Universitat de Lleida

Clusterización de aplicaciones paralelas para su planificación en entornos de cómputo multi-cluster

Hector Blanco de Frutos

Escola Politècnica Superior
Departament dÍnformàtica i Enginyeria Industrial

Clusterización de aplicaciones paralelas para su planificación en entornos de cómputo multi-cluster

Memòria presentada per obtenir el grau de
Doctor per la Universitat de Lleida
per

Hector Blanco de Frutos

Dirigida per

Fernando Guirado Fernández **Josep Lluís Lérida Monsó**
Universitat de Lleida

Programa de doctorat en Enginyeria

Lleida, setembre de 2012

Prólogo

A lo largo de la historia de la informática, siempre ha existido la necesidad de ejecutar aplicaciones con requisitos de computación cada vez más elevados. Ésta necesidad ha provocado que se haya potenciado enormemente la investigación y el desarrollo de nuevos métodos y estrategias para tratar y ejecutar aplicaciones con altos requerimientos de procesamiento. A finales de la década de 1980 y principios de la de 1990 apareció una nueva arquitectura de cómputo, llamada Cluster, que permitía aprovechar los recursos de grupos de ordenadores de sobremesa unidos mediante una conexión de red, eliminando la necesidad de utilizar super-computadores de elevado coste para poder acceder a elevadas capacidades de procesamiento.

En la actualidad existen una gran disponibilidad de recursos computacionales, repartidos en diferentes departamentos de una misma organización, empresa, o universidad. Esta arquitectura de cómputo distribuido, llamada Multi-Cluster, ofrece la oportunidad a los investigadores nuevas oportunidades para estudiar y desarrollar nuevas técnicas de planificación para poder aprovechar estos recursos y poder ejecutar aplicaciones con requisitos cada vez más elevados. La tarea de planificación en un entorno Multi-Cluster es compleja, con numerosas líneas de investigación abiertas en la actualidad. En primer lugar, la cantidad de recursos que han de ser gestionados puede ser muy grande, y además estos recursos pueden ser heterogéneos. En segundo lugar, los diferentes clusters de un entorno Multi-Cluster están conectados mediante un enlace de red, y si las aplicaciones paralelas no son asignadas adecuadamente, estos enlaces pueden saturarse y degradar significativamente el rendimiento de estas aplicaciones. En la literatura existen múltiples estrategias para resolver el proceso de planificación de aplicaciones paralelas. Las estrategias más comunes en

la actualidad evalúan las aplicaciones paralelas de forma aislada, sin tener en cuenta los requisitos de las otras aplicaciones que están presentes en la cola de espera del sistema. Algunos estudios han concluido que mediante la evaluación de grupos de aplicaciones se pueden tomar decisiones de planificación que pueden mejorar el rendimiento del conjunto de aplicaciones, y se puede mejorar el aprovechamiento de los recursos del sistema.

En el presente trabajo de tesis abordamos el problema de la planificación on-line de múltiples aplicaciones paralelas en entornos Multi-Cluster heterogéneos y con co-asignación. Para ello, proponemos nuevas técnicas que tratan tanto la agrupación de las aplicaciones como su asignación, considerando las características de los recursos del sistema, así como los requisitos del grupo de aplicaciones en cuanto al cómputo y la comunicación.

En el presente estudio evaluamos la complejidad y el rendimiento de las técnicas propuestas, comparándolas con otras técnicas comúnmente utilizadas en la literatura, utilizando para ello trazas de entornos reales. Finalmente, estudiamos su aplicabilidad en entornos reales y presentamos posibles líneas de trabajo futuro a considerar.

Agradecimientos

Desearía agradecer a todas las personas que me han apoyado en el transcurso de ésta tesis doctoral. En primer lugar, deseo expresar mi gratitud a mis tutores, el Dr. Fernando Guirado Fernández y al Dr. Josep Lluís Lérida Monsó, por su apoyo continuado y su inestimable labor como directores. Pacientemente han supervisado cada aspecto de mi estudio, guiándome en la dirección correcta. Sin su ayuda no habría sido capaz de finalizar con éxito la tesis.

Quiero agradecer especialmente a todos los miembros del Grupo de Computación Distribuida (GCD) de la Universitat de Lleida (UdL). Ha sido una gran experiencia el haber podido trabajar estrechamente con éstas grandes personas: Concepció Roig, Francesc Giné, Francesc Solsona, Fernando Cores, Josep M. Solà, Valentí Pardo, Albert Saiz y Xavier Faus.

En el transcurso de mi investigación he compartido grandes momentos con mis compañeros de la universidad: Damià Castellà, Josep Rius, Miquel Orobitg, Ignasi Barri, Ivan Teixidó, Anabel Usié, Jordi Vilaplana, Eloi Gabaldón y Alberto Montañola. Muchas gracias a todos ellos por los excelentes momentos que hemos compartido.

Desearía agradecer también a todos los miembros del Departamento de Industrias de la Universidad Técnica Federico Santa María, especialmente a los miembros del campus de Vitacura, en Santiago de Chile. En el año 2010, durante dos meses, tuve la gran oportunidad de visitar éste departamento, y poder trabajar estrechamente con grandes personas y sentirme valorado y arropado. En particular me gustaría expresar un especial agradecimiento al Dr. Víctor M. Albornoz, quien me acogió como a un miembro más de su familia, y que participó muy activamente en mi investigación.

Por último, y no por ello menos importante, deseo dedicar ésta tesis a todos mis amigos, y principalmente, a mi familia. Especialmente a mis padres Andrés y Dorita, a mi hermana Montse, y a mi abuela Juana. A los demás miembros de mi familia, los que están, y los que desgraciadamente nos han abandonado. A todos ellos, familia y amigos, aunque no los nombre personalmente, les agradezco el apoyo, paciencia y comprensión que han demostrado durante estos largos años.

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Capítulo 1

Introducción

A lo largo de la historia de la informática, siempre ha existido la necesidad de ejecutar aplicaciones con requisitos de computación cada vez más elevados. Ésta necesidad ha provocado que se haya potenciado enormemente la investigación y el desarrollo de nuevos métodos y estrategias para poder tratar y ejecutar aplicaciones con altos requerimientos de procesamiento.

Una de las primeras propuestas para poder ejecutar aplicaciones con elevados requisitos de cómputo fue la utilización de super-computadores. Estos entornos consistían en un único computador compuesto por varios procesadores y un espacio de memoria compartido por todos ellos. Este tipo de entornos, a pesar de proporcionar una elevada capacidad de cómputo, tenían una escalabilidad limitada, ya que no podían instalarse más procesadores a un mismo super-computador de forma indefinida.

Con la reducción del precio y el aumento de las prestaciones de los computadores de trabajo, o *workstations* a finales de los años 1980 y principio de los años 1990 surgió el interés por aprovechar estos recursos de cómputo. Con esto, apareció una nueva arquitectura, llamada *Cluster*, que permitía conectar mediante un enlace de red diferentes nodos, y aprovechar la capacidad de cómputo del conjunto. Esta nueva arquitectura permitió que aplicaciones cada vez más grandes, y con mayores requisitos de cómputo pudieran ser ejecutadas, al tener esta arquitectura una capacidad de escalabilidad mucho más elevada.

En la actualidad existen una gran disponibilidad de recursos computacionales, repartidos en diferentes departamentos de una misma organización, em-

presa, o universidad. Esta disponibilidad da la oportunidad a los investigadores para estudiar y desarrollar nuevas técnicas de planificación para poder aprovechar estos recursos y poder ejecutar aplicaciones con requisitos cada vez más elevados. Diferentes arquitecturas distribuidas se han diseñado para poder aprovechar estos recursos de procesamiento. En este capítulo se introduce al lector el concepto de computación distribuida, mostrando los entornos y arquitecturas distribuidas más populares. En la sección 1.1, se introducen los conceptos de computación paralela y computación distribuida, dos conceptos que muchas veces son confundidos. A continuación, se enumeran y detallan las arquitecturas de computación distribuida más comunes. De entre los diferentes sistemas distribuidos, ésta tesis se basa en los entornos Multi-Cluster.

Teniendo presente esta elección, en la sección 1.2 se explican detalladamente las características principales de éste tipo de entorno, las ventajas que aporta, y los principales retos que implica. Una vez que han sido introducido los conceptos principales, y se ha definido el contexto en el que se enmarca esta tesis, se presentan las motivaciones y los objetivos principales para la realización de la misma. Por último, se presenta la estructura general de la memoria.

1.1. Computación distribuida

Los términos "computación paralela" y "computación distribuida" se solapan, y no hay una distinción clara entre ellos, ocurriendo a menudo que un mismo sistema pueda ser clasificado en ambas categorías. Un sistema distribuido consiste en grupos de ordenadores conectados a través de una red, y la computación distribuida se refiere a la forma en que una aplicación es ejecutada en más de un ordenador, de forma simultánea. Se pueden utilizar los siguientes criterios para distinguir un sistema paralelo de uno distribuido:

- En un sistema paralelo, todos los procesadores tienen acceso a una memoria compartida. Ésta memoria compartida puede ser utilizada para intercambiar información entre los procesadores.
- En un sistema distribuido, cada procesador tiene su propia memoria privada. La información es intercambiada entre procesadores mediante el uso de mensajes.

Tomando en consideración estas definiciones, los sistemas que son analizados en este trabajo de tesis son los pertenecientes al grupo de los sistemas distribuidos.

Se entiende por sistemas de cómputo distribuido aquellos en que sus componentes, situados en computadores en red, se comunican y coordinan únicamente a través de paso de mensajes [DKC05]. Estas tecnologías de computación han emergido recientemente como nuevos paradigmas para resolver problemas computacionales complejos. En las dos últimas décadas, diferentes estudios científicos han reportado numerosos avances y nuevas técnicas para aprovechar los recursos de estos sistemas. Otro de los aspectos que ha contribuido al rápido desarrollo de nuevas aplicaciones para aprovechar estos entornos distribuidos en campos como el de la ciencia y la industria, ha sido el desarrollo continuado de redes de alta velocidad.

Dentro de la categoría de sistemas distribuidos se puede encontrar un amplio abanico de sistemas diferentes. A continuación se enumeran y se presentan algunos de los entornos de cómputo distribuido más ampliamente utilizados, y que están más ligados con el presente trabajo.

- **Cluster:** Se define un cluster como una colección de recursos de computación formado normalmente por ordenadores comunes de escritorio, conectados entre sí y que se comportan como una única computadora, alcanzando en ocasiones, y según su configuración, rendimientos comparables al de los super-computadores. El interés por el estudio del paradigma cluster cosechó un gran interés a principios de los años 90, debido a su posibilidad de poder alcanzar altas prestaciones de cómputo mediante el uso de recursos de bajo coste, y a los avances en las redes de comunicaciones [ELvD+96, LLM88, RBMS97, SSB+95].

  En la actualidad, la tecnología multi-core ofrece la posibilidad de multiplicar las unidades de procesamiento de un cluster sin suponer un coste adicional elevado. La reducción de la relación coste-rendimiento, y el continuo desarrollo de estrategias para el uso eficiente de los recursos y el aumento de las prestaciones, ha hecho que el uso de sistemas cluster haya crecido enormemente tanto en el sector industrial como en el de investigación. Buena prueba de este interés es que hoy, cerca del 82 % de los super-computadores de la lista TOP500 [top] está formada por entornos cluster.

  En la Figura 1.1 se muestra un diagrama que representa la distribución básica de un sistema cluster. En la figura se pueden observar los distintos computadores, encargados de llevar a cabo las tareas de cómputo, conectados a través de una red de comunicaciones dedicada.

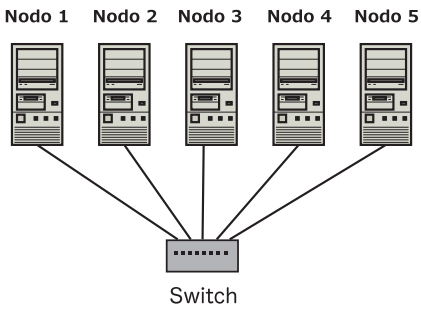


Figura 1.1: Estructura de un entorno Cluster

- **Multi-Cluster:** Con el paso de los años, el uso de entornos cluster se

ha extendido ampliamente, de forma que es frecuente la instalación de varios de estos entornos en una misma organización. La posibilidad de unir los recursos de una misma organización, para obtener una mayor capacidad de cómputo, ha despertado un gran interés en los últimos años [ccg, iee]. Esto ha propiciado la aparición de nuevos modelos de cómputo distribuido, como Multi-Cluster [iee, BB01], que incorporan técnicas de planificación de trabajos y gestión de recursos para aprovechar al máximo la capacidad de los recursos de cómputo distribuidos en varios clusters.

Un Multi-Cluster se puede definir como la unión de un conjunto de clusters mediante una red de comunicaciones dedicada dentro de una misma organización o institución. La característica arquitectónica principal que diferencia un Multi-Cluster respecto a otros entornos de cómputo distribuido, como Grid o Peer-to-Peer, es que las redes de comunicaciones entre clusters se conectan mediante enlaces dedicados. Esto tiene varias implicaciones de gran importancia. La primera es que un Multi-Cluster tiene un ancho de banda fiable y predecible entre los recursos de diferentes clusters [JAA07], a diferencia de los sistemas distribuidos que se conectan a través de internet. La segunda, es que la disponibilidad de los recursos de un sistema Multi-Cluster es más predecible y controlable, mientras que en el resto de sistemas estos recursos son muy dinámicos y mutables, de forma que no se puede garantizar su disponibilidad en el tiempo.

En la Figura 1.2 se puede observar el diagrama básico de un entorno Multi-Cluster. En ella, un conjunto de clusters, cada uno con sus respectivos computadores, se conectan a un switch central, mediante enlaces de red dedicados.

- **Grid:** El paradigma Grid apareció hace unos años como alternativa a los caros supercomputadores dedicados al HPC (High Performance Computing), y que en los últimos años ha despertado un importante interés [ccg]. La computación Grid se define como "compartir recursos en forma coordinada y resolver problemas en organizaciones virtuales multi-institucionales de forma dinámica" [KF98]. En un sistema Grid se unen

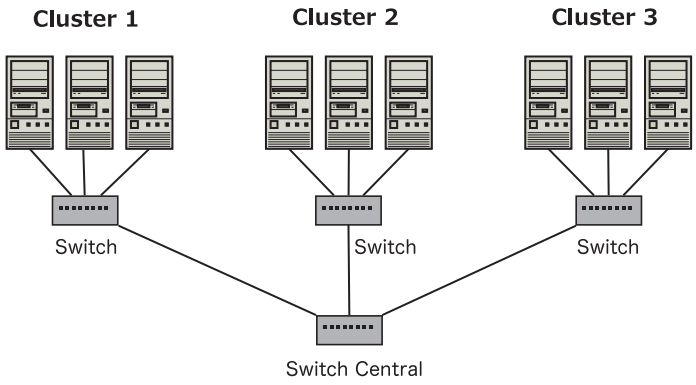


Figura 1.2: Estructura de un entorno Multi-cluster

de forma virtual un número elevado de computadores, clusters y/o supercomputadores, separados entre si geográficamente, y conectados habitualmente a través de Internet [Sto07]. Una de las ventajas de este sistema distribuido es que permite realizar una mayor cantidad de cómputo de la que sería posible en un único computador o en un único cluster dentro de una única institución. Sin embargo, la computación Grid tiene algunas desventajas, como son la elevada complejidad del software de gestión y administración (como el Globus Toolkit [Fos05]) y el elevado coste de gestión y mantenimiento.

En la Figura 1.3 se presenta el diagrama que representa a un sistema grid. En esta figura se pueden observar los diferentes tipos de sistemas de cómputo (Cluster, PCs d escritorio, super-computadores ...etc.), separados en instituciones diferentes, y conectados a través de internet. Y por último, un computador, o conjunto de computadores, que con la ayuda de paquetes de software dedicados, gestiona el funcionamiento de esta red de elementos virtuales de cómputo.

- **Peer-to-Peer:** La computación peer-to-peer apareció como un nuevo paradigma a partir de la computación cliente-servidor tradicional, y en un principio tuvo un impacto importante en el diseño de sistemas de compartición de ficheros. En este paradigma de computación, y a diferencia de los otros sistemas de computación, los usuarios tienen respon-

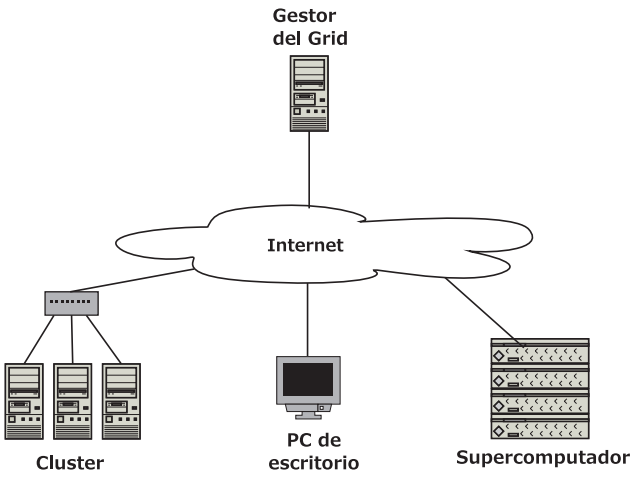


Figura 1.3: Estructura de un entorno grid

sabilidades equivalentes, así que pueden asumir al mismo tiempo el rol de usuario y de servidor. Los peers comparten parte de sus capacidades de procesamiento, de forma que el resto de usuarios puedan beneficiarse de ellas, sin la necesidad de servidores centralizados que coordinen esta compartición de recursos. Esto permite que se puedan construir sistemas distribuidos peer-to-peer a través de internet con un reducido coste. Un ejemplo de plataforma de cómputo peer-to-peer es *CodiP2P* [CIR+09], la cual permite aprovechar los recursos ociosos de ordenadores de escritorio a través de internet para ejecutar aplicaciones paralelas.

En la Figura 1.4 se muestra una representación de un sistema distribuido peer-to-peer. Como se puede observar, los diferentes computadores, que usualmente suele consistir en PCs de escritorio, no están conectados a través de una red dedicada, sino que se conectan unos a otros, normalmente a través de internet, que es menos predecible y controlable.

La posibilidad de disponer de un mayor número de recursos para la ejecución de aplicaciones con mayores requerimientos, las ventajas que se pueden obtener del uso compartido de los mismos, y la posibilidad de utilizar clusters ya existentes en una organización, hace de la planificación en entornos Multi-Cluster, un área de estudio emergente. Por estos motivos, y con el propósito

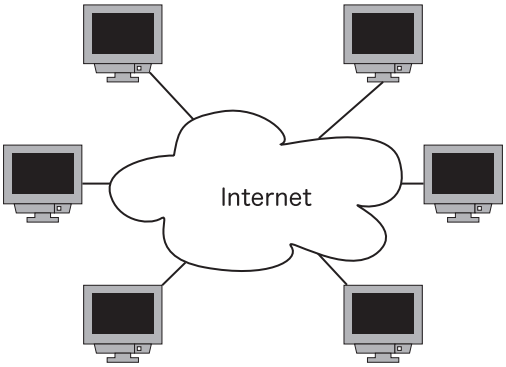


Figura 1.4: Estructura de un entorno distribuido Peer-to-Peer

de estudiar nuevas técnicas de planificación que nos permitan obtener mejor rendimiento de las aplicaciones paralelas en entornos Multi-Cluster, el presente trabajo se centra en este entorno de cómputo distribuido.

En la Sección 1.2 se presenta de forma detallada el paradigma Multi-Cluster, con sus características principales, así como los retos más importantes que se presentan al trabajar con este tipo de entorno distribuido.

1.2. Paradigma Multi-Cluster

Un entorno Multi-Cluster se compone de varios clusters de una misma organización, conectados a una red, mediante enlaces dedicados. Los recursos que componen los distintos clusters pueden tener características diferentes, y por lo tanto, afectar al tiempo de ejecución de las aplicaciones.

Este tipo de entorno, correctamente gestionados, ofrece la posibilidad de aumentar la capacidad de cómputo, aprovechando sistemas ya existentes en las organizaciones, y por lo tanto, la ejecución de aplicaciones paralelas más complejas y con requerimientos más elevados. Sin embargo, la dificultad y la complejidad en la gestión de los recursos de cómputo y de comunicación que componen un entorno Multi-Cluster aumenta enormemente respecto a la de un entorno cluster. Este hecho ha provocado que en los últimos años haya crecido el interés por estudiar formas de mejorar la gestión y aprovechamiento de este tipo de entornos. [BB01, AD03, BE07, SQR10].

En la Sección 1.2.1 se analizan las distintas posibilidades de organización de la planificación en un entorno Multi-Cluster. En la Sección 1.2.2 se analizan las implicaciones del problema de la heterogeneidad de los recursos, y en la Sección 1.2.3 se presentan el concepto de co-asignación, y sus implicaciones.

1.2.1. Organización de la planificación de aplicaciones paralelas

La organización de la planificación en un entorno Multi-Cluster puede ser dividida en dos categorías, dependiendo de si se usa un planificador global para todo el entorno, o si cada uno de los clusters utiliza su propio planificador local [BE07]. En el caso de usar un planificador global, es posible que cada uno de los clusters mantenga además su propio planificador. Teniendo esto en cuenta, se presentan tres categorías de organización: *centralizada*, *descentralizada* y *jerárquica*.

- **Planificación centralizada**: En un esquema de planificación centralizada (Figura 1.5), un único planificador global se encarga de gestionar de forma centralizada todos los recursos y aplicaciones del sistema.

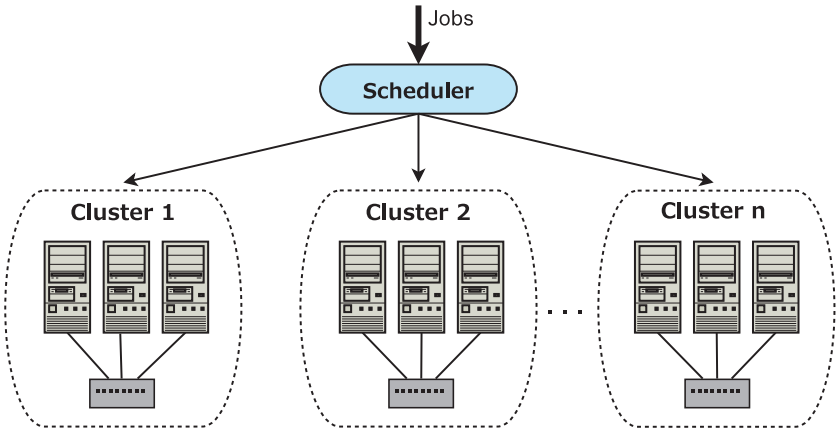


Figura 1.5: Planificación centralizada

Este planificador dispone de un conocimiento global de todo el sistema, lo que le permite realizar una gestión eficiente de los recursos y de las aplicaciones. Sin embargo, el hecho de que toda la tarea de gestión recaiga en un único planificador provoca que sea más fácil que este se sature cuanto más grande sea el entorno que ha de gestionar. Por lo tanto, su escalabilidad y tolerancia a fallos es menor.

Este esquema es el más adecuado cuando todos los recursos se encuentran confinados a un único dominio administrativo, donde la cantidad de los recursos, y la heterogeneidad de los mismos puede ser controlada.

- **Planificación descentralizada**: En un esquema de planificación descentralizada (Figura 1.6), no existe un planificador global en sí, sino que cada cluster dispone de su propio planificador local, que se encarga de gestionar los recursos y aplicaciones paralelas de su correspondiente dominio, interactuando con el resto de planificadores del entorno, intercambiando información y trabajos.

  Esta organización de planificación es escalable y soporta una mejor tolerancia a fallos. Además, evita la saturación que se produce en un esquema centralizado, al disponer cada cluster de su propio planificador. Sin embargo, la implementación de un esquema descentralizado es más compleja, ya que los planificadores de los diferentes clusters han de poder comunicarse entre si para intercambiar información sobre su estadio,

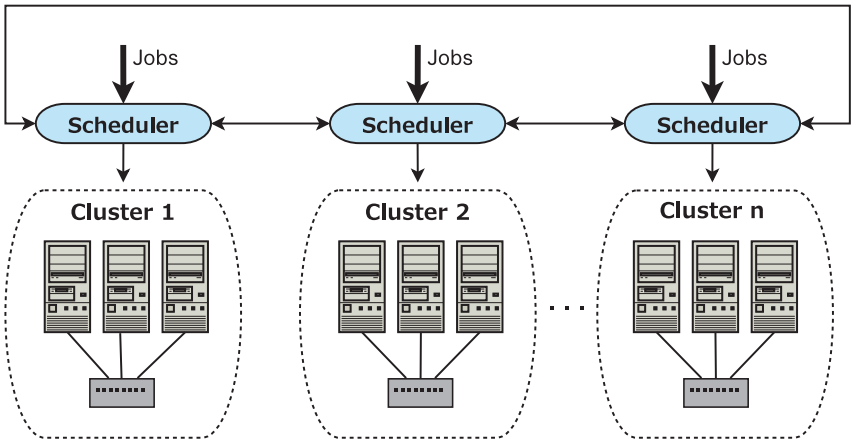


Figura 1.6: Planificación descentralizada

especialmente en el caso en que se han de compartir recursos de varios clusters para la ejecución de una aplicación paralela.

Este esquema es más adecuado cuando los diferentes clusters están repartidos geográficamente, conectados a través de una red donde las latencias pueden jugar un papel muy importante.

- **Planificación jerárquica**: Un esquema de planificación jerárquica (Figura 1.7) se presenta como una solución intermedia que permite resolver algunos inconvenientes del esquema centralizado, pero sin ser tan complejo como el esquema descentralizado. En una organización de planificación jerárquica, las tareas de gestión de los recursos y planificación de las aplicaciones paralelas se reparten en distintos niveles. Un planificador global se sitúa en el nivel más alto, y se encarga de gestionar los planificadores del nivel inmediatamente inferior. Estos otros planificadores, a su vez, gestionarán otros planificadores que puedan tener en un nivel inferior al suyo, y así sucesivamente hasta alcanzar a los recursos finales, en los niveles más bajos.

  Este esquema es una solución intermedia, y es especialmente adecuado en entornos donde los recursos están distribuidos a lo largo de varios entornos, donde el número de recursos y su heterogeneidad pueden ser elevados. Algunos ejemplos de sistemas Multi-Cluster jerárquicos son las

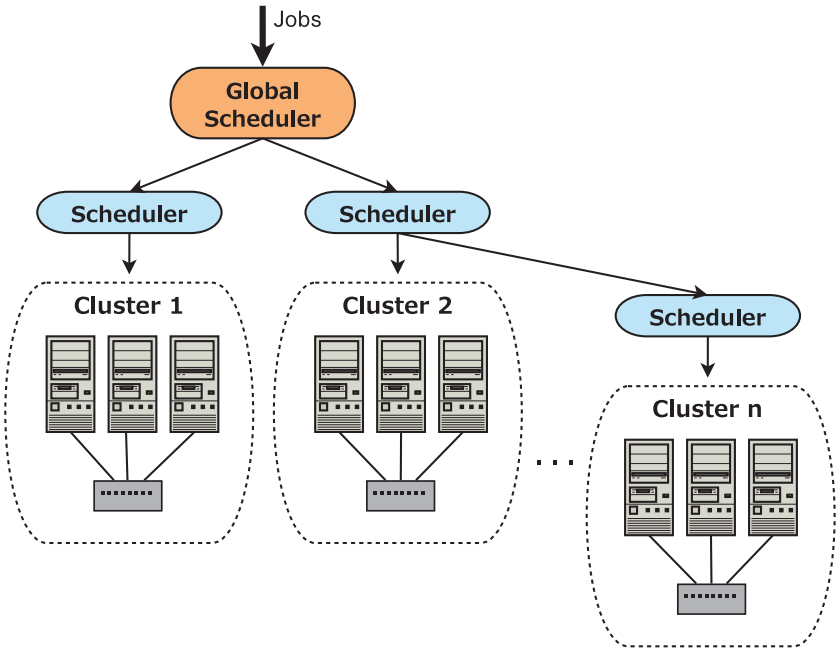


Figura 1.7: Planificación jerárquica

plataformas *CISME*[HGH+05] y *MetaLoRaS* [LSG+06].

En el presente trabajo estamos interesados en el aprovechamiento de los recursos de cómputo de un entorno académico o de investigación, compuesto por un conjunto de clusters distribuidos a lo largo de diferentes dominios administrativos, y unidos mediante enlaces de red de ámbito local. Cada uno de estos dominios dispone de su propio sistema de planificación, que se encarga de gestionarlo. Un esquema de planificación jerárquica es el más adecuado para la estructura de Multi-Cluster sugerida, y es el esquema que se asumirá en el resto del trabajo de tesis.

1.2.2. Heterogeneidad de los recursos en el entorno Multi-Cluster

Un gran reto que se encuentra al trabajar en un entorno Multi-Cluster es la heterogeneidad de los recursos. En un entorno de tipo cluster, normalmente los recursos que lo forman son homogéneos, es decir, todos tienen características iguales. Diversos estudios han sido desarrollados para resolver el problema de

la asignación de aplicaciones paralelas en sistemas homogéneos [TF99, CV01, SKSS02a].

Sin embargo, al trabajar en un entorno Multi-Cluster, los recursos de los diferentes clusters que lo componen pueden tener características diferentes. Imaginemos el caso de una institución que dispone de varios clusters, que ha ido adquiriendo y poniendo en marcha a lo largo del tiempo, con lo que los computadores de los clusters más antiguos probablemente tendrán una potencia inferior a los computadores de los clusters más modernos, y estos se enlazan para aprovechar los recursos del conjunto, constituyendo un Multi-Cluster.

En casos como este, dependiendo del cluster al que sean asignadas las aplicaciones paralelas, dispondrán de recursos más o menos potentes, y por lo tanto, su tiempo de ejecución podrá verse afectado. Por lo tanto, resulta de gran importancia tener en cuenta esta heterogeneidad en los recursos a la hora de determinar la asignación de las aplicaciones paralelas. Sin embargo, esto implica que la complejidad del problema de planificación se incrementa [JLPS05].

En la literatura se pueden encontrar diferentes estudios que han tratado con la asignación de aplicaciones en entornos heterogéneos. En [HYD+00] se analizan un conjunto de heurísticas para la planificación estática de aplicaciones, que re-ordenan las aplicaciones presentes en la cola del sistema, según diferentes criterios, y se proponen mejoras al funcionamiento de las mismas. En [YDPS00] se analizan y se proponen mejoras en heurísticas para la planificación dinámica de aplicaciones. En [BSB+01] se analizan diferentes heurísticas de la literatura, basados en la priorización de aplicaciones o en el balanceo de carga, algoritmos de búsqueda como *Tabú* y A^*, y algoritmos genéticos como *GSA* que basan la búsqueda de soluciones en la evolución de una población de agentes. Técnicas como *backfilling* [SP94], que avanzan aplicaciones de la cola del sistema también han sido utilizadas para la planificación en entornos, tanto homogéneos como heterogéneos [SKRS03, SF05, BCP+08].

En el Capítulo 2 se trata en detalle un modelo de aplicación que tiene en cuenta la heterogeneidad de los recursos, y como ésta afecta al tiempo de ejecución de las aplicaciones.

1.2.3. Co-asignación de aplicaciones paralelas

Otro gran reto que se encuentra al trabajar en un entorno muli-cluster es el de la co-asignación. Esta técnica permite la asignación de las tareas de una aplicación a lo largo de varios clusters. Gracias a esto, se pueden asignar en un sistema aplicaciones con un número mayor de tareas, que normalmente no hubiesen tenido oportunidad de ejecutarse al no estar disponible el número de recursos necesarios en un único cluster. Otra importante ventaja del uso de la co-asignación es la reducción de la fragmentación interna, al poderse aprovechar recursos repartidos a lo largo de diferentes clusters, lo que incrementa el rendimiento de las aplicaciones, ya que reduce los tiempos de espera en la cola del sistema [SCJG00, JLPS05]. La fragmentación interna se produce cuando las tareas de cada aplicación se asignan en un único cluster y dejan libres una cantidad de recursos que no pueden ser asignados a la siguiente aplicación de la cola del sistema. Sin embargo, si unimos los recursos libres repartidos entre los distintos clusters, se puede aumentar el número de aplicaciones asignadas, y mejorar la utilización de los recursos del entorno Multi-Cluster.

En la Figura 1.8 se muestra un ejemplo de asignación en un entorno Multi-Cluster. En este ejemplo, se asume un Multi-Cluster compuesto por 3 clusters, cada uno de ellos compuesto a su vez, por 3 nodos de cómputo.

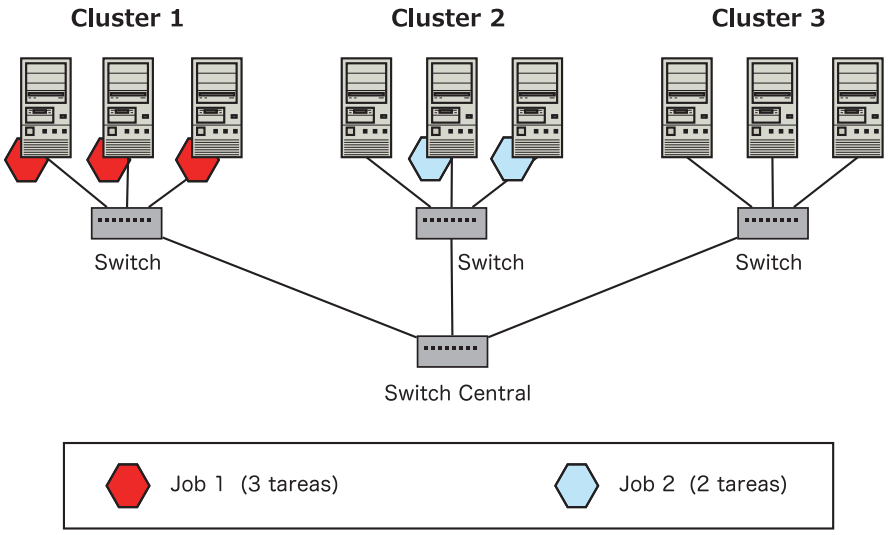


Figura 1.8: Ejemplo de asignación en un entorno Multi-Cluster

Se intentan asignar dos aplicaciones (Job 1 y Job 2), formadas por 3 y 2 tareas, respectivamente. Como se puede observar, cada una de las dos aplicaciones puede ser asignada a un cluster, y por lo tanto no es necesario realizar co-asignación.

En la Figura 1.9 se muestra un ejemplo de co-asignación en un entorno Multi-Cluster. En este segundo ejemplo, se representa el mismo entorno Multi-Cluster que se ha usado en el ejemplo anterior. En este caso, sin embargo, una de las aplicaciones (Job 1) está compuesta por 4 tareas, y por lo tanto, no puede ser asignada en ningún cluster en particular. En este caso, mediante co-asignación las tareas pueden ser asignadas repartiendo las mismas entre diferentes clusters. En el ejemplo, tres tareas de la aplicación 1 han sido asignadas en el cluster 1, y la cuarta ha sido asignada en el cluster 2, donde también se han asignado las dos tareas de la aplicación 2.

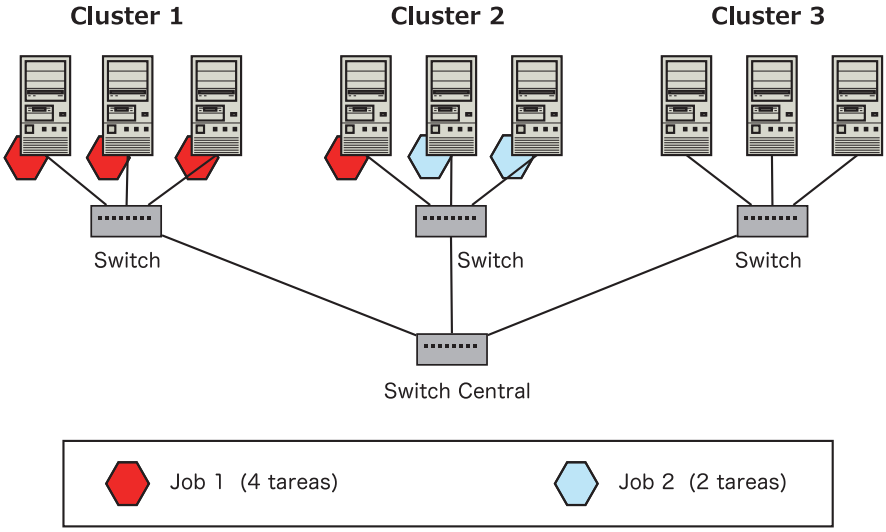


Figura 1.9: Ejemplo de co-asignación en un entorno Multi-Cluster

Hasta este momento se han mostrado las ventajas y posibilidades que ofrece el uso de la técnica de co-asignación. Sin embargo, su uso puede implicar un empeoramiento en el rendimiento general cuando varias aplicaciones co-asignadas tienen que competir por el ancho de banda de los canales de comunicación. En estas situaciones, si no se tiene precaución a la hora de asignar las tareas de las diferentes aplicaciones, se podría llegar a saturar estos canales de comunicación, y el rendimiento de las aplicaciones empeoraría sustancialmente

[BE07].

En la Figura 1.10 se muestra un ejemplo de co-asignación, donde dos aplicaciones compiten por el ancho de banda de los canales de comunicación. El entorno Multi-Cluster es el mismo que se ha utilizado en los dos ejemplos anteriores. Sin embargo, en este caso, las dos aplicaciones que han de asignarse están compuestas por 4 tareas, por lo que ninguna de ellas puede ser asignada en un único cluster y han de ser co-asignadas.

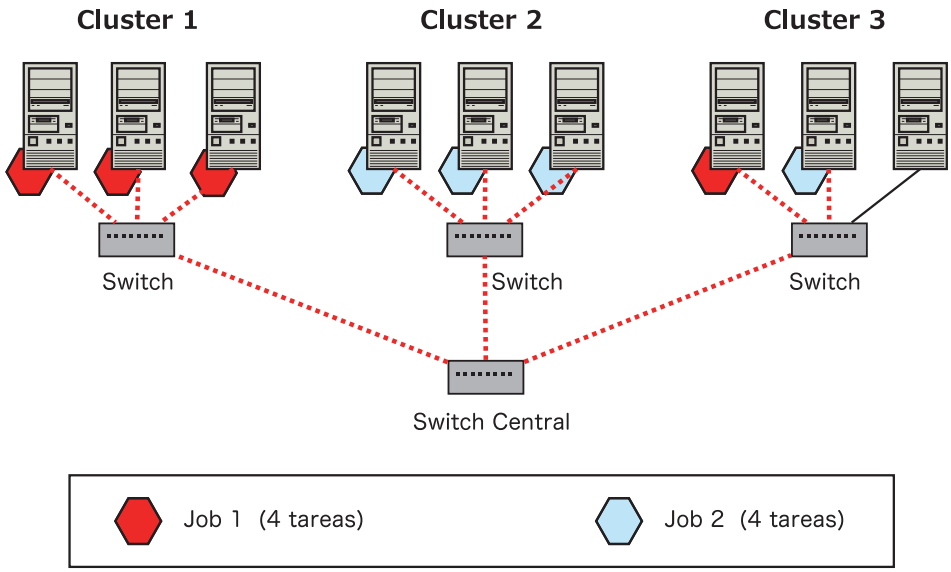


Figura 1.10: Efecto de la co-asignación sobre los enlaces de comunicaciones

En este ejemplo, la primera aplicación ha sido co-asignada de forma que 3 de sus tareas han sido asignadas al Cluster 1 y la tarea restante, al Cluster 3. En el caso de la segunda aplicación, 3 de sus tareas han sido asignadas al Cluster 2, y la tarea restante, al Cluster 3. Como se puede observar, las diferentes tareas de las dos aplicaciones comunicarán a través de los enlaces de red inter-cluster (línea punteada en rojo), y competirán en particular por el ancho de banda del enlace del Cluster 3, que se encuentra compartido por ambas aplicaciones. En una situación como la presentada en este ejemplo, si los requisitos de ancho de banda de las aplicaciones fuesen muy elevadas, el canal de comunicación compartido por ambas podría llegar a saturarse, y el rendimiento de las dos aplicaciones podría verse afectado seriamente. Por lo

tanto, es de vital importancia tratar de evitar las situaciones de saturación en los canales de comunicación en aquellas situaciones en las que se aplique co-asignación.

En la literatura, se pueden encontrar diferentes estudios basados en co-asignación. En [BE07] se analizan diferentes técnicas de planificación que usan co-asignación, y se concluye que usar co-asignación sin restricciones no es recomendable. Otros estudios han tratado con la co-asignación, desarrollando técnicas de balanceo de carga [YTCC08], seleccionando los mejores recursos [NLYW05], o tratando de minimizar el numero de enlaces de red inter-cluster usados [JLPS05] pero sin encontrar un compromiso entre ambos criterios.

En el Capítulo 2 se expone un modelo de aplicación que tiene en cuenta la co-asignación de las aplicaciones, y trata de valorar el efecto que esto puede producir en los canales de comunicación, y por lo tanto, en los tiempos de ejecución de las aplicaciones (Sección 2.2.2). Este modelo, por sus características y consideraciones será la base de nuestro trabajo.

1.3. Estado del arte

Las técnicas de planificación han evolucionado al ritmo que lo ha hecho la arquitectura de los sistemas de cómputo. Los primeros estudios sobre distintas arquitecturas en sistemas de memoria distribuida fueron realizados por Feitelson y Rudolph [FR90], y se centra únicamente en sistemas centralizados o paralelos. No es hasta principios del siglo XXI que se propone una arquitectura de planificación jerárquica para sistemas distribuidos [TD01]. Actualmente un gran número de sistemas comerciales y de código abierto [Con, PBS, Ope, Loa, Mau] utilizan una arquitectura de planificación jerárquica. En [BE02, BBE03] se analizan distintas arquitecturas de planificación, y concluye que la mejor ópción es la de usar una arquitectura con un planificador local en cada cluster, y un planificador global con la información del resto del sistema para poder efectuar planificaciones más eficientes. Numerosos estudios [SvANS00, SKSS02b, EHS+02, Aba04, TRA+06, LSG+08] avalan los buenos resultados obtenidos en sistemas distribuidos usando una arquitectura de planificación jerárquica.

En la literatura existen múltiples estrategias para el proceso de planificación de aplicaciones paralelas. Estas se pueden dividir en dos categorías; *modo off-line* y *modo on-line*.

En el modo **off-line**, se tiene conocimiento de todas las aplicaciones que han de ser planificadas, y por lo tanto, para tomar las decisiones de planificación se puede considerar todo el conjunto de aplicaciones paralelas al mismo tiempo [FRS05]. En el modo **on-line**, únicamente se tiene conocimiento de las aplicaciones paralelas que han ido llegando al sistema, y por lo tanto, las decisiones de planificación están restringidas a un conjunto más reducido de aplicaciones y se desconoce las que pueden llegar en el futuro [Sga98]. El enfoque principal del presente trabajo de tesis se centra en sistemas Multi-Cluster donde las aplicaciones llegan de forma dinámica, y por lo tanto la planificación ha de realizarse en modo on-line. En los casos donde la tasa de llegada es baja, es decir, las aplicaciones llegan muy separadas y la cola del sistema tiene pocas aplicaciones, la posible interacción entre ellas en la compartición de recursos de cómputo y/o comunicación es poco importante. En estos casos solo nos de-

bemos centrar en la asignación de los recursos disponibles según los requisitos de la aplicación. En los casos donde la tasa de llegada es alta, las aplicaciones se acumulan en la cola y pueden interactuar en la utilización de los recursos de comunicación, y entonces es necesario no solo escoger bien los recursos, sino asignar las aplicaciones de forma que la compartición de recursos les afecte lo menos posible. Para abordar este problema se pueden aplicar técnicas de clusterización, o empaquetamiento, que nos permita evaluar la planificación del conjunto de aplicaciones, según los objetivos de rendimiento deseados.

Habitualmente las técnicas on-line asignan una única aplicación al mismo tiempo, sin tomar en consideración los requisitos y características del resto de aplicaciones presentes en la cola del sistema. Esto provoca que una información muy relevante para mejorar el rendimiento global no es considerada. La mayoría de estrategias, además, consideran las aplicaciones recorriéndolas en el orden en que van llegando al sistema. Esto puede provocar que algunos recursos del sistema que están disponibles, al final no sean aprovechados, por lo que el grado de utilización del sistema empeora. Para tratar de resolver esta limitación se han propuesto otras técnicas como llamada *backfilling*, que permite avanzar aplicaciones pequeñas de la parte posterior de la cola del sistema, y así aprovechar esos recursos disponibles [SP94]. Ésta técnica se implementó originalmente en el scheduler *EASY* [Lif95]. Existen dos versiones de la técnica *backfilling*, la conservativa y la agresiva. En la primera, las reservas realizadas son respetadas a la hora de asignar las aplicaciones, mientras que en la versión agresiva, éstas reservas pueden ser ser ignoradas, decidiendo avanzar otra aplicación, retrasando aquellas que se encontraban reservadas. Sabin et al. proponen en [SKRS03] una estrategia basada en backfilling para la planificación de aplicaciones paralelas en sistemas heterogéneos, definiendo simultáneamente múltiples reservas. El estudio concluye que el uso de backfilling conservativo en sistemas heterogéneos proporciona mejores resultados que los obtenidos con técnicas basadas en backfilling agresivo. En [WMW02], Ward et al. proponen el uso de una versión relajada de backfilling conservativo en la que se permite avanzar aplicaciones con baja prioridad si esto no provoca un retraso elevado en las aplicaciones de alta prioridad con reservas activas. En [TEF07], Tsafrir et al. proponen la incorporación de mecanismos de predicción del tiempo de

ejecución de las aplicaciones en la planificación basada en backfilling, debido a que se considera que las estimaciones de tiempo proporcionadas por los usuarios son imprecisas, y esto podría aumentar el número de aplicaciones que son descartadas por el sistema. En [YYWZ00], Yuan et al. proponen mejoras sobre el scheduler *EASY*, entre las que destaca la capacidad de utilizar predicciones del tiempo de ejecución para decidir avanzar aplicaciones, manteniendo la capacidad para evitar la violación de reservas por parte de otras aplicaciones. En resumen, estos estudios proponen técnicas que obtienen mejoras en el rendimiento de las aplicaciones mediante la modificación del orden de ejecución de las mismas.

Las técnicas previamente presentadas son extensivamente usadas en entornos distribuidos. Sin embargo, están basadas en características específicas de los entornos para las que han sido diseñadas. Además, en la mayoría de los casos, estas técnicas asumen la utilización de aplicaciones con tareas independientes, esto es, sin considerar ningún tipo de restricción y efecto en las comunicaciones. El presente trabajo de tesis está enfocado principalmente en la planificación de conjuntos de aplicaciones paralelas con tareas dependientes, en que las comunicaciones pueden jugar un papel importante. En la Sección 2.2 se presenta en detalle el modelo de aplicación que se asume en el trabajo de tesis.

La investigación en entornos Grid y Multi-Cluster ha proporcionado múltiples técnicas de planificación, basadas en diferentes criterios: coste, makespan, reducción de las comunicaciones, etc. Feng et al. proponen en [FSZX03] un modelo de optimización de coste de deadline para la planificación de una aplicación, basado en programación entera binaria que respeta los requisitos de calidad del servicio (*QoS*) de las aplicaciones. Buyya et al. proponen en [BMAV05] una algoritmo con restricciones de coste de ejecución y plazo de finalización (*deadline*) para la planificación eficiente de una aplicación, considerando también los requisitos de calidad del servicio. Este algoritmo se basa en un modelo de economía del procesamiento, de forma que el coste de ejecución es evaluado en forma del gasto que supone de ejecutar una aplicación en un conjunto determinado de recursos.

Jones et al. proponen en [JLPS05] un modelo de aplicación paralela que

asume aplicaciones paralelas con tareas dependientes. En el estudio se propone una estrategia de planificación que minimiza la utilización de los canales de comunicación inter-cluster agrupando por lo menos un 75 % de las tareas de cada aplicación en un único cluster. Esta técnica tiene en cuenta la co-asignación de las aplicaciones paralelas, así como las interacciones de las mismas. Sin embargo, no se tienen en cuenta los requisitos de procesamiento de las aplicaciones. Naik et al. proponen en [NLYW05] modelos basados en programación lineal que proporciona la asignación óptima, según los mejores recursos de procesamiento o los mejores recursos de comunicación, pero sin tener ambos tipos de recursos en cuenta al mismo tiempo. Estas están restringidas a la optimización de un único criterio, sin poder llegar a un compromiso entre los requisitos de procesamiento y los de comunicación. Para tratar de salvar ésta restricción, Lérida et al. proponen en [LSG+08] un modelo basado en programación lineal que obtiene la asignación óptima de una aplicación paralela con tareas dependientes, teniendo en cuenta tanto los recursos de procesamiento como de comunicación de las aplicaciones. Este modelo considera la co-asignación de aplicaciones, y descarta aquellas soluciones en que se produce saturación en los canales de comunicación. Sin embargo, estas técnicas evalúan una única aplicación al mismo tiempo, y por lo tanto, no se tienen en cuenta los requisitos del resto de aplicaciones que están presentes en la cola del sistema. Esto provoca que no todas las posibles decisiones de planificación sean tenidas en cuenta, y que recursos del sistema sean desaprovechados.

Diferentes estudios han explotado la planificación de aplicaciones concurrentes, con el fin de mejorar el rendimiento de un conjunto de aplicaciones, teniendo en cuenta las características del conjunto. Martino et al. [MM02] y Xhafa et al. [XCA07] proponen la utilización de algoritmos genéticos para resolver la planificación de conjuntos de aplicaciones en entornos Grid heterogéneos, tratando de mejorar la utilización de los recursos o con el objetivo de minimizar el makespan del conjunto de aplicaciones. Shmueli et al. proponen en [SF05] un algoritmo de optimización basado en *look-ahead* para realizar una selección local óptima de un conjunto de aplicaciones a la hora de aplicar backfilling, basada en programación dinámica. En este estudio se concluye que considerando varias aplicaciones a la vez, el proceso de planificación po-

día mejorar el rendimiento del conjunto. Shah et al. proponen en [SQR10] un algoritmo cuasi-óptimo para el empaquetamiento de aplicaciones, y capaz de reducir significativamente la fragmentación externa. Estas técnicas intentan asignar aquellas aplicaciones paralelas que mejor quepan en los espacios libres, sin considerar otras oportunidades de empaquetamiento. Algunos estudios han mostrado que tratando de modificar el orden de ejecución de las aplicaciones se puede mejorar el rendimiento global y la utilización del sistema, además de permitir nuevas oportunidades de optimización [SKSS02a, SF05].

Munir et al. propone en [MLS07] una heurística que tiene en cuenta el ancho de banda de los enlaces de red para respetar la calidad del servicio y reducir el makespan. Sin embargo, en este estudio se asumen aplicaciones paralelas compuestas por tareas independientes. En [KDM09], Kumar et al. estudian la planificación de aplicaciones en entornos heterogéneos desde el punto de vista del coste de ejecución. En este estudio propone un modelo lineal basado en programación entera MIP, y un conjunto de heurísticas para minimizar este coste. El modelo lineal proporciona resultados óptimos, pero a costa de un elevado coste computacional, debido a la dificultad del problema, que es NP-Hard. Sin embargo, esto le permite evaluar el resultado obtenido con las heurísticas propuestas, y determina que estos resultados son cuasi-óptimos. Garg et al., en [GBS10] proponen tres meta-heurísticas que gestionan y minimizan el coste de ejecución y el makespan de las aplicaciones paralelas. En [BMP+10], Benoit et al. proponen un algoritmo óptimo para la planificación off-line de conjuntos de aplicaciones, que minimiza el máximo cuociente entre el tiempo que una aplicación paralela ha estado en el sistema y el que hubiese estado si hubiese sido asignada sola en el sistema. Para el escenario de planificación on-line, proponen una heurística basada en el conocimiento extraído del algoritmo del escenario off-line. Sin embargo, estos estudios consideran únicamente aplicaciones con tareas independientes, sin restricciones de comunicación.

Teniendo en cuenta los diferentes estudios que se han propuesto en la literatura para la planificación de aplicaciones paralelas en entornos heterogéneos, se ha comprobado que algunos de ellos se han enfocado en la evaluación de una única aplicación, compuesta por tareas dependientes o independientes, según el estudio. Por otro lado, algunos estudios han tratado de mejorar el rendimiento

de un conjunto de aplicaciones, evaluándolas todas de forma simultánea, pero asumiendo aplicaciones formadas por tareas independientes. Todo lo anterior plantea como hipótesis para este trabajo que se puede mejorar el rendimiento de conjuntos de aplicaciones paralelas compuestas por tareas dependientes, evaluándolas de forma simultánea, de forma que además se pueda mejorar el aprovechamiento de los recursos del sistema, y en ese sentido se enfoca el presente trabajo de tesis.

1.4. Motivación y objetivos

Como se ha presentado en la Sección 1.3, en la literatura es posible encontrar múltiples soluciones al problema de la planificación de aplicaciones paralelas en entornos Multi-Cluster heterogéneos. Habitualmente estas técnicas han sido desarrolladas con el objetivo de evaluar únicamente una aplicación, sin considerar el resto de aplicaciones que pudiesen estar en la cola del sistema, desaprovechando posibles oportunidades de planificación. Algunos estudios han mostrado que realizar el proceso de planificación teniendo en cuenta varias aplicaciones de forma simultánea, permite aprovechar nuevas oportunidades de asignación, lo que puede implicar mejoras en el rendimiento global de las aplicaciones y en la utilización del sistema. Sin embargo, estas técnicas suelen asumir aplicaciones paralelas con tareas independientes, en las que no se consideran los efectos de la comunicación.

El objetivo global de este trabajo de tesis es el estudio de nuevas técnicas de planificación en entornos Multi-Cluster heterogéneos con co-asignación, que permitan mejorar el rendimiento global de un conjunto de aplicaciones paralelas, evaluadas de forma múltiple, teniendo en cuenta la disponibilidad de los recursos y su heterogeneidad, así como los requisitos de procesamiento y de comunicación de las aplicaciones paralelas.

Este objetivo global se concreta considerando los siguientes objetivos específicos:

1. Modelar el tiempo de ejecución de aplicaciones paralelas formadas por tareas dependientes, considerando la disponibilidad de los recursos y su heterogeneidad, teniendo en cuenta los requisitos de procesamiento y de comunicación de estas aplicaciones.

2. Modelar la optimización de la utilización de los recursos en un entorno Multi-Cluster heterogéneo, teniendo en cuenta la co-asignación de aplicaciones, que puede provocar congestión en los canales de comunicación.

3. Modelar el comportamiento de la planificación múltiple, en la que un conjunto de aplicaciones paralelas son evaluadas de forma simultánea.

4. Estudiar los métodos de planificación más habituales de la literatura y proponer e implementar nuevas técnicas de planificación múltiple, siguiendo los criterios expuestos anteriormente.

5. Evaluar las técnicas propuestas mediante el uso tanto de workloads sintéticos presentes en la literatura así como workloads extraídos de trazas reales de entornos de cómputo.

1.5. Estructura del documento

El resto del trabajo de tesis se estructura de la forma siguiente. En el Capítulo 2 se presenta la modelización del problema. En la Sección 2.1 se presenta un modelo que representa el funcionamiento de un entorno Multi-Cluster heterogéneo. En la Sección 2.2 se presenta un modelo que permite expresar el tiempo de ejecución de una aplicación paralela formada por tareas dependientes, teniendo en cuenta sus requisitos de procesamiento y de comunicación, así como la disponibilidad de los recursos del sistema Multi-Cluster.

En el Capítulo 3 se presenta nuestra propuesta de planificación múltiple en entornos Multi-Cluster. Para cada una de las técnicas propuestas se describe su implementación, se desarrolla un conjunto de experimentos para evaluar su comportamiento y rendimiento, y se presentan las conclusiones extraídas del estudio realizado. En la Sección 3.1 se presenta *Package Allocation Strategy* (PAS), una técnica consistente en una mecanismo heurístico de selección de aplicaciones, y un modelo MIP capaz de proponer una asignación óptima para un conjunto de aplicaciones. En la Sección 3.2 se presenta *Ordering and Allocation Strategy* (OAS), una técnica consistente en un modelo MIP capaz de proponer una solución cuasi-óptima, determinando tanto el orden de ejecución como la asignación para un conjunto de aplicaciones paralelas con el objetivo de mejorar el rendimiento del conjunto, y mejorar el aprovechamiento de los recursos del sistema Multi-Cluster. Finalmente, en la Sección 3.3 se presenta *Minimum Execution Slowdown* (MES), una técnica heurística de planificación, capaz de determinar el orden de ejecución y la asignación para un conjunto de aplicaciones con el fin de reducir el slowdown de ejecución del conjunto, que permite además reducir significativamente el tiempo en obtener las soluciones.

En el Capítulo 4 se presentan las principales conclusiones que se extraen del trabajo realizado, y se exponen el conjunto de líneas de investigación abiertas que consideramos interesante abordar en un trabajo futuro.

Capítulo 2

Caracterización del entorno Multi-Cluster y de las aplicaciones paralelas

En el presente capítulo se presenta un modelo que representa las características de un sistema Multi-Cluster heterogéneo y no dedicado (Sección 2.1), y un modelo que permite expresar el tiempo de ejecución de una aplicación paralela, una vez asignada al entorno Multi-Cluster, considerando la heterogeneidad tanto de los recursos de procesamiento como de comunicación (Sección 2.2).

El uso de modelos analíticos permite representar mediante formulación matemática el rendimiento y comportamiento de las aplicaciones ejecutadas bajo unas determinadas condiciones del entorno, y es de gran importancia para el desarrollo de técnicas de planificación.

2.1. Modelización del entorno Multi-Cluster

Un entorno Multi-Cluster consiste en la unión de un conjunto de clusters mediante una red de comunicaciones dedicada, estando los recursos del mismo, dentro de una misma organización, empresa, o campus universitario, tal y como fue presentado en la Sección 1.2. Una característica importante es que en un entorno Multi-Cluster la disponibilidad y capacidad de los recursos es

predecible y controlable, mientras que otros entornos de computación, como Grid, se caracterizan por ser dinámicos y mutables, con lo que no se puede garantizar la disponibilidad de estos recursos en el tiempo. La otra característica principal de un entorno Multi-Cluster es que las redes de comunicación de los diferentes Clusters se conectan mediante enlaces de red dedicados. Este hecho tiene una implicación importante, y es que el comportamiento y la capacidad de estos enlaces de comunicación es predecible, mientras que en otros entornos que se comunican a través de internet, el comportamiento del cual es mucho más dinámico y difícil de controlar y predecir.

Teniendo estos aspectos en cuenta, se asume un entorno Multi-Cluster formado por un número α de clusters de tamaño arbitrario y con recursos heterogéneos. De esta forma, definimos $M = \{C_1, ..., C_\alpha\}$ como el conjunto de clusters; se define $R = \{R_1^1, R_2^1, ..., R_{n-1}^\alpha, R_n^\alpha\}$ como el conjunto de recursos de procesamiento del Multi-Cluster, siendo n es el número total de nodos. Cada cluster está conectado a los demás mediante un enlace dedicado, a través de un switch central. De esta forma, se define $\mathcal{L} = \{\mathcal{L}_1, ..., \mathcal{L}_\alpha\}$ como el conjunto de enlaces inter-cluster, siendo $\mathcal{L}_k$ el enlace que conecta el cluster C_k con el switch central.

En la figura 2.1 se muestra una representación gráfica a modo de ejemplo de un entorno Multi-Cluster genérico. En esta figura, el Multi-Cluster que se representa está formado por $n = 11$ nodos y $\alpha = 3$ clusters $\{C_1, C_2, C_3\}$, formados por $\{3, 5, 3\}$ nodos cada uno respectivamente. Por lo tanto, $C_1 = \{R_1^1, R_2^1, R_3^1\}$, $C_2 = \{R_4^2, R_5^2, R_6^2, R_7^2, R_8^2\}$ y $C_3 = \{R_9^3, R_{10}^3, R_{11}^3\}$. Los clusters se encuentran conectados a un switch central mediante sus respectivos enlaces de comunicaciones $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$.

En un entorno heterogéneo y no dedicado, la capacidad de procesamiento de un nodo depende de su potencia de procesamiento y de su disponibilidad. Definimos la **Potencia relativa** P^r de un nodo r como la relación entre la potencia de procesamiento del nodo r respecto a otro nodo que se toma como referencia. Este nodo de referencia usualmente será un nodo del entorno multicluster, pues resulta más fácil relacionar recursos que se encuentran en el mismo entorno, aunque es posible utilizar un computador externo al entorno multicluster. Cuando $P^r = 1$ significa que el nodo r tiene la misma potencia relativa

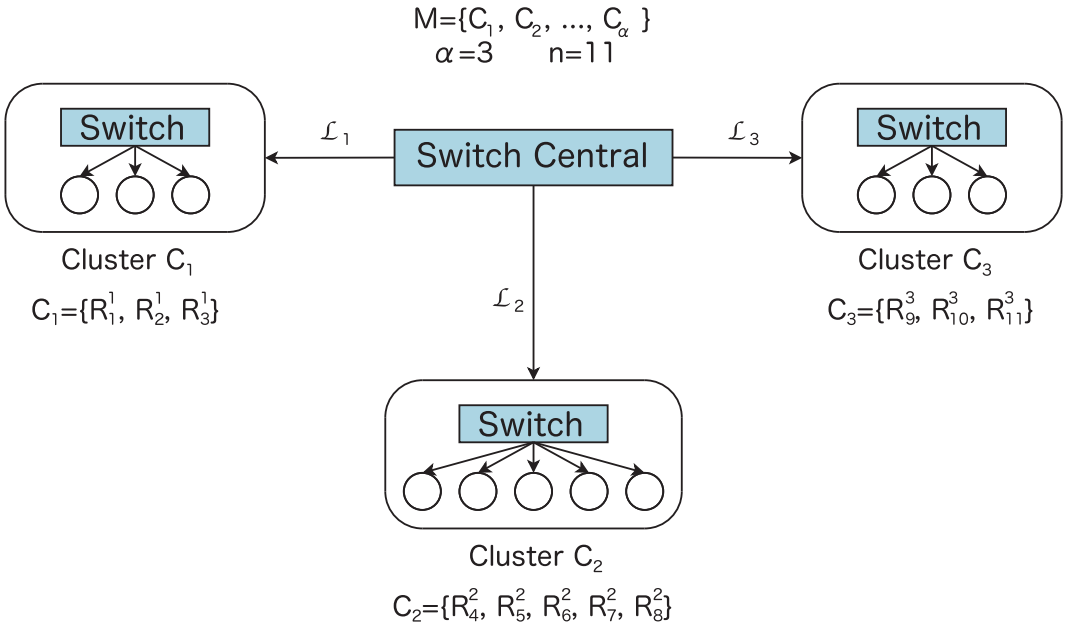


Figura 2.1: Diagrama de un entorno Multi-Cluster genérico

que el nodo usado como referencia, $P^r < 1$ cuando el nodo tiene una potencia relativa inferior. Es posible tomar como referencia un nodo con una potencia inferior, y en ese caso tendríamos que $P^r > 1$. La potencia de procesamiento de un nodo se puede medir promediando los tiempo de procesamiento obtenidos para diferentes tipos de benchmarks [DZ97].

Se define **Disponibilidad** A^r de un nodo r como el porcentaje de CPU que se encuentra disponible, y representado mediante un valor normalizado en el rango $\{0 \ldots 1\}$. En un entorno no dedicado puede darse el caso que un nodo esté ejecutando aplicaciones locales, compartiendo su tiempo de CPU, lo que afecta a su grado de disponibilidad. Se puede aproximar la disponibilidad de un nodo r relacionando el promedio de aplicaciones en el sistema con el uso de los ciclos de procesamiento [WSH00].

Teniendo en cuenta las consideraciones anteriores, definimos la **Potencia Efectiva** Γ^r de un nodo r como la capacidad de ese nodo para ejecutar tareas, expresado en la ecuación 2.1.

$$\Gamma^r = P^r \cdot A^r \tag{2.1}$$

De este modo, cuando $\Gamma^r = 1$ el nodo r es capaz de ejecutar una tarea a una velocidad igual a la del nodo de referencia. $\Gamma^r < 1$ implicará que el nodo r ejecutará las aplicaciones a una velocidad inferior a la del nodo de referencia. En cambio, si $\Gamma_r > 1$ el nodo r es capaz de ejecutar las aplicaciones a una mayor velocidad que el nodo utilizado como referencia. El efecto de este parámetro en el procesamiento de las tareas asignadas será tratado en detalle en la Sección 2.2.1.

Los enlaces de comunicaciones del Multi-Cluster también han de ser tenidos en cuenta, ya que estos serán compartidos tanto por los procesos locales como por las aplicaciones paralelas que se hayan de ejecutar en el entorno. Se define el **Máximo Ancho de Banda** disponible MBW^k como la capacidad máxima del canal $\mathcal{L}_k$ para transmitir datos. Las comunicaciones pueden afectar a las aplicaciones paralelas que comparten ese canal, ya que si se congestiona, podría ralentizar su ejecución al retardar la retransmisión de los datos computados entre las tareas de la aplicación paralela. El efecto de las comunicaciones sobre las aplicaciones será tratado en detalle en la Sección 2.2.2.

El modelo de entorno Multi-Cluster presentado caracteriza una representación general de este tipo de entornos, y puede ser adaptada para representar configuraciones más específicas. Una de estas configuraciones podría ser un entorno Multi-Cluster, donde algunos de los clusters podrían estar conectados en forma de cascada a un switch intermedio, en vez de estar todos conectados a un único switch central. Este entorno podría ser representado mediante el modelo propuesto, determinando los valores de ancho de banda de los canales de los diferentes clusters, reduciendo su valor para representar que ese canal está siendo compartido por varios clusters. De esta forma, otras configuraciones más específicas podrían ser representadas, ajustando de forma adecuada los parámetros del modelo.

2.2. Modelización de aplicación paralela

Desde la creación y uso de sistemas distribuidos, se han desarrollado diferentes modelos de aplicaciones para poder aprovechar eficientemente las capacidades de estos recursos computacionales. Dos de los paradigmas más ampliamente utilizados en el desarrollo de aplicaciones paralelas son *Task Parallelism* y *Data Parallelism*. Mientras que en el primer caso, diferentes funciones de la aplicación trabajan sobre un mismo bloque de datos, en el segundo un mismo conjunto de funciones trabajan sobre bloques de datos diferentes. El presente trabajo se centra en el paradigma de *Data Parallelism*. Dentro de este paradigma, se pueden distinguir dos modelos de programación principales: *Master-Worker* [GKYL01, BMP+10] y *Bulk-Synchronous Parallel* (BSP) [SHM97].

En el modelo *Master-Worker* (MW), una aplicación paralela se compone de una tarea principal, o *Master*, y una serie de tareas secundarias, o *Workers*. En este tipo de aplicaciones, la tarea *Master* es la encargada de dividir el problema en sub-problemas que son asignados a cada uno de los *Workers*. Estos, una vez han finalizado su parte de cómputo, retornan a la tarea *Master* los resultados obtenidos. Finalmente la tarea *Master* reconstruye la solución final a partir de los resultados parciales. Este modelo ha sido utilizado ampliamente en diferentes áreas, como análisis y diseños de ingeniería [ASGH95], simulaciones de *Monte-Carlo* [BRL99], optimización matemática [ANF03], o como base para plataformas de cómputo colaborativo a través de internet como Condor [Con], SETI@Home [Set] y BOINC [boi].

En el modelo *Bulk-Synchronous Parallel* (BSP), una aplicación paralela está compuesta por un conjunto de tareas similares, que trabajan de forma colaborativa. Cada una de las tareas consta de una serie de iteraciones en las que se realiza una etapa de cómputo seguido de una etapa de comunicación, en el que las tareas se sincronizan y pueden compartir parte de los datos procesados. Este modelo de aplicación paralela es ampliamente utilizado en diferentes áreas, principalmente en computación científica, en aplicaciones de simulación de dinámicas de fluidos [CG95], de computación electromagnética [MPW94], simulaciones de plasma [NNS95], o estudios matemáticos y de combinatoria

[Sal04, KT06].

A parte del modelo de programación en el que se han diseñado las aplicaciones, éstas pueden diferenciarse por su estructura de tareas: independientes y dependientes. Una aplicación con tareas independientes, típicamente dividirá un problema en trozos más pequeños, sin dependencia de datos, y los procesará en los recursos del entorno. Una variante de este tipo son las llamadas *aplicaciones paramétricas* [BFH03], en las que una aplicación se replica a diferentes recursos de cómputo, con valores de entrada diferentes para el problema, que se procesan de forma independientemente. Por lo general, en este tipo de aplicaciones, las comunicaciones pasan a un segundo plano, ya que estas se suelen limitar a los procesos de transmisión del conjunto pequeño de datos de entrada y a la recepción de los resultados, una vez que la ejecución ha finalizado.

En el caso de una aplicación con tareas dependientes, ésta está formada por un conjunto de tareas con algún tipo de dependencia entre ellas: sincronismo o precedencia [Pin02, CJLL95]. En este tipo de aplicaciones, las comunicaciones pueden jugar un papel mucho más importante, ya que la dependencia de las tareas implica que se realizará algún proceso de comunicación, más allá de la simple comunicación de resultados. Esto es más evidente en el caso de las aplicaciones en que existe un proceso de sincronía, como en el caso de las aplicaciones *BSP*, en el que sucesivas iteraciones constan de una etapa de procesamiento, seguida de un punto de sincronía en el que parte de las tareas, o todas, deben realizar una comunicación entre ellas para compartir datos o para notificar la finalización de su respectiva iteración. En estos casos, pues, es de vital importancia tener en cuenta la disponibilidad y capacidad de los canales de comunicación que se utilizarán en la planificación de la aplicación, a parte de la correcta selección de los recursos de procesamiento.

En el presente trabajo de tesis se adopta como modelo para la aplicación paralela general, propuesto por Lérida en [LSG+08], que representa los aspectos generales de los diferentes modelos y tipos de aplicación que se han expuesto. Estos aspectos principales a tener en cuenta, son:

1. Las aplicaciones están formadas por un número fijo de tareas.

2. Las aplicaciones pueden estar formadas por un número diferente de ta-

reas.

3. Las tareas de una aplicación pueden ser dependientes entre ellas.

4. Una aplicación puede estar compuesta por una parte de procesamiento, y otra de comunicación, pudiendo esta última ser nula para representar casos en los que las tareas fueran independientes y no colaborativas.

Siguiendo estas pautas, cada aplicación paralela j, perteneciente al conjunto de aplicaciones J, presentes en la cola de espera del sistema, está compuesta por un número fijo de tareas τ_j, que trabajan de forma colaborativa entre ellas. Cada una de estas tareas tiene unos requisitos de procesamiento y comunicación similar a las otras. Esta representación más general permite que aplicaciones tanto comunicativas como no comunicativas puedan representarse, sin entrar en detalle en su composición interna. Por ejemplo, una aplicación con tareas independientes podrá representarse como una aplicación con una parte de procesamiento mucho más pesada que la de comunicación, pudiendo tener ésta última una importancia prácticamente nula. En cambio, aplicaciones con tareas dependientes como por ejemplo las *BSP* pueden a su vez ser representadas como una aplicación en que la parte de comunicación ya tendría un peso más importante en la composición de la aplicación. De cara a la simplificación del modelo, y para considerar los casos extremos, se asume un esquema de comunicación entre tareas siguiendo el patrón *all-to-all*, en el que todas las tareas que componen la aplicación j comunican entre ellas. Siguiendo este esquema, se considera que una aplicación no finaliza su ejecución hasta que la más lenta de sus tareas haya finalizado.

Se asume una asignación estática de las tareas de una aplicación. Esto implica que una vez que se hayan asignado las diferentes tareas y se inicie la ejecución de la aplicación, esta asignación no podrá ser alterada. Esta asignación podrá ser realizada mediante co-asignación (Sección 1.2.3), lo que permite la ejecución de aplicaciones que no podrían haber sido asignadas a un único cluster. Además, se reduce la fragmentación interna del entorno, al poderse aprovechar nodos sin asignar en los diferentes clusters.

Se define el tiempo base de una aplicación (Tb_j) como el tiempo de ejecución obtenido por esa aplicación en un entorno dedicado, que se toma co-

mo referencia. El tiempo de ejecución se desglosa en dos componentes, tal y como se muestran en la Figura 2.2: tiempo de procesamiento, y tiempo de comunicación. La relación de la parte de procesamiento respecto a la parte de comunicación se representa mediante el parámetro normalizado σ_j, y que es determinado a partir de las características de cada aplicación en particular. Con esto, σ_j representa la parte proporcional relativa al procesamiento, y $(1 - \sigma_j)$ representa la parte proporcional relativa a la comunicación.

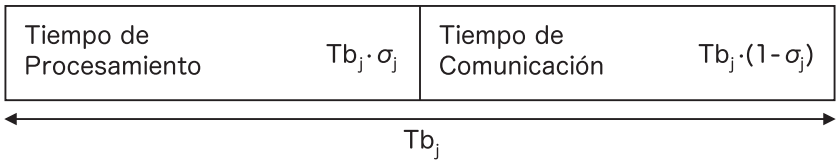


Figura 2.2: Representación del tiempo base de una aplicación paralela

La asignación a lo largo de diferentes clusters, puede implicar la utilización de recursos de procesamiento y la compartición de canales de comunicación de diferentes capacidades. En estas situaciones, tanto la heterogeneidad de los recursos de procesamiento como de comunicación, y el grado en que los canales de comunicación sean ocupados, tendrán un efecto sobre las aplicaciones paralelas. Este efecto se verá representado como un retraso en el tiempo final de ejecución respecto al tiempo base de las mismas. Cuanto más grande sea el retraso, mayor será el tiempo de esa aplicación para finalizar su ejecución. El grado de éste retraso irá en función de las capacidades y la disponibilidad de los recursos utilizados.

Teniendo en cuenta estas consideraciones, se define el tiempo de ejecución de una aplicación paralela (Te_j), mediante la ecuación 2.2.

$$Te_j = Tb_j \cdot ct_j, \quad \forall j \in J \tag{2.2}$$

donde Tb_j denota el tiempo base de la aplicación j, es decir, el tiempo de referencia obtenido ejecutando la aplicación en un entorno dedicado, y ct_j es el coste derivado de la utilización de los recursos asignados y que afecta a este tiempo base. Se asume que el tiempo base de cada aplicación j, es obtenida por el usuario mediante ejecuciones reales, monitoreo, o técnicas de benchmarking.

En estudios previos de la literatura [BE07, HHL+06, Wol03], el efecto sobre

el tiempo de ejecución de las aplicaciones paralelas era estimado a partir de las capacidades de procesamiento de los recursos asignados, sin considerar las comunicaciones. En [EHS+02] se presenta un modelo que considera únicamente una penalización fija en la comunicación, cuando se produce co-asignación. En [BBE03, BE01, EHS+02] se analiza mediante simulación el efecto de las comunicaciones sobre el rendimiento de determinadas aplicaciones, pero no se establece un método generalista válido. En otros estudios se presentan modelos que consideran únicamente el efecto de las comunicaciones sin tener en cuenta la heterogeneidad de los recursos de procesamiento [JLPS05, JAA07].

En [LSG+08] se propone una representación del coste de ejecución en el que se tienen en cuenta tanto la heterogeneidad de los recursos de procesamiento como los de comunicación. En el presente trabajo se toma como referencia esa representación del efecto sobre la ejecución de las aplicaciones paralelas, y se expresa mediante la ecuación 2.3.

$$ct_j = \sigma_j \cdot SP_j + (1 - \sigma_j) \cdot SC_j, \quad \forall j \in J \tag{2.3}$$

donde SP_j y SC_j denotan el retraso, o slowdown, producido por los recursos de procesamiento, y los canales inter-cluster usados, respectivamente. σ_j denota la relevancia del procesamiento respecto a la comunicación de la aplicación j.

En una situación ideal, donde los recursos empleados son dedicados y homogéneos, ambos factores de retraso toman la unidad como valor ($SP_j = SC_j = 1$), lo que implica que la ejecución de la aplicación no sufre ningún retraso.

En la Figura 2.3 se representa el tiempo de ejecución de una aplicación paralela. En esta figura se observa el tiempo de ejecución desglosado en sus dos componentes: el tiempo de procesamiento, y el tiempo de comunicación. Cada uno de ellos puede expresarse como el efecto que hace variar a la parte proporcional, representada por σ_j, de su tiempo base.

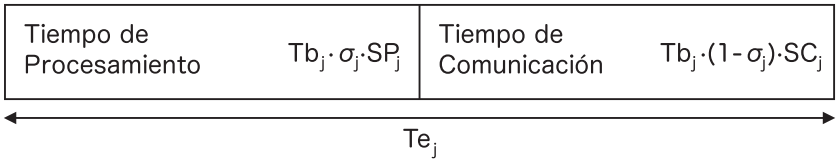


Figura 2.3: Representación del tiempo de ejecución de una aplicación paralela

A continuación se propone un método para obtener los valores de retraso de procesamiento (SP) (Sección 2.2.1) y de comunicación (SC) (Sección 2.2.2).

2.2.1. Slowdown de Procesamiento (SP)

Tal y como se explicó en la sección 2.1, cada nodo r del Multi-Cluster tiene una potencia efectiva Γ^r, que denota su capacidad para ejecutar aplicaciones. Podemos calcular el retraso de procesamiento SP_j^r que cada nodo r infligirá sobre la aplicación paralela j, mediante la ecuación 2.4.

$$SP_j^r = \begin{cases} 0 & \textit{cuando } r \notin S_j \\ (\Gamma_r)^{-1} & \textit{en caso contrario} \end{cases}, \quad \forall r \in R, j \in J \tag{2.4}$$

donde S_j es el conjunto de nodos asignados a la aplicación j para su ejecución. Un nodo asignado tendrá un efecto sobre la aplicación que será inversamente proporcional a su potencia efectiva. Es decir, cuanto más potente sea ese nodo, menor será el retraso que infligirá sobre la aplicación j, y viceversa. Los nodos que no son asignados a una aplicación no producen ningún retraso sobre la misma, por lo que el valor de su slowdown de procesamiento será cero, independientemente de la potencia efectiva del nodo.

Asumiendo la similaridad de las tareas que forman la aplicación j, y que estas se ejecutan de forma separada, se define el **Slowdown de Procesamiento** de la aplicación j como el slowdown máximo producido por los nodos asignados, tal y como se expresa en la ecuación 2.5.

$$SP_j = \max_{\forall r \in R}\{SP_j^r\}, \quad \forall j \in J \tag{2.5}$$

Es decir. el retraso de procesamiento de la aplicación j vendrá definido por el nodo asignado con menor potencia efectiva.

2.2.2. Slowdown de Comunicación (SC)

La caracterización de las comunicaciones se basa en el modelo propuesto por W. Jones en [JLPS05] para entornos homogéneos y dedicados. Jones no consideró la actividad del usuario local en la interacción de las comunicaciones

ni la posibilidad de disponer de recursos con distintas capacidades.

Se asume que el grado de congestión que se producirá en los canales de comunicación entre nodos del mismo cluster, será inferior al que se producirá en los canales de comunicación entre clusters, ya que estos últimos han de soportar las comunicaciones de las diferentes aplicaciones que se ejecutan a lo largo de todo el Multi-Cluster. Teniendo en cuenta esta consideración, y con el objetivo de simplificar la representación del efecto de las comunicaciones sobre las aplicaciones paralelas, no se tendrá en cuenta en el presente trabajo el retraso producido por los canales de comunicación entre nodos de un mismo cluster y se centrará exclusivamente en las comunicaciones entre diferentes clusters.

Suponiendo una aplicación j formada por τ_j tareas distribuidas entre diferentes clusters, y suponiendo que en el cluster $\mathcal{L}_k$ hay asignadas t_j tareas, el ancho de banda consumido para el canal de comunicación correspondiente al cluster $\mathcal{L}_k$ es expresado mediante la ecuación 2.6.

$$BW_j^k = \left(t_j^k \cdot PTBW_j\right) \cdot \left(\frac{\tau_j - t_j^k}{\tau_j - 1}\right), \quad \forall k \in 1 \ldots \alpha, j \in J \tag{2.6}$$

donde $PTBW_j$ es el ancho de banda requerido por cada tarea de la aplicación j, τ_j es el número de tareas que componen dicha aplicación, y t_j^k es el número de estas tareas que se han asignado en el cluster C_k. El primer término de la ecuación expresa el ancho de banda total consumido por las tareas que están asignadas dentro del cluster C_k. El segundo término de la ecuación expresa el porcentaje de comunicación que se realizará con el resto de clusters.

Cuando las aplicaciones co-asignadas consumen una cantidad de ancho de banda superior al disponible en un canal, este se congestiona, y se considera que se encuentra saturado. Ésta saturación tiene un efecto contraproducente sobre las aplicaciones, ya que puede incrementar notablemente su tiempo de comunicación, afectando por lo tanto a su tiempo total de ejecución en forma de un slowdown de comunicación (SC) más elevado. El grado de saturación de un canal $\mathcal{L}_k$ relaciona el máximo ancho de banda de éste canal con los requisitos de ancho de banda de las aplicaciones paralelas asignadas que lo comparten, y es expresado mediante la ecuación 2.7.

$$SAT^k = \frac{MBW^k}{\sum_{\forall j}(BW_j^k)}, \quad k \in 1 \ldots \alpha, j \in J \tag{2.7}$$

dónde MBW^k denota el máximo ancho de banda del canal $\mathcal{L}_k$. Cuando $SAT^k \geq 1$, el canal inter-cluster $\mathcal{L}_k$ no se encuentra saturado. En caso contrario, el canal $\mathcal{L}_k$ estará saturado, y retrasará la ejecución de las aplicaciones que estén compartiendo ese canal.

El grado de retraso que producirá el canal $\mathcal{L}_k$ sobre la aplicación j se expresa mediante la ecuación 2.8.

$$SC_j^k = \begin{cases} (SAT^k)^{-1} & \text{cuando } SAT^k < 1 \\ 1 & \text{en caso contrario} \end{cases}, \quad \forall k \in 1 \ldots \alpha, j \in J \tag{2.8}$$

Este retraso es inversamente proporcional al grado de saturación SAT_k. En el caso de que no se produzca saturación en el canal $\mathcal{L}_k$, no se producirá retraso sobre la aplicación j, por lo que $SC_j^k = 1$.

Teniendo en cuenta las consideraciones anteriores, el **Slowdown de Comunicación** de la aplicación j viene determinado por el canal inter-cluster utilizado con mayor retraso, tal y como se expresa en la ecuación 2.9.

$$SC_j = \max_{\forall k \in 1 \ldots \alpha} \{SC_j^k, \quad \forall j \in J\}, \tag{2.9}$$

Capítulo 3

Diseño y evaluación de nuevas técnicas de planificación en entornos Multi-Cluster

Las técnicas de planificación más comunes de la literatura tratan las aplicaciones una a una de forma aislada, sin tomar en consideración los requisitos y características del resto de aplicaciones que puedan estar presentes en la cola de espera del sistema. Estudios previos [SF05, SKSS02a] han concluido que mediante la asignación de grupos de aplicaciones se pueden obtener mejoras en el rendimiento global de este conjunto de aplicaciones, al poder tomar decisiones con el fin de mejorar el rendimiento del conjunto, y no el de una aplicación en concreto.

El presente trabajo de tesis está enfocado en la planificación *on-line* de aplicaciones paralelas en entornos Multi-Cluster heterogéneos, donde estas aplicaciones van llegando al sistema con una tasa elevada. Esta tasa de llegada provoca que las aplicaciones puedan irse acumulando en la cola de espera del sistema si los recursos de procesamiento han sido ocupados por otras aplicaciones. En estas situaciones, si las aplicaciones son evaluadas por separado y sin modificar el orden de ejecución de las mismas, se puede producir desaprovechamiento de los recursos disponibles.

Para mostrar el efecto positivo que puede producir el hecho de planificar un conjunto de aplicaciones de forma simultánea, se presenta a continuación

un ejemplo gráfico (Figura 3.1). En la Figura 3.1(a) se muestra la cola de aplicaciones que han de ser planificadas. El eje horizontal representa el tiempo base de las aplicaciones, mientras que el eje vertical representa el número de nodos que requiere cada aplicación.

Una estrategia que tenga en cuenta únicamente una aplicación al mismo tiempo y que no modifica el orden de ejecución de la cola, como por ejemplo *First Come First Serve* (FCFS), producirá una planificación en la que no se aprovechan al máximo los recursos, y se dejarán recursos desocupados durante más tiempo, tal y como se puede observar en la Figura 3.1(b), donde el primer nodo está desocupado durante gran parte del tiempo, así como otros nodos que no son utilizados durante un tiempo, hasta que no se liberan nodos suficientes para asignar la siguiente aplicación. En cambio, una estrategia que pueda tener en cuenta varias aplicaciones simultáneamente, pudiendo conseguir un mayor aprovechamiento los recursos del sistema y reduciendo el tiempo de finalización del conjunto, tal y como se muestra en la Figura 3.1(c).

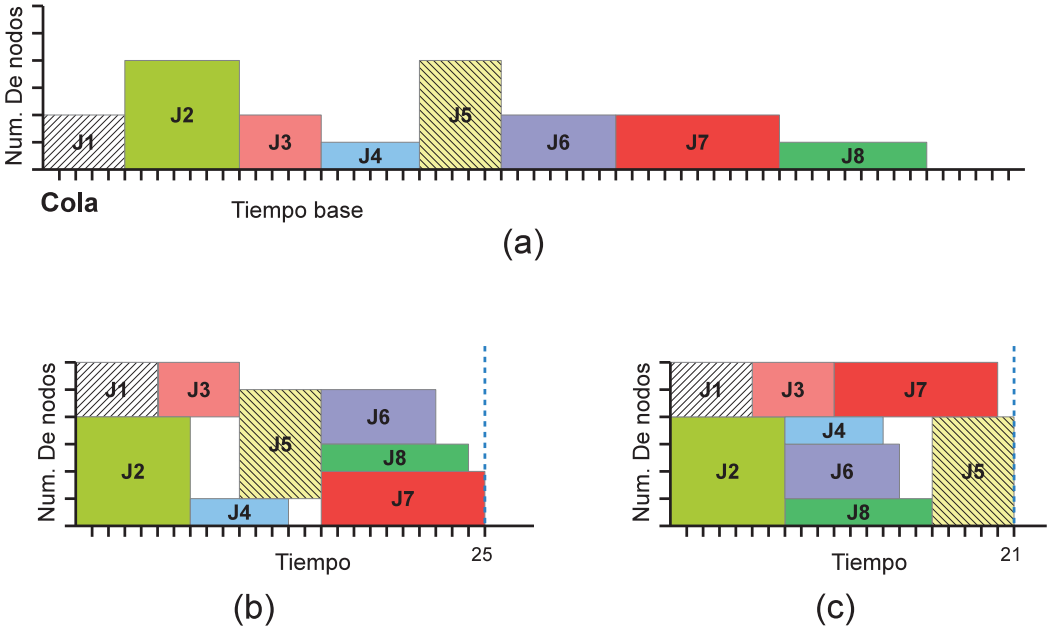


Figura 3.1: Representación de planificación de aplicaciones. (a) Cola de aplicaciones a planificar (b) Asignación FCFS individual de cada aplicación (c) Asignación del conjunto de aplicaciones

Teniendo en cuenta todo lo anteriormente expuesto, en el presente trabajo

de tesis se seguirá un planteamiento de planificación múltiple en el que un conjunto de aplicaciones sean evaluados de forma simultánea, pudiendo modificar el orden de ejecución de las mismas.

En este capítulo se presentan tres técnicas distintas para la planificación de aplicaciones paralelas en entornos Multi-Cluster heterogéneos. Estas técnicas tienen un objetivo común: tratar de minimizar el tiempo de ejecución de un conjunto de aplicaciones en espera en la cola del sistema, y maximizar la utilización de los recursos del entorno Multi-Cluster siempre que sea posible.

Para abordar el problema, la primera de las técnicas propuestas, denominada *Package Allocation Strategy* (PAS) (Sección 3.1), utiliza de forma combinada un conjunto de heurísticas de planificación para la ordenación de las aplicaciones, junto con un modelo MIP para decidir la mejor asignación de recursos. La segunda de las técnicas que se propone, denominada *Ordering and Allocating Strategy* (OAS) (Sección 3.2), se compone de un modelo MIP capaz tanto de determinar el orden de ejecución de las aplicaciones como su asignación de recursos. Por último, se propone una técnica heurística denominada *Minimum Execution Slowdown* (MESD) (Sección 3.3), que trata de mejorar el rendimiento global del conjunto de aplicaciones, reduciendo su tiempo de ejecución y mejorando la utilización de los recursos del sistema. Cada apartado se acompaña de un estudio experimental que evalúa la eficiencia de las propuestas presentadas, así como las conclusiones que se extraen de éste análisis.

A continuación se presenta cada una de las técnicas propuestas, las consideraciones tomadas en cuenta, y los modelos que las sustentan, además de un análisis de su rendimiento comparado con otras técnicas clásicas de la literatura.

3.1. PAS: Package Allocation Strategy

Las diferentes metodologías desarrolladas en este capítulo de la tesis toman como fuente el trabajo desarrollado por Lérida et al. [LSG+08], que comprende un modelo de programación entera mixta, o *Mixed Integer Programming* (MIP) implementado con el software de optimización *LINGO* [Lin], que era capaz de obtener la asignación óptima de una única aplicación paralela en un entorno Multi-Cluster, minimizando el tiempo de ejecución de dicha aplicación. La programación entera consiste en un programa matemático de optimización y la factibilidad en el cual todas o algunas de las variables son enteras, y es un problema *NP-Hard* [BW05]. Es uno de los métodos más utilizados para desarrollar modelos con el fin de obtener una solución óptima a un problema especificado, siendo ampliamente utilizado en el desarrollo de modelos para la resolución de problemas de producción y de transporte [Bal08]. Se trata de un procedimiento matemático que permite resolver un problema formulado mediante ecuaciones lineales, optimizando (maximizando o minimizando) una función objetivo que se encuentre sujeta a una serie de restricciones. Para determinar la solución óptima, un software de solución MIP explora el espacio de soluciones determinado por el problema, y lo va acotando hasta encontrar la solución que minimiza o máxima la función objetivo y que cumple con todas las restricciones especificadas. Cuando todas las variables que intervienen en la función objetivo han de tomar únicamente valores enteros, se dice que el problema es de programación entera. Cuando únicamente algunas variables del problema han de tomar valores enteros, se dirá que el problema es de programación entera mixta [Pin02, Bla02, Bru04].

Para determinar la asignación óptima de la aplicación, el modelo propuesto en [LSG+08] tenía en cuenta tanto los requisitos de procesamiento como de comunicación, descartando aquellas asignaciones en las que se produjera una saturación en los canales de comunicación. Sin embargo, el modelo estaba limitado a la evaluación de una única aplicación a la vez. Esto limita el abanico de posibles decisiones de planificación que se pueden tomar, al no tener en cuenta el efecto que el resto de aplicaciones, en espera de ser asignadas, pueden tener sobre la aplicación que iba a ser asignada.

En la presente sección se propone un nuevo modelo MIP con el fin de mejorar la calidad de las soluciones. Éste modelo MIP se integra en una técnica denominada *Package Allocation Strategy* (PAS), en la que se considera además la utilización de técnicas heurísticas clásicas para la ordenación de las aplicaciones paralelas en la cola del sistema.

La técnica propuesta se compone de dos componentes principales. Primero, una función de selección que se encarga de determinar un conjunto de aplicaciones de la cola de espera del sistema, sobre el que se va a tratar de optimizar su asignación. En segundo lugar, un modelo lineal en el que se optimiza la asignación con el fin de obtener el mejor tiempo de ejecución de todo el bloque de aplicaciones seleccionadas, evitando además la saturación de los canales de comunicación.

En la Figura 3.2 se muestra un diagrama de flujo de la estrategia PAS. La estrategia recibe como entrada la cola de aplicaciones del sistema. La función de selección (a) se encarga de construir un bloque de aplicaciones, cuyos requisitos de cómputo puedan ser satisfechos en el estado actual del entorno Multi-Cluster. Una vez seleccionado un subconjunto de las aplicaciones, se aplica a este el modelo propuesto de asignación MIP (2), el cual se encargará de producir la asignación múltiple para ese conjunto de aplicaciones.

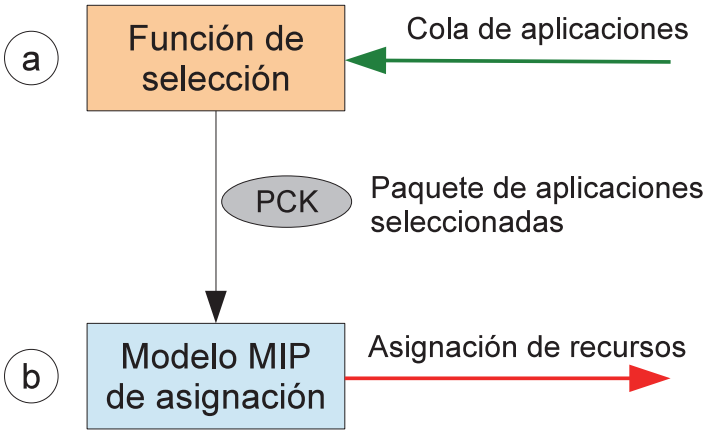


Figura 3.2: Diagrama de flujo de la técnica PAS

Para poder entender mejor el funcionamiento de la estrategia propuesta, en la Figura 3.3 se muestra un ejemplo del funcionamiento de la misma, y se

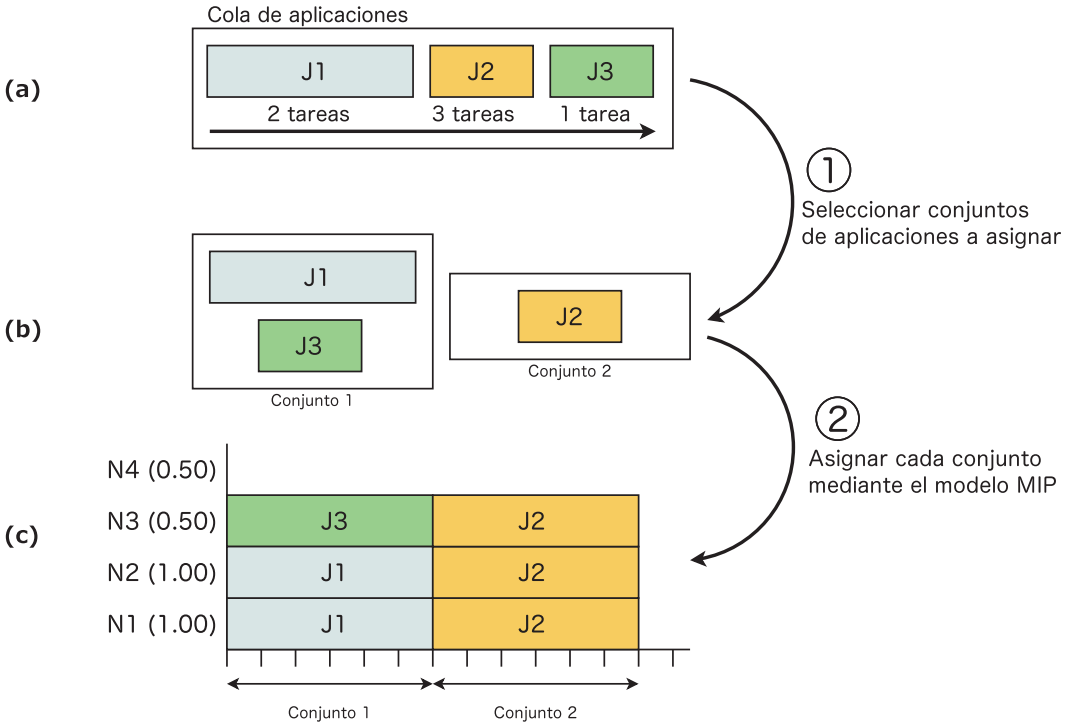


Figura 3.3: Pasos a seguir en el mecanismo de selección y asignación de PAS

ejemplifica la evaluación de un conjunto de aplicaciones. Como se puede observar, disponemos de una cola de aplicaciones en espera con tres aplicaciones de {2,3,1} tareas, respectivamente. El entorno multi-cluster está compuesto por cuatro nodos, con potencias efectivas {1.0, 1.0, 0.5, 0.5 }, respectivamente.

El primer paso que lleva a cabo la estrategia es la selección de las aplicaciones que tendrán que ser asignadas. La función de selección (1) recorre la cola de aplicaciones en espera, y construye un conjunto, o paquete, con aquellas aplicaciones que cumplen un determinado criterio. En el ejemplo de la figura, suponiendo que como criterio seleccionamos las aplicaciones, que están ordenadas por orden de llegada en la cola de espera, cuyo número de tareas pueden ser asignadas en los recursos disponibles. Entonces la función de selección construye el primer paquete de asignación con las aplicaciones *J1* y *J3*.

Una vez que disponemos de un paquete de aplicaciones, este es procesado por el modelo MIP, el cual trata de optimizar la asignación del conjunto en base a la disponibilidad de los recursos y las características del entorno Multi-

Cluster. Este modelo MIP (2) realiza una asignación óptima del paquete de aplicaciones, resultando la asignación de la aplicación *J1* en los nodos {*N1, N2*} y la aplicación *J3* en *N3*.

En el siguiente paso, la función de selección (1) construye el siguiente paquete con la aplicación que queda en la cola, *J2*. El modelo MIP (2) determina la asignación óptima para esta aplicación En el ejemplo, la aplicación de *J2* es asignada a los nodos { *N1, N2, N3* }.

En las secciones 3.1.1 y 3.1.2, la función de selección y el modelo MIP, son presentados en detalle, respectivamente.

3.1.1. Función de selección

Al trabajar con subconjuntos de aplicaciones, la selección de éstas se convierte en un aspecto de gran importancia, ya que la elección de unas aplicaciones u otras, puede hacer variar la asignación final, y por lo tanto, afectar al rendimiento del conjunto. Por ello, se propone la implementación de una función de selección en base a un criterio de selección y disponibilidad de los recursos del sistema, tal como se expresa en la ecuación 3.1.

$$PCK = \mathcal{F}(\mathcal{Q}, \mathcal{R}, \mathcal{C}) \tag{3.1}$$

donde PCK es el conjunto de aplicaciones seleccionadas por la función de selección $\mathcal{F}$. $\mathcal{Q}$ es la cola de aplicaciones del sistema, $\mathcal{R}$ es el conjunto de recursos disponibles del entorno, y $\mathcal{C}$ es el criterio determinado que han de cumplir estos recursos para que un subconjunto de aplicaciones sea seleccionado.

La función de selección $\mathcal{F}$ puede ser especificada siguiendo diversos criterios, como el agrupamiento según el tiempo base de las aplicaciones, el porcentaje de procesamiento respecto a la comunicación (σ_j), la disponibilidad de recursos de cómputo, la capacidad de memoria en los nodos etc. En el presente trabajo no se ha pretendido profundizar en heurísticas complejas para la selección del conjunto de aplicaciones, ya que la idea principal es demostrar que evaluar el conjunto de aplicaciones como una unidad permite obtener mejores resultados que haciéndolo de forma individual. Por este motivo, el criterio utilizado consiste en que las tareas presentes en el conjunto de aplicaciones puedan ser

asignadas en el número de recursos disponibles en el entorno Multi-Cluster. La función de selección, teniendo en cuenta el criterio descrito, es expresada mediante la ecuación 3.2.

$$\exists\ PCK \subseteq \mathcal{Q} \mid \sum_{j \in PCK} \tau_j \leqslant |\mathcal{R}'| \tag{3.2}$$

donde PCK es un subconjunto de aplicaciones de la cola del sistema ($\mathcal{Q}$), donde la suma del número de tareas de cada aplicación (τ_j) no sea superior al número de recursos disponibles en R', que representa el subconjunto de recursos del sistema ($\mathcal{R}' \subseteq \mathcal{R}$) que cumplen el criterio ($\mathcal{C}$), que en este caso son aquellos recursos disponibles, no asignados a otras aplicaciones.

Teniendo en cuenta las consideraciones anteriores, la función de selección propuesta ha sido implementada tal y como se muestra en el Algoritmo 1.

Algoritmo 1 Implementación de la función de selección de PAS

```
1: function SelectJobs(SJ: Conjunto de aplicaciones, n: Número de nodos disponibles)
2:     PCK ← ∅          // Pack de aplicaciones
3:     for all J in SJ do
4:         t ← número de tareas de J
5:         if t ≤ n then          //Si la aplicación cabe
6:             PCK ← PCK + J          //Añadir J al pack
7:             SJ ← SJ − J
8:             n ← n − t
9:         end if// Si la aplicación no cabe, pasar a la siguiente
10:    end for
11:    return PCK
12: end function
```

El algoritmo empieza creando el paquete de aplicaciones, llamado PCK, que inicialmente se encuentra vacío (linea 2). Todas las aplicaciones presentes en la cola del sistema son recorridas, una a una (lineas 3-10). Si la aplicación que se está evaluando actualmente necesita un número menor o igual de nodos a los disponibles en el sistema (linea 5), ésta se añade al conjunto PCK (linea 6), y se elimina de la cola de aplicaciones en espera (linea 7). El número de nodos disponibles del sistema son actualizados (linea 8). En el caso de que la aplicación actual hubiese requerido más recursos de los disponibles,

el algoritmo saltará a la siguiente aplicación y repetirá el proceso, hasta que todas las aplicaciones hayan sido evaluadas. Finalmente, el algoritmo devuelve el conjunto de aplicaciones PCK, que incluye todas aquellas aplicaciones que pueden ser asignadas con los recursos disponibles en la actualidad.

3.1.2. Modelo MIP para la asignación de recursos

El modelo MIP es el encargado de determinar la asignación de las tareas de las aplicaciones paralelas que han sido previamente seleccionadas mediante la función $\mathcal{F}$. La asignación que se obtiene, tiene como objetivo, minimizar el tiempo de ejecución global del conjunto de aplicaciones paralelas, considerando los requisitos de procesamiento y de comunicación de las aplicaciones, en un entorno multi-cluster.

En [EHS+02] se determinó que cierto grado de saturación en los canales de comunicación, en determinadas situaciones, podía ser permitido. Sin embargo, cuando se produce saturación en un canal de comunicación, el efecto que éste tiene sobre el tiempo de ejecución de las aplicaciones es muy difícil de predecir. Teniendo esto en cuenta, el modelo MIP que se propone descarta aquellas soluciones en que se produzca saturación en los canales, siendo de esta forma más restrictivo a la hora de encontrar una solución a la asignación.

En la Figura 3.4 se describe el modelo MIP propuesto, en la que aparecen el conjunto de parámetros del modelo (líneas 1-9), las variables de decisión (líneas 10-13), las restricciones principales (líneas 14-15), y la función objetivo que ha de minimizarse (línea 16).

A continuación se detallan cada uno de ellos.

3.1.2.1. Parámetros del problema

El conjunto de parámetros del modelo está dividido en dos grupos; la caracterización del entorno multi-cluster y los requisitos y características de las aplicaciones paralelas, en base a los modelos propuestos en el Capítulo 2.

El conjunto de parámetros que representan la caracterización del entorno multi-cluster son el conjunto de recursos de procesamiento (R), la potencia efectiva de cada uno de ellos (Γ_r), el conjunto de canales ($\mathcal{L}$), y el ancho de

Parámetros del Modelo

1. R: Conjunto de recursos de procesamiento.
2. $\mathcal{L}$: Conjunto de enlaces inter-cluster.
3. Γ_r: Potencia efectiva del recurso r, $\forall r \in R$
4. MBW_k: Ancho de banda máximo del enlace inter-cluster k, $\forall k \in \mathcal{L}$.
5. J: Conjunto de tareas a asignar
6. τ_j: Número de aplicaciones que componen cada aplicación j, $\forall j \in J$.
7. Tb_j: Tiempo base de la aplicación j, $\forall j \in J$.
8. $PTBW_j$: ancho de banda requerido por cada tarea de j, $\forall j \in J$
9. σ_j: factor de ponderación del procesamiento respecto de la comunicación, para la aplicación j, $\forall j \in J$.

Variables de decisión

10. $X_{(j,r)} = 1$ si j es asignada al recurso r, $\forall j \in J, \forall r \in R$
11. SP_j es el retraso de procesamiento de la aplicación j, $\forall j \in J$
12. $BW_{(j,k)}$: Ancho de banda consumido por la aplicación j en el canal k. $\forall j \in J, \forall k \in \mathcal{L}$
13. ABW_k: Ancho de banda disponible en el canal k, $\forall k \in \mathcal{L}$

Restricciones

14. Gang Matching
15. Evitar la saturación de los canales de comunicación

Función Objetivo

16. Minimizar el tiempo de ejecución global del conjunto de aplicaciones

Figura 3.4: Especificación del modelo MIP de PAS.

banda máximo disponible de cada uno (MBW_k).

Los parámetros que representa los requisitos y características de cada una de las aplicaciones paralelas del conjunto (J) son, el número de tareas que las componen (τ_j), el tiempo base (Tb_j), la proporción del procesamiento respecto a la comunicación (σ_j), y el ancho de banda requerido por tarea ($PTBW_j$).

3.1.2.2. Variables de decisión

Las variables de decisión son las variables que se devuelven como resultado del problema. El conjunto de variables se divide en dos grupos; las variables relativas a la asignación de las aplicaciones paralelas, y las variables relativas al estado de los canales de comunicación del multi-cluster.

La asignación de una aplicación a un recurso concreto se determina mediante una variable binaria($X_{(j,r)}$). Esta variable toma 1 como valor cuando una tarea de la aplicación j es asignada al recurso r. En cualquier otro caso, el valor de la variable será 0. El retraso producido por los recursos de procesamiento (SP_j) será utilizado para determinar el tiempo de ejecución (Te_j) de cada aplicación, expresado como:

$$Te_j = Tbj \times (\sigma_j \cdot SP_j + (1 - \sigma_j)), \quad \forall j \in J \tag{3.3}$$

Siendo SP_j expresado como:

$$SP_j = \max_{r \in R}((\Gamma_r)^{-1} \cdot X_{(j,r)}). \quad \forall j \in J \tag{3.4}$$

Para determinar el estado de los canales de comunicación del multi-cluster, se calcula el ancho de banda disponible (ABWk) en cada canal k, según las asignaciones para cada una de las aplicaciones. Para poder calcular este ancho de banda disponible, es necesario calcular el ancho de banda consumido por cada aplicación sobre cada canal de comunicación ($BW_{(j,k)}$), según la ecuación 2.6, definida en la Sección 2.2.2.

3.1.2.3. Restricciones

La restricción 3.5 garantiza el *Gang-Matching*, es decir, que todas las tareas de las aplicaciones paralelas son asignadas.

$$\sum_{j \in J, r \in R} (X_{(j,r)}) = \tau_j, \quad \forall j \in J \tag{3.5}$$

La ecuación 3.6 define el ancho de banda disponible en el canal k una vez asignadas las aplicaciones a sus respectivos recursos, y teniendo en cuenta el uso de los canales de comunicación, en el caso de haber realizado co-asignación.

$$ABW_k = MBW_k - \sum_{\forall j \in J} BW_{(j,k)}, \quad \forall j, k \tag{3.6}$$

La restricción 3.7 asegura que ninguna solución factible producirá saturación en los canales de comunicación inter-cluster. Esto se realiza asegurando que el ancho de banda disponible en un canal k, una vez realizadas las asignaciones de las aplicaciones, será mayor o igual a 0.

$$ABW_k \geq 0 \quad \forall k, t \tag{3.7}$$

Esta misma restricción se podría modificar incluso añadiendo un margen de saturación permitido con el fin de relajarla, tal como algunos autores afirman [EHS+02].

3.1.2.4. Función objetivo

Siguiendo el planteamiento propuesto en el modelo MIP del que se parte [LSG+08], donde se minimizaba el tiempo de ejecución de una aplicación, se ha rediseñado la función objetivo del modelo MIP propuesto con el fin de minimizar el tiempo de ejecución global del conjunto de aplicaciones que se evalúen, según la ecuación 3.8.

$$minimize\{ \quad \sum_{j \in J} Tb_j \cdot ct_j\} \tag{3.8}$$

donde $Tb_j \cdot ct_j$ representa el tiempo de ejecución de cada aplicación j, a

partir del tiempo base obtenido en un entorno dedicado y el factor de coste debido a los recursos y canales del entorno asignados, tal y como se detalla en la Sección 2.2. De esta forma, al minimizar el sumatorio de los tiempos de ejecución, no se está priorizando ninguna aplicación en concreto. Esta característica puede dar lugar a que una aplicación sufriera un retraso si con ello se consigue obtener el mínimo tiempo de ejecución global para el conjunto de aplicaciones.

3.1.3. Experimentación

En esta sección se desarrolla un estudio experimental detallado sobre el comportamiento de la estrategia *PAS*, y su efectividad a la hora de planificar aplicaciones paralelas de forma simultánea. Este análisis se divide en dos partes:

- Evaluación del rendimiento del mecanismo de asignación de recursos, en diferentes condiciones de heterogeneidad tanto de los recursos de procesamiento como de comunicación. Para evaluar el rendimiento del mecanismo de asignación de recursos, los resultados obtenidos por *PAS* son comparados con los obtenidos por otras estrategias de asignación presentes en la literatura. Este estudio se realiza sobre diferentes condiciones de los recursos de procesamiento y de comunicación, con el fin de evaluar el impacto que el estado del multi-cluster tiene sobre la calidad de las soluciones. Este estudio se presenta en la Sección 3.1.3.2.

- Evaluación del comportamiento de la función de selección de aplicaciones. En la la Sección 3.1.3.3 se presenta la evaluación del rendimiento de la función de selección se realiza comparando los resultados obtenidos por *PAS* aplicando las estrategias clásicas de selección de aplicaciones presentes en la literatura.

A continuación se presenta el entorno de experimentación utilizado.

3.1.3.1. Entorno de experimentación

El modelo MIP de la estrategia *PAS* se ha implementado usando el solver de programación lineal *CPLEX* [IBM], y las soluciones producidas, han sido aplicadas en el framework de simulación *GridSim* [TGP], caracterizado para trabajar como simulador de entornos multi-cluster.

Se ha definido un entorno Multi-Cluster heterogéneo compuesto por 4 clusters, formados cada uno por $\{24, 16, 16, 8\}$ nodos, respectivamente. Esta configuración ha sido escogida para garantizar una mayor heterogeneidad del entorno, al conformarse el multi-cluster en clusters con diferente número de recursos de cómputo.

En la evaluación del mecanismo de asignación, se analizan los resultados obtenidos por diferentes estrategias bajo diversas condiciones del entorno. Estas condiciones consisten en tres niveles de heterogeneidad de los recursos de procesamiento, y en diferentes niveles de carga de las comunicaciones. Se ha definido el *grado de heterogeneidad (H)* como la diferencia, en porcentaje, de potencia efectiva entre los respectivos clusters, teniendo en cuenta que los nodos dentro de un mismo cluster son homogéneos. De esta forma, se han definido tres niveles de heterogeneidad de los recursos de procesamiento:

- H=10 %. Bajo nivel de heterogeneidad, es decir los recursos de cada cluster son prácticamente idénticos.
- H=50 %. Nivel medio de heterogeneidad, donde hay bastante diferencia entre la potencia efectiva de los diferentes clusters.
- H=90 %. Nivel elevado de heterogeneidad, donde la diferencia entre la potencia efectiva de los diferentes clusters es elevada.

La carga de las comunicaciones se ha determinado mediante el uso de un parámetro conocido como *Bisection Bandwidth (BSBW)* [JAA07]. Este parámetro, medido en Gigabytes/segundo (GB/s), evalúa el efecto de la comunicación entre dos mitades en un sistema de comunicación, y determina el rendimiento del peor caso posible en una red, ya que está relacionada con el costo de mover datos de un lado del sistema a otro. Este parámetro ajusta los

requisitos de comunicación de las aplicaciones, de forma que un valor elevado del mismo repercutirá en aplicaciones más comunicativas, y por lo tanto, los canales de comunicación del sistema tendrán una carga más elevada. Se han tomado valores en el rango 0.1GB/s a 1.1GB/s, desde un bajo coste/carga de comunicación a un nivel alto de carga, en intervalos de 0.1GB/s.

En la Tabla 3.1 se muestra el rango en las potencias efectivas utilizadas para modelar la heterogeneidad. Como se puede observar, en algunos casos se utilizan potencias efectivas mayores a 1, lo que significa que una aplicación podría ejecutarse más rápido que en su tiempo de referencia, denotado por el tiempo base (Tb_j). Para cada uno de los niveles de heterogeneidad, se ha evaluado el efecto de diferentes grados de carga en las comunicaciones mediante el parámetro BSBW, en el rango de 0.1GB/s (baja carga) a 1.1GB/s (carga alta), en intervalos de 0.1GB/s.

Grado de heterogeneidad	Poténcia efectiva más baja	Poténcia efectiva más alta
10 %	0.9	1.0
50 %	0.8	1.2
90 %	0.6	1.4

Tabla 3.1: Caracterización del grado de heterogeneidad

Para definir el workload a utilizar se han tomado como referencia varios estudios de la literatura. En [LGW05] se presentó una detallada caracterización de un entorno Multi-Cluster, donde se utilizaban diferentes funciones de distribución para representar las características y comportamiento de las aplicaciones que se ejecutaban en algunos multi-clusters de producción. En [LF03] se concluye que es habitual la prevalencia de aplicaciones paralelas con un número de tareas que es potencia de dos. Finalmente, en [JLPS05] se caracterizan los requisitos de comunicación, en función del parámetro BSBW.

En la Tabla 3.2 se muestran los valores que caracterizan las aplicaciones paralelas que se han utilizado durante la experimentación, obtenidos de los estudios de caracterización presentados previamente. En la Figura 3.5 se representan las funciones de distribución utilizadas para caracterizar el tiempo entre llegada de aplicaciones, el histograma con la proporción de aplicaciones

en función de su número de tareas, y la función de distribución del tiempo base de las aplicaciones.

Número de aplicaciones	100
Tiempo base promedio	170 s
Tiempo entre llegadas	$weibull(\lambda = 82{,}6, k = 0{,}6)$
Tamaño de las aplicaciones	$gamma(\alpha = 4{,}04, \beta = 0{,}77)$ prevalencia de "potencias de dos"
Tiempo base	$weibull(\lambda = 200, k = 1)$
Requisitos de comunicación	$PNBW^j = BSBW\left(\frac{4(n_T^j - 1)}{(n_T^j)^2}\right)$

Tabla 3.2: Caracterización del workload

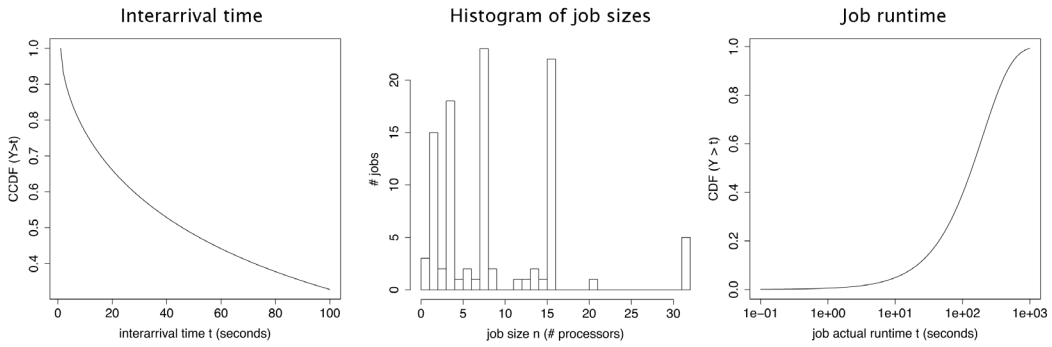


Figura 3.5: Características del workload

Para la comparación de resultados del rendimiento del mecanismo de asignación de recursos, se han utilizado dos estrategias de asignación presentes en la literatura:

1. *Chunk Big Small (CBS)*, es una técnica presentada por W. Jones en [JLPS05], tiene como objetivo principal reducir la utilización de los canales de comunicación. Para conseguirlo, la estrategia intenta agrupar las tareas de cada aplicación en un "chunk", o conjunto, de por lo menos el 75 % de las tareas, en un mismo cluster, de forma que las comunicaciones externas sean reducidas. La selección de los recursos de procesamiento no es importante para esta estrategia. Esto resulta de gran interés en este

estudio, ya que se podrá comparar *PAS* con una técnica que únicamente intenta reducir el uso de comunicaciones.

2. *Job Preference on Resources (JPR)*, es una técnica presentada por V. Naik en [NLYW05], que asigna las aplicaciones paralelas en los mejores recursos, co-asignando cuando es necesario. En su forma original, la estrategia puede ser utilizada para asignar seleccionando los mejores recursos de procesamiento, o los mejores recursos de comunicación, pero no teniendo ambos criterios en cuenta al mismo tiempo. En el presente estudio, se ha configurado la estrategia para que considere los mejores recursos de procesamiento, co-asignando en aquellas situaciones en que sea necesario, pero sin tener en cuenta las capacidades de los recursos de comunicación. Se ha decidido no utilizar la versión de comunicación de JPR por ser considerada esta opción por la técnica CBS.

Ambas técnicas han sido implementadas sobre un modelo MIP con el solver *CPLEX*, de forma que *CBS* proporcionará una asignación óptima que agrupe las tareas al máximo, y *JPR* proporcionará una asignación óptima de forma que utilice los mejores recursos de procesamiento disponibles.

En los diferentes experimentos, los resultados son evaluados mediante dos métricas ampliamente utilizadas en la literatura:

- *Makespan*: Mide el tiempo que ha sido necesario para completar la ejecución del conjunto completo de aplicaciones.
- *Tiempo de respuesta*: Se utiliza para medir el tiempo promedio que las aplicaciones pasan en el entorno hasta completar su ejecución. Sus componentes son el tiempo de ejecución y el tiempo de espera, promedios.

3.1.3.2. Evaluación de la asignación de recursos

En esta sección se evalúa el rendimiento del método de asignación de recursos de *PAS*.

El primer estudio experimental hace referencia a la comparación de los resultados obtenidos por la estrategia PAS con los obtenidos por las estrategias CBS y JPR. Para este primer estudio, la función de selección de PAS ha sido

fijada se ha fijado para funcionar en la forma de FCFS, para forzar que las tres estrategias se comparen en igualdad de condiciones. Luego, los resultados de las tres estrategias comparadas se muestran en función de las diferentes métricas que se han evaluado: makespan y tiempo de respuesta promedio, expresados en segundos. Por último se realiza una comparación del número de aplicaciones que han sido co-asignadas por cada estrategia. Las diferentes figuras siguen el mismo patrón para representar los resultados. Se divide en los tres casos de heterogeneidad: 10 %, 50 % y 90 %, y para cada uno de estos casos, se muestran los resultados en función de la carga de comunicación establecida por el parámetro BSBW.

En la Figura 3.6 se evalúa el makespan y el tiempo de respuesta con un grado de heterogeneidad de los recursos de procesamiento más bajo: un 10 %. Esto significa que se puede considerar que el entorno es prácticamente homogéneo. En estos casos, la habilidad de una estrategia para determinar la asignación de las aplicaciones en función de los recursos de procesamiento no es tan importante, puesto que al ser prácticamente homogéneos, las mejoras o pérdidas que puedan resultar de ello son mínimas. En un sistema con estas características, las diferencias más significativas las marcará el uso de los canales de comunicación.

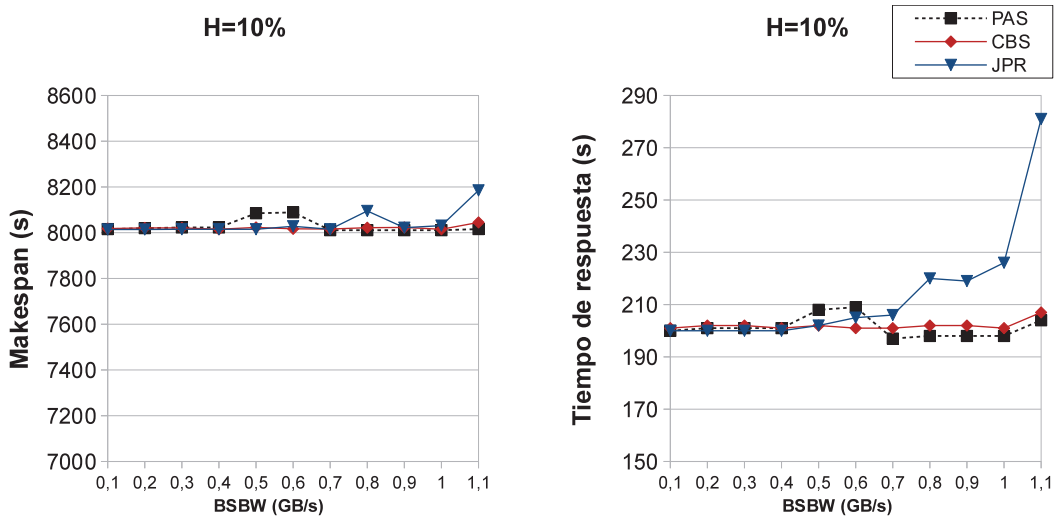


Figura 3.6: Comparativa de makespan y tiempo de respuesta con una heterogeneidad del 10 %

Cuando la carga de comunicaciones (BSBW) es inferior a 0.6GB/s, el ancho de banda disponible en los canales es suficiente para las comunicaciones de todas las aplicaciones, y por lo tanto, las tres estrategias han obtenido resultados muy similares. Sin embargo, cuando esta carga aumenta, a partir de 0.7GB/s, el ancho de banda disponible para las aplicaciones se ha reducido, y por lo tanto, estas se ven afectadas en mayor medida. En estos casos, *JPR* es incapaz de determinar una asignación adecuada, al tener en cuenta únicamente las características de procesamiento de los recursos para determinar la asignación. Este efecto se puede observar principalmente en el tiempo de respuesta promedio, que crece de forma significativa, sobre todo en los casos con una carga de comunicación muy elevada (BSBW $\approx$ 1.1GB/s). En el caso de las estrategias *PAS* y *CBS*, sus resultados se mantienen muy estables tanto en el makespan como en el tiempo de respuesta promedio, sin verse afectados por el incremento en la carga de las comunicaciones. Esto se debe a que ambas estrategias tienen en cuenta las comunicaciones a la hora de determinar las asignaciones. *PAS* lo hace evitando de forma explícita la saturación de los canales, y *CBS* agrupando las tareas en un único cluster para reducir la utilización de los canales.

En los casos con alta heterogeneidad de los recursos de procesamiento (50 % y 90 %), mostrados en las figuras 3.7 y 3.8, respectivamente, las potencias efectivas de los clusters pasan a jugar un papel mucho más importante. Esto se debe a que estos clusters ahora son más heterogéneos entre ellos, y el hecho de asignar una aplicación existiendo diferencias entre los recursos, puede repercutir de forma muy importante en su tiempo de ejecución. Para valores de carga de comunicación inferiores a 0.6GB/s, *PAS* y *JPR* son capaces de sacar provecho de la heterogeneidad de los recursos de procesamiento, y obtienen unos resultados muy similares, tanto en makespan como en tiempo de respuesta promedio. Por contra, *CBS* obtiene peores resultados, siendo estos más claramente visibles en la comparativa del tiempo de respuesta promedio, y para un alto grado de heterogeneidad. Esto se debe a que únicamente tiene en cuenta la agrupación de las tareas en un mismo cluster a la hora de realizar las asignaciones, sin tener en cuenta las capacidades de procesamiento de los recursos.

Cuando la carga de las comunicaciones crece por encima de los 0.6GB/s, se puede observar como los resultados de *JPR* empeoran, debido a que no considera el estado de los canales de comunicación, mientras que *CBS* obtiene un comportamiento bastante estable, ya que sigue asignando las aplicaciones tratando de agrupar las tareas de un mismo cluster. Sin embargo, sus resultados no son óptimos, al no tener en cuenta el alto grado de heterogeneidad de los recursos de procesamiento. Por contra, *PAS* consigue obtener los mejores resultados, incluso para altos niveles de carga en las comunicaciones. Su capacidad para considerar tanto el procesamiento como las comunicaciones, permiten que pueda adaptar las asignaciones de forma más inteligente viéndose menos afectado por los diferentes grados de heterogeneidad en los recursos de procesamiento y los diferentes niveles de carga de comunicaciones.

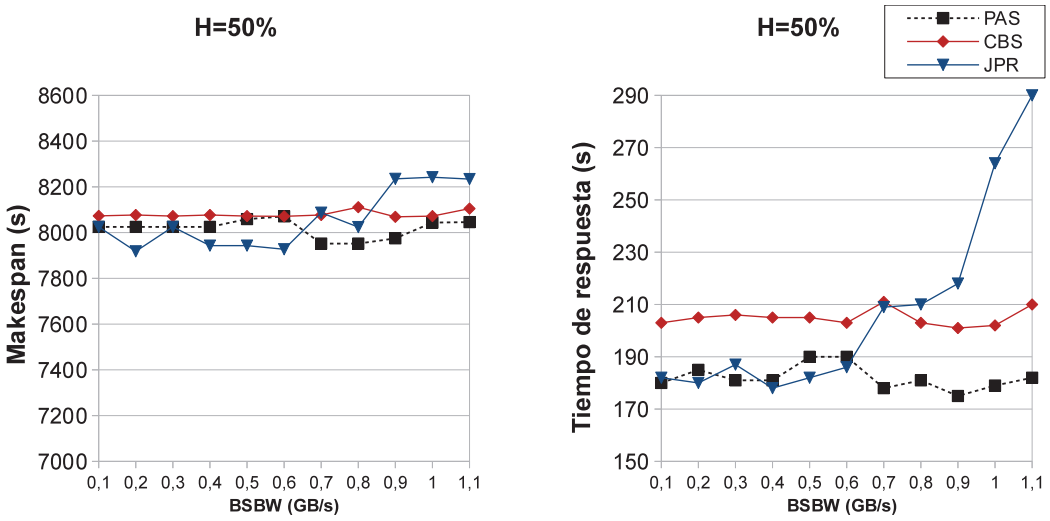


Figura 3.7: Comparativa de makespan y tiempo de respuesta con una heterogeneidad del 50 %

Con el fin de evaluar el comportamiento de las tres técnicas en cuanto al tratamiento de la red, hemos analizado el efecto de cómo las condiciones del entorno influye sobre el número de aplicaciones paralelas co-asignadas. En la Figura 3.9 se muestran los resultados de esta comparativa, para los tres grados de heterogeneidad, y para diferentes niveles de carga de las comunicaciones. Hay que tener en cuenta que únicamente aquellas aplicaciones que son co-asignadas, comunican a través de varios clusters, y pueden ver afec-

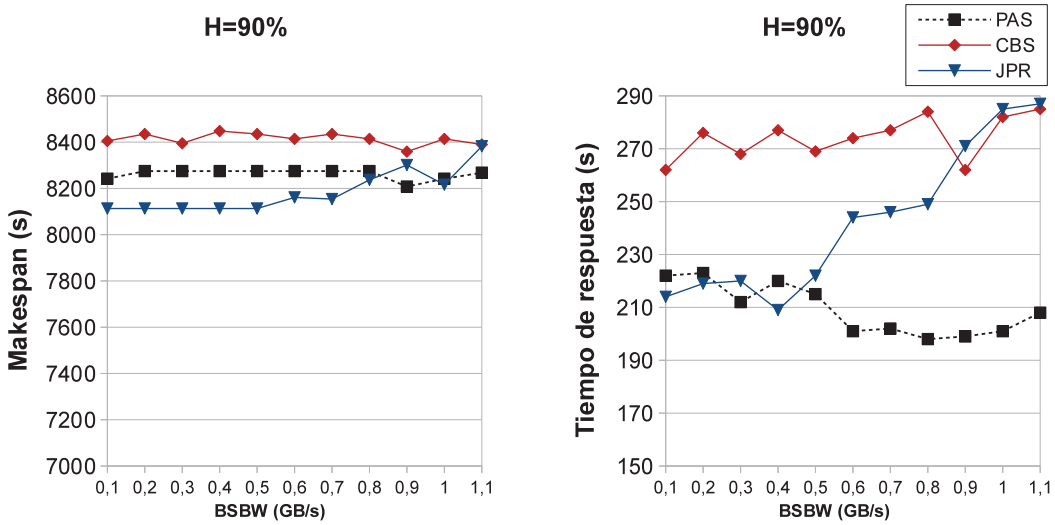


Figura 3.8: Comparativa de makespan y tiempo de respuesta con una heterogeneidad del 90 %

tado su tiempo de ejecución por el efecto de la saturación de los canales de comunicación.

A priori parece que reducir al mínimo posible el número de aplicaciones que se co-asignan es la solución ideal. Sin embargo, hay situaciones en las que puede resultar positivo permitir un nivel de co-asignación más elevado para poder aprovechar las potencias efectivas de diferentes clusters y así poder mejorar el tiempo de ejecución de las aplicaciones, así como la utilización de los recursos, especialmente cuando la carga de los canales de comunicación no es muy elevada.

En la Figura 3.9 se muestra el grado de co-asignación obtenido para las diferentes técnicas. Como se puede observar, *JPR* es la estrategia que ha obtenido un grado de co-asignación más elevado, en torno al 60 % de las aplicaciones. Esto se debe a que la estrategia trata de utilizar los recursos de procesamiento más potentes, sin considerar las comunicaciones, y por lo tanto, a la cantidad de aplicaciones que ha co-asignado. En el otro extremo, *CBS* es la estrategia que ha tenido un grado de co-asiganción más reducido, en torno al 14 % de las aplicaciones. Esto se debe a que trata de agrupar las tareas de las aplicaciones en un mismo cluster, reduciendo así la comunicación entre clusters. La estrategia *PAS* , debido a su capacidad de tener en cuenta tanto el procesamiento

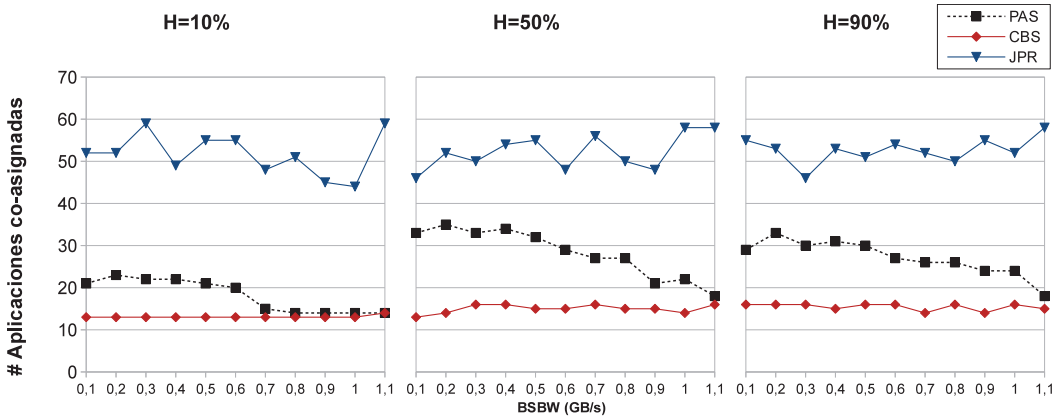


Figura 3.9: Comparativa del grado de co-asignación

como las comunicaciones, ajusta el nivel de co-asignación en función del estado del entorno multi-cluster. En casos donde la heterogeneidad de los recursos de procesamiento es elevada y la carga de las comunicaciones reducida, co-asigna más aplicaciones para aprovechar las potencias efectivas de los nodos de diferentes clusters y mejorar el tiempo de ejecución de estas aplicaciones. En cambio, cuando el nivel de carga de las comunicaciones es elevada, realiza una menor co-asignación de aplicaciones, con el objetivo de minimizar las comunicaciones entre clusters, y evitar que las aplicaciones puedan ser afectadas por la saturación en los canales.

3.1.3.3. Evaluación de la función de selección

Una vez evaluado el rendimiento del método de asignación múltiple de *PAS*, en esta sección se evalúa el efecto que diferentes técnicas de selección de aplicaciones tienen sobre la calidad de las soluciones proporcionadas por la estrategia:

1. *First Come First Served (FCFS)*. Selecciona las aplicaciones en el orden en que estas llegan al sistema.

2. *Short Jobs First (SJF)*. Selecciona las aplicaciones en función de su número de tareas, de menor a mayor.

3. *Big Jobs First (BJF)*. Selecciona las aplicaciones en función de su número

de tareas, de mayor a menor.

4. *Fit Processor First Served (FPFS).* Sigue el esquema de FCFS en la cola de espera, pero seleccionando aquellas aplicaciones que quepan en los recursos libres del sistema.

5. *Short Processing Time (SPT).* Selecciona las aplicaciones en función de su tiempo base, de menor a mayor.

6. *Long Processing Time (LPT).* Selecciona las aplicaciones en función de su tiempo base, de mayor a menor.

Para esta evaluación se ha utilizado el mismo entorno de experimentación que el usado en la sección previa, con la excepción de que la carga de las comunicaciones se ha fijado a un valor de BSBW igual a 0.7GB/s. Este valor representa un término medio en el que las comunicaciones ya tienen un efecto sobre el tiempo de ejecución de las aplicaciones, pero sin ser el factor predominante, como sí ocurriría con valores más elevados. Además, se comparan los resultados de los diferentes métodos de selección en tres condiciones de heterogeneidad distintas: 10 %, 50 % y 90 %.

En la Figura 3.10 se muestran los resultados en cuanto al tiempo de respuesta promedio de las aplicaciones, dividido en sus componentes: tiempo de espera y tiempo de ejecución, con el fin de evaluar por separado cada uno de estos factores. En el eje horizontal se representan los diferentes criterios de selección que se han aplicado en la estrategia *PAS* para cada configuración de heterogeneidad, y en el eje vertical se representa el tiempo de respuesta promedio, medido en segundos. Como puede observarse, el tiempo de ejecución promedio es prácticamente idéntico para todos los métodos de selección. Esto se debe a que todos los métodos tienen como base el mismo mecanismo de asignación de recursos. que como se ha observado en la Sección 3.1.3.2, es capaz de asignar de forma óptima, consiguiendo buenos resultados.

Es en los tiempos de espera donde se observan grandes diferencias entre los diferentes métodos de selección. En este sentido, ningún método de selección obtiene claramente los mejores resultados para todos los casos. Por ejemplo, *PAS-FPFS* es el método que obtiene un tiempo de respuesta más bajo en el

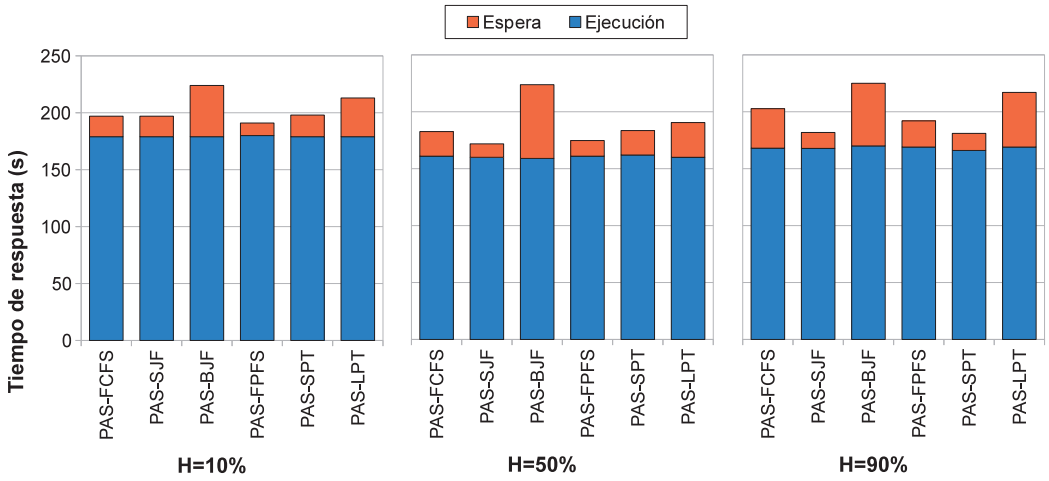


Figura 3.10: Comparativa del tiempo de respuesta promedio

caso de heterogeneidad del 10 %, pero en cambio en el caso del 90 % obtiene unos resultados medios, siendo claramente superada por *PAS-SPT* y *PAS-SJF*. Otras situaciones similares pueden observarse con otras estrategias en los tres casos de heterogeneidad. estrategias como *PAS-BJF* y *PAS-LPT* no obtienen en general buenos resultados en ninguno de los diferentes casos de estudio. Esto se debe a que priorizan las aplicaciones grandes: la primera, por número de tareas, y la segunda por tiempo base. Esto provoca que al ocupar un elevado número de recursos, o ocuparlos durante un tiempo elevado, otras aplicaciones tengan que esperar para poder iniciar su ejecución, lo que incrementa el tiempo de respuesta promedio.

También resulta interesante el efecto que las diferentes técnicas de ordenación pueden tener sobre el tiempo que ha tardado en ejecutarse el workload, ya que puede evidenciar como las diferentes técnicas aprovechan los recursos a lo largo de la ejecución de todo el workload. En la Figura 3.11 se muestra los resultados relativos al makespan. Como se puede observar, las diferencias entre los diferentes métodos es muy pequeña en cuanto al makespan, pero se produce un comportamiento muy similar al observado en la evaluación del tiempo de respuesta promedio: ninguno de los métodos comparados es claramente mejor que los otros. Métodos que obtienen buenos resultados en un caso de estudio, sin embargo, obtienen peores resultados en otro de los casos. Por ejemplo, se puede observar como *PAS-SJF* es el método con un makespan más reducido

en el caso de heterogeneidad del 10 %, obtiene buenos resultados en el caso del 50 %, pero sin embargo, en el caso del 90 %, obtiene peores resultados.

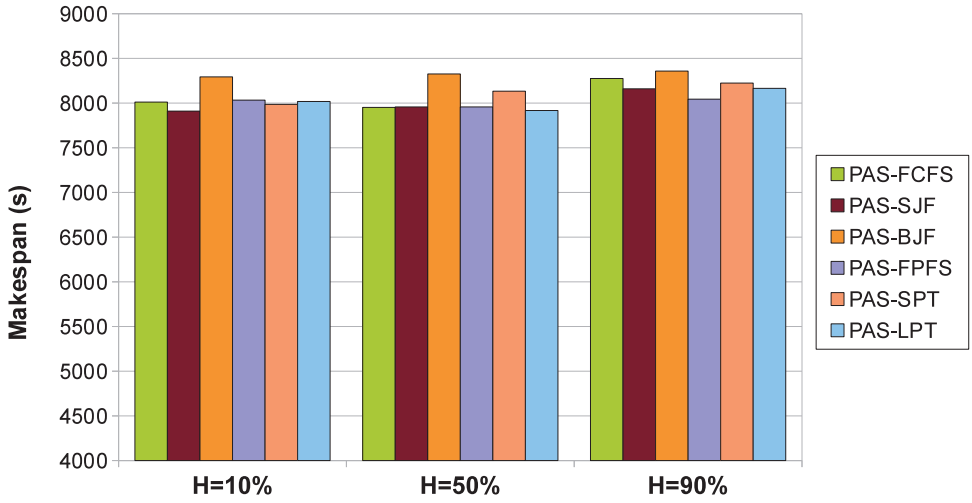


Figura 3.11: Comparativa de makespan

Teniendo en cuenta los resultados analizados, podemos concluir que la elección de un método de selección tiene un impacto significativo en el rendimiento de las aplicaciones, pero que a su vez la elección de un método de selección adecuado es complicada, ya que el método que obtiene buenos resultados en un caso, puede obtener resultados peores en otro.

3.1.4. Conclusiones

Se ha presentado una estrategia que se compone de dos pasos: la selección de un conjunto de aplicaciones para ser asignadas, y la asignación óptima a los recursos de ese conjunto.

La función de selección de aplicaciones permite la utilización de diferentes técnicas clásicas de ordenación, como FCFS, SJF, FPFS ...etc. El modelo MIP es utilizado para determinar la asignación óptima del conjunto de aplicaciones en los recursos disponibles del Multi-Cluster.

En los resultados experimentales se ha observado la importancia del efecto de la heterogeneidad de los recursos de procesamiento y de la carga de las comunicaciones en el proceso de la asignación de aplicaciones. La capacidad

de *PAS* de tener en cuenta tanto las características de los recursos de procesamiento como de comunicaciones, ha permitido la mejora del makespan y del tiempo de respuesta promedio de las aplicaciones, independientemente de las condiciones del entorno Multi-Cluster. Se ha realizado además, un estudio del grado de co-asignación, para evaluar cómo las diferentes estrategias se adaptaban a diferentes niveles de carga de las comunicaciones. *PAS* ajusta la cantidad de aplicaciones que ha co-asignado en función del estado de las comunicaciones y de las capacidades de los recursos de procesamiento, para evitar la saturación de los canales de comunicación, a la vez que se intentaba utilizar los mejores recursos para mejorar el rendimiento de las aplicaciones, y mejorar la utilización de los recursos según la carga del sistema.

La selección de las aplicaciones para ser evaluadas por el modelo MIP ha demostrado ser un factor crítico, al no haberse podido determinar un método de selección claramente mejor que los otros. Métodos que obtenían buenos resultados en unas condiciones, obtenían peores resultados en otras. Esto es debido a que se evalúan de forma simultánea únicamente aquellas aplicaciones que caben en los recursos disponibles del sistema, lo que impide que se puedan explotar todas las posibles opciones de planificación. Esto puede provocar que aplicaciones que se encuentran en la cola de espera sean asignadas sin tener en cuenta las características del resto de aplicaciones, y que una vez que estas otras aplicaciones se asignen no lo hagan a los recursos que permitirían mejorar el rendimiento global del conjunto. Esto abre nuevas líneas de investigación, de cara a estudiar técnicas que proporcionen una selección de aplicaciones que mejore el rendimiento de las mismas, y un mejore aprovechamiento de los recursos del sistema Multi-Cluster.

Atendiendo a las consideraciones anteriores, consideramos que es posible mejorar el rendimiento global de las aplicaciones si estas son consideradas como un todo, evaluando no solo un conjunto de aplicaciones presentes en la cola de espera, sino todas ellas de forma simultánea. Para poder hacerlo, la solución pasa por añadir al método la capacidad de seleccionar y ordenar estas aplicaciones, de forma que puedan evaluarse todas las situaciones en que las aplicaciones estarán ejecutándose en el sistema Multi-Cluster al mismo tiempo, y sacar provecho de la capacidad de procesamiento de los nodos.

3.2. OAS: Ordering and Allocation Strategy

En la Sección 3.1 hemos presentado una estrategia de planificación, *Package Allocation Strategy (PAS)*, basada en un modelo MIP para la asignación de recursos y una función de selección donde podemos implementar heurísticas basadas en diferentes criterios para la selección y empaquetamiento de aplicaciones. Esta estrategia es capaz de determinar la asignación óptima de un conjunto de aplicaciones simultáneamente, de forma que se minimice el tiempo de ejecución global de las mismas. Para tomar las decisiones de asignación, trata de minimizar el tiempo de ejecución de un conjunto de aplicaciones considerando el estado de los recursos tanto de procesamiento como de comunicaciones, evitando aquellas asignaciones que pueden saturar los canales de comunicación. Para determinar el conjunto de aplicaciones a evaluar, se sirve de una función de selección, que puede implementar diferentes criterios. Se ha comprobado de forma experimental que la estrategia obtiene buenos resultados comparada con otras estrategias de asignación de la literatura. Sin embargo, los resultados han evidenciado que la elección de un criterio para la función de selección de aplicaciones tiene un impacto significativo sobre la calidad de las soluciones, observándose además que ninguna de las heurísticas de selección implementadas ha obtenido mejores resultados que las demás.

En la presente sección se propone una estrategia de planificación *Ordering and Allocation Strategy (OAS)*, cuyo objetivo es el de evitar la necesidad de realizar una selección previa de las aplicaciones a ser tratadas, tal y como se daba en la estrategia PAS. La estrategia OAS también se basa en un modelo MIP, pero con la capacidad de evaluar de forma conjunta un número de aplicaciones que puede abarcar incluso toda la cola de espera. Para ello, y teniendo en cuenta que los recursos de procesamiento son finitos, OAS es capaz de definir el orden de ejecución de las aplicaciones. El hecho de evaluar toda la cola de espera del sistema permite considerar las interacciones entre todas las aplicaciones, y tomar las mejores. Este nuevo modelo, al igual que el anterior, considera tanto los requisitos de procesamiento como de comunicación de las aplicaciones, y se descartan aquellas soluciones que pueden provocar la saturación de los canales de comunicación. El uso de un modelo MIP de una

elevada complejidad, como éste, permite obtener soluciones muy próximas al óptimo, lo que permite evaluar la calidad de las soluciones proporcionadas por otras estrategias, actuando por lo tanto como referencia.

La gran diferencia de OAS respecto de la estrategia PAS es que no está limitado al número de aplicaciones que pueden ser tratadas, obviando por lo tanto la necesidad de realizar una selección previa. Para poder hacerlo, y teniendo presente que hay un número finito de recursos, OAS debe establecer el orden en que las aplicaciones acceden a estos recursos, definiendo de esta forma su orden de ejecución.

Para poder definir este orden de ejecución de las aplicaciones, se debe establecer una forma de discretizar el tiempo, que determine si un recurso está o no ocupado en un momento determinado. Para ello se ha definido el concepto de "slot". Un slot es un periodo de tiempo, con un tamaño predeterminado. Así, en un slot se puede determinar si un nodo de procesamiento esta asignado a la ejecución de una tarea de una aplicación paralela, o si por el contrario se encuentra ocioso. Desde el punto de vista de las aplicaciones paralelas, éstas, cuando son asignadas para su ejecución en el entorno Multi-Cluster, tienen un conjunto de slots asignados de forma consecutiva. El momento de inicio de la aplicación se identifica por el momento de inicio del primer slot que se le asigna. Por el contrario, el momento de finalización puede venir dado por cualquier instante de tiempo presente en el último slot asignado. Mediante la comparación del conjunto de slots de diferentes aplicaciones, se puede determinar además qué recursos ocupar, qué enlaces de comunicación comparten y durante qué periodos de tiempo. Esto es de gran importancia para poder determinar los retrasos producidos por la comunicación, así como evaluar existencia o no de saturación del canal, en el caso de que dos o más aplicaciones hicieran uso de los canales entre clusters en un mismo slot.

Un factor muy importante a la hora de usar la discretización del tiempo mediante el uso de slots es el tamaño de estos. Cuanto más pequeño es el tamaño, más precisa es la representación del tiempo, siendo la unidad el tamaño de slot ideal. Sin embargo, cuanto más pequeño es este tamaño, más elevado es el número de slots necesarios para representar la linea de tiempo en la que se evalúan las aplicaciones, y esto produce un aumento en el número de variables

utilizadas por el modelo, creciendo de esta forma el coste computacional del mismo.

En la Figura 3.12 se muestran dos ejemplos de planificación usando dos representaciones diferentes del tiempo mediante slots; (a) usando un tamaño de slot igual a 4, y (b) usando un tamaño de slot igual a 2. En una representación con un tamaño de slot grande, el número de slots a tener en cuenta se reduce, pero también se producen momentos de inactividad en los recursos que podrían ser usados para iniciar nuevas evaluaciones por parte del solver. Además habrá espacios de tiempo que serán desaprovechados donde no se podrán asignar aplicaciones, aumentando así la estimación del tiempo global de ejecución, como se puede observar en la Figura 3.12(a). En una representación en la que se establezca un tamaño de slot menor, se mejora la precisión de la representación, y los espacios de tiempo en el que los recursos se encuentran ociosos se reducen de forma notable. El número de slots a considerar en el proceso aumenta, pero la mejora de la precisión en la representación permiten reducir el tiempo de ejecución del conjunto, como se puede observar en la Figura 3.12(b).

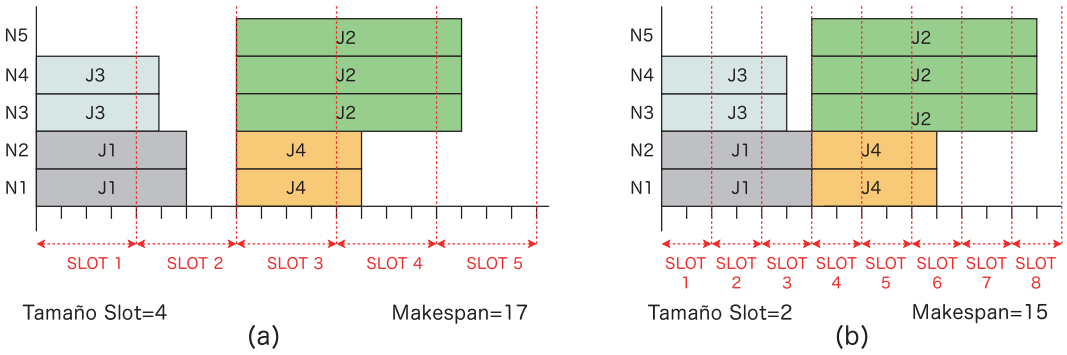


Figura 3.12: Representación de planificación con slots. (a) con tamaño de slot igual a 4 (b) con tamaño de slot igual a 2

Por lo tanto, es de gran importancia determinar el valor adecuado al tamaño de los slots de tiempo que nos permita acotar un equilibrio entre un coste computacional razonable y la calidad de las soluciones. En la Sección 3.2.2.1 se lleva a cabo un estudio para determinar el efecto que tiene el tamaño del slot en la calidad de las soluciones, y en el tiempo necesario para obtenerlas.

Teniendo esto en cuenta, se ha decidido utilizar como función objetivo del modelo MIP la minimización del makespan del conjunto de aplicaciones. Así,

de forma implícita se está forzando a que éstas, al ser asignadas al entorno, se haga de forma que en su conjunto liberen lo antes posibles los recursos de cómputo, reduciendo sus ejecuciones individuales, lo que se observará como una compactación del cómputo global.

3.2.1. Definición del modelo MIP

En la Figura 3.13 se describe el modelo MIP propuesto. A continuación, se detallan el conjunto de parámetros del modelo (Sección 3.2.1.1), las de variables de decisión (Sección 3.2.1.2), las restricciones principales (Sección 3.2.1.3) y la función objetivo que ha de minimizarse (Sección 3.2.1.4).

3.2.1.1. Parámetros del modelo

El conjunto de parámetros del modelo está dividido en tres grupos; la caracterización del entorno multi-cluster, los requisitos y características de las aplicaciones paralelas, siguiendo el modelo propuesto en las secciones anteriores, y por último, las nuevas variables correspondientes a la definición de los slots de tiempo.

El conjunto de parámetros que representan la caracterización del entorno multi-cluster son el conjunto de recursos de procesamiento (R), la potencia efectiva de cada uno (Γ_r), el conjunto de canales ($\mathcal{L}$), y el ancho de banda máximo disponible de cada uno (MBW_k).

Los parámetros que representan los requisitos y características de las aplicaciones paralelas son el conjunto de aplicaciones (J), el número de tareas que las componen (τ_j), su tiempo base (Tb_j), la proporción del procesamiento respecto a la comunicación (σ_j), y el ancho de banda requerido por tarea ($PTBW_j$).

Los parámetros que permiten manejar los slots son,el conjunto de slots (T), el número de slots en los que se divide el espacio de tiempo en que pueden asignarse y ordenarse las aplicaciones (θ), y el tamaño que tendrán estos slots (η).

Parámetros del modelo

1. R: Conjunto de recursos de procesamiento.
2. $\mathcal{L}$: Conjunto de enlaces inter-cluster.
3. Γ_r: Potencia efectiva del recurso r, $\forall r \in R$
4. MBW_k: Ancho de banda máximo del enlace inter-cluster k, $\forall k \in \mathcal{L}$.
5. J: Conjunto de aplicaciones a asignar
6. τ_j: número de tareas que componen cada aplicación j, $\forall j \in J$.
7. Tb_j: Tiempo base de la aplicación j, $\forall j \in J$.
8. $PTBW_j$: ancho de banda requerido por cada tarea de j, $\forall j \in J$
9. σ_j: factor de ponderación del procesamiento respecto la comunicación, $\forall j \in J$.
10. T: Conjunto de slots de tiempo.
11. θ: Número total de slots de tiempo. Deadline para el conjunto de aplicaciones.
12. η: Tamaño de los slots de tiempo.

Variables de decisión

13. $Z_{(j,r)} = 1$ si j es asignada al recurso r, $\forall j \in J, r \in R$
14. $X_{(j,r,t)} = 1$ si j es asignada al recurso r en el slot t, $\forall j \in J, r \in R, t \in T$
15. $Y_{(j,t)} = 1$ si j inicia su ejecución en el slot t, $\forall j \in J, t \in T$
16. s_j: slot en el que la aplicación j inicia su ejecución, $\forall j \in J$
17. f_j: slot en el que la aplicación j finaliza, $\forall j \in J$
18. SP_j es el retraso de procesamiento de la aplicación j, $\forall j \in J$
19. $BW_{j,k,t}$: Ancho de banda consumido por la aplicación j en el canal k en el slot t. $\forall j \in J, k \in \mathcal{L}, t \in T$
20. $ABW_{k,t}$: Ancho de banda disponible en el canal k, en el slot t, $\forall k \in \mathcal{L}, t \in T$

Función Objetivo

21. Minimizar el makespan global del conjunto de aplicaciones

Figura 3.13: Especificación del modelo MIP de OAS.

3.2.1.2. Variables de decisión

El conjunto de variables de decisión se divide en tres grupos; las variables relativas a la asignación de las aplicaciones paralelas, las variables relativas a los slots, y las variables con resultados intermedios sobre el entorno y las aplicaciones.

La asignación de una aplicación a un recurso en un slot concreto se determina mediante una variable binaria ($X_{j,r,t}$). Esta variable toma 1 como valor cuando una tarea de la aplicación j es asignada al recurso r en el slot de tiempo t. En cualquier otro caso, esta variable toma valor 0. Una segunda variable binaria ($Z_{j,r}$) se utiliza para identificar únicamente el recurso al que se asigna una aplicación, independientemente del slot.

El slot en que las aplicaciones inician su ejecución (s_j), se determina a través de una variable binaria ($Y_{j,t}$), que toma valor 1 si la aplicación j inicia su ejecución en el slot t, y 0 en caso contrario. El slot en que una aplicación finaliza su ejecución (f_j) se utiliza, junto con el resto de variables, para identificar el orden de ejecución de las aplicaciones, y las interacciones temporales entre ellas.

Por último, disponemos de un conjunto de variables auxiliares que serán necesarios para determinar el valor de las variables anteriores y para el funcionamiento de las restricciones del problema. La variable SP_j determina el retraso en el tiempo de ejecución de las aplicaciones producido por los recursos de procesamiento. La variable $BW_{j,k,t}$ determina el ancho de banda que cada aplicación ha consumido en un canal en un slot concreto, y $ABW_{k,t}$ determina el ancho de banda disponible en un canal en un slot determinado, una vez se han establecido las asignaciones.

3.2.1.3. Restricciones

Las restricciones del modelo nos permiten acotar el conjunto de soluciones válidas.

Al igual que se realizaba en la estrategia PAS, el modelo propuesto ha de asegurar el *Gang-Matching*, es decir, que todas las tareas que componen una aplicación paralela, han de ser asignadas. Además, teniendo en cuenta la dis-

cretización del tiempo, ha de asegurarse que estas tareas se asignan en recursos diferentes, y todas ellas han de iniciar su ejecución en el mismo instante de tiempo, y en consecuencia, en el mismo slot.

Uno de los objetivos principales del modelo es el de evitar la saturación en los canales entre clusters, al igual que en la estrategia PAS, por lo que deben incluirse las restricciones necesarias para que se produzcan soluciones con saturación.

A continuación se presentan cada una de las restricciones.

- La restricción 3.9 asegura que un recurso no se asigna a más de una aplicación paralela al mismo tiempo.

$$\sum_{\forall j \in J} X_{(j,r,t)} \leq 1, \quad \forall r, t \tag{3.9}$$

- La restricción 3.10 establece el valor de la variable binaria $Z_{(j,r)}$, que toma valor 1 si la aplicación j es asignada al recurso r, y 0 en caso contrario

$$Z_{(j,r)} = \max_{t \in T} \{X_{(j,r,t)}\}, \quad \forall j, r \tag{3.10}$$

- La restricción 3.11 asegura el *gang-matching*, es decir, que todas las tareas de cada aplicación (τ_j) son asignadas.

$$\sum_{\forall r \in R} Z_{(j,r)} = \tau_j, \quad \forall j \tag{3.11}$$

- La restricción 3.12 asegura que todas las tareas de cada aplicación (τ_j) son ejecutadas en el mismo tiempo, pero en recursos diferentes.

$$\sum_{\forall r' \in R} (X_{(j,r',t)}) \geq (\tau_j \cdot X_{(j,r,t)}), \quad \forall j, r, t \tag{3.12}$$

- Las restricciones 3.13 y 3.14 definen la variable binaria $Y_{(j,t)}$, que toma 1 como valor cuando la aplicación paralela j inicia su ejecución en el slot de tiempo t. En cualquier otro caso, la variable toma 0 como valor.

$$X_{(j,r,t')} \leq 1 - Y_{(j,t)}, \quad \forall j, r \wedge t' \in 1..(t-1) \tag{3.13}$$

$$\sum_{\forall t \in T} Y_{(j,t)} = 1, \quad \forall j \tag{3.14}$$

- La restricción 3.15 define el valor de la variable s_j, que identifica el slot de tiempo en el que la aplicación j inicia su ejecución. El valor es obtenido a partir de la variable $Y_{(j,t)}$.

$$s_j = \sum_{\forall t \in T} (Y_{(j,t)} \cdot t) - 1, \quad \forall j \tag{3.15}$$

- La restricción 3.16 determina el valor de la variable f_j, que identifica el slot de tiempo en el que la aplicación j finaliza su ejecución. Este valor es obtenido a partir del recurso más lento asignado a esa aplicación.

$$f_j = \max_{\forall r \in R, t \in T} (X_{(j,r,t)} \cdot t), \quad \forall j \tag{3.16}$$

- La restricción 3.17 define el coste que influye sobre el tiempo de ejecución de cada aplicación paralela. Este valor se obtiene a partir del retraso producido por el recurso de procesamiento usado más lento. Al evitarse el retraso producido por las comunicaciones, su efecto sobre el coste se fija a la unidad.

$$ct_j = \sigma_j \cdot SP_j + (1 - \sigma_j), \quad \forall j \tag{3.17}$$

- Las restricciones 3.18 y 3.19 se encargan de asegurar que los slots de

tiempo en los que ejecuta cada aplicación son contiguos, y que el número de ellos está en concordancia al tiempo de ejecución de la aplicación paralela, y al número de tareas de cada aplicación.

$$(f_j - s_j) \cdot \eta \leq (Tb_j \cdot ct_j), \quad \forall j \tag{3.18}$$

$$(\sum_{\forall r \in R, t \in T} X_{(j,r,t)} \cdot \eta) \geq (Tb_j \cdot ct_j \cdot \tau_j), \quad \forall j \tag{3.19}$$

- La restricción 3.20 define el ancho de banda disponible en el canal k en el slot de tiempo t, una vez las aplicaciones han sido asignadas a sus respectivos recursos, y teniendo en cuenta las comunicaciones que se realizarán entre clusters, en el caso de haber realizado co-asignación.

$$ABW_{k,t} = MBW_k - \sum_{\forall j \in J} BW_{(j,k,t)}, \quad \forall j, k, t \tag{3.20}$$

- La restricción 3.21 asegura que ninguna solución factible producirá saturación en los canales de comunicación inter-cluster. Esto se realiza asegurando que el ancho de banda disponible en un canal k, una vez realizadas las asignaciones de las aplicaciones, será mayor o igual a 0.

$$ABW_{k,t} \geq 0, \quad \forall k, t \tag{3.21}$$

- El conjunto de restricciones 3.22 acota los valores que podrán tomar las variables de decisión. $Z_{(j,r)}$, $X_{(j,r,t)}$ y $Y_{(j,t)}$ son variables binarias, por lo que sus valores podrán ser 0 o 1. Las variables s_j y f_j que determinan el slot de inicio y de final de las aplicaciones, así como el coste de ejcución

de las mismas ct_j, han de ser valores positivos.

$$Z_{(j,r)} \in \{0,1\}, \quad X_{(j,r,t)} \in \{0,1\}, \quad Y_{(j,t)} \in \{0,1\}, \quad s_j \geq 0, \quad f_j \geq 0, \quad ct_j \geq 0 \tag{3.22}$$

3.2.1.4. Función objetivo

El objetivo del modelo propuesto es el de minimizar el *makespan* del conjunto de aplicaciones paralelas, y que se expresa mediante la ecuación 3.23.

$$minimize\{ \quad \max_{\forall j \in J}(f_j) \quad \} \tag{3.23}$$

El *makespan* puede ser expresado como el último slot en que ha finalizado una aplicación. Por lo tanto, la función objetivo propuesta buscará aquella solución en que el último slot utilizado sea lo más bajo posible, respetando las restricciones previamente detalladas.

3.2.2. Experimentación

En esta sección se realiza un estudio experimental detallado del comportamiento de la estrategia *OAS*. El estudio tiene dos objetivos principales:

- Evaluar la influencia del tamaño de slot de tiempo en la calidad de las soluciones que proporciona el modelo
- Analizar la efectividad de la estrategia, comparándola con la estrategia *PAS*, presentada en la Sección 3.1, y con otras estrategias comúnmente utilizadas en la literatura.

A continuación se presenta el entorno de experimentación utilizado para llevar a cabo el estudio propuesto.

El modelo MIP propuesto ha sido implementado usando el solver de programación lineal *CPLEX* [IBM]. Además, se ha utilizado el framework *GridSim* [TGP] para simular la planificación de las aplicaciones en un entorno Multi-Cluster.

El problema de la planificación de aplicaciones con tareas independientes en entornos heterogéneos es NP-Completo, y esto implica un elevado coste computacional para determinar la solución óptima. Debido a esto, el tamaño del entorno experimental ha sido establecido en tamaño más reducido. Se ha configurado un Multi-Cluster heterogéneo compuesto por 3 clusters, conectados al switch central a través de un enlace de comunicaciones Gigabit. Cada cluster está compuesto por 4 nodos, homogéneos, y con potencias efectivas distintas para cada cluster, tal y como se detalla en la Tabla 3.3.

Cluster	Nodos	Potencia efectiva (Γ_r)
C1	4	1.0
C2	4	0.75
C3	4	0.5

Tabla 3.3: Caracterización del entorno multi-cluster

Hemos utilizado un conjunto de workloads basado en el workload utilizado en la Sección 3.1.3. Debido a la elevada complejidad del modelo MIP, los workloads han sido configurados para estar compuestos por un número más reducido de aplicaciones, formadas por un número reducido de tareas, pero con un tiempo base más elevado. Los workloads han sido fijados para estar compuestos por 8 aplicaciones, cada una de ellas compuestas por, entre 1 y 12 tareas, con un tiempo base promedio por aplicación de 67×10^4 segundos.

El comportamiento de *OAS* en cuanto al tratamiento de las comunicaciones puede ser extrapolado de las observaciones sobre la estrategia *PAS*, pues ambas utilizan un mecanismo de asignación muy similar.

Para evaluar la calidad de las soluciones proporcionadas por las diferentes estrategias, en los experimentos propuestos, se estudian los resultados sobre las siguientes métricas:

1. *Makespan*: Mide el tiempo que ha sido necesario para completar la ejecución del conjunto completo de aplicaciones.

2. *Grado de compactación*: Métrica normalizada que se utiliza para medir el grado de utilización de los recursos, evaluando el porcentaje de tiempo en

que los nodos han sido utilizados. Cuanto más grande es la compactación, mayor es la utilización de los recursos del sistema.

3. *Slowdown de ejecución*: Métrica normalizada que mide el retraso promedio que han sufrido las aplicaciones en su tiempo de ejecución, respecto al tiempo base de referencia.

4. *Tiempo de respuesta*: Se utiliza para medir el tiempo promedio que las aplicaciones pasan en en el entorno desde su llegada hasta la finalización de su ejecución.

3.2.2.1. Análisis del tamaño de los slots de tiempo

Como se ha indicado previamente, *OAS* utiliza una representación del tiempo basada en *slots*, que permite establecer el orden de ejecución de las aplicaciones, así como evaluar las interacciones entre las mismas. El tamaño de los slots de tiempo puede limitar la calidad de las soluciones o el tiempo de decisión, ya que para tamaños muy reducidos la complejidad aumenta enormemente, mientras que para tamaños muy grandes se desperdicia mucho tiempo de utilización de los recursos, y por lo tanto, empeora la calidad de las decisiones y el rendimiento de las aplicaciones.

Para poder determinar el impacto que el tamaño de los slots ejerce sobre la calidad de las soluciones, evaluamos las soluciones obtenidas para un mismo workload, pero variando el tamaño de slot.

En la Figura 3.14 se muestran el makespan y el tiempo de solución para cada uno de los casos de estudio. El eje horizontal representa los diferentes tamaños de slot que se han utilizado, correspondientes al porcentaje relativo al tiempo base promedio del conjunto de aplicaciones, siendo estos valores $\{15\,\%, 25\,\%, 50\,\%, 100\,\%, 150\,\%, 200\,\%, 250\,\%\}$. Finalmente las soluciones obtenidas por el solver CPLEX son aplicadas al simulador GridSim. El eje vertical derecho muestra el tiempo que el solver *CPLEX* ha necesitado para proporcionar la solución de la planificación para cada uno de los caso. Finalmente, el eje vertical izquierdo muestra el makespan que se ha obtenido en cada caso de estudio, medido en segundos.

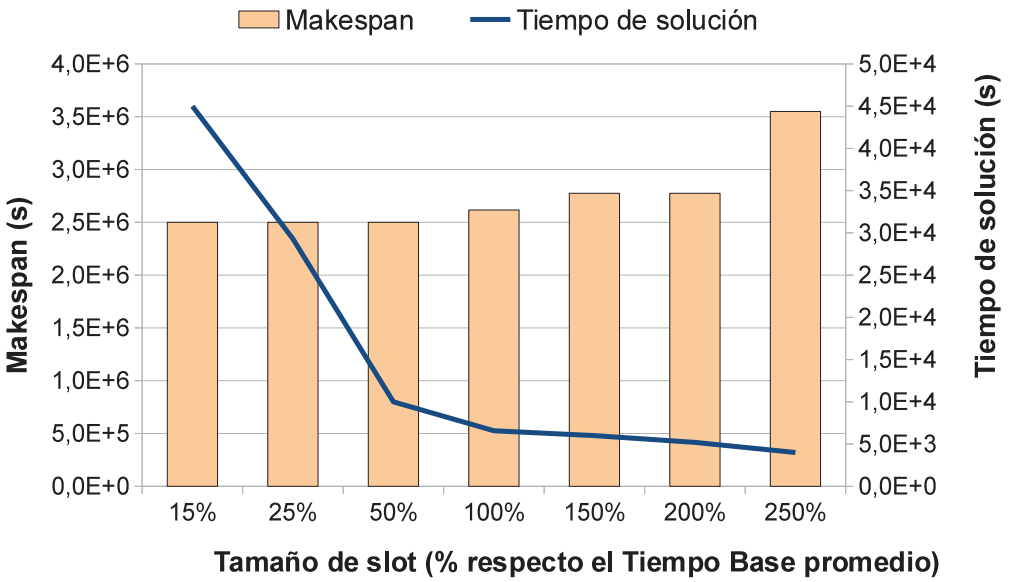


Figura 3.14: Comparación de tiempo de solución y calidad en función del tamaño de slot

Como se puede observar, cuanto más grande es el tamaño del slot, más elevado es el makespan producido. Este efecto se produce porque el uso de slots grandes provoca que la disponibilidad de recursos se reduzca durante largos periodos de tiempo, incluso cuando aplicaciones que ya han finalizado su ejecución han liberado estos recursos. En consecuencia, el comienzo de las siguientes aplicaciones se retrasa hasta el siguiente slot. Se aprecia que cuanto menor sea el tamaño de slot utilizado, menor será el makespan que se producirá en la planificación. Igualmente, se puede observar que por debajo de un tamaño de slot del 50 % respecto al tiempo base promedio, el makespan que se produce es el prácticamente mismo. Esto indica que por debajo de este valor, a pesar de contar con un espacio de soluciones más amplio, *CPLEX* ya ha encontrado una solución muy próxima a la solución óptima del conjunto, y no es capaz de proporcionar una que reduzca aún más el makespan.

Respecto al tiempo de solución de la planificación, se puede observar como éste aumenta al reducirse el tamaño de slot utilizado. Este aumento viene producido por el incremento de la complejidad del problema, ya que implica el uso de un mayor número de slots para representar la linea de tiempo de la planificación. Por contra, al aumentar el tamaño de los slots, la cantidad

de los mismos necesaria para representar el tiempo, se reduce, y con ello la complejidad y el tiempo necesario para proporcionar una solución al problema. A partir de un tamaño de slot del 50 % respecto al tiempo base promedio, el tiempo de solución se mantiene de forma contenida, creciendo de forma importante al usar slots más pequeños.

La utilización del sistema es un aspecto de gran importancia que hay que considerar en una planificación, y puede medirse a través del grado de compactación, que mide el porcentaje del tiempo en el que los recursos están ocupados. Cuanto mayor sea la compactación, menor será el tiempo en que los recursos han estado desocupados, esperando para ser utilizados. Los resultados para esta métrica se muestran en la Figura 3.15.

Como se puede observar, el efecto que se produce sobre el grado de compactación sigue un comportamiento inverso al observado sobre el makespan en la Figura 3.14: cuanto más pequeño es el slot utilizado, más elevado es el grado de compactación de la planificación. El efecto de un makespan más reducido se puede entender como si el workload hubiese sido comprimido para ocupar menos espacio, deja menos recursos sin ser utilizados.

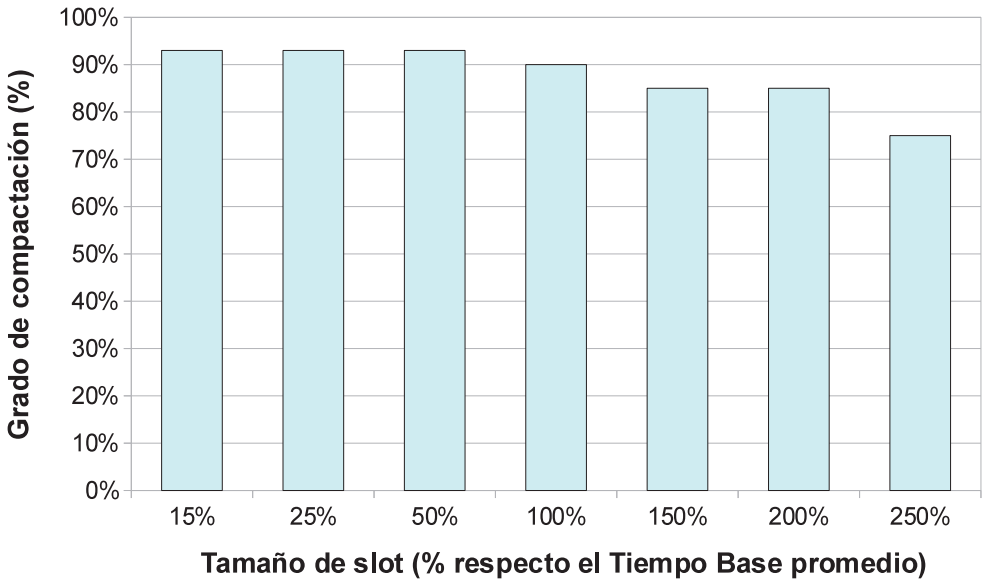


Figura 3.15: Impacto del tamaño de slot sobre el grado de compactación

El objetivo de la estrategia es el de minimizar el makespan del conjunto de

aplicaciones, y aprovechar al máximo la utilización de los recursos. Otra métrica que nos permite medir el efecto sobre el rendimiento de las aplicaciones es el tiempo de respuesta. En la Figura 3.16, se representa el tiempo de respuesta promedio de las aplicaciones, con sus dos componentes: el tiempo de espera y de ejecución promedio. Como puede observarse, el uso de un slot demasiado grande provoca que se incremente el valor tanto del tiempo de espera como de ejecución promedios, mientras que en el caso de usar un slot pequeño, los dos componentes del tiempo de respuesta promedio ven reducido su valor.

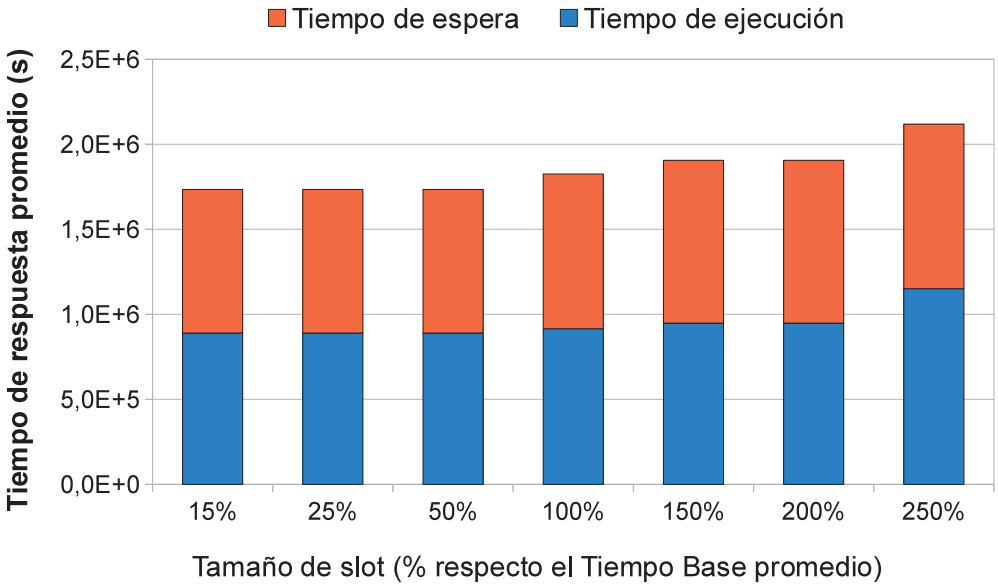


Figura 3.16: Impacto del tamaño de slot sobre el tiempo de respuesta promedio

El efecto sobre el tiempo de ejecución se puede medir más detalladamente evaluando el retraso que éste ha sufrido respecto al tiempo base promedio de referencia. Los resultados se muestran en la Figura 3.17. Como se puede observar, cuanto mayor es el tamaño de slot utilizado, el tiempo de ejecución promedio sufre un retraso mayor respecto al tiempo base, lo que significa que las aplicaciones han tardado más en ejecutarse.

Teniendo en cuenta los resultados de los diferentes experimentos, se puede concluir que el tamaño de los slots de tiempo tiene un impacto significativo, tanto sobre la calidad de las soluciones como sobre el tiempo necesario para obtener la planificación. Cuanto más pequeño es el tamaño de slot utilizado,

mejores son los resultados obtenidos por la estrategia. Sin embargo, esta mejora de calidad en la solución es a costa de un aumento considerable del tiempo necesario para producirla. A pesar de esto, se ha determinado que un tamaño de slot igual al 50 % del tiempo base promedio del conjunto de aplicaciones, constituye un compromiso entre calidad de la solución y el tiempo necesario para obtenerla. Se ha observado que en valores por debajo del 50 %, el solver *CPLEX* no ha sido capaz de proporcionar soluciones con mejores resultados. Por este motivo, se ha decidido utilizar este tamaño de slot del 50 % para la realización del resto de experimentos sobre *OAS*.

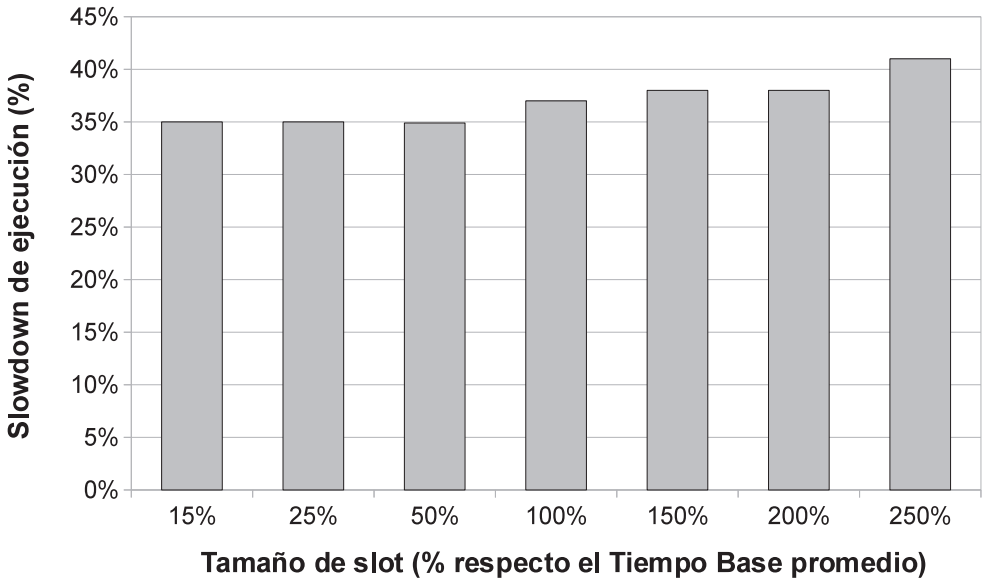


Figura 3.17: Impacto del tamaño de slot sobre el slowdown de ejecución promedio

3.2.2.2. Evaluación del rendimiento

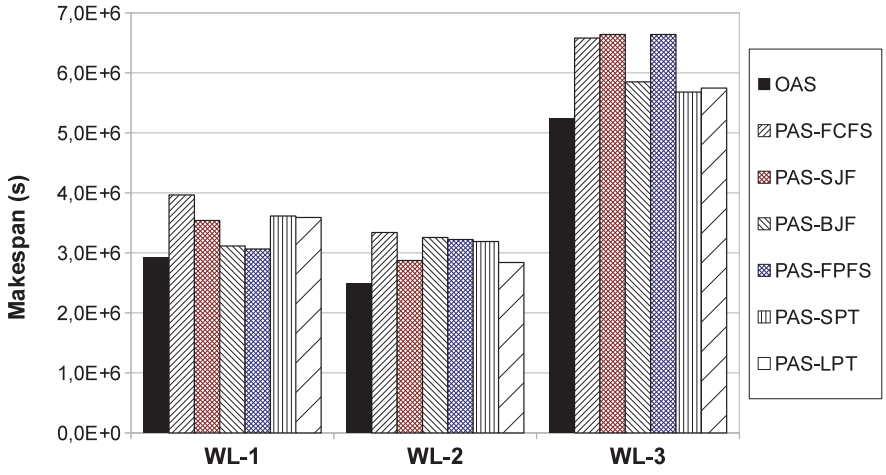
Para analizar el rendimiento de las asignaciones obtenidas por *OAS*, se compararán con los resultados de *PAS*, además de otras estrategias comúnmente utilizadas en la literatura: *First Come First Served* (FCFS), *Short Jobs First* (SJF), *Big Jobs First* (BJF), *Fit Processors First Served* (FPFS), *Short Processing Time* (SPT) y *Long Processing Time* (LPT). En este estudio, la estrategia PAS ha sido configurada para utilizar en la función de selección las

estrategias de la literatura previamente enumeradas.

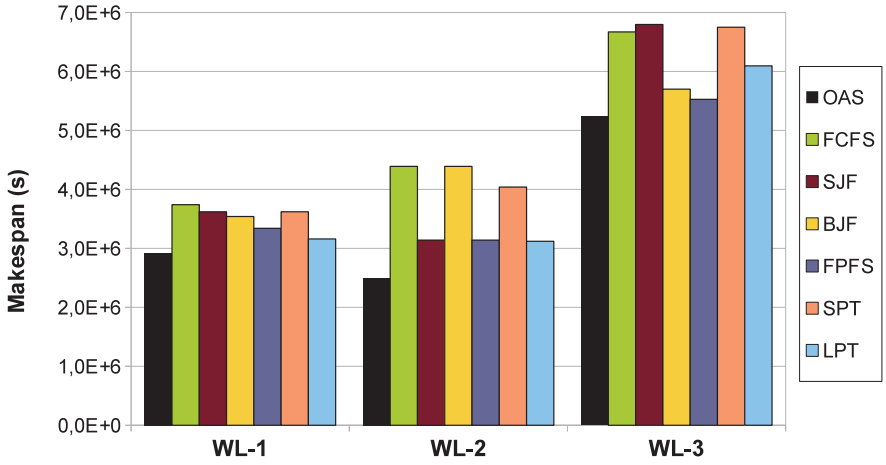
El abanico de pruebas se ha realizado sobre un conjunto de workloads sintéticos, con las características presentadas al principio de la presente sección. Tomando en consideración las conclusiones obtenidas en el análisis de la Sección 3.2.2.1, se ha asumido un tamaño de slot del 50 % del tiempo base promedio para todas las pruebas de *OAS* . Se han tomado tres workloads representativos para analizar y presentar visualmente los resultados. El primero de ellos, *WL-1*, ha sido diseñado para intentar favorecer aquellas estrategias que tratan de priorizar las aplicaciones que usan los recursos que se encuentran ociosos. En cambio, el workload *WL-2* ha sido diseñado para que favorezca aquellas estrategias que tratan de priorizar las aplicaciones con un número reducido de tareas. Por último, el workload *WL-3* ha sido diseñado sin intentar favorecer ninguna estrategia en particular.

En la Figura 3.18 se muestran los resultados respectivos al makespan, comparando *OAS* con *PAS* (Figura 3.18(a)), y con las estrategias clásicas (Figura 3.18(b)). El eje horizontal representa los tres workloads evaluados, y el eje vertical representa el makespan, medido en segundos. Se puede observar que cada estrategia tiene un comportamiento diferenciado, con resultados variados.

Se observa que la estrategia que obtiene buenos resultados en un workload, sin embargo, presenta peores resultados en otro workload: así, se observa que *FPFS* en general rinde bien en el workload *WL-1*, mientras que *SJF* lo hace mejor en *WL-2*. En el caso de la comparativa con las diferentes versiones de *PAS*, se puede observar un comportamiento similar, tal y como se analizó en la Sección 3.1.3.3: algunos métodos de selección dan buenos resultados en unos casos, pero peores en otros, lo que impide que sea adecuada su utilización para todas las situaciones. Sin embargo, como puede observarse, *OAS* es capaz de producir asignaciones que minimizan el makespan, siendo 21 % la mejora promedio respecto a *PAS*, y del 30 % respecto a las estrategias clásicas. Esto se debe a su capacidad de evaluar de forma global y simultánea el conjunto de aplicaciones de la cola, mientras que *PAS* se ve restringida a subconjuntos de aplicaciones, y en el caso de las estrategias clásicas están limitadas a evaluar las aplicaciones una a una. Por este motivo le permite tomar mejores decisiones de planificación , aprovechando mejor los recursos, reduciendo el makespan del



(a) Comparación con PAS



(b) Comparación con estrategias clásicas

Figura 3.18: Comparativa del makespan. (a) con respecto a la estrategia PAS. (b) con respecto a estrategias clásicas de la literatura

conjunto de aplicaciones, liberando antes todos los recursos ocupados, para que puedan ser aprovechados por las siguientes aplicaciones.

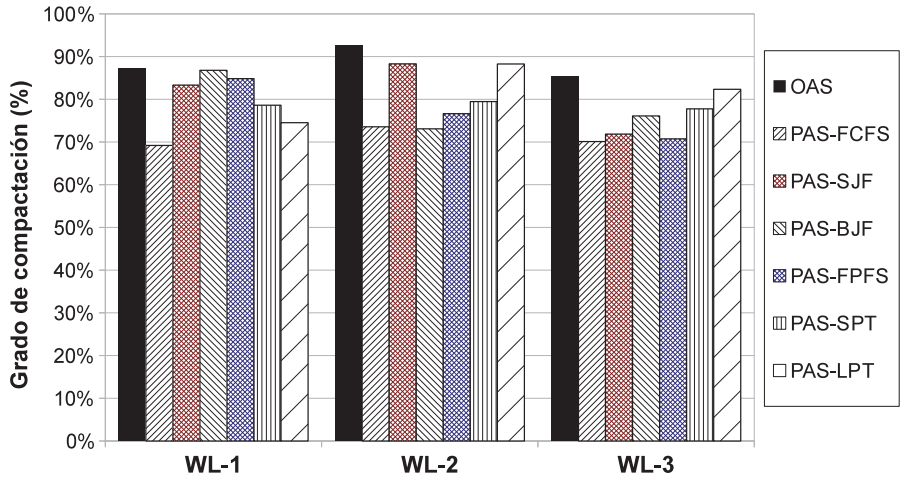
En la Figura 3.19 se muestran los resultados obtenidos para esta comparativa Como puede observarse, los métodos de selección empleados sobre la estrategia *PAS* (Figura 3.19(a)) obtienen unos resultados bastante estables, todos ellos por encima del 70 % de compactación, obteniendo alguna de las estrategias valores próximos a los de *OAS*, como es el caso de *BJF* en el workload *WL-1*. Sin embargo, esta estrategia obtiene malos resultados en el caso del workload *WL-2*. En el caso de las estrategias clásicas (Figura 3.19(b)), la dispersión de valores es mucho más alta, con valores que oscilan entre el 57 % y el 92 % de compactación.

En algún caso, como *FPFS* en el workload *WL-1*, obtiene el mayor grado de compactación, pero sin embargo su makespan no era el más reducido, tal y como se ha mostrado en la Figura 3.18(b).

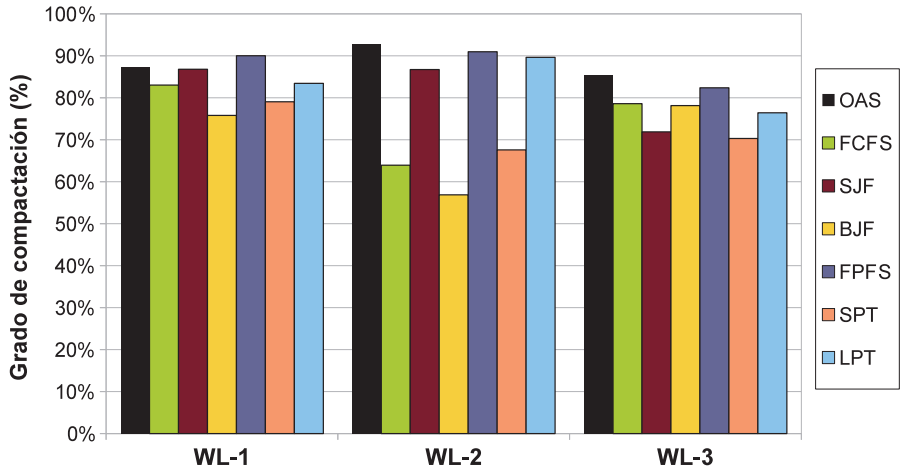
Una vez evaluado el comportamiento de las diferentes estrategias respecto al punto de vista del sistema, pasamos a analizar como se comportan en cuanto al rendimiento de las aplicaciones. La Figura 3.20 muestra los resultados comparativos del tiempo de respuesta promedio, dividido en sus dos componentes: tiempo de espera, y tiempo de ejecución.

Cuanto menor es el tiempo de respuesta, significa que la aplicación ha permanecido menos tiempo en el sistema, representando un mejor resultado. En el eje horizontal se muestran los nombres de las diferentes estrategias, para cada uno de los workloads evaluados. En el eje vertical se representa el tiempo de respuesta promedio de las aplicaciones, medido en segundos. Como se puede observar, en la comparación con los diferentes métodos de selección funcionando sobre *PAS* (Figura 3.20(a)), la estrategia *OAS* obtiene buenos resultados, para todos los workloads. Las dos estrategias intentan asignar los mejores recursos, evitando la saturación de los canales de comunicación.

En los tiempos de espera, donde sí se aprecian diferencias importantes, *OAS* consigue unos buenos resultados, que solo son superados, por muy poco margen, por las estrategias *PAS-SJF* y *PAS-FPFS* en el workload *WL-1*. Sin embargo, es importante recordar que estas dos estrategias no ofrecían buenos valores de makespan en dicho workload. Esto evidencia que *OAS* es capaz

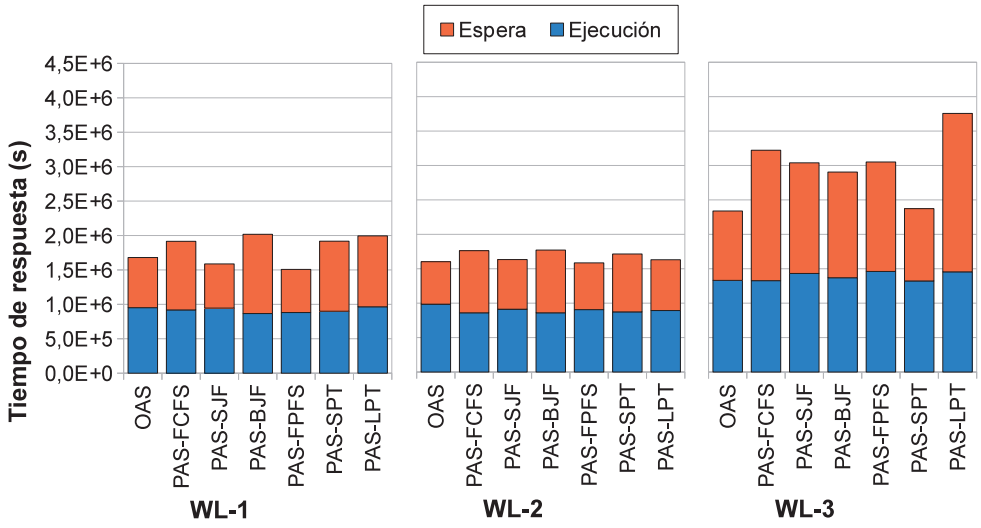


(a) Comparación con PAS

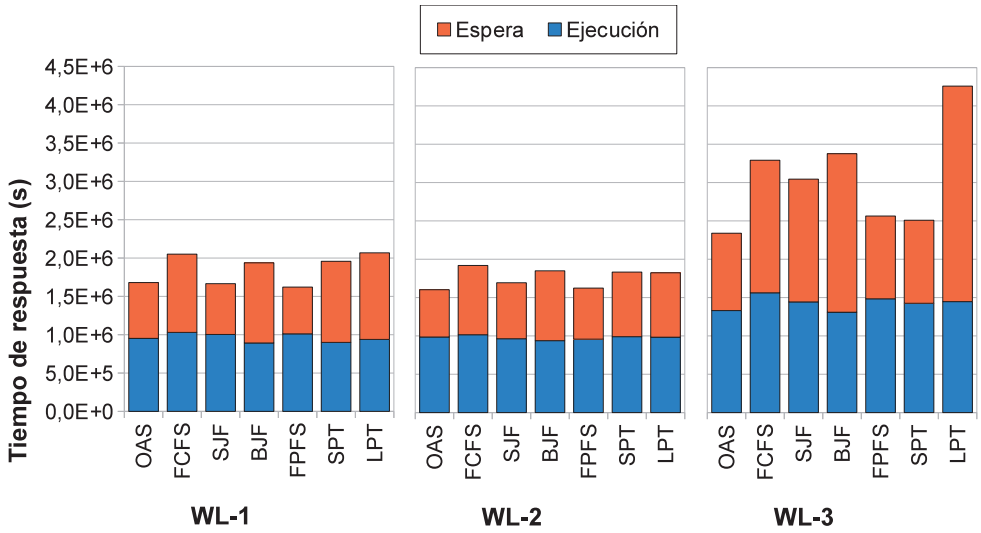


(b) Comparación con estrategias clásicas

Figura 3.19: Comparativa del grado de compactación. (a) con respecto a la estrategia PAS. (b) con respecto a estrategias clásicas de la literatura



(a) Comparación con PAS



(b) Comparación con estrategias clásicas

Figura 3.20: Comparativa del tiempo de respuesta promedio. (a) con respecto a la estrategia PAS. (b) con respecto a estrategias clásicas de la literatura

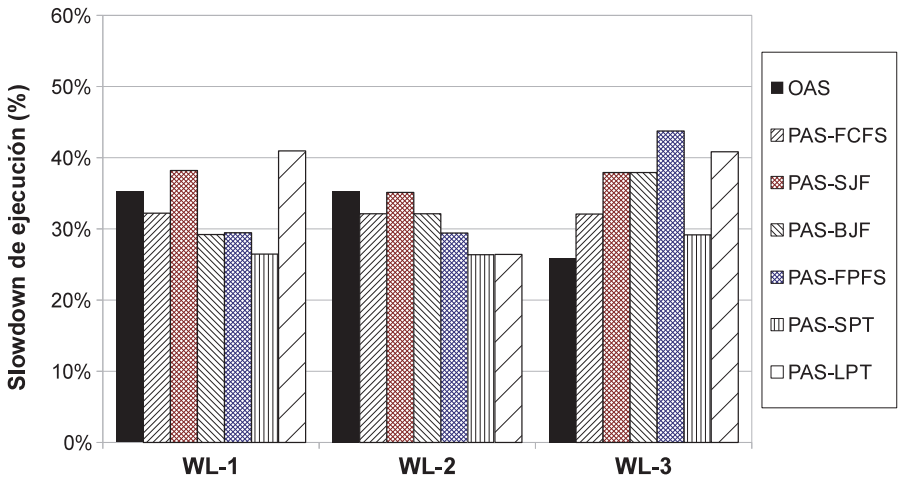
de realizar una planificación equitativa, que mejora tanto el rendimiento del conjunto de aplicaciones, como la utilización del sistema. Un comportamiento muy similar se observa en la comparación con las estrategias clásicas (Figura 3.20(b)), donde ninguna de ellas es, en general, mejor que las otras. En esta comparación, se puede observar una variabilidad algo mayor en el tiempo de ejecución promedio, siendo las soluciones proporcionadas por *OAS* siempre de las mejores. Lo mismo es aplicable a los tiempos de espera, donde en general *OAS* obtiene los mejores resultados. De los resultados obtenidos se puede concluir que *OAS* es capaz de obtener el menor tiempo de respuesta, independientemente del tipo de workload.

Por último, se evalúa el retraso que se produce sobre el tiempo de ejecución de las aplicaciones paralelas. Éste retraso se representa como la diferencia normalizada del tiempo ejecución promedio de las aplicaciones respecto al tiempo base promedio de las mismas. Los resultados de esta comparativa se muestran en la Figura 3.21.

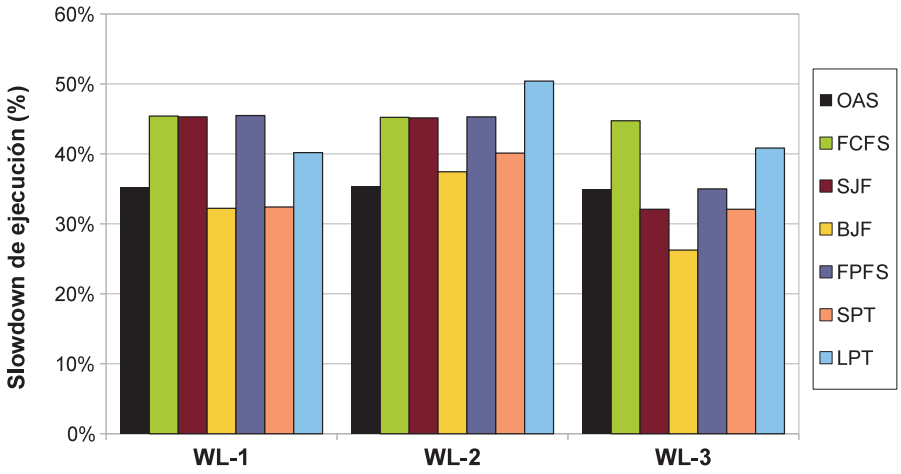
Como se puede observar, tanto en la comparación con los métodos de selección de *PAS* (Figura 3.21(a)), como en la comparación con las estrategias clásicas (Figura 3.21(b)), *OAS* obtiene buenos resultados, aunque se ve superada en algunos casos por las estrategias BJF y SPT. Hay que tener en cuenta, sin embargo, que estas estrategias en esos casos han obtenido peores resultados en el resto de métricas evaluadas. Esto se debe a que asignar aplicaciones una a una, intentando mejorar su tiempo de ejecución puede ocupar recursos que hubiese sido preferible dejar para otra aplicación, provocando que deba esperar, o que sea asignado a recursos menos adecuados. La estrategia *OAS* evita estas situaciones gracias a su capacidad de evaluar todo el conjunto de aplicaciones de forma simultánea.

3.2.3. Conclusiones

Se ha propuesto una nueva técnica denominada *Ordering and Allocation Strategy (OAS)*, una estrategia basada en un modelo MIP, que es capaz de determinar el orden de ejecución y la asignación de recursos a un conjunto de aplicaciones, evaluándolas de forma simultánea. La estrategia tiene en cuenta



(a) Comparación con PAS



(b) Comparación con estrategias clásicas

Figura 3.21: Comparativa del slowdown de ejecución. (a) con respecto a la estrategia PAS. (b) con respecto a estrategias clásicas de la literatura

los requisitos de procesamiento y de comunicación, y evita aquellas soluciones en las que se provocaría una saturación en los canales de comunicación. Esta técnica supera la restricción principal de la estrategia *PAS*, la cual únicamente podía evaluar subconjuntos de aplicaciones, que por sus características pudiesen ser ejecutadas, en el momento de su evaluación, por el sistema.

OAS implementa una representación del tiempo mediante slots, que permite evaluar las interacciones entre las distintas aplicaciones. El uso de slots para representar el tiempo en el modelo es un parámetro crítico y por ello, se ha llevado a cabo un análisis para determinar el impacto que podían tener sobre la calidad de las soluciones. Analizando los resultados, se ha observado que cuanto menor es el tamaño de slot, el resultado es más próximo al óptimo, aunque a expensas de un mayor tiempo de solución. En cambio, cuanto más grande es el tamaño de slot, menor es la calidad de las soluciones, así como el tiempo requerido para obtenerla. Se ha determinado que el uso de un tamaño de slot del 50 % del tiempo base promedio de las aplicaciones, permite al modelo proporcionar soluciones muy cercanas al óptimo en un tiempo acotado, y en consecuencia, éste ha sido el valor utilizado para el resto de experimentos.

Se han comparado los resultados en diferentes workloads con *PAS*, además de con otras estrategias presentes en la literatura. Analizando los resultados, se ha observado que *OAS* es capaz de producir soluciones que en general obtienen mejores resultados para las diferentes métricas utilizadas. *OAS* es capaz de producir soluciones que minimizan el makespan, y que incrementan la utilización de los recursos del sistema. Desde el punto de vista del rendimiento de las aplicaciones, es capaz de mejorar los tiempos de respuesta, y obtener valores reducidos de slowdown de ejecución. Estos resultados se deben a la capacidad de *OAS* de evaluar de forma simultánea el conjunto de aplicaciones en la cola de espera, teniendo en cuenta que podrán compartir los recursos del sistema durante el mismo tiempo, considerando tanto el procesamiento como las comunicaciones.

A pesar de proporcionar soluciones muy cercanas al óptimo, el tiempo requerido por *OAS* para obtenerlas hace que su implementación en schedulers on-line en entornos de producción no sea apropiada. Sin embargo, el uso de esta estrategia permite disponer de una solución para la reducción del tiempo

de ejecución de aplicaciones paralelas próxima a la óptima. Esto es de gran utilidad para el desarrollo de otras estrategias basadas en heurísticas que reduzcan el problema, puesto que una solución óptima puede ser utilizada como referencia con la cual compararse.

Teniendo en cuenta las limitaciones de *OAS*, pero también la forma en que determina la planificación de las aplicaciones, en la siguiente sección se propone desarrollar una estrategia con objetivos similares a los de *OAS*, capaz de obtener una solución en un tiempo razonablemente inferior.

3.3. MESD: Minimum Execution Slowdown

En la Sección 3.2 se ha presentado la estrategia *Ordering and Allocation Strategy (OAS)*, consistente en un modelo MIP, capaz de producir soluciones en las que el tiempo de ejecución del conjunto de aplicaciones se reduce, además de mejorar la utilización de los recursos del sistema, determinando el orden de ejecución y la asignación óptima, para un conjunto de aplicaciones, evaluándolas de forma simultánea. *OAS* tiene en cuenta tanto los requisitos de procesamiento como de comunicaciones a la hora de tomar sus decisiones.

Debido a la elevada complejidad del problema a resolver, el tiempo necesario para calcular la solución al mismo es elevado. Por ese motivo, la caracterización de los entornos Multi-Cluster y los workloads de aplicaciones paralelas a tratar han sido restringidos a un tamaño reducido.

Partiendo de los resultados obtenidos con *OAS*, se propone diseñar e implementar una estrategia con unos objetivos similares, pero con la capacidad de proporcionar la solución en un tiempo reducido.

En esta sección se presenta la estrategia heurística *Minimum Execution Slowdown* (MESD).

3.3.1. Metodología de funcionamiento de la estrategia

Las estrategias que han sido presentadas a lo largo del presente trabajo, *Package Allocation Strategy (PAS)* en la Sección 3.1, y *Ordering and Allocation Strategy (OAS)* en la Sección 3.2, han tratado de solucionar el problema de la planificación de aplicaciones paralelas principalmente mediante modelos programación entera MIP. Esta técnica permite obtener una solución óptima, o casi-óptima según la configuración del modelo, a un problema dado. Sin embargo, su complejidad, es muy elevada, lo que ha provocado que el tiempo necesario para obtener una solución es elevado.

Las técnicas más ampliamente utilizadas para resolver el problema de la planificación se basa en el uso de *meta-heurísticas* [KX11]. Éstas técnicas son adecuadas para su uso en la planificación de aplicaciones, debido a que permiten desarrollar métodos que tengan en cuenta diferentes atributos de la planificación, perseguir múltiples objetivos, o adaptarse para la planificación

tanto en los modos *on-line* como *off-line*.

Los métodos *meta-heurísticos* pueden dividirse en varios grupos, siendo los más eficientes y ampliamente utilizados las heurísticas *ad-hoc* [BSB+01], las heurísticas basadas en *búsqueda local* como *búsqueda tabu* [ABN00], y los métodos basados en poblaciones, conocidos como *algoritmos evolutivos* [LAH09, KXK09].

Las heurísticas *ad-hoc* se caracterizan por ser métodos desarrollados a medida para la solución de un problema en concreto. Consisten en un conjunto de algoritmos diseñados a partir del conocimiento que se ha extraído del problema, y determinan una solución en base a criterios especificados. Son de gran utilidad en la optimización con un único objetivo, como reducir el tiempo de ejecución, el número de canales de comunicación utilizados, maximizar la utilización de los recursos del sistema, etc. También son de gran utilidad para buscar soluciones en las que no es necesario que el resultado esté muy próximo a un óptimo global. Además, tienen un coste computacional reducido, lo que permite obtener una solución en un tiempo mínimo.

Los métodos basados en *búsqueda local* consisten en la construcción de un espacio de soluciones a partir del problema propuesto, y en la exploración de este espacio por parte del algoritmo, con el fin de localizar aquellas soluciones que satisfagan los requerimientos especificados. Su utilización es adecuada para la búsqueda de sub-óptimos, u óptimos locales, y su coste computacional es superior al de las heurísticas *ad-hoc*.

Por último, los métodos basados en *poblaciones*, realizan un proceso de optimización de una solución siguiendo el mecanismo de la evolución de seres vivos. Unos agentes son creados, y en ellos se especifican ciertos atributos del problema que se desea resolver. A lo largo de diversas etapas, el agente habrá mutado, se habrá reproducido, y acabará evolucionando en un conjunto que representará una solución óptima. Éste tipo de métodos son adecuados para la búsqueda de soluciones muy próxima al óptimo global. Esto tiene el efecto de repercutir en un coste computacional superior al de las heurísticas *ad-hoc* y las de *búsqueda global*. Debido a las características del problema a resolver se ha decidido por usar una implementación *ad-hoc*.

En el desarrollo de la heurística que se propone en este apartado, el punto

de partida ha sido el conocimiento obtenido a partir de la estrategia *OAS*, que proporciona una planificación muy próxima a la óptima del conjunto de aplicaciones, pero que sin embargo, requiere de un tiempo muy elevado para obtener dicha planificación. El estudio experimental realizado sobre la estrategia ha permitido identificar una serie de patrones que pueden ser empleados como criterios en el proceso de determinar planificaciones que reduzcan el tiempo de ejecución de las aplicaciones paralelas. Éstos se pueden resumir en:

- La estrategia *OAS* ha intentado asignar los mejores recursos de cómputo, evitando asignarlos si ello no implicaba una mejora en el rendimiento de la aplicación, reduciendo así el número de recursos infra-utilizados.

- Es capaz de modificar el orden de ejecución de las aplicaciones, de forma que el rendimiento del conjunto de aplicaciones aumente.

Los resultados obtenidos por las estrategias estrategias *PAS* y *OAS* demuestran que tratar el conjunto de aplicaciones en vez de hacerlo de forma individual, a la vez que se considera el orden en que estas aplicaciones deben ser ejecutadas, permite obtener una reducción notable en los tiempos de ejecución globales, a la vez que se optimiza el uso de los recursos de procesamiento y comunicación. Por este motivo, el objetivo de la estrategia *MESD* se basa en seguir ambas premisas.

Otro aspecto importante, además de definir la propia heurística, es determinar la forma en que ésta debe ser aplicada. Si ésta es llamada cada vez que una aplicación entra en la cola de espera del sistema, la aplicación sería evaluada de forma aislada. Por ende, resultaría imposible obtener ventaja del orden de ejecución, además de no poderse tener en cuenta los requisitos y características de otras aplicaciones para poder tomar una decisión equitativa con el conjunto. Por otro lado, si se intenta esperar para poder evaluar simultáneamente un conjunto de aplicaciones muy grande, se pueden provocar tiempos de espera elevados en las aplicaciones, así como incrementar la posibilidad de que recursos del sistema quedaran ociosos de forma innecesaria. Teniendo esto en cuenta, la forma en que se propone que *MESD* sea aplicada se basa en las siguientes situaciones:

1. Si en la cola de espera se encuentra una única aplicación, y se disponen de recursos suficientes para asignarla, se evalúa ésta aplicación de forma aislada. Al tratarse una sola aplicación, únicamente se determina su asignación, de forma que pueda acceder a los nodos más potentes, pero evitando usarlos si ello no implica una reducción del tiempo de ejecución, tal y como se ha deducido del comportamiento de la estrategia *OAS*.

2. Si en la cola de espera se encuentra una única aplicación, pero no se disponen de recursos suficientes para asignarla, ésta deberá esperar en la cola. Puede darse la situación de que otras aplicaciones entren en la cola del sistema antes de que la actual pueda ser asignada. En este caso, *MESD* será llamada cuando se liberen recursos, y tomará como entrada el conjunto de aplicaciones que se encuentren en la cola.

3. Si hay una o más aplicaciones en el sistema que ya han sido evaluadas por *MESD*, dichas aplicaciones se asignarán según ha establecido *MESD*. Durante este tiempo, otras aplicaciones pueden entrar en la cola del sistema. En esta situación, *MESD* no será ejecutado hasta que todas las aplicaciones previamente evaluadas sean asignadas. Una vez hecho, la estrategia evaluará normalmente el conjunto de aplicaciones que han ido entrando en la cola del sistema.

En las siguientes secciones se presentan en detalle los mecanismos de asignación de recursos y de ordenación de aplicaciones.

3.3.2. Asignación de recursos

En el estudio sobre el comportamiento y rendimiento de la estrategia *OAS*, realizado en la Sección 3.2.2.2, se observa como la estrategia asigna las aplicaciones de forma que se utilicen aquellos recursos que permiten reducir el tiempo de ejecución, pero con el compromiso de que estos no sean infra-utilizados. Cuando es necesario co-asignar una aplicación, y utiliza recursos de varios clusters, uno de los clusters puede ser más potente que el otro. En estos casos, el tiempo de ejecución de la aplicación es determinado por el cluster más lento, y es preferible dejar libres los nodos del cluster rápido, para que puedan ser

aprovechados por otras aplicaciones. El hecho de liberar estos recursos más rápidos no implica una pérdida de rendimiento para la aplicación si no se utilizan recursos más lentos que los del cluster más lento que ya se había asignado.

Teniendo en cuenta el comportamiento de la estrategia *OAS*, se diseña el método de asignación de la estrategia *MESD* en base a dos pasos:

1. *Calcular la mejor asignación.* Teniendo en cuenta la disponibilidad de los recursos en el momento en que se aplica la estrategia, se determina, para cada aplicación paralela, aquella asignación que minimiza su tiempo de ejecución. Esta asignación define el menor tiempo de ejecución posible para cada una de las aplicaciones paralelas a ser evaluadas.

2. *Optimizar los recursos.* En el paso de asignación previamente realizado, se pueden dar casos en los que se utilicen recursos de diferentes clusters, con potencias efectivas (Γ_r) de diferentes valores. El tiempo de ejecución de una aplicación paralela viene determinado por los recursos utilizados con el menor valor de Γ_r, por lo que el hecho de utilizar recursos más rápidos no repercute en una reducción de su tiempo de ejecución. Por lo tanto, en este segundo paso, para cada una de las asignaciones determinadas en el primer paso, se mueven tareas asignadas en recursos rápidos al cluster más lento utilizado. Esta re-asignación ayuda a reducir la utilización de los canales de comunicación entre clusters, así como el liberar recursos que se encontraban infra-utilizados, proporcionando mejores oportunidades de asignación para futuras aplicaciones.

En la Figura 3.22 se muestra un ejemplo del proceso de asignación de recursos a las aplicaciones. Se asume un entorno multi-cluster formado por dos clusters: C1=$\{$N1, N2$\}$ con potencia efectiva $\Gamma_{N1} = \Gamma_{N2} = 0{,}75$, y C2=$\{$N3, N4$\}$ con $\Gamma_{N3} = \Gamma_{N4} = 0{,}5$, siendo los recursos del cluster C1 más potentes que los del cluster C2. Se supone, para el ejemplo, una aplicación formada por tres tareas, lista para ser asignada. En la figura, el eje horizontal representa el tiempo de ejecución, y el eje vertical representa los recursos de procesamiento, con sus respectivas potencias efectivas. Los dos pasos del proceso de asignación son mostrados uno al lado del otro.

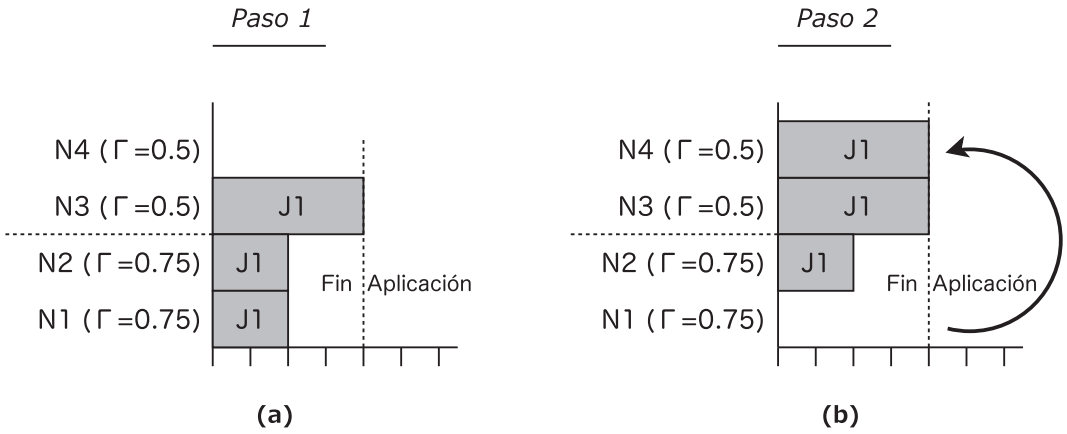


Figura 3.22: Asignación de recursos de MESD

En el primer paso (Figura 3.22(a)), bajo la suposición de minimizar el tiempo de ejecución de la aplicación, se determina la asignación de la aplicación, que para este ejemplo se define como {N1, N2, N3}. Como puede observarse, el tiempo de ejecución viene determinado por el recurso más lento utilizado, que en este caso es N3, con una potencia efectiva $\Gamma_3 = 0{,}5$. Como se puede observar, el uso de los recursos más potentes no implica que el tiempo de ejecución se pueda reducir. En esta situación, el segundo paso (Figura 3.22(b)) re-define la asignación, trasladando una tarea del nodo N1 al nodo N4. Esta nueva asignación no penaliza el tiempo de ejecución de la aplicación, aunque es capaz de liberar un recurso más potente, que estaba siendo infra-utilizado, y que de esta forma podrá ser utilizado por futuras asignaciones.

3.3.3. Ordenación de aplicaciones

Como se ha podido observar en la estrategia *OAS*, la ordenación de las aplicaciones a la hora de planificar su ejecución puede tener un impacto importante en el rendimiento global. En la estrategia *MESD* se propone un mecanismo de ordenación que es capaz de evaluar el conjunto completo de aplicaciones que son evaluadas, de forma que se mejore el rendimiento global. *OAS* decidía la ordenación de las aplicaciones priorizando aquellas aplicaciones favorecían una reducción del makespan. Esto se puede entender como la priorización de aquellas aplicaciones con un menor slowdown de ejecución en cada momento

puntual.

Para lograr esta minimización global con una planificación equitativa para todas las aplicaciones, la solución propuesta consiste en seleccionar, en cada paso del algoritmo, aquella aplicación cuyo slowdown de ejecución sea menor, considerando el estado y disponibilidad de los recursos. Este proceso se resume en los siguientes pasos:

1. *Calcular el slowdown de ejecución.* Se calcula, para cada aplicación, la diferencia en el tiempo de ejecución obtenido, teniendo en cuenta el estado y disponibilidad actual de los recursos, respecto al obtenido si el entorno multi-cluster se considerase en modo dedicado.

2. *Selección de aplicación.* La aplicación cuyo slowdown sea menor, es seleccionada. Si no hubiesen recursos suficientes en el sistema para estimar la asignación de ninguna aplicación, el algoritmo determina aquella aplicación que finaliza su ejecución en primer lugar, libera sus recursos, añadiéndolos al conjunto de recursos disponibles, y vuelve a evaluar las aplicaciones, considerando la nueva situación del entorno.

Este proceso es repetido hasta que todas las aplicaciones de la cola de espera han sido evaluadas. Al modificar el orden de ejecución de las aplicaciones de la cola de espera, se puede producir una situación de *resource starvation*. Esta situación se da cuando un mecanismo de ordenación va priorizando la ejecución las aplicaciones, y algunas de ellas van siendo relegadas sucesivamente, y de forma indefinida, al final de la cola, aumentando enormemente sus tiempos de espera. Mediante el uso de reservas, este tipo de situaciones se pueden evitar [SF05]. *MESD* evalúa el conjunto de aplicaciones como un todo, estableciendo reservas a las aplicaciones una vez se ha determinado su orden de ejecución, y no vuelve a ser llamada hasta que todas las aplicaciones con reserva han sido asignadas. De esta forma se evitan las situaciones que podrían provocar *resource starvation*.

3.3.4. Implementación de la estrategia

El algoritmo principal de la estrategia ha sido implementado tal y como se muestra en el Algoritmo 2. Los parámetros de entrada son: el conjunto de aplicaciones paralelas a tratar y el conjunto de recursos del entorno multi-cluster. El resultado del algoritmo es la planificación del conjunto de aplicaciones, indicando el orden de ejecución de cada una, así como su asignación a los recursos del entorno.

Algoritmo 2 Implementación del algoritmo MESD

```
1: function MainAlgorithm(SJ: Conjunto de aplicaciones, SR: Conjunto de recursos)
2:     for all J in SJ do          //Calcula asignaciones ideales
3:         Ideal_Allocation[J] ← CalculateAllocation(J, SR)
4:     end for
5:     while SJ ≠ ∅ do     // Mientras haya aplicaciones por asignar
6:         min_exec ← ∞
7:         Selected_Job ← NULL
8:         for all J in SJ do      //Calcula asignaciones reales
9:             Allocation[J] ← CalculateAllocation(J, SR)
10:            if Allocation[J] ≠ NULL then      //Si J puede ser asignada
11:                if min_exec < (Allocation[J] − Ideal_Allocation[J]) then
12:                    min_exec ← (Allocation[J] − Ideal_Allocation[J])
13:                    Selected_Job ← J
14:                end if
15:            end if
16:        end for
17:        if Selected_Job = NULL then      //Si ningún job seleccionado
18:            Localizar J' en Scheduling_List que finalice primero
19:            SR ← SR + Allocation[J']      //Libera recursos usados por J'
20:        else
21:            Scheduling_List ← (Selected_Job, Allocation[Selected_Job])
22:            SJ ← SJ − Selected_Job
               //Actualiza disponibilidad de recursos
23:            SR ← SR − Allocation[Selected_Job]
24:        end if
25:    end while
26:    return Scheduling_List
27: end function
```

El algoritmo primeramente calcula la asignación ideal para todas las aplicaciones, asumiendo que el entorno multi-cluster fuera dedicado (lineas 2-4). Estas asignaciones determinarán el límite inferior en tiempo de ejecución para cada una de las aplicaciones paralelas.

A continuación, usando la función $CalculateAllocation(J, SR)$ (linea 9), se calcula la asignación para cada aplicación, considerando la disponibilidad actual de los recursos del sistema. Esta función se detalla en el Algoritmo 3, y devuelve la mejor asignación posible, con el menor número posible de recursos infra-utilizados.

Se calcula para cada aplicación su slowdown de ejecución obtenido en la asignación con el estado actual de los recursos, y la asignación ideal calculada al principio. La aplicación con la menor diferencia, es añadida a la lista ordenada de planificación, se elimina del conjunto de evaluación, y se actualiza la disponibilidad de los recursos teniendo en cuenta los nodos utilizados por la aplicación seleccionada (lineas 21-23).

Algoritmo 3 Implementación de la asignación de recursos de MESD

```
1: function CalculateAllocation(J: Aplicación a tratar, SR: Conjunto
   de recursos, ordenados por potencia efectiva)
2:     n ← número de tareas de J
3:     Allocation[J] ← primeros n nodos de SR
4:     if #Clusters in Allocation[J] > 1 then        //Si co-asignación
5:         Mover tareas de los recursos rápidos, al cluster más lento utilizado.
6:     end if
7:     return Allocation[J]
8: end function
```

En caso de no haber encontrado ninguna aplicación que se pudiera asignar con la disponibilidad de los recursos actuales, el algoritmo estima cual será la primera aplicación en finalizar su ejecución (linea 17), libera los recursos que utilizaba (lineas 18-19), y vuelve a repetir el proceso para localizar la aplicación más adecuada bajo las nuevas condiciones.

Con el objetivo de encontrar los recursos más adecuados con una infra-utilización mínima, se ha implementado el mecanismo de asignación de recursos tal y como se muestra en el Algoritmo 3. El algoritmo tiene como parámetros de entrada la aplicación paralela a tratar, y un conjunto de recursos disponibles,

ordenados por su potencia efectiva. Primero, se calcula el número de tareas requeridas por la aplicación (linea 2). Los primeros n nodos del conjunto de recursos son asignados a la aplicación (linea 3). A continuación, si se ha sido necesaria la utilización de co-asignación (linea 4), es decir, el número de clusters utilizados en la asignación es mayor a 1, las tareas asignadas a los recursos con potencia efectiva más alta son re-asignadas a los recursos utilizados más lentos (linea 5). Finalmente, la asignación definitiva para la aplicación es devuelta (linea 7). En esta implementación se ha priorizado la asignación de tareas en el mismo cluster, minimizando de esta forma la co-asignación y por ende la saturación de los canales de comunicación inter-cluster.

Para ilustrar el funcionamiento de la estrategia, se muestra un ejemplo de la ejecución de los algoritmos. En este ejemplo se asume un único cluster formado por 5 nodos, siendo sus respectivas potencias efectivas: $\Gamma_{N1} = \Gamma_{N2} = 0{,}75$, $\Gamma_{N3} = 0{,}5$, $\Gamma_{N4} = 0{,}25$ y $\Gamma_{N5} = 0{,}15$. El conjunto de aplicaciones que esperan para ser asignadas están detalladas en la Tabla 3.4, donde se muestran sus características principales: número de tareas (τ_j), relación del volumen del procesamiento respecto a la comunicación (σ_j), tiempo base (Tb_j), su tiempo de ejecución considerando la mejor asignación posible, suponiendo que todo el sistema Multi-Cluster estuviera disponible. El ejemplo está construido como iteraciones sobre el algoritmo principal de la estrategia.

Aplicación	τ_j	σ_j	Tb_j	Mejor tiempo de ejecución posible	Asignación ideal
J1	2	0.5	100	116.7s	$N1, N2$
J2	3	0.7	50	100s	$N1, N2, N3$
J3	2	0.5	75	87.5s	$N1, N2$
J4	2	0.7	100	123.33s	$N1, N2$

Tabla 3.4: Conjunto de aplicaciones del ejemplo de MESD

- *Iteración 1*:

  Primero se calculan la asignación ideal y la asignación teniendo en cuenta el estado actual de los recursos. En la Tabla 3.4 aparecen la asignación ideal y el tiempo estimado de ejecución, para cada aplicación del ejemplo

propuesto. Inicialmente todos los recursos se encuentran disponibles, y por lo tanto, el tiempo estimado de ejecución para cada aplicación es el mismo que el producido en la asignación ideal, siendo 0 el slowdown de ejecución para todos los casos. En esta situación, las aplicaciones son evaluadas por orden de llegada, para reducir su tiempo de espera siendo la aplicación seleccionada para asignarse en la primera iteración J1. A continuación, la disponibilidad de los recursos es actualizada, y $\{N1, N2\}$ pasan a estar no disponibles para las asignaciones siguientes.

- *Iteración 2*:

  En esta segunda iteración, únicamente los nodos N3, N4 y N5 están disponibles, y las aplicaciones J2, J3 y J4 se encuentran esperando para ser asignadas. La mejor asignación, que reduce el tiempo de ejecución, es calculada de nuevo para cada una de las aplicaciones, pero teniendo en cuenta el nuevo estado de los recursos del multi-cluster. Los resultados se muestran en la Tabla 3.5. Como se puede observar, la aplicación J3, que es asignada a los nodos N3 y N4, es la que tiene un slowdown de ejecución menor entre su tiempo estimado con los recursos actuales y la asignación ideal. Por lo tanto, J3 es seleccionada para ser asignada, y la disponibilidad de los recursos se actualizan teniendo en cuenta esta asignación.

Aplicación	Tiempo Ejecución	Diferencia respecto ideal	Asignación
J2	258.3s	158.3s	$N3, N4, N5$
J3	**187.5s**	**100s**	**N3, N4**
J4	330s	206.7s	$N3, N4$

Tabla 3.5: Resultados de la segunda iteración. Diferencia entre la asignación ideal y la asignación basada en el estado actual de los recursos.

- *Iteración 3*:

  En la tercera iteración, únicamente el nodo N5 se encuentra disponible. Sin embargo, tanto la aplicación J2 como la aplicación J5 requieren de más nodos para su ejecución. En casos como este, el algoritmo determina

cual será la primera aplicación en finalizar su ejecución, que en el caso de este ejemplo corresponde a J1. Se liberan los nodos ocupados por esta aplicación, N1 y N2, por lo que las aplicaciones J2 y J5 ya pueden ser evaluadas de la misma forma que en la *Iteración 1*. El proceso se repetirá hasta que todas las aplicaciones hayan sido asignas.

El resultado final, con el orden de ejecución y la asignación de las aplicaciones, se muestra en la Figura 3.23. Como puede observarse, el orden en que han de ejecutarse las aplicaciones ha variado respecto al orden de llegada de los mismos. Las aplicaciones J3 y J4 han sido adelantadas, mientras que la aplicación J2 ha sido retrasada.

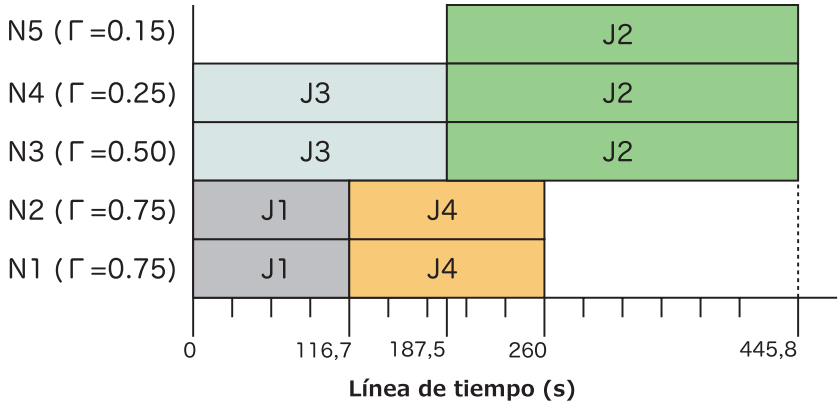


Figura 3.23: Planificación resultante del ejemplo de MESD

A continuación se evalúan de forma experimental las capacidades de la estrategia.

3.3.5. Experimentación

En esta sección se desarrolla un estudio experimental detallado sobre el comportamiento y efectividad de la estrategia *MESD*. Este estudio se divide en dos partes:

1. Comparación de rendimiento respecto a OAS. Para ello, se realiza una comparación de sus resultados con los obtenidos por la estrategia *OAS*, en la que basa su funcionamiento

2. Evaluar la efectividad de la estrategia, comparando sus resultados con los obtenidos por otras estrategias de la literatura: *First Come First Served* (FCFS), *Short Jobs First* (SJF), *Big Jobs First* (BJF), *Fit Processors First Served* (FPFS), *Short Processing Time* (SPT) y *Long Processing Time* (LPT).

Los diferentes algoritmos de la estrategia han sido implementados en el mecanismo de planificación del simulador *GridSim* [TGP], caracterizado para operar como simulador de entorno multi-cluster.

3.3.5.1. Comparación de rendimiento respecto a OAS

Para comprobar que el comportamiento de la estrategia propuesta es similar al de *OAS*, se comparan los resultados obtenidos por *MESD* con los obtenidos por la estrategia *OAS*, además de compararse con los resultados obtenidos por estrategias clásicas presentes en la literatura. Para este apartado se ha optado por utilizar el mismo entorno Multi-Cluster y los mismos workloads utilizados en la experimentación de *OAS* (Sección 3.2.2).

La primera comparativa que realizamos es de los valores del makespan. En la Figura 3.24 se muestran los resultados. En el eje horizontal se representan los tres workloads de la prueba, y en el eje vertical se representa el makespan, medido en segundos. Como se puede observar, los resultados obtenidos por *MESD* son muy cercanos a los obtenidos por *OAS*, teniendo ambos un comportamiento similar, aunque siendo mejores los resultados obtenidos por *OAS*, al proporcionar soluciones óptimas. Esto es debido a que *OAS* realiza una búsqueda exhaustiva del espacio de soluciones para proporcionar una solución óptima, según las condiciones del problema, y por lo tanto ha hecho avanzar aplicaciones que la estrategia *MESD* no ha hecho avanzar. Cabe destacar que *MESD* obtiene en general unos valores de makespan un 15 % más bajos que los obtenidos por las estrategias clásicas. Esto significa, que la planificación proporcionada por *MESD* permite finalizar la ejecución del conjunto de aplicaciones en un menor tiempo.

La característica que se ha evaluado a continuación es la utilización de los recursos del sistema por las diferentes estrategias. Esto se ha hecho a partir

del grado de compactación del workload (presentado en la Sección 3.1.3), que mide el porcentaje de tiempo en el que los recursos han estado ocupados, y se muestran los resultados en la Figura 3.25. Como se puede observar, los resultados obtenidos por *OAS* y *MESD* siguen una tendencia similar, siendo *OAS* la estrategia que obtiene un valor más elevado, y *MESD* entre las mejores.

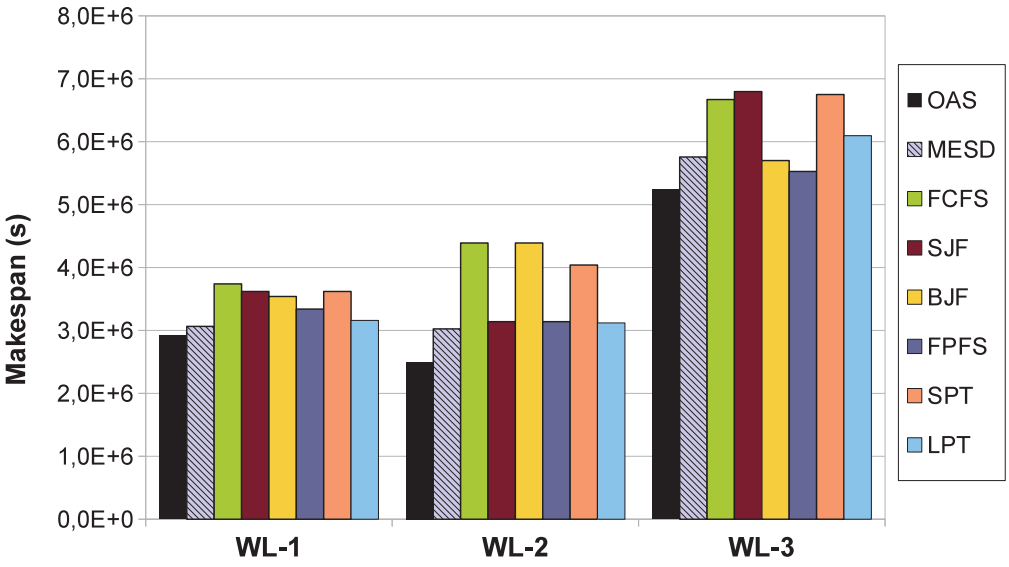


Figura 3.24: Comparativa del makespan

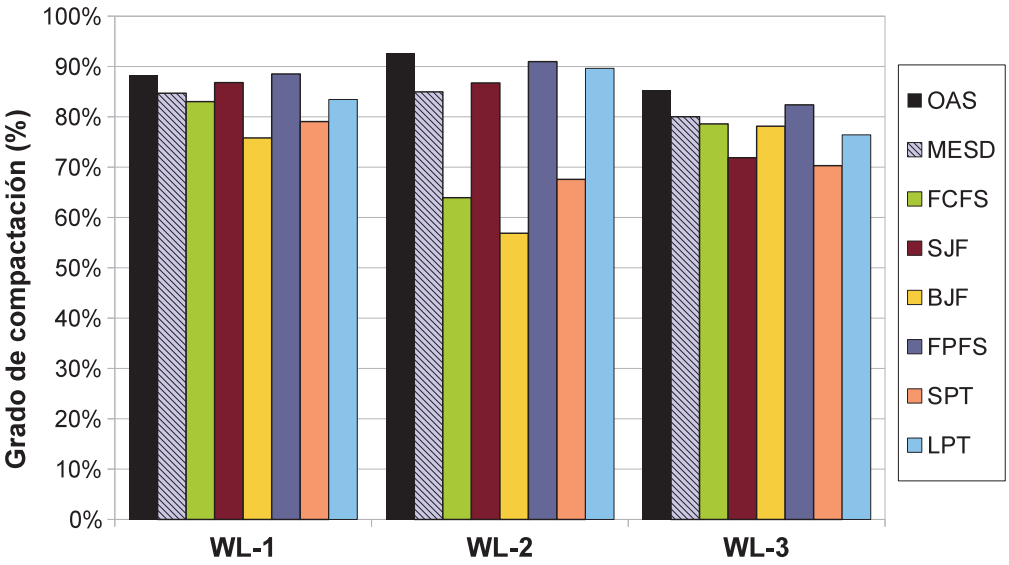


Figura 3.25: Comparativa del grado de compactación

A continuación, se compara el impacto de las asignaciones sobre el tiempo de ejecución. Para ellos se calcula el slowdown, o retraso promedio que han sufrido las aplicaciones en su tiempo de ejecución. Los resultados se muestran en la Figura 3.26. En este caso se puede observar como en general los resultados obtenidos por *MESD* son mejores que los obtenidos por *OAS*. Esto se debe a que a diferencia de *OAS*, la estrategia *MESD* no busca la solución óptima a una función objetivo, sino que al tratarse de una heurística *ad-hoc*, sigue la ejecución de unos algoritmos. En el caso de *MESD*, la priorización de las aplicaciones es en base a la minimización del slowdown de ejecución, y este efecto se ha podido observar en esta comparativa. En el workload WL-3 se observa como algunas de las estrategias clásicas obtienen mejores resultados. Esto es debido a que han podido determinar un orden de ejecución que ha reducido el retraso en el tiempo de ejecución, pero que sin embargo, han implicado un makespan más elevado (Figura 3.24).

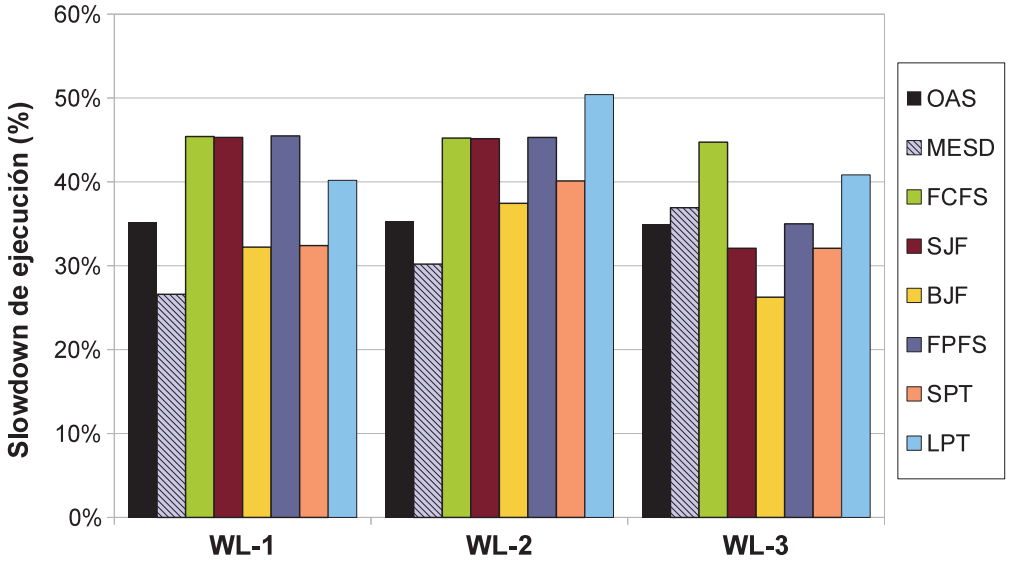


Figura 3.26: Comparativa del slowdown de ejecución

Por último se evalúa el tiempo de respuesta promedio, que se compone del tiempo de ejecución y del tiempo de espera. Los resultados de la comparación se muestran en la Figura 3.27. Como puede observarse, *OAS* y *MESD* son las estrategias que obtienen en general mejores resultados, tanto en tiempo de

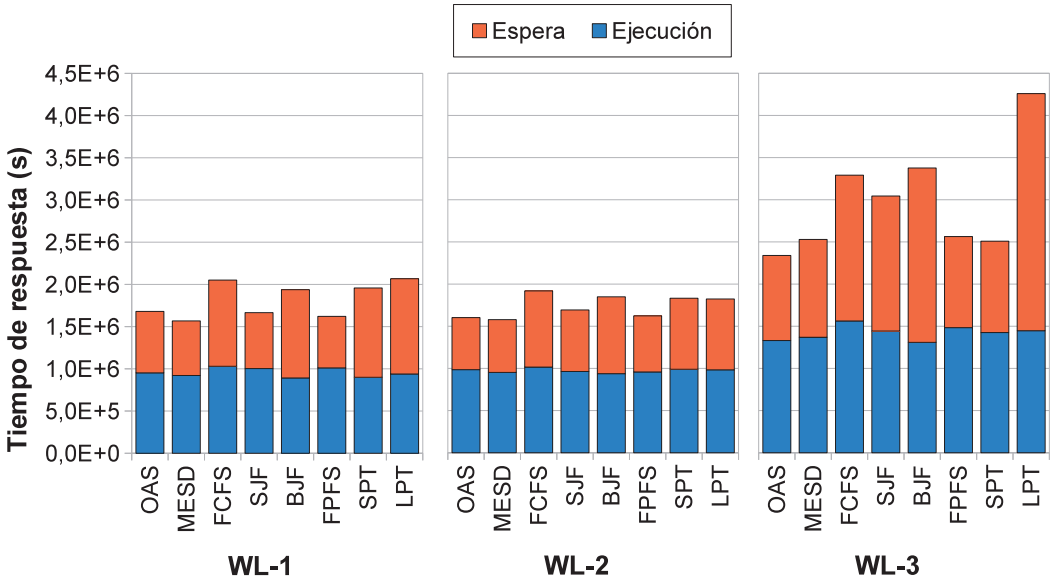


Figura 3.27: Comparativa del tiempo de respuesta

espera como en tiempo de ejecución. Además, la tendencia de ambas estrategias es muy similar

Teniendo en cuenta los resultados que se han analizado, consideramos que el comportamiento de la estrategia *MESD* es el esperado: Obtiene en general buenos resultados en las diferentes comparativas, y la tendencia de los mismos sigue la tendencia de los resultados de la estrategia *OAS*, de la que basa parte de su funcionamiento. A pesar de ser similares, los resultados de *OAS* son mejores, al realizar esta estrategia una búsqueda de la solución óptima. Sin embargo, *MESD* tiene como punto positivo su reducido coste computacional, lo que le permite obtener una solución en un tiempo razonablemente inferior. Este aspecto es importante de cara a la posible implementación de la estrategia *MESD* en un scheduler on-line en el que van entrando aplicaciones en el sistema cada pocos segundos, y se necesita determinar una planificación de forma prácticamente inmediata.

3.3.5.2. Evaluación del rendimiento sobre workloads reales

Una vez *MESD* ha sido validada respecto a la estrategia en la que está basada, *OAS*, se desarrolla un estudio experimental para evaluar el rendimiento

de la misma respecto a estrategias clásicas de la literatura. Debido que la complejidad computacional de la estrategia *MESD* es mucho más reducida que la de la estrategia *OAS*, este análisis puede ser llevado a cabo usando un entorno con un tamaño representativo de un Multi-Cluster real.

El sistema Multi-Cluster ha sido diseñado para estar formado por cuatro clusters heterogéneos, conectados al switch central mediante un enlace Gigabit. Cada cluster está compuesto por 60 nodos homogéneos dentro del mismo cluster, y está caracterizado tal y como se muestra en la Tabla 3.6.

	Num. Nodos	Potencia efectiva
Cluster 1	60	1.0
Cluster 2	60	1.5
Cluster 3	60	2.0
Cluster 4	60	1.0

Tabla 3.6: Caracterización del entorno multi-cluster del experimento

Para la realización del estudio, se ha diseñado un conjunto de tres workloads, creados a partir de las trazas reales del super-computador *HPC2N* [wor]. Cada uno de los workloads está formado por unas 15.000 aplicaciones. La mayoría de las aplicaciones de estos workloads son de cómputo intensivo, con un tiempo base promedio de 15.520 segundos, y un promedio de 9 tareas por aplicación.

La caracterización detallada, representativa para los tres workloads, se muestra en la Figura 3.28. En la figura se muestra el histograma de los tiempos base, del número de tareas de que se componen las tareas, así como de los requisitos de comunicación. Además. se muestra la función de distribución de los tiempos de llegada de las aplicaciones, que sigue un comportamiento en el que existen intervalos de tiempo donde se registran pocas llegadas de nuevas aplicaciones y se restringe a otros tasas de llegada más altas. Así mismo, como se ha podido observar en otros centros de cómputo, un número destacable de aplicaciones serie son ejecutadas en él. La llegada de las aplicaciones sigue un comportamiento en el que cada poco tiempo nuevas aplicaciones entran en el sistema, dándose en momentos puntuales un ritmo de llegada más alto. Esto provoca que el sistema se encuentre más congestionado, y más aplicaciones

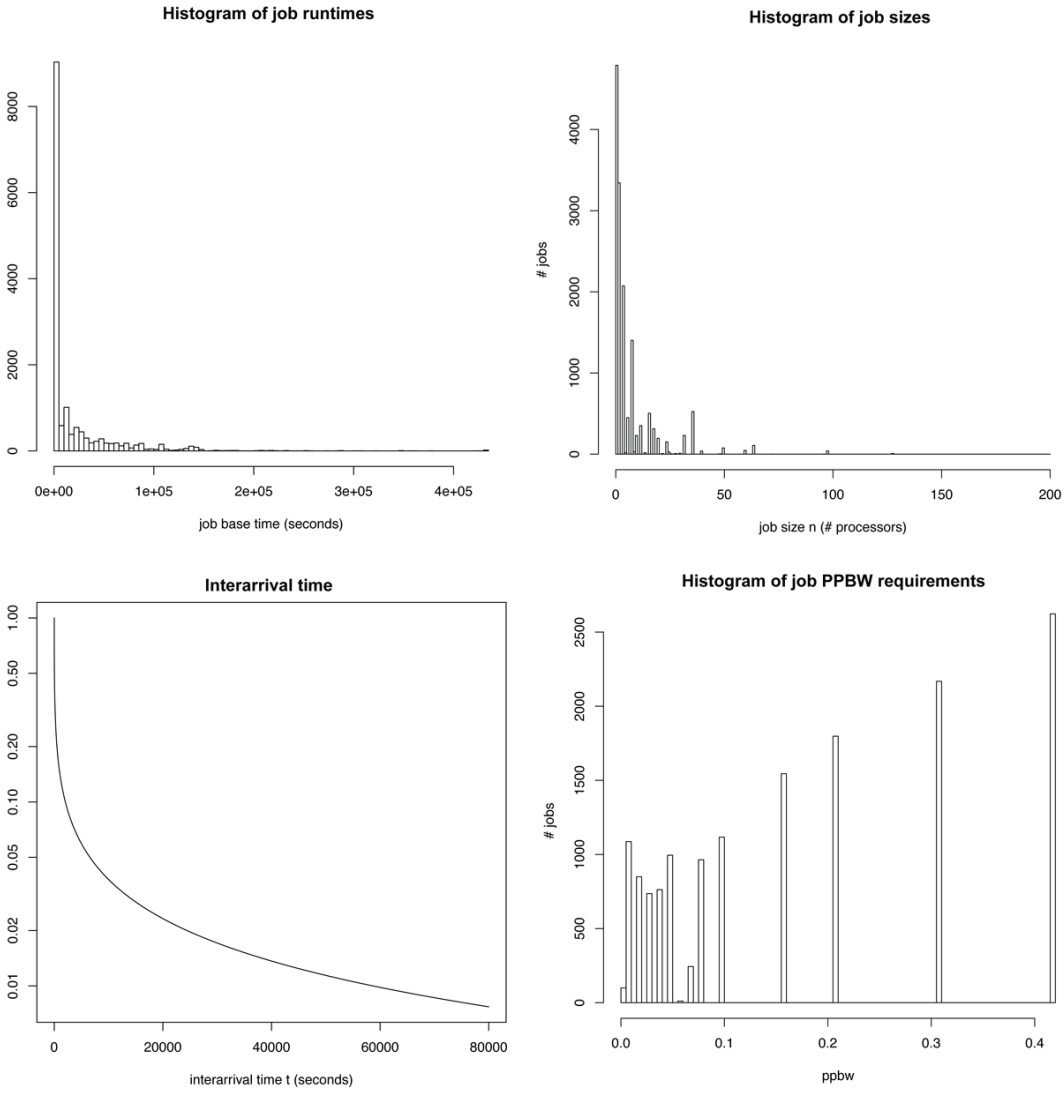


Figura 3.28: Características del workload

se acumulen en la cola de espera. Por último, se puede observar un reparto relativamente equitativo respecto a los requisitos de ancho de banda de las aplicaciones, gran parte de ellas teniendo valores por debajo de los 0.1 GB/s.

La comparativa de los resultados del makespan para las diferentes estrategias se presenta en la Figura 3.29. El eje horizontal representa los tres workloads del experimental, y el eje vertical representa el makespan, medido en segundos. Como se puede observar, las diferencias entre las diferentes estrategias es mínima. Esto se debe a que en workloads tan grandes, donde las aplicaciones van llegando en diferentes tiempos, es común que en la parte final lleguen aplicaciones que hagan retrasar el makespan. Esto enmascara los efectos que las mejoras en tiempos de ejecución puedan haber provocado respecto a esta métrica. Por lo tanto, es necesario evaluar el rendimiento de las aplicaciones examinando los resultados respecto a otras métricas.

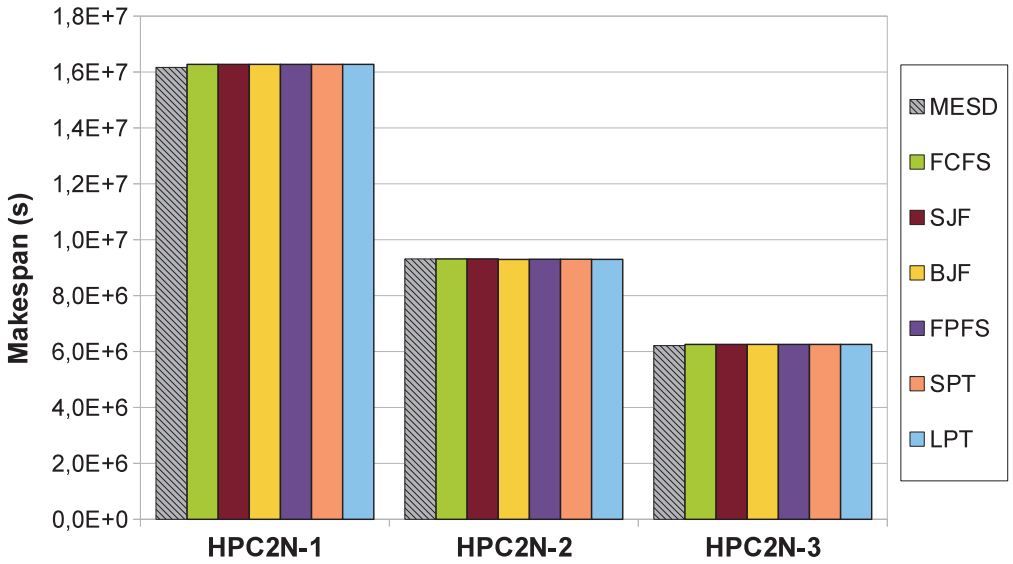


Figura 3.29: Comparativa del makespan

A continuación se evalúa el tiempo de respuesta promedio de las aplicaciones, que contiene información sobre los tiempos de ejecución y de espera promedios. Estos resultados comparativos se muestran en la Figura 3.30. Como se puede observar, *MESD* es capaz de reducir significativamente los tiempos de ejecución promedio de las aplicaciones, gracias a su habilidad para determinar

un orden de ejecución que reduzca el slowdown de ejecución, y a su mecanismo de asignación que utiliza aquellos nodos más adecuados para cada aplicación, liberando aquellos más potentes que no le implican una mejora de rendimiento y que pueden ser aprovechados por otras aplicaciones. Esto también tiene un efecto sobre el tiempo de espera. Como se puede observar, las estrategias *SJF*, *FPFS* y *SPT* obtienen los mejores tiempos de espera, debido a que son capaces de avanzar rápidamente las aplicaciones más pequeñas, y en el otro extremo, *FCFS*, *BJF* y sobre todo *LPT* obtienen unos tiempos de espera muy elevados, debido a que no son capaces de hacer avanzar rápidamente aplicaciones y que se liberen recursos para iniciar lo antes posible la ejecución de otras que están en espera. La estrategia *MESD* es capaz de obtener unos tiempos de espera, que a pesar de no ser los más bajos, sí que están entre los mejores. Estos tiempos de espera junto con los tiempos de ejecución obtenidos, permiten que el tiempo de respuesta promedio esté entre los mejores resultados de la comparativa.

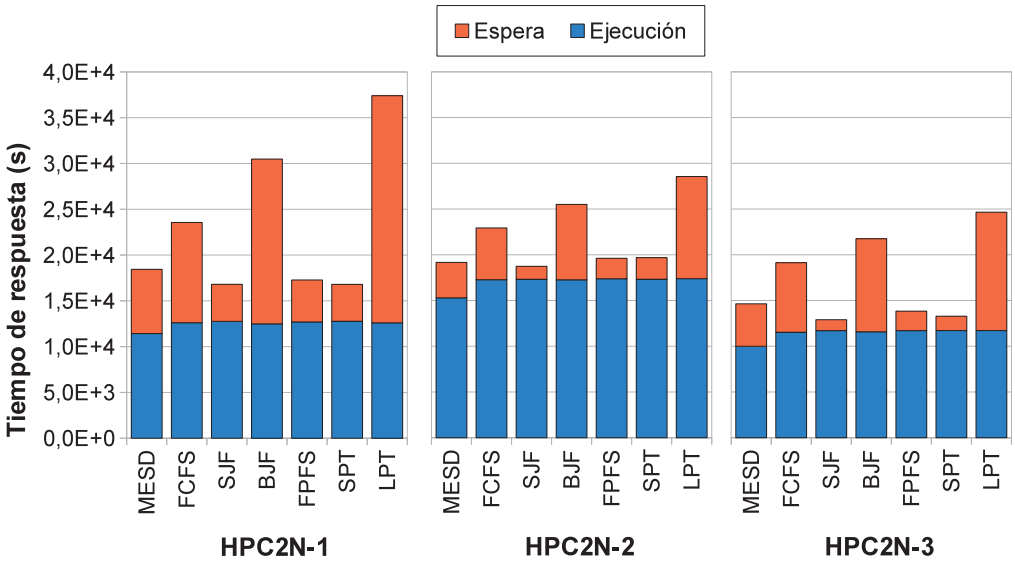


Figura 3.30: Comparativa del tiempo de respuesta

Como ha sido comentado, *MESD* determina el orden de ejecución de las aplicaciones priorizando en cada momento puntual a aquella aplicación que tenga un slowdown de ejecución más bajo con los recursos disponibles. El efecto del mecanismo de asignación de recursos, pero sobre todo de determinación

del orden de ejecución, puede evaluarse para el conjunto de las aplicaciones, comparando el slowdown de ejecución promedio. En la Figura 3.31 se muestran estos resultados. Debido a que en el entorno experimental se han utilizado potencias efectivas iguales y superiores a 1 para los diferentes clusters, significa que las aplicaciones podrán ejecutarse más rápido que en el entorno dedicado de referencia que se ha utilizado para determinar su tiempo base (Tb_j). Esto implica que los retrasos de las aplicaciones podrán ser negativos. Esta métrica normalizada se ha representado en el eje vertical. Como se puede observar, las estrategias clásicas han obtenido unos resultados similares, siendo *BJF* la estrategia clásica que mejores valores de slowdown obtiene, aunque hay que recordar que por contra obtenía unos tiempos de espera promedios elevados. Sin embargo, gracias a sus mecanismos de asignación y en este caso, particularmente el de ordenación, *MESD* ha conseguido que sus planificaciones mejoren significativamente el slowdown de ejecución promedio de las aplicaciones.

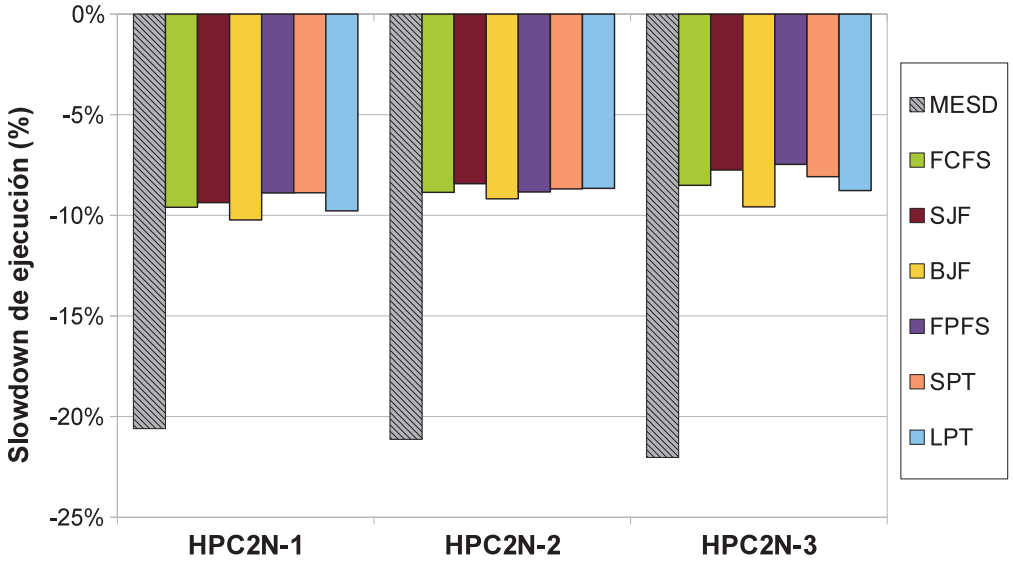


Figura 3.31: Comparativa del slowdown de ejecución

El análisis de los slowdown de ejecución promedio es una forma interesante de evaluar el efecto que la forma de planificar de las diferentes estrategias ejercen sobre la ejecución de las aplicaciones. Sin embargo, para poder obtener alguna información acerca de como las diferentes estrategias tratan el problema

de la saturación de los recursos, es necesario realizar un análisis más detallado de estos slowdowns. Debido al elevado número de aplicaciones que componen los workloads, y a que una parte de ellas tienen un tiempo base reducido, hemos decidido centrar el análisis en las 5.000 aplicaciones con un tiempo base más elevado, ya que es en las aplicaciones de mayor duración donde el retraso puede tener una gran importancia. Se ha tomado uno de los workloads como referencia, y se comparan los resultados obtenidos por *MESD* con los obtenidos por la estrategia que mejores resultados de slowdown ha obtenido, que en los tres casos ha sido *BJF*.

Los resultados de este análisis en detalle se muestra en la Figura 3.32. En ella se muestran en el eje horizontal los rangos de slowdown de ejecución, medidos como un porcentaje, y en el rango vertical el número de aplicaciones que, asignadas por cada una de las dos estrategias, su slowdown está dentro de ese rango.

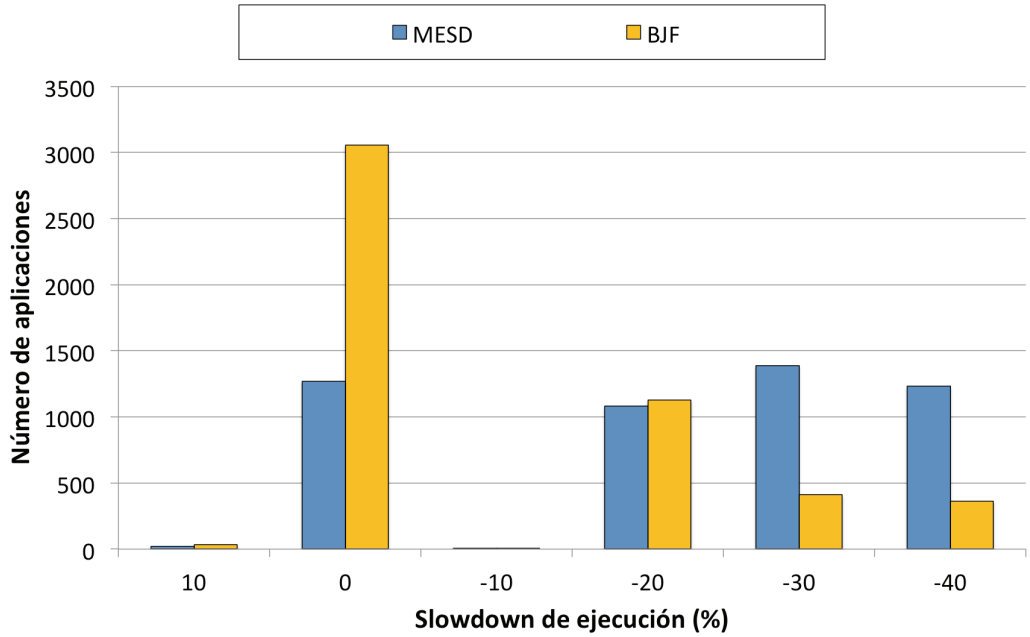


Figura 3.32: Frecuencia en slowdown de ejecución en las 5.000 aplicaciones con mayor tiempo base

Como se puede observar, las asignaciones que ha producido la estrategia *BJF* han ralentizado en general más a las aplicaciones, teniendo la mayoría de ellas un slowdown elevado. En el caso contrario, *MESD* ha sido capaz de

realizar asignaciones de forma que los slowdowns son más equilibrados, siendo mayor el número de aplicaciones que tienen un slowdown de ejecución más reducido.

Adicionalmente, se ha evaluado como se han visto afectadas las diferentes estrategias por la saturación de los canales de comunicación. Para realizar esto, se ha calculado el número de aplicaciones, del total de 15.000 del workload, que han sufrido slowdown de ejecución provocado por la saturación de los canales. Además se muestra el número de aplicaciones que han sido co-asignadas. Los resultados de ésta comparación se muestran en la Tabla 3.7.

Estrategia	Aplicaciones que han saturado (%)	Aplicaciones co-asignadas (%)
MESD	0.30 %	15.55 %
FCFS	1.95 %	19.79 %
SJF	1.85 %	21.17 %
BJF	0.70 %	17.81 %
FPFS	0.25 %	20.39 %
SPT	0.20 %	19.33 %
LPT	2.53 %	18.85 %

Tabla 3.7: Comparación del efecto de la saturación sobre las aplicaciones

Como se puede observar en la tabla, *MESD* ha obtenido unos resultados que se encuentran entre los mejores. A pesar de que estrategias como *SPT* y *FPFS* han obtenido unos resultados ligeramente mejores, hay que tener en cuenta que el porcentaje de aplicaciones que han co-asignado es superior, y que los valores de slowdown de ejecución que han obtenido, eran peores que los obtenidos por *MESD*. Sin embargo, la estrategia *MESD* ha sido la estrategia con un número más reducido de aplicaciones co-asignadas. El hecho de mover tareas de los recursos más potentes a otros más lentos no implica una pérdida de rendimiento, consigue liberar recursos para puedan ser aprovechados por otras aplicaciones. Pero además, reduce el uso de los canales de comunicación, ya que se intenta agrupar las tareas de una aplicación en un mismo cluster, y como se ha podido observar, tiene un efecto positivo de cara a reducir el grado de saturación de los canales.

Por último se analiza el comportamiento de las diferentes estrategias en cuanto a la utilización del sistema. A diferencia de los otros experimentos del presente trabajo, en este workload, que está formado por un número elevado de aplicaciones, el grado de compactación no resulta una métrica adecuada para evaluar la utilización de los recursos. Esta métrica va muy ligada al makespan, y como se ha observado en la comparativa correspondiente, los valores obtenidos por las diferentes estrategias son muy similares. Sin embargo, la utilización de los recursos se puede analizar directamente contabilizando el tiempo total en que los diferentes recursos de procesamiento han estado ejecutando.

Los resultados de este estudio se muestran en la Figura 3.33. En el eje horizontal se muestran los 240 nodos del sistema multi-cluster, ordenados de menor a mayor potencia efectiva, y agrupados en función del cluster al que pertenecen, mostrándose en cada división la potencia efectiva del correspondiente cluster. En el eje vertical, se representa el tiempo que cada nodo ha estado procesando.

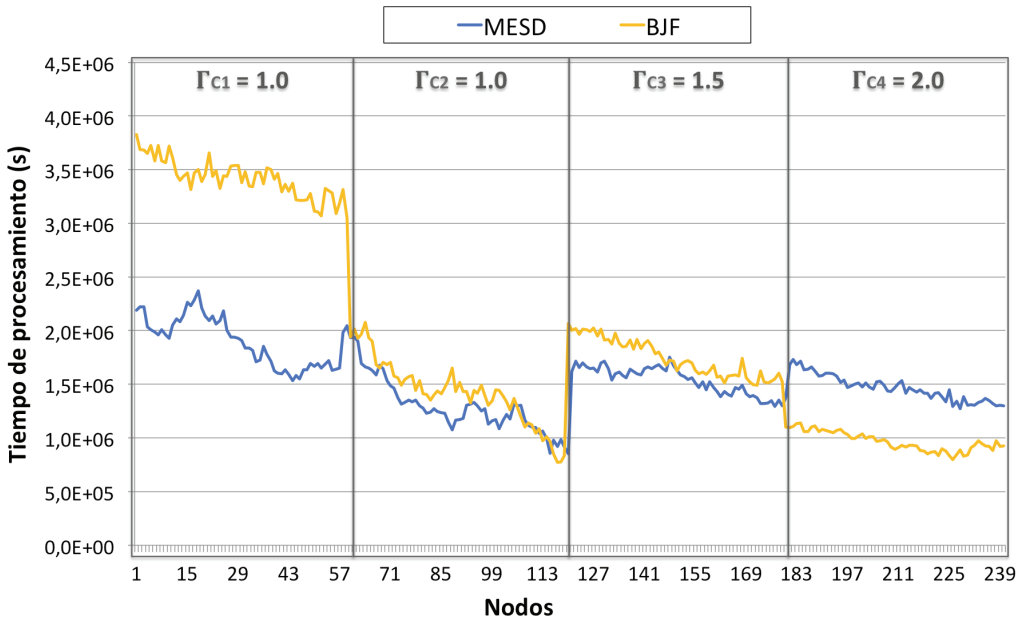


Figura 3.33: Relación de la utilización de los nodos de cómputo

Como se puede observar, *BJF* utiliza en general mucho más los nodos del primer cluster, mientras que utiliza en mucha menor medida los nodos del cluster más potente, con potencia efectiva 2.0. En el caso de la estrategia

MESD, gracias a su habilidad para ordenar los recursos, y luego liberarlos, es capaz de realizar una utilización mucho más homogénea de los recursos de los diferentes clusters. Con esto, se puede observar como *MESD* ha realizado un balanceo de carga, de forma que los recursos más potentes compensan el tiempo de procesamiento de los recursos con menor potencia, y se realiza un mejor aprovechamiento de los recursos del sistema Multi-Cluster.

3.3.5.3. Conclusiones

Se ha presentado *Minimum Execution Slowdown* (MESD), una estrategia consistente en una heurística, que es capaz de determinar el orden de ejecución y la asignación a recursos de un conjunto de aplicaciones, evaluándolos de forma simultánea. El funcionamiento de la estrategia se basa en los patrones de funcionamiento observados en la estrategia *OAS*, basada en un modelo MIP. Estos patrones se corresponden con la priorización de las aplicaciones intentando que su slowdown de ejecución sea reducido, y el de tratar de liberar recursos potentes si otra aplicación puede aprovecharlos. Teniendo esto en cuenta, la estrategia *MESD* asigna las aplicaciones de forma que en un primer paso se asignan los mejores recursos de procesamiento, y en un segundo paso se mueven tareas de los recursos más potentes al cluster utilizado más lento, siempre que esto no implique un empeoramiento del rendimiento. Con este movimiento de tareas se consigue que aquellos recursos potentes que son utilizados, pero que no implican una mejora del rendimiento, queden libres para otras aplicaciones puedan aprovecharlos. El orden de ejecución de las aplicaciones es determinado teniendo en cuenta las asignaciones de todas las aplicaciones en un momento puntual, de forma que se ordenan de forma que en cada momento se asigne aquella aplicación que sufra un retraso menor en su tiempo de ejecución. De esta forma, se pretende que las planificaciones que proporcione la estrategia *MESD* puedan mejorar el rendimiento de las aplicaciones, realizando además un mejor aprovechamiento de los recursos del sistema.

Se ha desarrollado un conjunto de experimentos para validar el correcto comportamiento de la estrategia. Para ello, se han comparado los resultados obtenidos por la estrategia *MESD* con los obtenidos por *OAS* y otras estrategias clásicas de la literatura. Debido a la elevada complejidad computacional

de la estrategia *OAS*, se ha utilizado un entorno experimental de tamaño reducido. Analizando los resultados obtenidos, se ha observado como los resultados obtenidos por *MESD* siguen la tendencia de comportamiento de los obtenidos por *OAS*, y superan los resultados obtenidos por las estrategias clásicas.

El rendimiento de la estrategia ha sido analizado comparando los resultados obtenidos por *MESD* y otras estrategias clásicas, sobre un entorno experimental diseñado a partir de un super-computador real. Los resultados obtenidos han mostrado que *MESD* es capaz de proporcionar planificaciones que mejoran significativamente el tiempo de ejecución y de espera promedios, resultando en conjunto en unos buenos tiempos de respuesta promedio. Además, es capaz de mejorar significativamente el slowdown que sufren las aplicaciones.

Se ha analizado también el retraso de ejecución en detalle para las diferentes aplicaciones, y se ha determinado como *MESD* es capaz de reducir el numero de aplicaciones que se ven afectadas, y que ven retrasado su tiempo de ejecución, debido a la saturación de los canales de comunicación. Finalmente, se ha observado como las planificaciones proporcionadas por *MESD* realizan un aprovechamiento mucho más homogéneo de los recursos que el resto de estrategias evaluadas.

Capítulo 4

Conclusiones

4.1. Conclusiones

En el presente trabajo de tesis se ha abordado el problema de la planificación on-line de múltiples aplicaciones paralelas en entornos Multi-Cluster heterogéneos y con co-asignación. Para ello, se han propuesto nuevas técnicas que tratan tanto la agrupación de las aplicaciones como su asignación, considerando las características de los recursos del sistema, así como los requisitos del grupo de aplicaciones en cuanto al cómputo y la comunicación.

Para poder aprovechar adecuadamente los recursos de un entorno Multi-Cluster es necesario desarrollar técnicas de planificación que tengan en cuenta los aspectos mencionados anteriormente. Habitualmente las técnicas de planificación han evaluado las aplicaciones paralelas una a una, sin tener en cuenta los requisitos de las otras aplicaciones que puedan estar en la cola de espera del sistema. Algunos estudios han tratado la asignación de conjuntos de aplicaciones, y han concluido que se pueden obtener mejoras en el rendimiento global del conjunto de aplicaciones, y mejorar la utilización de los recursos del sistema. Sin embargo, estos estudios usualmente han asumido aplicaciones compuestas por tareas independientes, sin considerar las comunicaciones.

En una primera propuesta, partiendo del modelo MIP de planificación propuesto por J. L. Lérida, se ha presentado una estrategia, llamada *Package Allocation Strategy* (PAS), con la capacidad de evaluar un conjunto de aplicaciones para su planificación. La estrategia se compone de dos pasos: la selección

de un conjunto de aplicaciones para ser asignadas, y la asignación óptima a los recursos de ese conjunto. La función de selección de aplicaciones permite la utilización de diferentes técnicas clásicas de ordenación, como FCFS, SJF, FPFS etc. La asignación óptima del conjunto de aplicaciones en los recursos disponibles del sistema Multi-Cluster es determinada por un modelo MIP, teniendo en cuenta las características del sistema y los requisitos de las aplicaciones, tanto de cómputo como de comunicación.

En los resultados experimentales se ha observado la importancia del efecto de la heterogeneidad de los recursos de procesamiento y de la carga de las comunicaciones en el proceso de la asignación de aplicaciones. La capacidad de *PAS* de tener en cuenta tanto las características de los recursos de procesamiento como de comunicaciones, ha permitido la mejora del makespan y del tiempo de respuesta promedio de las aplicaciones, independientemente de las condiciones del entorno Multi-Cluster.

Las aportaciones realizadas en este apartado aparecen publicadas en:

- H. Blanco, D. Castellà, J. L. Lérida, F. Guirado, *Multiple Job Allocation in Multicluster System*, VECPAR 2010, 9th International Meeting, Poster Briefing, Electronic Ed., Junio 2010

- H. Blanco, A. Montañola, F. Guirado and J. L. Lérida, *Fairness Scheduling for Multi-Cluster Systems Based on Linear Programming*, CMMSE'10: 10th International Conference Computational and Mathematical Methods in Science and Engineering, pp. 227-239, Julio 2010

- H. Blanco, F. Guirado, J. L. Lérida, *Impact of Integer Programing on multiple-job scheduling in heterogeneous multi-cluster environments*, exposición en el 2011 CYTED-HAROSA Workshop, 2011

- H. Blanco and J. L. Lerida and F. Cores and F. Guirado, *Multiple Job Co-Allocation Strategy for Heterogeneous Multi-Cluster Systems Based on Linear Programming*, Journal of Supercomputing, 58(3), pp. 394-402, 2011

En segundo lugar, debido al hecho de que la estrategia *PAS* está restringida a tratar aquellas aplicaciones cuyo número total de tareas es inferior al de

recursos disponibles en el sistema, se ha propuesto una nueva estrategia denominada *Ordering and Allocation Strategy (OAS)*, que mediante un modelo MIP es capaz de determinar el orden de ejecución y la asignación de recursos a un conjunto de aplicaciones, evaluándolos de forma simultánea. La estrategia tiene en cuenta los requisitos de procesamiento y de comunicación, y evita aquellas soluciones en las que se provocaría una saturación en los canales de comunicación. Esta técnica supera la restricción principal de la estrategia *PAS*, la cual únicamente podía evaluar subconjuntos de aplicaciones cuyo número total de tareas fuese inferior al número de nodos disponibles en el sistema.

OAS implementa una representación del tiempo mediante slots, que permite evaluar las interacciones entre las distintas aplicaciones. El uso de slots para representar el tiempo en el modelo es un parámetro crítico y por ello, se ha llevado a cabo un análisis para determinar el impacto que podría tener estos sobre la calidad de las soluciones y la complejidad del problema. Se ha determinado que un tamaño de slot del 50 % del tiempo base supone un compromiso entre la calidad de la solución y el tiempo necesario para obtenerla. Se han comparado los resultados obtenidos mediante diferentes workloads con *PAS*, además de con otras estrategias presentes en la literatura. Analizando los resultados, se ha observado que la estrategia *OAS* es capaz de producir soluciones con mejores resultados en cuanto a la minimización del makespan, resultando soluciones más compactas, y que provocan un mejor aprovechamiento de los recursos del sistema. Las soluciones proporcionadas por *OAS* han reducido el slowdown de ejecución de las aplicaciones, lo que ha provocado una mejora en su tiempo de respuesta.

Las aportaciones realizadas en este apartado aparecen publicadas en:

- H. Blanco, F. Guirado, J. L. Lérida, V. M. Albornoz, *MIP Model Scheduling for Multi-Clusters*, 10th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms, HeteroPar'2012, Aceptado, pendiente de publicación, Agosto 2012

- H. Blanco, F. Guirado, J. L. Lérida, V. M. Albornoz, *OAS: A MIP Model for Ordering and Allocating Parallel Jobs on Multi-Cluster Systems*, XVI Congreso Latino-Iberoamericano de Investigación Operativa

CLAIO, Aceptado, pendiente de publicación, Septiembre 2012

- H. Blanco, F. Guirado, J. L. Lérida, V. M. Albornoz, *MIP Model Scheduling for BSP Parallel Applications on Multi-Cluster Environments*, 3PGCIC'12: 7th International Conference on P2P, Parallel, GRID and Cloud Internet Computing, Aceptado, pendiente de publicación, Noviembre 2012

Finalmente, y con el objetivo de buscar una alternativa a la estrategia *OAS* capaz de proponer una solución en tiempos más asequibles, adecuados para entornos reales de producción, se ha presentado *Minimum Execution Slowdown* (MESD), una estrategia heurística con la capacidad de determinar el orden de ejecución y la asignación a recursos de un conjunto de aplicaciones, evaluándolos de forma simultánea, obteniendo la solución en un tiempo reducido. La estrategia *OAS* prioriza las aplicaciones de forma que su slowdown de ejecución sea reducido de forma que el makespan sea minimizado, y tratando de liberar recursos potentes si otra aplicación puede aprovecharlos. La estrategia *MESD* ha sido diseñada para tener en cuenta el comportamiento de *OAS*.

Teniendo esto en cuenta, la estrategia *MESD* asigna las aplicaciones de forma que en un primer paso se asignan los mejores recursos de procesamiento, y en un segundo paso se mueven tareas de los recursos más potentes al cluster utilizado más lento, siempre que esto no implique un empeoramiento del rendimiento. Con este desplazamiento de tareas se consigue que aquellos recursos potentes que son utilizados, pero que no implican una mejora del rendimiento, queden libres para otras aplicaciones puedan aprovecharlos en futuras asignaciones.

Se han realizado simulaciones con distintos tipos de workload obtenidos a partir de trazas de entornos reales. Se han comparado el uso de la nueva técnica heurística, *MESD*, con la técnica exacta anteriormente propuesta, *OAS*, además de con técnicas comúnmente utilizadas en la literatura. Analizando los resultados, se puede observar que *MESD* es capaz de proporcionar planificaciones que mejoran significativamente el slowdown de ejecución, mejorando los tiempos de respuesta. Además, la estrategia proporciona asignaciones en las que se mejora la utilización de los recursos del sistema.

Las aportaciones realizadas en este apartado aparecen publicadas en:

- *Ordering and Allocating Parallel Jobs on Multi-Cluster Systems*, CMMSE'12: 12th International Conference Computational and Mathematical Methods in Science and Engineering, pp. 196-206, Julio 2012

4.2. Trabajo futuro

Estudiando los resultados obtenidos por las diferentes estrategias propuestas, y a medida que este trabajo de tesis se ha ido desarrollando, han ido surgiendo las siguientes nuevas líneas de interés que pueden contribuir en la mejora de la planificación de aplicaciones paralelas en entornos Multi-Cluster heterogéneos:

- El coste computacional del modelo MIP propuesto en la estrategia *OAS* ha demostrado ser muy elevado. Una alternativa al uso de un solver MIP podría ser la utilización de *Metaheurísticas* para optimizar el tiempo de búsqueda del problema, tales como algoritmos de búsqueda basados en trayectoria, como son *Búsqueda Tabú* (TS) [Glo77], *Simulated Annealing* (SA) [KGV83], *Variable Neighborhood Search* (VNS) [Mla97], y *Iterated Local Search* (ILS) [LMS02], o *Metaheurísticas* basadas en poblaciones, como *Scatter Search* (SS) [Glo98], *Estimation of Distribution Algorithm* (EDA) [Müh98], *Particle Swarm Optimization* (PSO) [Ken99] y *Ant Colony Optimization* (ACO) [DS03]. En algunos estudios se han propuesto soluciones híbridas, en las cuales se han combinado heurísticas basadas en algoritmos de búsqueda local con métodos heurísticos constructivos [FR99], y algoritmos basados en teoría de juegos con *metaheurísticas* basadas en algoritmos genéticos [XK10].

- Las estrategias propuestas en este trabajo de tesis operan sobre las aplicaciones que se encuentran en el sistema. Sin embargo, en sistemas dinámicos donde nuevas aplicaciones van llegando cada cierto tiempo, podría resultar de gran interés tomar en consideración también las posibles aplicaciones que podrían ir llegando al sistema. Por otro lado la ocupación de los recursos en entornos Multi-Cluster en ocasiones puede compartirse con usuarios locales. En estas situaciones, la ocupación suele ser un factor dinámico. Los *procesos estocásticos* [Doo53] es una rama de las matemáticas que trata sobre los diferentes caminos que puede tomar el sistema a lo largo del tiempo, como evolucionará su estado, en base a cálculos probabilísticos y a datos históricos sobre el sistema. En base a cálculos probabilísticos se podrían definir diferentes posibles estados del

sistema, y tratar de solucionar la planificación de las aplicaciones considerando la disponibilidad futura de los recursos y/o las características de las aplicaciones.

El uso de procesos estocásticos supone un coste computacional aún más elevado, pero en los últimos años se han propuesto técnicas para solucionar modelos estocásticos de programación entera [AAEG+03, AKP03, EGMP07] en tiempos más razonables.

- Los criterios utilizados en la estrategia *MESD* a la hora de realizar la planificación de un conjunto de aplicaciones paralelas, teniendo en cuenta la reducción del slowdown de ejecución, y el balanceo de carga con el fin de mejorar la utilización de los recursos, ha permitido obtener buenos resultados en los diferentes estudios experimentales. Sin embargo, se podría estudiar la incorporación de otros criterios en la heurística, así como el realizar refinamientos sobre los ya existentes, y que podrían suponer mejoras en los resultados. Algunos criterios que se podrían considerar serían la priorización de las aplicaciones tratando de reducir su tiempo de respuesta, que considera la espera y la comunicación, o tratar de priorizar las aplicaciones paralelas de forma que se agrupen aquellas que tengan características similares.

- Los estudios experimentales desarrollados sobre la estrategia *MESD* han mostrado que ésta puede trabajar con conjuntos elevados de aplicaciones. Se han utilizado varios workloads extraídos de trazas de entornos de cómputo reales. Sería interesante ampliar estos estudios, con un abanico mayor de workloads y de entornos Multi-Cluster, así como implementar la estrategia en un Multi-Cluster de producción, y estudiar su comportamiento en esas condiciones.

Bibliografía

[AAEG+03] A. Alonso-Ayuso, L. F. Escudero, A. Garín, M. T. Ortuño, and G. Pérez. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26(1):97–124, 2003.

[Aba04] J. H. Abawajy. Dynamic parallel job scheduling in multi-cluster computing systems. In *Lecture Notes in Computer Science, Computational Science ICCS'04*, volume 3036/2004, pages 27–34, 2004.

[ABN00] A. Abraham, R. Buyya, and B. Nath. Nature's heuristics for scheduling jobs on computational grids. In *8th IEEE International Conference on Advanced Computing and Communications*, pages 45–52, 2000.

[AD03] J. H. Abawajy and S. Dandamudi. Parallel job scheduling on multicluster computing systems. In *IEEE International Conference. CLUSTER'03*, pages 11–18, 2003.

[AKP03] S. Ahmed, A. J. King, and G. Parija. A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24, 2003.

[ANF03] K. Aida, W. Natsume, and Y. Futaka. Distributed computing with hierarchical master-worker paradigm for parallel branch and bound algorithm. In *Cluster Computing and the Grid CCGrid'03*, pages 156–163, 2003.

[ASGH95] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parameterised simulations using distributed workstations. In *Symposium on High Performance Distributed Computing*, 1995.

[Bal08] M. L. Balinski. *50 Years of Integer Programming: 1958–2008*, chapter 6. Springer, 2008.

[BB01] R. Buyya and M. Baker. Emerging technologies for multicluster/grid computing. In *IEEE International Conference. CLUSTER'01*, page 457, 2001.

[BBE03] S. Banen, A.I.D. Bucur, and D.H.J. Epema. A measurement-based simulation study of processor co-allocation in multicluster systems. In *JSSPP'03: Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 105–128. Springer-Verlag, 2003.

[BCP+08] R. Baraglia, G. Capannini, M. Pasquali, D. Puppin, L. Ricci, and A. D. Techiouba. *Making Grids Work*, chapter Backfilling Strategies for Scheduling Streams of Jobs On Computational Farms. Springer, 2008.

[BE01] A.I.D. Bucur and D.H.J. Epema. The influence of communication on the performance of co-allocation. In *In JSSPP'01: Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing*, pages 66–86. Springer-Verlag, 2001.

[BE02] A.I.D. Bucur and D.H.J. Epema. Local versus global schedulers with processor co-allocation in multicluster systems. In *Lecture Notes in Computer Science, Job Scheduling Strategies for Parallel Processing*, volume 2537/2002, pages 184–204, 2002.

[BE07] A.I.D. Bucur and D.H.J. Epema. Schedulling policies for processor coallocation in multicluster systems. *IEEE TPDS*, 18(7):958–972, 2007.

[BFH03] F. Berman, G. Fox, and T. Hey. *Grid Computing: Making the Global Infrastructure a Reality*, chapter 33. Wiley Sons, 2003.

[Bla02] J. Blazewicz. *Scheduling Computer and Manufacturing Processes*. Springer, 2002.

[BMAV05] R. Buyya, M. Murshed, D. Abramson, and S. Venugopal. Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm. *Software Practice and Experience*, 35(5):491–512, 2005.

[BMP+10] A. Benoit, L. Marchal, J. F. Pineau, Y. Robert, and F. Vivien. Scheduling concurrent bag-of-tasks applications on heterogeneous platforms. *IEEE Transactions On Computers*, 59(2):202–217, 2010.

[boi] University of California, BOINC, http://boinc.berkeley.edu.

[BRL99] J. Basney, R. Raman, and M. Livny. High throughput monte carlo. In *9th SIAM Conference on Parallel Processing for Scientific Computing*, 1999.

[Bru04] P. Brucker. *Scheduling Algorithms*. Springer, 2004.

[BSB+01] T.D. Braun, H.J. Siegel, N. Beck, L. Bölöni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, D.A. Hensgen, and R.F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel & Distributed Computing*, 61(6):810–837, 2001.

[BW05] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, 2005.

[ccg] IEEE International Symposium on Cluster Computing and the Grid,
http://www.ccgrid.org.

[CG95] P. I. Crumpton and M. B. Giles. Multigrid aircraft computations using the oplus parallel library. In *Proceedings Parallel CFD '95*, pages 339–346, 1995.

[CIR+09] D. Castella, I.Barri, J. Rius, F. Solsona, F. Gine, and F. Guirado. Codip2p: A peer-to-peer architecture for sharing computing resources. In *International Symposium on Distributed Computing and Artificial Intelligence DCAI'08*, volume 50/2009, pages 293–303, 2009.

[CJLL95] P. Chrétienne, E. G. Coffman Jr., J. K. Lenstra, and Z. Liu. *Scheduling Theory and its Applications*. Wiley Sons, 1995.

[Con] Deparment of computer sciences, university of wisconsin-madison ,condor, http://research.cs.wisc.edu/condor.

[CV01] S. H. Chiang and M. K. Vernon. Production job scheduling for parallel shared memory systems. In *International Parallel and Distributed Processing Symposium, IPDPS'02*, page 47, 2001.

[DKC05] J. Dollimore, T. Kindberg, and G. Coulouris. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science Series)*. Addison Wesley, May 2005.

[Doo53] J. L. Doob. *Stochastic Processes*. Wiley, 1953.

[DS03] M. Dorigo and T. Stützle. *Handbook of Metaheuristics*, volume 57 of *International Series In Operations Research and Management Science*, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, pages 251–285. Kluwer Academic Publisher, 2003.

[DZ97] X. Du and X. Zhang. Coordinating parallel processes on networks of workstations. *Journal of Parallel & Distributed Computing*, 2(46):125–135, 1997.

[EGMP07] L. F. Escudero, A. Garín, M. Merino, and G. Pérez. The value of the stochastic solution in multistage problems. *Top*, 15(1):48–64, 2007.

[EHS+02] C. Ernemann, V. Hamscher, U. Schwiegelshohn, R. Yahyapour, and A. Streit. On advantages of grid computing for parallel job scheduling. In *IEEE/ACM International Conference CCGRID'02*, 2002.

[ELvD+96] D.H.J. Epema, M. Livny, R. van Danzig, X. Evers, and J. Pruyne. A worldwide flock of condors: load sharing among workstation clusters. *Future Generation Computer Systems*, 12(1):53–65, 1996.

[Fos05] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *Lecture Notes in Computer Science*, volume 3779, pages 2–13, 2005.

[FR90] D.G. Feitelson and L. Rudolph. Distributed hierarchical control for parallel processing. *Computer*, 23(5):65–77, 1990.

[FR99] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, (6):109–133, 1999.

[FRS05] D.G. Feitelson, L. Rudolph, and U. Schiwiegelshohn. Parallel job scheduling - a status report. In *LNCS*, volume 3277, pages 1–16, 2005.

[FSZX03] H. Feng, G. Song, Y. Zheng, and J. Xia. A deadline and budget constrained cost-time optimization algorithm for scheduling dependent tasks in grid computing. In *Grid and Cooperative Computing*, pages 113–120, 2003.

[GBS10] S. K. Garg, R. Buyya, and H. J. Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, 26:1344–1355, 2010.

[GKYL01] J. P. Goux, S. Kulkarni, M. Yoder, and J. Linderoth. Master-worker: An enabling framework for applications on the computational grid. *Cluster Computing*, 4(1):63–70, 2001.

[Glo77] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciencies*, (8):156–166, 1977.

[Glo98] F. Glover. A template for scatter search and path relinking. In *Lecture Notes in Computer Science, Artificial Evolution*, number 1363, pages 13–54, 1998.

[HGH+05] M. Hanzich, F. Giné, P. Hernández, F. Solsona, and E. Luque. Cisne: A new integral approach for scheduling parallel applications on non-dedicated clusters. In *Euro-Par'05*, volume 3648/2005, page 643, 2005.

[HHL+06] M. Hanzich, P. Hernández, E. Luque, F. Giné, F. Solsona, and J. L. Lérida. Using simulation, historical and hybrid estimation systems for enhancing job scheduling on nows. In *In Proceedings of 2006 IEEE International Conference on Cluster Computing*, pages 1–12, 2006.

[HYD+00] P. Holenarsipur, V. Yarmolenko, J. Duato, D. K. Panda, and P. Sadayappan. Characterization and enhancement of static mapping heuristics for heterogeneous systems. In *International Conference of High Performance Computing HIPC'00*, pages 37–48, 2000.

[IBM] IBM. CPLEX, http://www.ibm.com/software/integration/optimization/cplex-optimizer.

[iee] IEEE Computer Society Task Force on Cluster Computing, http://www.ieeetfcc.org.

[JAA07] B. Javadi, M.K. Akbari, and J.H. Abawajy. A performance model for analysis of heterogeneous multi-cluster systems. *Parallel Computing*, 32(11-12):831–851, 2007.

[JLPS05] W. Jones, W. Ligon, L. Pang, and D. Stanzione. Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. *Journal of Supercomputing*, 34(2):135–163, 2005.

[KDM09] S. Kumar, K. Dutta, and V. Mookerjee. Maximizingbusiness value by optimal assignment of jobs to resources in grid computing. *European Journal of Operational Research*, 194(3), 2009.

[Ken99] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *IEEE Congress on Evolutionary Computation (CEC 1999)*, pages 1931–1938, 1999.

[KF98] C. Kesselam and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[KT06] P. Krusche and A. Tiskin. Efficient longest common subsequence computation using bulk-synchronous parallelism. In *ICCSA*, volume 3984, pages 165–174, 2006.

[KX11] J. Kolodziej and F. Xhafa. Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids. *International Journal of Applied Mathematics and Computer Science*, 21(2):243–257, 2011.

[KXK09] J. Kolodziej, F. Xhafa, and L. Kolanko. Hierarchic genetic scheduler of independent jobs in computational grid environment. In *23rd European Conference on Modelling and Simulation*, 2009.

[LAH09] H. Liu, A. Abraham, and A. E. Hassanien. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, 26(8):1336–1343, 2009.

[LF03] U. Lublin and D.G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel & Distributed Computing*, 11(63):1105–1122, 2003.

[LGW05] H. Li, D. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputing. In *Job Scheduling Strategies for Parallel Proc. JSSPP'04*, pages 176–193, 2005.

[Lif95] D. Lifka. The anl/ibm sp scheduling systems. In *Job Scheduling Strategies for Parallel Processing*, volume 949, pages 295–303, 1995.

[Lin] Lindo. Lingo
http://www.lindo.com.

[LLM88] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *8th International Conference of Distributed Computing Systems*, pages 104–111, 1988.

[LMS02] H. R. Lourenço, O. Martin, and T. Stützle. *Handbook of Metaheuristics*, chapter Iterated local search, pages 321–353. Kluwer Academic Publishers, 2002.

[Loa] IBM, LoadLeveler, http://www.ibm.com/developerworks/wikis/x/iyapaw.

[LSG+06] J. L. Lérida, F. Solsona, F. Giné, M. Hanzich, P. Hernández, and E. Luque. Metaloras: A predictable metascheduler for non-dedicated multiclusters. In 630-641, editor, *ISPA'06*, 2006.

[LSG+08] J.L. Lérida, F. Solsona, F. Giné, J.R. García, and P. Hernández. Resource matching in non-dedicated multicluster environments. In *VECPAR 2008*, pages 160–173, 2008.

[Mau] Adaptive computing , Maui Cluster Scheduler, http://www.clusterresources.com/products/maui-cluster-scheduler.php.

[Mla97] N. Mladenovic. Variable neighborhood search. *Coputers & Operations Research*, 24(11):1097–1100, 1997.

[MLS07] E. U. Munir, J. Li, and S. Shi. Qos sufferage heuristic for independent task scheduling in grid. *Information Technology Journal*, 6(8):1166–1170, 2007.

[MM02] V. D. Martino and M. Mililotti. Scheduling in a grid computing environment using genetic algorithms. In *International Parallel and Distributed Processing Symposium, IPDPS'02*, page 297, 2002.

[MPW94] P. B. Monk, A. K. Parrott, and P. J. Wesson. A parallel finite element method for electromagnetic scattering. *COMPEL*, Supp.A(13):237–242, 1994.

[Müh98] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, pages 303–346, 1998.

[NLYW05] V.K. Naik, C. Liu, L. Yang, and J. Wagner. Online resource matching for heterogeneous grid environments. In *IEEE/ACM International Conference CCGRID'05*, volume 2, pages 607–614, 2005.

[NNS95] M. Nibhanupudi, C. Norton, and B. Szymanski. Plasma simulation on networks of workstations using the bulk synchronous parallel model. In *In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1995.

[Ope] Adaptive computing ,TORQUE Resource Manager, http://www.adaptivecomputing.com/products/open-source/torque.

[PBS] Altair engineering ,the Portable Batch System, http://www.pbsworks.com.

[Pin02] M. Pinedo. *Scheduling: Theory, algorithms, and systems*. Prentice Hall, 2002.

[RBMS97] D. Ridge, D. Becker, P. Merkey, and T. Sterling. Beowulf: Harnessing the power of parallelism in a pile-of-pcs. In *IEEE Aerospace*, pages 79–91, 1997.

[Sal04] F. Abu Salem. A bsp parallel model for the gottfert algorithm over f2. In *Parallel Processing and Applied Mathematics*, volume 3019, pages 217–224, 2004.

[SCJG00] Q. Snell, M. J. Clement, D. B. Jackson, and C. Gregory. The performance impact of advance reservation meta-scheduling. In *Workshop on Job Scheduling Strategies for Parallel Processing, IPDPS'00/JSSPP'00*, pages 137–153, 2000.

[Set] University of california, SETI@Home, http://setiathome.berkeley.edu/.

[SF05] E. Shmueli and D.G. Feitelson. Backfilling with lookahead to optimize the packing of parallel jobs. *Journal of Parallel & Distributed Computing*, 65(9):1090–1107, 2005.

[Sga98] J. Sgall. On-line scheduling. In *Lecture Notes in Computer Science, Online Algorithms*, volume 1442/1998. Springer, 1998.

[SHM97] D. B. Skillicorn, J. M. D. Hill, and W. F. McColl. *Questions and Answers about BSP*. Oxford University Computing Laboratory, 1997.

[SKRS03] G. Sabin, R. Kettimuthu, A. Rajan, and P. Sadayappan. Scheduling of parallel jobs in a heterogeneous multi-site environment. In *Lecture Notes in Computer Science*, volume 2862/2003, pages 87–104, 2003.

[SKSS02a] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. In *Job Scheduling Strategies for Parallel Proc. JSSPP'02*, pages 55–71, 2002.

[SKSS02b] V. Subramani, R. Kettimuthu, S. Srinivasan, and S. Sadayappan. Distributed job scheduling on computational grids using multiple simultaneous requests. In *IEEE International Symposium on High Performance Distributed Computing HPDC'02*, pages 359–366, 2002.

[SP94] D. D. Sharma and D. K. Pradhan. Job scheduling in mesh multicomputers. In *International Conference on Parallel Processing ICCP'94*, pages 251–258, 1994.

[SQR10] S.M. Hussain Shah, K. Qureshi, and H. Rasheed. Optimal job packing, a backfill scheduling optimization for a cluster of workstations. *Journal of Supercomputing*, 54(3):381–399, 2010.

[SSB+95] T. Sterling, D. Savarese, D.J. Becker, J.E. Dorband, U.A. Ranawake, and C.V. Packer. Beowulf: A parallel workstation for scientific computation. In *24th International conference on Parallel Processing*, pages 11–14, 1995.

[Sto07] H. Stockinger. Defining the grid: a snapshot on the current view. *Journal of Supercomputing*, 42(1):3–17, 2007.

[SvANS00] J. Santoso, G.D. van Albada, B. A. A. Nazief, and P. M. A. Sloot. Hierarchical job scheduling for clusters of workstations. In *Advanced School for Computing and Imaging ASCI'00*, pages 99–105, 2000.

[TD01] T. Thanalapati and S. Dandamudi. An efficient adaptive scheduling scheme for distributed memory multicomputers. *IEEE Transactions on Parallel and Distributed Computers*, 12(7):758–768, 2001.

[TEF07] D. Tsafrir, Y. Etsion, and D.G. Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. In *IEEE Transaction on Parallel and Distributed Systems*, volume 18(6), pages 789–803, 2007.

[TF99] D. Talby and D.G. Feitelson. Supporting priorities and improving utilization of the ibm sp scheduler using slack-based backfilling. In *International Parallel Processing, Symposium on Parallel and Distributed Processing IPPS/SPDP*, pages 513–517, 1999.

[TGP] The University of Melbourne The Gridbus Project, GRIDS Lab. GridSim simulation framework, http://www.buyya.com/gridsim.

[top] TOP500 Supercomputer Sites, http://www.top500.org.

[TRA+06] A. Tchernykh, J. M. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, and S. Zhuk. Dynamic parallel job scheduling in multi-cluster computing systems. In *Lecture Notes in Computer Science, Parallel Processing and Applied Mathematics*, volume 3911/2006, pages 774–781, 2006.

[WMW02] W. A. Ward, C. L. Mahood, and J. E. West. Scheduling jobs on parallel systems using a relaxed backfill strategy. In *Job Scheduling Strategies for Parallel Proc. JSSPP'02*, pages 88–102, 2002.

[Wol03] R. Wolski. Experiences with predicting resource performance online in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 2003.

[wor] Parallel workloads archive, http://www.cs.huji.ac.il/labs/parallel/workload/.

[WSH00] R. Wolski, N. T. Spring, and J. Hayes. Predicting the cpu availability of time-shared unix systems on the computational grid. *Cluster Computing*, 4(3):293–301, 2000.

[XCA07] F. Xhafa, J. Carretero, and A. Abraham. Genetic algorithm based schedulers for grid computing systems. *International Journal of*

Innovative Computing, Information and Control, 3(5):1053–1071, 2007.

[XK10] F. Xhafa and J. Kolodziej. A game-theoretic and hybrid genetic meta-heuristics model for security-assured scheduling of independent jobs in computational grids. In *International Conference on Complex, Intelligent and Software Intensive Systems*, 2010.

[YDPS00] V. Yarmolenko, J. Duato, D. K. Panda, and P. Sadayappan. Characterization and enhancement of dynamic mapping heuristics for heterogeneous systems. In *International Workshop on Parallel Processing ICPP'00*, volume 1970/2000, pages 437–444, 2000.

[YTCC08] C. Yang, H. Tung, K. Chou, and W. Chu. Well-balanced allocation strategy for multiple-cluster computing. In *IEEE International Conference. FTDCS'08*, pages 178–184, 2008.

[YYWZ00] Y. Yuan, G. Yang, Y. Wu, and W. Zheng. Pv-easy: a strict fairness guaranteed and prediction enabled scheduler in parallel job scheduling. In *ACM International Symposium on High Performance Distributed Computing HPDC'10*, pages 240–251, 2000.