Rendimiento de Estrategias de Calendarización Considerando Fluctuación de Tiempo de Ejecución de Tareas en un Grid Computacional

Performance of Scheduling Strategies Considering Fluctuation of Tasks Execution Time in a Computational Grid

[1]Juan Manuel Ramírez, [2]Ana Isabel Rodríguez, [3]Andrei Tchernykh, [2]Jesús Alberto Verduzco
[1]Universidad de Colima, México, jmramir@ucol.mx
[2]Instituto Tecnológico de Colima, Colima, México, kidoki42@hotmail.com, javrtesis@yahoo.fr
[3]Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada, BC México, chernykh@cicese.mx

Resumen

Este documento describe el análisis experimental de estrategias de calendarización de un GRID computacional utilizando un modelo jerárquico de dos niveles. En el primer nivel, un broker asigna cada tarea del usuario a un recurso del conjunto disponible. En el segundo nivel, cada recurso crea un calendario de ejecución para las tareas paralelas asignadas, mediante su propio calendarizador local. Dado que generalmente el tiempo de ejecución de las tareas, es diferente al tiempo estimado por los usuarios, hemos realizado una evaluación del rendimiento de las estrategias considerando esta variación que llamamos fluctuación de tiempo. Las estrategias se han evaluado utilizando el simulador GSimula, considerando logs de carga de trabajo de supercomputadoras reales.

Abstract

This paper describes the experimental analysis of scheduling strategies for a computational Grid using two level hierarchy model. At the first level, a broker allocates computational jobs to the resource from a set of available ones. At the second level, each resource generates a schedule of the parallel jobs assigned to it by its own local scheduler. Based on the fact that the actual job execution time is different from the estimation provided by users, we made a performance evaluation of Grid scheduling strategies, by considering such a variation called time fluctuation. Scheduling strategies are evaluated considering workloads of real supercomputers.

1. Introducción

1.1. Grid

Desde los orígenes de la computación uno de los objetivos fundamentales ha sido el procesamiento de mayores cantidades de información, en menor tiempo y con mayor precisión, teniendo como premisa el consumo de la menor cantidad de recursos posibles. La necesidad creciente de recursos de cómputo e infraestructuras que proporcionen el soporte para estos recursos dio origen a conceptos como: programación paralela, clusters, P2P, y GRID, siendo este último el que toma realce entre los restantes, debido a las implicaciones derivadas de este concepto.

En la actualidad existen numerosos proyectos que buscan utilizar el poder de cómputo ocioso de miles o millones de computadoras personales alrededor del mundo tales como: *Project Grid '5000* [3], *Compute Against Cancer* [4], *United Devices Inc.* [5], *Folding@Home* [6], *Genome@Home* [7], *Evolution@Home*, [8], *Golem@Home* [9], *Mars Clickworkers* [10], *Climate Prediction* [11], *Gamma Flux* [12]. Estos proyectos son un claro ejemplo de la existencia de problemas de gran reto, que exigen una gran capacidad de procesamiento.

Un GRID[1,23] se puede definir como una infraestructura de cooperación entre instituciones que disponen de una serie de recursos de cómputo de uso colectivo. Con esto nos referimos a que instituciones como universidades, empresas, centros de investigación y otros grupos, se unen formando una comunidad que le permite a cada usuario trabajar con mayores recursos de cómputo a un bajo costo, contar con un sistema tolerante a fallas[2] y tener al alcance la capacidad de balanceo de carga del sistema de manera automática.

Entre los proyectos de Grid desarrollados en Estados Unidos podemos mencionar: *TERAGRID (NSF)* [13] que es una infraestructura abierta de descubrimiento científico, *National Middleware Initiative (NSF NMI)* [14], proyecto que trata necesidades críticas para infraestructura de software con el fin de apoyar la

investigación científica y de ingeniería, *Earth System Grid (ESG)* [15], plataforma de alto desempeño para ser usada en la investigación del clima, *NEESgrid* [16], proyecto creado para realizar estudios de terremotos, *BIRN* (Biomedical Informatics Research Network) [17], desarrollado para investigar nuevas técnicas para el tratamiento de las enfermedades.

También existen proyectos desarrollados en Europa y Asia, entre ellos podemos mencionar el *European DataGrid* [20], el *ApGrid* [18], y el *TW Grid* [19].

Actualmente existe una gran variedad de organismos que intentan cubrir la creciente demanda de sistemas GRID con desarrollos propios. Entre estos destaca el Globus Alliance [21] que es una comunidad de organizaciones e individuos que desarrollan tecnologías fundamentales para construir infraestructuras de GRID, La aplicación principal desarrollada por la Globus Alliance es el Globus Toolkit [21,22,34], el cual forma parte de la infraestructura de muchos proyectos de Grid actuales, tanto comerciales como científicos.

1.2. Arquitecturas de C-GRID

La arquitectura de C-GRID para sistemas manejadores de recursos [23] es influenciada por la manera en la que el calendarizador está estructurado. La estructura del calendarizador depende del número de recursos en los cuales los trabajos son calendarizados y el dominio en el cual los recursos son localizados. En base a esto existen tres modelos de calendarización: *centralizada*, *descentralizada* y *jerárquica*. Un descripción completa de estos modelos se puede encontrar en [26]. En este investigación nosotros nos enfocamos en el modelo jerárquico (Figura 1).

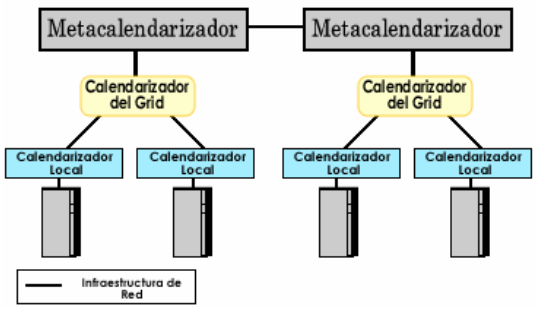

Figura. 1. Calendarizador de estructura jerárquica.

1.3. Estructura del artículo

La siguiente parte del artículo está estructurada de la siguiente manera. En la sección 2 se muestra la calendarización jerárquica de dos niveles, en la sección 3 se proponen los criterios de evaluación, en la sección 4 se hace un análisis experimental, finalmente, en la sección 5 se presentan las conclusiones y trabajo futuro.

2. Calendarización jerárquica de dos niveles

2.1. Estrategias para asignación de trabajos

Las estrategias para asignación de trabajos en este modelo de C-GRID permiten tanto la asignación, en cola, de trabajos listos para ejecutarse, como la selección de recursos y procesadores a utilizar. Actualmente trabajamos con múltiples estrategias de asignación de trabajos, agrupadas por el método que emplean para desempeñarse. Las estrategias que no necesitan crear un calendario previo tales como *Random, Min_Lp (Min Load per-proc), Min_PL (Min Parallel Load), Min Lower Bound (Min_LB),* y aquellas que si requieren datos de un calendario, y por lo tanto crean sus calendarios previos como *Min_CT (Min Completion Time), Min_SWCT (Min Sum of Weighted Completion Time), Min_WT (Min Waiting Time), Min_SWWT (Min Sum of Weighted Waiting TimeMin_U (Min Utilization), Min_ST (Min Start Time), Min_TA (Min Turnaround)* [28,41].

Las estrategias que no requieren calendario basan su decisión en los parámetros de los trabajos ya asignados y disponibles para el broker. En las estrategias que requieren calendario, se envía el trabajo que se va a asignar, a todos los nodos, y cada uno de ellos construye un calendario con este trabajo y los que ya tenga asignados. Cada nodo regresa la información resultante al broker y éste toma la decisión de asignación al nodo correspondiente según la estrategia utilizada, por ejemplo el promedio menor de tiempo de espera (Mean Waiting Time).

En este experimento utilizamos las siguientes estrategias para asignación de trabajos:

a) No requieren calendario previo:

Min_LP (Min Load per-proc). Selecciona el nodo con la mínima carga por procesador (número de trabajos sobre el número de procesadores en el nodo).

Min_PL (Min Parallel Load). Selecciona el nodo con la mínima carga paralela por procesador (suma de los tamaños de los trabajos sobre el número de procesadores en el nodo).

Min Lower Bound (Min_LB). Selecciona el nodo con la menor cota inferior de tiempo de terminación de los trabajos considerando los ya asignados más el que está por asignar. En lugar del tiempo actual de ejecución de un trabajo, que no está disponible en calendarización *non-clairvoyant*, es usado el valor proporcionado por el usuario al registrar el trabajo, o el tiempo de ejecución estimado.

b) Requieren calendario previo:

Min_CT (Min Completion Time). Se elige el nodo con el menor tiempo de terminación estimado de todos sus trabajos asignados, tomando en cuenta el trabajo por asignar.

Min_WT (Min Waiting Time). Estrategia que selecciona el nodo con el promedio menor de tiempo de espera de los trabajos asignados, incluyendo el trabajo por asignar.

Min_U (Min Utilization). Elige el nodo con la utilización mínima.

Min_ST (Min Start Time). Se selecciona el nodo que ofrece el menor tiempo de inicio de ejecución para la tarea a asignar.

2.3. Estrategias de calendarización local

Los algoritmos de calendarización local utilizados en este artículo son:

FCFS (First-Come-First-Serve): Es una de las estrategias más simples. El calendarizador comienza los trabajos en el orden en que se encuentra la lista localmente. Si no hay suficientes recursos disponibles, el calendarizador espera hasta que el trabajo pueda iniciarse. Los demás trabajos están detenidos en cola.

LSF (Largest-Size-First): Estrategia en la cual los trabajos en la cola local son organizados por tamaños en orden decreciente y posteriormente calendarizados con FCFS.

Backfilling-EASY-FirstFit: Busca llenar el espacio vacío del *bin*, ocasionado al no poder iniciar la ejecución inmediata de un trabajo, con otros trabajos más pequeños siempre y cuando éstos no retrasen la ejecución del trabajo de la cabeza de la cola.[29]

3. Criterios de evaluación

Para la evaluación del desempeño de las diferentes estrategias usamos criterios comunes de evaluación, los cuales los dividimos en tres: criterio del sistema, criterio del usuario y criterio del algoritmo de calendarización.

De acuerdo al criterio del sistema:

- **Utilización** (utilization). Se define como la fracción del tiempo en la cual el C-GRID fue utilizado $U_G = W_G /(C_G * m_G)$, donde W_G es la cantidad de trabajo del C-GRID, C_G es el tiempo de finalización de ejecución de todas las tareas en el C-GRID y m_G es el número total de procesadores en el C-GRID.
- **Rendimiento del procesamiento** (throughput). Es el número de tareas finalizadas por unidad de tiempo en el C-GRID. Se calcula como n / C_G donde n es el número total de trabajos en el C-GRID**.**

De acuerdo al criterio del usuario:

- **Tiempo de permanencia promedio** (mean turnaround time). Es el promedio del tiempo que tardan todas las tareas desde que entran a la cola local hasta que terminan su ejecución. Se calcula como $\frac{1}{n}\sum_{j=1}^{n} t_t^j$ donde $t_t^j = C^j - r^j$, C^j es tiempo de finalización de la tarea j, r^j es tiempo de entrega de la tarea j.
- **Tiempo de espera** (waiting time). Está definido como el tiempo promedio de espera de las tareas antes de iniciar su ejecución. Se calcula como $\frac{1}{n}\sum_{j=1}^{n} t_w^j$ donde $t_w^j = t_s^j - r^j$, t_s^j es tiempo de inicio de ejecución de la tarea *j*.
- **Coeficiente de respuesta** (response ratio)**.** Está definido como el promedio de los coeficientes de respuesta de todas las tareas. Se define como $\frac{1}{n}\sum_{j=1}^{n} (t_w^j + p^j)/\, p^j$, donde el coeficiente de respuesta para cada tarea es $(t_w^j + p^j)/\, p^j$, donde p^j es el tiempo de ejecución y t_w^j es el tiempo de espera de la tarea *j*.

De acuerdo al criterio del algoritmo de calendarización (estrategia de asignación) consideramos:

- **Coeficiente de desempeño** (competitive ratio). Es la medida de rendimiento del C-GRID definida como $\rho = C_G / C_{LB}$, donde C_G es el tiempo de la finalización de todas las tareas, y C_{LB} el tiempo mínimo posible para terminarlas calculado como $\max(W_G / m_G, p_G^{\max})$ donde $p_G^{\max}$ es el máximo tiempo de ejecución de las n tareas.

4. Análisis experimental

Actualmente la simulación es la única manera posible para que los sistemas distribuidos de gran escala de recursos heterogéneos con diferentes estrategias de asignación de recursos puedan ser evaluados, además es efectiva cuando se trabaja con problemas de gran escala que involucran un gran número de recursos y usuarios. También hace posible explorar diferentes tipos de

sistemas operando bajo una variedad de cargas de trabajos y algoritmos.

Existen pocas herramientas disponibles para simulaciones de aplicaciones de calendarización en ambientes de C-GRID. Los más conocidos son: Bricks [30], MicroGrid [31], SimGrid [32] y GridSim [33], sin embargo, las características de estos simuladores no se encontraron adecuadas para el propósito de esta investigación, por esta razón, fue creado el programa GSimula [28, 35, 36, 37] (actualmente en desarrollo) el cual evalúa múltiples estrategias de asignación de trabajos a recursos y varios algoritmos de calendarización local proporcionando resultados para criterios de evaluación comunes (explicados en la sección III). Permite la especificación de diferentes configuraciones de C-GRID y diferentes cargas de trabajo reales y sintéticas.

4.1. Modelo de simulación

Utilizamos un modelo de C-GRID con una estructura de calendarización jerárquica de dos niveles (Figura 2).

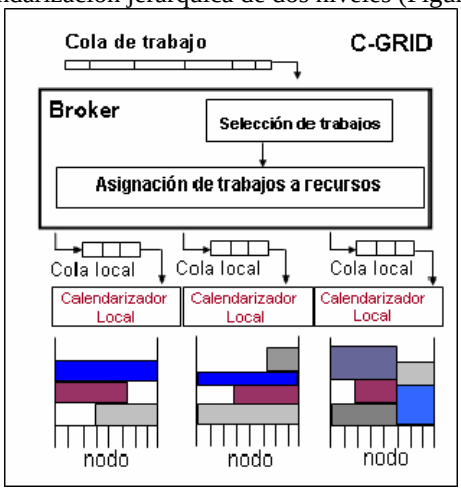

Figura. 2. Configuración del modelo C-GRID

4.2. Carga de trabajo

Las cargas de trabajo conocidas están basadas en logs recolectados de sistemas paralelos específicos de gran escala en uso y contienen toda la información relevante necesaria para la simulación. Pero tales cargas no representan usuarios del C-GRID, y pueden afectar la exactitud de la simulación y la evaluación de las estrategias de calendarización para el ambiente del C-GRID. Las cargas del C-GRID deben contener una mezcla de dos o más cargas, donde cada una refleja el uso normal de la computadora correspondiente como una parte del C-GRID. Usando la carga de trabajo tal como fue almacenada, el calendarizador corre el riesgo de optimizar el sistema para diferentes tiempos locales y zonas horarias. Esto motiva la necesidad de identificar el tiempo de envío, el tiempo local y la zona horaria de cada log para generar mezclas de carga de trabajo adecuadas para la evaluación del C-GRID.

En este experimento se utiliza una mezcla de cargas de trabajo diferente para cada uno de los C-GRID modelados, dicha mezcla es elaborada en base a los logs reales[38] de cada una de las máquinas consideradas en el C-GRID.

4.3. Configuración del C-GRID

Para la realización de nuestros experimentos utilizamos dos C-GRID con diferentes configuraciones, las cuales se muestran en la tabla 1 y la tabla 2

.

Nodo	Origen	Procs.
1	SDSC IBM SP2	128
2	NASA Ames RC Intel iPSC/860	128
3	OSC Linux Cluster	178
4	SDSC Intel Paragon	352
5	CTC IBM SP2	430
6	LANL CM-5	1024
7	SDSC IBM SP3 Blue Horizon	1152

Tabla 1. Configuración del C-GRID 1

Nodo	Origen	Procs.
1	DAS2 - Delft University of technology (fs3) Pentium-III Linux cluster	64
2	DAS2 - Utrecht University (fs4) Pentium-III Linux cluster	64
3	LPC(Laboratoire de Physique Corpusculaire) Part of the LCG (Large hadron collider Computing Grid project) 3GHz Pentium-IV Xeon Linux Cluster	140
4	DAS2 - Vrije University Amsterdam (fs0) Pentium-III Linux cluster	144
5	SDSC DataStar, IBM p655/p690 Partition: 3 batch partition (7 p690 nodes (32 proc, 128GB))	224
6	SDSC DataStar IBM p655/p690 Partition: 2 batch partition (171 p655 nodes (8 proc, 16GB))	1368

Tabla 2. Configuración del C-GRID 2

4.4. Experimentos

El tiempo de ejecución de un trabajo no es conocido sino hasta que finaliza su ejecución, y para tomar decisiones en base al tiempo de ejecución se utiliza un tiempo estimado, generalmente, por el usuario. En la mayoría de los casos, dicha estimación es muy superior al tiempo de ejecución real, dado que se busca proteger el trabajo de una eventual terminación forzada por parte del sistema por haber excedido su límite de tiempo.

En estos experimentos utilizamos el tiempo de ejecución dado en los logs reales y realizamos una modificación aleatoria uniforme dentro de un rango de 50% a 110% del tiempo original, a esto lo hemos llamado *fluctuación de tiempo* (TF). Esto se logra, estableciendo dos parámetros, un porcentaje del tiempo estimado, que se tomará como valor central, y un porcentaje que establece el rango de fluctuación en base al valor central. En la figura 3 se muestra el rango de fluctuación logrado con un promedio de fluctuación de 80% y ±30% de variación con respecto a dicho valor central.

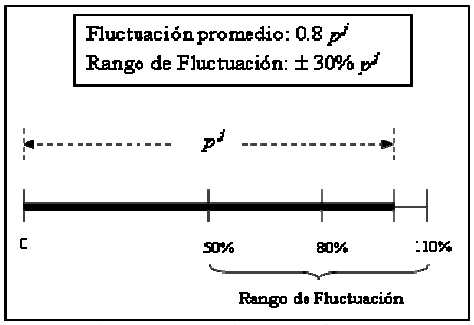

Figura. 3. Rango de fluctuación de tiempo de ejecución

El objetivo de estos experimentos es determinar mediante simulación, las estrategias de selección de recursos que generan mejores resultados independientemente de la fluctuación del tiempo estimado. Todos los experimentos utilizan algunas de las estrategias mencionadas anteriormente. La nomenclatura utilizada para nombrar las estrategias es: (A+B-C) la cual consiste de la estrategia de asignación (A) más una de las estrategia de calendarización local (C), y para las estrategias de asignación que requieren un calendario previo se especifica además el algoritmo de calendarización utilizado (B).

El algoritmo para selección de trabajos de la cola utilizado fue FIFO en todos los experimentos. Las estrategias utilizadas fueron:

- Min_LB-(FCFS, LSF, BEFF)
- Min_PL-(FCFS, LSF, BEFF)
- Min_Lp-(FCFS, LSF, BEFF)
- Min_CT+(FCFS, BEFF) – (FCFS, BEFF)
- Min_WT+(FCFS, BEFF) – (FCFS, BEFF)
- Min_U+(FCFS, BEFF) – (FCFS, BEFF)
- Min_ST+(FCFS, BEFF) – (FCFS, BEFF)

Los experimentos fueron realizados con los siguientes parámetros: Modo de calendarización: off-line; Número de trabajos por experimento: Grid 1: 70000, Grid 2: 60000; Número de experimentos por simulación: 1; Coeficiente de fluctuación promedio: 0.8; Porcentaje del rango de fluctuación: 30%; Tipo de distribución: uniforme.

En la simulación se utilizaron todas las combinaciones para cada C-GRID, con el parámetro TF y sin él, para cada combinación. Los resultados de la simulación fueron divididos en los criterios de evaluación descritos en la sección III, para posteriormente comparar el rendimiento de cada combinación tomando en cuenta la fluctuación del tiempo.

Para el criterio del sistema en el Grid 1 los experimentos arrojaron los siguientes resultados:

La estrategia que mejor resultado generó fue Min_CT+FCFS-FCFS, puesto que permaneció casi sin variación en el aspecto de fluctuación de tiempo y con los valores más altos para las dos métricas (entre el 99 y el 100%). Sin embargo, las estrategias Min_CT+FCFS-BEFF y Min_CT+BEFF-BEFF siguieron muy de cerca de la primera por lo que podemos considerar a las tres estrategias como las mejores de este criterio.

Otras estrategias que vale la pena mencionar por su comportamiento casi constante y que no quedaron muy alejadas de las mejores son: Min_ST+FCFS-BEFF, Min_LB-BEFF,Min_WT+FCFS-BEFF, Min_ST+ BEFF-FCFS. Un poco más abajo, pero de comportamiento constante están las estrategias: Min_ST-FCFS-LSF, Min_CT+FCFS-LSF, Min_LB-LSF y Min_CT+BEFF-FCFS. Es importante señalar que las peores estrategias para este Grid y este criterio son: Min_U+FCFS-BEFF, Min_U+FCFS-FCFS y Min_U+FCFS_LSF (Figura 4).

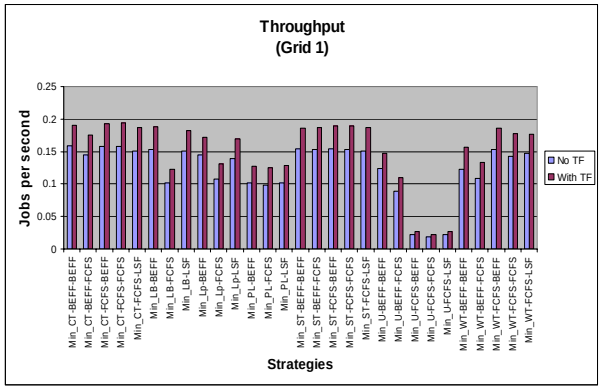

Figura. 4. Gráfica comparativa para la métrica throughput, Grid 1

Con respecto al criterio del usuario también en el Grid 1 se aprecia lo siguiente:

La estrategia con mejor rendimiento en las tres métricas utilizadas, con y sin fluctuación de tiempo, es Min_Lp-BEFF, seguida de cerca por Min_WT+FCFS-BEFF y Min_U+BEFF-BEFF. Las estrategias Min_WT+BEFF-BEFF vienen en tercer lugar, con una variación un poco más grande en la métrica response ratio (116% sin fluctuación de tiempo). Las demás métricas en promedio están por arriba del 5% de

diferencia con respecto a las mejores, sin embargo, es de hacer notar que hay muy poca variación en sus resultados con y sin fluctuación de tiempo. Las peores estrategias en este criterio son Min_U+FCFS-BEFF, Min_U+FCFS-FCFS y Min_U+FCFS_LSF (Figura 5 y 6).

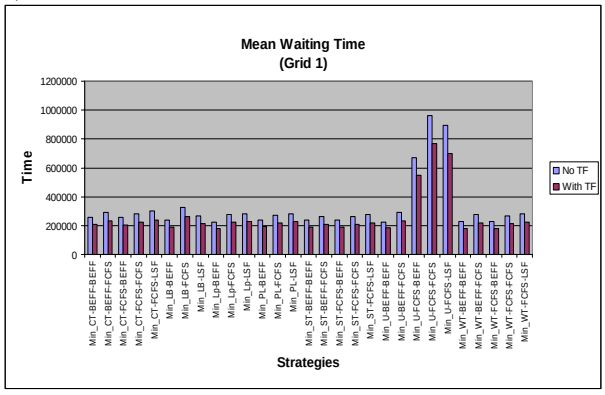

Figura. 5. Gráfica comparativa para la métrica Waiting Time, Grid 1

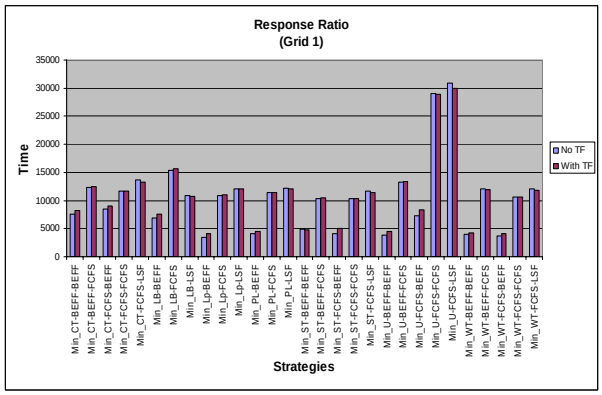

Figura. 6. Gráfica comparativa para la métrica Response Ratio, Grid 1

En el criterio del algoritmo la estrategia Min_CT+FCFS-BEFF generó los mejores resultados, seguida muy de cerca por las estrategias Min_CT+BEFF-BEFF y Min_CT+FCFS-FCFS. Se puede considerar como siguiente mejor estrategia a Min_ST+FCFS-BEFF, que en ambos casos (con fluctuación de tiempo y sin él) apareció en cuarto lugar, pero con una variación muy pequeña con respecto a las mejores (3% y 2%). Las estrategias Min_ST+FCFS-FCFS, Min_LB-BEFF, Min_WT+FCFS-BEFF, Min_ST+BEFF-BEFF y Min_ST+BEFF-FCFS mostraron resultados con una variación entre el 103% y 104% con respecto a las mejores. Las peores estrategias son las mismas que para los criterios anteriores. (Figura 7)

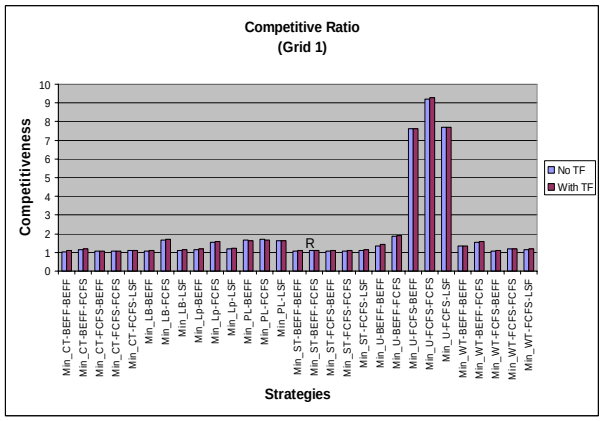

Figura. 7. Gráfica comparativa para la métrica Competitive Ratio, Grid 1

En lo que respecta al Grid 2 en el criterio del sistema, la estrategia Min_WT+FCFS-BEFF proporcionó el mejor rendimiento, siendo la mejor de todas con fluctuación del tiempo de ejecución, y la segunda (con 99%) cuando no existió fluctuación del tiempo.

La segunda mejor estrategia fue Min_ST+FCFS-BEFF, aunque podemos observar también que la estrategia Min_WT+BEFF-BEFF sin fluctuación del tiempo generó el mejor rendimiento en utilization y throughput, sin embargo al aplicar la fluctuación del tiempo, su rendimiento bajó al 93%. Min_CT+FCFS-BEFF mantuvo un rendimiento aceptable con un 96% sin fluctuación de tiempo para ambas métricas, y con fluctuación de tiempo se incrementó a 99%.

Otras dos estrategias que demostraron tener un rendimiento aceptable y poca variación al considerar la fluctuación del tiempo son Min_CT+BEFF-BEFF y Min_CT+FCFS-FCFS (Figura 8).

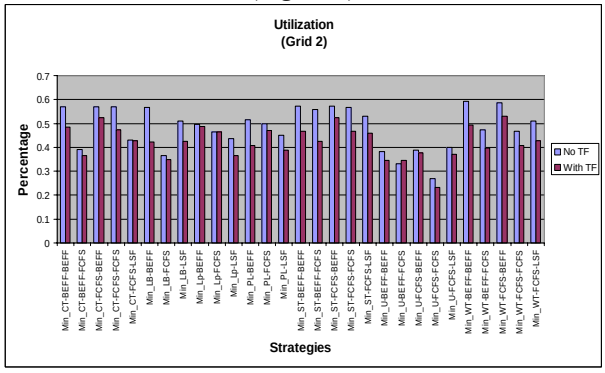

Figura. 8. Gráfica comparativa para la métrica Utilization, Grid 2

Al analizar el criterio del usuario la estrategia Min_U+BEFF-BEFF es la que mejor comportamiento presentó, ya que en la métrica Response Ratio generó el mejor resultado tanto con fluctuación de tiempo como sin él. En las métricas de waiting time y turnaround,

aunque sus resultados no son los mejores, su variación no es relevante.

En segundo lugar podemos considerar la estrategia Min_PL-BEFF puesto que observó un comportamiento aceptable, aún cuando no fue la mejor en ninguna métrica, su variación con respecto a los resultados de la mejor métrica no son relevantes.

Podemos mencionar que la estrategia Min_WT+FCFS-BEFF generó los mejores resultados en lo que respecta a las métricas waiting time y turnaround, y aunque en la métrica response ratio tuvo algunas variaciones, podemos considerar esta estrategia también como una de las mejores. Otras estrategias relevantes por su comportamiento casi constante son Min_WT+BEFF-BEFF, Min_LB-BEFF, Min_Lp-BEFF, Min_ST+BEFF-BEFF y Min_ST+FCFS-BEFF. (Figura 9 y 10).

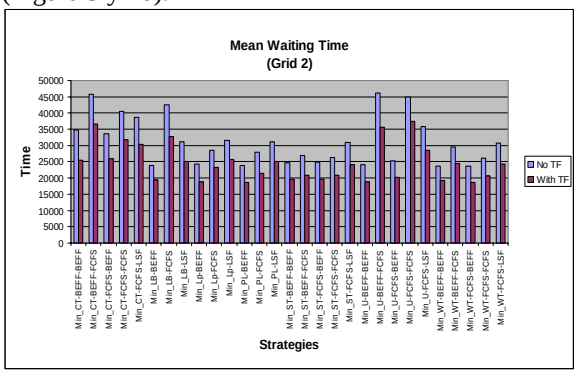

Figura. 9. Gráfica comparativa para la métrica Mean Waiting Time, Grid 2

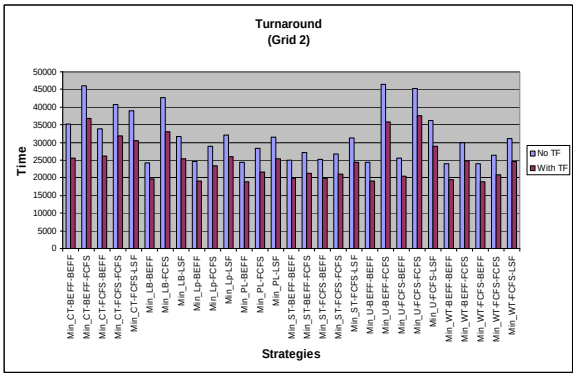

Figura. 10. Gráfica comparativa para la métrica Turnaround, Grid 2

En lo que respecta al criterio del algoritmo podemos ver lo siguiente:

Para la métrica de competitividad, la única estrategia que mostró un comportamiento invariable ante una fluctuación del tiempo y sin él fue Min_CT+FCFS-BEFF, que si bien no fue la mejor en ninguno de los dos casos, su rendimiento está muy cercano al mejor (104% en ambos casos).

Otras estrategias que mostraron poca variación son: Min_WT+FCFS-BEFF, Min_WT+BEFF-BEFF, Min_ST+BEFF-BEFF, Min_ST+FCFS-BEFF y Min_ST+FCFS-FCFS.

También es de hacer notar que la estrategia Min_ST+BEFF-FCFS mantuvo un comportamiento casi sin variación aunque un poco alejada del mejor resultado (106% y 107%).(Figura 11).

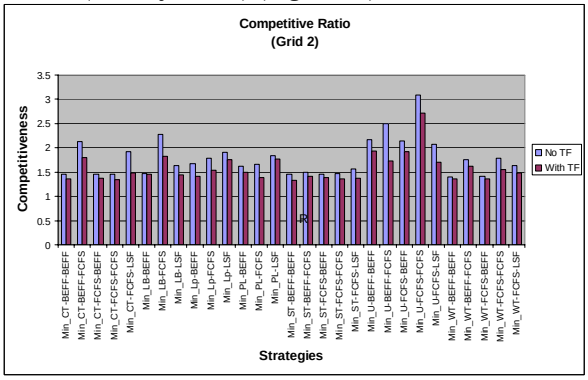

Figura. 11. Gráfica comparativa para la métrica Competitive Ratio, Grid 2

5. Conclusiones y trabajo futuro

En este trabajo analizamos el comportamiento que tienen algunas estrategias de calendarización en un GRID computacional jerárquico, tomando en cuenta la fluctuación del tiempo de ejecución estimado de tareas paralelas en dos configuraciones diferentes de C-GRID.

Para el criterio de Sistema, las estrategias que demostraron el mejor desempeño fueron: Min_ST+FCFS-BEFF, Min_CT+FCFS-BEFF y Min_WT+FCFS-BEFF. Cabe hacer notar que el algoritmo local en los tres casos es Backfilling y que las estrategias de asignación de trabajos utilizan el algoritmo FCFS para elaborar el calendario previo.

Considerando el criterio de usuario, las estrategias Min_U+BEFF-BEFF, Min_WT+FCFS-BEFF y Min_Lp-BEFF demostraron tener un buen desempeño independientemente de la configuración del C-GRID y de la fluctuación del tiempo.

En lo que respecta al criterio del algoritmo, observamos que la mejor estrategia fue Min_CT+FCFS-BEFF seguida por Min_WT+FCFS-BEFF.

Podemos observar que la estrategia Min_WT+FCFS-BEFF es común a los tres criterios y se puede considerar como la mejor opción para la calendarización en un C-GRID jerárquico, asimismo podemos observar también que el algoritmo de calendarización local que prevaleció en las mejores

estrategias fue BEFF, confirmando su ya conocida eficiencia.

Como trabajo futuro podemos plantear la necesidad de ampliar las estrategias de calendarización, tanto locales como para asignación de trabajos a recursos. Es importante también estudiar el costo (en tiempo de ejecución) de las estrategias aquí utilizadas.

REFERENCIAS

[1] http://www.gridcomputing.com/gridfaq.html

[2] http://es.wikipedia.org/wiki/Grid_computing

[3] Project Grid '5000. www.irisa.fr/paris/web/65k.html

[4] Compute Against Cancer. http://www.computeagainstcancer.org

[5] United Devices Inc. http://www.ud.com

[6] Folding@Home. http://www.stanford.edu/group/pandegroup/Cosm/

[7] Genome@Home. http://genomeathome.stanford.edu/

[8] Evolution@Home. http://www.evolutionary-research.org

[9] Golem@Home. Genetically Organized Lifelike Electro Mechanics). http://www.demo.cs.brandeis.edu/golem/

[10] Mars Clickworkers. http://clickworkers.arc.nasa.gov/

[11] Climate Prediction. http://www.climateprediction.net

[12] Gamma Flux. http://www.jansson.net/

[13] TERAGRID. http://www.teragrid.org/

[14] National Middleware Initiative. http://www.nsf-middleware.org

[15] Earth System Grid. http://www.earthsystemgrid.org

[16] NEESGrid. http://it.neesgrid.org

[17] Biomedical Informatics Research Network http://www.nbirn.net

[18] ApGrid. http://www.apgrid.org

[19] TW Grid. http://www.twgrid.org

[20] European DataGrid Project. http://www.edg.org

[21] Globus Toolkit Version 4: Software for Service-Oriented Systems. I. Foster. *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2005.

[22] Globus Toolkit. http://www.globus.org/

[23] I. Foster, C. Kesselman, "The Grid In A Nutshell", in *Grid Resource Management State of the Art and Future Trends*, Edited by Jarek Nabrzyski 1 edition September 2003.

[24] Condor High Throughput Computing. http://www.cs.wisc.edu/condor/

[25] Platform Computing. http://www.platform.com

[26] Volker Hamscher, Uwe Schwiegelshohn, Achim Streit, and Ramin Yahyapour. "Evaluation of Job-Scheduling Strategies for Grid Computing". In Proceedings of the 7th International Conference on High Performance Computing HiPC-2000, Vol. 1971:191—202 of Lecture Notes in Computer Science, Springer, Berlin, *Lecture Notes in Computer Science* LNCS 1971, Bangalore Indien 2000.

[27] H. Smith. http://users.cs.cf.ac.uk/C.L.Mumford/heidi/Background.html

[28] A. Tchernykh, J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk. "Two Level Job-Scheduling Strategies for a Computational Grid". In Parallel Processing and Applied Mathematics, Wyrzykowski et al. (Eds.): The Second Grid Resource Management Workshop (GRMW'2005) in conjunction with the Sixth International Conference on Parallel Processing and Applied Mathematics - PPAM 2005. Poznan, Poland, September 11-14, 2005, *Lectures Notes in Computer Science vol. 3911*, Springer-Verlag, 2006.

[29] D. A. Lifka, "An Extensible Job Scheduling System For Massively Parallel Processor Architectures", Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the Illinois Institute of Technology Chicago, Illinois, May, 1998

[30] Aida K, Takefusa A, Nakada H, Matsuoka S, Sekiguchi S, Nagashima U. "Performance evaluation model for scheduling in a global computing system". *The International Journal of High Performance Computing Applications* 2000; **14**(3).

[31] Song H, Liu X, Jakobsen D, Bhagwan R, Zhang X, Taura K, Chien A. "The MicroGrid: A scientific tool for modelling computational Grids". *Proceedings of IEEE Supercomputing (SC 2000)*, Dallas, TX, 4–10 November 2000.

[32] Casanova H. "Simgrid: A toolkit for the simulation of application scheduling". *Proceedings 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia, 15–18 May. IEEE Computer Society Press, 2001.

[33] Rajkumar Buyya and Manzur Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", *The Journal of Concurrency and Computation: Practice and Experience* (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[34] I. Foster and C. Kesselman. "Globus: A metacomputing infrastructure toolkit". *International Journal of Supercomputer Applications*, 11(2):115–129, 1997.

[35] Elisa Salazar, Manuel Valenzuela, Daniel Mendoza, Yair Castro, Juan Manuel Ramírez, Edelmira Rodríguez, Andrei Tchernykh "Evaluación de las estrategias de calendarización para sistemas distribuidos de tipo grid computacional". XVI Escuela Nacional de Optimización y Análisis Numérico (Enoan 2006). Facultad de Ciencias Básicas de la Universidad Autónoma de Tlaxcala. Marzo 26-31 de 2006.

[36] Yair Castro, Daniel Mendoza, Juan Manuel Ramírez, Edelmira Rodríguez, Elisa Salazar, Andrei Tchernykh, Manuel Valenzuela. "Evaluación de las Estrategias de Calendarización en Sistemas Distribuidos de Tipo Grid Computacional". CICC 2005- 6° Congreso Internacional de las Ciencias Computacionales, 28 al 30 de Sep de 2005

[37] Yair Castro, Daniel Mendoza, Juan Manuel Ramírez, Edelmira Rodríguez, Elisa Salazar, Andrei Tchernykh, Manuel Valenzuela. "Evaluación de las Estrategias de Calendarización en Sistemas Distribuidos de Tipo Grid Computacional". Simposio Supercomputo Noroeste 2006. Universidad de Sonora , FEBRERO 21-23, 2006

[38] Parallel Workloads Archive. URL http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

[39] D. G. Feitelson, "Packing Schemes for Gang Scheduling". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson, L. Rudolph (Eds.), Springer-Verlag 1996, Lect. Notes Computer Science vol. 1162, pp. 89-110. http://www.cs.huji.ac.il/labs/parallel/workload/models.html

[40] U. Lublin, D. G. Feitelson, "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs", Technical Report 2001-12, School of Computer Science and Engineering, The Hebrew University of Jerusalem, Oct 2001.

[41] S. Zhuk, A. Chernykh, A. Avetisyan, S. Gaissaryan, D. Grushin, N. Kuzjurin, A. Pospelov, and A. Shokurov, "Comparison of Scheduling Heuristics for Grid Resource Broker," presented at PCS2004 Third International Conference on Parallel Computing Systems, Colima, Colima, México, 2004.