Scheduling paralelo orientado a cluster de ordenadores no dedicados

Mauricio Hanzich[1], Francesc Giné[2], Porfidio Hernández[1], Francesc Solsona[2] y Emilio Luque[1]

[1] Departamento de Informática, Universidad Autónoma de Barcelona, España.
{porfidio.hernandez, emilio.luque}@uab.es, mauricio@aomail.uab.es

[2] Departamento de Informática e Ingeniería Industrial, Universidad de Lleida, España.
{sisco, francesc}@eup.udl.es

Resumen

La demanda actual de grandes capacidades de cómputo ha comportado un importante progreso de los sistemas paralelos. Asimismo, diferentes estudios constatan la baja utilización de los recursos de cómputo disponibles en una red de ordenadores. Considerando ambas situaciones, la comunidad científica ha trabajado en el desarrollo de entornos que permitan el uso de estas redes de ordenadores, también denominadas clusters, con una doble funcionalidad: ejecutar aplicaciones paralelas aprovechando los recursos ociosos presentes a lo largo del cluster, sin perturbar el rendimiento de las aplicaciones ejecutadas por los usuarios locales.

Este trabajo presenta un entorno, denominado CISNE, enmarcado en esta línea de trabajo. CISNE incide tanto en la planificación espacial como temporal de las aplicaciones distribuidas, garantizando, al mismo tiempo, que el usuario local presente en cada uno de los nodos que constituyen el cluster, no perciba una ralentización en el rendimiento de sus aplicaciones. Los resultados obtenidos, medidos con herramientas que han sido desarrolladas a tal efecto, muestran la viabilidad de nuestras propuestas.

Palabras claves: Procesamiento Paralelo, Sistemas Distribuidos, Sistemas Operativos.

1. INTRODUCCIÓN

Las mejoras recientes en el ancho de banda y latencia para redes LAN, la infrautilización de los entornos de estaciones de trabajo [1, 2] cuando se utilizan exclusivamente para tareas interactivas, y el bajo costo económico de los mismos, han hecho de los clusters de ordenadores no dedicados, una arquitectura atractiva para ejecutar diversas cargas: secuenciales interactivas y paralelas.

El objetivo de nuestro proyecto es la implementación de una *Máquina Virtual Paralela* (MVP), con una doble funcionalidad: por una parte, la ejecución de aplicaciones interactivas correspondientes a tareas del usuario local, y por otra parte, permitir lanzar aplicaciones paralelas que puedan aprovechar los recursos infrautilizados por las aplicaciones interactivas. Esta MVP debe proporcionar un entorno donde las aplicaciones paralelas obtengan un beneficio claro del hecho de ejecutarse en un entorno no dedicado, a la vez que minimice la interferencia de las aplicaciones paralelas sobre la interactividad de las cargas locales.

La consecución de estos objetivos comporta nuevos retos a afrontar por parte de la comunidad científica: el manejo eficiente de la multiprogramación de las tareas paralelas ejecutadas sobre la MVP, la predicción de los parámetros significativos de las aplicaciones a ejecutar en el cluster, el diseño de métricas de usuario a efecto de proporcionar una idea clara de la potencia del cluster y la implementación de técnicas de *Planificación Espacial* (P.E.: distribución de los procesadores entre las aplicaciones); suponen nuevos retos a afrontar por los investigadores.

Por otra parte, también es deseable que la solución propuesta incluya un sistema de coplanificación (*Planificación Temporal* – P.T.) que permita disminuir el tiempo de espera de las aplicaciones paralelas ante los eventos de comunicación y sincronización. De esta forma los recursos se utilizan más eficientemente y se optimizan los tiempos de ejecución de las aplicaciones paralelas.

En este trabajo se describe un nuevo entorno, denominado CISNE, que aprovecha los recursos de cómputo disponibles en un cluster no dedicado, para crear una MVP que cumpla con los objetivos citados anteriormente. La implementación de este entorno ha supuesto mayoritariamente modificaciones de middlewares existentes (PVM), pequeñas modificaciones en el kernel de los sistemas utilizados (Linux ver. 2.2), y desarrollo en espacio de usuario. La implantación por tanto, en entornos universitarios y centros de investigación o producción, supondría ampliar el abanico de posibles aplicaciones a ejecutar, o redimensionar aplicaciones ya existentes sin ningún coste adicional.

Este trabajo se estructura comenzando por la descripción del entorno que se ha implementado (CISNE). Luego se muestra una sección de experimentación utilizando el entorno definido. Finalmente, se detallan las conclusiones y trabajo futuro.

2. EL ENTORNO CISNE

El objetivo fundamental del entorno CISNE (*Cooperative & Integral Scheduling for Non-dedicated Environments*) es el de implementar la MVP que hemos citado anteriormente. De esta forma se han de afrontar problemas en dos aspectos ortogonalmente opuestos: el *Planificación Temporal* (P.T.) y el *Planificación Espacial* (P.E.). Para dar una propuesta que afronte el primer aspecto (P.T.) se ha utilizado un sistema llamado CSC (*CoScheduling Cooperante*) desarrollado por nuestro grupo. Dicho sistema proporciona un marco de ejecución dentro del cuál las aplicaciones paralelas pueden ser coplanificadas (coordinadas), a la vez que los recursos asignados a las mismas son balanceados y la capacidad de respuesta interactiva de las aplicaciones interactivas es preservada.

Desde el punto de vista del P.E., CISNE incorpora otro sistema implementado por nosotros denominado LoRaS (Long Range Scheduler), cuya responsabilidad es la de distribuir las aplicaciones que conforman la carga paralela sobre el cluster de ordenadores. Esta distribución debe realizarse considerando el estado del entorno, las cargas a lanzar y el funcionamiento del entorno CSC con el que debe integrarse.

La Figura 1 muestra la interacción entre ambos sistemas, tanto en el nodo servidor, utilizado para lanzar las aplicaciones paralelas, como en el resto de nodos compartidos entre las aplicaciones paralelas y las locales. A continuación se procederá a describir cada uno de los dos sistemas que integran el entorno CISNE

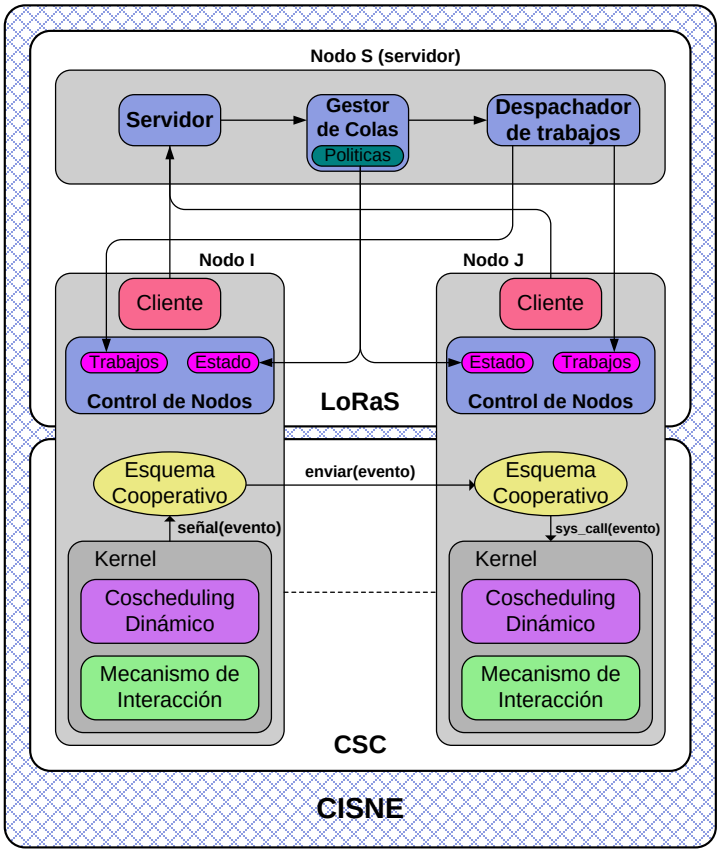


Figura 1. Arquitectura del sistema CISNE

2.1 Sistema de *Planificación Temporal*: CSC

CSC se encarga de la gestión temporal de los recursos de cómputo asignados, tanto a las tareas paralelas como a las locales, a lo largo del cluster. De acuerdo con este propósito, CSC tiene en cuenta los siguientes tres objetivos:

- *El rendimiento de las tareas locales debe preservarse*. Los usuarios locales sólo permitirán la ejecución de tareas paralelas en sus máquinas, si esto no implica una ralentización del entorno y de su capacidad de respuesta. Por esta razón, CSC debe limitar tanto los recursos como el grado de multiprogramación de las aplicaciones paralelas, de modo que se preserve el rendimiento de las tareas locales dentro de unos márgenes aceptables por los usuarios de los mismas.
- *La coplanificación de las tareas paralelas que se comunican*. Un problema intrínseco de las técnicas de tiempo compartido es el modo de garantizar la planificación simultanea (coplanificación) de aquellas tareas que se comunican entre sí, denominadas tareas cooperantes. El objetivo de la coplanficación, como se puede observar en la Figura 2, es minimizar el tiempo de espera ante los eventos de comunicación y sincronización, tiempo indicado como $T_{s\text{-}r}$. En función del mecanismo utilizado para coplanificar las tareas, las técnicas de coscheduling se clasifican en técnicas *con control explícito*, basadas en planificar simultáneamente todas las tareas de una misma aplicación mediante un cambio de contexto global y *técnicas con control implícito*, caracterizadas por predecir la necesidad de coscheduling a partir de la información local, obtenida en cada nodo, ante la ocurrencia de ciertos eventos de comunicación y sincronización. La flexibilidad

asociada a las técnicas implícitas permite que estas se puedan adaptar dinámicamente a la alta variabilidad de un entorno cluster no dedicado, hecho que aconseja su uso en este tipo de entornos.

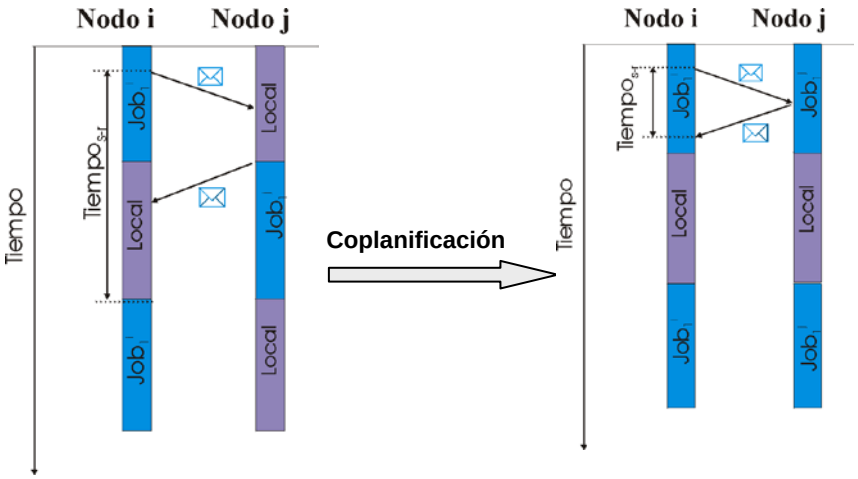


Figura 2. Ganancia que surge de la utilización de un entorno de coplanificación.

- *El balanceo de recursos de CPU y memoria asignados a cada tarea paralela perteneciente a un mismo trabajo a lo largo del cluster*. La coplanificación, entendida de manera tradicional, no es suficiente para asegurar la progresión de múltiples aplicaciones paralelas en un cluster no dedicado. Esto es debido a que las tareas que forman una misma aplicación distribuida deben de disponer de una prioridad similar, en la asignación de recursos, tanto de CPU como de memoria, en cada uno de los nodos donde se ejecutan, de lo contrario unas tareas progresarán más rápido que otras, lo que comportará una descoordinación global y, por tanto, un deficiente rendimiento de la aplicación distribuida. Con objeto de paliar este comportamiento, los recursos asignados a las tareas que forman una misma aplicación deben ser asignados de un modo uniforme a lo largo del cluster.

2.1.1 Arquitectura

CSC, como puede observarse en la Figura 3, está constituido por tres diferentes módulos. Cada módulo se encarga de alcanzar uno de los objetivos anteriormente citados. A continuación se describe la funcionalidad de cada uno de estos tres módulos.

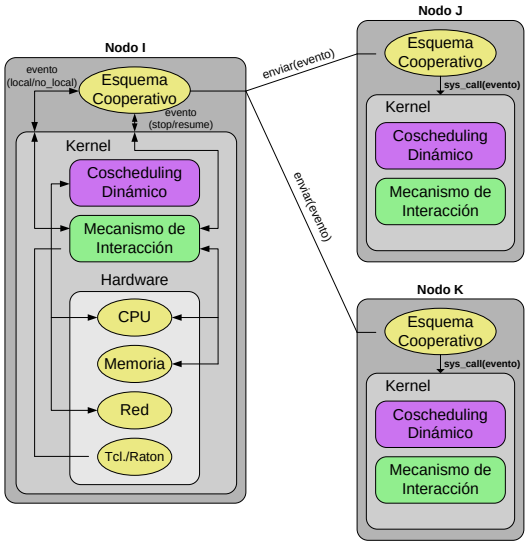


Figura 3 Arquitectura del sistema CSC

Coscheduling Dinámico

Las estrategias de coplanificación implícitas permiten que las aplicaciones paralelas compartan los recursos disponibles con tareas interactivas en un entorno NOW. De acuerdo con diferentes estudios mostrados en la literatura [3, 4], CSC utiliza una estrategia de coplanificación dinámica con bloqueo inmediato. Esta estrategia consiste en que una tarea que espera un mensaje se bloquea de inmediato, de modo que tras la recepción del mensaje esperado, es depertada y planificada, incluso causando la preempción del proceso actualmente en ejecución.

Mecanismo de interacción

A efecto de asegurar el progreso de los trabajos paralelos sin introducir una sobrecarga excesiva en las tareas locales, lo que se propone es la utilización de un *contrato social* [5]. Esto significa establecer un acuerdo acerca del porcentaje mínimo de recursos de CPU y memoria, asignados a las tareas paralelas dentro de cada nodo. Con respecto a la CPU, una política con contrato social comporta que la longitud del quantum asignado a las tareas paralelas sea limitada proporcionalmente a dicho porcentaje. Con respecto a la memoria, el contrato social conlleva que la memoria sea dividida, proporcionalmente a dicho porcentaje, en dos porciones: una asignada a las tareas paralelas y el resto a las locales. De este modo, en aquellas situaciones donde la memoria sea desbordada, la aplicación paralela con mayores requerimientos de memoria será detenida y desalojada de memoria. Este modo de proceder significa que CSC varía dinámicamente el grado de multiprogramación de las aplicaciones paralelas, de acuerdo con los requerimientos de memoria de las tareas, tanto locales como paralelas.

Esquema Cooperativo

El balanceo de recursos entre los procesos que forman una misma aplicación distribuida no puede ser alcanzado utilizando exclusivamente la información proporcionada por el nodo local, si no que es necesario el intercambio de información entre aquellos nodos en los que residen tareas cooperantes. La cuestión a plantear se centra en cuál es la información que se debe compartir, considerando que un exceso puede causar que la eficiencia del sistema disminuya, incrementando su complejidad y limitando su escalabilidad. Por esta razón, sólo aquellos eventos locales ocurridos en un nodo que provoquen una reasignación de los recursos de cómputo asignados a las tareas paralelas, como consecuencia de la aplicación de la política de contrato social, serán notificados. En concreto, cada nodo enviará los siguientes cuatro eventos al resto de nodos que tienen tareas pertenecientes a la misma aplicación penalizada por dicha reasignación:

- *LOCAL*: cuando se registra actividad por parte del usuario local en el nodo emisor.
- *NO_LOCAL*: cuando ya no se registra actividad del usuario local en el nodo emisor.
- *STOP*: cada vez que una tarea paralela en el nodo ha sido detenida.
- *RESUME*: cada vez que una tarea paralela en el nodo es reanudada.

De este modo, cuando un nodo determinado reciba la notificación de un evento específico reasignará los recursos de acuerdo con su estado actual y el tipo de evento recibido.

2.2 Sistema de Planificación Espacial: LoRaS

Desde el punto de vista de la plataforma de Planificación Espacial se debe tratar el problema de la distribución de las aplicaciones paralelas sobre el conjunto de nodos que se disponen. Para esto se han de considerar ciertos aspectos que se pueden observar en la Figura 4.

Considerando la *división del espacio de procesadores*, que se muestra en dicha figura, se han de definir criterios desde dos aspectos separados:

- *Tipo de particionamiento*: determina la flexibilidad con que se puede dividir el conjunto de procesadores en particiones asignables a las aplicaciones.
- *Selección de nodos*: se definen los criterios utilizados para determinar la calidad de un nodo en un momento para lanzar un trabajo determinado. Alternativas en este

sentido son: el MPL, la existencia de usuarios locales, el balanceo de carga a lo largo del cluster y la carga de memoria o CPU.

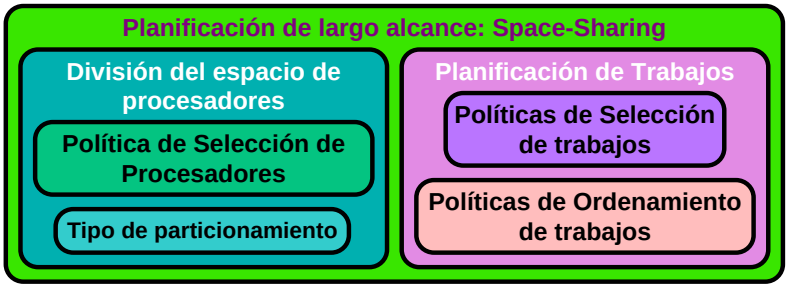


Figura 4 Taxonomía aplicada al Planificación Espacial

El otro aspecto que se ha de afrontar es la *planificación de trabajos*, la cual trata con los criterios que se han de adoptar a efecto de *seleccionar un trabajo* para su ejecución, o por otra parte las políticas utilizadas para *ordenar* (*priorizar*) *los trabajos* a medida que llegan al sistema para ser ejecutados. Cabe destacar que en la literatura normalmente estos aspectos se consideran en conjunto, pero en nuestra propuesta los hemos separado de forma tal que pueden combinarse para obtener criterios más complejos al momento de tener que planificar un trabajo.

2.2.1 Arquitectura

Considerando la cantidad de aspectos relevantes a tener en cuenta, el sistema LoRaS se ha diseñado con dos objetivos:

1. Facilitar la integración de CSC como entorno de *Planificación Temporal*.
2. Proporcionar un entorno flexible que permita la implementación de diversas políticas de planificación de forma simple.

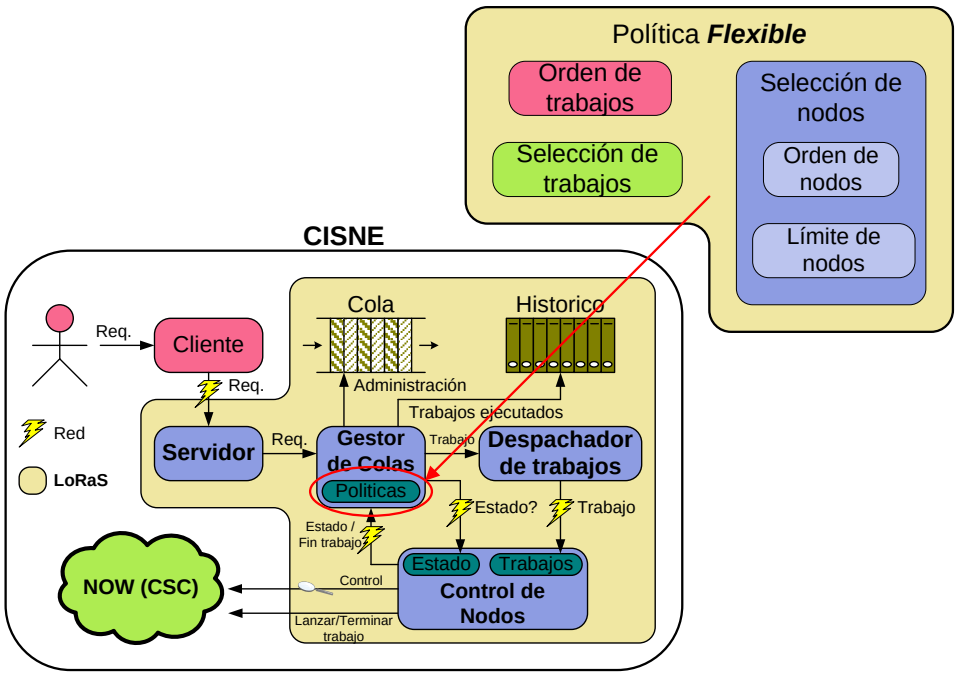


Figura 5. Arquitectura del esquema LoRaS y la política Flexible

La arquitectura que se ha adoptado para alcanzar estos objetivos puede verse en la Figura 5. En dicha figura podemos observar el módulo *servidor,* que es quién recibe las peticiones de ejecución y les da formato antes de entregarlo al módulo *gestor de colas*. El *gestor de colas* tomará las decisiones de planificación adecuadas en función de la política implementada. Dicha política podrá tomar sus decisiones utilizando el estado del entorno, las características de la aplicación y hasta un registro histórico de las aplicaciones ejecutadas. Si al

llegar al sistema un trabajo no puede ser ejecutado, el *gestor* lo colocará en una cola de espera. Una vez que se ha decidido lanzar una aplicación, el gestor interactúa con el módulo *despachador*, que es quién conoce cómo debe lanzarse la aplicación en función del tipo que sea (PVM, MPI, sockets, etc.). Para realizar su tarea el módulo *despachador* entrega la petición al módulo de *control de nodos*, del nodo donde debe lanzarse la aplicación (este módulo se encuentra como un proceso dentro de cada nodo del cluster), y éste finalmente es el encargado de lanzar y controlar la finalización del trabajo. Además, el módulo de *control de nodos* es el responsable de recolectar y aportar información sobre el estado del entorno al gestor de colas.

Cabe destacar que para que el sistema sea funcional, debe poseer una serie de criterios dentro del gestor de colas que tomen las decisiones de planificación. En este caso se ha implementado una política (*Flexible*), que toma en consideración los aspectos planteados en la Figura 4, combinando políticas de selección de trabajos con políticas de ordenamiento (priorización). Si bien los módulos de *orden de trabajos* y *selección de trabajos* son independientes y puede combinarse cualquier criterio implementado en uno con cualquiera implementado en el otro, a efecto del presente estudio solo tres configuraciones (*mezclas*) se han evaluado (Orden-Selección):

- *SJF-JFirst*: la política de ordenamiento *SJF* prioriza los trabajos en orden creciente de su tiempo de ejecución, mientras que la política de selección *JFirst* evalúa solamente el primer trabajo en la cola de ejecución para determinar si es factible lanzarlo. De esta forma obtenemos un criterio de planificación que respeta completamente el orden de los trabajos, pero a costa de no realizar el mejor aprovechamiento de recursos, debido a que solo consideramos el trabajo que se encuentra en la cabeza de la cola y otros podrían ser mejores candidatos.

- *FCFS-FFit*: en este caso los trabajos son ejecutados en el orden en que arriban (*FCFS*), lo que proporciona un marco justo para las aplicaciones, libre de starvation, que garantiza que ningún trabajo deberá esperar por la ejecución de otro que llegue después al sistema. Desde el punto de vista de la política de selección de trabajos se utiliza el criterio de seleccionar el primer trabajo que se encuentra en la cola (*FFit*), recorriéndola según el orden impuesto por *FCFS*, y que pueda ser ejecutado bajo el estado actual del entorno. De esta forma obtenemos un criterio combinado que balancea el orden de los trabajos con el aprovechamiento de recursos.

- *SNPF-BFit*: bajo este criterio de priorización (*SNPF*) los trabajos se ordenan (de forma creciente) en función de la cantidad de procesadores solicitados. Sin embargo al combinar este criterio con una política de selección de tipo *BFit*, que examina toda la cola de trabajos para encontrar aquel que haga una mejor utilización de los recursos restantes, hace que el criterio de ordenamiento sea ignorado. Por ende, solamente se considera el estado del entorno y la utilización de recursos, dejando de lado la prioridad de los trabajos.

De lo especificado para cada *mezcla* se puede deducir que las políticas de selección de trabajos determinan la influencia de las políticas de ordenamiento. De esta forma las tres mezclas propuestas proporcionan entornos que priorizan los trabajos (SJF-JFirst), el estado del entorno (SNPF-BFit) y que balancean los procesos de selección y orden de trabajos (FCFS-FFit).

Desde el punto de vista de los criterios utilizados para *seleccionar un conjunto de nodos* para ejecutar una aplicación determinada, estos están constituidos a partir de dos módulos: el de *limite de nodos*, que elimina del conjunto de nodos del cluster aquellos en los que ya no sea posible incrementar la carga paralela, y el de *orden de nodos*, que es responsable de elegir de entre los nodos restantes, luego de aplicar el proceso de limitación, aquellos más idóneos para lanzar una aplicación. Los criterios utilizados para ambos módulos son el *MPL* (de las aplicaciones lanzadas por el sistema) y la *carga de memoria*. Cabe destacar que el MPL máximo que se ha considerado en el presente estudio es de 4, basado en los resultados obtenidos en [7] para el entorno CSC.

2.2.2 Implementación de CISNE en un cluster PVM-Linux

En la implementación del sistema CISNE se ha utilizado el entorno descrito en la Figura 6. En dicha figura se puede observar que el sistema CISNE se encuentra implementado a diferentes niveles en función del módulo que se trate. El sistema de P.T. (CSC) se encuentra parcialmente implementado dentro del kernel de Linux en lo referente a los módulos de: los *mecanismos de interacción* y la *coplanificación dinámica*. Esto es así porque los tiempos en que deben tomarse las decisiones de planificación sólo pueden alcanzarse dentro del kernel. Por otro lado, el módulo de *balanceo de recursos* se encuentra implementado dentro del demonio de PVM. De esta forma se aprovecha la estructura de comunicaciones que PVM ya posee implementada.

El entorno LoRaS, por su parte, se encuentra íntegramente implementado en espacio de usuario, a partir de un diseño realizado en el lenguaje de modelado UML, he implementado luego utilizando C++. Para desarrollar el sistema LoRaS por completo, se han requerido aproximadamente unas 10000 líneas de código entre la plataforma general y la política *Flexible* implementada a efecto de este estudio.

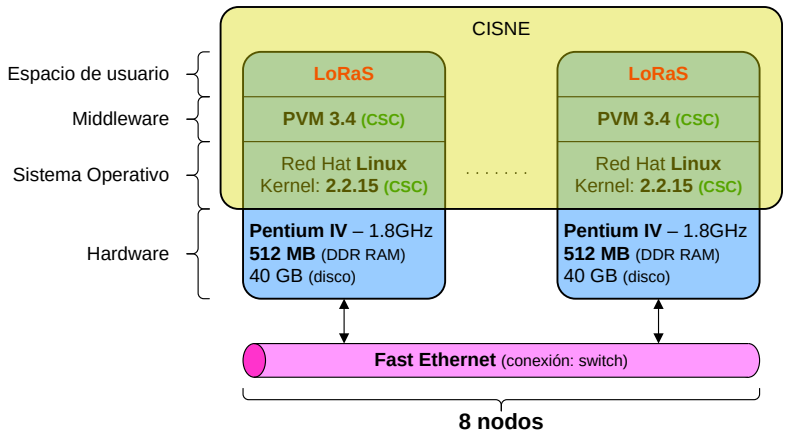


Figura 6 Entorno de implementación del sistema CISNE.

3. EXPERIMENTACIÓN

El objetivo de la experimentación en este trabajo es el de analizar las *mezclas* de políticas definidas en la especialización del entorno LoRaS, bajo diferentes estados del entorno y para cargas diversas. De esta forma podremos caracterizar la *estabilidad* (baja dispersión de las métricas) de las diferentes alternativas (mezclas), lo que a su vez nos permitirá en un futuro garantizar al usuario paralelo algunos parámetros, como el tiempo de ejecución o de espera de las aplicaciones que se lancen. Cabe destacar que para realizar estas pruebas se ha utilizado el entorno como se ha definido en la Figura 6.

De esta forma, antes de realizar la experimentación, será necesario definir las cargas a utilizar, los diferentes escenarios en los que se han utilizado y cuál ha sido el proceso de evaluación.

3.1 Cargas utilizadas

Considerando el entorno sobre el que se pretende aplicar el sistema deberemos construir dos tipos de cargas: una *carga paralela*, que constituye el cómputo distribuido que deseamos realizar y una *carga local* que nos permite representar un entorno no dedicado.

La carga paralela se define como una "*secuencia de peticiones de ejecución de aplicaciones paralelas*". Dicha secuencia se ha generado utilizando una aplicación desarrollada por Feitelson [6], la cual utiliza una serie de distribuciones estadísticas (basadas en caracterizaciones de entornos como la Universidad de Berkley, el LANL o el LLNL), para construir un listado de *arribos de peticiones de ejecución*. A cada petición en el listado, *luego* se le ha asociado una aplicación paralela, tomada del conjunto formado por las aplicaciones IS (orientada a comunicación), y MG (orientada a cómputo) del conjunto de benchmarks del NAS.

De esta forma se han generado las siguientes cargas: orientada a *comunicación* (75% de aplic. IS y 25% de aplic. MG – *carga de red*), orientada a *cómputo* (25% de aplic. IS y 75% de aplic. MG – *carga de CPU*) y *balanceada* (50% de aplic. IS y 50% de aplic. MG – *carga mixta*).

Las cargas locales por su parte, se han generado a partir de un conjunto de benchmarks construidos para caracterizar el comportamiento del usuario local. Para representar de una forma realista el comportamiento de estos usuarios, se ha analizado su comportamiento en un laboratorio de docencia perteneciente a la Universidad de Lleida [2]. Con la información obtenida se han creado 3 perfiles que nos permiten caracterizar al conjunto de usuarios: un usuario de tipo *XWindows* que posee altos requerimientos de interactividad (62% del total de usuarios), uno que hemos denominado de tipo *Internet* cuya mayor actividad es la navegación por la web (33% del total de los usuarios) y uno que hemos llamado de tipo *Shell*, con poca necesidad de interactividad pero con una mayor necesidad de poder de cómputo (5% del total de los usuarios).

3.2 Escenarios evaluados

Una vez que hemos construido las cargas es importante definir cómo se han utilizado para realizar la experimentación. Dicha evaluación se ha realizado a partir de 3 escenarios que podemos observar en la Tabla 1.

	Entorno	Paginación
Escenario 1	Dedicado	No permitida
Escenario 2	No dedicado	No permitida
Escenario 3	No dedicado	Permitida

Tabla 1 Escenarios en los que se ha evaluado el sistema.

Cada uno de estos escenarios se ha evaluado para el conjunto de mezclas de políticas definidas anteriormente (SJF-JFirst, FCSF-FFit, SNPF-BFit).

En el caso del entorno *no dedicado* se han utilizado cargas locales en 4 de los 8 nodos que constituyen nuestro entorno de ejecución (Figura 6). Desde el punto de vista de la paginación, se ha permitido entre un 10% y un 15% por sobre el tamaño de la memoria principal.

Cabe destacar que todos estos escenarios se han evaluado para todas las cargas paralelas generadas (carga de *Red*, *CPU* y *Mixta*), las cuales han sido ejecutadas al menos 5 veces cada una para cada escenario y mezcla de políticas definida.

3.3 Proceso de evaluación

Considerando que el objetivo de la experimentación es el de determinar la estabilidad de las mezclas de políticas descritas anteriormente, hemos realizado el siguiente proceso de evaluación:

- *Evaluar un conjunto de métricas*: en primer lugar hemos evaluado un conjunto de métricas desde el punto de vista del *usuario paralelo* (turnaround[1] y slowdown[2]) y del *sistema* (makespan[3]). Es importante destacar que con el objetivo de caracterizar la estabilidad de las mezclas, para cada métrica evaluada se ha considerado además su desviación estándar. Así mismo se han tomado mediciones respecto de la incidencia de las cargas paralelas sobre la capacidad de respuesta de las aplicaciones locales (latencia de las llamadas al sistema).
- *Ponderar las métricas evaluadas*: una vez obtenidos los resultados para las métricas evaluadas y a efecto de valorar de forma general las mezclas utilizadas, se han ponderado las métricas obtenidas. De este modo es factible analizar de forma

[1] **Turnaround** = tiempo de espera en la cola + tiempo de ejecución de la aplicación.
[2] **Slowdown** = tiempo de ejecución (aplic.) en un entorno no dedicado / tiempo de ejecución en un entorno dedicado
[3] **Makespan** = slowdown aplicado a la carga (en lugar de a cada aplicación en particular).

global el comportamiento de cada mezcla, y en definitiva la estabilidad que muestran en cada caso. Para realizar este proceso de ponderado se ha definido la siguiente expresión:

$$V(mezcla) = \left(\sum métrica_i \times peso\_métrica_i\right) + \left(\sum desv\_métrica_i \times peso\_desv\_métrica_i\right)$$

Dicha expresión se ha aplicado al conjunto de todas las métricas de usuario paralelo y sistema, valorando más la desviación de cada métrica (estabilidad) que la métrica en si (Figura 7.b). Cabe destacar que a menor valor de la expresión más estabilidad reportará la mezcla.

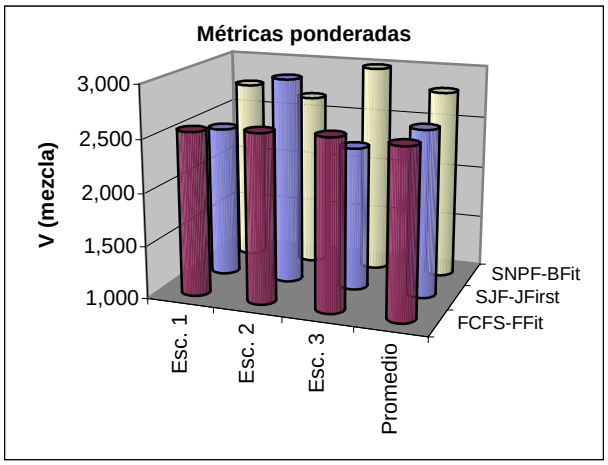
3.4 Resultados

Dentro de la experimentación realizada se deben considerar dos aspectos separados pero ambos relevantes: por una parte la utilidad del sistema para el usuario paralelo, que viene dada a partir de la estabilidad, y por ende predecibilidad de las políticas evaluadas; y por otra parte la nula incidencia de las cargas paralelas en la capacidad de respuesta de las aplicaciones locales.

3.4.1 Resultados obtenidos desde el punto de vista del usuario paralelo

Haciendo uso de los elementos descritos hasta aquí en la experimentación, hemos generado la Figura 7.a, donde se puede observar la aplicación de la expresión *V(mezcla)* a las mezclas evaluadas en cada escenario sobre el que hemos ejecutado las cargas.

De esta figura cabe destacar dos aspectos relevantes. En primer lugar la mezcla de políticas FCFS-FFit (que balancea el proceso de selección de trabajos con el de priorización de los mismos), obtiene una gran estabilidad respecto de la obtenida por las otras mezclas. En el caso de la mezcla SJF-JFirst podemos observar buenos resultados para entornos muy cargados (esc.1) o muy descargados (esc. 3), pero con una inestabilidad mayor para el escenario 2, que es el que más interesa en nuestro entorno por ser el caso no dedicado pero sin cargar demasiado el sistema. SNPF-BFit por su parte, muestra valores bastante mayores en cualquier caso, lo que tiene sentido si consideramos que la políticas solamente considera el estado del entorno y obvia el orden de los trabajos por completo.



a)

Métricas	Pesos
Slowdown – ***Sl***	**3,0**
Slowdown – desv. estándar – $\boldsymbol{Sl_{avg}}$	**2,6**
Turnaround – ***Tr***	**1,8**
Turnaround – desv. estándar – $\boldsymbol{Tr_{avg}}$	**1,0**
Makespan – ***Mk***	**3,0**
Makespan – desv. estándar – $\boldsymbol{Mk_{avg}}$	**2,6**

b)

Figura 7 Valores de V(mezcla) obtenidos para las mezclas evaluadas (a) utilizando los pesos de la tabla (b) para las métricas ponderadas.

Por otra parte podemos observar que FCFS-FFit no solo da buenos resultados en promedio, sino que además, e independientemente del escenario, el valor de estabilidad encontrado es prácticamente el mismo. De esta forma podemos decir que la mezcla es estable

independientemente del estado del entorno, lo cual favorece la predecibilidad en cuanto a los tiempo de espera y ejecución de las aplicaciones paralelas.

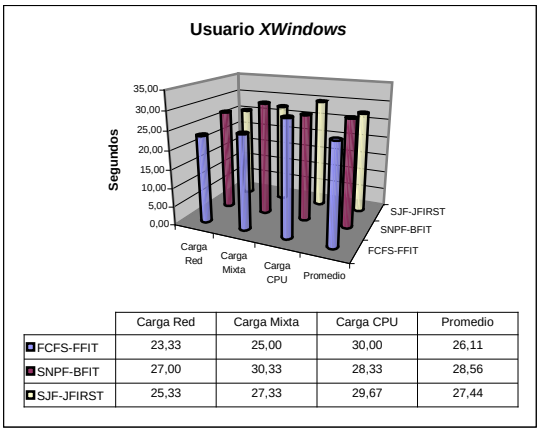
3.4.2 Resultados obtenidos desde el punto de vista del usuario local

Si bien hemos visto que es posible obtener políticas estables de planificación, es importante además que estas políticas no sobrecarguen el sistema a niveles tales, que la capacidad de respuesta interactiva se vea disminuida de forma que el usuario local sea capaz de detectarlo. Para poder observar la incidencia de la carga paralela en el sistema y cómo esto afecta a la carga local se ha evaluado la *latencia de las llamadas al sistema*. Los valores correspondientes a un entorno descargado, con un usuario local, y con límites aceptables de latencia, pueden observarse en la Tabla 2. Cabe destacar que los límites que se muestran en dicha tabla se han calculado para el entorno de evaluación definido, utilizando instrumentación incluida en los benchmarks de las cargas locales.

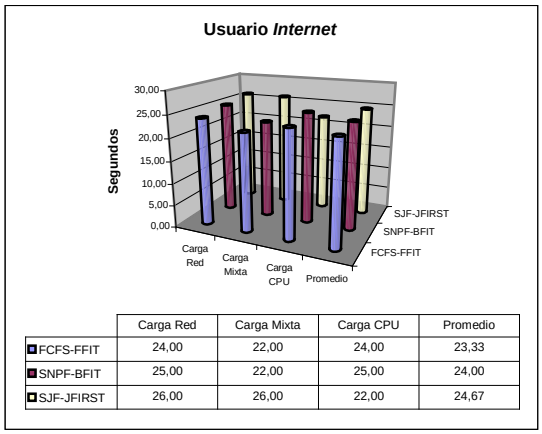
Estado del entorno	**Latencia**
Totalmente inactivo	**13** μseg
Usuario *XWindows*	**14** μseg
Usuario *Internet*	**14** μseg
Usuario *Shell*	**14** μseg
Límites aceptables	
Pérdida de capacidad levemente detectable	**32** μseg
Pérdida de capacidad intolerable	**56** μseg

Tabla 2 Valores relevantes respecto de la latencia de las llamadas al sistema

Cabe destacar que estos valores dados en microsegundos no tienen significado representativo por su valor absoluto, ya que ningún usuario local podrá detectar un cambio en la latencia de la magnitud planteada. Sin embargo, su relevancia está dada por el volumen de llamadas al sistema que realiza una aplicación interactiva (entre el 40% y el 50% del tiempo total de procesamiento). De esta forma los valores dados en microsegundos se traducen en milisegundos en cuanto a la capacidad de respuesta de la aplicación. Dichos valores sí son detectables por el usuario local.



	Carga Red	Carga Mixta	Carga CPU	Promedio
FCFS-FFIT	23,33	25,00	30,00	26,11
SNPF-BFIT	27,00	30,33	28,33	28,56
SJF-JFIRST	25,33	27,33	29,67	27,44



	Carga Red	Carga Mixta	Carga CPU	Promedio
FCFS-FFIT	24,00	22,00	24,00	23,33
SNPF-BFIT	25,00	22,00	25,00	24,00
SJF-JFIRST	26,00	26,00	22,00	24,67

a) **b)**

Figura 8 Latencia promedio en las llamadas al sistema para los usuarios XWindows (a) e Internet (b).

En la Figura 8 se puede observar que para los usuarios que se han evaluado[4], la latencia en las llamadas al sistema nunca alcanza el límite, prácticamente indetectable, de 32 microsegundos definido en la Tabla 2. La evaluación que se muestra en dicha figura

[4] el usuario *shell* no se contempla debido a su baja frecuencia y su falta de necesidad de interactividad.

corresponde al escenario 3 (cargas de red, CPU y mixta con paginación permitida), donde la carga impuesta sobre el sistema es la mayor de todas las evaluadas en la experimentación, y por ende el peor de los casos para nuestro entorno.

De esta forma podemos asegurar que las aplicaciones paralelas no imponen una carga que afecte a las aplicaciones locales más allá de límites detectables y que por ende se cumple con otro de los objetivos establecidos para el sistema CISNE.

4. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se presenta una alternativa para implementar una máquina virtual paralela que permite aprovechar los recursos ociosos, presentes en una red de ordenadores, para realizar cómputo paralelo. Dicha alternativa se implementa a partir de un entorno llamado CISNE que integra dos sistemas: CSC que afronta el problema de la *planificación temporal* y por otra parte LoRaS que ataca cuestiones referentes a la *planificación espacial*.

El objetivo del trabajo ha sido mostrar el funcionamiento del sistema propuesto, desde el punto de vista de la estabilidad que presentan un conjunto de métricas, para diferentes estados del entorno y cargas paralelas variadas. Todo esto además, sin incidir en la capacidad de respuesta interactiva de cada nodo del cluster. De esta forma hemos llegado a la conclusión de que, un criterio de planificación que balancea la prioridad de los trabajos con el estado del entorno en un momento determinado, es más estable que criterios que se orientan en uno u otro sentido.

En el futuro pretendemos agregar consideraciones de tiempo al proceso de selección de trabajos de forma tal que sea factible no sólo analizar el estado del entorno actual, sino también en el futuro. Con esto seremos capaces de garantizar al usuario paralelo algunos parámetros como el tiempo de espera o el tiempo de ejecución de sus aplicaciones, dentro de un margen concreto y con una certeza determinada.

5. BILIOGRAFIA

1. A. Acharya y S. Setia **– "Availability and Utility of Idle Memory in Workstation Clusters" –** En *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Volume 27, pp 35/46,* Atlanta, Georgia, United States **–** Junio 1999**.**

2. Francesc Giné – "**CoScheduling Cooperativo: Una Propuesta de CoScheduling Orientada a Clusters No Dedicados Multiprogramados**" – *Tesis Doctoral, Departamento de Informatica, Universitat Autònoma de Barcelona* – Junio 2004**.**

3. F. Solsona – "**Coscheduling Techniques for Non-dedicated Cluster Computing**" – *Tesis Doctoral, Departamento de Informatica, Universitat Autònoma de Barcelona* – Julio 2002.

4. A. B. Yoo y M. A. Jette – "**A coscheduling technique for large symmetric multiprocessor clusters**" – En *Job Scheduling Strategies for Parallel Processing, volume 2221 of Lecture Notes in Computer Science, pages 21-40* – 2001.

5. R. H. Arpaci, A. C. Dusseau, A. M. Vahdat, L. T. Liu, T. E. Anderson y D. A. Patterson – "**The interaction of parallel and sequential workloads on a network of workstations**" – En *Proceedings of ACM SIGMETRICS/PERFORMANCE'95, pp 267-278* – 1995.

6. D. G. Feitelson – "**Packing schemes for gang scheduling**" – En *Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag. Lecture Notes in Computer Science, Volume 1162, pp. 89-110* – 1996.

7. M. Hanzich, F. Giné, P. Hernández, F. Solsona, L. Lérida y E. Luque – **"Coscheduling and Multiprogramming Level on a non-dedicated Cluster"** – En *EuroPVM/MPI 2004, Budapest, Hungría. Lecture Notes in Computer Science* – Septiembre 2004.