

Análisis Experimental de un Algoritmo de Calendarización en Línea para Grid con Modelo Descentralizado

Ariel Quezada-Pina, Andrei Tchernykh

*Centro de Investigación Científica y de Educación Superior de Ensenada
{aquezada,chernykh}@cicese.mx*

Resumen

Se discute un algoritmo de calendarización no clarividente, sin ejecución discontinua de tareas paralelas en línea para un Grid computacional. En nuestro modelo, el Grid está constituido por un conjunto de máquinas con diferente número de procesadores idénticos. El balanceo de carga se lleva a cabo utilizando migración de tareas entre las distintas máquinas. El objetivo es mejorar el tiempo de terminación del calendario. Se muestran resultados experimentales comparando el desempeño del algoritmo contra algoritmos aplicados a un modelo jerárquico de dos niveles.

1. Introducción

En un Grid se pueden compartir servicios de red, capacidad de almacenamiento digital, sensores, licencia de software y poder de cómputo a través de dominios que están geográficamente distribuidos [4]. Una capa de los protocolos en el Grid es la calendarización de las tareas en recursos disponibles [16]. Ésta consiste en asignar un conjunto de tareas a un conjunto de máquinas con el objetivo de optimizar algún parámetro de desempeño, o conjunto de ellos.

Existen diferentes modelos de calendarización para Grid [8]. En el modelo jerárquico las máquinas que ejecutan las tareas se encuentran en el nivel inferior de una jerarquía. El nivel superior es un calendarizador (*broker*) de Grid. En este modelo la comunicación es descendente, desde el broker hacia las máquinas. No existe comunicación entre las máquinas o calendarizadores que forman un mismo nivel. Por esta razón, el broker tiene una visión general de las solicitudes de ejecución de tareas, mientras que los detalles específicos acerca del

estado de los recursos permanecen ocultos del mismo. La administración de un recurso específico es trabajo del sistema de administración local, que conoce a detalle la información del estado de la máquina y los trabajos que recibió para su ejecución. Este tipo de modelo tiene una escalabilidad limitada y cierto grado de tolerancia a fallas [8,7].

El broker asigna cada una de las tareas de una cola global a las máquinas en el primer nivel, base de algún criterio (Figura 1a), por ejemplo, minimizar el tiempo de terminación del calendario. Cada máquina utiliza un calendarizador local para ejecutar las tareas que le fueron asignadas. En [18,19] se presentan 4 niveles de información disponible en los cuales se pueden basar distintas estrategias de asignación de los trabajos. Cada nivel tiene la información de los niveles anteriores más información adicional. El nivel 1 dispone solo del número de tareas a ejecutar en cada máquina. El nivel 2 conoce el tamaño de las tareas. El nivel 3 cuenta con el tiempo de ejecución de las tareas. Si esto no es posible, los usuarios proporcionan una estimación del mismo al someter las tareas. El nivel 4 tiene acceso a la información de los calendarios locales de cada máquina.

Denominan MPS_Alloc a la estrategia de asignación y PS al calendarizador local. Entonces, la estrategia de calendarización de dos niveles, MPS, está definida como $MPS = MPS_Alloc + PS$.

El modelo descentralizado (Figura 1b) puede verse como un modelo de calendarización de un solo nivel. Este modelo considera las máquinas con comunicación punto a punto [8]. El modelo descentralizado carece de un calendarizador de Grid que lleve a cabo asignación de tareas con un objetivo global de Grid, como lo hace el modelo jerárquico. Sin embargo, se han propuesto diferentes estrategias para lograr el balanceo de cargas en las máquinas, como en [6, 17].

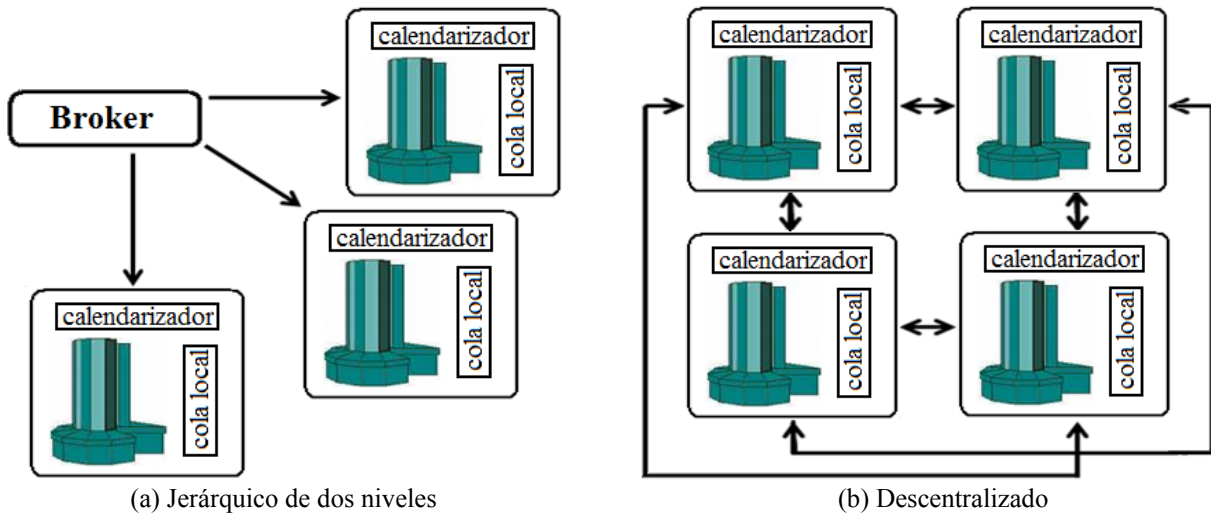


Figura 1. Modelos de Grid

El modelo descentralizado aborda de forma natural diferentes características de un Grid, como son: tolerancia a fallas, escalabilidad, política de calendarización múltiple y autonomía de cada sitio. Sin embargo, este modelo presenta algunos problemas propios de un modelo descentralizado; coasignación y rastreo de ejecución de tareas, por ejemplo [8].

2. Trabajo previo

Se han propuesto diferentes sistemas de calendarización que se han implementado en distintos tipos de Grid. Sin embargo, todavía existen muchos problemas abiertos en esta área [2].

Desde el punto de vista del sistema, un objetivo típico de un calendarizador de Grid es lograr balanceo de la carga entre las máquinas. En teoría de calendarización, este objetivo es conocido comúnmente como minimización del tiempo de terminación del calendario del sistema. Este objetivo es utilizado principalmente como criterio para el problema de calendarización estático y presenta algunos problemas, especialmente en el escenario en línea con trabajos independientes, ver [17]. A pesar de esto, es simple de utilizar y comúnmente empleado para estudios teóricos, por ejemplo [1]. Por esta razón, se utiliza la minimización de tiempo de finalización de calendario como objetivo de este trabajo.

El mejor algoritmo conocido *PS* para el problema de calendarización en línea no clarividente tiene un factor de competitividad de $2-1/m$, donde m denota el número de procesadores en el sistema paralelo [10]. Así, la cota mínima para el factor de competitividad

para cualquier algoritmo en línea de dos niveles *MPS* es al menos $2-1/m$.

En [11] un problema relacionado fue abordado. Los autores modelan un sistema que consiste de m clusters, todos con $size$ procesadores idénticos y proponen un algoritmo que garantiza un factor de competitividad de 4. También muestran que se puede obtener una cota de 3 para el caso específico de que el último trabajo terminado requiera a lo más $m/2$ procesadores.

Algunos estudios se han realizado abordando puntos específicos del proceso de calendarización en un Grid. Por ejemplo en [13] se estudia de forma experimental cómo la estimación de tiempo de ejecución de las tareas proporcionado por el usuario afectan las estrategias que utilizan esta información para realizar la asignación y calendarización de tareas. Por otra parte, en [7], proponen un modelo de fallas para el estudio de estrategias de calendarización en Grid de forma experimental. Aún más, existen estudios dedicados exclusivamente al análisis de la carga utilizada para someter a prueba estrategias de calendarización, como [14, 3]. O bien, existen estudios que evalúan solo de forma experimental el rendimiento de diferentes estrategias de calendarización, como [9, 15]

También se han estudiado estrategias las cuales no requieren específicamente que el modelo de calendarización sea centralizado o distribuido. Uno de ellos utiliza un concepto de admisibilidad y consiste en que cada tarea puede calendarizarse solo en un rango determinado de las máquinas disponibles en el Grid. Este rango está basado en el tamaño de las máquinas donde puede ser ejecutada la tarea a ser calendarizada. Este sencillo esquema se propuso en [19] y en [12] presentan un estudio teórico de este esquema de admisibilidad aplicado a estrategias de

calendarización de Grid usando un modelo jerárquico de dos niveles. En [5] complementan este trabajo con un estudio experimental utilizando las estrategias propuestas en [12].

3. Definición del problema

El Grid consiste de m máquinas. La máquina i tiene s_i procesadores. Sin pérdida de generalidad, se ordenan las máquinas por número de procesadores de forma no decreciente, teniendo $s_1 \leq s_2 \leq \dots \leq s_m$ [18].

Cada tarea j es caracterizada por la tripleta $(r_j, size_j, p_j)$, con fecha de entrega $r_j \geq 0$, tamaño $1 \leq size_j \leq s_m$ y tiempo de procesamiento p_j . La notación $i=a(j)$ es utilizada para denotar que la tarea j será ejecutada en la máquina i . El tiempo de terminación del trabajo j en el calendario S está definido por $C_j(S)$. Por simplicidad se utilizará C_j . Un calendario es posible si $r_j + p_j \leq C_j$ se mantiene para todas las tareas y si para todo tiempo t ,

$$s_i \geq \sum_{j|C_j - p_j \leq t < C_j \wedge i=a(j)} size_j \quad (1)$$

cada tarea debe de ser ejecutada en $size_j$ procesadores, en una sola máquina sin interrupción. No se permite ejecución multisitio o ejecución discontinua (*preemption*).

En este trabajo se aborda el problema $GPM/r_j, size_j, p_j/C_{max}$. El Grid se considera como un conjunto de máquinas paralelas con procesadores idénticos. La calendarización de tareas se realiza en línea. El tiempo de ejecución de cada tarea no se conoce hasta haberla concluido (no clarividente, *non clairvoyant*). El objetivo es minimizar el tiempo de terminación del calendario.

Una forma de medir el desempeño de estrategias de calendarización es la métrica factor de competitividad, la cual está definida como:

$$\rho = \frac{C_{max}}{C_{opt}}, \quad (2)$$

donde C_{max} y C_{opt} representan el tiempo de terminación del calendario de una estrategia y de un calendario óptimo, para una misma instancia del problema, respectivamente.

4. Algoritmo de robo de tareas (*stealing*)

El algoritmo de robo de tareas (*ST*) fué propuesto en [17]. El algoritmo considera la existencia de dos colas de tareas por máquina, A y B . Las tareas están exclusivamente en una sola cola A o B , las cuales están definidas como sigue:

$$A_i = \{j | \max\{\frac{s_i}{2}, s_{i-1}\} < size_j \leq s_i\}$$

(3)

$$B_i = \{j | s_{i-1} < size_j \leq \frac{s_i}{2}\} \quad (4)$$

La cola A_i contiene a las tareas que ocupan más del 50% de los procesadores disponibles en la máquina i . Y B_i , las tareas que ocupan menos del 50% de los procesadores de la misma máquina. Las tareas en A_i y B_i solo se pueden calendarizar en la máquina i o más grandes. Este valor frontera, de 50% de procesadores, puede variarse. En el presente trabajo se utiliza la nomenclatura ST50, para referirse al algoritmo ST original, con valor frontera de 50%.

Las tareas se calendarizan en línea de la siguiente forma: En la máquina i se ejecutan las tareas contenidas de la cola A_i . Una vez terminadas las tareas de esta cola, la máquina puede “robar” tareas de máquinas más pequeñas cuando tiene procesadores disponibles y pueden ser ejecutadas inmediatamente. Si no hay tareas en cola A_i y otras maquinas más pequeñas, se ejecutan las tareas de la categoría B_i .

En [17], donde puede consultarse una descripción completa del algoritmo ST, probaron que no existe un algoritmo de tiempo polinomial que siempre produzca un calendario con $\rho < 2$ para el problema $GPM/size_j/C_{max}$ y cualquier dato de entrada, a menos que $P=NP$.

El algoritmo ST garantiza que cada máquina tendrá siempre ocupados al menos el 50% de sus procesadores antes de que inicie la ejecución de la última tarea. Este algoritmo tiene un factor de competitividad $\rho < 3$ para todo tipo de datos de entrada y toda configuración de Grid en el caso de sometimiento concurrente de tareas. Y para el problema $GPM/r_j, size_j, p_j/C_{max}$ garantiza un factor de competitividad de $\rho < 5$ en el caso de sometimiento de tareas en línea [17]. Este es el mejor algoritmo conocido a la fecha para este problema.

5. Configuraciones de Grid

Se realizaron experimentos en simulación para comparar el desempeño de estrategias de calendarización con modelo jerárquico de 2 niveles y el algoritmo ST. Se utilizaron dos bitácoras de ejecución de tareas, utilizando bitácoras de diferentes centros de supercómputo, para simular una carga de trabajo de un Grid. Los logs elegidos tienen tareas con tamaños desde un procesador hasta el tamaño del maquina más grande de la configuración de Grid utilizada.

Tabla 1. Características de Grid 1

	Localidad	Procs	Log
1	KTH- Swedish Royal Institute of Technology	100	KTH-SP2-1996-2.swf, 28489 tareas, 204 usuarios
2	SDSC-SP2 – San Diego Supercenter SP2	128	SDSC-SP2-1998-3.1-cln.swf, 73496 tareas, 437 usuarios
3	HPC2N-High Performance Computing Center North, Sweden	240	HPC2N-2002-1.1-cln.swf , 527371 tareas, 256 usuarios
4	CTC-Cornell Theory Center	430	CTC-SP2-1996-2.1-cln.swf , 79302 tareas, 679 usuarios
5	LANL-Los Alamos National Lab	1024	LANL-CM5-1994-3.1-cln.swf, 201387 tareas, 211 usuarios
6	SDSC-BLUE –San Diego Supercenter Blue Gene	1152	SDSC-BLUE-2000-3.1-cln.swf, 250,440 tareas, 468 usuarios
7	SDSC-DS – San Diego Supercenter Data Star	1368	SDSC-DS-2004-1-cln.swf, 96089 tareas, 460 usuarios

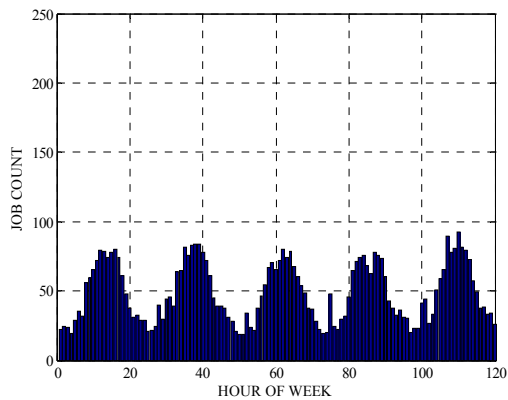
Tabla 2. Características de Grid 2

	Localidad	Procs	Log
1	DAS2 - University of Amsterdam	64	Gwa-t-1-anon_jobs-reduced.swf , 1124772 tareas, 333 usuarios
2	DAS2 - Delft University of Technology	64	
3	DAS2 – Utrecht University	64	
4	DAS2 - Leiden University	64	
5	KTH - Swedish Royal Institute of Technology	100	KTH-SP2-1996-2.swf, 28489 tareas, 204 usuarios
6	DAS2- Vrije University Amsterdam	144	Gwa-t-1-anon_jobs-reduced.swf (cont.)
7	HPC2N - High Performance Computing Center North, Sweden	240	HPC2N-2002-1.1-cln.swf , 527371 tareas, 256 usuarios
8	CTC - Cornell Theory Center	430	CTC-SP2-1996-2.1-cln.swf , 79302 tareas, 679 usuarios
9	LANL -.Los Alamos National Lab	1024	LANL-CM5-1994-3.1-cln.swf, 201387 tareas, 211 usuarios

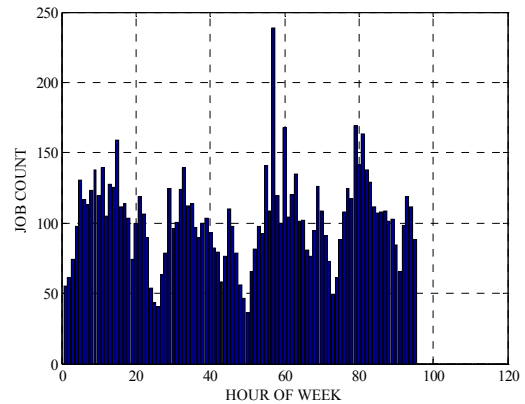
El número de tareas utilizadas representa el número de tareas que se tomaron de dicho log en la unión de logs. Estos logs y su estadística relevante están disponibles en [3].

En Tabla 1 y Tabla 2 se muestran las características de las dos configuraciones de Grid

utilizadas. De Figura 2 a Figura 6 se muestran estadísticas de las bitácoras correspondientes a las configuraciones de Grid utilizadas para los experimentos.

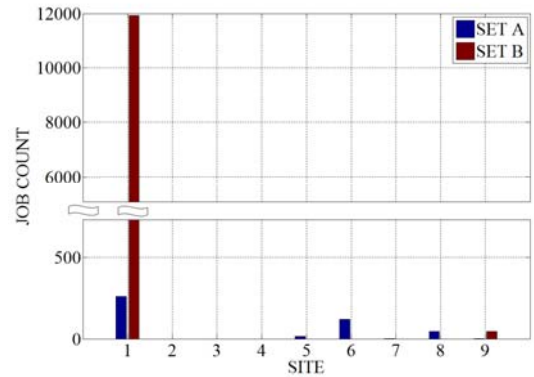
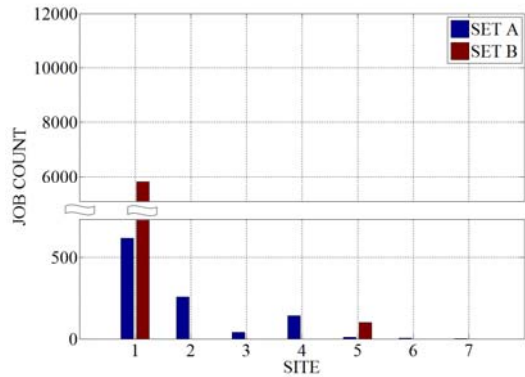


(a) Grid 1, 5 días

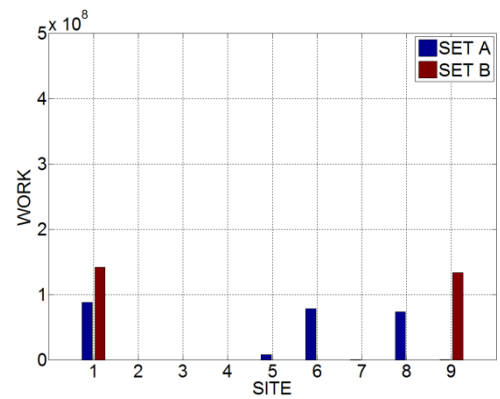
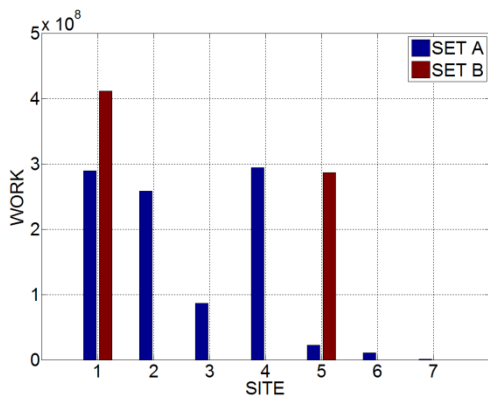


(b) Grid 2, 4 días

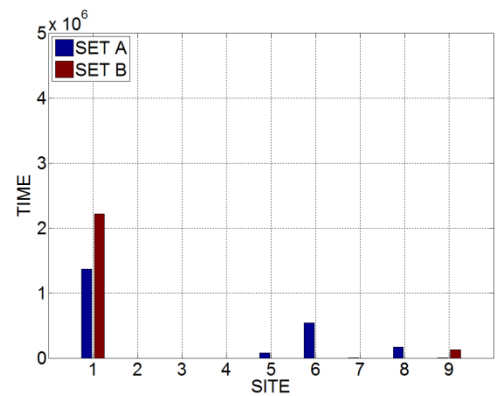
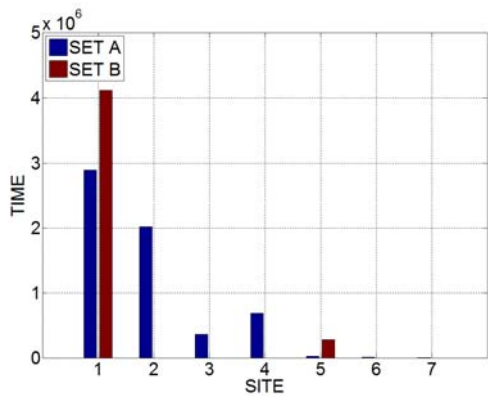
Figura 2. Número promedio de tareas sometidas por hora para Grid 2



(a) Grid 1 (b) Grid 2
 Figura 3. Número promedio de tareas por colas A y B para ST50 en Grid 2



(a) Grid 1 (b) Grid 2
 Figura 4. Consumo promedio de recursos por colas A y B para ST50 en Grid 2

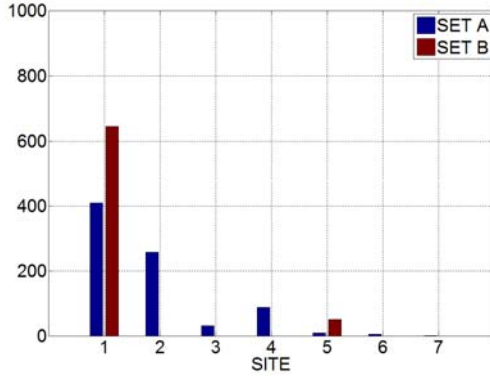


(a) Grid 1 (b) Grid 2
 Figura 5. Consumo promedio de recursos por procesador por colas A y B para ST50 en Grid 2

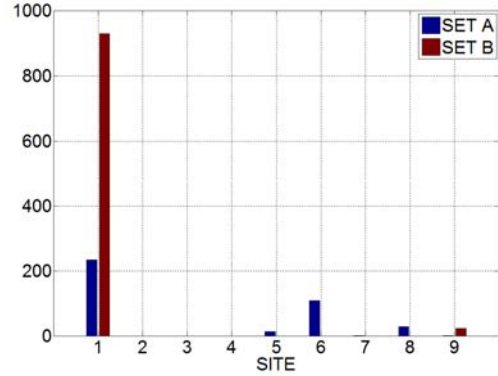
6. Resultados experimentales

Se realizaron 30 experimentos con cada configuración de Grid presentada. Los períodos de

prueba fueron de 4 días (Grid2) y 5 días (Grid1). Los parámetros para evaluar los resultados utilizados fueron: ρ and C_{max} .



(a) Grid 1



(b) Grid 2

Figura 6. Carga paralela por procesador por colas A y B para ST50 en Grid 2

Para calcular el factor de competitividad se utilizó una cota inferior en lugar del tiempo de finalización de calendario óptimo, mostrada en ecuación 5.

$$C_{\max}^* \geq \hat{C}_{\max}^* = \max \left\{ \max_j (r_j + p_j), \max_{1 \leq i \leq m} \frac{\sum_{j: size_j > m_{i-1}} size_j \cdot p_j}{\sum_{v=1}^m m_v} \right\} \quad (5)$$

6.1 Variantes de Algoritmo ST

Se realizó un análisis experimental de variaciones del algoritmo ST. Se utilizaron dos órdenes de búsqueda de tareas en las colas A y B: $A_b, A_{i-1}, A_{i-2}, \dots, B_i$ y $A_b, B_b, A_{i-1}, A_{i-2}, \dots$. Otras dos variantes relacionadas con la selección de tareas de acuerdo a su tamaño fueron probadas: primero en ajustar (*First Fit*) y mejor en ajustar (*Best Fit*). Además, otros dos órdenes de búsqueda de tareas en las colas de tareas fueron comparadas: A_b, A_{i-1}, \dots, A_0 , luego B_0, B_1, \dots, B_b y A_b, A_{i-1}, \dots, A_0 luego B_b, B_{i-1}, \dots, B_0 . Se realizaron experimentos con todas estas variantes sin que los resultados mostraran una mejoría estadística del caso promedio. Por lo tanto, el algoritmo original de ST, respecto al orden en la búsqueda de tareas en las colas y en la selección de tareas, fue utilizado para los experimentos posteriores.

6.2 Variantes de Valor Frontera entre Colas A y B

El algoritmo ST utiliza un valor frontera, entre los conjuntos A y B, de 50%. Sin embargo, este valor frontera puede variarse y lograr diferentes resultados. Se hicieron experimentos con Grid 1 y Grid 2 variando el valor frontera entre los conjuntos A y B, de 0% a 100%. Si el valor frontera es 0% (*ST0*) cada una de las tareas pertenecen a la cola A de alguna de las máquinas. En cambio, si el valor frontera es del

100% (*ST100*) significa que cada una de las tareas pertenecen a la cola B de alguna de las máquinas.

6.3 Balanceo de la Carga

Se realizó una comparación entre calendarización por sitio individual y multisitio. Los resultados de la Tabla 3 corresponden a simulaciones con periodo de 5 días con la configuración Grid 1. Cada columna muestra el tiempo de finalización del calendario ejecutado localmente con el algoritmo *backfilling easy first fit (BEFF)* a cada sitio individual. Estos tiempos son comparados con los tiempos de finalización, en segundos, obtenidos por ST50 y ST100 al utilizarlos en configuración de Grid. C_{\max} corresponde al máximo valor obtenido entre todos los calendarios locales. ST50 y ST100 muestran valores más pequeños que C_{\max} debido a que tareas sometidas a sitios muy cargados han sido movidas a otros sitios para su ejecución.

6.4 Comparación de Algoritmo ST y Estrategias para Modelo de Jerárquico de Dos Niveles

La Figura 9 muestra el factor de competitividad promedio y su desviación estándar de 3 estrategias de calendarización de dos niveles y dos versiones del algoritmo ST: $Min_Lp+BEFF$, $Min_PL+BEFF$, $Min_ST+BEFF$, *ST50* y *ST100*. Esta figura muestra el efecto del esquema de balanceo de cargas en el algoritmo ST en el tiempo de finalización de calendario. Es fácil observar que los algoritmos ST50 y ST100 superan a las otras estrategias en todos los experimentos y muestran una menor dispersión en la desviación estándar.

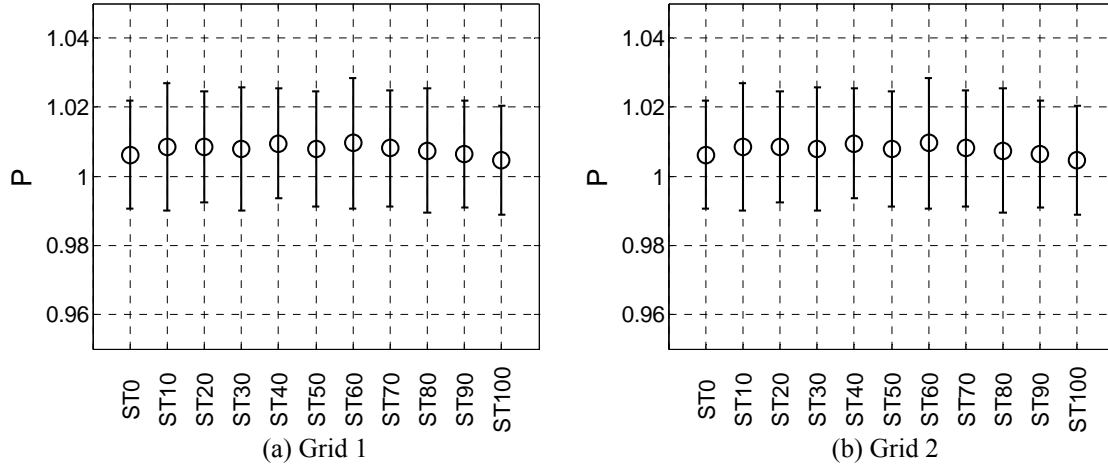


Figura 7. Factor de competitividad promedio y desviación estándar contra variaciones de valor de frontera de colas A y B, en Grid 2

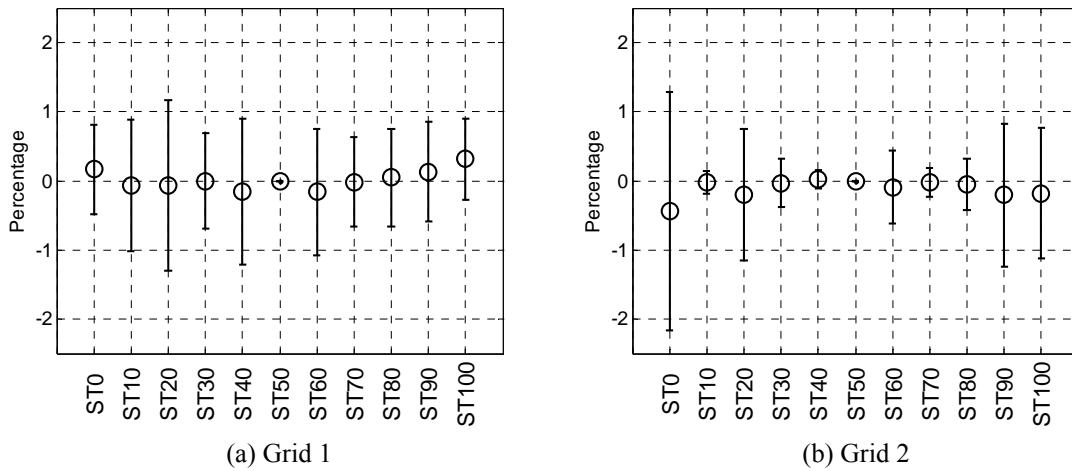


Figura 8. Promedio de diferencias de variaciones de valor frontera de colas A y B con respecto de ST50, en Grid 2

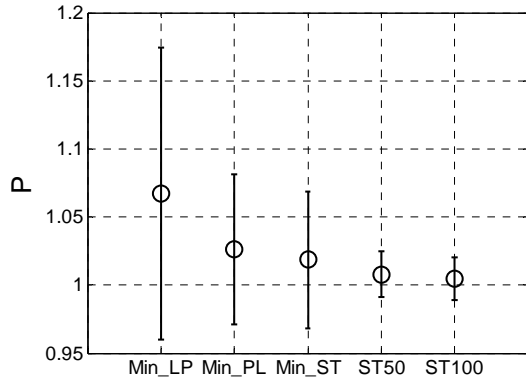
Tabla 3. Tiempo de finalización para ejecución por sitio individual y multisitio

CTC_SP2	LANL_CM_5	KTH-SP2	HPC2N	SDSC_BLUE	SDSC_DS	SDSC_SP2	C_{\max}	ST50	ST100
541441	566248	606940	945042	597634	556420	162926	945042	543117	542547

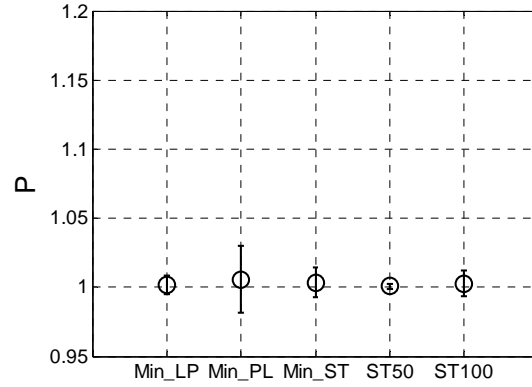
Cuando los sitios son independientes, las tareas pueden ser sometidas a sitios muy cargados debido a la información imprecisa en la etapa de asignación. En cambio, con un esquema descentralizado de un solo nivel, sitios que se encuentran ligeramente cargados son sujetos a cargas de otros sitios y en promedio, el tiempo de finalización de los sitios cargados disminuye.

Las Tablas 4 y 5 muestran el efecto de migración de tareas y recursos consumidos entre los sitios, al realizar el robo de tareas para el balanceo de la carga

entre los sitios del Grid. La Tabla 4 representa una matriz de migración M que muestra la migración de tareas del sitio original de sometimiento al sitio de ejecución. M_{ij} representa el número de trabajos promedio que fueron migrados del sitio i al sitio j . También se muestra el número total de tareas migrado a cada sitio. La Tabla 5 representa una matriz de migración M que muestra la información de la Tabla 4, expresada en porcentaje del total de tareas promedio por sitio. La columna total muestra el porcentaje total de tareas migradas del sitio i .



(a) Grid 1



(b) Grid 2

Figura 9. Comparación de factor de competitividad promedio y su desviación estándar

Tabla 4. Número de tareas promedio migradas desde el sitio de sometimiento al sitio de ejecución

(a) Grid 1

Sitio i J	1	2	3	4	5	6	7
1	0	762	967	648	821	1229	1010
2	0	0	9	24	57	81	82
3	0	0	0	3	7	13	12
4	0	0	0	0	28	39	48
5	0	0	0	0	0	36	41
6	0	0	0	0	0	0	4
7	0	0	0	0	0	0	0
total	0	762	976	675	913	1398	1197

(b) Grid 2

Sitio i j	1	2	3	4	5	6	7	8	9
1	0	2121	1184	710	1025	583	784	591	981
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	2	4	4	3
6	0	0	0	0	0	0	24	35	48
7	0	0	0	0	0	0	0	0.3	0.6
8	0	0	0	0	0	0	0	0	18
9	0	0	0	0	0	0	0	0	0
total	0	0	0	0	1025	585	812	630	1050

La Tabla 6 muestra la migración total de cantidad de trabajo del sitio de sometimiento original al sitio de su ejecución. MW_{ij} representa el porcentaje de consumo de recursos de las tareas que han sido sometidas al sitio j y ejecutadas en el sitio i (W_{ij}), sobre el consumo total sometido al sitio j (W_j).

$$MW_{ij} = \frac{W_{ij}}{W_j} * 100 \quad (6)$$

De la Figura 10 a la Figura 13 se observa el efecto del robo de tareas en el balanceo de la carga

entre los distintos sitios. Puede observarse el promedio del número de tareas y consumo de recursos ejecutados, de las tareas en las colas A y B, tanto por sitio como por procesador de cada sitio. En la Figura 10, pareciera que el algoritmo no realiza una buena distribución de tareas para la configuración de Grid 2 entre los primeros 4 sitios, los cuales tienen igual número de procesadores. Sin embargo, en la Figura 11, se observa que el algoritmo ST fue efectivo al balancear la cantidad de recursos consumidos.

Tabla 5. Migración promedio número de tareas expresadas en porcentaje
(a) Grid 1

Sitio i J	1	2	3	4	5	6	7	Total
1	0	11.8	15.0	10.1	12.8	19.1	15.7	84.5
2	0	0	3.5	9.3	22.1	31.4	31.8	98.1
3	0	0	0	7.3	17.1	31.7	29.3	85.4
4	0	0	0	0	19.7	27.5	33.8	81.0
5	0	0	0	0	0	32.4	36.9	69.4
6	0	0	0	0	0	0	80.0	80.0
7	0	0	0	0	0	0	0	0.0

(b) Grid 2

Sitio i j	1	2	3	4	5	6	7	8	9	Total
1	0	17.4	9.7	5.8	8.4	4.8	6.4	4.9	8.1	65.5
2	0	0	0	0	0	0	0	0	0	0.0
3	0	0	0	0	0	0	0	0	0	0.0
4	0	0	0	0	0	0	0	0	0	0.0
5	0	0	0	0	0	12.5	25	25	18.8	81.3
6	0	0	0	0	0	0	19.8	28.9	39.7	88.4
7	0	0	0	0	0	0	0	30	60	90.0
8	0	0	0	0	0	0	0	0	39.1	39.1
9	0	0	0	0	0	0	0	0	0	0.0

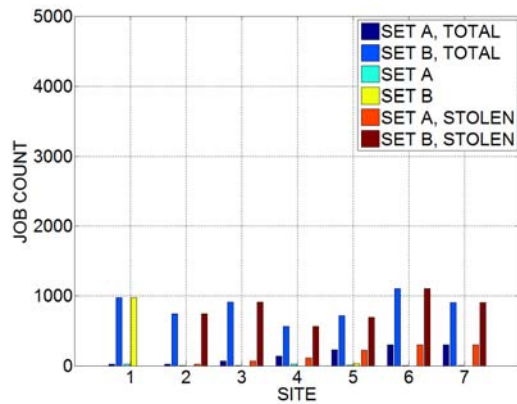
Tabla 6. El porcentaje promedio de trabajo migrado desde el sitio de sometimiento al sitio de ejecución

(a) Grid 1

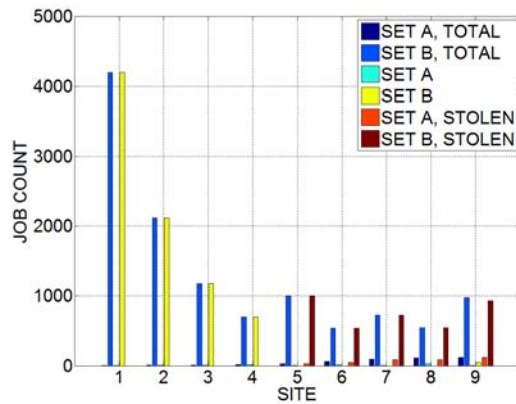
Sitio i J	1	2	3	4	5	6	7	Total
1	0	7.4	12.5	13.6	17.8	22.5	19.5	93.3
2	0	0	3.5	9.4	21.2	27.5	26.1	87.7
3	0	0	0	6.5	22.5	31.3	28.7	89.0
4	0	0	0	0	20.8	27.9	34.2	83.0
5	0	0	0	0	0	33.1	37.3	70.4
6	0	0	0	0	0	0	70.9	70.9
7	0	0	0	0	0	0	0	0

(b) Grid 2

Sitio i j	1	2	3	4	5	6	7	8	9	Total
1	0	9.6	8.9	0.8	12.9	12.2	16.6	11.5	9.5	82
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	16	31.9	24.2	17.7	89.8
6	0	0	0	0	0	0	19.9	29.9	36.6	86.4
7	0	0	0	0	0	0	0	15	54.6	69.6
8	0	0	0	0	0	0	0	0	45.6	45.6
9	0	0	0	0	0	0	0	0	0	0

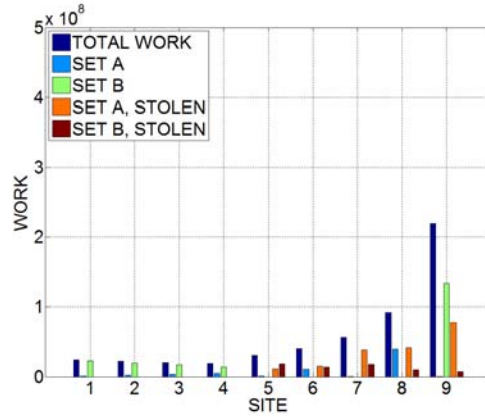
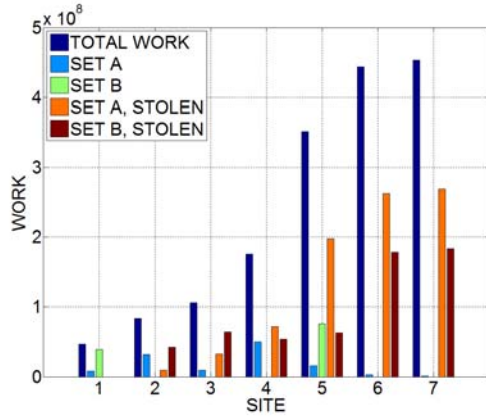


(a) Grid 1



(b) Grid 2

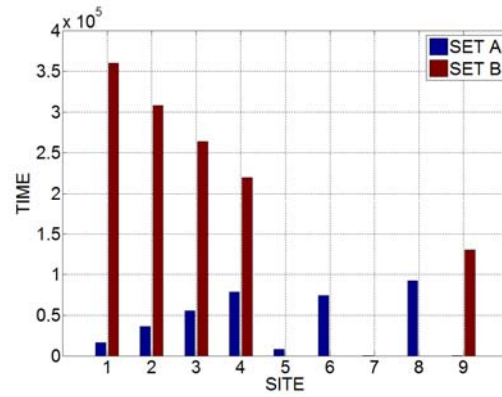
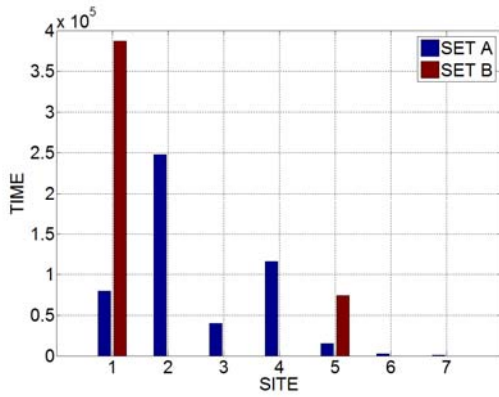
Figura 10. Número promedio de tareas ejecutadas de los conjuntos A y B ($ST50$)



(a) Grid 1

(b) Grid 2

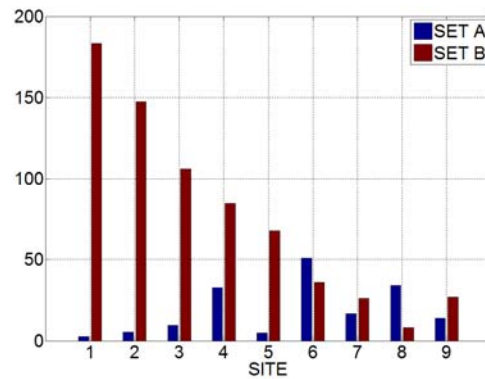
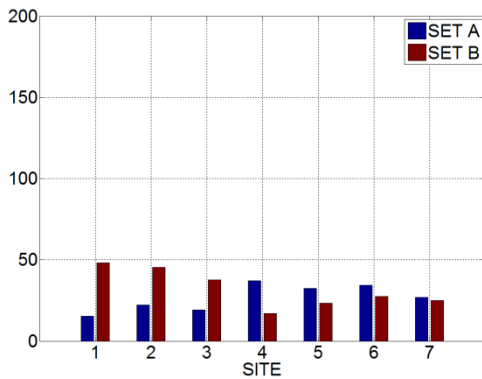
Figura 11. Consumo de recursos ejecutado promedio en las colas A y B ($S750$)



(a) Grid 1

(b) Grid 2

Figura 12. Consumo de recursos ejecutado promedio por procesador en las colas A y B ($S750$)



(a) Grid 1

(b) Grid 2

Figura 13. Carga paralela por procesador en las colas A y B ($S750$)

7. Conclusiones

El presente trabajo provee un análisis experimental del algoritmo de robo de tareas para calendarización en línea de un Grid computacional. Los resultados obtenidos muestran que el uso de un modelo descentralizado supera en desempeño a distintas

estrategias de modelo jerárquico de dos niveles, para minimización de C_{max} . Las estrategias con las que se comparó ST fueron: $Min_Lp+BEFF$, $Min_PL+BEFF$, $Min_ST+BEFF$. Se mostró que el empleo de un esquema de robo de tareas realiza la minimización del tiempo de finalización de calendario del sistema en las configuraciones propuestas, al lograr de manera

efectiva el balanceo de carga entre las distintas máquinas que forman un Grid.

Se propusieron distintas variantes del algoritmo ST. Se comprobó que algunas de ellas no generan resultados con diferencia estadística significativa. Estas variantes incluyen *Best Fit* y *First Fit* en la selección de tareas y distintos órdenes de búsqueda en las colas *A* y *B*. Sin embargo, variaciones en el valor frontera de las listas *A* y *B* produjeron resultados experimentales dentro del 1% respecto al óptimo, para la métrica factor de competitividad.

Como trabajo futuro se propone el análisis del algoritmo ST en modelos de Grid, donde características del modelo, por ejemplo, la comunicación punto a punto y la información de calendarios de otras máquinas, se cumple solo de forma parcial.

8. Referencias

[1] Albers, S. Better bounds for online scheduling, *SIAM Journal on Computing*, vol. 29, no. 2, pp. 459–473, 1999.

[2] Dong, F. and S. G. Akl. Scheduling Algorithms for Grid Computing: State of the Art and Open Problems. Technical Report No. 2006-504 School of Computing, Queen's University Kingston, Ontario. 55 pp. 2006.

[3] Feitelson, D.
<http://www.cs.huji.ac.il/labs/parallel/workload/logs.html>. (2010)

[4] Foster, I. and C. Kesselman. The Grid: Blue Print for a New Computing Infrastructure. Morgan Kaufman. San Francisco, CA. USA. 677pp. 1999.

[5] González-García, J. Evaluación Experimental de Estrategias de Calendarización en Grid Computacional Utilizando un Esquema de Admisibilidad. Memorias del ANIEI-UABC 2009 - XXII Congreso Nacional y VIII Congreso Internacional de Informática y Computación de la Asociación Nacional de Instituciones de Educación en Tecnologías de la Información, A.C. (ANIEI).

[6] Grimme, C., J. Lepping and A. Papaspyrou. Prospects of Collaboration between Compute Providers by means of Job Interchange. En: Frachtenberg, E. y U. Schwiegelshohn (eds.). Proceedings of the 13th Job Scheduling Strategies for Parallel Processing. LNCS. Springer. 2007.

[7] Hirales-Carbajal, A., A. Tchernykh, T. Röblitz, R. Yahyapour. A Grid Simulation Framework to Study Advance Scheduling Strategies for Complex Workflow Applications. HPGC High Performance Grid Computing in conjunction with the 24th IEEE International Parallel and Distributed Processing

Symposium April 19-23, 2010, ATLANTA (Georgia) USA (accepted for publication)

[8] Krauter, K., R. Buyya and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software-Practice and Experience*. 32(2). 135-164 p. 2002.

[9] Madrigal, D., L. Galaviz, J. Ramírez, R. Sánchez, A. Tchernykh, J. Verduzco. "Estrategias de Calendarización para un Grid Computacional" CiComp'06.- 1er. Congreso Internacional de Ciencias Computacionales, a celebrarse en la ciudad de Ensenada, B.C., México del 6 al 8 de Noviembre de 2006

[10] Naroska, E. and U. Schwiegelshohn, On an online scheduling problem for parallel jobs *Information Processing Letters*, vol. 81, no. 6, pp. 297–304, March 2002.

[11] Pascual, F., K. Rządca y D. Trystram. Cooperation in multi-organization scheduling. *Concurrency and Computation: Practice and Experience*, 21(7):905–921, May 2009.

[12] Ramírez, J., D. Madrigal, A. Tchernykh Scheduling Strategies for a Computational Grid. . Information Systems and Technology 2007. MATI-PVK April 4th, 2007, Moscow

[13] Ramírez, J., A. Rodríguez, A. Tchernykh, J. Verduzco. Rendimiento de Estrategias de Calendarización Considerando Fluctuación de Tiempo de Ejecución de Tareas en un Grid Computacional. Conferencia Latinoamericana de Computación de Alto Rendimiento, CLCAR 2007. Santa Marta, Colombia del 13 al 18 de Agosto de 2007. p. 273-281

[14] Salazar-Ricarte, E. y A. Tchernykh. Workload for grid scheduling strategies evaluation Information Systems and Technology 2007. MATI-PVK April 4th, 2007, Moscow

[15] Salazar, E., M. Valenzuela, D. Mendoza, Y. Castro, J. Ramírez, E. Rodríguez, A. Tchernykh. Evaluación de las Estrategias de Calendarización en un Grid Computacional Jerárquico CICC 2005.- 6°. Congreso Internacional de las Ciencias Computacionales, Villa de Alvarez, Colima, Mex. (September 28-30, 2005) , 2005

[16] Schopf, J.M. Ten actions when Grid Scheduling. En: J. Nabrzyski, J.M. Schopf y J. Weglarz (eds). Grid Resource management: State of the Art and Future Trends. Kluwer Academic Publisher, Boston, USA, 15-24 p. 2004.

[17] Schwiegelsohn, U., A. Tchernykh y R. Yahyapour. Online Scheduling in Grids, IPDPS 2008, IEEE International Symposium on Parallel and

Distributed Processing, April 14-18, 2008, Miami, Florida USA pp. 1-10

[18] Tchernykh, A., U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin. Online Hierarchical Job Scheduling in Grids, CoreGRID Symposium in conjunction with EuroPar 2008 Conference, Las Palmas de Gran Canaria (Spain), August 26th-29th, in book "From Grids to Service and Pervasive Computing" Thierry Priol and Marco Vanneschi (Eds.). pp. 77-91, Springer-Verlag, 2008

[19] A. Tchernykh, U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin "Online Hierarchical Job Scheduling on Grids with Admissible Allocation", Journal of Scheduling, Volume 13, Issue 5, pp. 545—552. Springer-Verlag, Netherlands, 2010. DOI: 10.1007/s10951-010-0169-x. ISSN: 1094-6136 (Print) 1099-1425