Servicios de Asignación y Planificación de Recursos Grid

Allocation and Scheduling Services of Grid Resources

Álvaro Fernández[1], Elisa Heymann[2], José Salt[1], Miquel A. Senar[2]

[1]Instituto de Física Corpuscular
Valencia, España
{Alvaro.Fernandez, Jose.Salt}@ific.uv.es
[2]Unitat d'Arquitectura d'Ordinadors i Sistemes Operatius
Universitat Autònoma de Barcelona
Barcelona, España
{elisa.heymann, miquelangel.senar}@uab.es

Resumen

El concepto de Grid ha surgido en los últimos años para denominar un conjunto de recursos computacionales heterogéneos distribuidos pertenecientes a distintas organizaciones. El principal objetivo del proyecto europeo *CrossGrid* en el cual están involucrados el CSIC , la UAB y RedIris, entre otros, es poder ejecutar aplicaciones distribuidas interactivas en un entorno grid. Este artículo presenta las técnicas y servicios que estamos estudiando y desarrollando dentro de este proyecto para conseguir una eficiente utilización de los recursos grid para este tipo de aplicaciones. Cuando se lance una aplicación se utilizarán estos servicios para automáticamente localizar y seleccionar los recursos donde se va a ejecutar, hacer una reserva de los mismos si es necesario, lanzar la aplicación y monitorizar su ejecución hasta que finalmente termine.

Abstract

The Grid concept has arisen in the last years to call a heterogeneous group of distributed computational resources that belong to different organizations. The main objective of the European project *CrossGrid*, in which CSIC, UAB and RedIris participate, among others, is being able to execute interactive distributed applications in a grid environment. This paper presents the techniques and services that we are studying and developing within this project to achieve an efficient usage of the grid resources for this kind of applications. When a job is submitted, these services will be used to search and allocate the resources in an automatic manner, make a reservation for them, if necessary, launch the application and monitor its execution until it finally completes.

1. INTRODUCCIÓN

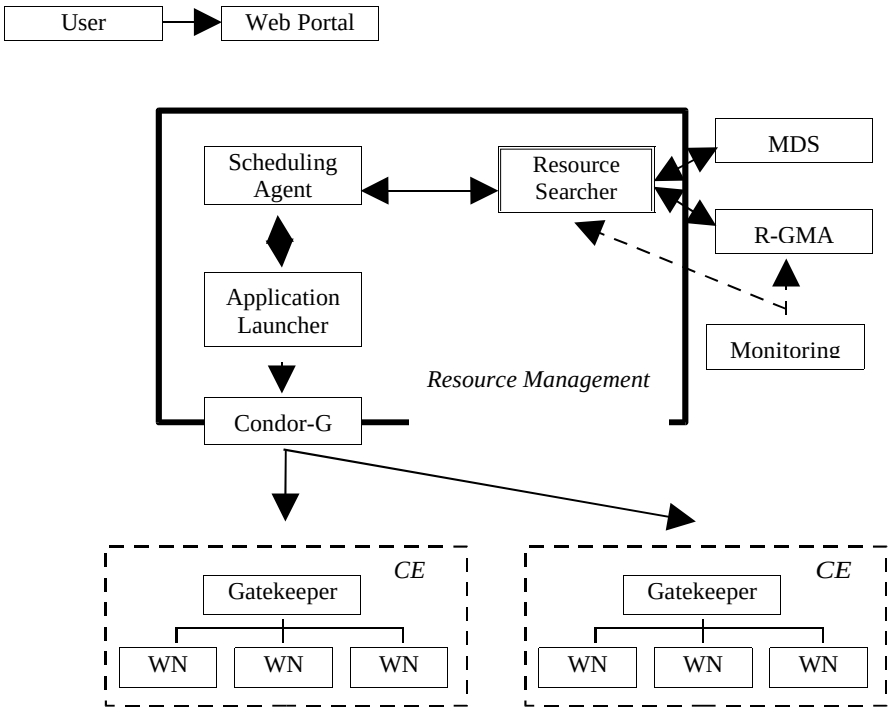
Dentro de la problemática de la computación Grid, uno de los aspectos de mayor importancia y en el que se está investigando más activamente, es el de la gestión de los recursos que forman parte de esta infraestructura de computación distribuida.

Estos recursos pueden ser de varios tipos, distinguiendo principalmente aquellos que forman parte de la infraestructura misma como pueden ser recursos *Hardware* (cpus, memoria, almacenamiento secundario), recursos *Software* (sistema operativo, paquetes específicos de software), y recursos de *Red* (disponibilidad de un ancho de banda determinado, calidad del servicio). Además se pueden identificar otros recursos como requerimientos de *Tiempo*, durante el cual deben estar disponibles los recursos anteriores, o recursos como *Datos* necesarios para una computación dada.
La dificultad del acceso eficaz a los recursos computacionales y de red incluye muchos de los problemas intrínsecos de la naturaleza de los sistemas computacionales distribuidos, con algunos añadidos. Al ser un sistema dinámico los recursos disponibles pueden cambiar con el tiempo, con lo que se debe tener un sistema para descubrir y contar con estos recursos disponibles. Se pueden añadir nuevos nodos de cómputo que aporten cpus y memoria, y nodos de almacenamiento que supongan otros lugares para almacenar datos; y también pueden desaparecer, dejando de estar disponibles éstos y el posible software que pudieran albergar. La utilización de los mismos, así como la de las redes que los intercomunican puede variar con el tiempo, haciendo difícil que se pueda producir una utilización ideal de estos recursos.
Por ello es necesario que existan unos servicios que tengan en cuenta todos estos parámetros y que facilite la utilización del sistema de una manera eficaz para que todos los usuarios puedan ejecutar sus trabajos, sean cuales sean sus requerimientos.

2. SERVICIOS PARA ASIGNACIÓN Y PLANIFICACIÓN DE RECURSOS

Para hacer más fácil la utilización de estas infraestructuras distribuidas que forman el Grid, surge la necesidad de establecer una arquitectura global que sea plasmada en la práctica en una serie de servicios básicos en forma de *middleware*, y que simplificarán el modo de desarrollar aplicaciones que puedan hacer uso de estas infraestructuras.
En particular el proyecto Crossgrid está orientado principalmente a aplicaciones paralelas e interactivas, por lo que se requiere que el control sobre los recursos y la planificación de los distintos trabajos de estas aplicaciones en concreto sea eficiente, automático y de una manera consistente.
Las tareas relacionadas con la planificación y la selección eficiente de recursos están basadas en un *middleware* proporcionado por el proyecto *Datagrid* , y cuya entidad principal es el *Resource Broker* (RB), que proporcionará los servicios necesarios para la ejecución automática de trabajos secuenciales.
Mediante estos servicios un usuario es capaz de enviar un trabajo junto con una descripción de los requerimientos del mismo, y el RB se encargará de buscar y seleccionar los recursos necesarios.
Cuando se hayan localizado los recursos se pasará a la siguiente etapa que consiste en mandar el trabajo para su ejecución utilizando aquellos seleccionados, tomando las precauciones necesarias y reenviando el trabajo si algo fallara. Continuamente se monitorizará el estado del trabajo para ver que el trabajo se está ejecutando correctamente, reenviando el trabajo a estos u otros recursos si algo fallara. Se obtendrá finalmente la salida del mismo para reenviarla al usuario, haciendo la ejecución totalmente transparente para éste.
Aunque nuestro proyecto se beneficia de estos servicios, extensiones a éstos y nuevos componentes específicos son necesarios para el control de aplicaciones paralelas interactivas escritas en MPI, y también para solventar algunas deficiencias del sistema original, dando lugar a los *Scheduling Agents* (SA) , que constituyen una extensión de la funcionalidad proporcionada por el RB.

En los siguientes puntos comentaremos los servicios relacionados más específicamente con la gestión y planificación de recursos, así como los cambios más importantes en estos componentes para acomodar las necesidades de las aplicaciones interactivas.



2.1. SELECCIÓN DE RECURSOS

Una parte importante dentro del sistema de planificación de recursos es poder averiguar cuáles están disponibles en un momento determinado y hacer una selección de los mismos para ejecutar la aplicación. Cuando el usuario envía un trabajo junto con una descripción del mismo utilizando el lenguaje JDL , éste será transformado por el SA a un *classad* que contiene los requerimientos y preferencias del trabajo. Los recursos también tienen asociados *classads* definiendo lo que cada recurso provee y que están publicados en un sistema de información, generalmente MDS o R-GMA.

En el caso de la arquitectura propuesta en Crossgrid .se adoptan los *nodos de computa*ción (*Computing Elements* o *CE)* que son la representación de una granja local compuesta por diferentes *nodos trabajadores* (*Working nodes* o *WN),* y que son el punto de acceso a la misma. Adicionalmente existen *nodos de almacenamiento* (*Storage Elements* o *SE)* desde donde se accederán los datos.

En la primera fase de la selección el *Resource Selector* consultar qué recursos cumplen los requerimientos, parciales o totales, que el trabajo está imponiendo. En el caso de que mandemos un trabajo secuencial éste se deberá ejecutar en un solo CE, ya que únicamente requerirá el uso de una cpu que se corresponderá finalmente con un WN de ese CE. En este caso se construirá una lista de posibles CEs que cumplen todos los requerimientos, ordenados según alguna función de preferencia (número de cpus del CE libres, potencia de cálculo, memoria libre, etcétera) que el usuario puede establecer.

En el caso de un trabajo MPI se necesitarán varias cpus para correr el trabajo, que serán seleccionadas en un único CE o en un grupo de ellos, con el siguiente criterio:

1. Primero se intentan seleccionar los CEs que tienen todas las cpus requeridas libres, con lo que se intenta correr primero en un solo cluster para evitar las latencias de los mensajes entre diferentes clusters.
2. Después se forman grupos de CEs que cumplan los requisitos necesarios para cada uno de ellos, y que conjuntamente reúnan el número de cpus libres requeridas.
3. La ordenación entre los diferentes grupos se hace de menor a mayor número de CEs diferentes, y dentro de los grupos con el mismo número de elementos, se utilizará la función de preferencia ponderada con el número de componentes.

2.2. **PLANIFICACIÓN Y RESERVAS TEMPORALES**

Parte de la planificación se ha hecho con el servicio anterior de selección de recursos, ordenando los recursos a utilizar con una serie de criterios que acabamos de mostrar. En la siguiente fase se enviará el trabajo al primer CE o grupo de CEs de la lista en el caso de trabajos MPI. Debido a la naturaleza en la que se obtienen los datos sobre los CEs y sus cpus libres por ejemplo, que no es completamente actualizada en tiempo real, puede darse el caso de que se produzcan casos en los que el RB/SA considere alguna cpu como libre cuando realmente no lo está porque se acaba de mandar un trabajo. Este es sólo un ejemplo de los casos en los que puede fallar el intento de ejecución de un trabajo, por lo cual es necesario un método para poder reenviarlo a la siguiente selección de la lista.
Además se pueden dar posibles casos de interbloqueos entre diferentes trabajos mandados que requieran varias cpus, cuando un trabajo se está esperando un WN que está ocupado por un segundo trabajo, y éste está esperando un WN que está ocupado por el primero. Esto se soluciona con un método de reservas locales temporal que afecta al *Resource Selector*. Cuando se decide a qué grupo de CEs enviar el trabajo, se establece una reserva por un espacio de tiempo, de manera que estos recursos no serán tenidos en cuenta por el siguiente trabajo que busque recursos.
Estas reservas son locales al SA y no es un sistema general de reservas, sino que simplemente modifica los servicios locales de selección de recursos. Cuando se ha superado cierto intervalo, o el trabajo ha acabado o fallado en esos recursos, la reserva es retirada y esos recursos pueden volver a ser utilizados.

2.3. PRIORIDADES

Las aplicaciones interactivas requieren un tiempo de respuesta corto para que el usuario pueda estar satisfecho con la ejecución de la misma. Para permitir que este tipo de aplicaciones puedan ejecutarse anteriormente a otras, por ejemplo aplicaciones en *batch*, introducimos el concepto de prioridades. Con las prioridades nos es posible determinar qué trabajos son necesarios ejecutar anteriormente, de manera que si tenemos varios trabajos pendientes de ejecución, podremos determinar el orden en que se van a mandar.
Además en el caso de que todos los recursos estén ocupados por la ejecución de trabajos, y nos llegue uno más prioritario, será posible determinar qué trabajos pueden ser parados momentáneamente para permitir la ejecución del nuevo más prioritario

2.4. ENVÍO DEL TRABAJO

Cuando tenemos todas las decisiones hechas sobre los recursos a los que mandar el trabajo, y las distintas prioridades sobre otros, se deben realizar los pasos adecuados

para permitir la ejecución del mismo. Dependiendo del tipo de trabajo, si es en *batch* o interactivo, si es secuencial o paralelo escrito en las diferentes variantes de MPI soportadas, se realizaran diferentes aproximaciones para conseguir el objetivo. El *Application Launcher* envía los ejecutables de forma confiable a los distintos recursos seleccionados implicados, y empezar la ejecución. Utilizaremos Condor-G y ampliaciones al mismo para llevar a cabo estas tareas.

3. CONCLUSIONES Y TRABAJO FUTURO

Hemos presentado nuestra aproximación y los servicios desarrollados relacionados con la gestión de recursos Grid en un proyecto como CrossGrid en el que estamos participando. Con estos servicios damos soporte tanto a trabajos secuenciales como trabajos paralelos MPI sobre uno o varios clusters.
El Scheduling Agent supone para el usuario el punto de acceso al grid, y reúne todos los servicios relatados para la gestión de los recursos grid y que los trabajos finalmente se ejecuten de manera transparente al usuario, además de correcta y eficientemente.
En la actualidad las ideas planteadas por los *Scheduling Agents*, se han implementado sobre el Resource Broker directamente dando una funcionalidad añadida. Sin embargo otro de nuestros objetivos es solucionar la falta de escalabilidad que presenta el RB cuando el número de trabajos y de usuarios es alto. Al ser un servicio centralizado plantea un cuello de botella cuando la utilización es alta. Adicionalmente un fallo en el RB afectaría a todos los trabajos que están siendo ejecutados, además de no poder atender nuevas peticiones. En este sentido estamos descentralizando los servicios y haciendo el sistema más modular y ligero, para que un fallo en un componente no afecte al resto y que el sistema pueda seguir funcionando dentro de lo posible hasta que se reinicie el componente fallido.
Además estamos ampliando el sistema de planificación de recursos utilizando, a parte de datos estáticos sobre los recursos, información dinámica como el rendimiento, carga y disponibilidad, localización de datos y disponibilidad de réplicas, y el estado de la red. Por otra parte se utilizará información dinámica sobre cada aplicación a planificar, obtenida en tiempo de ejecución mediante los servicios de monitorización desarrollados en otras partes del proyecto, que ayudará además a predecir requerimientos en el futuro inmediato, y a mejorar en las acciones tomadas en cuanto a la planificación.

REFERENCIAS

[1] European CrossGrid Project, http://www.crossgrid.org

[2] F. Giacomini, F. Prelz. “Definition of architecture, technical plan and evaluation criteria for scheduling, resource managements, security and job description”, http://server11.infn.it/workload-grid/docs/DataGrid-01-D1.2-0112-0-3.pdf

[3] M.A.Senar, M. Sottilaro, S. Becco, “Scheduling Agent Design Document”, CG3.2-D3.2-v1.0-UAB020-SchedulingAgentsDesign.doc, Crossgrid Internal document

[4] A. Fernandez, M.A.Senar, M. Sottilaro, E. Heymann, J. Bergés, “WP3.2 Prototipe Description”, http://gridportal.fzk.de/distribution/crossgrid/crossgrid/wp3/wp3_2-scheduling/docs/CG3.2-D3.3-v1.1-UAB010-PrototypeDescription.pdf

[5] F. Pacini, “Job Description Language Howto”, http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.pdf

[6] E. Heymann, M.A.Senar, A. Fernandez, J. Salt, “The Eu-Crossgrid approach for Grid Application Scheduling”, 1st European Across Grids Conference February, 13th-14th, 2003.

[7] Rajesh Raman, Miron Livny, Marvin Solomon, “Matchmaking: Distributed Resource Management for High Throughput Computing”, Proc of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 1998

[8] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, Steven Tuecke, “CondorG: A Computation Management Agent for Multi-Institutional Grids”, Journal of Cluster Computing, vol. 5, páginas 237-246, 2002