Metodología y Herramienta para Ejecución de Flujos de Trabajo en Grids

Jimmy Ortega, Sergio Naranjo y Emilio Hernández
{jimmy.ortega,sergio.naranjo}@ldc.usb.ve, emilio@usb.ve

Universidad Simón Bolívar, Departamento de Computación,
Apartado 89000, Caracas 1080-A, Venezuela

Resumen Se presenta una metodología de trabajo y una aplicación interactiva complementaria, denominada JobDag, que permite la ejecución de comandos sobre archivos seleccionados de un área de trabajo. Permite representar flujos de trabajo basados en DAGs (Grafos Dirigidos Acíclicos), para su ejecución diferida sobre plataformas remotas de tipo Cluster y de tipo Grid. Se probó JobDag con paquetes de comandos como SU (Seismic Unix), para los que se definieron los tipos de objeto que manipula y los comandos que aplican sobre dichos objetos.

1. Introducción

La interacción con plataformas de cómputo remotas, por ejemplo multiprocesadores o clusters de computadoras a los que se accede a través de grids computacionales, puede hacerse a través de una variedad de herramientas. La forma más simple es enviando los trabajos a una cola remota usando comandos ejecutados desde un shell, como por ejemplo las interfaces para enviar trabajos a la cola de una plataforma de tipo cluster como SLURM [1] o Torque [2], o las que se pueden usar para enviar trabajos a un Grid, por ejemplo de tipo gLite [3] o Globus Toolkit 4 [4] .

En este trabajo se describe el uso combinado de una plataforma de conexión a un grid basada en máquina virtual, como Xen [5] o VirtualBox [6], con una herramienta interactiva instalada en dicha máquina virtual, denominada JobDag. Esta herramiente está orientada a ejecución de flujos de procesos bajo un esquema DAG (Grafo Dirigido Acíclico, en sus siglas en inglés), tanto en clusters como en grids. Tiene como objeto realizar la ejecución diferida del DAG y de todos los comandos asociados al mismo, para habilitar el acceso a plataformas de cómputo paralelo compartidas, que normalmente se utilizan a través de colas de trabajo.

El esquema de ejecución de múltiples comandos bajo una relación de precedencia de ejecución ha sido muy utilizada, por ejemplo a través del comando "make", disponible en toda la familia de sistemas de operación basados en Unix. La secuenciación de comandos bajo un esquema de flujo de trabajo o *workflow*, que puede ser lineal o seguir una arquitectura de tipo DAG , también ha sido objeto de mucha investigación, y se han propuesto herramientas para definir

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 775-784, 2008.

workflows [7,8,9,10], lenguajes para especificar workflows [11,12] y taxonomías de ejecución bajo estos esquemas: [13,14], entre otros esquemas y técnicas específicas, de mucha utilidad, en particular en entornos grid.

La característica principal de estas herramientas de definición de flujos de trabajo es que se construye un grafo cuyos nodos son las acciones a ejecutar. En el caso de JobDag, los nodos son los objetos a manipular y las aristas son las acciones a ejecutar, es decir, se elabora el grafo dual. Los objetos están contenidos en archivos y en la versión actual de JobDag se identifican por la extensión del archivo, aunque se planea distinguir los tipos de objeto por su estructura, en lugar de hacerlo por la extensión.

Esta forma de especificación permite una interfaz más familiar con herramientas que un usuario inexperto está acostumbrado a usar. Provee una interfaz más parecida a los programas exploradores de ventanas, como Konqueror o Galeon, con el objeto de que los comandos se definan con la acción "Abrir con...", típicamente activable con el botón derecho del ratón. Además permite la definición de diferentes "Ambientes de Ejecución", como por ejemplo un cluster en particular o un grid, ambientes que el usuario puede seleccionar entre una serie de aditamentos o *plugins* de ejecución. Incluso diferentes clusters suelen tener diferentes requerimientos de ejecución. Otra de las características importantes es que ofrece la posibilidad de introducir de manera sencilla, con una interfaz de ventana, comandos de otros paquetes, de modo que puedan combinarse comandos provenientes de diferentes paquetes.

El resto de este artículo está organizado de la siguiente forma. La sección 2 describe la metodología de trabajo y el ambiente en que se utiliza la herramienta. La sección 3 describe el diseño a nivel macro de JobDag. La sección 4 describe con más detalle los componentes. La sección 5 describe los resultados hasta el momento y muestra algunos ejemplos. La sección 6 ofrece nuestras conclusiones y sugerencias para el trabajo futuro.

2. Metodología de trabajo con JobDag

Esta herramienta y otras están diseñadas para trabajar en un contexto bien específico, que consiste en acceder a un Grid de tipo gLite o GT4 a través de una máquina virtual, tal como Xen o VirtualBox, conectada al grid a través de un túnel seguro, que da entrada al grid aún cuando éste se implemente sobre una Red Privada Virtual.

En este contexto las interfaces de las aplicaciones pueden ser desarrolladas directamente como cualquier aplicación de escritorio (ejemplo: utilizar interfaces en forma de ventanas tales como Qt o GTK) en vez de utilizar una interfaz de Portal Web. El esquema de conexión se representa en la figura 1.

Como sucede con la mayor parte de las aplicaciones hechas para enviar trabajos a un grid, JobDag se usa de modo interactivo para definir las tareas que posteriormente se enviarán a ejecutar en modo "batch" o por lotes. Se diseñó una interacción similar a la que se realiza con herramientas de navegación en sistemas de archivos, como konqueror (KDE) o galeon (GNOME), en el sentido de que

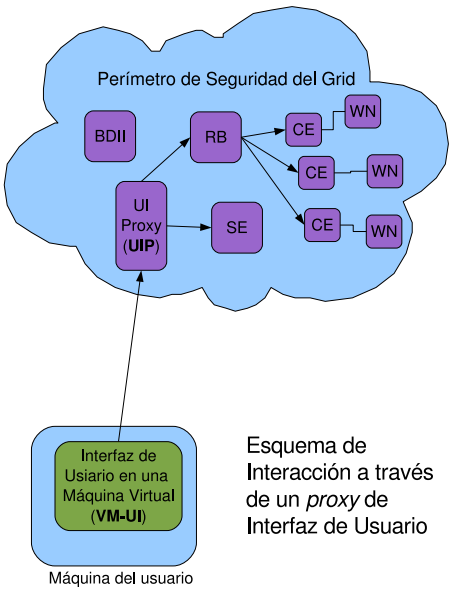

Figura 1. Proxy para Interfaces de Usuario

se utiliza el botón derecho del ratón para ejecutar la acción "Abrir con...". Sólo los programas del ambiente de ejecución seleccionado que aplican sobre ese tipo de archivo aparecerán como opción. Se va construyendo un DAG en el que los nodos son los íconos de los archivos (todavía no generados) y las acciones están representadas por las aristas del DAG. Estos íconos los llamamos *archivos futuros*. Esto contrasta con aplicaciones similares de definición de flujos de trabajo, en las que se representa el grafo dual, es decir, los nodos son las aplicaciones que se invocan. En la figura 2 se muestra la apariencia del uso de JobDag, en la que se va construyendo el DAG para ejecución defierida. Los archivos futuros se muestran de forma transparente, indicando que se espera por los resultados de la ejecución.

La forma de comenzar a generar el DAG de ejecución es importando un archivo ya existente del espacio de trabajo del usuario, por ejemplo, el sistema de archivos local. Otra forma consiste en invocar un programa que genera un archivo inicial, por ejemplo a partir de una especificación que se introduce en forma interactiva.

3. Diseño e Implementación de JobDag

El diseño de esta herramienta se enfoca a la posibilidad de ser utilizada con varios tipos de plataformas de cómputo, en particular de tipo grid. El módulo principal realiza la interacción con el usuario y genera las estructuras de datos

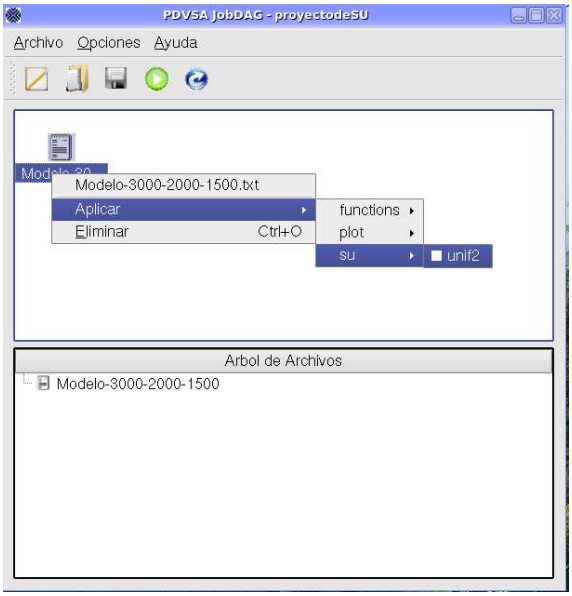

Figura 2. Interacción con JobDAG

que representan cada uno de los objetos que maneja, principalmente archivos XML que representan las funciones, los ambientes de ejecución, y los DAGs, entre otros.

Los módulos que invocan la ejecución del DAG en las plataformas remotas son independientes, se asocian como aditamentos o *plugins* del módulo central (ver figura 3). Estos aditamentos deben leer y analizar sintácticamente las estructuras XML generadas por el módulo central, para proceder a generas los *scripts* de ejecución (archivos JDL en el caso de los grids basados en Globus) y realizar la invocación remota, por ejemplo invocar el comando *globus-job-submit*.

Las principales características de esta herramienta son:

1. Se puede incrementar sus posibilidades a través de aditamentos o *plugins* independientes del módulo principal, que aumentan el número de tipos de plataformas de cómputo soportadas, el cambio de imágenes del programa y la carga de nuevas listas de funciones a algún ambiente.
2. Permite definir definir Ambientes de Trabajo (descritos en la siguiente sección), que incluyen listas de funciones. Se puede predeterminar funciones por defecto sobre las extensiones de archivo, modificar parámetros de los ambientes de ejecución, modificar las características de las funciones cargadas en la aplicación y alternar las vistas de iconos a listas en la visualización.
3. La interacción es similar a la que se realiza con herramientas de navegación en sistemas de archivos, como konqueror (KDE) o galeon (GNOME) o el explorador de ventanas de Windows.

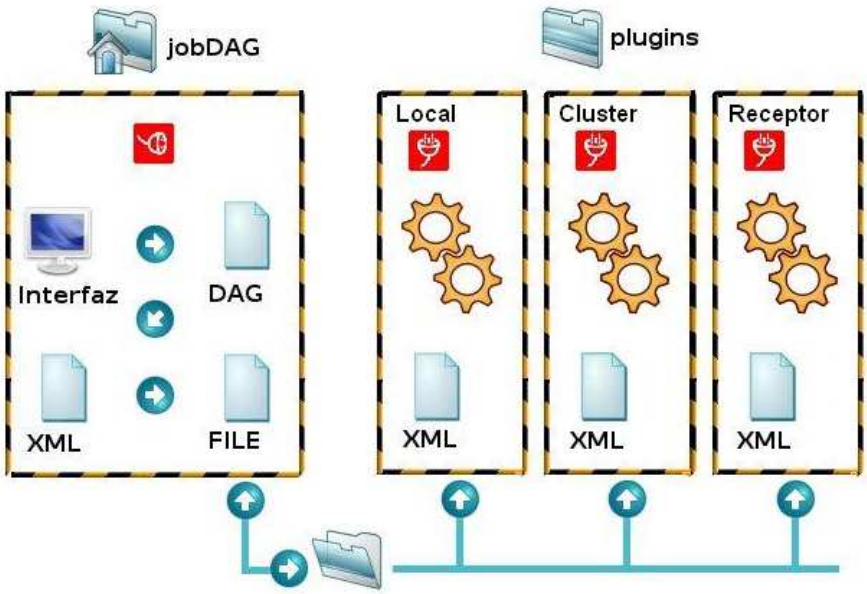

Figura 3. Diseño de JobDag

4. Todas las estructuras de archivos tienen formato XML, evitando la dependencia de manejadores de bases de datos de ningún tipo. Tanto los archivos del sistema, como los archivos de usuario están creados bajo tal especificación.

4. Otras características de JobDag

Algunas características adicionales hacen de JobDag una herramienta de fácil manejo, algunas de las cuales se mencionan a continuación:

4.1. Plantillas

En JobDag se puede guardar el grafo generado como una plantilla, de modo que puede aplicarse sobre un conjunto similar de archivos iniciales. La plantilla

se guarda con la misma estructura del DAG de ejecución, con la diferencia de que el nombre de los archivos iniciales y el nombre de los archivos intermedios se guarda como parte del nombre que tendrán los archivos al aplicar la plantilla. El usuario especificará el prefijo de dichos nombres en cada caso.

4.2. Ambientes de Ejecución

Los Ambientes de Ejecución representan plataformas de ejecución con acceso a *listas de funciones*. Al utilizar jobDAG para hacer un DAG y ejecutarlo en un cluster, podría seleccionarse un Ambiente de Ejecución (descrito abajo) y asignar un nombre al DAG, hacer la carga de alguno o cambiar las preferencias. A un Ambiente de Ejecución se le asocia:

1. Un aditamento o *plugin* de Ejecución.
2. Un aditamento o *plugin* de Recuperación de Salidas
3. Un conjunto de Listas de Comandos
4. Opcionalmente, aditamentos de Visualización

Estos componentes se explican a continuación.

4.3. Aditamentos (*plugins*) asociados a los Ambientes de Ejecución

Los aditamentos de ejecución son programas que se deben desarrollar aparte y agregar a JobDag como parte del conjunto de opciones de ejecución que se ofrece. Están estrechamente vinculados con la forma en que se envían los trabajos a la cola de trabajos de una plataforma remota, aunque se ofrece también un aditamento de ejecución en la plataforma local.

Durante la operación de JobDag, una vez construido el grafo, se puede proceder a enviarlo a la plataforma de ejecución, momento en el cual se invoca el aditamento correspondiente. El aditamento de ejecución puede mostrar una pantalla de interacción para afinar los parámetros de ejecución. La información de ejecución se extrae de la especificación XML del grafo, y el aditamento de ejecución extrae la información de este archivo, construye el *script* o los *scripts* necesarios para la ejecución del DAG y los envía a ejecución en la plataforma remota, guardando la información necesaria para recuperar los resultados posteriormente.

Un aditamento de recuperación de salidas permite traer al ambiente del usuario los archivos generados, tanto finales como intermedios. Cuando el usuario actualiza la información que ofrece la vista gráfica, se ejecuta automáticamente este aditamento, lo que produce un cambio visual en el ícono, que deja de ser transparente. Adicionalmente, se indica el estado de la ejecución del DAG.

4.4. Listas de Comandos

Como parte de las actividades administrativas y de configuración de JobDag, se pueden agregar comandos a los ya existentes. Los comandos se agregan

interactivamente, y se agrupan en lo que llamamos Listas de Comandos. Estas listas están asociadas a paquetes que han sido instalados en determinados Ambientes de Ejecución. En consecuencia, es posible asociar libremente las Listas de Comandos a los Ambientes de Ejecución. Dependiendo del Ambiente de Ejecución seleccionado inicialmente, las opciones de comandos que aparecerán para cada tipo de archivo serán las que estén presentes en las Listas de Comandos asociadas.

Las Listas de Comandos quedan especificadas en archivos XML, que contienen toda la información a ser desplegada en el momento de seleccionar un comando específico con el botón derecho del ratón.

5. Resultados

Hemos hecho pruebas preliminares con JobDag, ejecutando remotamente en un cluster 48 procesadores, AMD 2.4 GHz interconectados con Myrinet 10Gbps full duplex. Las pruebas se hicieron desde la red local donde se encuentra el cluster, de modo que el tiempo de retardo en la ejecución está principalmente determinado por el tiempo en la cola de procesos a ser ejecutados. Se hicieron pruebas con paquetes de comandos como SU (Seismic Unix [15]), que es un conjunto de comandos para procesamiento sísmico. Como un ejemplo sencillo del tipo de ejecuciones con las que se hicieron las pruebas de JobDag, se muestra en la figura 5 la especificación de un DAG sencillo cuyo resultado final es un par de imágenes, una de las cuales se muestra en la figura 5. Esta imagen es generada por el comando de SU llamado *supswigb*. La imagen se abre de forma natural desde JobDag usando el ratón, una vez que el archivo de salida haya sido transferido desde el grid después de la ejecución del DAG.

No se proveen cifras de tiempo de ejecución del mismo JobDag porque son irrelevantes frente a los tiempos de encolamiento en el grid y el tiempo de ejecución de las tareas, una vez que se le asignan recursos computacionales. La ventaja de JobDag reside en la capacidad expresiva de flujos de trabajo, de un modo intuitivo, similar a la manera en que se trabaja con archivos en un ambiente de ventanas tradicional.

En estos momentos se tiene una versión funcional, con aditamentos o *plugins* de ejecución local y de envío a un grid de tipo gLite, implementado sobre un API provisto por el software de gLite. Está en desarrollo una versión que agrega el manejo de ejecución de tareas bajo un esquema de barrido de parámetros (*parameter sweeping*), lo que permite definir experimentos computacionales con muchas corridas en una sola especificación.

6. Conclusiones y Trabajo Futuro

La utilización de plataformas de cómputo intensivo, como clusters de computadoras y grids de cómputo y datos, necesita de interfaces que faciliten la utilización de dichas plataformas por parte de usuarios no especialistas en el área de

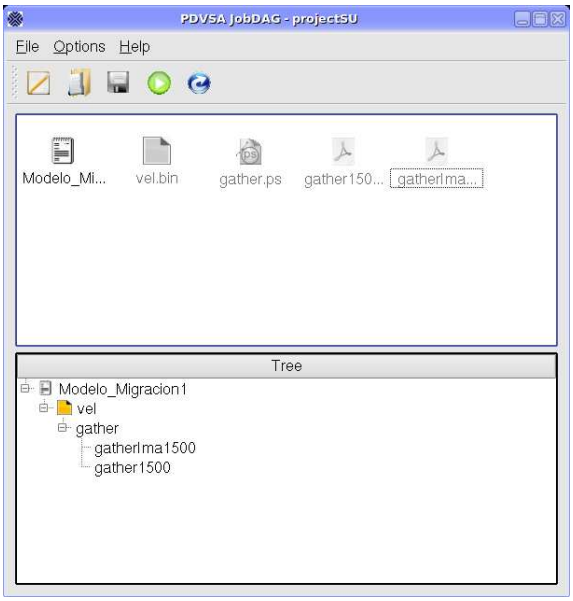

Figura 4. Ejemplo: especificación del DAG

computación. La combinación de (1) el uso de una máquina virtual que se adhiere momentáneamente a un grid y (2) aplicaciones como JobDag, que permite la planificación y ejecución de scripts de forma diferida, facilita la interacción con un grid, en relación con herramientas como Portales Web.

En particular, JobDag proporciona herramientas para modelar dentro del programa los ambientes o plataformas de cómputo en los cuáles serán ejecutados los comandos e interactuar con dichos entornos, ya sean remotos o locales. Tiene la ventaja de poseer una interfaz que es familiar para usuarios de ambientes de ventanas, en los que se muestran los íconos de los archivos y con el ratón se aplican acciones sobre los archivos. Con el botón derecho se puede hacer "Abrir con.." y JobDag muestra todos los programas que pueden ejecutarse sobre el archivo en forma diferida. La ejecución se invoca después de haber definido toda la secuencia de comandos.

Estamos actualmente trabajando en la integración de este esquema de flujo de trabajo (*workflow*) con un esquema de especificación de barrido de parámetros (*parameter sweeping*), lo que permitirá definir una mayor gama de experimentos computacionales.

Referencias

1. A. Yoo, M. Jette, and M. M. Grondona. Slurm: Simple linux utility for resource management. *Lecture Notes in Computer Science*, 2862:44–60, 2003.

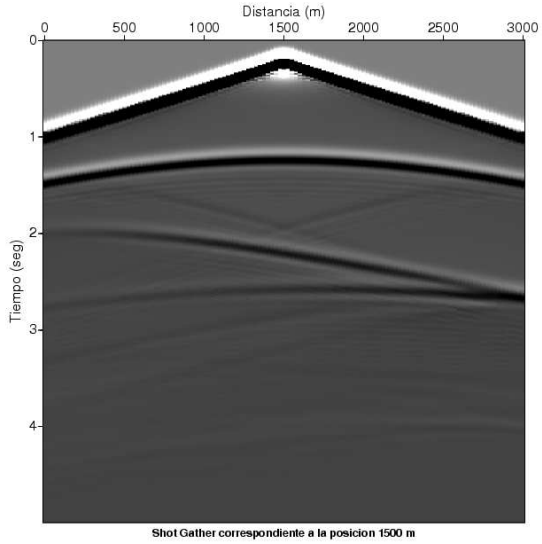

Figura 5. Ejemplo: ejecución y uno de los archivos finales

2. Cluster resources inc. TORQUE Resource Manager, 2007. http://www.clusterresources.com/.
3. gLite. Lightweight Middleware for Grid Computing, 2006. http://glite.web.cern.ch/glite/.
4. Ian T. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.
5. B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R.Ñeugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2003.
6. Virtual Box Team. Virtual Box. http://www.virtualbox.org/.
7. D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Res*, 34(Web Server issue), July 2006.
8. Soonwook Hwang and C. Kesselman. Grid workflow: a flexible failure handling framework for the grid. In *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, pages 126–137, 2003.
9. Junwei Cao, S. A. Jarvis, S. Saini, and G. R. Nudd. Gridflow: workflow management for grid computing. In *Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on*, pages 198–205, 2003.
10. K. Amin, G. von Laszewski, M. Hategan, N. J. Zaluzec, S. Hampton, and A. Rossi. Gridant: a client-controllable grid workflow system. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference*, pages 10 pp.+, 2004.
11. T. Fahringer, J. Qin, and S. Hainzer. Specification of grid workflow applications with agwl: an abstract grid workflow language. In *Cluster Computing and the Grid,*

2005. CCGrid 2005. IEEE International Symposium on, volume 2, pages 676–685 Vol. 2, 2005.
12. The Condor Team. DAGMan (Directed Acyclic Graph Manager), 2007. http://www.cs.wisc.edu/condor/dagman/.
13. Jia Yu and Rajkumar Buyya. A taxonomy of workflow management systems for grid computing. *Journal of Grid Computing*, 3(3-4):171–200, September 2005.
14. Geoffrey Charles Fox and Dennis Gannon. Special issue: Workflow in grid systems. *Concurrency and Computation: Practice and Experience*, 18(10):1009–1019, 2006.
15. Center for Wave Phenomena. CWP/SU: Seismic Unix, 2007. http://www.cwp.mines.edu/cwpcodes/index.html.