Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Avances en la Migración de Software a Plataformas Grid en la UCV

Jaime Parada, Robinson Rivas, Daniel Espinoza, Esther González

RT 2012-01

Centro de Computación Paralela y Distribuida de la UCV

CCPD-UCV

Caracas, Mayo 2012.

Avances en la Migración de Software a Plataformas Grid en la UCV

Jaime A. Parada D., Robinson Rivas, Daniel Espinoza, Esther González
Centro de Computación Paralela y Distribuida
Universidad Central de Venezuela
{jaime.parada, robinson.rivas}@ciens.ucv.ve

Abstract

At Universidad Central de Venezuela (UCV) we developed over the years many applications for parallel architectures, in areas such as petroleum, image processing and hydrologic models. However, porting these applications to grids has proven to be a hard and slow process.

In this work, we explore which jobs and applications are suitable for a Grid platform, and how to modify them if they are not. Additionally, we present the advances in the last two years: the porting of the chemistry system Cativic, the atmospheric models AIRES and BRAMS and the development of EasyGrid, We are in the first stages of Grid technology use, and this paper shows how UCV is looking forward in order to adapt this model of High Performance Computing for everyday scientific tasks.

Resumen

En la Universidad Central de Venezuela se han desarrollado aplicaciones en diferentes ámbitos de la ciencia para aprovechar arquitecturas paralelas. Sin embargo, la migración de estas aplicaciones a plataformas Grid ha sido lenta, en parte debido a problemas en la implantación de las plataformas requeridas.

En este trabajo, se explora brevemente cuáles trabajos y aplicaciones son adecuados para ejecutarlos sobre plataformas Grid y cómo modificarlos si no son adecuados para dicha plataforma de ejecución. Posteriormente se presentan los avances que se han hecho para acercar la tecnología de grids a la comunidad científica, haciendo un balance de las aplicaciones EasyGrid, Cativic y la adaptación de BRAMS y AIRES en el ambiente operativo de la UCV.

Se presentan las adaptaciones hechas, las dificultades enfrentadas, y una perspectiva acerca de las nuevas aplicaciones que se están llevando al Grid UCV.

1. Introducción

Las plataformas Grid pueden describirse brevemente como la unión de las tecnologías de Computación de Alto Desempeño, redes de Alta Velocidad, sofisticadas plataformas de software de control (*middleware*) y recursos humanos especializados, trabajando juntos para resolver problemas de cómputo distribuido de alta complejidad y demanda computacional [1]. Este tipo de infraestructuras son importantes especialmente en áreas donde se requiere alto poder de cómputo, de almacenamiento, o una mezcla de ambas cosas, por ejemplo en Física de Altas Energías, Química Computacional o modelado ambiental. Otro grupo de aplicaciones son aquellas que demandan recursos de naturaleza intrínsicamente distribuidos.

El término *habilitación para Grid (Grid enabling) - o gridización* (*gridification*) - de una aplicación, no significa simplemente ejecutar un trabajo sobre una plataforma Grid. La *gridización* implica alguna transformación del código de la aplicación para que se ejecute y de alguna manera aproveche la naturaleza distribuida de la plataforma Grid.

En la Universidad Central de Venezuela (UCV), específicamente en el Centro de Computación Paralela y Distribuida de la Facultad de Ciencias, se han desarrollado y adaptado algunas aplicaciones para la resolución de problemas de cómputo paralelo, sin embargo las mismas no han sido llevadas aún a plataformas grid. Los primeros intentos se han hecho con software desarrollado en otras instituciones como CATIVIC [2], BRAMS [3] y AIRES [4]. Estas aplicaciones representan ejemplos de software que requiere altos tiempos de cómputo, por lo que su adaptación al grid representa un avance para los usuarios. Otro trabajo adelantado ha sido el de acercar el grid a los potenciales usuarios (físicos y químicos especialmente) mediante el desarrollo de EasyGrid [5], una herramienta basada en la plataforma .NET de Microsoft.

A pesar de los avances, la *gridización* de estas aplicaciones ha sido lenta y se han presentado diversos inconvenientes, principalmente debido a la falta de una infraestructura adecuada. En este artículo se presentan los primeros avances para la creación de la infraestructura Grid UCV y las aplicaciones que han dado origen al proyecto.

2. G*ridización* de Aplicaciones

2.1. Definición de *Gridización*

Una definición simple de gridización de una aplicación es [11]: "*Ser capaz de crear una aplicación que utilice los recursos distribuidos disponibles sobre una plataforma Grid de forma más efectiva que simplemente ejecutar la aplicación sobre un único recurso local o remoto*". Una definición más detallada es [12]: "*La gridización se refiere a la adaptación o el desarrollo de un programa para proporcionarle la capacidad de interactuar con un middleware grid, de forma tal que pueda ser planificado para ejecución y utilice recursos a partir de un conjunto dinámico y distribuido de "recursos grid" de forma que cumpla efectivamente con las necesidades del programa*"

Cómo se habilita una aplicación para ejecutarse sobre un Grid es aún un área de investigación, sin un enfoque estándar. Una plataforma de cómputo distribuido – como lo es una plataforma Grid – ofrece la posibilidad de ejecutar múltiples programas en distintos computadores con el potencial de reducir el tiempo total de ejecución o aprovechar las capacidades de almacenamiento distribuido. Estos múltiples computadores pueden ser utilizados principalmente de dos formas: a) Separadamente para resolver problemas individuales, o b) Colectivamente para resolver un único problema. En las siguientes secciones, veremos algunos enfoques para llevar a cabo este cometido.

2.2. Barrido de Parámetros

En ciertas áreas científicas se necesita ejecutar el mismo programa muchas veces pero con diferentes datos de entrada. Este tipo de problemas son muy comunes en experimentos que involucran simulaciones. Si los experimentos se ejecutan bajo múltiples configuraciones, se pueden desarrollar leyes y teorías que pueden encapsular conocimientos acerca de los sistemas bajo estudio. Este tipo de problemas son ejemplos de experimentos de barrido de parámetro (Parameter Sweep Experiments - PSE) o, como también se les conoce, aplicaciones de barrido de parámetros (Parameter Sweep Applications - PSA). El científico intenta "barrer" a lo largo del espacio de parámetros con valores distintos de parámetros de entrada en busca de una solución. Existen diferentes razones para realizar el barrido de parámetros: por ejemplo el que no exista una forma matemática cerrada para la solución y se deba emplear un enfoque de ensayo y error. Normalmente, existirán diversos parámetros que pueden ser modificados obteniendo una vasta combinación de valores de entrada. Se requiere de algún mecanismo automatizado que permita especificar sobre cuál o cuáles parámetros se realizará el barrido y de qué forma serán planificados los diversos trabajos en la plataforma Grid.

El barrido de parámetros podría implementarse de manera simple, enviando múltiples archivos de descripción de trabajo (uno para cada conjunto de parámetros) pero esta solución es ineficiente. Existen diversas maneras de especificar el barrido de parámetros en los lenguajes de descripción de trabajo. Por ejemplo, en Condor [13] un archivo de descripción de trabajo puede servir para enviar más de un trabajo permitiendo que el programa se ejecute con diferentes parámetros u opciones. Otros ejemplos son Nimrod/G [14] y APST (AppLeS Parameter Sweep Template) [15].

2.3. Utilización de un Programa Existente Ejecutándose en Múltiples Computadores sobre el Grid

En algunos problemas, los datos necesarios para resolverlo pueden ser divididos en partes y procesados independientemente. Una vez que los datos han sido divididos se pueden emplear computadores individuales sobre cada parte de los datos. Aplicaciones que realizan búsquedas sobre grandes bases de datos pueden encajar bien en este esquema.

Tomemos como ejemplo BLAST (Basic Local Alignment Search Tool) que es utlizado en bioinformática para encontrar correspondencias estadísticas entre secuencias de genes sobre grandes bases de datos. Un usuario de BLAST puede realizar una consulta sobre una secuencia que es comparada con la base de datos de secuencias conocidas con la finalidad de descubrir relaciones o correspondencias de dicha secuencia con una familia de genes. Si se desea localizar sólo una secuencia y la base de datos puede ser particionada, cada nodo de trabajo puede trabajar con una parte de dicha base de datos [13].

Por supuesto, existe un costo significativo debido al envío de las partes a los distintos sitios, pero una vez realizado, los datos pueden ser reutilizados. Si se envían diversas consultas de forma simultánea, cada nodo debería tener acceso a la base de datos completa. Habrá una cantidad significativa de tráfico de red para acceder a la base de datos a menos que ésta se encuentre en cada sitio. Estos dos enfoques descritos, permiten acelerar las búsquedas de BLAST sin requerir que la aplicación sea reescrita o alterada.

Otro conjunto de programas que podría interesar ejecutar sobre una plataforma Grid, son los programas heredados. Ejecutar programas heredados sobre plataformas para las que no fueron escritos es todo un reto; hacerlo sobre una plataforma Grid es un reto aún mayor. Lo primero que hay que determinar es si se puede ejecutar sobre un recurso particular del Grid o si el código puede modificarse para que así sea. Dado que los recursos del Grid son heterogéneos, lo mejor que

puede pasar es que los binarios se correspondan con algún recurso de cómputo y con su sistema operativo. Si existe el código fuente documentado, reescribirlo es una posibilidad. Muchas aplicaciones heredadas no poseen el código documentado, o vienen cerradas del fabricante, de forma que reescribirlo no es una opción. Un proyecto que está tratando de direccionar algunos de los problemas de portar código heredado al Grid es el *Grid Enabling Legacy Software* [16].

Otro enfoque utilizado ha sido el empleo de servicios Web. Bajo este esquema el código de la función es *envuelto* para producir un servicio Web. Posteriormente se puede utilizar la URL del servicio Web para acceder a la aplicación desde cualquier parte.

2.4. Escribir una Aplicación Específicamente para un Grid

Se puede escribir una aplicación desde cero que utilice las facilidades que brinda el Grid. Entre las facilidades se suelen encontrar:

- acciones para iniciar transferencias de archivos entre diversas locaciones
- localización de recursos adecuados para el trabajo a ejecutar
- inicio de nuevos trabajos desde una aplicación, etc.

Este enfoque es el más complejo de todos, debido principalmente a que las API del middleware cambian de una versión a otra. Se han hecho algunos progresos e intentos para desarrollar estándares y API de más alto nivel que cualquiera pueda usar, pero aún falta mucho para considerar trabajo como realizado. A nivel del Grid middleware, existe el API Globus [17], y sobre esta se encuentra el API CoGkit. A más alto nivel, encontramos UNICORE [18] y gLite [6].

2.5 Utilización de Múltiples Computadores para Resolver un Único Problema

La computación Grid ofrece la posibilidad de utilizar múltiples computadores para resolver un solo problema y decrementar su tiempo de ejecución (programación paralela). Los dos modelos arquitectónicos principales utilizados para resolver problemas en paralelo son los sistemas de memoria compartida y los sistemas de memoria distribuida. La programación de memoria compartida es aplicable a sistemas de cómputo con multiprocesadores que tengan memoria compartida físicamente, tales como los sistemas multinúcleo.

En los sistemas de memoria distribuida, cada computador posee su propia memoria y se utiliza el paso de mensajes para intercambiar información entre los computadores. La especificación estándar para un API de paso de mensajes de más amplio uso actualmente es MPI (Message Passing Interface), la cual suplantó a PVM (Parallel Virtual Machine) a mediados de los años 90. Existen diversas implementaciones disponibles de MPI, entre las cuales se puede destacar MPICH-2 y OpenMPI. La implementación más destacada para programar sobre plataformas Grid es MPICH-G2 [19].

MPICH-G2 es una implementación de MPI lista para trabajar sobre plataformas Grid que utiliza los servicios de Globus (e.g. job-startup, security) y permite a los programadores acoplar diversas máquinas para ejecutar aplicaciones MPI, aún cuando puedan ser de distintas arquitecturas [20]. De manera automática, MPICH-G2 convierte los datos en mensajes enviados entre máquinas de diferentes arquitecturas y soporta comunicación multiprotocolo, seleccionando automáticamente entre TCP o la implementación MPI nativa que exista en las máquinas que están ejecutando el programa. Sin embargo, MPICH-G2 requiere que los servicios de Globus estén disponibles en todos los computadores participantes, para contactar a cada máquina remota, autenticar al usuario en cada una e iniciar la ejecución (e.g. *fork* de procesos, espacio en la colas, etc.).

Es importante destacar que independientemente del modelo de progamación empleado, se debe tener la infraestructura computacional adecuadamente instalada y configurada.

3. UCV Grid

El entorno operativo de la UCV consta de cuatro servidores SUN dedicados al Grid y un cluster para la ejecución de los programas (basado en Pentium III). En cada uno de los servidores del Grid se instaló uno de los componentes de gLite [6] que es el middleware utilizado como base de nuestra instalación. El Servidor principal de mayor capacidad se usó como Working Node (WN), y los otros tres tienen las funciones de Computing Element (CE), Workload Management Server (WMS), y Storage Element (SE). Adicionalmente, un servidor dedicado se instaló como BDII.

gLite fue seleccionado como middleware para mantener compatibilidad con las plataformas sugeridas por el CERN y en general con los sitios asociados al proyecto EELA2 [7].

El Sistema Operativo base del grid y del cluster de la UCV es Scientific Linux 4. En la Figura 1 se muestra un esquema de la infraestructura de servidores del Grid UCV.

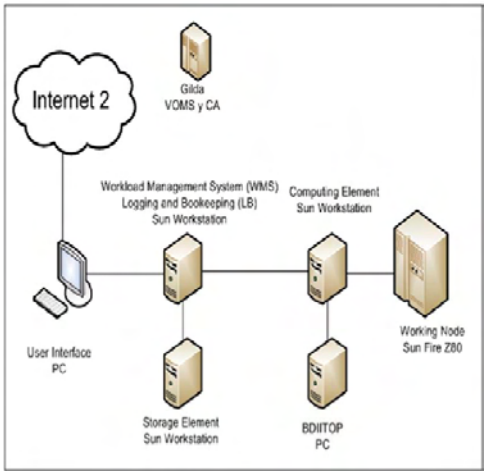

Figura 1: Arquitectura del Grid UCV

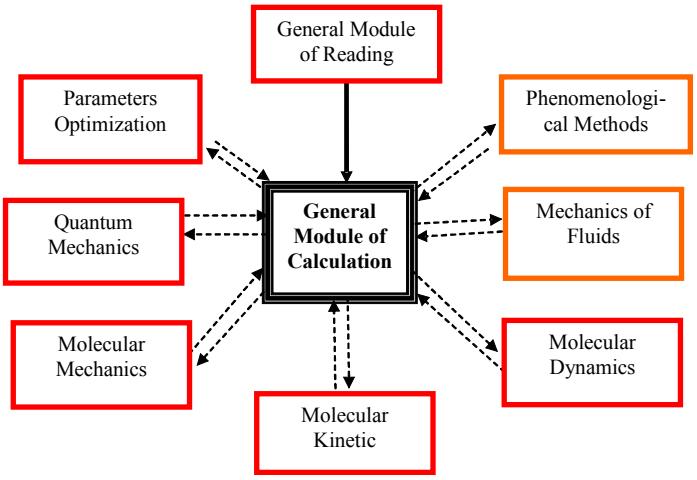

Figura 2: Módulos de CATIVIC

El Grid se encuentra ubicado físicamente en el Laboratorio del Centro de Computación Paralela y Distribuida (CCPD) de la Escuela de Computación. Los equipos son parte de un proyecto Fonacit y los nodos del Grid se encuentran conectados a Internet 2. En el CCPD se ha trabajado en la gridización de diversas aplicaciones científicas. Adicionalmente, se ha desarrollado un sistema que facilita el acceso y uso de los recursos del Grid, vía web. A continuación encontrarán la descripción de estos avances.

4. CATIVIC

Una herramienta comúnmente utilizada para el diseño de nuevos catalizadores es el modelado computacional, para el que se utilizan una serie de técnicas como química cuántica, mecánica clásica y mecánica estadística. Sin embargo, métodos usados en química cuántica como los métodos ab initio resultan inviables para los efectos prácticos cuando se quieren modelar moléculas grandes debido a lo lento de los cálculos.

Para solventar estos problemas, el Grupo de Química Computacional del IVIC desarrolló CATIVIC [2] usando métodos paramétricos. Este software permite modelar moléculas de mayor tamaño, mediante una serie de módulos para las diversas operaciones de cómputo como se muestra en la Figura 2. CATIVIC fue desarrollado en Fortran a lo largo de varios años, y actualmente se encuentra paralelizado mediante las librerías OpenMPI.

El principal problema de CATIVIC es el tiempo de ejecución requerido para el modelado cuando se requiere realizar muchas pruebas, o cuando las moléculas son complejas. Esto motivó a intentar su gridización a partir de 2008.

4.1. *Gridización de CATIVIC*

CATIVIC puede ser ejecutado tanto en modo secuencial como en paralelo. Para su gridización, se decidió utilizar la versión secuencial, y enviar su ejecución a cada nodo disponible de la grid. En este caso, los pasos seriales se coordinaron mediante AMGA [21], lo que permitió reducir los tiempos de espera asociados al Working Node de gLite.

Una vez hechas las pruebas de funcionamiento, descritas en [8], se procedió a desarrollar una interfaz de usuario para facilitar el uso de CATIVIC desde fuera de la UCV a los usuarios registrados.

4.2. Interfaces de usuario de Grid-CATIVIC

A continuación se detallan algunas de las interfaces generadas para el proyecto. Mediante estas interfaces, a los usuarios se les facilita el uso de la herramienta, la carga de archivos de prueba y la descarga de resultados, así como el monitoreo de las actividades del sistema.

En la Figura 3 se muestra la página de ingreso al sistema. Los usuarios deben estar previamente registrados.

Una vez autenticados, los usuarios pueden enviar sus trabajos a ejecución, y monitorear el estado de los envíos y los resultados. Estos resultados luego son almacenados en una base de datos, de la que pueden extraerse a voluntad, sin necesidad de explícitamente ejecutar los comandos de gLite.

En la Figura 4 se muestra la página para el envío de trabajos, mientras en la Figura 5 se muestran ejemplos de diversos estados de la ejecución de tareas. Una explicación más detallada de este trabajo se presenta en [9]

Figura 3: Entrada al Sistema

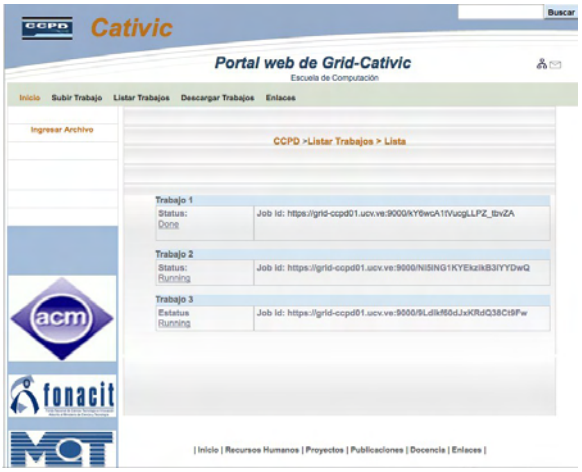

Figura 4: Carga de Trabajos a Cativic

5. AIRES

AIRES (AIR-Shower Extended Simulations) [4] es un software para modelado de partículas desarrollado por S. J. Sciutto, del Departamento de Física de la Universidad Nacional de la Plata en Argentina. Esta escrito completamente en Fortran y consiste en más de 730 rutinas. El software permite simular lluvias de partículas producidas después de la incidencia de rayos cósmicos de alta energía en la atmósfera.

AIRES ofrece un espacio temporal para la propagación de partículas en un entorno realista, donde las características de la atmósfera, el campo geomagnético y la curvatura de la tierra se tienen en cuenta adecuadamente. Un procedimiento de muestreo estadístico es utilizado cuando el número de partículas en la lluvia es muy grande.

AIRES modela el comportamiento de rayos gamma, electrones, positrones, muones, piones, kaones, mesones de ETA, lambda bariones, nucleones y antinucleones.

5.1. Gridización de AIRES

La simulación de lluvias de partículas de alta energía requiere de un uso intensivo de CPU que puede exigir días e incluso semanas de tiempo de procesador. El programa AIRES fue diseñado teniendo en cuenta este hecho, por lo que incluye un mecanismo de “auto-guardado”, que almacena periódicamente en un archivo IDF (Internal Dump File) todos los datos relevantes de la ejecución. En el caso de una falla del sistema, el proceso de simulación puede ser reiniciado a partir del último “auto-guardado”, evitando así la perdida de todo el tiempo de simulación que había sido realizado. El procesamiento que existe entre dos auto-guardados consecutivos es llamado una corrida.

Figura 5: Estado de Ejecución de Tareas

```
Task ejemplo
Primary proton
PrimaryEnergy 150 TeV
TotalShowers 3
End
```

Figura 6: Directivas de IDL

Una tarea típica tiene varios procesos y varias corridas en cada proceso. Por ejemplo, una tarea puede indicar “simular diez lluvias de protones”. Para definir la cantidad, el tipo de partículas y otros parámetros de entrada (energía inicial, etc.) se

utiliza una serie de archivos de entrada en un lenguaje de definición de entrada específico llamado Input Directive Language (IDL). En la Figura 6 se muestra un ejemplo de este tipo de directivas

Para gridizar AIRES se decidió dividir las diferentes simulaciones, de manera que se pueda enviar tareas por separado a cada WN disponible, aprovechando como entrada los archivos IDL de los usuarios. Luego se envía una versión precompilada de los módulos a ejecutar, y se transfieren los archivos de entrada. Esto se muestra en la Figura 7, mediante un extracto del segmento de ejecución de los script del JDL. Los códigos completos de los JDL para gLite, se encuentran en [10].

```
#!/bin/sh
cp /usr/local/bin/airesrc /$HOME/.airesrc
ruta=$PWD
cp PE* $HOME
cd $HOME
mkdir aires
cp PE* aires
cd aires
/usr/local/bin/airesstatus $2 > status.out
STATUS='awk '/currently/ { print $7 }' status.out'
while [ "$STATUS" != "NOT" ]
do
        AUX=$STATUS
        /usr/local/bin/airesstatus $2 > status.out
        STATUS='awk '/currently/ { print $7 }' status.out'
        if [ "$AUX" != "$STATUS" ]
        then
                echo
                echo "Status Actual ----> $STATUS"
        fi
done
/usr/local/bin/airesstatus $2
cd ..
/bin/tar -czvf aires.tar.gz aires
cp aires.tar.gz $ruta
```

Figura 7: Extracto del JDL de Ejecución de AIRES

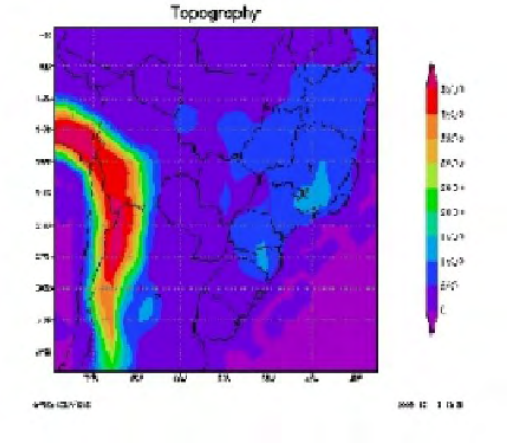

Figura 8: Gráfico de Salida de BRAMS

Para facilitar el uso de la aplicación por parte de los usuarios, estos scripts fueron encapsulados en una aplicación web que se encarga de la ejecución y el monitoreo del estado de AIRES en los Working Nodes. Los detalles de esta aplicación se muestran en la sección 7.

6. BRAMS

BRAMS (Brazilian Regional Atmospheric Modeling System) [3] fue desarrollado por un conjunto de instituciones; ATMET (Atmospheric Meteorological and Environment Technologies), IME/USP (Instituto de Matemática y Estadística de la Universidad de Sao Paulo), IAG/USP (Instituto de Astronomía Geofísicas y Ciencias Estadísticas de la Universidad de Sao Paulo) y CPTEC/INPE (Centro de Predicción del Tiempo y Estudios Climatológicos), con el propósito de crear una nueva versión del modelo RAMS (Regional Atmospheric Modeling System), para ser usada en trópicos.

BRAMS es un modelo de mesoescala; en estos modelos de predicción del tiempo, la resolución horizontal y vertical es suficiente para pronosticar fenómenos meteorológicos. Dichos fenómenos, que a menudo son producto del forzamiento de la topografía o de los litorales, o están relacionados con la convección, presentan algunos de los mayores retos a la hora de hacer el pronóstico. La visibilidad, la turbulencia y el estado del mar pueden variar enormemente en distancias de pocos kilómetros y sus repercusiones pueden ser enormes.

BRAMS genera como salida una serie de parámetros que luego pueden ser usados para graficar las predicciones. En la Figura 8 se muestra uno de los gráficos típicos generados a través de las salidas del sistema.

6.1. Gridización de BRAMS

Al igual que AIRES, BRAMS requiere importantes tiempos de cómputo y recursos computacionales, aún para datos de entrada relativamente pequeños. Esto ha sido un aliciente para la migración de BRAMS a ambientes Grid. Los intentos más importantes se han hecho en el INPE, a través del proyecto GBRAMS, que busca:

- Evaluar la viabilidad del uso de la infraestructura Grid para la ejecución de modelos numéricos meteorológicos.
- Generar tres años de modelaje meteorológico usando el BRAMS. Los datos generados serán para tres regiones de Brasil: Norte, Noreste y Sur/Sureste.
- Comparar tres middleware: Globus, OAR/CIGRI y OURGRID.

La interacción entre el meteorólogo y el Grid es realizada a través de un portal web, cuyo objetivo es permitir el envío de trabajos y la recuperación de los resultados. Cada nuevo trabajo es almacenado en una base de datos. El scheduler está encargado de buscar los trabajos en la base de datos y ejecutarlos en una máquina del Grid. Después de la ejecución, los resultados estarán disponibles en el portal web.

El proyecto GBRAMS está en fase de ejecución, y no contempla el uso de gLite. En la UCV, se decidió hacer pruebas de BRAMS con este middleware para adaptarlo a nuestra plataforma operativa.

La aplicación BRAMS se asume debe estar instalada en los WN del ambiente grid, ya que depende de una serie de librerías y opciones de compilador que dificultan el envío del código fuente o el uso de opciones precompiladas. En el caso de múltiples WN, es el WMS quien debe decidir en qué nodos hará la ejecución. El framgento de JDL que ejecuta el BRAMS en los nodos se muestra en la Figura 9.

```
cat > $JOBS << EOF
Type = "Job";
JobType = "Normal";
Executable = "/bin/sh";
Arguments = "nodo_brams.sh";
InputSandbox = {"BRAMS_$FECHA.tar.gz","nodo_brams.sh"};
StdOutput = "brams.out";
StdError = "brams.err";
OutputSandbox = {"BRAMS_$FECHA.tar.gz","brams.err","brams.out"};
ShallowRetryCount = 1;
EOF
```

Figura 9: Extracto del JDL de Ejecución de BRAMS

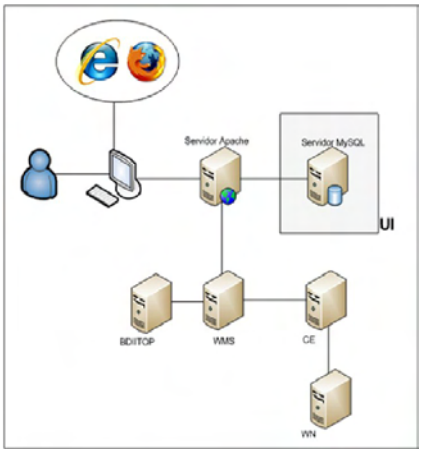

Figura 10: Arquitectura del Ambiente Web del Grid UCV

Luego de realizar la creación del JDL, se procede a enviar el Job al WMS. Al igual que con AIRES, se hace uso del comando glite-wms-Job-submit, especificando con la opción -d el usuario que lo está enviando, y con -o el archivo donde se almacenará el ID del Job.

La salida se almacena en el archivo submit.txt, para luego, haciendo uso del programa awk, extraer del archivo el ID del Job enviado. Al igual que en los casos anteriores, se verifica que la salida del comando glite-wms-Job-submit no haya generado ningún error.

El script ejecutado en el WN se encarga de ejecutar todas las fases necesarias para realizar el modelado. Este script realiza una serie de modificaciones en archivos de configuración necesarios para la ejecución del modelo, durante las fases de preprocesamiento, procesamiento y postprocesamiento. La fase de preprocesamiento es la encargada de tomar los archivos que contienen el estado del tiempo de la atmósfera y transformarlos a un formato que sea legible para BRAMS, mediante el uso de la aplicación grib2dp, que convierte archivos en formato grib y los transforma a un formato RALPH2.

Al tener los archivos en un formato legible para BRAMS, se puede iniciar la fase de procesamiento. El archivo RAMSIN es el encargado de especificarle a BRAMS todos los parámetros necesarios para su ejecución.

Finalmente se realiza el último paso de la fase de procesamiento, la que realiza el pronóstico como tal, utilizando archivos generados por las ejecuciones previas.

Al finalizar el procesamiento, se inicia la fase de postprocesamiento. En esta fase se toman los archivos generados por la fase de procesamiento y se transforman en archivos con extensión .ctl y .grb, estos pueden ser visualizados con la herramienta grads.

Los códigos completos de los JDL para ejecutar BRAMS sobre la platafroma gLite, se encuentran en [10]. En la siguiente sección se muestra el trabajo de aplicación web hecho para AIRES y BRAMS.

7. BRAMS y AIRES Via Web

Tanto para BRAMS como para AIRES se diseñó un portal específico que permitiese el acceso mediante un sitio web. A diferencia del sitio generado para CATIVIC, este sitio se diseñó con la intención de alojar diversas aplicaciones mediante una interfaz de ejecución común. La arquitectura se muestra en la Figura 10.

Figura 11: Formato de Ingreso a BRAMS Web

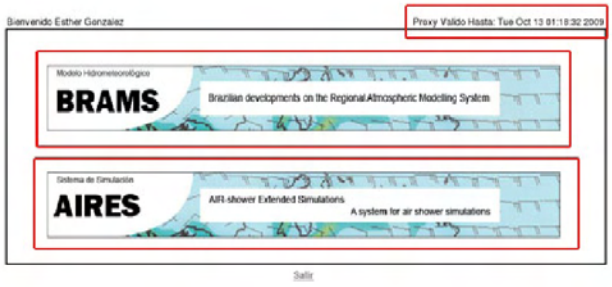

Figura 12: Interfaz Web de BRAMS/AIRES

En esta arquitectura, los aspectos de seguridad y los pasos requeridos para el registro, creación de certificados y otras medidas de validación propias de gLite se dejaron de parte del servidor, por lo que la aplicación web sólo está disponible para los usuarios que previamente hayan sido registrados y con certificados vigentes, como se muestra en la Figura 11.

Con este certificado, ingresa al sitio que se muestra en la Figura 12.

Cada una de las aplicaciones tiene sus script de ejecución, y permisos diferentes, por lo que un usuario podría tener acceso solamente a una de las aplicaciones. En la figura 12, el usuario tiene acceso a las dos aplicaciones. Una vez dentro de alguna de las aplicaciones, se presentan los menús que se muestran en la Figura13.

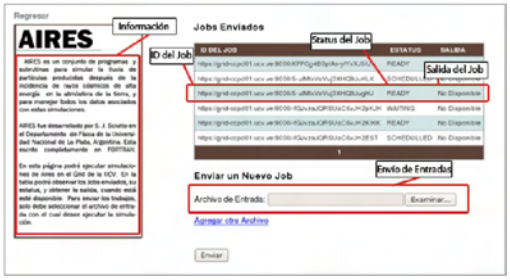

Figura 13: Menú de Tareas de AIRES

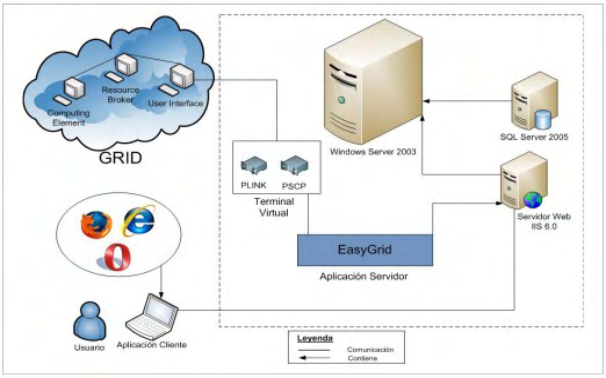

Figura 14: Arquitectura de EasyGrid

En el menú se muestran los estados de los diferentes trabajos enviados, se da la opción de enviar múltiples trabajos, y se permite la descarga de los archivos de salida de cada modelo. En ninguno de los casos se grafica la salida, esta tarea depende del usuario una vez descargados los datos.

8. EasyGrid

Los problemas generados a los usuarios al momento de ejecutar sus aplicaciones, han llevado a diseñar una aplicación que permita utilizar cualquiera de las aplicaciones previamente gridizadas sin que los usuarios deban ejecutar los comandos, crear los scripts, o dominar los aspectos de seguridad y validación de gLite. Desde 2008, se ha trabajado en el desarrollo y prefeccionamiento de EasyGrid [5], un sistema que permite el uso de las aplicaciones del Grid UCV de una forma simple para el usuario, con interfaces basadas en herramientas visuales sencillas.

EasyGrid combina la facilidad de uso de las interfaces basadas en Microsoft .NET y tecnologías de software libre para el back-end. La arquitectura de EasyGrid se muestra en la Figura 14.

EasyGrid se diseñó de forma modular, como se muestra en la Figura 15. Esto evita los problemas asociados a aplicaciones monolíticas. Por ejemplo, para añadir nuevas aplicaciones al portal AIRES/BRAMS, es necesario construir nuevos scripts y modificar el código del servidor. En EasyGrid, el diseño en módulos permite añadir archivos de configuración necesarios para los nuevos scripts de las nuevas aplicaciones, sin que para ello sea necesario modificar otros módulos.

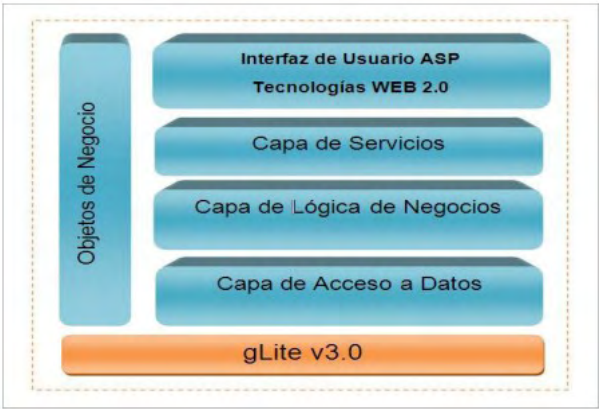

Figura 15: Capas de EasyGrid

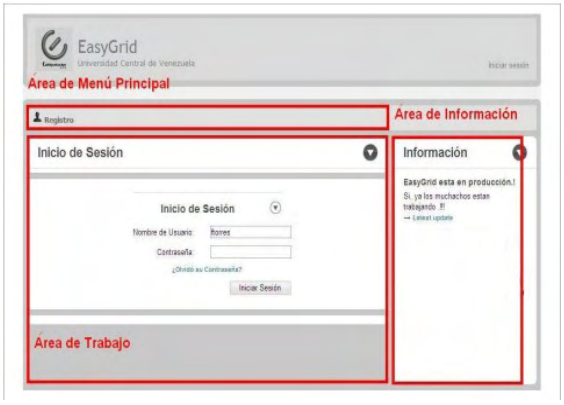

Figura 16: Entrada a EasyGrid

8.1. Funcionalidades de EasyGrid

Las principales funcionalidades del sistema son, a nivel administrativo, las relacionadas con el ingreso de nuevas aplicaciones a la plataforma. Sin embargo, EasyGrid no resuelve los detalles relacionados a la compilación, instalación, o copia de las aplicaciones en los WN, una aplicación en el ambiente EasyGrid se asume instalada en los lugares destino, o se envian los scripts necesarios para la compilación. En cualquier caso, es el WMS quien debe ubicar los servidores y WN adecuados para la ejecución de los requerimientos de los usuarios.

Los usuarios tienen una cantidad de opciones en EasyGrid que no están contempladas en los portales desarrollados para CATIVIC, AIRES y BRAMS, entre ellas:

Tareas: el usuario puede configurar tareas para su ejecución, revisar el estado de sus trabajos actuales y anteriores, añadir aplicaciones, y añadir comentarios y descripciones personalizadas a sus trabajos

Resultados: los resultados de la ejecución (exitosos o no) pueden ser revisados y descargados directamente de la aplicación, pueden almacenarse en la Base de Datos de EasyGrid para posterior análisis, y pueden ser eliminados o modificados.

Información Personal: cada usuario tiene la potestad de administrar individualmente su información, claves, y preferencias de ejecución. Sin embargo, lo referente a los certificados requeridos para ejecución, son creados y asignados por el administrador del sistema.

8.2. Interfaces de usuario

Las principales interfaces de usuario se muestran a continuación. Una vez registrados los usuarios, se autentican para comenzar una sesión de trabajo (Figura 16).

Hay dos vistas principales.: la vista de usuario y la de administrador, que le permite manejar las cuentas de usuario, asignar o renovar certificados, determinar permisos y otras operaciones administrativas sobre usuarios y aplicaciones. La vista de administrador se muestra en las Figuras 17 y 18.

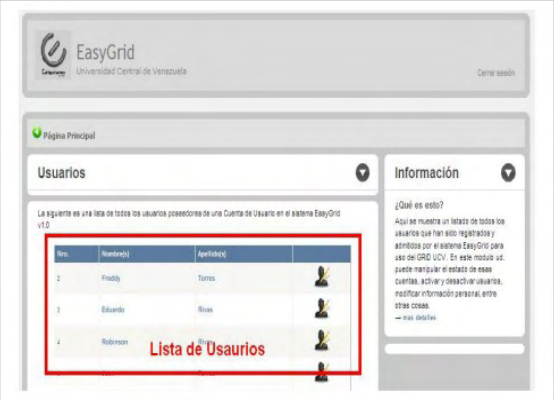

Figura 17: Vista de Administración de Usuarios

Figura 18: Vista de Administración de Aplicaciones

Con respecto a los usuarios, las vistas principales se refieren al manejo de los trabajos enviados al grid. Los trabajos y sus estados se presentan en forma gráfica, como se muestra en la Figura 19.

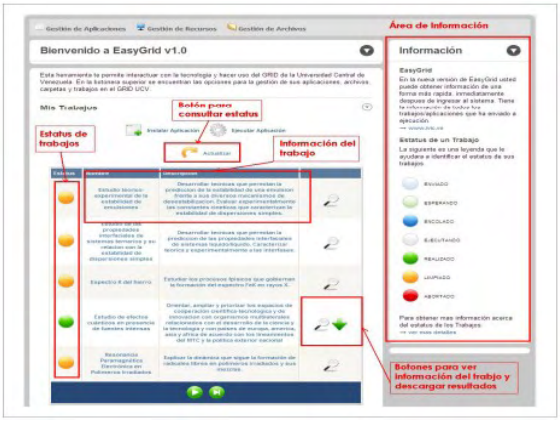

Figura 19: Estado de las Tareas del Usuario

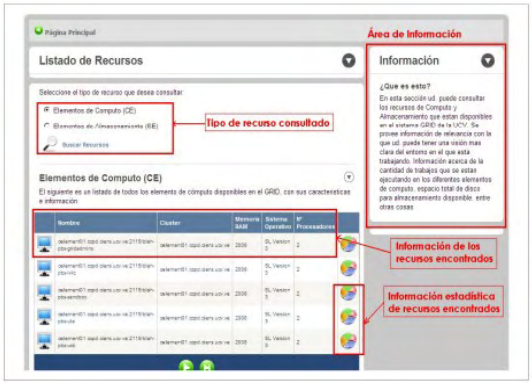

Figura 20: Recursos del Grid

El usuario también puede ver los recursos disponibles en el Grid. En ese caso, puede ver toda la información que gLite acceda a través de sus BDII. No solamente se muestran los recursos con sus descripciones, también pueden mostrarse estadísticas de uso y características de los diferentes recursos involucrados (Figura 20).

El resultado de la ejecución es monitoreado constantemente. Cuando el estado del job es done (terminado) el usuario puede descargar los resultados, que además se almacenan en la base de datos interna de EasyGrid, lo que permite su consulta posterior sin necesidad de re-ejecutar la tarea. La página de descarga se muestra en la Figura 21.

EasyGrid está en estado de prototipo, y se han hecho varias modificaciones a partir de la fecha original de creación en 2008.

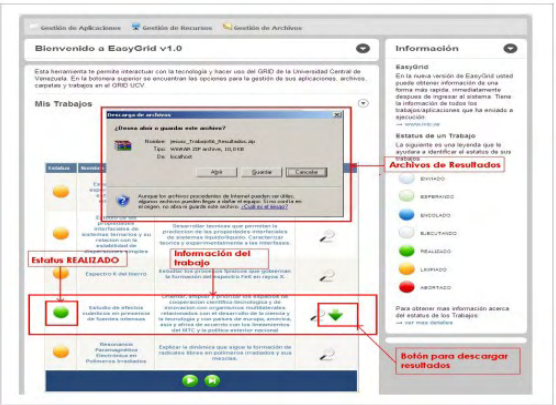

Figura 21: Descarga de Resultados

9. Conclusiones

Hasta la fecha, el Centro de Computación Paralela de la Universidad Central de Venezuela ha trabajado en la instalación y mantenimiento de una infraestructura Grid basada en gLite y Scientific Linux, que ha servido a los propósitos de dominar esta tecnología y formar a los administradores de sistemas que este tipo de plataformas requieren. Sin embargo, la gridización de software paralelo ha sido una tarea compleja y laboriosa, que no ha marchado al ritmo deseado.

Una de las principales razones, es la falta de soporte adecuado para el middleware gLite. Por otro lado, el grupo de trabajo ha variado con el paso del tiempo, lo que implica continuo re-aprendizaje y dificulta darle continuidad a los proyectos.

A pesar de estos inconvenientes, se ha logrado generar un grupo de trabajo que esperamos pueda interactuar con otros centros e instituciones, y sobre todo, poder portar al grid nuevas aplicaciones de uso común para la comunidad científica.

Las aplicaciones gridizadas han funcionado de forma satisfactoria una vez instaladas. Esto demuestra la factibilidad y posibilidades de la migración de otro tipo de programas, especialmente en el dominio de las aplicaciones de petróleo y tratamiento de imágenes.

Finalmente, el desarrollo de EasyGrid, aún en etapa experimental, promete ayudar en el objetivo de masificar el uso del grid entre las diversas comunidades de usuarios.

10. Trabajos Futuros

Hay una serie de trabajos que es necesario realizar para consolidar el Grid de la UCV como plataforma plenamente operativa. Entre las principales tareas a futuro se tienen:

- Adaptación de nuevas aplicaciones, y generación de una metodología adecuada para simplificar esta tarea
- Incorporación de la UCV a un entorno Grid nacional e internacional a través de la red de alta velocidad CLARA
- Mejoras del sistema EasyGrid para incorporar nuevas funcionalidades a partir de las experiencias recogidas.
- Realización de análisis de desempeño detallados para cada aplicación llevada al Grid. En este momento, las aplicaciones no han sido evaluadas en detalle en los aspectos de aceleración o mejoras del rendimiento.

Finalmente, es importante destacar que la UCV debe incorporarse al Grid de Venezuela, aún en etapa de consolidación. Para este paso, se requiere la conjunción de esfuerzos en las principales universidades nacionales. Una vez incorporados a este Grid nacional, se realizarán pruebas de desempeño de las aplicaciones mostradas en este trabajo.

11. Agradecimientos

El Centro de Computación Paralela y Distribuida, agradece al Consejo de Desarrollo Científico y Humanístico de la UCV el financiamiento de parte de este trabajo mediante el proyecto PI-03-7398-2008/1. El entrenamiento del personal del CCPD fue fundamentalmente soportado (dentro y fuera de Venezuela) gracias al proyecto EELA2 de la Comisión Europea, con el invaluable apoyo de la Universidad de Los Andes (ULA), en las personas de Luis Núñez, Herbert Hoeger y Gilberto Díaz.

12. Referencias

[1] I. Foster, C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Capítulo 2, Morgan-Kaufman, 1999.
[2] F. Ruette, M. Sanchez, G. Martorell, C. Gonzalez, R. Añez, A. Sierralta, L. Rincon, C. Mendoza. "CATIVIC: Parametric quantum chemistry package for catalytic reactions", International Journal of Quantum Chemistry, 96 (4), 2004.
[3] S.R. Freitas, K. Longo, M. Silva Dias, P. Silva Dias et al., "Brazilian developments on the Regional Atmospheric Modelling System (CATT-BRAMS). Part 1: Model description and evaluation", Atmospheric Chemistry and Physics, volume 7, number 3, 2007.
[4] S. J. Sciutto. "Current status of the AIRES air shower simulation system", Proceedings of th 29th International Cosmic Ray Conference Pune, India, 2005.
[5] E. Rivas, F. Torres, J. Parada, R. Rivas, "Easygrid: Sistema Web para Facilitar el Acceso a Recursos en Plataformas Grid", Proceedings del XXI Workshop en Sistemas Distribuidos y Paralelos de la Soc. Chilena de Ciencias de la Computación, Punta Arenas, Chile, noviembre 2008.
[6] E. Laure, S.M. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, F. Hemmer, A. Di Meglio, A. Edlund, "Programming the Grid with gLite", Computational Methods in Science and Technology, 12(1), 2006.
[7] EELA2: E-science grid facility for Europe and Latin America, proyecto sustentado por la Comisión Europea, http://www.eu-eela.eu/.
[8] D. Espinoza, J. Parada, R. Rivas. "Advances on Cativic Gridification: A Quantum Chemistry System", Proceedings of the 2nd EELA2 Int. Conference, Choroni, Venezuela, November 2009.
[9] D. Espinoza, "Implantación de una plataforma Grid basada en gLite y Scientific Linux para el Centro de Computación Paralela y Distribuida", Trabajo de Grado, Escuela de Computación, Universidad Central de Venezuela, Septiembre 2009.
[10] E. González, "Adecuación a ambientes Grid de dos aplicaciones de Computación de Alto Desempeño en el dominio de Física de Altas Energías y Simulación Atmosférica", Trabajo de Grado, Escuela de Computación, Universidad Central de Venezuela, Septiembre 2009.
[11] B. Wilkinson, "Grid Computing. Techniques and Applications", Chapman & Hall / CRC Computational Science Series, Capítulo 9, 2010.
[12] K. Nolan, "Approaching the challenge of Grid-enabling applications", Open Source Grid & Cluster Conference, Oakland, CA, 12-16 Mayo, 2008.
[13] Condor Project Homepage, http://research.cs.wisc.edu/condor/.
[14] Abramson D. Buyya R. and Giddy J., "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", Proceedings of the HPC ASIA 2000, the 4th. International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, IEEE Computer Society Press, USA, 2000.
[15] Henri Casanova, Graziano Obertelli, Francine Berman, and Rich Wolski. "The apples parameter sweep template: User-level middleware for the grid", Proceedings of SuperComputing, 2000.
[16] http://www.csse.monash.edu.au/~davida/griddles/index.htm.
[17] Globus Grid Software, http://www.globus.org/.
[18] UNICORE (Uniform Interface to Computing Resources), http://www.unicore.eu/.
[19] N.T. Karonis, B. Toonen, I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface", Journal of Parallel and Distributed Computing - Special issue on computational grids. Volume 63 Issue 5, May 2003.
[20] F. Berman, G. Fox, A. Hey, "Grid Computing: Making the Global Infrastructure a Reality", Wiley, 2003.
[21] Arda Metadata Catalogue Project, http://amga.web.cern.ch/amga/.