BoT-R: Una técnica para reducir la fragmentación en la meta-planificación por adelantado de Sistemas Grid

Luis Tomás, Carmen Carrión y Blanca Caminero [1]

Resumen— **Proporcionar QoS en entornos Grid, donde los recursos son de tipo heterogéneo y están dispersos geográficamente, es una tarea compleja que necesita ser estudiada. Una solución comúnmente propuesta es el uso de reservas por adelantado de los recursos. Sin embargo, dichas reservas no son siempre posibles en este tipo de entornos. Por esto, la solución propuesta por los autores es realizar una planificación por adelantado de dichos recursos pero sin reservarlos físicamente. De esta forma es más fácil que los recursos estén disponibles cuando los trabajos lo necesiten.**

Sin embargo, este tipo de planificación de trabajos puede generar fragmentación en el uso de los recursos. Esto significa que habrá periodos de tiempo libres en los recursos con escasas opciones de ser aprovechados, y por consiguiente, pudiendo llevar a una baja utilización de los mismos. Por esta razón se ha desarrollado una técnica que solventa estos problemas de fragmentación. Dicha técnica consiste en replanificar trabajos previamente planificados. A tal fin se han implementado heurísticos que deciden sobre qué intervalos se tienen que aplicar dichas técnicas, así como qué recursos y trabajos serán incluidos en el proceso. Esta técnica de replanificación se ha evaluado usando un entorno Grid real consistente en recursos heterogéneos pertenecientes a diferentes organizaciones. Los resultados ponen de manifiesto la eficiencia de la propuesta para proporcionar una mayor calidad de servicio a los usuarios.

Palabras clave— **Fragmentación, Grid, Meta-planificación por adelantado, Predicción, Replanificación**

I. Introducción

Los Grids son entornos computacionales altamente heterogéneos y distribuidos que permiten la ejecución de aplicaciones masivamente paralelas en ciencia, ingeniería y comercio [1]. Esta variabilidad hace que la calidad de servicio (QoS) sea altamente deseable pero difícil de proveer en la práctica.

La reserva por adelantado de recursos ha sido identificada en varios trabajos como un requisito básico para poder garantizar dicha QoS [2], pero éstas no son siempre factibles ya que no todos los gestores de recursos proporcionan dicha funcionalidad, así como la existencia de otro tipo de capacidades de los recursos que no pueden ser fácilmente reservados, como el ancho de banda. Por ello, nuestro trabajo está basado en planificación por adelantado, donde los recursos y los periodos de tiempo en los que ejecutar un trabajo son seleccionados por adelantado (y tenidos en cuenta para la toma de siguientes decisiones) pero sin hacer ninguna reserva física de ellos. Para ello, previamente se ha propuesto una arquitectura para la meta-planificación por adelantado de trabajos (SA-Layer [3]) que tiene en cuenta el estado de los recursos del sistema a la hora de tomar las decisiones de planificación. Sin embargo, se puede generar fragmentación durante dicho proceso, conllevando que algunos trabajos puedan ser rechazados incluso cuando la capacidad restante del sistema sería suficiente como para poder aceptarlos si la planificación previa se hubiera llevado de otra forma.

Por ello, el objetivo de este trabajo es solventar los problemas relacionados con la fragmentación generada al planificar por adelantado. Para ello se extiende la arquitectura mencionada con el fin de llevar a cabo técnicas de replanificación que alivien los problemas de fragmentación y por consiguiente, que mejoren el uso hecho de los recursos. Dichas técnicas se basan en replanificar tareas en conjunto (Bag of Tasks, BoT) teniendo en cuenta sus requisitos temporales a la hora de seleccionar cuál se planifica primero, en vez de tener sólo en cuenta su entrada al sistema. De este modo se tiene una mayor información sobre los trabajos a planificar, pudiendo llevar a cabo un mejor alojamiento de los trabajos en los recursos. Finalmente, una evaluación del rendimiento se lleva a cabo usando un entorno de pruebas real que incluye recursos heterogéneos compartidos a través de diferentes organizaciones.

La estructura que sigue el artículo es la siguiente: La Sección II hace una revisión de las propuestas existentes para la provisión de QoS en sistemas Grid. En la Sección III se comentan los problemas principales encontrados al llevar a cabo una meta-planificación por adelantado de trabajos. La Sección IV detalla la principal contribución de este artículo, la implementación de la funcionalidad necesaria para llevar a cabo una replanificación de trabajos. Una evaluación del rendimiento de la propuesta es presentada en la Sección V y finalmente la Sección VI muestra las conclusiones y posibles líneas de trabajo futuro.

II. Trabajos Relacionados

El objetivo principal de este trabajo es proveer QoS en entornos Grid. Esto ha sido también investigado en numerosos proyectos basados en reservas por adelantado, como *GARA* [4], *Grid Capacity Planning* [5] o *VIOLA* [6]. Todas estas técnicas tienen el mismo inconveniente: no todos los recursos pueden ser físicamente reservados. Debido a esta limitación nuestro trabajo se basa en planificar el uso de los recursos por adelantado pero sin llevar a cabo reservas físicas de los recursos [3]. Sin embargo, para realizar dicha planificación por adelantado, se necesitan llevar a cabo predicciones sobre el estado futuro de

[1]Instituto de Investigación en Informática de Albacete (I^3A), Universidad de Castilla-La Mancha, e-mail: {luistb,carmen,blanca,}@dsi.uclm.es

los recursos y red, así como sobre la duración de los trabajos. Un estudio sobre varias técnicas de predicción se puede encontrar en [7]. Las técnicas para tales predicciones incluyen el uso de modelos estadísticos teniendo en cuenta las ejecuciones previas [8] y heurísticos basados en características de los trabajos y recursos [9]. En [8], se muestra que aunque la carga exhiba propiedades complejas puede ser predecible teniendo en cuenta su comportamiento pasado. En [9], una evaluación de varios modelos de series lineales de tiempos es usada para la predicción de cargas futuras de la CPU.

Por otro lado, la fragmentación es un conocido problema en el alojamiento de trabajos en los recursos, decrementando el uso que se hace de éstos [10]. Sin embargo, medir la fragmentación presente en un sistema Grid no es una tarea fácil, ya que hay que tener en cuenta la fragmentación en el tiempo y entre recursos [11]. Por ejemplo, en [12] las características de estos problemas de fragmentación en la asignación de memoria son estudiados, sin embargo dicho dominio no encaja con el de los Grid ya que la memoria puede ser considerada como homogénea (1 byte siempre es 1 byte) mientras que esto no es cierto para los recursos en un Grid (X tiempo de computación en un recurso no es igual que en otro).

Por lo que respecta a la replanificación de tareas, un trabajo interesante que investiga como la precisión de la información disponible afecta a la provisión de recursos es [13], donde se usan técnicas de *backfilling* para proporcionar dicha provisión de recursos. Sin embargo, dicho escenario no encaja totalmente con el nuestro, ya que nosotros tenemos en cuenta no sólo posibles deadline establecidos para los trabajos, sino que también se pueden establecer restricciones temporales relacionadas con el tiempo de inicio de dichos trabajos. Es por esto que en nuestro caso el *backfilling* es implícito. Otro trabajo relacionado con la replanificación de trabajos es [14], pero centrado en entornos Grid dedicados, que no es nuestro caso.

III. Meta-planificación por adelantado

Para llevar a cabo las tareas de planificación por adelantado, nuestro sistema necesita tener información acerca de las decisiones de alojamiento de trabajos tomadas previamente. En función de éstas se toman las siguientes decisiones con el objetivo de no solapar ejecuciones. Así pues, se necesitan hacer predicciones sobre el tiempo necesario para ejecutar los trabajos en los recursos. Sin embargo, los recursos pertenecientes a un Grid pueden variar dinámicamente ya que pueden dejar de formar parte de éste, entrar o fallar en cualquier momento. Dicho dinamismo también se ve afectado por el hecho de que los recursos no sólo ejecutan tareas de las aplicaciones pertenecientes al Grid, sino que también de los usuarios locales de dichos recursos. Por lo tanto, todo tiene que ser tenido en cuenta a la hora de hacer dichas predicciones.

El proceso seguido para llevar a cabo una planificación por adelantado se muestra en la Figura 1 y es como sigue [3]:

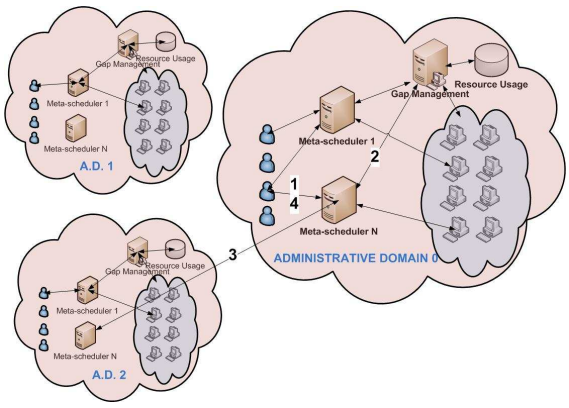

Fig. 1. Proceso de meta-planificación por adelantado.

1) El usuario manda una petición al meta-planificador del dominio local proporcionando la QoS deseada: tiempo de inicio (*start time*) y fin (*deadline*) que se requiere para la aplicación (opcionales).
2) El meta-planificador se comunica con el *Gap Management* y ejecuta un algoritmo de búsqueda de periodos ociosos en los recursos (huecos), obteniendo el recursos y el periodo de tiempo en el que se ejecutará la aplicación, cumpliendo las QoS solicitadas.
3) Si no es posible alojar el trabajo en el dominio local, se establece una comunicación con otros meta-planificadores de otros dominios. Esto se llevaría a cabo mediante técnicas P2P [15], [16].
4) Si aún así no es posible el cumplir con dichas QoS, se establece un proceso de renegociación con el usuario para intentar variar las condiciones solicitadas. Dicho proceso se lleva a cabo mediante Service Level Agreements (SLA) [17].

Para obtener el tiempo que será necesario para completar la ejecución de una aplicación en un determinado recurso se necesitan técnicas de predicción. Éstas calculan los tiempos necesarios para ejecutar una aplicación en base a sus ejecuciones previas y después se reajusta dicho parámetro teniendo en cuenta las predicciones hechas para el estado futuro del recurso en el que será ejecutado, así como el del ancho de banda disponible. Sin embargo, a pesar de hacer este proceso lo más preciso posible, algunos trabajos pueden ser rechazados debido a la fragmentación que se produce en cualquier proceso de alojamiento de tareas. Entre las razones para ello, se encuentran [11]:

- **Fragmentación:** Pequeños huecos libres repartidos a lo largo del tiempo y entre los recursos. Una posible solución a este problema es tratar de compactar dichos huecos para formar otros de mayor tamaño que puedan ser capaces de alojar nuevos trabajos.
- **Decisiones previas desfavorables:** Incluso si la utilización no es demasiado alta puede haber rechazo de trabajo debido a que una decisión previa imposibilita la aceptación de un trabajo que llega con posterioridad al sistema. Esto se debe a que puede usar los únicos huecos que serían factibles para el subsiguiente trabajo. Por tanto, este problema se produce debido a la imposibilidad de saber cuales van a ser las futuras peticiones.

Estos problemas incluso pueden empeorar cuando se tienen trabajos que pueden tener restricciones

temporales de tiempos de inicio y/o finalización.

IV. Replanificación de bolsas de trabajos

Para solucionar dichos problemas, se propone un módulo en nuestro sistema, llamado "*Replanificación de bolsas de trabajos*" o "*Bag of Task Replanning (BoT-R)*", que se ejecuta periódicamente con el objetivo de reducir la fragmentación presente en el sistema. Dicho proceso se realiza sólo en el dominio local con el objetivo de no producir retrasos en las transferencias de red y de hacerlo más escalable. Además, se lleva a cabo por intervalos, puesto que puede haber intervalos que estén lo suficientemente bien planificados como para no necesitar dicha reordenación de trabajos (no existe apenas fragmentación). Por otro lado, los trabajos cuya ejecución no se encuentre entre los límites del intervalo a replanificar, no serán tenidos en cuenta en el proceso de replanificación. De esta forma, no se producirán migraciones de trabajos.

Teniendo en cuenta estos aspectos, el proceso de replanificación se puede dividir en dos etapas:

1) Estimar la necesidad de realizar replanificación de trabajos. En tal caso, se estima sobre qué intervalos de tiempo se tiene que aplicar (Algoritmo 1).

2) Llevar a cabo el proceso de replanificación sobre los intervalos seleccionados, decidiendo qué recursos y trabajos serán tenidos en cuenta (Algoritmo 2).

A pesar de que el Algoritmo 1 se ejecuta cada cierto tiempo, no siempre es necesario lanzar tareas de replanificación (ejecución del Algoritmo 2). Se hace necesario usar una métrica que nos permita medir la fragmentación existente en un sistema Grid [11].

Con este objetivo, el Algoritmo 1 comprueba diferentes parámetros para tomar o no la decisión de replanificación, que conlleva la ejecución del Algoritmo 2. Más específicamente, se chequea que haya un número considerable de trabajos en el sistema (línea 12). En caso negativo, no hay necesidad de replanificación ya que la carga en el sistema no sería significativa. En caso positivo, se comprueban otros parámetros. En primer lugar, se selecciona todo el periodo de tiempo en el que hay trabajos planificados, dividiéndolo en intervalos de *strip* slots. Después, para cada intervalo se comprueba la fragmentación existente, comprobando la carga del sistema y el estado del *Gap Management*. De este modo sólo replanificar los intervalos que realmente necesiten dicho reordenamiento de tareas.

Por lo que respecta a la carga del sistema, se comprueba que el número de trabajos sea de al menos el doble que el de recursos (línea 14). En caso contrario, no sería necesaria una replanificación ya que la carga no sería suficiente como para generar problemas de fragmentación. Por otro lado, también se chequea que el uso de los recursos estimado para ese intervalo se encuentre entre dos umbrales (ru_{low} y ru_{high}), puesto que si no, no habría necesidades de replanificación, ya que o bien habría suficientes slots libres o por contra habría muy pocos (poca fragmentación), haciendo difícil sacar provecho de la replanificación.

Algorithm 1 BoT-R Trigger executed every L period

1: Let St_{fJ} = the first start time of the scheduled jobs
2: Let Et_{lJ} = the latest end time of the scheduled jobs
3: Let P = the period between $[St_{fJ}, Et_{lJ}]$
4: Let P_i = the ith period between $[St_{fJ}, Et_{lJ}]$ in strips of *strip* slots
5: Let Ocu_{P_i} = the percentage of resource occupation into period P_i
6: Let ru_{high} = the maximum resource usage threshold
7: Let ru_{low} = the minimum resource usage threshold
8: Let $\#Gaps_{P_i}$ = number of gaps into period P_i
9: Let $Gaps_{P_i}^{SIZE}$ = the mean size of all gaps into period P_i
10: Let $gapSize$ = the maximum gap size threshold
11: Let $BoT-R_Algorithm(P_i)$ = the BoT-R function over period P_i
12: **if** ($\#Jobs > \#Resources$) **then**
13: **for** each $P_i \in P$ **do**
14: **if** ($\#Jobs > 2 * \#Resources$) and (Ocu_{P_i} in (ru_{low},ru_{high})) and ($\#Gaps_{P_i} > \#Resources$) and ($Gaps_{P_i}^{SIZE} < gapSize$) **then**
15: BoT-R_Algorithm(P_i)
16: **end if**
17: **end for**
18: **end if**

Algorithm 2 BoT-R Algorithm

1: **Input:** start time and end time of the interval to perform replanning (P_i)
2: Let R = set of resources known to GridWay $\{R_1, R_2, \ldots, R_n\}$
3: Let J = set of already scheduled jobs $\{J_1, J_2, \ldots, J_m\}$
4: Let P_i = the period to replan
5: Let ResourceFilter() = function which obtains the resources to defragment
6: Let JobFilter(R) = function which obtains the jobs scheduled to resources R
7: Let JobIntervalFilter(P_i,J) = function which obtains the jobs of J whose full execution is within period P_i
8: Let SortJobsByStartTime(J) = function which obtains a sorted list of the jobs J taking into account its start time information
9: R' = ResourcesFilter()
10: J' = JobFilter(R')
11: $J_{toDefrag}$ = JobIntervalFilter(P_i,J')
12: J_{sorted} = SortJobsByStartTime($J_{toDefrag}$)
13: **for** each $resource_i \in R'$ **do**
14: **for** each $job_j \in J_{sorted}$ **do**
15: **if** job_j can be allocated in $resource_i$ **then**
16: Schedule job_j to $resource_i$
17: Drop job_j of jobs list J_{sorted}
18: **end if**
19: **end for**
20: **end for**

Con arreglo al estado del Gap Management, y suponiendo que se cumplen las condiciones arriba indicadas, el número de huecos y su tamaño medio se calculan. Así, en el caso de que el número de huecos sea alto (mayor que el de recursos) y su tamaño medio sea inferior a un cierto umbral (*gapSize*), el algoritmo de replanificación de tareas (Algoritmo 2) es ejecutado para el intervalo seleccionado (línea 15).

Una vez que el sistema ha estimado qué intervalos necesitan ser replanificados, el Algoritmo 2 se ejecuta para cada uno de ellos. Dicho algoritmo toma como datos de entrada el intervalo a replanificar (definido por el tiempo de inicio y fin del periodo), y sigue el siguiente proceso:

1) Decidir qué recursos estarán envueltos en el proceso de replanificación (línea 9). Para ello, un proceso de filtrado se aplica sobre los recursos para eliminar aquellos que presenten una carga excesivamente alta, ya que no tienen fragmentación aprovechable.

2) En segundo lugar, se eligen los trabajos asociados a los recursos seleccionados en el punto anterior

(línea 10).
3) Después, los trabajos obtenidos en el punto 2 son filtrados también para sólo tener en cuenta aquellos cuya entera ejecución se encuentra dentro de los límites del periodo a replanificar (línea 11).
4) La lista de trabajos resultante se ordena teniendo en cuenta las restricciones de tiempo de inicio de los trabajos (línea 12).
5) Finalmente, para cada recurso incluido en el proceso (línea 13), la lista ordenada de trabajos se recorre en orden (línea 14) tratando de alojar tantos trabajos como sea posible en cada recurso (línea 16), reduciendo el número de huecos entre alojamientos de trabajos contiguos. Cuando no se pueden alojar más trabajos en un recurso, se pasa a intentar alojar los trabajos restantes (también siguiendo el orden) en el siguiente recurso, y así sucesivamente.

El problema de replanificar n trabajos en m máquinas diferentes en las que el tiempo de ejecución de cada trabajo puede variar, no sólo dependiendo de la máquina, sino también dependiendo del tiempo en el que se ejecute, hacen este problema NP-complejo [18]. Por ello, no existe ningún algoritmo polinomial capaz de organizar o alojar los trabajos en los recursos de forma óptima, y menos aún de forma escalable. Por consiguiente, en este trabajo se ha implementado una solución sencilla (Algoritmo 2) que no proporciona una solución perfecta, pero, al tener mayor información sobre los trabajos a planificar, puede llevar a cabo dichas acciones de planificación de una forma más óptima que cuando los trabajos fueron planificados sólo teniendo en cuenta los trabajos previamente alojados.

V. Experimentos y resultados

Esta sección detalla el framework donde las técnicas de replanificación (BoT-R) han sido implementadas, proporcionando una evaluación del rendimiento de dicha propuesta.

A. Framework de meta-planificación

El soporte para llevar a cabo una meta-planificación por adelantado ha sido implementado sobre el meta-planificador GridWay [19], y recibe el nombre de *Scheduler in Advance Layer (SA-Layer)*. Es una capa intermedia entre los usuarios y el meta-planificador, como muestra la Figura 2. El uso de los recursos se divide en intervalos de tiempo, llamados slots, y SA-Layer tiene que planificar el uso de esos recursos. Con dicho fin, se hace uso de una estructura de datos (representada por *Data Structure* en la Figura 2) para mantener una traza del uso de los recursos. En este trabajo se ha elegido como estructura de datos los *árboles rojo-negro* [20] con el objetivo de poder desarrollar técnicas que eficientemente puedan identificar los periodos disponibles para las nuevas peticiones de ejecución de trabajos, y sin tener que examinar todos los existentes.

El módulo *Gap Management* maneja esta estructura de datos y se encarga de mantenerla. Para hacer las búsquedas en dicha estructura de datos se representa la información de una manera geométrica (para

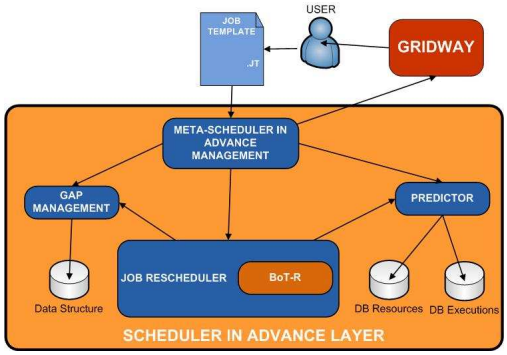

Fig. 2. Scheduler in Advance Layer (SA-layer).

más información sobre este proceso ver [3], [20]).

Debido al desconocimiento de la duración que tendrán los trabajos en los diferente recursos, se necesitan desarrollar algoritmos que estimen cuantos *slots* son necesarios para completar dichas ejecuciones (*Predictor* en la Figura 2). Con este objetivo, el tiempo total necesario para la ejecución de un trabajo en un recurso se estima teniendo en cuenta las características de los trabajos, la potencia y uso de las CPUs de los recursos y el estado de la red.

Dichas predicciones sobre tiempos de ejecución son muy difíciles de obtener con precisión debido a que hay deferencias en el rendimiento de los recurso dentro de un Grid, no sólo entre ellos, sino también durante el tiempo. Por esto, SA-Layer obtiene las predicciones para los tiempos de ejecución separadas de las que se hacen para las estimaciones sobre el tiempo necesario para completar las transferencias por la red [3]. Además, se usan funciones de suavizado exponencial para obtener predicciones sobre el estado futuro de la red y de los recursos [21].

Por lo que respecta a los tiempos de ejecución, se calcula como el tiempo medio de ejecución usado en ejecuciones previas de la aplicación en el recurso objetivo. Después de esto, mediante una función de suavizado exponencial se calcula el estado futuro de la CPU para el recurso. Con esta información se reajusta el tiempo para tener en cuenta el estado futuro del recurso. Finalmente, la información es otra vez ajustada teniendo en cuenta cómo de predecible es últimamente el recurso en el que se ejecutará el trabajo.

Respecto a las transferencias, primero se calcula el ancho de banda que habrá disponible durante el intervalo en el que el trabajo se tiene que ejecutar (función de suavizado exponencial). Con esta información y la del número total de bytes a transferir se calcula el tiempo total que se necesitará para completar las transferencias.

B. Descripción del entorno de pruebas

La evaluación del rendimiento se ha realizado en un Grid real formado por recursos pertenecientes a dos universidades españolas. En la Universidad de Castilla–La Mancha (UCLM) hay diferentes recursos localizados en dos edificios. En el *Instituto de Investigación en Informática de Albacete* (I3A), está, por un lado, la máquina que realiza las tareas de planificación, y por otro, varios recursos computacionales

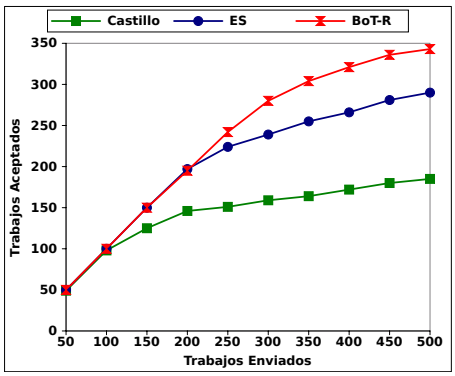

Fig. 3. Ratio de trabajos planificados para el workload 1.

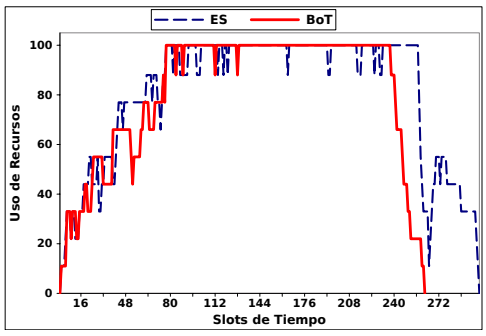

Fig. 4. Uso de recursos sobre el tiempo.

(11 ordenadores pertenecientes a distintos usuarios). En otro edificio, llamado *Escuela Superior de Ingeniería Informática* (ESII), hay un clúster con 88 cores y PBS como sistema de planificación, también compartido con otros usuarios. Por otro lado, hay otro recurso computacional localizado en la *Universidad Nacional de Educación a Distancia*, UNED) de Madrid. Por lo tanto, la red que une dichos recursos es *Internet*. Es importante tener en cuenta que las máquinas pertenecen a otros usuarios, así que tienen su propia carga de fondo (tanto de CPU como de red), haciendo el entorno incluso más realista.

C. Carga de trabajo usada

Como carga de trabajo para testear el funcionamiento de las técnicas de replanificación se ha usado uno de los test presentes en los benchmarks GRASP [22], llamado *3node*. Este test consiste en enviar un fichero desde un nodo fuente a otro de computación, el cual lleva a cabo una búsqueda de patrones, generando un fichero de salida con el número de aciertos. Finalmente, el fichero de salida se manda a otro nodo.

La ventaja que nos proporciona esta prueba es que acepta parámetros para hacerla más o menos intensiva en computación, así como más o menos demandante de red. Así pues, modificando dichos parámetros se puede crear una carga de trabajo muy versátil generando trabajos de muy distinto tipo y con diferentes requerimientos y/o comportamientos. En esta evaluación, el parámetro *compute_scale* toma valores entre 0 y 20, mientras que *output_scale* lo hace entre 0 y 2. Así, el fichero de entrada es de 100 MB y se crean ficheros de salida de hasta 200 MB.

D. Evaluación del rendimiento

Se comparan diferentes implementaciones de SA-Layer con una sencilla implementación del algoritmo presentado por Castillo [20]. Dicho algoritmo está basado en una función lineal para calcular el tiempo necesario para completar la ejecución de un trabajo en un recurso. Sin embargo, esta función lineal no tiene en cuenta las diferencias de rendimiento que se pueden dar en un recurso a lo largo del tiempo.

Las implementaciones de SA-Layer comparadas aquí son: (1) una implementación con la replanificación de bolsas de trabajos (etiquetada como `BoT-R`) y (2) una implementación previa del SA-Layer presentada en [21] (etiquetada como `ES`), que no proporciona ninguna herramienta para la replanificación de trabajos. Ambas técnicas usan funciones de suavizado exponencial para estimar mejor los tiempos necesarios para la ejecución de los trabajos en los recursos.

Para evaluar el rendimiento de dichas técnicas se ha usado el ratio de trabajos planificados (los que han sido aceptados para ser ejecutados) y el uso de los recurso a lo largo del tiempo.

Por lo que respecta a la carga de trabajo, se han generado dos tipos diferentes usando la prueba descrita en la Sección V-C. Primero, se ha generado una carga de trabajo enviando trabajos de tipo 3node uno tras otro (llamada Workload 1), es decir, cuando un trabajo se envía y el sistema lo ha aceptado o rechazado, entonces se envía el siguiente. El objetivo de esta prueba es demostrar que usando BoT-R se mejora el uso de los recursos gracias a una reducción en la fragmentación generada, y como resultado de ello, un mayor número de trabajos son aceptados. Para ello la Figura 3 muestra el ratio de trabajos planificados para el Workload 1, representando la relación entre trabajos enviados (eje x) y trabajos aceptados (eje y). Dicha gráfica muestra una clara mejora con el uso de la replanificación (BoT-R), ya que se acepta un mayor número de trabajos al existir un mayor número de *slots* libres juntos. `BoT-R` obtiene un incremento en el número de trabajos aceptados de más de un 85 % comparado con `Castillo` y de casi un 20 % sobre `ES`, demostrando la eficiencia de hacer dichas replanificaciones en forma de BoTs.

En la Figura 4 se representa el porcentaje de uso de los recursos durante el tiempo que dura la ejecución de Workload 2, donde los trabajos se envían continuamente pero con un ratio de 4 trabajos por minuto. Por claridad, las estadísticas para `Castillo` no están representadas puesto que el número de tareas aceptadas por esta técnicas es muy inferior a los obtenidas por SA-Layer (con o sin replanificación). Dicha figura resalta que teniendo menos fragmentación, el uso de los recursos también mejora. De hecho, el intervalo de tiempo durante el cual los recursos están siendo usados al 100 % se incrementa, a la vez que el tiempo necesario para acabar todas los trabajos pertenecientes a Workload 2 se reduce. Además, se observa un comportamiento más estable (menos fluctuación en el porcentaje de uso de los recursos) cuando se usa la técnica de replanificación.

VI. Conclusiones y trabajo futuro

Existen diversas investigaciones científicas cuyo objetivo principal es proporcionar QoS en entornos Grid realizando reservas físicas en los recursos. Sin embargo, dichas reservas no son siempre posibles en este tipo de entornos. Por ello, se propone la realización de una metaplanificación por adelantado como una posible solución. Ambos tipos de planificación conllevan diversos retos, como el desarrollo de algoritmos eficientes y escalables, el estudio y aplicación de técnicas de predicción del tiempo necesario para acabar la ejecución de una aplicación en un determinado recursos, así como el estado futuro de éste. Además, se necesitan técnicas que permitan evitar una pobre utilización de los recursos debido a la fragmentación generada en el proceso de planificación.

Este artículo propone una extensión al framework presentado en [21] para poder abordar los problemas de fragmentación que pueden conllevar una baja utilización de los recursos disponibles. Esta nueva característica del sistema replanifica tareas ya planificadas con el objetivo de reducir la fragmentación haciendo un uso más eficiente de los recursos. De esta forma, un grupo de trabajos son replanificados teniendo en cuenta sus requerimientos de tiempo de inicio en vez de su llegada al sistema (que es como fueron previamente planificados). Al tener más información sobre los trabajos que hay que planificar (como una bolsa de trabajos), este proceso puede llevarse a cabo más eficientemente.

Aparte de presentar el framework mejorado, se hace una comparación entre usar o no dicha estrategia de replanificación. De esta forma, se ensalza la importancia de replanificar los trabajos teniendo mayor información sobre ellos para de esta forma aceptar un mayor número de peticiones y por tanto incrementando la QoS proporcionada a los usuarios.

Como trabajo futuro es interesante es el desarrollo de técnicas que permitan hacer una mejor predicción de los tiempos necesarios para completar las transferencias. Para ello, una posible opción es tratar de reservar ancho de banda dónde y cuándo sea posible. También puede resultar muy ventajoso tratar de planificar los datos como si fueran otro recurso así como, comparar el framework propuesto con otro que lleve a cabo reservas físicas por adelantado de los recursos.

Agradecimientos

Este trabajo ha sido apoyado conjuntamente por el MEC y MICINN Español y la Comisión Europea (fondos FEDER) a través de los proyectos "Consolider Ingenio-2010 CSD2006-00046" y "TIN2009-14475-C04". Conjuntamente con la JCCM a través del proyecto "PII1C09-0101-9476".

Referencias

[1] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2 edition, 2003.

[2] L.-O. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert. The virtual resource manager: an architecture for SLA-aware resource management. *Proc. of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2004.

[3] Luis Tomás, Agustín C. Caminero, Carmen Carrión, and Blanca Caminero. Network-aware meta-scheduling in advance with autonomous self-tuning system. *Future Generation Computer Systems*, 27(5):486 – 497, 2011.

[4] Alain Roy and Volker Sander. *Grid Resource Management*, chapter GARA: A Uniform Quality of Service Architecture, pages 377–394. Kluwer Academic Publishers, 2003.

[5] Mumtaz Siddiqui, Alex Villazón, and Thomas Fahringer. Grid capacity planning with negotiation-based advance reservation for optimized QoS. In *Proc. of the Conference on Supercomputing (SC)*, Tampa, USA, 2006.

[6] O. Waldrich, Ph. Wieder, and W. Ziegler. A meta-scheduling service for co-allocating arbitrary types of resources. In *Proc. of the 6th Intl. Conference on Parallel Processing and Applied Mathematics (PPAM)*, Poznan, Poland, 2005.

[7] Menno Dobber, Rob van der Mei, and Ger Koole. A prediction method for job runtimes on shared processors: Survey, statistical analysis and new avenues. *Performance Evaluation*, 64(7-8):755–781, 2007.

[8] Peter A. Dinda. The statistical properties of host load. *Scientific Programming*, 7(3-4):211–229, 1999.

[9] Hai Jin, Xuanhua Shi, Weizhong Qiang, and Deqing Zou. An adaptive meta-scheduler for data-intensive applications. *Intl. Journal of Grid and Utility Computing*, 1(1):32–37, 2005.

[10] W Smith, Ian Foster, and V Taylor. Scheduling with advanced reservations. In *Proc. of the 14th Intl. Parallel and Distributed Processing Symposium (IPDPS)*, Washington, USA, 2000.

[11] Julius Gehr and Jorg Schneider. Measuring Fragmentation of Two-Dimensional Resources Applied to Advance Reservation Grid Scheduling. In *Proc. of the 9th Intl. Symposium on Cluster Computing and the Grid (CCGRID)*, Shanghai, China, 2009.

[12] Paul R. Wilson, Mark S. Johnstone, Michael Neely, and David Boles. Dynamic storage allocation: A survey and critical review. In *Proc. of the Intl. Workshop on Memory Managment (IWMM)*, Kinross, UK, 1995.

[13] Marcos Dias De Assunção and Rajkumar Buyya. Performance analysis of multiple site resource provisioning: effects of the precision of availability information. In *Proc. of the 15th Intl. Conference on High Performance Computing (HiPC)*, Bangalore, India, 2008.

[14] Yves Caniou, Ghislain Charrier, and Frederic Desprez. Analysis of Tasks Reallocation in a Dedicated Grid Environment. In *Proc. of the Intl. Conference on Cluster Computing (CLUSTER)*, Heraklion, Greece, 2010.

[15] Agustín Caminero, Omer Rana, Blanca Caminero, and Carmen Carrión. Network-aware heuristics for inter-domain meta-scheduling in Grids. *Journal of Computing and System Sciences*, 77(2):262 – 281, 2011.

[16] Marcos Dias de Assunção, Rajkumar Buyya, and Srikumar Venugopal. InterGrid: A case for internetworking islands of Grids. *Concurrency and Computation: Practice and Experience*, 20(8):997–1024, 2008.

[17] Javier Conejero, Luis Tomás, Carmen Carrión, and Blanca Caminero. QoS Provisioning with Meta-Scheduling in Advance within SLA-based Grid Environments. *Computing and Informatics*, 31:73 – 88, 2012.

[18] J. Bank and F. Werner. Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. *Mathematical and Computer Modelling*, 33(4-5):363 – 383, 2001.

[19] E. Huedo, R. S. Montero, and I. M. Llorente. A modular meta-scheduling architecture for interfacing with pre-WS and WS Grid resource management services. *Future Generation Computing Systems*, 23(2):252–261, 2007.

[20] Claris Castillo, George N. Rouskas, and Khaled Harfoush. Efficient resource management using advance reservations for heterogeneous Grids. In *Proc. of the Intl. Parallel and Distributed Processing Symposium (IPDPS)*, Miami, USA, 2008.

[21] Luis Tomás, Agustín Caminero, Carmen Carrión, and Blanca Caminero. Exponential Smoothing for Network-aware Meta-scheduler in Advance in Grids. In *Proc. of the 6th Intl. Workshop on Scheduling and Resource Management on Parallel and Distributed Systems (SRMPDS)*, San Diego, USA, 2010.

[22] G. Chun, H. Dail, H. Casanova, and A. Snavely. Benchmark probes for Grid assessment. In *Proc. of 18th Intl. Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, New Mexico, 2004.