Scheduling en el paradigma Grid

Javier Echaiz Jorge Ardenghi

Laboratorio de Investigación de Sistemas Distribuidos (LISiDi)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca (8000), Argentina
{je,jra}@cs.uns.edu.ar

Resumen

Un sistema grid conforma una red dinámica de recursos de cómputo heterogéneos que trabajan conjuntamente formando un entorno único y uniforme. Un sistema grid puede abarcar diferentes dominios administrativos, y puede ser capaz de soportar cambios dinámicos en las organizaciones participantes y en los requerimientos de cómputo [5].

Para lograr este objetivo, un sistema grid debe ser capaz de proveer acceso a los recursos que los usuarios requieran, incluyendo procesamiento, datos, y aplicaciones. Este acceso debe estar respaldado por una infraestructura de seguridad a nivel del grid, pero que al mismo tiempo permita diversas y cambiantes políticas de seguridad locales. Además, el sistema debe ser diseñado con la capacidad de tolerar fallas a nivel de nodos individuales y cambios en la composición del grid (hardware, configuración de los sistemas y potencialmente sistema operativo). Por último es importante que un sistema con estas características sea simple de instalar y administrar.

Para lograr este prometedor potencial es esencial contar con algoritmos de planificación (*scheduling*) eficientes y efectivos. Desafortunadamente, los algoritmos tradicionales de scheduling empleados en sistemas paralelos y distribuidos, los cuales suelen correr sobre recursos homogéneos y dedicados, e.g., clusters de computadoras, no funcionan bien bajo el paradigma grid [2]. En este trabajo exploraremos algunos de los problemas abiertos relacionado con el problema del scheduling distribuido en sistemas grid.

Palabras Clave: scheduling, grid, gestión de recursos.

1. Introducción

Un grid de cómputo es una infraestructura de hardware y software que provee acceso confiable, consistente, extensivo y barato a sistemas de cómputo con gran capacidad de procesamiento [12]. Un grid es un entorno compartido implementado mediante una infraestructura de servicios permanente y basada en estándares capaz de soportar, crear y compartir recursos. Los recursos pueden ser computadoras, espacio de almacenamiento, instrumentos, aplicaciones (software), y datos, todos ellos conectados mediante una red (típicamente Internet) y una capa de software (*middleware*), la cual provee servicios de seguridad, monitoreo, gestión de recursos, etc. Los recursos pertenecen potencialmente a diferentes organizaciones y por lo tanto son compartidas bajo políticas que definen qué se comparte, quién puede acceder y bajo qué condiciones puede hacerlo [11]. El problema real y específico del concepto grid es la coordinación de los recursos compartidos y cómo resolver problemas bajo una organización virtual multi-institucional [13].

Desde el punto de vista de los sistemas de scheduling, es posible aplicar un mayor nivel de abstracción para grid, ignorando algunos componentes estructurales, como la autenticación, autorización, control de acceso, y descubrimiento de recursos. En este trabajo se adoptará la

siguiente definición de grid: *"un grid es un tipo de sistema paralelo y distribuido que posibilita compartir recursos geográficamente distribuidos dinámicamente, dependiendo de su disponibilidad, capacidad, performance, costo, y requerimientos de calidad de servicio (QoS) especificados por el usuario"* [1].

Desde el punto de vista de la funcionalidad es posible encontrar una arquitectura lógica del subsistema de scheduling de tareas en grid. Por ejemplo, Zhu [24] propone una arquitectura común de grid scheduling. Además, es posible generalizar un proceso de scheduling bajo grid en tres etapas: descubrimiento y filtrado de recursos, selección y scheduling de recursos según objetivos específicos, y envío de tareas *job submission* [17]. En este trabajo trataremos únicamente el segundo paso.

2. El problema del scheduling en el paradigma grid

A continuación describiremos brevemente los componentes funcionales de un sistema de scheduling bajo el paradigma grid.

Básicamente un Grid scheduler (GS) recibe aplicaciones provenientes de usuarios grid, seleccione los recursos necesarios para estas aplicaciones según la información proveniente del Grid Information Service module, y finalmente genera mapeos de aplicaciones a recursos, basados en los objetivos especificados y la predicción de la performance de los recursos. A diferencia de sus contrapartidas en los sistemas distribuidos y paralelos tradicionales, los schedulers grid no pueden controlar los recursos en forma directa, sino que funcionan como *brokers* o agentes [3], o fuertemente acoplados a las aplicaciones, e.g., scheduling a nivel de aplicación [4, 22]. No están necesariamente ubicados en el mismo dominio que los recursos que administran. De hecho, se puede contar con múltiples schedulers organizados para formar diferentes estructuras (centralizas, jerárquicas, y descentralizadas [16]) según diferentes necesidades, como performance o escalabilidad. Aunque un scheduler a nivel grid (o *metascheduler*, como se lo suele llamar en la literatura, e.g., en [15]) no es un componente fundamental en la infraestructura grid (e.g., no se incluye en Globus Toolkit [14], el estándar de facto), no hay dudas que un componente de scheduling es crucial para poder explotar el potencial del paradigma, incorporando recursos provenientes tanto de supercomputadoras como de *desktops*.

Un scheduler de grid necesita información acerca del estado de los recursos disponibles para poder lograr una planificación apropiada, especialmente cuando se toma en cuenta la naturaleza dinámica y heterogénea del grid. Para obtener dicha información se emplea un *Grid Information Service* (GIS), responsable de recolectar y predecir la información de estado, como capacidad de CPU, tamaño de la memoria, ancho de banda de red, disponibilidad de software y carga de un nodo en un dado período particular. GIS puede responder pedidos de información sobre recursos o enviar información a subscriptores. Posiblemente el ejemplo más conocido de GIS sea el *Monitoring and Discovery System* (MDS) [9] de Globus.

Para lograr un scheduling razonable, además de la información "cruda" del GIS, son necesarias las propiedades de la aplicación (por ejemplo requerimientos de memoria y almacenamiento, dependencia de subtareas en un job y volumen de comunicaciones, cantidad aproximada de instrucciones, etc.), y performance de un recurso para diferentes clases de aplicaciones. Para extraer propiedades de las aplicaciones se emplea *Application Profiling* (AP), mientras que el *Analogic Benchmarking* (AB) se utiliza para medir que tan bien puede un recurso desempeñarse en relación a un dado tipo de tarea [18, 21]. A partir del conocimiento obtenido a partir del AP y del AB, y siguiendo cierto modelo de performance [2], la estimación del costo computa el costo de las planificaciones candidatas, de entre las cuales elegirá aquella que pueda maximizar

las funciones de objetivo.

El módulo de *Launching and Monitoring* (LM) (también conocido como *binder* [7]) implementa una determinada planificación enviando las aplicaciones hacia los recursos seleccionados, enviando datos de entrada y ejecutables si fuese necesario, y monitoreando la ejecución de las aplicaciones. Un buen ejemplo de un LP es el *Grid Resource Allocation and Management* (GRAM) de Globus [8].

El *Local Resource Manager* (LRM) completa el cuadro, es el responsable principal de dos tareas: (1) scheduling local dentro de un dominio de recursos, donde además de los jobs de los usuarios grid provenientes del exterior se ejecutan jobs del dominio local de usuarios, y (2) reporta al GIS la información sobre de los recursos. Dentro de un dominio, uno o más schedulers locales corren bajo distintas políticas establecidas por el gestor de recursos. Ejemplos de este tipo de schedulers locales incluyen a OpenPBS [19] y a Condor [6]. Además, un LRM recolecta información de recursos locales empleando herramientas como Network Weather Service [23], Hawkeye [6] y Ganglia [20], y reportan esta información sobre el estado de los recursos al GIS.

3. Desafíos de scheduling en el paradigma grid

Los algoritmos de scheduling fueron intensamente estudiados en los sistemas distribuidos y paralelos tradicionales, como las máquinas SMP (*symmetric multiple processor*, MPP (*massively parallel processors*) y COW (*cluster of workstations*). Observando estos trabajos previos puede verse que los algoritmos de scheduling están evolucionando junto con la arquitectura de los sistemas distribuidos y paralelos[1].

Si bien es posible buscar inspiración en trabajos previos, los modelos de scheduling tradicionales generalmente producen en la práctica pobres planificaciones grid. La razón de este pobre desempeño puede encontrarse en las asunciones subyacentes de los sistemas tradicionales [2]:

- Todos los recursos residen bajo un único dominio administrativo.
- Con el objetivo de prover *single system image* [10], el scheduler controla todos los recursos.
- El conjunto de recursos es invariante.
- La contención causada por las aplicaciones que arriban puede manejarse mediante el scheduler según ciertas políticas, de forma tal que su impacto en la performance que el nodo puede proveer a cada aplicación puede ser acertadamente predecible.
- Las aplicaciones (cómputos) y sus datos residen en el mismo sitio o la obtención de los datos es un proceso altamente predecible, lo cual puede verse como un *overhead* constante.

Desafortunadamente, todas estas asunciones no se mantienen bajo el paradigma grid. En él, muchas características únicas provocan que el diseño de algoritmos de scheduling sea desafiante [24]. De entre dichas características las de mayor relevancia son:

Heterogeneidad y autonomía. La heterogeneidad de los nodos y de las redes subyacentes origina diferentes capacidades para el procesamiento de jobs y el acceso a datos. La autonomía de los nodos individuales y de los diferentes dominios de administración dificultan la tarea de un scheduler de grid capaz de adaptarse a las variadas políticas locales.

Dinamismo de performance. La dinámica del grid impacta fuertemente en la performance global del sistema, dificultando la tarea de un scheduler de grid, pues suelen emplearse modelos de performance clásicos.

[1] Por cuestiones de espacio este trabajo no presenta la evolución de los algoritmos de scheduling.

Selección de recursos y separación de cómputos y datos. Las ventajas que puede traer un recurso de cómputo que presenta bajo costo computacional puede verse neutralizado por su elevado costo de acceso a un nodo de almacenamiento.

4. Trabajos futuros

En este trabajo se plantea la importancia de contar con algoritmos de scheduling eficientes a nivel de grid para lograr planificaciones efectivas. Es útil el análisis de trabajos previos relacionados con sistemas distribuidos y paralelos tradicionales como punto de partida. Sin embargo es claro que el paradigma grid introduce una problemática nueva y por lo tanto nuevos aspectos deben considerarse.

En la sección anterior se identificaron y describieron brevemente los tres desafíos de investigación más importantes en este tópico. De esta forma sostenemos que éste sigue siendo un área de investigación tanto interesante como relevante.

A partir de este trabajo se pretende continuar en esta línea de investigación tratando de obtener nuevos algoritmos generales no solo para scheduling bajo grid sino para grandes sistemas distribuidos en general.

Referencias

[1] Mark Baker, Rajkumar Buyya, and Domenico Laforenza. Grids and grid technologies for wide-area distributed computing. *Softw, Pract. Exper*, 32(15):1437–1466, 2002.

[2] Francine Berman. High-performance schedulers. In Ian Foster and Carl Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 279–309. Morgan Kaufmann, San Francisco, CA, 1999. Chap. 12 12.1 Scheduling applications, Timeframe-Specific Predictions Dynamic Information 287 Adaptation 287 Program Model 289 Performance Scheduling Policy 292 12.4 Case Study: The AppLeS Project 294 12.5 Scheduling Digital Sky Survey Analysis 297 Dynamic Monitoring Mechanisms 305 High Level Language Support 306 Integration with Other Software Tools 306 Assistance for Multischeduling 307.

[3] Francine Berman, Richard Wolski, Henri Casanova, Walfredo Cirne, Holly Dail, Marcio Faerman, Silvia M. Figueira, Jim Hayes, Graziano Obertelli, Jennifer M. Schopf, Gary Shao, Shava Smallen, Neil T. Spring, Alan Su, and Dmitrii Zagorodnov. Adaptive computing on the grid using appleS. *IEEE Trans. Parallel Distrib. Syst*, 14(4):369–382, 2003.

[4] Francine D. Berman, Rich Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao. Application-level scheduling on distributed heterogeneous networks. In *CD-ROM Proceedings of Supercomputing'96*, Pittsburgh, PA, November 1996. IEEE.

[5] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[6] Condor. Condor, high throughput computing. http://www.cs.wisc.edu/condor.

[7] Keith D. Cooper, Anshuman Dasgupta, Ken Kennedy, Charles Koelbel, Anirban Mandal, Gabriel Marin, Mark Mazina, John M. Mellor-Crummey, Francine Berman, Henri Casanova, Andrew A. Chien, Holly Dail, Xin Liu, Alex Olugbile, Otto Sievert, Huaxia Xia, L. Johnsson, B. Liu, M. Patel, Daniel A. Reed, W. Deng, Celso L. Mendes, Zhiao Shi, Asim YarKhan, and Jack Dongarra. New

grid scheduling and rescheduling methods in the grADS project. In *IPDPS*. IEEE Computer Society, 2004.

[8] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A resource management architecture for metacomputing systems. *Lecture Notes in Computer Science*, 1459:62–??, 1998.

[9] Karl Czajkowski, Carl Kesselman, Steven Fitzgerald, and Ian T. Foster. Grid information services for distributed resource sharing. In *HPDC*, pages 181–194. IEEE Computer Society, 2001.

[10] Javier Echaiz and Jorge Ardenghi. Single System Image: Pilar de los Sistemas de Clustering. *V Workshop de Investigadores en Ciencias de la Computación, WICC 2003*, pages 210–214, May 2003.

[11] Foster and Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, volume 2, 2003.

[12] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. MORGAN-KAUFMANN, 1999.

[13] Ian T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *CCGRID*, pages 6–7. IEEE Computer Society, 2001.

[14] Globus. The Globus Alliance. http://www.globus.org.

[15] Tibor Gyires. A resource allocation protocol for providing quality of service in grid computing. In *AICT/ICIW*, page 44. IEEE Computer Society, 2006.

[16] Volker Hamscher, Uwe Schwiegelshohn, Achim Streit, and Ramin Yahyapour. Evaluation of job-scheduling strategies for grid computing. In Rajkumar Buyya and Mark Baker, editors, *GRID*, volume 1971 of *Lecture Notes in Computer Science*, pages 191–202. Springer, 2000.

[17] J. Schopf. Ten Actions When SuperScheduling. Technical report, document of Scheduling Working Group, Global Grid Forum, July 2003.

[18] Ashfaq A. Khokhar, Viktor K. Prasanna, Muhammad E. Shaaban, and Cho-Li Wang. Heterogeneous computing: Challenges and opportunities. *IEEE Computer*, 26(6):18–27, 1993.

[19] OpenPBS. Open Portable Batch System. http://www.openpbs.org.

[20] Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler. Wide area cluster monitoring with ganglia. In *CLUSTER*, page 289. IEEE Computer Society, 2003.

[21] Siegel, Dietz, and Antonio. Software support for heterogeneous computing. *CSURV: Computing Surveys*, 28, 1996.

[22] Xian-He Sun and Ming Wu. Grid harvest service: A system for long-term, application-level task scheduling. In *17th International Parallel and Distributed Processing Symposium (IPDPS-2003)*, pages 25–25, Los Alamitos, CA, April 22–26 2003. IEEE Computer Society.

[23] Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, October 1999.

[24] Y. Zhu. A Survey on Grid Scheduling Systems. Technical report, Department of Computer Science, Hong Kong University of science and Technology, 2003.