Revista Electrónic@ de Medioambiente. UCM.
IUCA
ISSN 1886-3329

Computación en *Grid*: una Solución Asequible para las Simulaciones Medioambientales

Enrique Moratilla Temprado
Raúl Durán Díaz
Rafael Rico López
Dpto. de Automática (Universidad de Alcalá de H.)
(enrique.moratilla, raul.duran, rafael.rico)@uah.es

Recibido: 3 de octubre de 2006
Aceptado: 13 de noviembre de 2006

RESUMEN

Se presenta la computación en *grid* como una opción barata y asequible, desde el punto de vista del desarrollo de aplicaciones, para realizar computación de altas prestaciones con modelos de simulación medioambiental. El procedimiento propuesto permite la agregación de recursos de diferentes organizaciones, con el consiguiente incremento de la productividad de las inversiones. También se muestran soluciones de cálculo numérico que minimizan la elevada carga de comunicaciones inherente a los sistemas distribuidos.

Palabras clave: Simulación medioambiental, computación en *grid*, cálculo numérico.

GRID COMPUTING: AN AFFORDABLE SOLUTION FOR ENVIRONMENTAL SIMULATIONS

ABSTRACT

This paper presents grid computing as a non-expensive and affordable option, from the point of view of application development, to leverage high performance computing using environmental simulation models. The proposed method allows the aggregation of resources provided by different organizations, thus allowing a substantial increase in the return of investments. Numerical computing solutions reducing the high communications load inherent to distributed systems are also shown.

Keywords: Environmental simulation, grid computing, numerical computing.

GRILLE DE CALCUL: UNE SOLUTION ABORDABLE POUR DES SIMULATIONS ENVIRONNEMENTALES

RÉSUMÉ

Cet article présente la grille de calcul comme option, bon marché et abordable, du point de vue de développement des applications, pour profiter le calcul à haute performance en utilisant des modèles environnementaux de simulation. La marche à suivre proposée permet l'agrégation des ressources de différents organismes augmentant

le débit des investissements. En outre, des solutions de calcul numérique sont montrées qui réduisent la charge élevée de communications inhérente aux systèmes distribués.
Mots clé: Simulation environnementale, grille de calcul, calcul numérique.

1. INTRODUCCIÓN

En investigación medioambiental la simulación se ha convertido en una herramienta de trabajo básica. Los modelos utilizados incorporan complejas representaciones matemáticas de procesos físicos y químicos que requieren una gran capacidad de cálculo si pretenden ser precisos.

Para conseguir una potencia de cálculo suficiente, se recurre a la computación de altas prestaciones que se construye sobre técnicas basadas en el paralelismo tanto de los programas a ejecutar como de los recursos físicos que se utilizan. La computación de altas prestaciones suele ser cara por los equipos usados y por el grado de especialización de los programadores.

La computación en *grid* se revela como una plataforma emergente en el mundo de la computación de altas prestaciones, que puede aliviar notablemente los costes de los equipos así como la exigencia en la preparación de los programadores y desarrolladores.

Se propone aquí un procedimiento de trabajo, fundamentado en la agregación de recursos de diferentes organizaciones para constituir un entorno de computación en *grid*, barato y accesible desde el punto de vista del desarrollo de aplicaciones. El procedimiento permite un mejor aprovechamiento de las inversiones realizadas en equipos de computación e incrementa la productividad.

2. NECESIDAD DE LA COMPUTACIÓN PARALELA

Dentro del terreno medioambiental, la computación de altas prestaciones es aplicable, entre otros ejemplos, a los siguientes: prevención de catástrofes; integración de SIG con análisis de agentes naturales, socio-económicos, etc.; modelado de la reflectividad de la cubierta vegetal en teledetección; simulación de las emisiones de gases generados por los medios de transporte; análisis de series largas de registros de precipitaciones; modelos marinos, climáticos, geológicos, etc. En todos los casos se emplean modelos físico-químicos descritos mediante ecuaciones integrales o diferenciales que finalmente son resueltos mediante técnicas numéricas. Con el fin de lograr la precisión necesaria, el problema se subdivide infinitesimalmente forzando a la máquina a realizar millones de iteraciones. El resultado es que crecen considerablemente tanto los ciclos de computación como el área de memoria de datos empleados.

La capacidad de cómputo de los ordenadores se ha incrementado notablemente en los últimos años, haciendo desaparecer las categorías intermedias entre las dos únicas existentes hoy en día: el computador de sobremesa o PC y los supercomputadores. Además, éstos últimos están construidos a base de los procesadores que encontramos en aquellos. Por ejemplo, el supercomputador *MareNostrum*, perteneciente al Centro Nacional de Supercomputación, presente en la lista TOP500 [1] de los computadores más potentes del mundo con un rendimiento de pico de 42,35 Teraflops, está formado por 4.812 procesadores *IBM Power PC* [2].

Aunque durante las últimas décadas la velocidad de los procesadores ha aumentado vertiginosamente, la computación secuencial tiene un tope tecnológico derivado de la limitación en la capacidad de integración y, por tanto, de la velocidad de

reloj. Una de las maneras de superar este problema es utilizar distintas máquinas en paralelo siempre que el problema computacional se pueda dividir. La asignación a múltiples recursos de un problema computacional particionado posibilita un procesamiento concurrente que o bien disminuye el tiempo de ejecución o bien permite abordar problemas con más datos.

3. DISTINTAS SOLUCIONES *HARDWARE*

La paralelización, como técnica más utilizada en la computación de altas prestaciones, hace referencia a dos aspectos bien diferenciados: la agregación de recursos *hardware* para que trabajen concurrentemente y la partición de las tareas *software* para que puedan ser asignadas a los diferentes elementos de proceso. En esta sección trataremos el primer aspecto.

Hoy en día, los procesadores de cualquier computador personal ya incorporan algún tipo de división funcional orientada al procesamiento concurrente. En los últimos tiempos los fabricantes se han lanzado al desarrollo de multiprocesadores *on chip* dotando a su gama más potente del doble núcleo. Tratan de vender la idea de que al poseer dos procesadores en el mismo *chip* obtendremos el doble de rendimiento. Por desgracia, en la actualidad todavía existen pocos programas comerciales que aprovechen esta característica.

Entre los sistemas masivamente paralelos encontramos diferentes arquitecturas: sistemas multiprocesador, computadores vectoriales y computadores matriciales. Constituyen una única máquina o sistema fuertemente acoplado ya que todos los procesadores suelen compartir la memoria como recurso común. No es el objeto del presente artículo describir técnicamente cada uno de ellos pero en todos los casos estamos hablando de muy altos rendimientos, en la frontera de la supercomputación actual, construidos a medida normalmente y utilizados en aplicaciones estratégicas de gobiernos o grandes compañías. Su precio excede ampliamente las posibilidades de financiación de cualquier proyecto de investigación de rango convencional.

Para necesidades de computación menos extremas contamos con los *clusters*. Se clasifican dentro de los sistemas multicomputador, es decir, aquellos donde se agregan computadores completos que podrían trabajar independientemente. Son sistemas débilmente acoplados aunque físicamente las máquinas están próximas o incluso residen en el mismo bastidor. No comparten la memoria y, en consecuencia, la comunicación entre unidades de proceso se realiza mediante paso de mensajes. Existen *clusters llave en mano* diseñados por fabricantes especializados pero también se han extendido los *clusters off the shelf* construidos por los propios usuarios finales siguiendo el esquema del *Beowulf computing* [3]. Su difusión es debida a dos factores: por un lado, la relación precio/rendimiento de los ordenadores personales es muy buena y, por otra parte, las redes de interconexión convencionales de hasta 1.000 Mbps permiten comunicar máquinas con una sobrecarga tolerable. La conectividad se puede montar sobre un protocolo común de comunicación (como puede ser el estándar TCP/IP) y la ejecución concurrente de tareas se puede implantar utilizando un paradigma de programación basado en memoria distribuida y paso de mensajes, como por ejemplo MPI (***M****enssage* ***P****assing* ***I****nterface*). Representan una alternativa viable a los supercomputadores aunque su precio también puede desbordar las posibilidades habituales.

El *grid computing* representa un modelo emergente de computación. Proporciona una alta productividad gracias a que admite la agregación dinámica de cualquier recurso computacional que esté conectado a una red global de computación como puede ser Internet. Se comparten de manera distribuida recursos tales como capacidad de cómputo

o capacidad de almacenamiento. Su funcionamiento se basa en la instalación de una capa *middleware* que sirve de enlace entre los recursos independientemente del *hardware* y del sistema operativo de cada máquina.

En el plano teórico, un *grid* de computación es un sistema multicomputador pero aún más desacoplado que los *clusters* ya que engloba máquinas pertenecientes a dominios administrativos diferentes. La idea original consiste en formar una infraestructura a nivel mundial que proporcione servicios a petición del usuario de manera completamente transparente a él, al igual que ocurre, por ejemplo, con el suministro eléctrico [4].

4. PARADIGMAS DE LA PROGRAMACIÓN EN PARALELO

Existen varios métodos de programación en computadores paralelos. La clave está en el modo en el que los diferentes programas concurrentes (procesos) cargados en diferentes recursos físicos de procesamiento se comunican entre sí. Se puede hacer una clasificación sencilla basada en el esquema de memoria que manejen las unidades de procesamiento (memoria compartida o memoria distribuida) ya que esta característica impone diferencias en la comunicación entre procesos. Así, en el caso de memoria compartida podemos usar programación de hebras mientras que en el caso de memoria distribuida usaremos el paso de mensajes.

Bajo el paradigma de hebras, un único proceso puede tener múltiples trayectorias de ejecución concurrentes. Es algo así como si un programa creara una serie de subrutinas que pueden avanzar concurrentemente. Las hebras se comunican utilizando memoria compartida. Dado que todos los procesadores "ven" un mapa de memoria común, el tamaño sin particionar de los datos a procesar puede ser tan grande como toda la memoria disponible.

En el caso del paso de mensajes, el programador hace uso de una serie de funcionalidades de biblioteca para pasar información entre unos procesadores y otros. Los procesos se van a comunicar entre sí enviando y recibiendo mensajes y cada proceso utiliza sólo memoria local. Si el tamaño de los datos a procesar es muy grande hay que hacer un particionado que aloje una parte de ellos en la memoria local de cada unidad de procesamiento. La transferencia de información requiere que se creen operaciones cooperativas para ser utilizadas en cada proceso: por ejemplo, una operación de envío de mensaje debe tener una operación de recepción del mensaje en algún proceso.

Finalmente, podemos considerar un tercer método de paralelización cuando los procesos no necesitan ningún tipo de comunicación. Se conoce como paralelismo de datos. En este escenario, se realiza un reparto de los datos entre los distintos elementos de procesamiento y se arrancan los procesos que operarán de manera independiente hasta el final. Es adecuado para procesos en los que los resultados obtenidos sobre una parte del problema no afectan al resto o bien para ejecuciones paramétricas, es decir, aquellas en las que cada proceso trabaja con un conjunto de datos de entrada diferente.

5. COMPUTACIÓN EN *GRID*: BUENAS PRESTACIONES Y COSTE CONTENIDO

Como ya hemos indicado, la computación en *grid* constituye un tipo de sistema paralelo y distribuido que permite la compartición, selección y agregación dinámica de recursos autónomos geográficamente distribuidos. Una de sus grandes ventajas radica en que puede utilizar como infraestructura de comunicaciones tanto una red privada

como Internet. A través de Internet, vamos a poder compartir no sólo información sino ciclos de computación o capacidad de almacenamiento. De esta manera, un conjunto de organizaciones independientes pueden unir temporalmente, bajo un único *grid* de computación, recursos de procesamiento pertenecientes a dominios administrativos autónomos. La unión temporal de máquinas hace crecer la capacidad de cómputo, aumenta la productividad de las mismas y abarata el coste por ciclo de computación sin que cada organización pierda el dominio administrativo.

Otra ventaja del *grid* es que no impone requisitos muy exigentes a las máquinas que lo componen ni en potencia de cálculo ni en capacidad de comunicación. Esta característica permite dar una segunda oportunidad a equipos obsoletos rentabilizando aún más inversiones ya amortizadas y, por tanto, abaratando el ciclo de computación. Basta con unir una serie de máquinas en una red privada para poder empezar a realizar procesamiento concurrente.

Estas ventajas encajan bien con el tipo de uso que se le da a las máquinas en el entorno de investigación universitario, en el que se alternan periodos de cálculo intensivo con periodos de ausencia casi total de trabajo computacional debido a la gran dispersión de tareas que realiza el personal docente e investigador.

Como es lógico pensar, no todo son ventajas. El gran problema de la computación en *grid* se encuentra en el elevado coste temporal de las comunicaciones entre procesos. El rendimiento del sistema completo se ve fuertemente afectado por la lentitud en la compartición de datos. El efecto se acentúa cuando se usa Internet como infraestructura de comunicaciones.

El inconveniente de las comunicaciones ha derivado en que se utilice el *grid* para implantaciones de paralelismo de datos, en las que los procesos no necesitan comunicarse entre sí, o bien ensamblado con un planificador de tareas que se encarga de lanzar trabajos desacoplados. Aunque existen muchos problemas que encajan con estos casos, quedan fuera aquellos que utilizan modelos matemáticos basados en la descomposición de dominios. Sería deseable, por tanto, encontrar alguna solución para ellos adecuada a la computación en *grid*.

6. TÉCNICAS DE CÁLCULO NUMÉRICO EN EL *GRID*

En la literatura se han descrito últimamente diversas aplicaciones aptas para ser procesadas en *grid* [5, 6, 7]. Una de las técnicas más utilizadas en el cálculo numérico paralelo es la descomposición de dominios, cuya idea es dividir el dominio del problema original en diversos subdominios. La principal desventaja es el fuerte acoplamiento entre los subdominios, lo que supone una alta carga de comunicaciones, que es perjudicial para el rendimiento en un *grid*.

Sin embargo, recientemente ha sido presentado un novedoso método de descomposición de dominios (DD), denominado descomposición de dominios probabilística (PDD), que rebaja notablemente las necesidades de comunicación entre dominios [8]. El núcleo del método consiste en obtener la solución en algunos puntos del dominio sin resolver previamente el problema completo de condiciones de contorno que es el verdadero responsable de las comunicaciones. La solución en esos puntos se obtiene mediante simulaciones de Monte-Carlo y luego se extienden interpolando en todas las interfases. Así cada subdominio se “independiza” del resto.

Colateralmente se consigue otro beneficio con el método PDD: la solución alcanzada en una máquina para un subdominio es válida y, por tanto, utilizable aunque

el resto de máquinas no hayan terminado. De esta manera se evitan los complicados problemas de equilibrado de carga, cuyo objetivo es conseguir un reparto adecuado de trabajo, de modo que máquinas rápidas y lentas finalicen simultáneamente su tarea.

7. MARCO EXPERIMENTAL PROPUESTO

Con el fin de probar el comportamiento de un *grid* de computación y de experimentar el rendimiento de la novedosa técnica PDD, se ha diseñado el entorno experimental de la Figura 1. Se han formando dos subredes *Fast Ethernet* a 100 Mbps, unidas entre sí por una red *Ethernet* más lenta de 10 Mbps. El diseño pretende modelar el comportamiento real cuando se utiliza la infraestructura de Internet para unir temporalmente dominios administrativos independientes.

Cada subred cuenta con 8 máquinas. La subred A se ha establecido sobre las máquinas de un laboratorio de alumnos y se ha utilizado en los momentos en los que no estaba ocupado si bien esta circunstancia no es impedimento para la ejecución de nuestras tareas siempre que no se les asigne una prioridad muy alta. La subred B se ha construido utilizando máquinas obsoletas que ya habían sido retiradas de todo uso. Con ello se pretende demostrar las posibilidades de los *grid* de computación en el abaratamiento del ciclo de computación alargando la vida útil de los equipos. Las características de las máquinas utilizadas se enumeran en la Tabla 1. Se ha incluido el coste orientativo con el fin de dar una idea aproximada de la inversión a realizar.

En cuanto al *software*, se ha instalado como sistema operativo *Linux Fedora Core*, un sistema operativo de propósito general y basado exclusivamente en *software* libre. El *middleware* utilizado para la implementación del *grid* ha sido *Globus Toolkit*, también de *software* libre y estándar de *facto* hoy en día [9, 10]. Finalmente, como librería de funciones de paso de mensajes se ha instalado MPICH-G2, especialmente diseñado para el entorno *Globus*. La compilación, instalación y configuración de toda esta capa *software* en todas las máquinas del *grid* puede llevar entre 80 y 120 horas de trabajo. Hay que hacer notar que la instalación del sistema operativo *Linux* en las máquinas de menos capacidad se puede limitar al entorno en *modo comando* prescindiendo de toda la parte gráfica del sistema que es mucho más consumidora de recursos.

Como caso práctico se propone la resolución numérica de la ecuación de Laplace en una dimensión aplicando el método de diferencias finitas y descomposición de dominios. El programa se va a escribir en FORTRAN 90 paralelizado mediante rutinas de paso de mensajes MPI. Las pruebas se realizarán lanzando un número creciente de procesos de 1 a 16 que se asignarán a diferentes máquinas y obteniendo el tiempo total de cómputo en cada situación. Con el fin de hacer más ilustrativos los resultados, se harán las pruebas tanto en el entorno *grid* como en un *cluster*. La Tabla 2 muestra las características del *cluster SunFire*. Obsérvese que los procesadores que lo componen son más potentes que los del *grid* y que la red de comunicaciones que lo interconecta también es 10 veces más rápida.

8. RESULTADOS Y CONCLUSIONES

La Figura 2 muestra los resultados obtenidos en función del número de procesos en los que se ha paralelizado el problema de prueba. Se presenta el comportamiento tanto sobre la plataforma *grid* como sobre la plataforma *cluster* [11]. Cuando se utiliza el método de descomposición de dominios clásico (DD), observamos que en el *grid* de computación el tiempo se dispara debido al fuerte acoplamiento de la aplicación y al lastre de las comunicaciones. En el caso del *cluster* el efecto negativo de la comunicación entre procesos también se advierte ya que el

tiempo de cómputo crece según aumenta el número de procesos, pero el incremento es muy ligero.

Sin embargo, la utilización del método de descomposición de dominios probabilística (PDD) hace mejorar mucho la curva de tiempos sobre el *grid* de computación aproximando los resultados a los obtenidos sobre el *cluster* utilizando DD. Hay que hacer notar que las comunicaciones dentro del *cluster* son 10 veces más rápidas que en el interior de cada subred del *grid*. Este hecho hace aún mejores los resultados del método PDD sobre el *grid*.

Podemos concluir, entonces, que la tecnología *grid* es una alternativa conveniente para realizar computación de altas prestaciones a bajo coste. Posibilita la salvaguarda de inversiones ya que permite dar un segundo uso a máquinas que no utilizábamos y posibilita la agregación temporal de dominios administrativamente autónomos incrementando la productividad de infraestructuras de cálculo de organizaciones independientes y la colaboración entre grupos de investigación.

Por otra parte, el método de descomposición de dominios probabilístico (PDD) consigue reducir notablemente los tiempos de ejecución en la infraestructura *grid*, debido a la disminución de la carga de comunicaciones. De esta forma, su rendimiento se acerca al rendimiento que se obtiene en un *cluster* con el método de descomposición de dominios clásico (DD).

REFERENCIAS

[1] Top500 Supercomputer Sites: http://www.top500.org/.
[2] Barcelona Supercomputing Center: http://www.bsc.es.
[3] Beowulf Project. http://www.beowulf.org/.
[4] M. Chetty and R. Buyya. “Weaving Computational Grids: How Analogous Are They with Electrical Grids?”, Computing in Science and Engineering (CiSE), ISSN 1521-9615, Volume 4, Issue 4, Pages: 61-71, IEEE Computer Society Press and American Institute of Physics, USA, July-August 2002
[5] B.M. Boghosian, P.V. Coveney. “Scientic applications of Grid computing”. IEEE CS Press, 7, 10-13 September 2005.
[6] S. Dong, G.E. Karniadackis, N.T. Karonis. “Cross-site computations on the TeraGrid”. IEEE CS Press, 7, 14-23 September 2005.
[7] “Distributed Computing”, Science, Special Issue, 308 809-821, 2005.
[8] J.A. Acebrón, M.P. Busico, P. Lanucara, R. Spigler. “Probabilistically induced domain decomposition methods for elliptic boundary-value problems”. J. Comput. Phys., 210, 421-438, 2005.
[9] I. Foster. “Globus Toolkit version 4: Software for Service-Oriented Systems”. IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2005.
[10] Globus Toolkit. http://www.globus.org.
[11] J.A. Acebrón, J.L. Carnero, R. Durán, E. Moratilla, R. Rico, R. Spigler. “Nueva estrategia de descomposición de dominio adecuada para computación en grid”. Actas de las XIV Jornadas de Concurrencia y Sistemas Distribuidos. ISBN: 84-689-9292-5. 2006.

FIGURAS

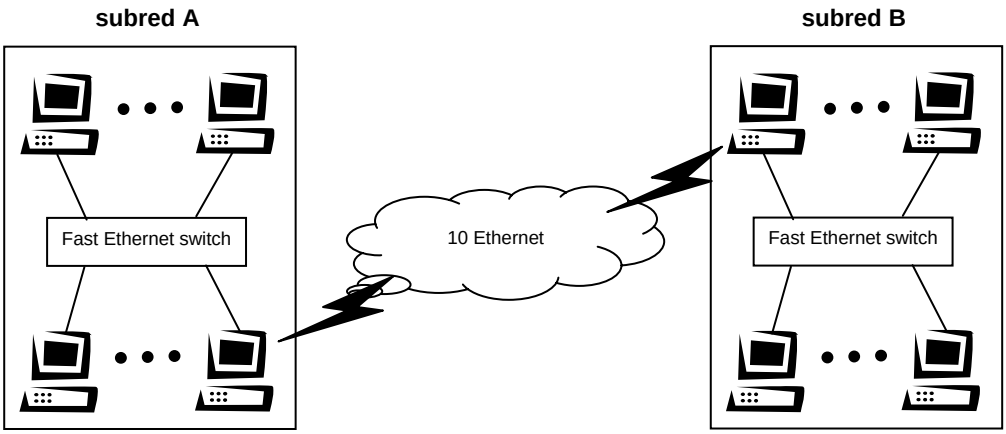

Figura 1. Esquema del entorno experimental construido.

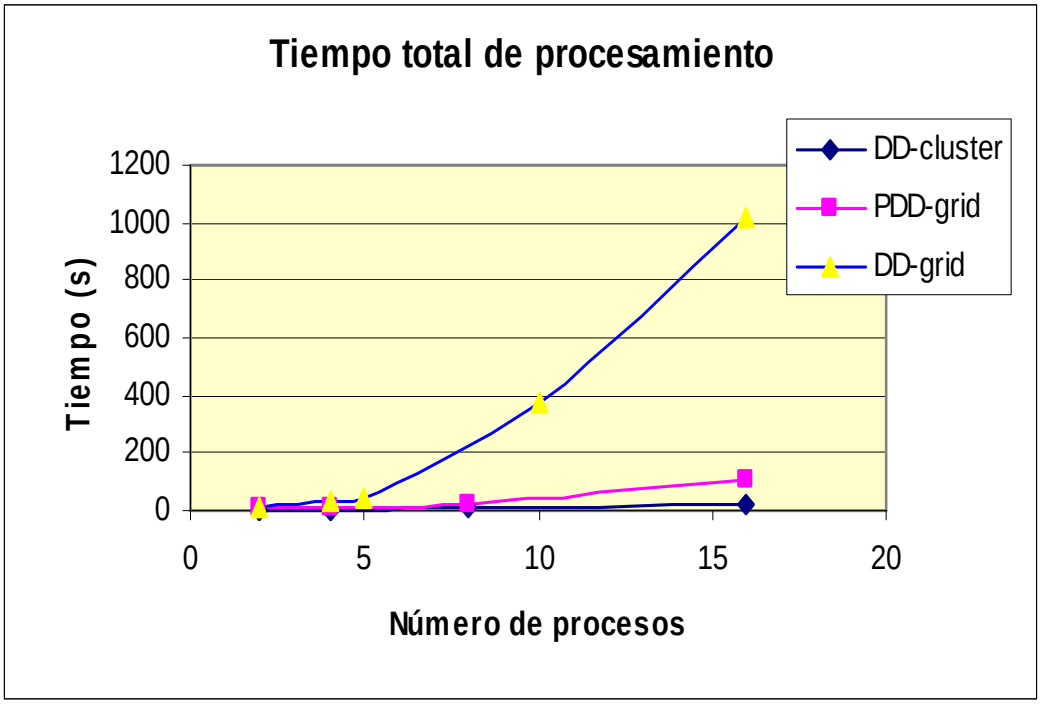

Figura 2. Tiempo de procesamiento del problema de prueba en función del número de procesos en los que se ha paralelizado.

TABLAS

	Cantidad	Características	Coste
Subred A	8	**Procesador** AMD Duron **Reloj** 1,3Ghz **Cache** 64 KB **Memoria** 256 MB **Disco duro** 40GB **Tarjeta de red** 100Mbps	amortizados
	1	switch 10/100 Mbps	≈ 50,00€
Subred B	8	**Procesador** AMD K6 **Reloj** 350Mhz **Cache** 64 KB **Memoria** 192 MB **Disco duro** 10GB **Tarjeta de red** 100Mbps	retirados
	1	switch 10/100 Mbps	≈ 50,00€
opcional	1	selector de monitor-teclado-ratón de 16 entradas	≈ 450,00€ (opcional)
	1	monitor y teclado	amortizados

Tabla 1. Características de las máquinas que componen el *grid*.

Procesador	Mobile AMD Athlon(tm) XP-M 1800+
Reloj	1,5 Ghz
Cache	256 KB
Memoria:	1 GB
Disco duro:	30 GB
Tarjeta de red:	1 Gbps

Tabla 2. Características de los nodos que componen el *cluster***.**