19,38 |
| 34 | 87,8494 | 2,5267 | 19,33 |
| 35 | 87,7834 | 2,4831 | 19,35 |
| 36 | 87,8082 | 2,5279 | 19,34 |
| 37 | 87,6728 | 2,4352 | 19,37 |
| 38 | ERROR | ERROR | ERROR |
| 39 | 87,7738 | 2,4978 | 19,35 |
| 40 | 87,992 | 2,6549 | 19,3 |
| 41 | 87,7816 | 2,5341 | 19,35 |
| 42 | 87,7001 | 2,4414 | 19,37 |
| 43 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 44 | 87,6683 | 2,4251 | 19,37 |
| 45 | 87,7705 | 2,5079 | 19,35 |
| 46 | 87,7448 | 2,484 | 19,36 |
| 47 | 87,7019 | 2,4526 | 19,36 |
| 48 | ERROR | ERROR | ERROR |
| 49 | 87,9249 | 2,5114 | 19,32 |
| 50 | 87,739 | 2,4891 | 19,36 |
| 51 | 87,6491 | 2,4238 | 19,38 |
| 52 | 87,7248 | 2,468 | 19,36 |
| 53 | ERROR | ERROR | ERROR |
| 54 | 87,613 | 2,4089 | 19,38 |
| 55 | 87,6312 | 2,4281 | 19,38 |
| 56 | 87,7255 | 2,4512 | 19,36 |
| 57 | 87,6734 | 2,439 | 19,37 |
| 58 | 88,0385 | 2,6086 | 19,29 |
| 59 | ERROR | ERROR | ERROR |
| 60 | 87,5847 | 2,3812 | 19,39 |
| 61 | 88,0009 | 2,6004 | 19,3 |
| 62 | 87,8042 | 2,5498 | 19,34 |
| 63 | 87,7295 | 2,4744 | 19,36 |
| 64 | ERROR | ERROR | ERROR |
| 65 | 87,6534 | 2,4209 | 19,38 |
| 66 | 87,9535 | 2,5523 | 19,31 |
| 67 | 87,6552 | 2,4336 | 19,38 |
| 68 | 87,7715 | 2,4998 | 19,35 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 87,6703 | 2,4141 | 19,37 |
| 71 | 87,7689 | 2,5315 | 19,35 |
| 72 | 87,7697 | 2,4956 | 19,35 |
| 73 | 87,748 | 2,5083 | 19,35 |
| 74 | ERROR | ERROR | ERROR |
| 75 | 87,6485 | 2,4473 | 19,38 |
| 76 | 87,9006 | 2,511 | 19,32 |
| 77 | 87,7164 | 2,4519 | 19,36 |
| 78 | 87,7201 | 2,4395 | 19,36 |
| 79 | 87,6754 | 2,4124 | 19,37 |
| 80 | ERROR | ERROR | ERROR |
| 81 | 87,5681 | 2,3685 | 19,39 |
| 82 | 87,7807 | 2,5129 | 19,35 |
| 83 | 87,7366 | 2,5009 | 19,36 |
| 84 | 87,7183 | 2,4552 | 19,36 |

| | | | |
|---|---|---|---|
| 85 | ERROR | ERROR | ERROR |
| 86 | 87,8025 | 2,4918 | 19,34 |
| 87 | 87,7082 | 2,4676 | 19,36 |
| 88 | 87,731 | 2,4713 | 19,36 |
| 89 | 87,6858 | 2,4316 | 19,37 |
| 90 | ERROR | ERROR | ERROR |
| 91 | 87,7338 | 2,4779 | 19,36 |
| 92 | 87,8432 | 2,5478 | 19,33 |
| 93 | 87,6676 | 2,4256 | 19,37 |
| 94 | 87,7907 | 2,5364 | 19,35 |
| 95 | ERROR | ERROR | ERROR |
| 96 | 87,6614 | 2,4574 | 19,37 |
| 97 | 87,7458 | 2,5232 | 19,36 |
| 98 | 87,7389 | 2,4724 | 19,36 |

Cuadro C.46: Ordis usando 4 nodos con tamaño de bloque 8

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 78,7584 | 10,5524 | 21,56 |
| 1 | 77,8154 | 10,3318 | 21,83 |
| 2 | 77,8325 | 9,9589 | 21,82 |
| 3 | 77,8371 | 9,9333 | 21,82 |
| 4 | 77,8652 | 10,0572 | 21,81 |
| 5 | 77,8097 | 10,3017 | 21,83 |
| 6 | ERROR | ERROR | ERROR |
| 7 | 77,7618 | 10,0941 | 21,84 |
| 8 | 78,1008 | 10,2965 | 21,75 |
| 9 | 78,0591 | 10,2282 | 21,76 |
| 10 | 77,8429 | 9,9005 | 21,82 |
| 11 | 77,8255 | 9,8562 | 21,82 |
| 12 | 77,7867 | 10,3468 | 21,83 |
| 13 | 77,8312 | 9,8955 | 21,82 |
| 14 | 77,7134 | 10,1403 | 21,85 |
| 15 | 78,3909 | 10,5492 | 21,66 |
| 16 | 77,7321 | 9,9934 | 21,85 |
| 17 | 77,7779 | 10,0977 | 21,84 |
| 18 | 77,8987 | 10,2108 | 21,8 |
| 19 | 77,7828 | 9,6847 | 21,83 |
| 20 | 77,7416 | 10,0501 | 21,85 |
| 21 | 77,7852 | 9,8972 | 21,83 |
| 22 | 78,6841 | 10,5093 | 21,58 |

| | | | |
|---|---|---|---|
| 23 | ERROR | ERROR | ERROR |
| 24 | 77,8718 | 9,8051 | 21,81 |
| 25 | 77,8683 | 10,1634 | 21,81 |
| 26 | 78,056 | 10,3816 | 21,76 |
| 27 | 77,7164 | 9,7668 | 21,85 |
| 28 | 77,7518 | 10,1761 | 21,84 |
| 29 | 77,8196 | 10,1959 | 21,82 |
| 30 | ERROR | ERROR | ERROR |
| 31 | 77,7874 | 9,6047 | 21,83 |
| 32 | 77,7907 | 10,1999 | 21,83 |
| 33 | 77,8056 | 9,6488 | 21,83 |
| 34 | 77,8118 | 10,1359 | 21,83 |
| 35 | 77,8277 | 10,5193 | 21,82 |
| 36 | 77,8616 | 9,6682 | 21,81 |
| 37 | ERROR | ERROR | ERROR |
| 38 | 77,8071 | 10,7446 | 21,83 |
| 39 | 77,7789 | 10,2642 | 21,84 |
| 40 | 77,8261 | 9,9511 | 21,82 |
| 41 | 77,83 | 10,4377 | 21,82 |
| 42 | 77,7352 | 10,1419 | 21,85 |
| 43 | ERROR | ERROR | ERROR |
| 44 | ERROR | ERROR | ERROR |
| 45 | 77,9032 | 9,9285 | 21,8 |
| 46 | 77,8172 | 9,9629 | 21,82 |
| 47 | 77,85 | 10,2939 | 21,82 |
| 48 | 77,7792 | 10,2977 | 21,84 |
| 49 | 77,7256 | 9,7992 | 21,85 |
| 50 | ERROR | ERROR | ERROR |
| 51 | 77,9177 | 10,0537 | 21,8 |
| 52 | 77,8339 | 9,8932 | 21,82 |
| 53 | 77,7887 | 9,9825 | 21,83 |
| 54 | 77,7778 | 9,7473 | 21,84 |
| 55 | 78,1377 | 10,6207 | 21,74 |
| 56 | 77,8376 | 9,9119 | 21,82 |
| 57 | ERROR | ERROR | ERROR |
| 58 | 77,8329 | 10,0352 | 21,82 |
| 59 | 77,7681 | 10,7405 | 21,84 |
| 60 | 77,7648 | 10,1926 | 21,84 |
| 61 | 77,8384 | 9,9522 | 21,82 |
| 62 | 78,0439 | 9,5921 | 21,76 |
| 63 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 64 | 77,7835 | 10,4694 | 21,83 |
| 65 | 77,8191 | 10,4475 | 21,82 |
| 66 | 77,7612 | 10,2504 | 21,84 |
| 67 | 77,8145 | 10,2477 | 21,83 |
| 68 | 77,7083 | 9,9067 | 21,86 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 78,3228 | 10,2031 | 21,68 |
| 71 | 77,833 | 10,4274 | 21,82 |
| 72 | 77,7274 | 10,321 | 21,85 |
| 73 | 280,4668 | 132,1872 | 6,06 |
| 74 | ERROR | ERROR | ERROR |
| 75 | 77,7726 | 10,0961 | 21,84 |
| 76 | 78,2388 | 10,2555 | 21,71 |
| 77 | 78,107 | 10,2242 | 21,74 |
| 78 | 77,7224 | 10,3618 | 21,85 |
| 79 | ERROR | ERROR | ERROR |
| 80 | ERROR | ERROR | ERROR |
| 81 | 77,7225 | 9,8934 | 21,85 |
| 82 | 78,5605 | 10,5028 | 21,62 |
| 83 | 77,8748 | 10,363 | 21,81 |
| 84 | ERROR | ERROR | ERROR |
| 85 | ERROR | ERROR | ERROR |
| 86 | 77,7443 | 10,1522 | 21,84 |
| 87 | 77,8136 | 9,8015 | 21,83 |
| 88 | 78,4146 | 10,4521 | 21,66 |
| 89 | ERROR | ERROR | ERROR |
| 90 | ERROR | ERROR | ERROR |
| 91 | ERROR | ERROR | ERROR |
| 92 | ERROR | ERROR | ERROR |
| 93 | 77,8881 | 10,0889 | 21,8 |
| 94 | 77,8663 | 10,6083 | 21,81 |
| 95 | ERROR | ERROR | ERROR |
| 96 | 77,77 | 10,1535 | 21,84 |
| 97 | ERROR | ERROR | ERROR |
| 98 | 77,7277 | 10,0277 | 21,85 |
| 99 | ERROR | ERROR | ERROR |

Cuadro C.47: Ordis usando 4 nodos con tamaño de bloque 16

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 88,012 | 3,2315 | 19,3 |
| 1 | 87,8153 | 3,0728 | 19,34 |
| 2 | ERROR | ERROR | ERROR |
| 3 | 87,8214 | 3,0825 | 19,34 |
| 4 | 87,963 | 3,1962 | 19,31 |
| 5 | 87,9889 | 3,2266 | 19,3 |
| 6 | 87,8757 | 3,115 | 19,33 |
| 7 | ERROR | ERROR | ERROR |
| 8 | 87,8954 | 3,13 | 19,32 |
| 9 | 87,9041 | 3,1675 | 19,32 |
| 10 | 87,9211 | 3,1887 | 19,32 |
| 11 | 87,9379 | 3,1731 | 19,31 |
| 12 | ERROR | ERROR | ERROR |
| 13 | 87,8358 | 3,1176 | 19,34 |
| 14 | 87,8937 | 3,1867 | 19,32 |
| 15 | 87,9253 | 3,1604 | 19,32 |
| 16 | 88,0249 | 3,2408 | 19,29 |
| 17 | ERROR | ERROR | ERROR |
| 18 | 87,8237 | 3,0909 | 19,34 |
| 19 | 88,0054 | 3,2181 | 19,3 |
| 20 | 87,8508 | 3,1188 | 19,33 |
| 21 | 87,907 | 3,1637 | 19,32 |
| 22 | 87,7946 | 3,0703 | 19,34 |
| 23 | ERROR | ERROR | ERROR |
| 24 | 87,8733 | 3,1117 | 19,33 |
| 25 | 87,9739 | 3,209 | 19,3 |
| 26 | 87,903 | 3,1577 | 19,32 |
| 27 | 87,8816 | 3,1348 | 19,33 |
| 28 | ERROR | ERROR | ERROR |
| 29 | 88,1311 | 3,2056 | 19,27 |
| 30 | 87,9305 | 3,1801 | 19,31 |
| 31 | 87,903 | 3,1543 | 19,32 |
| 32 | 87,95 | 3,1738 | 19,31 |
| 33 | ERROR | ERROR | ERROR |
| 34 | 87,9415 | 3,1899 | 19,31 |
| 35 | 87,9249 | 3,186 | 19,32 |
| 36 | 87,9181 | 3,1454 | 19,32 |
| 37 | 87,9539 | 3,1977 | 19,31 |
| 38 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 39 | 87,8419 | 3,1333 | 19,33 |
| 40 | 87,9319 | 3,1664 | 19,31 |
| 41 | 88,0357 | 3,22 | 19,29 |
| 42 | 87,815 | 3,1025 | 19,34 |
| 43 | 87,8526 | 3,0889 | 19,33 |
| 44 | ERROR | ERROR | ERROR |
| 45 | 87,9515 | 3,1662 | 19,31 |
| 46 | 88,0245 | 3,2467 | 19,29 |
| 47 | 87,9807 | 3,2369 | 19,3 |
| 48 | 87,8468 | 3,0922 | 19,33 |
| 49 | ERROR | ERROR | ERROR |
| 50 | 87,8843 | 3,1132 | 19,32 |
| 51 | 87,8944 | 3,1386 | 19,32 |
| 52 | 87,8628 | 3,1426 | 19,33 |
| 53 | 87,906 | 3,1413 | 19,32 |
| 54 | ERROR | ERROR | ERROR |
| 55 | 87,8925 | 3,1514 | 19,32 |
| 56 | 87,867 | 3,1635 | 19,33 |
| 57 | 87,8949 | 3,1411 | 19,32 |
| 58 | 87,9288 | 3,1723 | 19,31 |
| 59 | ERROR | ERROR | ERROR |
| 60 | 87,7946 | 3,0685 | 19,34 |
| 61 | 87,9606 | 3,1853 | 19,31 |
| 62 | 87,9008 | 3,1458 | 19,32 |
| 63 | 87,9315 | 3,1872 | 19,31 |
| 64 | 87,8751 | 3,1288 | 19,33 |
| 65 | ERROR | ERROR | ERROR |
| 66 | 87,8078 | 3,0615 | 19,34 |
| 67 | 87,9665 | 3,1998 | 19,31 |
| 68 | 87,8889 | 3,1688 | 19,32 |
| 69 | 87,8231 | 3,0903 | 19,34 |
| 70 | ERROR | ERROR | ERROR |
| 71 | 87,8013 | 3,0806 | 19,34 |
| 72 | 87,9335 | 3,1711 | 19,31 |
| 73 | 87,9349 | 3,1915 | 19,31 |
| 74 | 87,9614 | 3,1912 | 19,31 |
| 75 | ERROR | ERROR | ERROR |
| 76 | 87,8497 | 3,1113 | 19,33 |
| 77 | 87,9089 | 3,1667 | 19,32 |
| 78 | 87,8766 | 3,1341 | 19,33 |
| 79 | 88,0009 | 3,2148 | 19,3 |

| | | | |
|---|---|---|---|
| 80 | ERROR | ERROR | ERROR |
| 81 | 87,8299 | 3,1106 | 19,34 |
| 82 | 87,953 | 3,1895 | 19,31 |
| 83 | 87,9491 | 3,1815 | 19,31 |
| 84 | 87,8439 | 3,1488 | 19,33 |
| 85 | 87,8069 | 3,0709 | 19,34 |
| 86 | ERROR | ERROR | ERROR |
| 87 | 87,7782 | 3,0629 | 19,35 |
| 88 | 88,0042 | 3,2264 | 19,3 |
| 89 | 87,9889 | 3,2642 | 19,3 |
| 90 | 87,9134 | 3,1338 | 19,32 |
| 91 | ERROR | ERROR | ERROR |
| 92 | 87,9157 | 3,1216 | 19,32 |
| 93 | 87,9453 | 3,2004 | 19,31 |
| 94 | 87,8075 | 3,0864 | 19,34 |
| 95 | 87,933 | 3,1834 | 19,31 |
| 96 | ERROR | ERROR | ERROR |
| 97 | 87,8393 | 3,0919 | 19,33 |
| 98 | 87,8541 | 3,1414 | 19,33 |

Cuadro C.48: Ordis usando 4 nodos con tamaño de bloque 32

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 90,8525 | 14,4765 | 18,69 |
| 1 | 92,0267 | 15,3386 | 18,45 |
| 2 | 92,6202 | 14,7069 | 18,34 |
| 3 | 92,5485 | 15,257 | 18,35 |
| 4 | 92,7977 | 15,4004 | 18,3 |
| 5 | 92,6905 | 14,8605 | 18,32 |
| 6 | 92,792 | 15,1791 | 18,3 |
| 7 | 92,8747 | 14,791 | 18,29 |
| 8 | 90,6505 | 14,0967 | 18,73 |
| 9 | 91,7907 | 15,6987 | 18,5 |
| 10 | 91,1242 | 14,6289 | 18,64 |
| 11 | 91,2007 | 15,0914 | 18,62 |
| 12 | 92,8177 | 15,0395 | 18,3 |
| 13 | 93,0841 | 15,6125 | 18,25 |
| 14 | 92,8926 | 14,9313 | 18,28 |
| 15 | 93,2634 | 14,9466 | 18,21 |
| 16 | 92,0454 | 14,7407 | 18,45 |
| 17 | 92,3189 | 14,6363 | 18,4 |

| | | | |
|---|---|---|---|
| 18 | 93,1225 | 14,6309 | 18,24 |
| 19 | 93,3207 | 15,0212 | 18,2 |
| 20 | 92,6482 | 15,7908 | 18,33 |
| 21 | 90,6314 | 14,6444 | 18,74 |
| 22 | 92,5059 | 14,4855 | 18,36 |
| 23 | 92,5152 | 15,1471 | 18,36 |
| 24 | 93,1578 | 15,2683 | 18,23 |
| 25 | 92,9535 | 15,4715 | 18,27 |
| 26 | 93,1425 | 14,6722 | 18,23 |
| 27 | 92,6965 | 15,5793 | 18,32 |
| 28 | 93,7133 | 14,865 | 18,12 |
| 29 | 90,7822 | 14,1239 | 18,71 |
| 30 | 91,1084 | 15,7731 | 18,64 |
| 31 | 92,268 | 15,8555 | 18,41 |
| 32 | 93,6727 | 15,5305 | 18,13 |
| 33 | 93,536 | 15,1604 | 18,16 |
| 34 | 92,8726 | 15,456 | 18,29 |
| 35 | 93,9996 | 15,2683 | 18,07 |
| 36 | 92,9961 | 14,8416 | 18,26 |
| 37 | 93,0479 | 15,2186 | 18,25 |
| 38 | 92,6 | 14,3988 | 18,34 |
| 39 | 92,8097 | 14,8231 | 18,3 |
| 40 | 92,1257 | 16,0959 | 18,43 |
| 41 | 92,0399 | 15,3178 | 18,45 |
| 42 | 92,8665 | 15,0375 | 18,29 |
| 43 | 94,1015 | 15,4589 | 18,05 |
| 44 | 92,8461 | 15,2661 | 18,29 |
| 45 | 92,7994 | 14,786 | 18,3 |
| 46 | 91,1381 | 14,1695 | 18,63 |
| 47 | 92,9732 | 15,373 | 18,27 |
| 48 | 92,1812 | 14,4547 | 18,42 |
| 49 | 92,6017 | 14,908 | 18,34 |
| 50 | 90,4071 | 14,5584 | 18,79 |
| 51 | 91,7839 | 15,2014 | 18,5 |
| 52 | 93,6465 | 15,0323 | 18,14 |
| 53 | 93,2535 | 15,0354 | 18,21 |
| 54 | 92,6805 | 14,7765 | 18,32 |
| 55 | 93,1976 | 15,1257 | 18,22 |
| 56 | 91,6482 | 14,355 | 18,53 |
| 57 | 93,3127 | 15,1828 | 18,2 |
| 58 | 93,3118 | 14,9507 | 18,2 |

| | | | |
|---|---|---|---|
| 59 | 93,1207 | 15,494 | 18,24 |
| 60 | 92,8597 | 15,9492 | 18,29 |
| 61 | 92,4241 | 14,8964 | 18,38 |
| 62 | 93,3987 | 15,2882 | 18,18 |
| 63 | 92,7784 | 14,8042 | 18,31 |
| 64 | 93,0746 | 14,5872 | 18,25 |
| 65 | 93,1331 | 14,8292 | 18,24 |
| 66 | 93,8681 | 14,9775 | 18,09 |
| 67 | 93,1371 | 14,9721 | 18,23 |
| 68 | 92,5625 | 14,5856 | 18,35 |
| 69 | 93,6754 | 15,6294 | 18,13 |
| 70 | 92,1179 | 15,3138 | 18,44 |
| 71 | 92,1404 | 14,8011 | 18,43 |
| 72 | 93,3897 | 14,7238 | 18,19 |
| 73 | 93,7464 | 15,3456 | 18,12 |
| 74 | 93,5614 | 15,635 | 18,15 |
| 75 | 93,0546 | 14,7997 | 18,25 |
| 76 | 92,4949 | 14,3871 | 18,36 |
| 77 | 93,6872 | 14,9774 | 18,13 |
| 78 | 91,6362 | 14,6247 | 18,53 |
| 79 | 91,719 | 15,102 | 18,52 |
| 80 | 93,6011 | 15,2239 | 18,14 |
| 81 | 92,4847 | 15,1186 | 18,36 |
| 82 | 93,0655 | 14,6993 | 18,25 |
| 83 | 94,0144 | 15,7403 | 18,06 |
| 84 | 92,9318 | 15,1795 | 18,27 |
| 85 | 93,2295 | 15,4759 | 18,22 |
| 86 | 92,8893 | 14,9166 | 18,28 |
| 87 | 93,1954 | 14,6304 | 18,22 |
| 88 | 92,8023 | 14,5334 | 18,3 |
| 89 | 94,0216 | 15,3359 | 18,06 |
| 90 | 92,7545 | 14,8578 | 18,31 |
| 91 | 92,8962 | 14,8935 | 18,28 |
| 92 | 92,9583 | 14,621 | 18,27 |
| 93 | 93,0586 | 14,936 | 18,25 |
| 94 | 94,0512 | 15,1546 | 18,06 |
| 95 | 93,3764 | 14,8689 | 18,19 |
| 96 | 90,9581 | 13,8491 | 18,67 |
| 97 | 92,4354 | 15,2675 | 18,37 |
| 98 | 92,7857 | 14,5925 | 18,3 |
| 99 | 93,1398 | 14,8533 | 18,23 |

Cuadro C.49: Ordis usando 9 nodos con tamaño de bloque 4

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 58,199 | 15,8737 | 29,18 |
| 1 | ERROR | ERROR | ERROR |
| 2 | 60,4609 | 15,54 | 28,09 |
| 3 | 56,1644 | 14,7153 | 30,24 |
| 4 | ERROR | ERROR | ERROR |
| 5 | 58,7557 | 15,2247 | 28,9 |
| 6 | 58,0702 | 14,7019 | 29,25 |
| 7 | 59,0151 | 15,4948 | 28,78 |
| 8 | ERROR | ERROR | ERROR |
| 9 | 60,0573 | 16,3618 | 28,28 |
| 10 | 76,7255 | 27,0942 | 22,14 |
| 11 | ERROR | ERROR | ERROR |
| 12 | 59,2055 | 14,8516 | 28,69 |
| 13 | ERROR | ERROR | ERROR |
| 14 | 59,2129 | 15,4539 | 28,68 |
| 15 | 59,6759 | 15,4365 | 28,46 |
| 16 | ERROR | ERROR | ERROR |
| 17 | 58,3626 | 15,175 | 29,1 |
| 18 | 58,8681 | 14,8392 | 28,85 |
| 19 | ERROR | ERROR | ERROR |
| 20 | 57,9329 | 14,4771 | 29,32 |
| 21 | 68,7725 | 20,4718 | 24,69 |
| 22 | ERROR | ERROR | ERROR |
| 23 | 58,1634 | 14,8818 | 29,2 |
| 24 | 72,9275 | 20,6155 | 23,29 |
| 25 | ERROR | ERROR | ERROR |
| 26 | 57,6814 | 14,6389 | 29,44 |
| 27 | ERROR | ERROR | ERROR |
| 28 | 58,6434 | 16,0535 | 28,96 |
| 29 | 58,9139 | 14,5542 | 28,83 |
| 30 | ERROR | ERROR | ERROR |
| 31 | 58,9138 | 15,6265 | 28,83 |
| 32 | 59,8244 | 14,843 | 28,39 |
| 33 | ERROR | ERROR | ERROR |
| 34 | 59,5864 | 15,8611 | 28,5 |
| 35 | 75,9927 | 26,5682 | 22,35 |
| 36 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 37 | 59,7826 | 15,6128 | 28,41 |
| 38 | ERROR | ERROR | ERROR |
| 39 | 59,5532 | 15,6887 | 28,52 |
| 40 | 59,7953 | 15,1142 | 28,4 |
| 41 | ERROR | ERROR | ERROR |
| 42 | 57,5821 | 15,4985 | 29,49 |
| 43 | 59,1717 | 15,2998 | 28,7 |
| 44 | ERROR | ERROR | ERROR |
| 45 | 58,9989 | 14,7774 | 28,79 |
| 46 | 68,5193 | 20,2505 | 24,79 |
| 47 | ERROR | ERROR | ERROR |
| 48 | 60,2294 | 15,2238 | 28,2 |
| 49 | 71,4733 | 19,6843 | 23,76 |
| 50 | ERROR | ERROR | ERROR |
| 51 | 57,867 | 14,517 | 29,35 |
| 52 | ERROR | ERROR | ERROR |
| 53 | 58,0724 | 15,747 | 29,24 |
| 54 | 58,4771 | 15,0563 | 29,04 |
| 55 | ERROR | ERROR | ERROR |
| 56 | 57,779 | 14,7399 | 29,39 |
| 57 | 58,9762 | 14,836 | 28,8 |
| 58 | ERROR | ERROR | ERROR |
| 59 | 60,0824 | 16,164 | 28,27 |
| 60 | 78,7296 | 28,2733 | 21,57 |
| 61 | ERROR | ERROR | ERROR |
| 62 | 58,4097 | 14,9835 | 29,08 |
| 63 | ERROR | ERROR | ERROR |
| 64 | 59,1304 | 15,7165 | 28,72 |
| 65 | 59,159 | 14,9595 | 28,71 |
| 66 | ERROR | ERROR | ERROR |
| 67 | 56,9903 | 15,3806 | 29,8 |
| 68 | 58,9323 | 14,7391 | 28,82 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 58,1706 | 14,6666 | 29,2 |
| 71 | 68,3126 | 20,6208 | 24,86 |
| 72 | ERROR | ERROR | ERROR |
| 73 | 59,0451 | 15,4154 | 28,76 |
| 74 | 71,5887 | 20,4488 | 23,72 |
| 75 | ERROR | ERROR | ERROR |
| 76 | 58,7959 | 14,7524 | 28,89 |
| 77 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 78 | 57,1262 | 15,1689 | 29,73 |
| 79 | 59,4982 | 15,363 | 28,54 |
| 80 | ERROR | ERROR | ERROR |
| 81 | 59,3913 | 15,1271 | 28,6 |
| 82 | 59,7434 | 15,5102 | 28,43 |
| 83 | ERROR | ERROR | ERROR |
| 84 | 57,9984 | 15,281 | 29,28 |
| 85 | 75,1638 | 25,9191 | 22,59 |
| 86 | ERROR | ERROR | ERROR |
| 87 | 59,5132 | 15,2755 | 28,54 |
| 88 | ERROR | ERROR | ERROR |
| 89 | 58,7732 | 15,411 | 28,9 |
| 90 | 58,4462 | 14,3872 | 29,06 |
| 91 | ERROR | ERROR | ERROR |
| 92 | 56,3207 | 14,6463 | 30,15 |
| 93 | 59,0519 | 14,8714 | 28,76 |
| 94 | ERROR | ERROR | ERROR |
| 95 | 58,0645 | 14,8797 | 29,25 |
| 96 | 68,6434 | 20,8915 | 24,74 |
| 97 | ERROR | ERROR | ERROR |
| 98 | 59,3377 | 15,284 | 28,62 |

Cuadro C.50: Ordis usando 9 nodos con tamaño de bloque 8

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | ERROR | ERROR | ERROR |
| 1 | ERROR | ERROR | ERROR |
| 2 | 53,9334 | 18,0916 | 31,49 |
| 3 | ERROR | ERROR | ERROR |
| 4 | ERROR | ERROR | ERROR |
| 5 | 55,2522 | 18,1611 | 30,74 |
| 6 | ERROR | ERROR | ERROR |
| 7 | ERROR | ERROR | ERROR |
| 8 | 55,7894 | 18,6001 | 30,44 |
| 9 | ERROR | ERROR | ERROR |
| 10 | 55,121 | 18,0409 | 30,81 |
| 11 | ERROR | ERROR | ERROR |
| 12 | 54,1671 | 18,3565 | 31,35 |
| 13 | 54,6181 | 18,6923 | 31,09 |
| 14 | 55,5241 | 18,0663 | 30,59 |
| 15 | 56,259 | 18,3338 | 30,19 |

| | | | |
|---|---|---|---|
| 16 | 55,65 | 18,6288 | 30,52 |
| 17 | 56,1724 | 18,3487 | 30,23 |
| 18 | 56,2921 | 18,6937 | 30,17 |
| 19 | 55,7969 | 18,347 | 30,44 |
| 20 | 56,1934 | 18,9945 | 30,22 |
| 21 | 55,5201 | 18,7024 | 30,59 |
| 22 | 54,9638 | 17,8807 | 30,9 |
| 23 | 54,9587 | 17,8144 | 30,9 |
| 24 | 54,688 | 18,3486 | 31,05 |
| 25 | 55,8539 | 18,4593 | 30,41 |
| 26 | 55,2802 | 18,1493 | 30,72 |
| 27 | 56,438 | 18,6437 | 30,09 |
| 28 | 54,1281 | 18,3537 | 31,38 |
| 29 | 54,0326 | 18,3665 | 31,43 |
| 30 | 55,3817 | 18,3404 | 30,67 |
| 31 | 54,4929 | 17,8994 | 31,17 |
| 32 | 54,777 | 18,2009 | 31 |
| 33 | 54,9861 | 18,0573 | 30,89 |
| 34 | 54,8938 | 17,8832 | 30,94 |
| 35 | 55,7315 | 18,2846 | 30,47 |
| 36 | 54,7639 | 18,1179 | 31,01 |
| 37 | 54,4049 | 17,8605 | 31,22 |
| 38 | 54,2022 | 17,6639 | 31,33 |
| 39 | 53,8844 | 17,9214 | 31,52 |
| 40 | 54,898 | 17,9885 | 30,94 |
| 41 | 55,9364 | 18,4312 | 30,36 |
| 42 | 54,5126 | 18,036 | 31,15 |
| 43 | 54,932 | 18,3851 | 30,92 |
| 44 | 54,1952 | 18,5493 | 31,34 |
| 45 | 55,3813 | 18,6949 | 30,67 |
| 46 | 55,8706 | 18,6498 | 30,4 |
| 47 | 54,4493 | 17,858 | 31,19 |
| 48 | 53,8557 | 17,5309 | 31,53 |
| 49 | 55,9215 | 18,5945 | 30,37 |
| 50 | ERROR | ERROR | ERROR |
| 51 | 56,2708 | 18,6894 | 30,18 |
| 52 | ERROR | ERROR | ERROR |
| 53 | 55,8486 | 18,4083 | 30,41 |
| 54 | 55,9157 | 18,2522 | 30,37 |
| 55 | ERROR | ERROR | ERROR |
| 56 | 55,721 | 18,0403 | 30,48 |

| | | | |
|---|---|---|---|
| 57 | 55,5082 | 18,3071 | 30,6 |
| 58 | ERROR | ERROR | ERROR |
| 59 | 55,072 | 18,0773 | 30,84 |
| 60 | 54,4561 | 17,8705 | 31,19 |
| 61 | ERROR | ERROR | ERROR |
| 62 | 53,1404 | 18,0664 | 31,96 |
| 63 | 54,7742 | 18,2524 | 31,01 |
| 64 | ERROR | ERROR | ERROR |
| 65 | 55,2432 | 18,5257 | 30,74 |
| 66 | 55,4827 | 18,0611 | 30,61 |
| 67 | ERROR | ERROR | ERROR |
| 68 | 53,6925 | 18,3403 | 31,63 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 54,5366 | 18,1864 | 31,14 |
| 71 | 55,3578 | 18,2179 | 30,68 |
| 72 | ERROR | ERROR | ERROR |
| 73 | 55,0552 | 17,7571 | 30,85 |
| 74 | ERROR | ERROR | ERROR |
| 75 | 54,4917 | 18,3413 | 31,17 |
| 76 | 55,4796 | 18,2641 | 30,61 |
| 77 | ERROR | ERROR | ERROR |
| 78 | 56,2149 | 18,6454 | 30,21 |
| 79 | ERROR | ERROR | ERROR |
| 80 | 56,2992 | 18,5123 | 30,17 |
| 81 | 56,9995 | 19,17 | 29,8 |
| 82 | ERROR | ERROR | ERROR |
| 83 | 54,1569 | 17,6939 | 31,36 |
| 84 | ERROR | ERROR | ERROR |
| 85 | 54,4353 | 18,1034 | 31,2 |
| 86 | 56,187 | 18,9924 | 30,23 |
| 87 | ERROR | ERROR | ERROR |
| 88 | 55,7144 | 18,9831 | 30,48 |
| 89 | 55,4733 | 18,8503 | 30,62 |
| 90 | ERROR | ERROR | ERROR |
| 91 | 54,8375 | 17,581 | 30,97 |
| 92 | ERROR | ERROR | ERROR |
| 93 | 54,3247 | 18,1277 | 31,26 |
| 94 | 55,6809 | 18,4175 | 30,5 |
| 95 | ERROR | ERROR | ERROR |
| 96 | 54,9666 | 17,5981 | 30,9 |
| 97 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 98 | 53,7657 | 17,9951 | 31,59 |
| 99 | 57,1715 | 18,7212 | 29,71 |

Cuadro C.51: Ordis usando 9 nodos con tamaño de bloque 16

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 70,1865 | 20,5937 | 24,2 |
| 1 | ERROR | ERROR | ERROR |
| 2 | 58,3964 | 15,7516 | 29,08 |
| 3 | ERROR | ERROR | ERROR |
| 4 | 56,2662 | 15,5091 | 30,18 |
| 5 | 57,0316 | 15,4991 | 29,78 |
| 6 | ERROR | ERROR | ERROR |
| 7 | 56,9412 | 15,3618 | 29,83 |
| 8 | 58,0156 | 16,1668 | 29,27 |
| 9 | ERROR | ERROR | ERROR |
| 10 | 59,2263 | 16,7061 | 28,68 |
| 11 | 80,1953 | 30,0898 | 21,18 |
| 12 | ERROR | ERROR | ERROR |
| 13 | 57,7 | 15,5037 | 29,43 |
| 14 | ERROR | ERROR | ERROR |
| 15 | 57,8558 | 15,648 | 29,35 |
| 16 | 59,204 | 16,0944 | 28,69 |
| 17 | ERROR | ERROR | ERROR |
| 18 | 56,6723 | 15,6124 | 29,97 |
| 19 | 58,6606 | 16,5238 | 28,95 |
| 20 | ERROR | ERROR | ERROR |
| 21 | 57,9049 | 15,8469 | 29,33 |
| 22 | 69,3609 | 21,5063 | 24,49 |
| 23 | ERROR | ERROR | ERROR |
| 24 | 58,6306 | 16,2081 | 28,97 |
| 25 | 70,5141 | 20,347 | 24,08 |
| 26 | ERROR | ERROR | ERROR |
| 27 | 56,5351 | 15,105 | 30,04 |
| 28 | ERROR | ERROR | ERROR |
| 29 | 56,9885 | 16,0062 | 29,8 |
| 30 | 57,6412 | 15,7741 | 29,46 |
| 31 | ERROR | ERROR | ERROR |
| 32 | 57,3068 | 15,6473 | 29,64 |
| 33 | 57,1169 | 15,4817 | 29,73 |
| 34 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 35 | 57,9315 | 16,1917 | 29,32 |
| 36 | 77,1388 | 28,5309 | 22,02 |
| 37 | ERROR | ERROR | ERROR |
| 38 | 58,2252 | 15,7594 | 29,17 |
| 39 | ERROR | ERROR | ERROR |
| 40 | 57,2156 | 15,584 | 29,68 |
| 41 | 56,8656 | 15,3744 | 29,87 |
| 42 | ERROR | ERROR | ERROR |
| 43 | 57,2602 | 15,6684 | 29,66 |
| 44 | ERROR | ERROR | ERROR |
| 45 | ERROR | ERROR | ERROR |
| 46 | 56,8719 | 15,3102 | 29,86 |
| 47 | ERROR | ERROR | ERROR |
| 48 | ERROR | ERROR | ERROR |
| 49 | 58,0691 | 16,7842 | 29,25 |
| 50 | ERROR | ERROR | ERROR |
| 51 | ERROR | ERROR | ERROR |
| 52 | ERROR | ERROR | ERROR |
| 53 | ERROR | ERROR | ERROR |
| 54 | 57,2101 | 15,8065 | 29,69 |
| 55 | ERROR | ERROR | ERROR |
| 56 | ERROR | ERROR | ERROR |
| 57 | 57,3174 | 15,6183 | 29,63 |
| 58 | ERROR | ERROR | ERROR |
| 59 | ERROR | ERROR | ERROR |
| 60 | 58,486 | 16,9293 | 29,04 |
| 61 | ERROR | ERROR | ERROR |
| 62 | ERROR | ERROR | ERROR |
| 63 | 57,3103 | 15,8166 | 29,63 |
| 64 | ERROR | ERROR | ERROR |
| 65 | 57,7426 | 16,256 | 29,41 |
| 66 | ERROR | ERROR | ERROR |
| 67 | ERROR | ERROR | ERROR |
| 68 | 56,5316 | 15,7323 | 30,04 |
| 69 | ERROR | ERROR | ERROR |
| 70 | ERROR | ERROR | ERROR |
| 71 | 58,2586 | 15,8782 | 29,15 |
| 72 | ERROR | ERROR | ERROR |
| 73 | ERROR | ERROR | ERROR |
| 74 | 56,1266 | 15,2325 | 30,26 |
| 75 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 76 | ERROR | ERROR | ERROR |
| 77 | ERROR | ERROR | ERROR |
| 78 | ERROR | ERROR | ERROR |
| 79 | 57,3368 | 15,7478 | 29,62 |
| 80 | ERROR | ERROR | ERROR |
| 81 | ERROR | ERROR | ERROR |
| 82 | 69,6998 | 21,5285 | 24,37 |
| 83 | ERROR | ERROR | ERROR |
| 84 | ERROR | ERROR | ERROR |
| 85 | ERROR | ERROR | ERROR |
| 86 | ERROR | ERROR | ERROR |
| 87 | 56,5961 | 15,0719 | 30,01 |
| 88 | ERROR | ERROR | ERROR |
| 89 | ERROR | ERROR | ERROR |
| 90 | 68,3619 | 21,0298 | 24,84 |
| 91 | ERROR | ERROR | ERROR |
| 92 | ERROR | ERROR | ERROR |
| 93 | ERROR | ERROR | ERROR |
| 94 | ERROR | ERROR | ERROR |
| 95 | 57,8652 | 15,4615 | 29,35 |
| 96 | ERROR | ERROR | ERROR |
| 97 | ERROR | ERROR | ERROR |
| 98 | 68,8123 | 21,0784 | 24,68 |

Cuadro C.52: Ordis usando 9 nodos con tamaño de bloque 32

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 55,038 | 13,5906 | 30,86 |
| 1 | 55,0694 | 13,6017 | 30,84 |
| 2 | 56,8656 | 14,3537 | 29,87 |
| 3 | ERROR | ERROR | ERROR |
| 4 | ERROR | ERROR | ERROR |
| 5 | 55,8075 | 13,8433 | 30,43 |
| 6 | ERROR | ERROR | ERROR |
| 7 | 54,8181 | 13,4994 | 30,98 |
| 8 | ERROR | ERROR | ERROR |
| 9 | ERROR | ERROR | ERROR |
| 10 | 54,543 | 13,442 | 31,14 |
| 11 | ERROR | ERROR | ERROR |
| 12 | 54,8737 | 13,4908 | 30,95 |
| 13 | 54,53 | 13,4435 | 31,14 |

| | | | |
|---|---|---|---|
| 14 | 54,5747 | 13,613 | 31,12 |
| 15 | 54,4444 | 13,3913 | 31,19 |
| 16 | 54,8308 | 13,5665 | 30,97 |
| 17 | 58,2721 | 15,2499 | 29,14 |
| 18 | 54,7767 | 13,4712 | 31 |
| 19 | 54,4361 | 13,3871 | 31,2 |
| 20 | 54,6112 | 13,5687 | 31,1 |
| 21 | 54,8864 | 13,5923 | 30,94 |
| 22 | 54,8289 | 13,4953 | 30,97 |
| 23 | 56,681 | 14,3068 | 29,96 |
| 24 | 54,5619 | 13,5015 | 31,13 |
| 25 | 54,5791 | 13,4361 | 31,12 |
| 26 | 54,8285 | 13,4824 | 30,98 |
| 27 | 54,7955 | 13,6224 | 30,99 |
| 28 | 55,1661 | 13,6881 | 30,79 |
| 29 | 55,0897 | 13,6613 | 30,83 |
| 30 | 54,5352 | 13,3877 | 31,14 |
| 31 | 54,544 | 13,4261 | 31,14 |
| 32 | 54,6215 | 13,4746 | 31,09 |
| 33 | 55,0933 | 13,645 | 30,83 |
| 34 | 57,3947 | 14,4778 | 29,59 |
| 35 | ERROR | ERROR | ERROR |
| 36 | ERROR | ERROR | ERROR |
| 37 | 54,5479 | 13,4705 | 31,13 |
| 38 | ERROR | ERROR | ERROR |
| 39 | ERROR | ERROR | ERROR |
| 40 | 54,5087 | 13,4473 | 31,16 |
| 41 | ERROR | ERROR | ERROR |

Cuadro C.53: Ordis usando 16 nodos con tamaño de bloque 4

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | ERROR | ERROR | ERROR |
| 1 | ERROR | ERROR | ERROR |
| 2 | ERROR | ERROR | ERROR |
| 3 | ERROR | ERROR | ERROR |
| 4 | ERROR | ERROR | ERROR |
| 5 | ERROR | ERROR | ERROR |
| 6 | ERROR | ERROR | ERROR |
| 7 | ERROR | ERROR | ERROR |
| 8 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 9 | ERROR | ERROR | ERROR |
| 10 | ERROR | ERROR | ERROR |
| 11 | ERROR | ERROR | ERROR |
| 12 | ERROR | ERROR | ERROR |
| 13 | ERROR | ERROR | ERROR |
| 14 | ERROR | ERROR | ERROR |
| 15 | ERROR | ERROR | ERROR |
| 16 | ERROR | ERROR | ERROR |
| 17 | ERROR | ERROR | ERROR |
| 18 | ERROR | ERROR | ERROR |
| 19 | ERROR | ERROR | ERROR |
| 20 | ERROR | ERROR | ERROR |
| 21 | ERROR | ERROR | ERROR |
| 22 | ERROR | ERROR | ERROR |
| 23 | ERROR | ERROR | ERROR |
| 24 | ERROR | ERROR | ERROR |
| 25 | ERROR | ERROR | ERROR |
| 26 | ERROR | ERROR | ERROR |
| 27 | ERROR | ERROR | ERROR |
| 28 | ERROR | ERROR | ERROR |
| 29 | ERROR | ERROR | ERROR |
| 30 | ERROR | ERROR | ERROR |
| 31 | ERROR | ERROR | ERROR |
| 32 | ERROR | ERROR | ERROR |
| 33 | ERROR | ERROR | ERROR |
| 34 | ERROR | ERROR | ERROR |
| 35 | ERROR | ERROR | ERROR |
| 36 | ERROR | ERROR | ERROR |
| 37 | ERROR | ERROR | ERROR |
| 38 | ERROR | ERROR | ERROR |
| 39 | ERROR | ERROR | ERROR |
| 40 | ERROR | ERROR | ERROR |
| 41 | ERROR | ERROR | ERROR |
| 42 | ERROR | ERROR | ERROR |
| 43 | ERROR | ERROR | ERROR |
| 44 | ERROR | ERROR | ERROR |
| 45 | ERROR | ERROR | ERROR |
| 46 | ERROR | ERROR | ERROR |
| 47 | ERROR | ERROR | ERROR |
| 48 | ERROR | ERROR | ERROR |
| 49 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 50 | ERROR | ERROR | ERROR |
| 51 | ERROR | ERROR | ERROR |
| 52 | ERROR | ERROR | ERROR |
| 53 | ERROR | ERROR | ERROR |
| 54 | ERROR | ERROR | ERROR |
| 55 | ERROR | ERROR | ERROR |
| 56 | ERROR | ERROR | ERROR |
| 57 | ERROR | ERROR | ERROR |
| 58 | ERROR | ERROR | ERROR |
| 59 | ERROR | ERROR | ERROR |
| 60 | ERROR | ERROR | ERROR |
| 61 | ERROR | ERROR | ERROR |
| 62 | ERROR | ERROR | ERROR |
| 63 | ERROR | ERROR | ERROR |
| 64 | ERROR | ERROR | ERROR |
| 65 | ERROR | ERROR | ERROR |
| 66 | ERROR | ERROR | ERROR |
| 67 | ERROR | ERROR | ERROR |
| 68 | 36,2271 | 13,754 | 46,88 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 36,4995 | 13,8242 | 46,53 |
| 71 | ERROR | ERROR | ERROR |
| 72 | ERROR | ERROR | ERROR |
| 73 | 36,4822 | 13,8744 | 46,55 |
| 74 | ERROR | ERROR | ERROR |
| 75 | 36,5493 | 13,7926 | 46,47 |
| 76 | 36,1774 | 13,7891 | 46,94 |
| 77 | 36,1121 | 13,6568 | 47,03 |
| 78 | 35,986 | 13,6723 | 47,19 |
| 79 | 37,0822 | 14,1589 | 45,8 |
| 80 | 35,9844 | 13,7116 | 47,2 |
| 81 | 35,8419 | 13,5788 | 47,38 |
| 82 | 37,7312 | 14,4339 | 45,01 |
| 83 | 36,1511 | 13,7822 | 46,98 |
| 84 | 36,1583 | 13,687 | 46,97 |
| 85 | ERROR | ERROR | ERROR |
| 86 | 36,5747 | 13,8897 | 46,43 |
| 87 | 36,165 | 13,7021 | 46,96 |
| 88 | 35,6609 | 13,6515 | 47,62 |
| 89 | 36,9253 | 13,8758 | 45,99 |
| 90 | 36,7232 | 13,9193 | 46,25 |

| | | | |
|---|---|---|---|
| 91 | 36,1758 | 13,7305 | 46,95 |
| 92 | 38,1126 | 14,4861 | 44,56 |
| 93 | 36,1741 | 13,7482 | 46,95 |
| 94 | 37,268 | 14,0382 | 45,57 |
| 95 | 37,2499 | 14,1159 | 45,59 |
| 96 | 36,0637 | 13,6393 | 47,09 |
| 97 | 36,0955 | 13,6693 | 47,05 |
| 98 | 36,7714 | 13,9988 | 46,19 |
| 99 | 36,0881 | 13,781 | 47,06 |

Cuadro C.54: Ordis usando 16 nodos con tamaño de bloque 8

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | ERROR | ERROR | ERROR |
| 1 | 34,8859 | 15,2349 | 48,68 |
| 2 | ERROR | ERROR | ERROR |
| 3 | ERROR | ERROR | ERROR |
| 4 | 35,3697 | 15,5033 | 48,02 |
| 5 | ERROR | ERROR | ERROR |
| 6 | ERROR | ERROR | ERROR |
| 7 | 34,3588 | 15,127 | 49,43 |
| 8 | ERROR | ERROR | ERROR |
| 9 | 34,5939 | 15,3883 | 49,09 |
| 10 | ERROR | ERROR | ERROR |
| 11 | ERROR | ERROR | ERROR |
| 12 | ERROR | ERROR | ERROR |
| 13 | ERROR | ERROR | ERROR |
| 14 | 33,8917 | 14,9479 | 50,11 |
| 15 | ERROR | ERROR | ERROR |
| 16 | ERROR | ERROR | ERROR |
| 17 | ERROR | ERROR | ERROR |
| 18 | 35,2481 | 15,4925 | 48,18 |
| 19 | ERROR | ERROR | ERROR |
| 20 | ERROR | ERROR | ERROR |
| 21 | 35,5853 | 15,6753 | 47,73 |
| 22 | ERROR | ERROR | ERROR |
| 23 | ERROR | ERROR | ERROR |
| 24 | ERROR | ERROR | ERROR |
| 25 | 34,2047 | 15,0817 | 49,65 |
| 26 | ERROR | ERROR | ERROR |
| 27 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 28 | ERROR | ERROR | ERROR |
| 29 | ERROR | ERROR | ERROR |
| 30 | ERROR | ERROR | ERROR |
| 31 | ERROR | ERROR | ERROR |
| 32 | ERROR | ERROR | ERROR |
| 33 | ERROR | ERROR | ERROR |
| 34 | ERROR | ERROR | ERROR |
| 35 | ERROR | ERROR | ERROR |
| 36 | ERROR | ERROR | ERROR |
| 37 | ERROR | ERROR | ERROR |
| 38 | ERROR | ERROR | ERROR |
| 39 | ERROR | ERROR | ERROR |
| 40 | 34,2064 | 15,1204 | 49,65 |
| 41 | ERROR | ERROR | ERROR |
| 42 | ERROR | ERROR | ERROR |
| 43 | ERROR | ERROR | ERROR |
| 44 | 33,4876 | 14,7801 | 50,71 |
| 45 | ERROR | ERROR | ERROR |
| 46 | 34,0036 | 15,0066 | 49,95 |
| 47 | ERROR | ERROR | ERROR |
| 48 | ERROR | ERROR | ERROR |
| 49 | 34,0758 | 14,9894 | 49,84 |
| 50 | ERROR | ERROR | ERROR |
| 51 | 34,1227 | 15,0955 | 49,77 |
| 52 | ERROR | ERROR | ERROR |
| 53 | ERROR | ERROR | ERROR |
| 54 | 34,2076 | 14,9816 | 49,65 |
| 55 | ERROR | ERROR | ERROR |
| 56 | 34,3605 | 15,0793 | 49,43 |
| 57 | ERROR | ERROR | ERROR |
| 58 | 35,9789 | 15,4628 | 47,2 |
| 59 | ERROR | ERROR | ERROR |
| 60 | ERROR | ERROR | ERROR |
| 61 | 33,8208 | 15,016 | 50,22 |
| 62 | ERROR | ERROR | ERROR |
| 63 | ERROR | ERROR | ERROR |
| 64 | ERROR | ERROR | ERROR |
| 65 | ERROR | ERROR | ERROR |
| 66 | ERROR | ERROR | ERROR |
| 67 | ERROR | ERROR | ERROR |
| 68 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 69 | ERROR | ERROR | ERROR |
| 70 | ERROR | ERROR | ERROR |
| 71 | ERROR | ERROR | ERROR |
| 72 | ERROR | ERROR | ERROR |
| 73 | ERROR | ERROR | ERROR |
| 74 | ERROR | ERROR | ERROR |
| 75 | ERROR | ERROR | ERROR |
| 76 | ERROR | ERROR | ERROR |
| 77 | ERROR | ERROR | ERROR |
| 78 | ERROR | ERROR | ERROR |
| 79 | ERROR | ERROR | ERROR |
| 80 | 34,2636 | 15,0088 | 49,57 |
| 81 | ERROR | ERROR | ERROR |
| 82 | ERROR | ERROR | ERROR |
| 83 | 33,8355 | 14,9849 | 50,19 |
| 84 | ERROR | ERROR | ERROR |
| 85 | 34,2673 | 15,0909 | 49,56 |
| 86 | ERROR | ERROR | ERROR |
| 87 | ERROR | ERROR | ERROR |
| 88 | 33,9682 | 15,06 | 50 |
| 89 | ERROR | ERROR | ERROR |
| 90 | ERROR | ERROR | ERROR |
| 91 | 34,5808 | 15,0868 | 49,11 |
| 92 | ERROR | ERROR | ERROR |
| 93 | 36,305 | 15,6942 | 46,78 |
| 94 | ERROR | ERROR | ERROR |
| 95 | ERROR | ERROR | ERROR |
| 96 | 33,7851 | 14,9924 | 50,27 |

Cuadro C.55: Ordis usando 16 nodos con tamaño de bloque 16

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | ERROR | ERROR | ERROR |
| 1 | 37,8275 | 14,8143 | 44,9 |
| 2 | ERROR | ERROR | ERROR |
| 3 | ERROR | ERROR | ERROR |
| 4 | 38,4854 | 14,9475 | 44,13 |
| 5 | ERROR | ERROR | ERROR |
| 6 | 39,6113 | 15,753 | 42,87 |
| 7 | ERROR | ERROR | ERROR |
| 8 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 9 | 37,8273 | 14,785 | 44,9 |
| 10 | ERROR | ERROR | ERROR |
| 11 | ERROR | ERROR | ERROR |
| 12 | ERROR | ERROR | ERROR |
| 13 | ERROR | ERROR | ERROR |
| 14 | ERROR | ERROR | ERROR |
| 15 | ERROR | ERROR | ERROR |
| 16 | ERROR | ERROR | ERROR |
| 17 | 37,6088 | 14,8201 | 45,16 |
| 18 | ERROR | ERROR | ERROR |
| 19 | ERROR | ERROR | ERROR |
| 20 | 37,2367 | 14,7785 | 45,61 |
| 21 | ERROR | ERROR | ERROR |
| 22 | ERROR | ERROR | ERROR |
| 23 | 37,3062 | 14,7803 | 45,52 |
| 24 | ERROR | ERROR | ERROR |
| 25 | ERROR | ERROR | ERROR |
| 26 | 37,3249 | 14,8105 | 45,5 |
| 27 | ERROR | ERROR | ERROR |
| 28 | ERROR | ERROR | ERROR |
| 29 | 37,7559 | 14,8328 | 44,98 |
| 30 | ERROR | ERROR | ERROR |
| 31 | ERROR | ERROR | ERROR |
| 32 | 37,906 | 14,9534 | 44,8 |
| 33 | ERROR | ERROR | ERROR |
| 34 | ERROR | ERROR | ERROR |
| 35 | 37,9159 | 14,879 | 44,79 |
| 36 | ERROR | ERROR | ERROR |
| 37 | ERROR | ERROR | ERROR |
| 38 | ERROR | ERROR | ERROR |
| 39 | ERROR | ERROR | ERROR |
| 40 | 38,527 | 14,9741 | 44,08 |
| 41 | ERROR | ERROR | ERROR |
| 42 | ERROR | ERROR | ERROR |
| 43 | ERROR | ERROR | ERROR |
| 44 | ERROR | ERROR | ERROR |
| 45 | ERROR | ERROR | ERROR |
| 46 | ERROR | ERROR | ERROR |
| 47 | 37,803 | 14,853 | 44,93 |
| 48 | ERROR | ERROR | ERROR |
| 49 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 50 | 37,8237 | 14,8895 | 44,9 |
| 51 | ERROR | ERROR | ERROR |
| 52 | ERROR | ERROR | ERROR |
| 53 | ERROR | ERROR | ERROR |
| 54 | ERROR | ERROR | ERROR |
| 55 | 37,5458 | 14,7681 | 45,23 |
| 56 | ERROR | ERROR | ERROR |
| 57 | ERROR | ERROR | ERROR |
| 58 | ERROR | ERROR | ERROR |
| 59 | ERROR | ERROR | ERROR |
| 60 | 37,5841 | 14,8255 | 45,19 |
| 61 | ERROR | ERROR | ERROR |
| 62 | ERROR | ERROR | ERROR |
| 63 | ERROR | ERROR | ERROR |
| 64 | ERROR | ERROR | ERROR |
| 65 | 37,222 | 14,7178 | 45,63 |
| 66 | ERROR | ERROR | ERROR |
| 67 | ERROR | ERROR | ERROR |
| 68 | ERROR | ERROR | ERROR |
| 69 | ERROR | ERROR | ERROR |
| 70 | 38,1671 | 14,9195 | 44,5 |
| 71 | ERROR | ERROR | ERROR |
| 72 | ERROR | ERROR | ERROR |
| 73 | ERROR | ERROR | ERROR |
| 74 | ERROR | ERROR | ERROR |
| 75 | 37,3591 | 14,8167 | 45,46 |
| 76 | ERROR | ERROR | ERROR |
| 77 | ERROR | ERROR | ERROR |
| 78 | ERROR | ERROR | ERROR |
| 79 | ERROR | ERROR | ERROR |
| 80 | 37,3511 | 14,7748 | 45,47 |
| 81 | ERROR | ERROR | ERROR |
| 82 | ERROR | ERROR | ERROR |
| 83 | ERROR | ERROR | ERROR |
| 84 | ERROR | ERROR | ERROR |
| 85 | 38,0227 | 14,9117 | 44,67 |
| 86 | ERROR | ERROR | ERROR |
| 87 | ERROR | ERROR | ERROR |
| 88 | ERROR | ERROR | ERROR |
| 89 | ERROR | ERROR | ERROR |
| 90 | 37,6512 | 14,8466 | 45,11 |

| | | | |
|---|---|---|---|
| 91 | ERROR | ERROR | ERROR |
| 92 | ERROR | ERROR | ERROR |
| 93 | ERROR | ERROR | ERROR |
| 94 | 37,5114 | 14,7862 | 45,27 |
| 95 | ERROR | ERROR | ERROR |
| 96 | ERROR | ERROR | ERROR |
| 97 | ERROR | ERROR | ERROR |
| 98 | 37,6564 | 14,8552 | 45,1 |
| 99 | 37,3076 | 14,6957 | 45,52 |

Cuadro C.56: Ordis usando 16 nodos con tamaño de bloque 32

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 47,9892 | 14,2985 | 35,38 |
| 1 | 58,4184 | 17,8473 | 29,07 |
| 2 | ERROR | ERROR | ERROR |
| 3 | ERROR | ERROR | ERROR |
| 4 | ERROR | ERROR | ERROR |
| 5 | ERROR | ERROR | ERROR |
| 6 | ERROR | ERROR | ERROR |
| 7 | ERROR | ERROR | ERROR |
| 8 | ERROR | ERROR | ERROR |
| 9 | 42,7141 | 13,2372 | 39,75 |
| 10 | 47,3458 | 13,4769 | 35,86 |
| 11 | 65,5814 | 17,7594 | 25,89 |
| 12 | 50,1593 | 14,4116 | 33,85 |
| 13 | 46,2502 | 13,5189 | 36,71 |
| 14 | 50,7627 | 14,0495 | 33,45 |
| 15 | 44,6281 | 13,6445 | 38,05 |
| 16 | ERROR | ERROR | ERROR |
| 17 | 48,3937 | 14,901 | 35,09 |
| 18 | ERROR | ERROR | ERROR |
| 19 | ERROR | ERROR | ERROR |
| 20 | ERROR | ERROR | ERROR |
| 21 | ERROR | ERROR | ERROR |
| 22 | ERROR | ERROR | ERROR |
| 23 | ERROR | ERROR | ERROR |
| 24 | ERROR | ERROR | ERROR |
| 25 | ERROR | ERROR | ERROR |
| 26 | ERROR | ERROR | ERROR |
| 27 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 28 | ERROR | ERROR | ERROR |
| 29 | ERROR | ERROR | ERROR |
| 30 | ERROR | ERROR | ERROR |
| 31 | ERROR | ERROR | ERROR |
| 32 | ERROR | ERROR | ERROR |
| 33 | ERROR | ERROR | ERROR |
| 34 | ERROR | ERROR | ERROR |
| 35 | 46,8922 | 14,099 | 36,21 |
| 36 | 46,9348 | 14,1516 | 36,18 |
| 37 | 47,1367 | 14,1944 | 36,02 |
| 38 | 47,1537 | 14,198 | 36,01 |
| 39 | 46,6702 | 14,0526 | 36,38 |
| 40 | 46,6778 | 14,1112 | 36,38 |
| 41 | 46,9153 | 14,0826 | 36,19 |
| 42 | 47,0374 | 14,1617 | 36,1 |
| 43 | 46,6595 | 14,1335 | 36,39 |
| 44 | 46,8358 | 14,1142 | 36,25 |
| 45 | 46,9406 | 14,152 | 36,17 |
| 46 | 46,8462 | 14,1107 | 36,25 |
| 47 | 46,7018 | 14,1247 | 36,36 |
| 48 | 46,6832 | 14,1153 | 36,37 |
| 49 | 46,9058 | 14,139 | 36,2 |
| 50 | 46,7126 | 14,0932 | 36,35 |
| 51 | 46,879 | 14,1418 | 36,22 |
| 52 | 46,878 | 14,1504 | 36,22 |
| 53 | 46,8535 | 14,1369 | 36,24 |
| 54 | 46,8882 | 14,1236 | 36,21 |
| 55 | 46,6859 | 14,1086 | 36,37 |
| 56 | 47,1364 | 14,1618 | 36,02 |
| 57 | 48,2043 | 14,267 | 35,22 |
| 58 | 47,6421 | 14,3041 | 35,64 |
| 59 | 47,19 | 14,1514 | 35,98 |
| 60 | 46,662 | 14,0702 | 36,39 |
| 61 | 46,7558 | 14,1021 | 36,32 |
| 62 | 46,6674 | 14,1175 | 36,38 |
| 63 | 46,6681 | 14,0999 | 36,38 |
| 64 | 46,6179 | 14,1049 | 36,42 |
| 65 | 47,1414 | 14,112 | 36,02 |
| 66 | 47,56 | 14,289 | 35,7 |
| 67 | 46,9221 | 14,1492 | 36,19 |
| 68 | 46,6462 | 14,1054 | 36,4 |

| | | | |
|---|---|---|---|
| 69 | 47,1324 | 14,1551 | 36,03 |
| 70 | 47,6149 | 14,2872 | 35,66 |
| 71 | 47,1383 | 14,1359 | 36,02 |
| 72 | 46,9016 | 14,0993 | 36,2 |
| 73 | 46,6948 | 14,0706 | 36,36 |
| 74 | 47,0993 | 14,1595 | 36,05 |
| 75 | 46,8691 | 14,1252 | 36,23 |
| 76 | 46,8401 | 14,1256 | 36,25 |
| 77 | 46,8159 | 14,0839 | 36,27 |
| 78 | 46,7138 | 14,1023 | 36,35 |
| 79 | 46,6504 | 14,0653 | 36,4 |
| 80 | 46,661 | 14,1032 | 36,39 |
| 81 | 46,8496 | 14,0823 | 36,24 |
| 82 | 47,047 | 14,1312 | 36,09 |
| 83 | 46,7029 | 14,1292 | 36,36 |
| 84 | 47,3577 | 14,2104 | 35,85 |
| 85 | 47,1978 | 14,2294 | 35,98 |
| 86 | 46,9654 | 14,0741 | 36,15 |
| 87 | 46,6998 | 14,1016 | 36,36 |
| 88 | 47,1057 | 14,1618 | 36,05 |
| 89 | 46,9072 | 14,0942 | 36,2 |
| 90 | 46,6854 | 14,1184 | 36,37 |
| 91 | 46,8455 | 14,1035 | 36,25 |
| 92 | 47,9603 | 14,3061 | 35,4 |
| 93 | 46,6597 | 14,0706 | 36,39 |
| 94 | 46,8841 | 14,1532 | 36,22 |
| 95 | 46,9407 | 14,1278 | 36,17 |
| 96 | 46,8458 | 14,1586 | 36,25 |
| 97 | 46,6239 | 14,0698 | 36,42 |
| 98 | 46,9175 | 14,1068 | 36,19 |
| 99 | 47,0615 | 14,1833 | 36,08 |

Cuadro C.57: Ordis usando 25 nodos con tamaño de bloque 4

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 34,9725 | 14,4063 | 48,55 |
| 1 | 34,7686 | 14,3555 | 48,84 |
| 2 | 34,7496 | 14,3141 | 48,86 |
| 3 | 35,2249 | 14,4557 | 48,2 |
| 4 | 35,4949 | 14,4972 | 47,84 |
| 5 | 35,0457 | 14,4112 | 48,45 |

| | | | |
|---|---|---|---|
| 6 | 34,9525 | 14,3843 | 48,58 |
| 7 | 34,583 | 14,3566 | 49,1 |
| 8 | 34,8017 | 14,3534 | 48,79 |
| 9 | 35,2106 | 14,3757 | 48,22 |
| 10 | 35,1048 | 14,4426 | 48,37 |
| 11 | 35,4064 | 14,4506 | 47,96 |
| 12 | 34,7749 | 14,3599 | 48,83 |
| 13 | 34,761 | 14,3853 | 48,85 |
| 14 | 34,9302 | 14,3407 | 48,61 |
| 15 | 35,122 | 14,4767 | 48,34 |
| 16 | 34,5847 | 14,3433 | 49,1 |
| 17 | 34,7536 | 14,3375 | 48,86 |
| 18 | 34,9871 | 14,3896 | 48,53 |
| 19 | 35,4635 | 14,4956 | 47,88 |
| 20 | 34,9938 | 14,4003 | 48,52 |
| 21 | 34,5074 | 14,2832 | 49,21 |
| 22 | 34,826 | 14,385 | 48,76 |
| 23 | 36,3473 | 14,5776 | 46,71 |
| 24 | 35,9891 | 14,6147 | 47,18 |
| 25 | 35,5892 | 14,5294 | 47,71 |
| 26 | 34,5798 | 14,3474 | 49,1 |
| 27 | 34,5036 | 14,2839 | 49,21 |
| 28 | 34,9117 | 14,394 | 48,64 |
| 29 | 35,3166 | 14,4665 | 48,08 |
| 30 | 35,8245 | 14,5738 | 47,4 |
| 31 | 34,9698 | 14,4136 | 48,55 |
| 32 | 34,974 | 14,4025 | 48,55 |
| 33 | 35,704 | 14,5464 | 47,56 |
| 34 | 35,0177 | 14,4068 | 48,49 |
| 35 | 34,5797 | 14,3511 | 49,1 |
| 36 | 34,6098 | 14,3385 | 49,06 |
| 37 | 34,5671 | 14,3658 | 49,12 |
| 38 | 34,5597 | 14,2982 | 49,13 |
| 39 | 34,7879 | 14,371 | 48,81 |
| 40 | 34,7828 | 14,3371 | 48,82 |
| 41 | 34,7752 | 14,3848 | 48,83 |
| 42 | 34,5453 | 14,3191 | 49,15 |
| 43 | 34,9543 | 14,4211 | 48,58 |
| 44 | 34,7538 | 14,4048 | 48,86 |
| 45 | 34,9868 | 14,4003 | 48,53 |
| 46 | 34,9551 | 14,4161 | 48,58 |

| | | | |
|---|---|---|---|
| 47 | 35,2969 | 14,4319 | 48,1 |
| 48 | 35,6548 | 14,5441 | 47,62 |
| 49 | 35,7044 | 14,5986 | 47,56 |
| 50 | 34,7639 | 14,3841 | 48,84 |
| 51 | 34,8301 | 14,3726 | 48,75 |
| 52 | 35,0114 | 14,3777 | 48,5 |
| 53 | 35,4301 | 14,497 | 47,92 |
| 54 | 35,1734 | 14,426 | 48,27 |
| 55 | 34,8125 | 14,3894 | 48,77 |
| 56 | 34,5522 | 14,2905 | 49,14 |
| 57 | 34,818 | 14,4009 | 48,77 |
| 58 | 34,7955 | 14,3682 | 48,8 |
| 59 | 34,7758 | 14,3874 | 48,83 |
| 60 | 34,7681 | 14,3838 | 48,84 |
| 61 | 35,3394 | 14,4848 | 48,05 |
| 62 | 35,2342 | 14,4327 | 48,19 |
| 63 | 34,9533 | 14,4061 | 48,58 |
| 64 | 35,0571 | 14,4021 | 48,43 |
| 65 | 35,9658 | 14,5739 | 47,21 |
| 66 | 34,9672 | 14,4137 | 48,56 |
| 67 | 35,6734 | 14,5263 | 47,6 |
| 68 | 34,5834 | 14,3557 | 49,1 |
| 69 | 34,7011 | 14,337 | 48,93 |
| 70 | 35,4691 | 14,4626 | 47,87 |
| 71 | 34,7744 | 14,3538 | 48,83 |
| 72 | 34,9477 | 14,4251 | 48,59 |
| 73 | 35,0008 | 14,3922 | 48,51 |
| 74 | 34,775 | 14,3355 | 48,83 |
| 75 | 34,7715 | 14,4107 | 48,83 |
| 76 | 34,5818 | 14,2983 | 49,1 |
| 77 | 34,9651 | 14,3687 | 48,56 |
| 78 | 34,7385 | 14,3937 | 48,88 |
| 79 | 34,7672 | 14,3693 | 48,84 |
| 80 | 35,1587 | 14,3984 | 48,29 |
| 81 | 34,9909 | 14,4115 | 48,53 |
| 82 | ERROR | ERROR | ERROR |
| 83 | 34,7993 | 14,3352 | 48,79 |
| 84 | 35,4429 | 14,4936 | 47,91 |
| 85 | 34,7977 | 14,3718 | 48,8 |
| 86 | 34,7449 | 14,3819 | 48,87 |
| 87 | ERROR | ERROR | ERROR |

| | | | |
|---|---|---|---|
| 88 | 34,9973 | 14,3167 | 48,52 |
| 89 | 34,7917 | 14,3974 | 48,8 |
| 90 | 34,9928 | 14,4205 | 48,52 |
| 91 | 35,296 | 14,4894 | 48,11 |
| 92 | 35,0106 | 14,3773 | 48,5 |
| 93 | 35,8259 | 14,5436 | 47,39 |
| 94 | 35,5697 | 14,4851 | 47,74 |
| 95 | 34,9471 | 14,3537 | 48,59 |
| 96 | 34,7921 | 14,3686 | 48,8 |
| 97 | 34,9228 | 14,3788 | 48,62 |

Cuadro C.58: Ordis usando 25 nodos con tamaño de bloque 8

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 27,4062 | 14,0537 | 61,96 |
| 1 | 32,5641 | 14,8793 | 52,14 |
| 2 | 33,2868 | 15,4124 | 51,01 |
| 3 | 29,8929 | 14,22 | 56,8 |
| 4 | 30,8992 | 14,5252 | 54,95 |
| 5 | 45,6161 | 17,696 | 37,22 |
| 6 | ERROR | ERROR | ERROR |
| 7 | ERROR | ERROR | ERROR |
| 8 | ERROR | ERROR | ERROR |
| 9 | 27,2295 | 14,0266 | 62,36 |
| 10 | 27,1348 | 14,0317 | 62,57 |
| 11 | 27,3774 | 14,0488 | 62,02 |
| 12 | 27,1632 | 13,9582 | 62,51 |
| 13 | 27,1638 | 14,0044 | 62,51 |
| 14 | 36,8589 | 15,7367 | 46,07 |
| 15 | 27,2237 | 14,0293 | 62,37 |
| 16 | ERROR | ERROR | ERROR |
| 17 | ERROR | ERROR | ERROR |
| 18 | ERROR | ERROR | ERROR |
| 19 | ERROR | ERROR | ERROR |
| 20 | ERROR | ERROR | ERROR |
| 21 | 27,0733 | 14,011 | 62,72 |
| 22 | 27,2669 | 14,0422 | 62,27 |
| 23 | 27,2235 | 14,0266 | 62,37 |
| 24 | 27,1268 | 13,9553 | 62,59 |
| 25 | 27,1258 | 13,9211 | 62,6 |
| 26 | 27,6093 | 14,0687 | 61,5 |

| | | | |
|---|---|---|---|
| 27 | 27,5581 | 14,0144 | 61,61 |
| 28 | 27,3767 | 14,0226 | 62,02 |
| 29 | 27,2121 | 14,0536 | 62,4 |
| 30 | 27,2293 | 14,0699 | 62,36 |
| 31 | 26,8246 | 13,9276 | 63,3 |
| 32 | 27,095 | 13,9882 | 62,67 |
| 33 | 27,1424 | 14,1175 | 62,56 |
| 34 | 27,1213 | 13,9505 | 62,61 |
| 35 | 27,4024 | 14,05 | 61,96 |
| 36 | 27,5954 | 14,0531 | 61,53 |
| 37 | ERROR | ERROR | ERROR |
| 38 | 27,1437 | 13,9675 | 62,55 |
| 39 | 27,0212 | 14,0277 | 62,84 |
| 40 | 27,2296 | 14,0859 | 62,36 |
| 41 | 27,3814 | 14,0559 | 62,01 |
| 42 | 27,371 | 14,0444 | 62,03 |
| 43 | 27,0547 | 13,9427 | 62,76 |
| 44 | 27,1998 | 14,0238 | 62,43 |
| 45 | 27,1573 | 13,991 | 62,52 |
| 46 | 27,3934 | 13,9584 | 61,98 |
| 47 | 27,2222 | 14,0373 | 62,37 |
| 48 | 27,0479 | 13,9835 | 62,78 |
| 49 | 27,1902 | 13,9932 | 62,45 |
| 50 | 26,9808 | 14,0231 | 62,93 |
| 51 | 27,1189 | 13,9648 | 62,61 |
| 52 | 27,12 | 14,0148 | 62,61 |
| 53 | 27,1141 | 13,959 | 62,62 |
| 54 | 27,165 | 14,0099 | 62,51 |
| 55 | 26,9696 | 13,9938 | 62,96 |
| 56 | 27,1712 | 14,0462 | 62,49 |
| 57 | 27,5521 | 14,1287 | 61,63 |
| 58 | 27,4514 | 14,01 | 61,85 |
| 59 | 27,059 | 13,9938 | 62,75 |
| 60 | 27,1913 | 14,0195 | 62,44 |
| 61 | 27,1227 | 13,9653 | 62,6 |
| 62 | 27,2676 | 14,0701 | 62,27 |
| 63 | 27,1223 | 13,9857 | 62,6 |
| 64 | 26,9075 | 13,9564 | 63,1 |
| 65 | 27,169 | 13,9881 | 62,5 |
| 66 | 27,6103 | 14,1615 | 61,5 |
| 67 | 27,1425 | 13,9692 | 62,56 |

| | | | |
|---|---|---|---|
| 68 | 27,2109 | 13,9599 | 62,4 |
| 69 | 27,184 | 14,0145 | 62,46 |
| 70 | 27,4117 | 14,0663 | 61,94 |
| 71 | 27,3656 | 14,0451 | 62,05 |
| 72 | 27,5988 | 14,0909 | 61,52 |
| 73 | 27,3244 | 14,0608 | 62,14 |
| 74 | 27,3237 | 13,9361 | 62,14 |
| 75 | 27,6802 | 14,1089 | 61,34 |
| 76 | 26,8901 | 13,9037 | 63,14 |
| 77 | 27,1823 | 13,9621 | 62,47 |
| 78 | 27,1828 | 14,0622 | 62,46 |
| 79 | 27,2282 | 14,1205 | 62,36 |
| 80 | 27,1655 | 14,0131 | 62,5 |
| 81 | 27,195 | 13,9938 | 62,44 |
| 82 | 27,3786 | 14,0909 | 62,02 |
| 83 | 27,0368 | 14,0258 | 62,8 |
| 84 | 27,4264 | 14,0234 | 61,91 |
| 85 | 27,1366 | 14,0644 | 62,57 |
| 86 | 27,1467 | 14,0152 | 62,55 |
| 87 | 27,6157 | 14,1211 | 61,49 |
| 88 | 27,1699 | 14,0103 | 62,49 |
| 89 | 27,3025 | 13,9599 | 62,19 |
| 90 | 27,2407 | 14,0266 | 62,33 |
| 91 | 27,4333 | 14,0669 | 61,89 |
| 92 | 27,0489 | 13,9467 | 62,77 |
| 93 | 27,3171 | 14,0803 | 62,16 |
| 94 | 27,4313 | 14,101 | 61,9 |
| 95 | 26,9142 | 13,8786 | 63,09 |
| 96 | 27,1181 | 14,0012 | 62,61 |
| 97 | 27,1336 | 13,9893 | 62,58 |
| 98 | 27,1609 | 13,9981 | 62,51 |

Cuadro C.59: Ordis usando 25 nodos con tamaño de bloque 16

| Iteration | Total Time | Com Time | Flops Total |
|---|---|---|---|
| 0 | 36,1336 | 15,4587 | 46,99 |
| 1 | 38,1113 | 15,8013 | 44,55 |
| 2 | 36,8536 | 15,5406 | 46,07 |
| 3 | 37,3468 | 15,7108 | 45,46 |
| 4 | 37,2893 | 15,6354 | 45,53 |
| 5 | 38,2988 | 15,7627 | 44,33 |

| | | | |
|---|---|---|---|
| 6 | 38,769 | 15,9181 | 43,8 |
| 7 | 38,4688 | 15,8261 | 44,14 |
| 8 | 36,1432 | 15,4989 | 46,98 |
| 9 | 40,533 | 16,1913 | 41,89 |
| 10 | 36,6405 | 15,5444 | 46,34 |
| 11 | 36,5941 | 15,5256 | 46,4 |
| 12 | 36,7366 | 15,5332 | 46,22 |
| 13 | 36,8874 | 15,5545 | 46,03 |
| 14 | 37,3796 | 15,6957 | 45,42 |
| 15 | 37,6148 | 15,7841 | 45,14 |
| 16 | 37,3275 | 15,6706 | 45,49 |
| 17 | 38,1631 | 15,8147 | 44,49 |
| 18 | 37,01 | 15,6379 | 45,88 |
| 19 | 36,3773 | 15,5275 | 46,68 |
| 20 | 36,807 | 15,5886 | 46,13 |
| 21 | 37,673 | 15,7671 | 45,07 |
| 22 | 36,7621 | 15,5709 | 46,19 |
| 23 | 37,1508 | 15,6182 | 45,7 |
| 24 | 37,2556 | 15,6938 | 45,58 |
| 25 | 38,5375 | 15,9142 | 44,06 |
| 26 | 36,5836 | 15,5585 | 46,41 |
| 27 | 38,1146 | 15,8004 | 44,55 |
| 28 | 37,1236 | 15,6575 | 45,74 |
| 29 | 37,1226 | 15,6817 | 45,74 |
| 30 | 37,1866 | 15,6746 | 45,66 |
| 31 | 36,5791 | 15,5305 | 46,42 |
| 32 | 37,1019 | 15,5848 | 45,76 |
| 33 | 37,2691 | 15,7077 | 45,56 |
| 34 | 38,365 | 15,8285 | 44,26 |
| 35 | ERROR | ERROR | ERROR |
| 36 | 38,1635 | 15,7679 | 44,49 |
| 37 | 36,8147 | 15,5914 | 46,12 |
| 38 | ERROR | ERROR | ERROR |
| 39 | ERROR | ERROR | ERROR |
| 40 | 37,3068 | 15,6858 | 45,51 |
| 41 | 37,0352 | 15,6403 | 45,85 |
| 42 | 36,8908 | 15,5658 | 46,03 |
| 43 | 37,1999 | 15,6167 | 45,64 |
| 44 | 36,6932 | 15,586 | 46,27 |
| 45 | 37,5931 | 15,7482 | 45,17 |
| 46 | 36,9217 | 15,5907 | 45,99 |

| | | | |
|---|---|---|---|
| 47 | 37,7127 | 15,6884 | 45,02 |
| 48 | 37,9309 | 15,7873 | 44,76 |
| 49 | 37,3775 | 15,7225 | 45,43 |
| 50 | 37,9921 | 15,7825 | 44,69 |
| 51 | 37,0946 | 15,6858 | 45,77 |
| 52 | 37,1574 | 15,6614 | 45,7 |
| 53 | ERROR | ERROR | ERROR |
| 54 | 36,4442 | 15,5083 | 46,59 |
| 55 | 36,457 | 15,5162 | 46,57 |
| 56 | 37,2765 | 15,6685 | 45,55 |
| 57 | 37,9716 | 15,8199 | 44,72 |
| 58 | 37,3946 | 15,6967 | 45,41 |
| 59 | 37,1261 | 15,647 | 45,73 |
| 60 | 38,8195 | 15,9345 | 43,74 |
| 61 | 37,2674 | 15,6653 | 45,56 |
| 62 | 36,7822 | 15,6057 | 46,16 |
| 63 | 37,4301 | 15,7422 | 45,36 |
| 64 | 37,3435 | 15,7 | 45,47 |
| 65 | 36,6501 | 15,5447 | 46,33 |
| 66 | 38,0582 | 15,8362 | 44,61 |
| 67 | 37,2642 | 15,6869 | 45,57 |
| 68 | 37,6033 | 15,7094 | 45,15 |
| 69 | ERROR | ERROR | ERROR |
| 70 | 38,4447 | 15,8285 | 44,17 |
| 71 | 35,8568 | 15,4033 | 47,35 |
| 72 | 38,3974 | 15,8897 | 44,22 |
| 73 | 37,4876 | 15,7141 | 45,29 |
| 74 | 36,5061 | 15,4881 | 46,51 |
| 75 | 37,5008 | 15,6616 | 45,28 |
| 76 | ERROR | ERROR | ERROR |

Cuadro C.60: Ordis usando 25 nodos con tamaño de bloque 32

# Glosario

**CPU** CPU o unidad central de procesamiento, es el encargado interpretar funciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema. 17, 26, 27, 188

**CUDA** CUDA o Compute Unified Device Architecture (Arquitectura Unificada de Dispositivos de Cómputo).Es conjunto de herramientas de desarrollo creadas por nVidia que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de NVIDIA. 27, 28

**DIE** Es un sinónimo de chip, y en el contexto de los circuitos integrados es un pequeño bloque de material semiconductor en el que se crea un circuito funcional. 24

**ECC** ECC o Error-correcting code memory es un tipo de memoria que puede detectar y corregir los errores mas comunes de corrupción de datos internos dentro de la memoria, son ampliamente utilizados en servidores y aplicaciones científicas u financieras donde los errores en memoria no pueden ser tolerados. 34

**FLOPS** Los FLOPS o las operaciones de coma flotante por segundo son una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de coma flotante. 25

**GIS** GIS o sistema de información geográfica es un conjunto de herramientas que integra y relaciona diversos componentes que permiten la organización, almacenamiento, manipulación, análisis y modelización de grandes cantidades de datos procedentes del mundo real que están vinculados a una referencia espacial, facilitando la incorporación de aspectos sociales-culturales, económicos y ambientales que conducen a la toma de decisiones de una manera más eficaz. 25

**GNU GRUB** GNU GRUB o GNU GRand Unified Bootloader es un gestor de arranque múltiple, desarrollado por el proyecto GNU que nos permite elegir qué Sistema Operativo arrancar de los instalados.Se usa principalmente en sistemas operativos GNU/Linux. 68

**GPU** GPU o unidad de procesamiento gráfico, es un procesador orientado al calculo de gráficos u operaciones en coma flotante, su diseño difiere de la CPU en el que esta diseñado para un alto nivel de paralelismo. 17, 26–28

**HPC** HPC o también conocido como computación de altas prestaciones que se apoya en clústers , supercomputadores o computación paralela para resolver problemas computacionalmente complejos. 25

**InfiniBand** InfiniBand es un bus de comunicaciones serie de alta velocidad, baja latencia y de baja sobrecarga de CPU, diseñado para el paso de grandes cantidades de datos, la velocidad puede ir desde 8 Gbps hasta 96Gbps. 27, 28, 35

**LiLo** LiLo o Linux Loader, es un gestor de arranque que permite elegir, entre sistemas operativos Linux y otras plataformas, con cual se ha de trabajar al momento de iniciar un equipo con más de un sistema operativo disponible.. 68

**Mecanica Quantica** La mecánica cuántica es una disciplina de la física encargada de brindar una descripción fundamental de la naturaleza a escalas espaciales pequeñas. Esta disciplina física es la proporciona fundamento de la fenomenología del átomo, de su núcleo y de las partículas elementales. 25

**mime** MIME o Multipurpose Internet Mail Extensions son una serie de especificaciones para el paso de toda clase de archivos a través de internet de forma transparente al usuario . 55

**PXE** PXE o Preboot eXecution Environment, es un sistema orientado a la instalación de sistemas operativos a través de la red, de manera independiente de los dispositivos de almacenamiento de datos disponibles o sistemas operativos . 67, 68

**RAID1** RAID o también conocido como grupo/matriz redundante de discos independientes, es un sistema para las unidades de almacenamiento de

datos entre los que se distribuyen o replican datos. En concreto con RAID1 los discos están espejados por lo que en caso de fallo de uno, no produce perdidas de datos. 36

**Riesgo WAR** Write after Read (WAR) o anti-dependencia: Leer un operando y escribir en él en poco tiempo. Si la escritura finaliza antes que la lectura, la instrucción de lectura utilizará el nuevo valor y no el antiguo. 22

**Riesgo WAW** Write after Write (WAW) o dependencia de salida: Dos instrucciones que escriben en un mismo operando. La primera en ser emitida puede que finalice en segundo lugar, de modo que el operando final no tenga el valor adecuado. 22

**rootkit** El rootkit permite un acceso de privilegio continuo a un equipo, pero que mantiene su presencia activamente oculta al control de los administradores con tal de corromper el funcionamiento normal del sistema operativo o de otras aplicaciones. 42

**RRDtool** RRDtool o *round-robin database tool* es una herramienta para el almacenamiento de la información con una dimensión temporal, normalmente datos sobre temperaturas, uso de CPU, uso de red. Los datos se almacenan en una base de datos circular basada en búfer, por lo que la informacion del sistema permanece constante a lo largo del tiempo. 43–45

**sha256** Es una función hash criptográfica diseñada por la Agencia de Seguridad Nacional (NSA) y publicada en 2001 por el Instituto Nacional de Estándares y Tecnología (NIST) como un Estándar Federal de Procesamiento de la Información.
Una función hash es un algoritmo que transforma ("digiere") un conjunto arbitrario de elementos de datos, como puede ser un fichero de texto, en un único valor de longitud fija (el "hash"). El valor hash calculado puede ser utilizado para la verificación de la integridad de copias de un dato original sin la necesidad de proveer el dato original. 65

**Sun Grid Engine** SGE o Sun Grid Engine es un software de código abierto desarrollado por Sun Microsystems, cuya función principal es la gestión de un sistema manejador de recursos computacionales o procesos distribuidos en ambientes heterogéneos, de modo que se utilicen los dichos recursos de la manera más eficiente posible. 53, 61, 64, 69, 70, 73

**SWAP** SWAP o también llamado espacio de intercambio es una zona de disco disco encargada de almacenar imágenes o procesos que no han de mantenerse en la memoria principal. 68

**TFTP** TFTP o Trivial file transfer Protocol, es una versión simplificada de FTP con capacidades reducidas y que utiliza como medio de trasporte el protocolo UDP. Se utiliza para pasar pequeñas cantidades de datos entre cliente y servidor. 67

**Wake On Lan** WOL o Wake On Lan es un estándar de red que permite el arranque de maquinas de forma remota a traves de Ethernet. 71

**XDR** XDR es un estándar para la serialización de los datos, por ejemplo en una red de computadores. Permite la comunicación entre maquinas con diferentes arquitecturas y Sistemas Operativos. 43, 44