Despliegue Adaptativo de Aplicaciones en Sistemas Grid Basado en el Concepto de Autómatas Celulares

María Botón-Fernández[1], Francisco Prieto Castrillo[1] y Miguel A. Vega-Rodríguez[2]

Resumen—**Los entornos de computación grid se caracterizan por facilitar el despliegue de aplicaciones de cómputo masivo. A pesar de las ventajas de este nuevo paradigma, presenta un conjunto de problemas que están siendo activamente investigados por la comunidad científica. Uno de los problemas de este entorno, que está siendo muy estudiado en los últimos años, es la selección de los recursos que mejor se adapten a los requisitos de la aplicación. La naturaleza dinámica de este tipo de infraestructura dificulta dicho proceso, repercutiendo negativamente en el rendimiento de las aplicaciones.**

Por este motivo, en el presente estudio se propone un modelo de selección eficiente que permita a las aplicaciones auto-adaptarse a la naturaleza cambiante del entorno. Para ello, los recursos grid son modelizados como celdas de un autómata celular. Basado en el *juego de la vida*, **este autómata estará gobernado por un sencillo conjunto de reglas, de modo que los recursos que sobrevivan en cada intervalo de tiempo serán los más eficientes.**

Finalmente, para la evaluación del modelo se ha hecho uso de una grid real. De los resultados obtenidos se deduce que esta selección eficiente proporciona una reducción de los tiempos de ejecución así como un incremento de la tasa de tareas finalizadas, beneficiando el despliegue de las aplicaciones en estos entornos.

Palabras clave—**computación grid, autómata celular, optimización, aplicaciones paralelas**

I. Introducción

La computación grid [1]-[3] es un nuevo paradigma, dentro de la computación distribuida, basado en el uso de recursos de cómputo y almacenamiento de forma coordinada. Este tipo de entornos se caracteriza por gestionar recursos de naturaleza heterogénea que se encuentran dispersos geográficamente. Otra característica de los entornos grid es la creación de organizaciones virtuales (*VO*); cada *VO* está formada por una serie de instituciones y/o centros afiliados a un proyecto de investigación. Además, cada una de ellas tiene sus propias reglas para controlar la compartición de sus recursos.

A pesar de ser una infraestructura cada vez más utilizada frente a la computación paralela, presenta problemas en cuanto a la gestión, monitorización, descubrimiento y selección de los recursos, influyendo todo ello negativamente en el rendimiento de la grid. A esto debemos añadir que, al ser entornos geográficamente distribuidos, la comunicación entre sus elementos suele ser lenta y poco uniforme. Además, su naturaleza cambiante provoca que los recursos varíen en el tiempo en cuanto a disponibilidad y características. En definitiva, podría decirse que se trata de un paradigma escalable, que proporciona una capacidad de cómputo prácticamente ilimitada y que se encuentra todavía en fase de desarrollo. Debido a esta naturaleza dinámica y heterogénea, el encontrar un conjunto de recursos adecuado para cada aplicación se ha convertido en un reto muy estudiado por la comunidad científica en los últimos años. Una solución propuesta por varias investigaciones [4]-[7] consiste en dotar a las aplicaciones grid de una capacidad adaptativa a los cambios del entorno. Para ello, se definen frameworks que gestionen de una manera más eficientes los trabajos. Otros estudios [8] y [9] aplican el concepto de adaptabilidad en los sistemas grid, utilizando mecanismos de gestión que se adapten autónomamente a los cambios de la infraestructura. Todas estas contribuciones tienen un propósito común: mejorar el rendimiento de la infraestructura. Diferentes técnicas de scheduling, migración, balanceo de carga o disminución de errores de cuello de botella son aplicados para conseguirlo.

La investigación que se presenta en este trabajo se centra en resolver el problema de selección de recursos permitiendo a las aplicaciones auto-adaptarse cuando sea necesario. Una de las principales diferencias entre nuestro trabajo y los comentados anteriormente es que el modelo propuesto ha sido definido desde el nivel de aplicación (Figura 1). Es decir, se trata de una subcapa dentro de este nivel que guía a las aplicaciones durante su ejecución en la grid. No se emplean estrategias de scheduling, ni migración o reasignación de tareas, es decir, no se lleva a cabo ninguna operación a nivel de administración de la infraestructura.

Otra diferencia importante es que el modelo se centra en explotar la productividad de la infraestructura sin modificar las características o el comportamiento de los elementos grid. En tiempo de ejecución obtiene información de ciertos componentes, que se utiliza junto con una serie de métricas (sección II-B) para determinar cuales son los recursos más eficientes. En esta versión, denotada como *ERS-CA* (*Efficient Resources Selection Based on Cellular Automata*), los elementos grid a monitorizar quedan representados por celdas de un Autómata Celular (*AC*) y la selección de los mismos va a depender de un conjunto sencillo de reglas.

[1]Departamento de Ciencia y Tecnología, Ceta-Ciemat, e-mail: {maria.boton, francisco.prieto}@ciemat.es

[2]Dpto. de Tecnología de Computadores y Comunicaciones, Univ. Extremadura, e-mail: mavega@unex.es

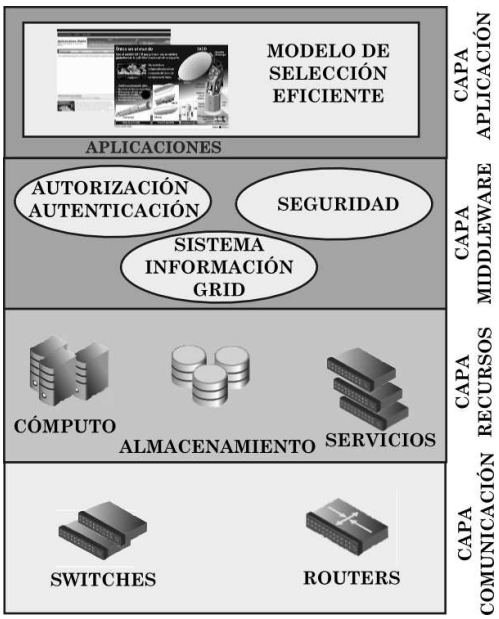

Fig. 1. Niveles básicos de una infraestructura grid. En el nivel de aplicación reside el modelo propuesto.

II. Modelo de Selección Eficiente. Diseñando la Grid como un Autómata Celular

A. Infraestructura Grid

Como se ha indicado anteriormente, los sistemas grid se caracterizan por su naturaleza heterogénea y su arquitectura jerarquizada (Figura 2). De hecho, los elementos que componen este tipo de entornos son recursos dedicados con una funcionalidad específica.

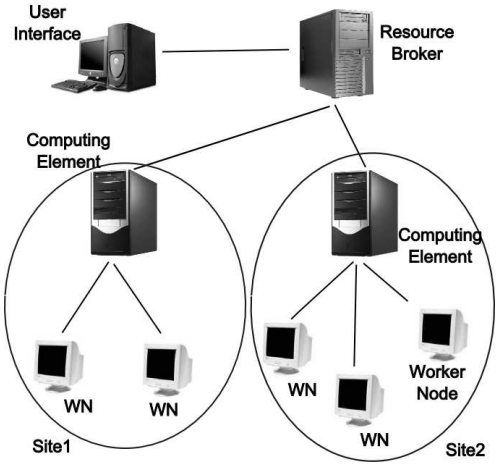

Fig. 2. Representación jerárquica de los elementos grid involucrados en el proceso de ejecución de trabajos.

La interacción de un usuario con una infraestructura grid tiene lugar a través del punto de acceso conocido como *interfaz de usuario* (*UI*), una máquina donde el usuario tiene una cuenta personal con sus certificados grid instalados. A través de la *UI* se lanzan las tareas a ejecución. Para el envío de trabajos es necesario crear un fichero de descripción por cada tarea a lanzar, donde se especifican las características y requisitos de la misma. Una vez el trabajo es enviado a la grid, éste es gestionado por el *Workload Management System* (*WMS*) cuya finalidad es aceptar los trabajos enviados, asignarlos al *Computing Element* (*CE*) más apropiado, registrar su estado durante toda la ejecución y devolver la salida generada. La máquina que soporta este servicio se conoce como *Resource Broker* (*RB*). En cuanto a los *CEs*, se trata de un tipo de planificador a nivel de site que decide en qué nodo de cómputo *Worker Node* (*WN*) se ejecuta la tarea. La elección del *CE* se hace aplicando un proceso conocido como *match-making*, según el cual se tiene en cuenta la disponibilidad y la bondad del recurso así como los requisitos de la tarea.

B. Determinando la Eficiencia de los Recursos Grid

A la hora de determinar en qué nodo ejecutar una determinada tarea, son varios los recursos que se ven implicados. Para decidir con cual de ellos vamos a trabajar se tienen presenten los objetivos marcados:

- Seleccionar los recursos más eficientes.
- Reducir el tiempo de ejecución de la aplicación.
- Mejorar la tasa de tareas finalizadas con éxito.

En esta decisión influyen también las características que hacen que nuestro modelo se desmarque de las investigaciones desarrolladas hasta el momento. Éstas son, por un lado, realizar el diseño a nivel de aplicación y no como una nueva técnica de scheduling o un nuevo sistema de planificación. Por otro lado, hacer uso de los recursos que tienen a su disposición los usuarios para su interacción con la infraestructura (conjunto de ordenes para envío, consulta, monitorización, etc sobre diferentes elementos grid) así como tener en cuenta las limitaciones que sufren. En consecuencia, el modelo va a monitorizar la eficiencia de los *CEs*, ya que es el elemento más próximo a los nodos de cómputo con los que el usuario puede interaccionar directamente desde la *UI*. Además, tan importante es que el nodo de cómputo tenga un buen rendimiento como que el planificador gestione adecuadamente su cola de trabajos.

Una vez determinado qué recursos vamos a seleccionar de manera eficiente, el siguiente paso es fijar cómo se va a medir esa eficiencia. Para ello, el modelo gestionará dos espacios de trabajo durante la ejecución de la aplicación: un espacio de tareas $\mathbf{J}$, compuesto por n tareas independientes a procesar, y un espacio de recursos heterogéneos $\mathbf{R}$, donde se incluyen los m *CEs* de la infraestructura. El conjunto de tareas J es dividido en una serie de subconjuntos, todos ellos del mismo tamaño, denotados como *unidades de proceso* o $\mathbf{P}_\alpha$, para una mejor gestión de dicho espacio. Para cada P_α se selecciona un subconjunto de recursos $\mathbf{RP}_\alpha \subset R$ de manera eficiente, lo que implica que antes de lanzar un nuevo P_α los valores de eficiencia de los recursos son actualizados. Cada RP_α tiene un *tiempo de vida lt* para ejecutar las tareas del P_α correspondiente. Los valores de eficiencia de los recursos son actualizados cuando

terminan todas las tareas de P_α o cuando lt finaliza. Cabe destacar que para la primera *unidad de proceso* el conjunto de recursos se escoge al azar, ya que al inicio de la ejecución el modelo no tiene métricas de eficiencia de los recursos (se consideran todos igual de eficientes).

En lo que a las métricas de eficiencia se refiere, son varios los parámetros definidos en el modelo. Todos ellos se detallan a continuación. En primer lugar se define el tiempo de procesamiento $\mathbf{T_i}$ de un *CE* **i** (Ec.1), que depende del tiempo de comunicación del recurso $\mathbf{Tcomm_i}$ con otros servicios grid y del tiempo de procesamiento $\mathbf{Tcomp_{j,i}}$ de cada tarea **j** alojada en el recurso i. $\mathbf{NT_i}$ es el número de tareas asignadas a dicho recurso.

$$T_i = Tcomm_i + \sum_{j \in NT_i} Tcomp_{j,i} \,. \qquad (1)$$

A partir de este parámetro se puede determinar el incremento del tiempo de procesamiento $\mathbf{\Delta T_i}$ del CE i que, como se observa en Ec.2, es relativo al conjunto RP_α al que pertenece el recurso. Los parámetros $\mathbf{T_{max}}$ y $\mathbf{T_{min}}$ son los valores máximo y mínimo de T_i alcanzados en RP_α.

$$\Delta T_i = (T_{max} - T_i)/(T_{max} - T_{min}) \,. \qquad (2)$$

Finalmente, el fitness del recurso $\mathbf{F_i}$ (Ec.3), que indicará como de eficiente ha sido, va a venir determinado por su tasa de tareas finalizadas con éxito $\epsilon_\mathbf{i}$ y por ΔT_i. Además, cada uno de estos parámetros - que se pueden considerar como los principales del modelo - llevan asociado un *peso* o valor de *relevancia* denotado como **a** y **b** respectivamente. Ambos parámetros son especificados por el usuario y facilitan el uso del modelo en un rango amplio de aplicaciones paralelas.

$$F_i = (a \cdot \epsilon_i + b \cdot \Delta T_i)/(a + b) \,. \qquad (3)$$

C. Modelo de Selección Eficiente Basado en Autómatas Celulares

Una vez que se ha establecido cómo medir la eficiencia de los recursos de una infraestructura grid, se define más detalladamente el proceso de selección eficiente. Como se ha indicado anteriormente, se va a hacer uso de metodologías basadas en *AC*.

Un *AC* [10] es un modelo computacional compuesto por un conjunto finito de estados, representados en una malla, que evolucionan a pasos discretos. En cada paso de tiempo, el autómata evoluciona mediante la aplicación de un conjunto de reglas (modificándose el valor de todas las celdas al mismo tiempo). El nuevo valor de una celda se obtiene en base a su valor actual y el de sus vecinas. Se puede decir, por tanto, que los *AC* son sistemas dinámicos capaces de dar lugar a un comportamiento complejo partiendo de unas premisas simples.

Existen multitud de *AC* aplicados a un amplio rango de disciplinas. Sin embargo, el más conocido es el denominado *juego de la vida* de John Conway [11]. Las reglas del mismo son simples y se especifican a continuación:

- Una celda *muerta* con exactamente 3 vecinos vivos **nace**.
- Una celda *viva* con 2 o 3 vecinos vivos **permanece viva**.
- Una celda *viva* con 4 o más vecinos vivos **muere** (superpoblación).
- Una celda *viva* con menos de 2 vecinos vivos **muere**.

El *AC* que se diseña en nuestro modelo eficiente está basado en este tipo de autómata, pero también en los *AC* de tipo red [12] [13]. La idea sería tener subredes de *AC*, todas ellas con las mismas reglas de evolución. Para el caso que nos ocupa, cada celda del autómata representa un *CE* de la infraestructura. Como el valor de cada celda se obtiene teniendo en cuenta a sus vecinos, el concepto de **vecindad** utilizado en el modelo propuesto es: *dos celdas del autómata son vecinas si los recursos que representan son vecinos geográficos.* En resumen, se aplica un criterio de proximidad geográfica (Figura 3), quedando la infraestructura utilizada [14] dividida en 8 subredes. El número de subredes resultante depende de la cantidad de recursos que hay disponibles en el entorno. Esta información queda especificada en un fichero de configuración que podrá ser manipulado por el usuario. Además, todas las subredes en *ERS-CA* tendrán al menos 3 recursos y estarán conectadas unas con otras.

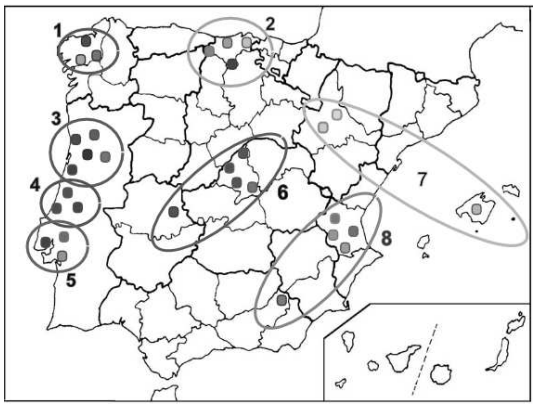

Fig. 3. Distribución geográfica de los recursos del proyecto Ibergrid y su agrupación en redes de autómatas.

Cada celda del autómata en *ERS-CA* puede tener tres valores posibles: *viva, muerta o inactiva.*

- Cuando un recurso se considera eficiente y va a formar parte de un nuevo RP_α su celda correspondiente estará **viva**.
- De la misma forma, todo recurso que no sea considerado lo suficientemente eficiente será tratado como una celda **muerta**.
- Finalmente, los recursos de la infraestructura que no estén disponibles u operativos tendrán el estado **inactivo**.

Durante el proceso de selección sólo dos celdas so-

brevivirán en cada subred: las dos que correspondan a los recursos menos cargados en el RP_α (éstas competirán con las supervivientes del resto de subredes para pasar al estado *vivo*. El resto de celdas vecinas de cada subred pasará al estado *muerto*). Este valor de carga de un recurso es obtenido en el instante wt dentro del tiempo de vida lt del bloque. Añadiendo este criterio se consigue, por una parte, no sobrecargar los recursos y, por otra, permitir que nuevos elementos puedan formar parte de una nueva solución (RP_α). A continuación, con todas las celdas supervivientes del autómata se construye una lista de candidatos a la que se le aplican las siguientes reglas:

- **Presión Selectiva**: solo la mitad más eficiente de esta lista de candidatos promociona para ser parte del nuevo RP_α.
- **Exploradora**: mientras haya *CE* inexplorados (no utilizados en ningún RP_α), se escogerá la celda correspondiente a uno de ellos de manera aleatoria para formar parte del nuevo RP_α (pasa a estado *vivo*).
- **Celdas vivas**: los recursos escogidos para formar parte del siguiente RP_α (escogidos al aplicar *la presión selectiva*) actualizarán su estado a *vivo*, mientras que el resto lo actualiza a *muerto*.

Aplicando estas reglas al conjunto de candidatos se pretende que el modelo sea lo más voraz posible en la búsqueda del conjunto de recursos que mejor se adapte a los requisitos de la aplicación. Finalmente, las reglas que rigen el comportamiento del modelo *ERS-CA* quedan recogidas en la Figura 4

PSEUDOCODIGO ERS-CA

Input: tareas de la aplicación, recursos de la infraestructura
Output: conjunto de soluciones

1. Preparar el entorno;
2. **mientras** haya unidades de proceso **hacer**
   2.1 Ejecutar y monitorizar la unidad de proceso P_α ;
   2.2 Actualizar las métricas de eficiencia de los recursos;
   2.3 Selección eficiente de RP_α utilizando CA;
      2.3.1 Evaluar la carga de los recursos;
      2.3.2 **Supervivientes:** los dos recursos menos cargados de cada subred promocionan como candidatos;
      2.3.3 **Presión selectiva**: sólo la mitad más eficiente del conjunto de supervivientes será parte de la solución;
      2.3.4 **Exploradora**: un recurso inexplorado se escoge al azar para ser parte de la solución;
3. **Fin mientras**

Fig. 4. Reglas que gobiernan el comportamiento del autómata para el modelo *ERS-CA* propuesto.

III. Evaluación del Modelo

Para conseguir una estimación fiable de las métricas definidas, el modelo *ERS-CA* debe ser probado en una infraestructura grid con un número significativo de *CEs*. Por este motivo, se hace uso de la infraestructura perteneciente a la Iniciativa Grid Nacional [15] (*ES-NGI*). Se trata de una infraestructura de computación distribuida que proporciona servicios a proyectos dentro de los campos de la Física de Altas Energías, Ciencias de la Tierra, Ciencias de la Vida, Astronomía, etc. Específicamente, como se ha indicado en la sección anterior, se hace uso de la infraestructura perteneciente al proyecto Ibergrid [14]. Se trata de un proyecto donde se unen las iniciativas grid nacionales de España y Portugal.

A continuación, se detallan los escenarios que se han desarrollado para esta fase de prueba del modelo. En ambos se compara el modelo *ERS-CA* con la selección tradicional en grid que hemos denotado como *TRS* (Traditional Resources Selection). En ésta, los *CEs* son escogidos en base a tres grandes criterios: disponibilidad, cercanía y requisitos de las tareas a ejecutar.

A. Primer Escenario de Prueba

En este escenario se pretende determinar hasta qué punto influye el aprendizaje del modelo en una buena especialización de los recursos. Es decir, averiguar el aprendizaje mínimo del modelo para apreciar mejoras significativas en el despliegue de las aplicaciones. Para ello, nos interesa corroborar que se cumplen los siguientes objetivos: *reducción del tiempo de ejecución de las aplicaciones e incremento de la tasa de tareas finalizadas con éxito.*

Se han definido 5 pruebas en esta etapa de evaluación, de manera que cada una realiza 10 simulaciones por cada versión: *ERS-CA* y *TRS*. Como se puede observar en la Tabla 1, el tamaño para el espacio J ha sido fijado a 200 tareas mientras que el tamaño de P_α va decreciendo en cada test. Al ir disminuyendo el tamaño de los subconjuntos de J se obtiene, proporcionalmente, un incremento en el número de iteraciones del modelo (lo que implica un mayor conocimiento del estado de la infraestructura y, en consecuencia, del rendimiento de sus elementos). Los parámetros a y b tomarán los valores 60% y 40% respectivamente, es decir, se le da una relevancia ligeramente mayor a la tasa ϵ_i de tareas finalizadas con éxito.

TABLA I
Valores asignados a P_α y J para el primer escenario de evaluación.

Test	Tamaño P_α	Iteraciones	Tamaño J
1	5	40	200
2	10	20	200
3	13	15	200
4	20	10	200
5	40	5	200

De los resultados obtenidos en esta primera fase de prueba (Figura 5) se concluye que el modelo *ERS-CA* mejora los tiempos de ejecución de la aplicación respecto a *TRS*, cumpliéndose uno de los objetivos marcados. Puede apreciarse que en los 3 últimos tests es donde *ERS-CA* alcanza una mayor diferencia, la cual es menos significativa en las primeras pruebas. Esto se debe a las reglas de supervivencia y presión selectiva establecidas, que aseguran que para cada P_α una cierta cantidad de recursos es evaluada.

Además, si tenemos en cuenta que para la asignación *tareas-recursos* se ha hecho uso de la técnica de *round robin*, podemos decir que cuanto mayor sea el número de tareas en P_α (el caso de los 3 últimos tests) mayor cantidad de recursos cargados habrá, provocando que otros nuevos sean elegidos más rápidamente. En definitiva, los recursos más eficientes son encontrados más rápidamente en las últimas pruebas.

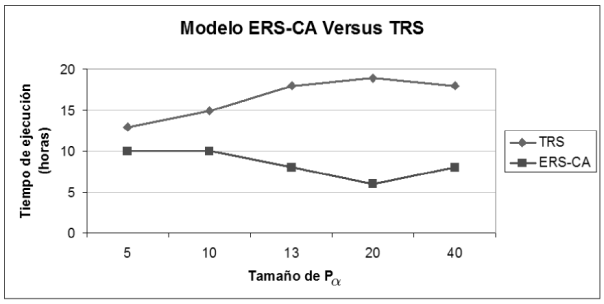

Fig. 5. Tiempo de ejecución obtenido en los tests aplicados a *ERS-CA* y *TRS* en el escenario 1. El modelo eficiente obtiene mejores resultados.

Por último, en cuanto a la tasa de tareas finalizadas con éxito, cabe destacar que frente a un 73% de tareas acabadas correctamente alcanzado por el *TRS* el modelo *ERS-CA* consigue un 87.2%.

B. Segundo Escenario de Prueba

En esta segunda fase de evaluación el objetivo es determinar el rango de aplicaciones posibles en las que aplicar el modelo. Es decir, verificar que el modelo es efectivo para las aplicaciones típicas de los entornos grid. Por este motivo, se irá variando en las diferentes pruebas que componen este escenario el tamaño del espacio J. En cuanto al tamaño de P_α se fija a 10 en todas las pruebas, como puede observarse en la Tabla 2. De los resultados obtenidos en el primer escenario se deduce que es el menor subconjunto con el que se puede trabajar para que el modelo sea efectivo.

TABLA II
RANGO DE VALORES DEL ESPACIO J EN EL SEGUNDO ESCENARIO.

Test	Tamaño P_α	Iteraciones	Tamaño J
1	10	5	50
2	10	10	100
3	10	20	200
4	10	30	300
5	10	40	400
6	10	50	500

Se espera que a medida que este tamaño de J aumente mejoren los tiempos de ejecución de la aplicación con respecto al *TRS*, ya que a mayor número de iteraciones mejor será el aprendizaje del modelo. Por otra parte, al igual que en el primer escenario, los valores para los parámetros a y b serán 60% y 40% respectivamente.

Nuevamente, de los resultados obtenidos (Figura 6) puede deducirse que el modelo *ERS-CA* mejora con mucho a la selección *TRS*. En los 2 primeros tests los valores de tiempo de ambas técnicas son próximos pero, a medida que J aumenta lo hace también la distancia entre ambos (como se había supuesto en un principio).

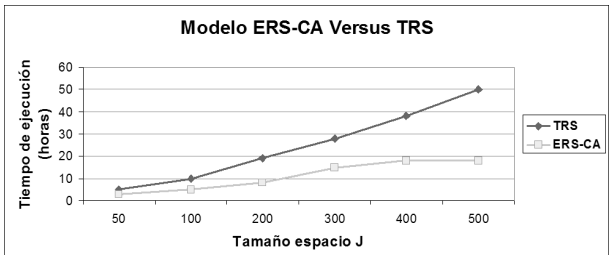

Fig. 6. Resultados obtenidos para *ERS-CA* y *TRS* en el segundo escenario de evaluación.

Por tanto, queda demostrado que el modelo de selección eficiente propuesto en este estudio mejora la productividad de la infraestructura, a la vez que reduce el tiempo de ejecución de las aplicaciones e incrementa la tasa de tareas finalizas con éxito.

IV. Conclusiones

En este trabajo nos hemos centrado en resolver el problema de selección de recursos en entornos grid mediante un despliegue adaptativo de las aplicaciones en los mismos. El modelo propuesto no sólo proporciona una capacidad auto-adaptativa a las aplicaciones frente a los cambios del entorno, sino que además mejora la productividad de la infraestructura. Para cumplir estos objetivos no es necesario un conocimiento en profundidad del entorno ni de la aplicación, sólo unos conceptos generales como capacidad de recursos, tareas a ejecutar, etc.

Esta estrategia inteligente actúa como una subcapa dentro del nivel de aplicación, siendo capaz de guiar de manera eficiente la ejecución de los experimentos científicos. Dos escenarios han sido desarrollados para la fase de evaluación. Cada uno de ellos está compuesto por un conjunto de pruebas, donde diferentes parámetros del modelo son variados para mostrar los beneficios y aplicabilidad del mismo. En ambos escenarios, el modelo (conocido como *ERS-CA*) es comparado con la selección tradicional de recursos en este tipo de infraestructura (denotada como *TRS*). De los resultados obtenidos se puede deducir que se consigue, también, una reducción del tiempo de ejecución de las aplicaciones y un incremento en la tasa de tareas finalizadas con éxito.

Como trabajo futuro se pretende aplicar nuevas metodologías en el proceso de selección, utilizar nuevos servicios y facilidades grid, profundizando en el conocimiento de la infraestructura para proporcionar una mejorar capacidad adaptativa a las aplicaciones.

Agradecimientos

La investigación de María Botón-Fernández está subvencionada mediante una beca de Formación de Personal Investigador otorgada por el Ministerio de Economía y Competitividad a través del Centro de Investigaciones Energéticas, Medio Ambientales y Tecnológicas (CIEMAT). Los autores quieren agradecer también el soporte de los Fondos Europeos para el Desarrollo Regional (FEDER).

Referencias

[1] I. Foster, C. Kesselman, *The Grid 2: Blueprint for a New Computing Infrastructure*, in: Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[2] I. Foster, C. Kesselman, S. Tuecke, *The Anatomy of the Grid. Enabling Scalable Virtual Organizations*, in: R. Sakellariou, J.A. Keane, J.R. Gurd, L. Freeman (eds.), Euro-Par 2001, LNCS, Vol. 2150, Springer-Verlag, Heidelberg, pp. 1-4, 2001.

[3] I. Foster, *What is the Grid? A Three Point Checklist*, in: GRIDtoday, Vol. 1, No. 6, Tabor Communications Inc, pp. 22-25, 2002.

[4] D. Groen, S. Harftst and S.P. Zwart, *On the Origin of Grid Species: The Living Application*, in: Proceedings of the 9th International Conference on Computational Science: Part I, LNCS, Vol. 5544, Springer-Verlag, Heidelberg, pp. 205-212, 2009.

[5] G. Wrzesinska, J. Maasen and H.E. Bal, *Self-adaptive Applications on the Grid*, in: 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programing, San Jose, California, USA, pp. 121-129, 2007.

[6] S.S. Vadhiyar and J.J. Dongarra, *Self Adaptivity in Grid Computing*, in: Concurrency and Computation: Practice & Experience, Vol. 17, Issue 2-4, John Wiley and Sons Ltd., pp. 235-257, 2005.

[7] E. Huedo, R.S. Montero and I.M. Llorente, *A Framework for Adaptive Execution in Grids*, in: Software: Practice & Experience, Vol. 34, Issue 7, John Wiley and Sons Inc., pp. 631-651, 2004.

[8] D.M. Batista and L.S. Da Fonseca, *A Survey of Self-adaptive Grids*, in: IEEE Communications Magazine, Vol. 48, Issue 7, IEEE Press Piscataway, pp. 94-100, 2010.

[9] H.N.L.C Keung, J.R.D. Dyson, S.A. Jarvis and G.R. Nudd, *Self-adaptive and Self-optimising Resource Monitoring for Dynamic Grid Environments*, in: DEXA'04 Proceedings of the Database and Expert Systems Applications, 15th International Workshop, IEEE Computer Society, Zaragoza, Spain, pp. 689-693, 2004.

[10] J.V. Neumann, *The Theory of Self-reproducing Automata*, in: A.W Burks (ed.), University of Illinois Press Champaign, IL, USA, 1966.

[11] M. Gardner, *The Fantastic Combinations of John Conway's New Solitaire Game "Life"*, in: Scientific American, Vol. 223, No. 4, Scientific American Inc., pp. 120-123, 1970.

[12] X.S. Yang and Y.Z.L. Yang, *Cellular Automata Neworks*, in: A. Adamatzky, L. Bull, B. De Lacy Costello, S. Stepney, C. Teuscher (eds.), Proceedings of Unconventional computing, Luniver Press, UWE and EPSRC, pp. 280-302, 2007.

[13] M. Tomassini, *Generalized Automata Networks*, in: S. E. Yacoubi, B. Chopard and S. Bandini (eds.), ACRI 2006, LNCS, Vol. 4173, Springer-Verlag, Heidelberg, pp. 14-28, 2006.

[14] Página Web del Proyecto Ibergrid, `http://www.ibergrid.eu/`

[15] Página Web de la Iniciativa Grid Nacional en España, `http://www.es-ngi.es/`