ESPECIFICACIÓN PARA LA INTEGRACIÓN DE SISTEMAS DE GRAN ESCALA, BASADO EN DRMAA

AUTOR
JOHN ALEXANDER DÍAZ GONZÁLEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2013

ESPECIFICACIÓN PARA LA INTEGRACIÓN DE SISTEMAS DE GRAN ESCALA, BASADO EN DRMAA

AUTOR
JOHN ALEXANDER DÍAZ GONZÁLEZ

Trabajo de grado para optar por el título de Ingeniero de Sistemas

DIRECTOR
PH.D. CARLOS JAIME BARRIOS HERNÁNDEZ

CODIRECTOR
PH.D. YIANNIS GEORGIOU

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2013

DEDICATORIA

A Dios y a la vida, por darme la oportunidad de recorrer un camino lleno de enseñanzas, en compañía de personas que han ido dejando huella en el recuerdo de mi vida.

A mis padres, Prudencio Díaz Crudy y María Teresa González Ardila, por su amor, comprensión, dedicación, confianza y por siempre tener las palabras precisas en los momentos indicados.

A mis abuelos, hermanos y aquellos familiares, que de alguna u otra forma siempre han estado pendientes de mí, apoyándome en procura de que todo me salga bien.

Y a todas esas personas que poco a poco y sin darme cuenta, se han ido convirtiendo en parte de mí, pues todos esos momentos compartidos han ayudado a que me forme como una persona íntegra y amigable.

John Alexander Díaz González

AGRADECIMIENTO

Al **Ph.D. Carlos Jaime Barrios Hernández**, director del proyecto, por su acompañamiento, observaciones y por su interés en el desarrollo del proyecto.

Al **Ph.D. Yiannis Georgiou**, codirector del proyecto, por aportar con su conocimiento, y estar siempre dispuesto a aclarar las dudas que surgieron en la realización del proyecto.

Al **Ing. Juan Carlos Escobar Ramírez**, administrador de los laboratorios de la Escuela de Ingeniería de Sistemas e Informática, por facilitar el acceso a recursos necesarios para el desarrollo del presente proyecto.

Al grupo de desarrolladores del **INRIA** en Grenoble, Francia, por facilitar el acceso a las fuentes de la herramienta CiGri, que actualmente se encuentra en desarrollo.

Al grupo de ingenieros de Bull Inc. por la colaboración con el entendimiento de la herramienta planificadora de trabajos SLURM.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

GLOSARIO

Middleware: Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

Kernel: Es un software que constituye la parte más importante del sistema operativo, facilitando a los distintos programas, acceso seguro al hardware del computador o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

Portabilidad: Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra.

Escalabilidad: Se puede definir como la capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

RESUMEN

TÍTULO: ESPECIFICACIÓN PARA LA INTEGRACIÓN DE SISTEMAS DE GRAN ESCALA, BASADO EN DRMAA[*]

AUTOR: JOHN ALEXANDER DÍAZ GONZÁLEZ[**]

PALABRAS CLAVE: Cluster, Computación Grid, Computación Cloud, Estándar

DESCRIPCIÓN: En los últimos años, la investigación ha experimentado un gran crecimiento, debido a nuevas herramientas, que permiten explorar con mayor profundidad los aspectos de cada proyecto investigativo. El aumento en la cantidad de datos que deben ser analizados, y la necesidad de que los procesos se realicen en poco tiempo, hacen necesaria la utilización de un gran poder computacional. Para suplir la demanda de cómputo avanzado, se recurre a la Computación de Alto rendimiento (HPC), mediante el uso de Clusters, Grid Computing y Cloud Computing.

Actualmente, los centros de investigación realizan aplicaciones para el análisis de datos que se adapten al Sistema Administrador de Recursos Distribuidos (DRMS), que se encuentre implementado en su plataforma de cálculo científico. La necesidad de grandes cambios sobre la aplicación, para poder ejecutarla en otros DRMS, restringe la portabilidad de la misma. Es por esto que DRMAA, especifica mecanismos estándares para el desarrollo de aplicaciones que puedan ser ejecutadas en diferentes DRMS.

El presente trabajo, pretende realizar un análisis de los sistemas planificadores de recursos, presentes en la infraestructura de cálculo avanzado de la Universidad Industrial de Santander, y evaluar la viabilidad de implementar el estándar DRMAA, para facilitar el desarrollo de aplicaciones que puedan ser portables entre los recursos integrados de la plataforma.

[*] Trabajo de grado en la modalidad de Investigación.
[**] Facultad de ingenierías Físico-mecánicas. Escuela de ingeniería de sistemas e informática. Director: PhD Carlos Jaime Barrios Hernández. Codirector: Ph.D. Yiannis Georgiou.

ABSTRACT

TITLE: SPECIFICATION FOR THE INTEGRATION OF LARGE SCALE SYSTEMS, BASED ON DRMAA[*]

AUTHOR: JOHN ALEXANDER DÍAZ GONZÁLEZ[**]

KEYWORD: Cluster, Grid computing, Cloud computing, Standard

DESCRIPTION: Over the past few years, research has experienced tremendous growth, due to new tools that allow you to explore in more depth the aspects of each research project. The increase in the amount of data that must be analyzed and the need for processes to be carried out in a short time require the use of a large computational power. To supplement the demand for an advanced computing, we resort to the High Performance Computing (HPC) through the use of Clusters, Grid Computing and Cloud Computing.

Currently, research centers perform applications for the analysis of data that will adapt to the Distributed Resource Management System (DRMS) which is implemented in its scientific computing platform. The need for great changes on the application to be able to run in other DRMS restricts the portability of it. Due to this fact, DRMAA specifies standard mechanisms for the development of applications that can be implemented in different DRMS.

The present work, attempts to make an analysis of the batch scheduler systems which are present in the infrastructure of advanced computation at the Universidad Industrial de Santander, and to evaluate the viability of implementing the DRMAA standard, to facilitate the development of applications that can be portable between the resources of the integrated platform.

[*] Undergraduate Final Project, Research modality

[**] Faculty of Physical-Mechanical Engineering, Systems Engineering and Computer Science School.
Director: PhD Carlos Jaime Barrios Hernández. Codirector: Ph.D. Yiannis Georgiou.

INTRODUCCIÓN

El notable desarrollo que ha experimentado la tecnología en las últimas décadas, ha permitido que investigadores de todas las partes del mundo afronten nuevos retos de investigación en diversas áreas de la ciencia. Y es que los avances tecnológicos en las herramientas, les ha permitido a los investigadores recopilar una mayor cantidad de datos, con un alto grado de precisión y fiabilidad, para que estos puedan obtener información relevante para el desarrollo de los proyectos. Dada la abrumadora cantidad de datos recolectados, y la necesidad de procesarlos en poco tiempo para tomar decisiones con base en la información obtenida, se hace necesaria la implementación de soluciones computacionales que soporten dichas demandas de procesamiento.

Desde la ciencia de la computación se ha apoyado el desarrollo de estos proyectos de investigativos, mediante el diseño e implementación de modelos que permiten optimizar el trabajo realizado por los computadores, disminuyendo el tiempo general de procesamiento de una aplicación. Entre estos modelos, se puede encontrar el diseño de supercomputadores, la conformación de Clusters de computadores con recursos disponibles en la organización, el agrupamiento de recursos geográficamente distribuidos y pertenecientes a diferentes organizaciones interesadas en formar una Grid computacional y la contratación de recursos bajo el esquema de computación Cloud.

El desarrollo de aplicaciones por parte de la comunidad de investigadores, está marcado por el esquema de computación avanzada que se esté utilizando, pues las aplicaciones deben responder a las especificaciones del ambiente en el que son ejecutadas y al Sistema Administrador de Recursos Distribuidos implementado allí. Por esto, estas aplicaciones no pueden ser fácilmente migradas entre las diferentes plataformas de cálculo, pues se debe realizar un gran esfuerzo

al reestructurar la aplicación para que se adapte al nuevo medio de ejecución. Por esto, en el Global Grid Forum, se estableció la especificación del estándar DRMAA, con el que se pretende poder realizar aplicaciones que mediante la implementación de una librería, puedan ser fácilmente portables entre los diferentes Sistemas Administradores de Recursos Distribuidos, que cuenten con la implementación del estándar DRMAA.

En el presente trabajo se busca analizar la especificación del estándar DRMAA, para los Sistemas Administradores de Recursos Distribuidos, implementados en las diferentes estructuras computacionales de cálculo avanzado que se encuentran en el campus universitario, esto con el fin de poder definir la mejor forma de integrar dichos recursos. El desarrollo del documento se encuentra organizado de la siguiente manera: en la sección 1, se encuentra la especificación del proyecto, entre lo que está el planteamiento del problema, la justificación y los objetivos del proyecto; en la sección 2, se expone el marco teórico con el contexto de las arquitecturas para el desarrollo de computación de alto rendimiento, y las herramientas utilizadas en el desarrollo del proyecto; en la sección 3, se hace un repaso por el estado del arte, exponiendo las versiones de las herramientas implementadas en la infraestructura de cálculo avanzado de la UIS, junto a su nivel de integración entre ellas; en la sección 4, se muestra la arquitectura planteada para la integración de recursos en la plataforma de cómputo avanzado de la UIS; la sección 5, expone el desarrollo de la investigación y los resultados obtenidos; en la sección 6, están las conclusiones surgidas del desarrollo del proyecto; finalmente en la sección 7, quedan las recomendaciones.

1 DESCRIPCIÓN DEL PROYECTO

1.1 PLANTEAMIENTO DEL PROBLEMA

Actualmente, el nivel de complejidad de la investigación científica que se realiza alrededor del mundo ha ido incrementando, es por esto que la comunidad de investigadores día a día demanda una mayor capacidad de procesamiento. Dado que las simulaciones que se están realizan en cada laboratorio son de vital importancia para entender y solucionar problemas concernientes a toda la humanidad, se ha realizado un gran esfuerzo por parte de entidades gubernamentales y del sector privado que realizan considerables inversiones monetarias en la creación de supercomputadores. Aunque los supercomputadores son una gran solución para la demanda de procesamiento, son demasiado costosos, y por esto, no todos los laboratorios o unidades investigativas tienen la disposición económica para acceder a uno que pueda suplir por completo sus necesidades.

La alta disponibilidad de recursos computacionales en las unidades investigativas y la dificultad en algunos casos de acceder a recursos económicos para la adquisición de supercomputadores, ha conllevado a buscar una estrategia que busque mitigar las demandas de procesamiento por medio de la interconexión de dichos recursos en una solución conocida como Cluster. Aunque este enfoque computacional ha aliviado un poco la necesidad de los investigadores, no se toma como una solución definitiva, puesto que la complejidad de sus simulaciones sigue aumentando y cada día ellos se ven más limitados para concluir sus investigaciones en tiempos aceptables.

Un enfoque planteado para suplir las grandes demandas de procesamiento, es la unión de Clusters bajo un ambiente Grid que permita compartir los recursos

dispuestos en cada uno de ellos, y obtener de esta forma una plataforma robusta con grandes prestaciones. Aunque esta es una forma de solucionar el problema de demanda de procesamiento, surge ahora el inconveniente con la heterogeneidad de los recursos dispuestos en la plataforma Grid.

1.2 JUSTIFICACIÓN

La integración de componentes heterogéneos en sistemas de gran escala es un problema abierto que involucra el reconocimiento de recursos y la interconexión de componentes, sin afectar el rendimiento y políticas de administración de recursos y trabajos que se ejecutan sobre las plataformas.

La disposición de múltiples recursos de cómputo contribuidos por los diferentes grupos de investigación de la Universidad industrial de Santander[1], para la integración de la plataforma de computación de alto rendimiento, GridUIS-2[2], ofrece mejoras en las prestaciones de la infraestructura de cómputo de la universidad. La particularidad de los proyectos desarrollados por los centros investigativos da una amplia heterogeneidad en los recursos unidos a la infraestructura, GridUIS-2.

Dadas las características heterogéneas de los recursos es necesaria la integración de los Cluster distribuidos entre los laboratorios de la universidad. Esta integración se realiza para mejorar la administración total de los recursos bajo una única herramienta que permita tener un control global de los trabajos que se ejecutan en la plataforma y a su vez pretende que los usuarios de los laboratorios participantes en el proyecto tengan una forma sencilla de utilizar la plataforma.

[1] Universidad Industrial de Santander, http://www.uis.edu.co/ [18]
[2] Proyecto que ofrece una plataforma de cómputo avanzado para investigadores de la Universidad Industrial de Santander, http://grid.uis.edu.co/ [19]

1.3 OBJETIVOS

1.3.1 Objetivo general

Proveer una especificación de implementación basada en DRMAA, para la administración de recursos y trabajos en sistemas de gran escala.

1.3.2 Objetivos específicos

- Caracterizar la especificación DRMAA versión 1 y 2, para proponer una documentación de implementación de plataforma.

- Evaluar la eficiencia de la implementación del estándar DRMAA realizando una comparación con el desempeño del enfoque Best Effort de OAR.

- Evaluar la eficiencia de la implementación del estándar DRMAA con SLURM en ambientes Grid Computing y Cloud.

- Proponer un mecanismo que permita la integración del manejador de recursos Clusters, SLURM con ambientes de computación Grid y Cloud.

1.4 VIABILIDAD DEL PROYECTO

El proyecto reúne características y condiciones técnicas que aseguran el cumplimiento de sus objetivos ya que:

- El software con el que se trabajará en el desarrollo del proyecto es de carácter libre y no repercutirá en gastos adicionales relacionados a licencias de uso o costos por el estilo.

- Actualmente el grupo de Supercomputación y Cálculo Científico[3] de la UIS (SC3), dispone de Clusters tanto académicos como de producción en los que se procederá a realizar las pruebas pertinentes del desarrollo del proyecto.

- Se cuenta con el apoyo del Ph.D. Yiannis Georgiou quien actualmente se encuentra trabajando con Bull Inc[4].

- El proyecto se encuentra asociado al grupo de SC3 por lo que se cuenta con el apoyo de ingenieros que están dispuestos a colaborar con orientación en temas relacionados al desarrollo del proyecto.

1.5 IMPACTO

La realización e implementación del siguiente proyecto permitirá aumentar la capacidad de cómputo con la que se cuenta en cada uno de los laboratorios, sin la necesidad de realizar grandes inversiones económicas para la adquisición y el mantenimiento de grandes maquinas dedicadas. El proyecto se enfoca en los siguientes puntos:

- Integrar los recursos computacionales que actualmente se encuentran dispersos, bajo una infraestructura que permita obtener el máximo provecho de ellos sin perturbar el trabajo que se realiza localmente.

- Aumentar la capacidad de procesamiento para las unidades y laboratorios que decidan compartir sus recursos con el proyecto y de esta forma beneficiarse del total de los recursos en el momento que necesiten realizar sus cálculos.

[3] Grupo de administración y soporte para la infraestructura de cálculo científico de la Universidad Industrial de Santander, http://sc3.uis.edu.co/ [21]
[4] Empresa europea líder en el mercado digital, http://www.bull.com/ [20]

- Establecer una administración de los componentes de forma centralizada descargando de trabajo a los investigadores que actualmente tienen que administrar sus propios centros, por lo que destinan tiempo a actividades que no son propiamente de investigación en su área.

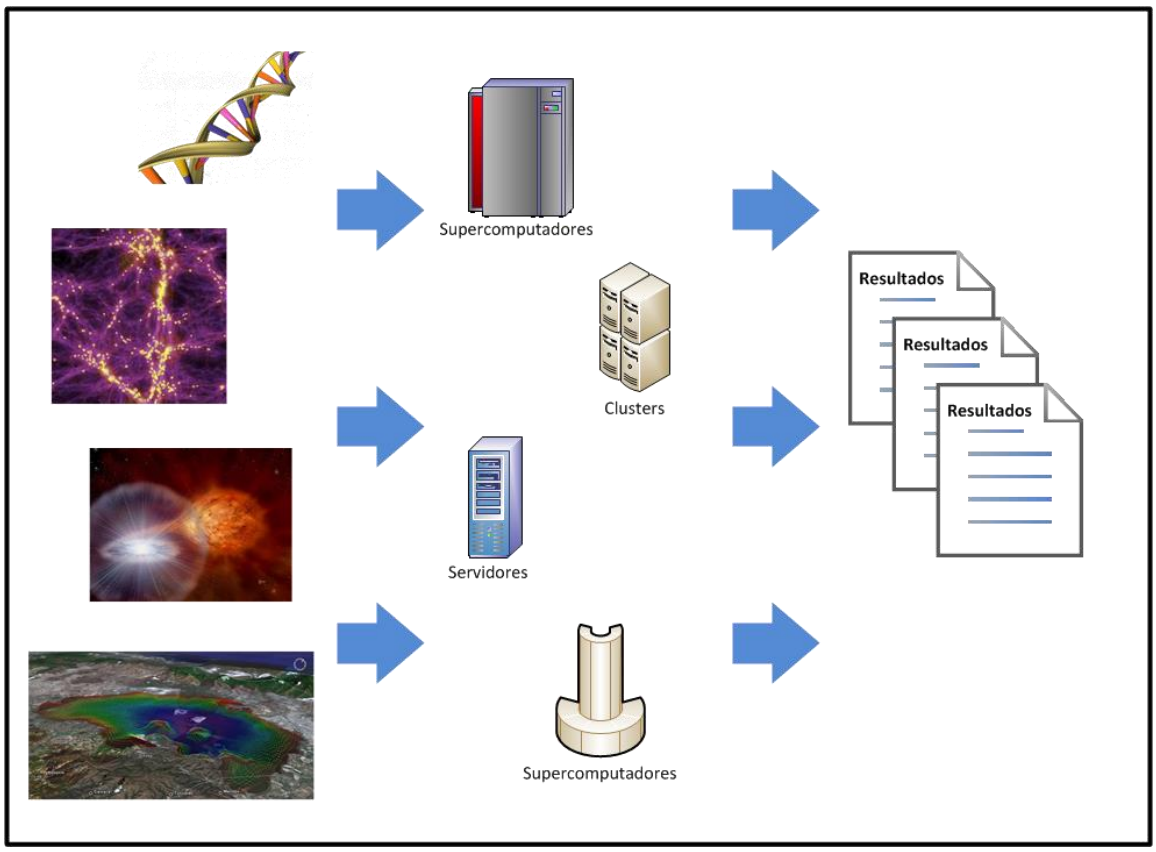
2 MARCO TEÓRICO

2.1 COMPUTACIÓN DE ALTO RENDIMIENTO

El desarrollo de la investigación ha estado marcado históricamente por la necesidad de herramientas que faciliten y agilicen el trabajo de los científicos. Actualmente, en los grupos de investigación, las empresas y las instituciones educativas, se han desarrollado sofisticadas aplicaciones computacionales que requieren de una alta capacidad de procesamiento para obtener resultados en el tiempo deseado. La respuesta a esta demanda computacional, se encuentra en la computación de alto rendimiento (*High Performance Computing, HPC*) también denominada Supercomputación, y caracterizada por proporcionar una velocidad de procesamiento considerablemente mayor a la que se puede obtener en los computadores convencionales más potentes del momento.

Alguna de las definiciones de Supercomputación están dadas por Neil Lincoln, quien dice que “Supercomputador es aquel sistema que está tan solo una generación por detrás de los requerimientos de cálculo de las investigaciones más avanzadas en la ciencia y la ingeniería” [1]. Otra definición está dada por A. J. Forty, para quien éste término está dirigido a aquellas máquinas que alcanzan un rendimiento significativamente superior al esperado por las tecnologías de sus días.

Figura 2-1, Uso de la computación de Alto Rendimiento



Fuente: Autor

Desde la década de los 70's, con la aparición de la Cray1[5], diseñada por Seymour Cray, y los ingenieros de *Control Data Corporation (CDC)*, y gracias al vertiginoso avance de la tecnología, se han planteado diferentes estrategias para realizar computación de alto rendimiento, que supla las necesidades de procesamiento demandadas actualmente. Entre las opciones de computación de alto rendimiento podemos encontrar:

- Sistemas de computación paralela.
- Sistemas Cluster.
- Sistemas Grid.

[5] El Cray-1 fue un supercomputador diseñado por Seymour Cray para Cray Research.

- Sistemas Cloud.

En los últimos años, la definición de computación de alto rendimiento ha cambiado dramáticamente. En 1998, un artículo publicado en el *Wall Street Journal* y titulado "*Attack of the killer micros*" describe como los sistemas de computación compuesto de muchos pequeños procesadores de bajo costo pueden hacer que los grandes supercomputadores se conviertan en obsoletos. [2]

2.2 COMPUTACIÓN CLUSTER

Un Cluster de computadores, es el agrupamiento de recursos informáticos interconectados mediante una red de baja latencia, que de forma conjunta ejecutan una serie de tareas en paralelo. Este sistema Cluster se muestra ante el usuario final como un único sistema con grandes prestaciones. La principal característica de un Cluster, es la unión de los diversos recursos dispuestos en cada uno de los computadores, para conformar una solución de bajo costo y alta escalabilidad para aquellas aplicaciones científicas y de ingeniería con grandes demandas de procesamiento y memoria.

Hacia 1955, Gene Amdahl, comenzó a realizar trabajos teóricos para tratar de maximizar los parámetros de desempeño en las arquitecturas computacionales del momento. A partir de ese instante, muchas personas comenzaron a investigar la forma de optimizar el rendimiento de computadores con arquitecturas paralelas.

A finales de los 70's, y principios de los 80's, aparecen las redes de computadores, y con ellas, la posibilidad de compartir no solo los recursos de almacenamiento, sino también el poder de cómputo para crear un procesamiento en paralelo. En 1977, el primer Cluster comercial fue *ARCNet*, desarrollado por la corporación *DataPoint*. A partir de aquí, una serie de productos popularizaron el concepto, hasta la puesta en marcha del proyecto *Beowulf*, en 1994, que implicaba

la interconexión en red local de computadores estándar, y gestionaba cómo estos interactuaban entre sí. La idea un éxito tan grande que incluso fue adoptada por la NASA. [3]

2.2.1 Características

- Un Cluster consta de 2 o más nodos conectados entre sí por un canal de comunicación.

- Cada nodo únicamente necesita un elemento de proceso, memoria y una interfaz para comunicarse con la red del Cluster.

- Los Cluster necesitan de software especializado, ya sea a nivel de aplicación o a nivel de núcleo.

- Todos los elementos del Cluster trabajan para cumplir una funcionalidad conjunta, sea la que sea. Es la funcionalidad la que caracteriza el sistema.

Los Clusters se pueden clasificar según tres aspectos básicos, el rendimiento, el balanceo de carga y la disponibilidad; Si bien, todos estos aspectos se pueden lograr con un Cluster, al optimizar uno, se pierden características de los otros, por lo tanto, según la aplicabilidad de los Clusters se han desarrollado diferentes líneas tecnológicas. [4]

- **Cluster de Alto Rendimiento (*HP, High Performance*):**
  Esta línea surge frente a la necesidad de supercomputación para determinadas aplicaciones, lo que se persigue es conseguir un gran número de máquinas individuales que actúen como una sola máquina muy potente, este tipo de Cluster se aplica mejor en problemas grandes y

complejos que requieren una cantidad enorme de potencia computacional. En general este está enfocado hacia las tareas que requieren gran poder computacional, grandes cantidades de memoria, o ambos a la vez, teniendo en cuenta que las tareas podrían comprometer los recursos por largos periodos de tiempo.

- **Cluster de Alta Disponibilidad (*HA, High Availability*):**

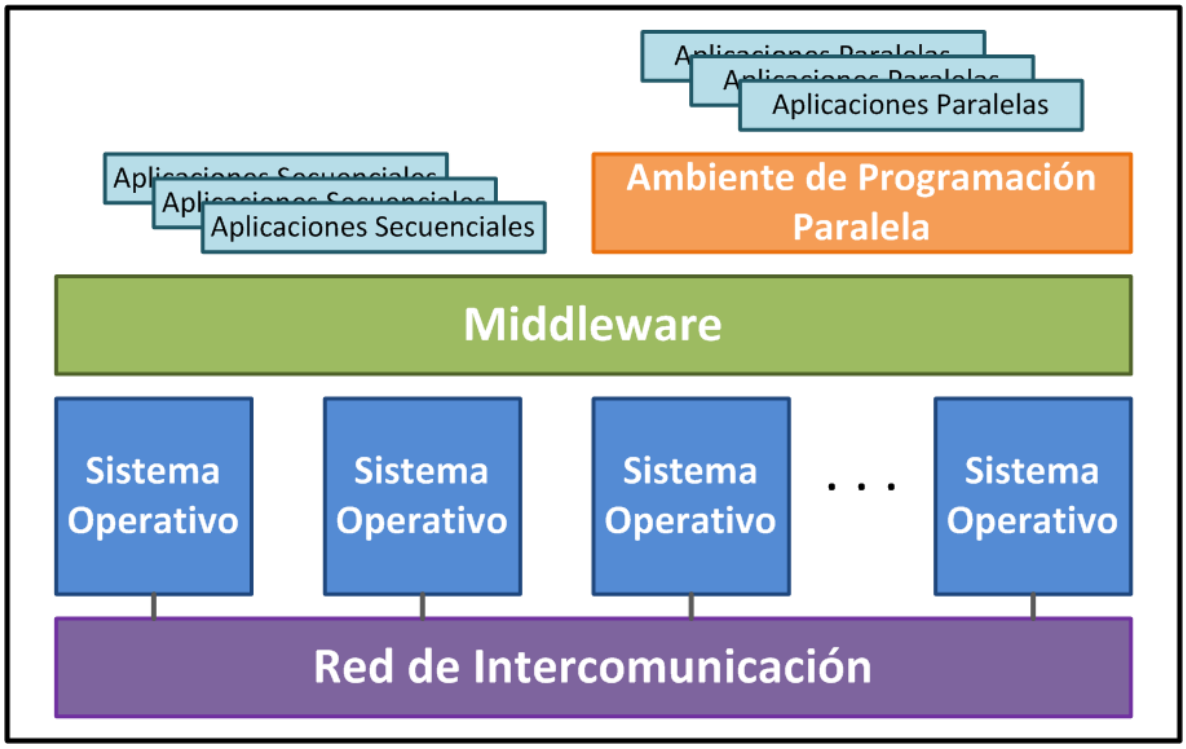
  El objetivo aquí es la máxima disponibilidad de servicios y el rendimiento sostenido lo cual se puede lograr con el mantenimiento de servidores que actúen entre ellos como respaldos de la información que sirven. La flexibilidad y robustez que proporcionan este tipo de Clusters, los hace necesarios en ambientes de intercambio masivo de información, almacenamiento de datos sensibles y allí donde sea necesaria una disponibilidad continua del servicio ofrecido.

- **Cluster de Alta Confiabilidad (*HR, High Reliability*):**

  Este tipo de tecnología de Clusters, es el destinado al balanceo de carga. Surge el concepto de Cluster de servidores virtuales, el cual permite que un conjunto de servidores de red compartan la carga de trabajo y de tráfico de sus clientes, aunque aparezca para estos clientes como un único servidor. Al balancear la carga de trabajo en un conjunto de servidores, se mejora el tiempo de acceso y la confiabilidad. Además, como es un conjunto de servidores el que atiende el trabajo, la caída de uno de ellos no ocasiona la caída total del sistema. Este tipo de servicio es de gran valor para las compañías que trabajan con grandes volúmenes de tráfico y trabajo en sus web.

2.2.2 Componentes

Figura 2-2, Componentes de la computación Cluster



Fuente: http://grid.uis.edu.co/index.php/Seminarios_Internos

- **Nodos:**

  Son los recursos computacionales sobre los cuales se realiza el cálculo de las aplicaciones, estos recursos pueden ser simples computadores de escritorio, sistemas multiprocesador o estaciones de trabajo.

- **Sistemas operativos:**

  Deben ser un sistema operativo de fácil uso y acceso, además este debe permitir múltiples procesos y la conexión de múltiples usuarios.

- **Conexiones de red:**

  Los recursos del Cluster necesitan un canal de comunicación entre ellos para el intercambio de información, estos nodos pueden ser interconectados mediante una simple red *Ethernet*, o se puede utilizar

tecnologías especiales de alta velocidad como *Fast Ethernet*, *Gigabit Ethernet*, *Myrinet*, *Infiniband*, *SCI*.

- **Middleware:**
  El *middleware* es un software que actúa generalmente entre el sistema operativo y las aplicaciones de usuario con el fin de proveer una única interfaz de acceso al sistema, la cual genera la sensación al usuario que está utilizando un único computador muy potente.

- **Herramientas para la optimización y mantenimiento del sistema:**
  Migración de procesos, puntos de control y reenvío de trabajos, balanceo de carga, tolerancia a fallas, etc.

- **Escalabilidad:**
  Debe poder detectar automáticamente nuevos nodos conectados al Cluster para proceder a su utilización.

- **Ambientes de programación paralela:**
  Los ambientes de programación paralela permiten implementar algoritmos que hacen uso de recursos compartidos: Unidad de procesamiento, memoria, datos y servicios.

2.2.3 OAR

OAR[6], es un sistema administrador de recursos Cluster y organizador de trabajos, que proporciona muchas de las características más importantes implementadas en sistemas similares. Este sistema se compone de módulos que únicamente interactúan con la base de datos, y se ejecutan como programas independientes.

[6] Planificador de tareas, http://oar.imag.fr/ [23]

Así, formalmente no hay un API[7] determinado, el sistema se define completamente por el esquema de la base de datos. Este enfoque facilita el desarrollo de módulos independientes, en los que cada uno de ellos puede ser desarrollado en cualquier lenguaje de programación que tenga acceso a una librería de base de datos. [5]

El diseño de OAR está basado en las siguientes herramientas de alto nivel:

- Motor de base de datos MySQL[8] o PostgreSQL[9].
- Lenguaje de script Perl[10].
- Mecanismo de restricción CPUSET.
- Herramienta escalable de administración Taktuk[11].

Al igual que sistemas organizadores similares OAR proporciona una explotación de los recursos Cluster. Es decir, OAR, no ejecuta trabajos en los recursos del Cluster, tan solo administra el acceso y el uso de estos nodos para la ejecución de las aplicaciones de los usuarios. [6]

Algunas de las características de OAR son las siguientes:

- Trabajos interactivos y por lotes.
- Reglas de admisión.
- Soporta organización múltiple.
- Múltiples colas con prioridad.
- Reservación de recursos.

[7] Interfaz de Programación de Aplicaciones API
[8] Motor de base de datos, http://www.mysql.com/
[9] Motor de base de datos, http://www.postgresql.org/
[10] Lenguaje de programación, http://www.perl.org/
[11] Es una herramienta para la ejecución remota de aplicaciones en un conjunto considerablemente grande de nodos, http://taktuk.gforge.inria.fr/

- Soporte de tareas moldeables.
- Comprobación de los nodos de cómputo.
- Soporte de nodos dinámicos.
- Suspensión y reenvío de trabajos.

La ejecución de aplicaciones de usuario que requieren de software específico no instalado por defecto en los nodos de cómputo del Cluster, se realiza mediante el despliegue de ambientes que contienen el sistema operativo, y las herramientas necesarias para ejecutar dicha aplicación. El proceso para el montaje del ambiente en cada uno de los nodos es realizado por Kadeploy[12]. [7]

Ventajas de OAR

- Sin un *daemon*[13] específico en los nodos.
- Soporta toda clase de aplicaciones paralelas de los usuarios.
- Las actualizaciones son realizadas sobre el servidor y no en los nodos de cómputo.
- CPUSET (Kernel Linux 2,6), integración que restringe el empleo de los recursos asignados.
- Las tareas administrativas son realizadas con el comando *taktuk*.
- Permite realizar peticiones jerárquicas de los recursos.
- Visualización de las decisiones internas del organizador por medio de la herramienta Gantt[14].
- Comparte parcial o completamente el tiempo.
- Soporta puntos de control y reenvío de trabajos.
- Soporta el enfoque *Best effort*.
- Soporta el despliegue de ambientes mediante Kadeploy.

[12] Herramienta para el despliegue de imágenes, http://kadeploy.imag.fr/
[13] Programa computacional que se ejecuta de forma oculta.
[14] Herramienta para el monitoreo de la ejecución de procesos en Clusters.

2.2.4 SLURM

SLURM[15] (*Simple Linux Utility for Resource Management*) es un sistema administrador de recursos de código abierto adecuado para el uso de grandes y pequeños Clusters Linux que no requiere de modificaciones en el kernel de los equipos [8]. Como un organizador de trabajos, SLURM tiene las siguientes tres funciones principales: [9]

- Asignación del acceso de los usuarios a los recursos de cómputo de forma exclusiva y/o no exclusiva para realizar procesamiento durante algún tiempo determinado.
- Proveer un *framework* para iniciar, ejecutar y monitorear los trabajos en el grupo de nodos asignados.
- Interviene la contienda de los recursos para manejar una cola de trabajo pendiente.

Las características generales de SLURM

- **Simplicidad:**
  SLURM, es lo suficientemente simple para que los usuarios finales comprendan el código fuente, y puedan añadir algo de funcionalidad extra. Los realizadores de SLURM, evitaron agregar características que no sean absolutamente necesarias para la comunidad en general.

- **Código abierto:**
  SLURM, está disponible para todo el mundo y seguirá siendo libre. El código fuente se distribuye bajo la licencia GNU *General Public License*.

- **Portabilidad:**

[15] Simple Linux Utility for Resource Management [24] [25]

SLURM, está escrito en el lenguaje de programación C, con un motor de autoconfiguración GNU. Aunque inicialmente fue escrito para Linux, debe ser fácilmente portable a otros sistemas operativos como UNIX. SLURM también soporta un mecanismo de propósito general que permita ser fácilmente compatible con una variedad de diferentes infraestructuras. En el archivo de configuración de SLURM se especifica el conjunto de módulos que se deben utilizar.

- **Independiente de interconexión:**
  SLURM actualmente soporta UDP/IP basada en la comunicación y la interconexión *Quadrics Elan3*. Añadir soporte para otras interconexiones, incluidas las limitadas topográficamente, es sencillo y se realiza mediante los mecanismos descritos anteriormente.

- **Escalabilidad:**
  SLURM está diseñado para escalar a Clusters con miles de nodos. El controlador SLURM para un Cluster con mil nodos ocupa alrededor de 2 MB de memoria, demostrando un excelente rendimiento. Los trabajos pueden especificar sus requerimientos de recursos en una amplia variedad de formas, incluyendo opciones de requerimientos y rangos.

- **Tolerancia a fallas:**
  SLURM puede manejar una variedad de diferentes fallas sin cargas de terminación de trabajo, incluyendo accidentes en el nodo que se encuentre ejecutando el controlador SLURM. Los trabajos de usuario pueden estar configurados para continuar su ejecución, a pesar de que en uno o más nodos de ejecución se presenten fallas. El comando de usuario para controlar un trabajo, *srun*, puede separar y unir de nuevo las tareas paralelas de un trabajo en cualquier momento. Los nodos asignados a la ejecución de un trabajo, están disponibles para ser reutilizados tan pronto

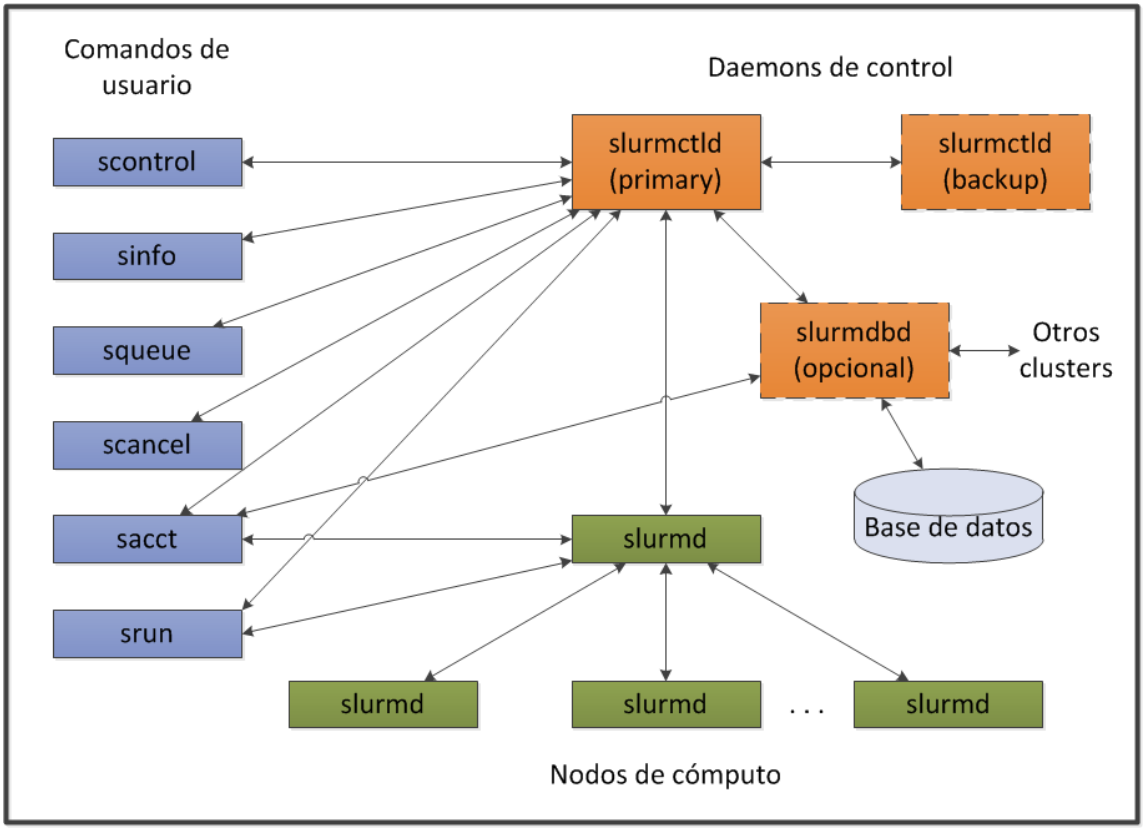
como terminen los trabajos asignados en ese nodo. Si algunos de los nodos no terminan por completo el trabajo a tiempo debido a problemas de hardware o software, únicamente se verá afectada la programación de estos nodos.

- **Seguridad:**
  SLURM emplea una tecnología de encriptación para autenticar los usuarios a los servicios y los servicios entre sí, con una variedad de opciones a través de mecanismos ya establecidos. SLURM no asume que estas redes físicamente son seguras, pero se asume que el Cluster por completo se encuentra dentro de un dominio administrativo con una base común de usuarios.

- **Sistema administrador amigable:**
  SLURM utiliza un simple archivo de configuración y minimiza el estado distribuido. Esta configuración puede ser cambiada en cualquier momento sin generar un impacto a los trabajos en ejecución. Los nodos heterogéneos en un Cluster pueden ser fácilmente administrados. Las interfaces de SLURM pueden ser usadas por secuencia de comandos y su comportamiento es altamente determinista.

La arquitectura de SLURM como se muestra en la figura 2-3, consta de un *daemon slurmd* ejecutándose en cada nodo, un *daemon* central *slurmctld* ejecutándose en el nodo administrador y cinco utilidades por línea de comandos: *srun, scancel, sinfo, squeue y scontrol* que pueden ejecutarse en cualquier host dentro del Cluster.

Figura 2-3, Arquitectura SLURM



Fuente: Bull, «Extreme computing, SLURM guide,» BULL CEDOC, 2010.

2.3 COMPUTACIÓN GRID

El modelo de computación en Grid, frecuentemente se ha definido realizando una comparación con el acceso a los recursos de una red eléctrica, debido a que en esta red, los aparatos electrónicos pueden obtener la energía necesaria para su funcionamiento, sin prestar mayor importancia al cómo o en donde se generan estos recursos. Al igual que las centrales eléctricas, el conjunto de recursos computacionales pertenecientes a una Grid, se encuentra geográficamente distribuidos, y los usuarios o entidades pueden ejecutar sus aplicaciones sin la necesidad de administrar o reconocer las maquinas en las que su aplicación se ha ejecutado.

En 1998 Carl Kesselmas e Ian Foster definieron, “Grid es la infraestructura de hardware y de software que proporciona un acceso serio, constante, penetrable y económico a capacidades computacionales de alta calidad”.

Otra definición propuesta por los mismos autores dice que la Grid son, “Recursos flexibles, seguros y coordinados, compartidos entre personas e instituciones conocidas como organizaciones virtuales”.

Actualmente la definición más extendida por la comunidad Grid es la propuesta por Ian Foster: “Una Grid, es un sistema que coordina recursos que no están sujetos a un control centralizado, usando protocolos e interfaces estandarizadas, abiertas y de propósito general, para proporcionar calidad de servicios no triviales”.

La Grid permite el intercambio, selección y agregación de una amplia variedad de recursos entre los que se puede nombrar: computadores, supercomputadores, sistemas de almacenamiento, dispositivos especializados, etc. Estos componentes generalmente se encuentran distribuidos geográficamente y pertenecen a diferentes organizaciones, con el fin de resolver problemas que impliquen cómputo a gran escala y uso intensivo de datos en la ciencia, industria y el comercio. [10]

Una de las principales diferencias entre la computación Grid y la computación Cluster es la forma en que los recursos son administrados; en el caso de los Clusters la asignación de los recursos es llevada centralizadamente por un administrador de recursos, y todos estos recursos trabajan juntos cooperativamente como un solo recurso unificado, en cambio, en el caso de la Grid la administración tiende a ser más descentralizada y no intenta proveer una sola imagen del sistema.

2.3.1 Características

Todo sistema que desee ser considerado como un sistema Grid, debe cumplir con tres características fundamentales en el funcionamiento del sistema: [11]

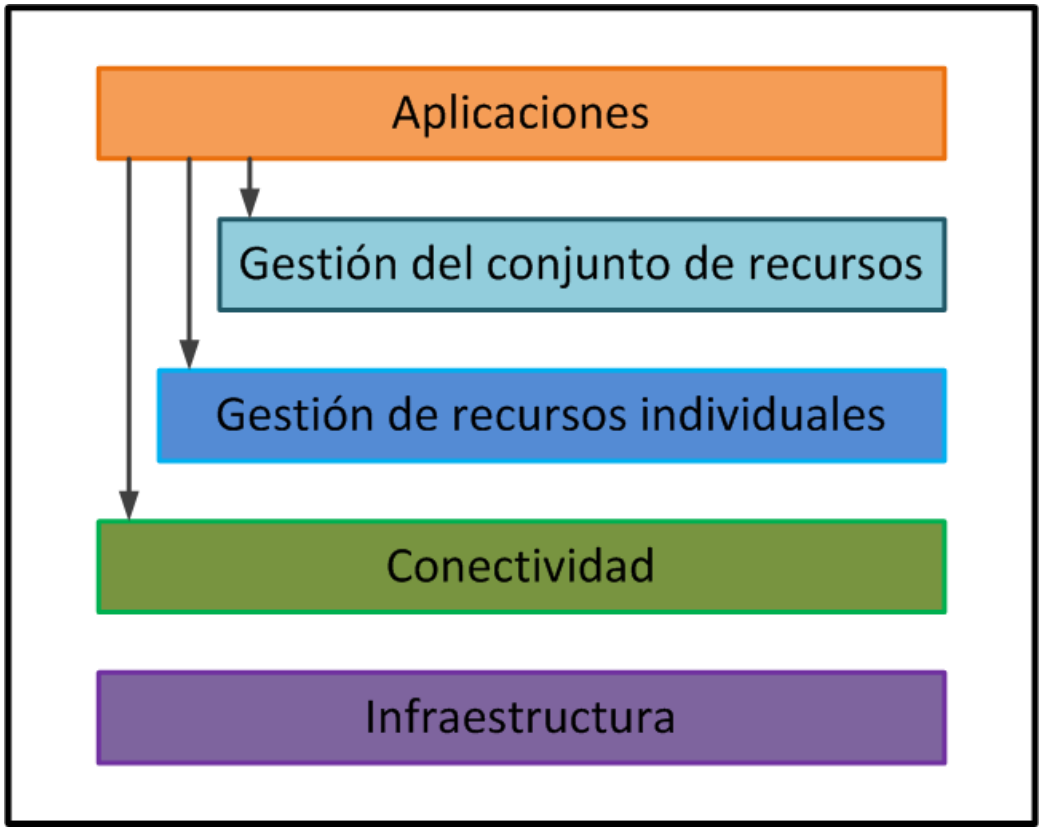
- **Sus recursos coordinados no están sujetos a un control central**
  Al carecer de un control centralizado en la infraestructura, el sistema evita la formación de cuellos de botella y facilita la integración de nuevos recursos computacionales que servirán como nodos de cálculo, por ejemplo estaciones de trabajo de usuarios frente a computadores centrales; unidades administrativas diferenciales de la misma organización; o de diferentes organizaciones. Esta integración de los recursos permite un mejor aprovechamiento de los recursos.

- **Utiliza un estándar, abierto, protocolos e interfaces genéricas**
  El sistema Grid, debe estar basado en protocolos e interfaces estándar, abiertos y de propósito general. Así, se simplifica la ejecución de aplicaciones a través de la Grid, permitiendo que puedan coexistir aplicaciones con distintas políticas de autenticación, autorización o acceso a los recursos.

- **Entrega las calidades no triviales de servicio**
  Una Grid le permite a los recursos que la constituyen, ser empleados de una forma coordinada entregando diferentes calidades de servicio, relacionadas por ejemplo con el tiempo de respuesta, rendimiento, disponibilidad y seguridad, y la asignación de múltiples recursos para conocer las demandas de los usuarios, por lo tanto, esta utilización de los sistemas combinados es significativamente más grande que la suma de sus partes.

2.3.2 Arquitectura grid

Figura 2-4, Arquitectura Grid



Fuente: I. Foster y C. Kesselmas, The Grid: Blueprint for a New Computing Infrastructure, New York: Morgan Kaufmann Publishers - Elsevier Inc., 2004.

- **Infraestructura**

  La infraestructura del Grid está compuesta por los recursos computacionales que se desean compartir. Estos pueden ser, ordenadores individuales, conjunto de ordenadores, Clusters, supercomputadores y sistemas de almacenamiento. En esta capa también se incluye la infraestructura de red y sus mecanismos de gestión y control.

- **Conectividad**

La capa de conectividad incluye los protocolos de comunicación y de seguridad que permiten la comunicación entre los recursos computacionales. Entre estos protocolos se pueden encontrar la pila de protocolos TCP/IP, protocolos en redes de alta velocidad, SSL o Certificados X.509. Esta capa es especialmente importante ya que en la computación Grid intervienen múltiples recursos de distintas organizaciones con distintas políticas de seguridad.

- **Gestión de recursos individuales**

  Esta capa incluye servicios y protocolos para el control y gestión de recursos individuales. En particular existen dos tipos de protocolos principales:

  - Protocolos de información. Permiten obtener información sobre un determinado recurso (características técnicas, carga actual, precio, número de procesadores o memoria disponible).
  - Protocolos de gestión. Permiten el control de un determinado recurso, esto es: acceso, arranque, parada, monitorización, contabilidad o auditoria del recurso.

- **Gestión del conjunto de recursos**

  Esta capa agrupa los servicios que gestionan conjuntos de recursos. Los servicios más comunes que se pueden encontrar en esta capa son:

  - Servicios de directorio.
  - Servicios de planificación y asignación.
  - Servicios de monitorización y diagnóstico.
  - Servicios de contabilidad.
  - Servicios de gestión de datos.

- **Aplicaciones**

Las aplicaciones Grid acceden a la infraestructura del Grid a través de las distintas capas. Según las exigencias de la aplicación, puede ser necesario pasar por todas las capas o conectarse directamente a la infraestructura.

2.3.3 CiGri

CIGRI[16], es una herramienta para computación distribuida a gran escala, que se basa en el concepto de computación Grid ligera, buscando aprovechar el tiempo en que los recursos de la Grid se encuentran sin ser utilizados. Sus principales características son su simplicidad, la escalabilidad y los mecanismos de tolerancia a fallas.

La implementación de CiGri, se puede realizar de una manera relativamente sencilla, pues no es necesaria la instalación de software adicional en cada uno de los Clusters que se encuentran unidos a la plataforma, dado que CiGri se compone de un servidor central (Servidor CiGir), que se encarga de la comunicación con los sistemas planificadores de tareas de cada Cluster, que para este caso utilizan OAR. Esta comunicación se realiza mediante el envío de trabajos, actuando como un usuario más del Cluster, y con un mínimo de prioridad en sus trabajos. [12]

Las aplicaciones que se pueden trabajar por medio de CiGri, se denominan "*Bag of Task* (*BoT*)", o grupo de tareas. Estas aplicaciones se constituyen básicamente por un grupo de tareas que se pueden ejecutar de forma separada al resto de tareas, realizando una parte específica del trabajo en general.

Dada la posibilidad de paradas inesperadas en la ejecución de los trabajos debido a fallas en un nodo o a la demanda de los recursos por parte de un usuario, CiGri cuenta con un mecanismo de tolerancia a fallas que establece unos puntos de

[16] CiGri, http://cigri.imag.fr/ [26] [27]

control (*checkpoint*), para que en caso de una parada abrupta del proceso, se pueda retomar el trabajo desde un punto determinado. [13]

CiGri, funciona como un usuario más de la plataforma, que recibe los trabajos de los usuarios y los redirige a los Clusters, por lo que de este modo cada usuario ya no tendrá inconvenientes con la necesidad de escoger los recursos sobre los cuales se ejecutarán sus aplicaciones, pues en ese caso tan solo tendrían que dirigirse los trabajos a CiGri, y de ahí se encolarían a la plataforma.

CiGri actualmente ofrece soporte únicamente para la integración de Clusters que se encuentren utilizando como su planificador de tareas a OAR y se encuentra en funcionamiento en la plataforma, Grid5000[17].

2.4 COMPUTACIÓN CLOUD

El *National Institute of Standards and Technology* (*NIST*) define Cloud Computing de la siguiente forma "Cloud Computing es un modelo que permite el acceso bajo demanda y a través de la red a un conjunto de recursos compartidos y configurables (como redes, servidores, capacidad de almacenamiento, aplicaciones y servicios) que pueden ser rápidamente asignados y liberados con una mínima gestión por parte del proveedor del servicio". [14]

2.4.1 Principales características

El NIST define lo siguiente como las cinco principales características de la computación en la nube.

- **Autoservicio bajo demanda.**

[17] Grid5000, https://www.grid5000.fr/ [22]

El usuario puede acceder a capacidades de computación en la nube de forma automática a medida que las vaya requiriendo sin necesidad de una interacción humana con sus proveedores de servicio Cloud.

- **Múltiples formas de acceder a la red.**
  Los recursos pueden ser accesibles a través de la red y por medio de mecanismos estándar que son utilizados por una amplia variedad de dispositivos de usuarios, desde teléfonos móviles a computadores portátiles.

- **Compartición de recursos.**
  Los recursos (almacenamiento, memoria, ancho de banda, capacidad de procesamiento, máquinas virtuales, etc.) de los proveedores son compartidos por múltiples usuarios, a los que se van asignando capacidades de forma dinámica según sus peticiones. Los usuarios pueden ignorar el origen y la ubicación de los recursos a los que acceden, aunque sí es posible que sean conscientes de su situación a determinado nivel, como el de CPD (Centro de Procesamiento de Datos) o el de país.

- **Elasticidad.**
  Los recursos se asignan y liberan rápidamente, muchas veces de forma automática, lo que da al usuario la impresión de que los recursos a su alcance son ilimitados y están siempre disponibles.

- **Servicio medido.**
  El proveedor es capaz de medir, a determinado nivel, el servicio efectivamente entregado a cada usuario, de forma que tanto proveedor como usuario tienen acceso transparente al consumo real de los recursos, lo que posibilita el pago por el uso efectivo de los servicios.

2.4.2 Modelo de servicio

- **Software como un servicio (SaaS)**

  Al usuario se le ofrece la capacidad de que las aplicaciones que su proveedor le suministra corran en una infraestructura Cloud, siendo las aplicaciones accesibles a través de, por ejemplo, un navegador web como en el caso del *webmail*, que es posiblemente el ejemplo más representativo, por lo extendido de este modelo de servicio. El usuario carece de cualquier control sobre la infraestructura o sobre las propias aplicaciones, excepción hecha de las posibles configuraciones de usuario o personalizaciones que se le permitan.

- **Plataforma como un servicio (PaaS)**

  Al usuario se le permite desplegar aplicaciones propias (ya sean adquiridas o desarrolladas por el propio usuario) en la infraestructura Cloud de su proveedor, que es quien ofrece la plataforma de desarrollo y las herramientas. En este caso, es el usuario quien mantiene el control de la aplicación, aunque no de toda la infraestructura subyacente.

- **Infraestructura como un servicio (IaaS)**

  El proveedor ofrece al usuario recursos como capacidad de procesamiento, de almacenamiento, o comunicaciones, que el usuario puede utilizar para ejecutar cualquier tipo de software, desde sistemas operativos hasta aplicaciones.

2.4.3 Modelo de desarrollo

- **Cloud privada**

Esta infraestructura Cloud provee un uso exclusivo para múltiples usuarios de una única organización.

- **Cloud comunitaria**
  En esta infraestructura se proporciona un uso exclusivo para los usuarios pertenecientes a una comunidad específica que ha compartido recursos.

- **Cloud pública**
  La infraestructura es operada por un proveedor que ofrece servicios al público en general.

- **Cloud híbrida**
  Esta infraestructura se compone de dos o de tres de los tipos de las infraestructura diferentes.

2.5 DRMAA

Distributed Resource Management Application API (*DRMAA*), es una especificación desarrollada por el *Global Grid Forum,* para facilitar la integración de los Sistemas Administradores de Recursos Distribuidos. El objetivo entonces, es posibilitarle a los programas la creación de aplicaciones, mediante una librería de instrucciones estándar, con la que podrán portar la aplicación entre diferentes plataformas, sin la necesidad de realizar cambios sustanciosos en el código de la aplicación o aún mejor, sin la necesidad de un conocimiento profundo del funcionamiento y las características internas del nuevo planificador de tareas sobre el que se va a ejecutar la aplicación.

En el desarrollo de una implementación del estándar DRMAA, se especifican los mecanismos con los que se puede realizar el envío, monitoreo y controlar de los trabajos. De aquí surge la facilidad para la portabilidad de las aplicaciones, pues al

ser estandarizado estos mecanismos de interacción con las plataformas, las aplicaciones podrán tomar una estructura definida.

La especificación DRMAA constituye un interfaz homogéneo y portable a diferentes DRMS para gestionar el envío, monitorización y control de trabajos, y para la recuperación del estado de los trabajos finalizados. Las implementaciones y las aplicaciones distribuidas de DRMAA, no tienen que particularizarse para un entorno DRMS determinado, o para unas políticas de desarrollo determinadas. Para ello se utilizan las categorías de trabajos y la especificación nativa del DRMAA. De esta manera, las políticas específicas de la aplicación se pueden traducir en simples cadenas de caracteres que interpretarán las implementaciones que utilicen el API DRMAA.

DRMAA ofrece interfaces de “categorías de trabajos” que encapsulan los detalles concretos del recurso donde se ejecutarán los trabajos, ocultando dichos detalles a las aplicaciones que utilicen la interfaz DRMAA. De esta forma, el administrador del recurso, puede crear una categoría de trabajo, sujeta a una determinada aplicación que se ejecutará usando DRMS. El nombre de dicha categoría se especifica como un atributo de ejecución del trabajo. De esta forma, la implementación DRMAA, puede usar ese nombre de la categoría para manejar los recursos específicos y los requisitos funcionales de los trabajos pertenecientes a esa categoría.

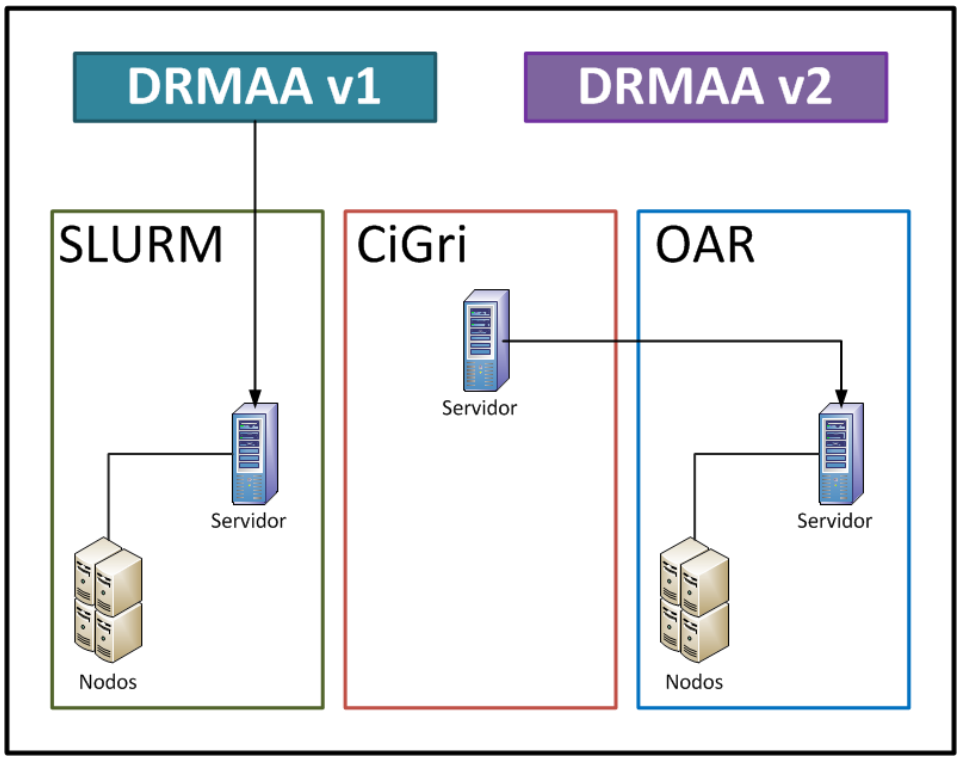
Por lo tanto, el concepto de categorías ofrece ocultación de las políticas determinadas de cada recurso a los trabajos que se estén ejecutando usando el API DRMAA. Por otro lado para facilitar la implementación de dichas categorías, se debe mantener una categoría por cada política del recurso utilizada. De esta forma, la especificación nativa de las categorías, permite interpretar cada política a través de una simple cadena de caracteres, que será interpretado por cada una de las librerías del API DRMAA.

De esta manera, el API DRMAA permite el desarrollo de aplicaciones en Grid ocultando al usuario cualquier detalle relativo al gestor de recursos distribuidos o al middleware utilizado. Por lo tanto, el desarrollador realizará un programa de forma secuencial, con sentencias muy parecidas al POSIX de Unix, y DRMAA, el metaplanificador o gestor de recursos distribuidos, y el middleware se encargarán del resto. En contrapartida, los resultados parciales de las ejecuciones deberán ser almacenados en ficheros ubicados en memoria secundaria aumentando en determinadas ocasiones, el tiempo total de ejecución.

3 ESTADO DEL ARTE

Los proyectos que se realizan por parte de los diferentes grupos de investigación de la Universidad Industrial de Santander, cuentan con el apoyo del proyecto de computación avanzada, GridUIS-2, que brinda el soporte, para que las aplicaciones que requieran de un gran potencial computacional, puedan ser ejecutas en la infraestructura de cálculo científico del proyecto.

Figura 3-1, Nivel de integración entre los componentes de GridUIS-2



Fuente: Autor

GridUIS-2, está compuesto por la agrupación de diversos recursos computacionales, distribuidos entre el campus central y la sede investigativa de la universidad. Entre estos recursos se puede contar con máquinas diseñadas

específicamente para el cálculo intensivo, clusters de computadores dedicados y no dedicados. La gestión de los recursos queda a cargo de dos sistemas administradores de recursos distribuidos OAR y SLURM, que actualmente se encuentran en la versión 2.5 cada uno.

En la integración de los recursos computacionales que se está realizando actualmente en la plataforma de cómputo avanzado, GridUIS-2, se está buscando la posibilidad de que se integren todos los Clusters que tengan como sistema administrador de recursos distribuidos a OAR o SLURM. Para este fin, actualmente se encuentra en etapa de prueba dentro de la infraestructura, la herramienta CiGri, que ofrece un enfoque de computación Grid ligera, para el envío de grupos de tareas, en la que los recursos van a ser utilizados por esta herramienta, únicamente en los momentos que no se encuentre realizando procesamiento por parte de usuarios con mayor prioridad sobre la plataforma.

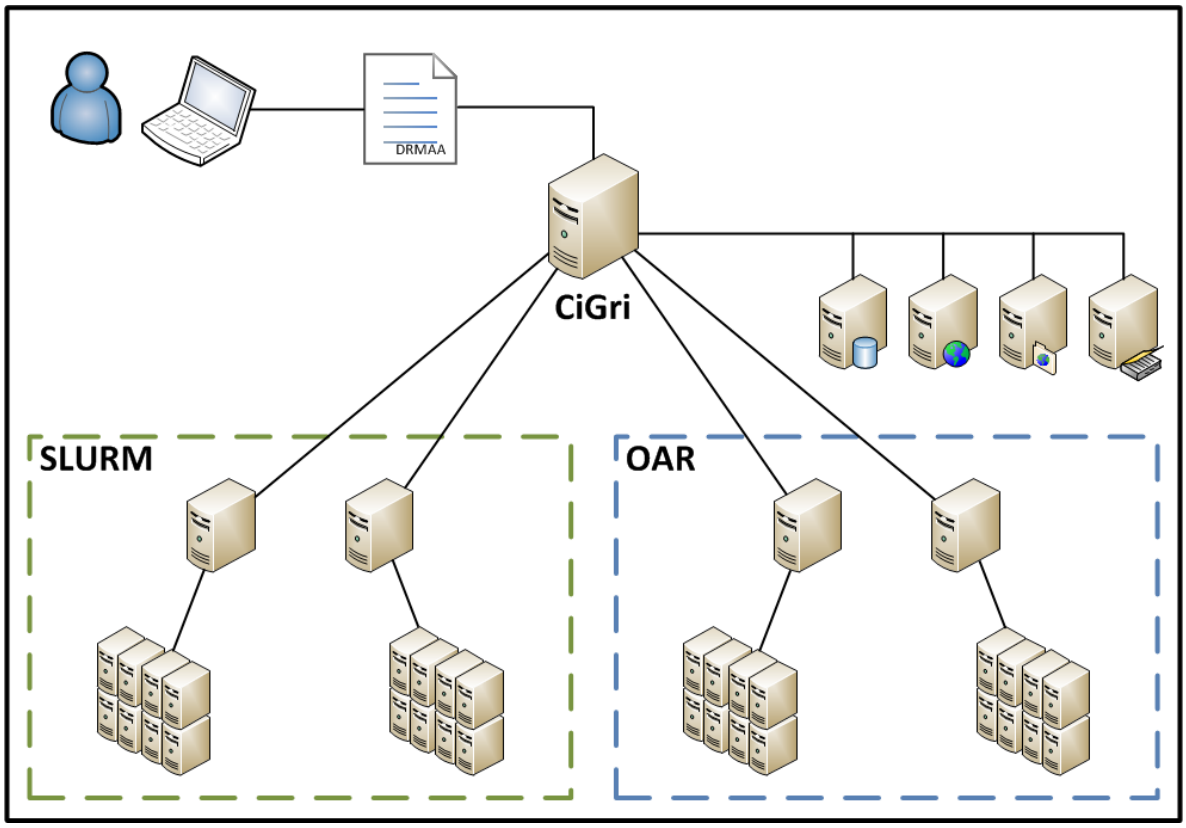
CiGri, que actualmente se encuentra en la versión 1.4.5, y en desarrollo la versión 3, puede ser integrado con el planificador de tareas OAR, mediante un API desarrollado bajo la arquitectura RestAPI, y con el que se dota a CiGri de la posibilidad de enviar, gestionar y cancelar grupos de tareas en Clusters OAR, bajo el enfoque Best Effort.

DRMAA, que en su versión más reciente la 2.0, no cuenta aún con implementaciones del estándar para ninguno de los sistemas administradores de recursos distribuidos, con los que actualmente se están trabajando en gridUIS-2, cuenta con una implementación del estándar en la versión 1.0.6, en done ofrece soporte para la versión 2.3 de SLURM.

4 ARQUITECTURA PROPUESTA

El desarrollo del presente trabajo, consistió en el estudio de la especificación del estándar DRMAA, creado por el Gobal Grid Forum, con el fin de establecer mecanismos de programación mediante la implementación de una librería que permite la creación de aplicaciones estandarizadas, que pueden ser portadas entre los diferentes sistemas administradores de recursos distribuidos que dispongan de una implementación del estándar DRMAA.

Figura 4-1, Integración propuesta para GridUIS-2



Fuente: Autor

Después de realizar el análisis de cada uno de los sistemas administradores de recursos distribuidos disponibles en la infraestructura GridUIS-2, y con el objetivo de establecer el mejor diseño para la integración de los recursos computacionales, bajo una herramienta que permita la administración y planificación de los trabajos enviados a los Clusters, de una forma centralizada y con un mejor aprovechamiento del tiempo ocioso de los recursos, se estableció el esquema planteado en la figura 4-1, como el mejor diseño de integración, teniendo en cuenta las herramientas y las políticas ya definidas en GridUIS-2, para el manejo de los recursos y los trabajos.

Como se aprecia en la figura 4-1, el esquema se basa en un servidor CiGri central, que se encarga del envío de aplicaciones compuestas de grupos de tareas, a los sistemas planificadores de trabajos de cada Cluster. Con una implementación del estándar DRMAA, para Cigri, las aplicaciones pueden ser realizadas con la librería DRMAA, y de esta forma pueden ser portadas para su ejecución en otras plataformas que cuenten con la implementación del estándar.

La determinación de este esquema como la forma de integrar los recursos de la plataforma, se debió a que en GridUIS-2, el sistema planificador de tareas dominante es OAR, y actualmente CiGri, cuenta con un API que permite el envío de trabajos a los Clusters que tengan como sistema administrador de recursos distribuidos a OAR. Por esto, y debido a que el estándar DRMAA, en su última versión no cuenta con una implementación para los sistemas planificadores presentes en la infraestructura GridUIS-2 (OAR, SLURM y CiGri), se estableció el esquema ya presentado.

Para esta integración, fue necesario un completo estudio de la especificación del estándar DRMAA en su versión más reciente (DRMAA v2), teniendo en cuenta el funcionamiento, la forma en que se manejan los trabajos, el tratamiento de errores a través de constantes definidas en el estándar que indican la causa del error, las

características con las que debe contar un trabajo y el funcionamiento de las rutinas encargadas del envío y gestión de los trabajos. Junto al estudio realizado del estándar DRMAA, también se analizó el funcionamiento de CiGri, las aplicaciones que pueden ser ejecutadas a través de él, la forma en cómo se gestiona el envío de trabajos, el tratamiento de los errores en la ejecución de las tareas, y el monitoreo del estado de la plataforma y del estado de los trabajos en ejecución.

En cuanto a la forma de integrar el sistema administrador de recursos distribuidos SLURM, para que pudiera ejecutar tareas enviadas por CiGri, se tomó como referencia el API implementado por OAR para desarrollar un API para SLURM. Este API fue desarrollado teniendo en cuenta el estilo de arquitectura diseñado especialmente para sistemas distribuidos de hipermedios, *Representational State Transfer* (REST). Por parte de CiGri, fue necesario el estudio y modificación de algunas librerías, para poder adherir la opción de que aceptara el envío y el tratamiento de trabajos a Clusters con el planificador de tareas SLURM.

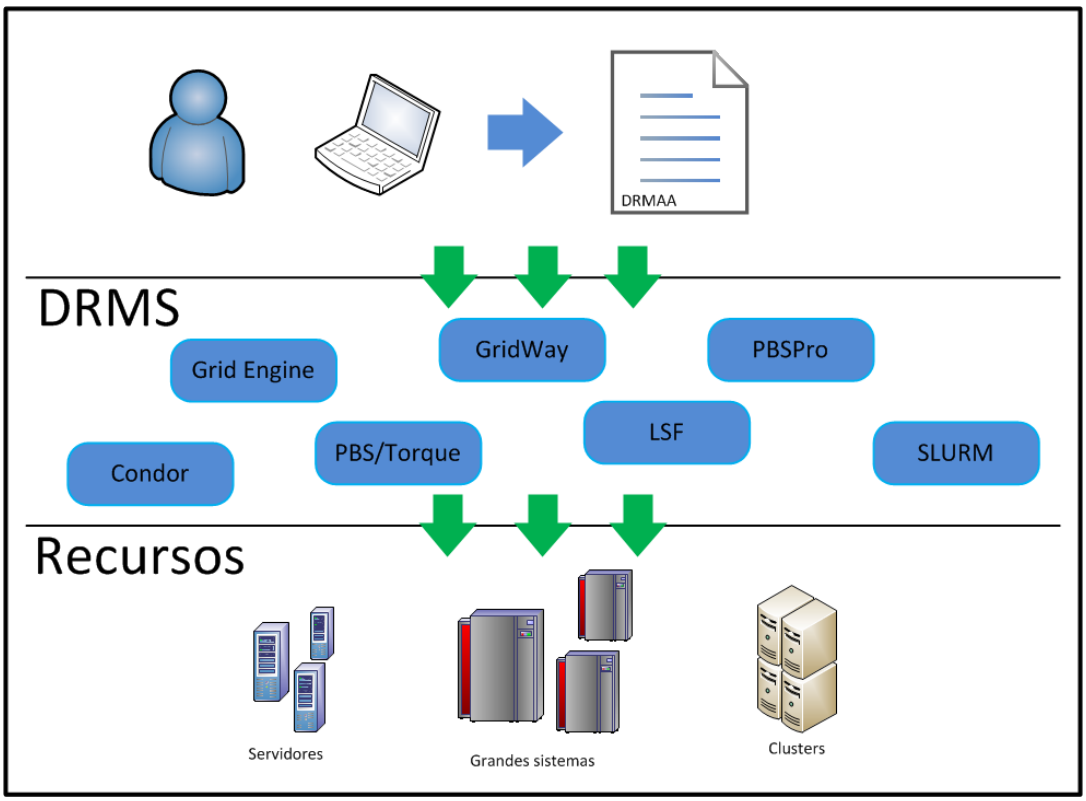
5 TRABAJO DESARROLLADO

5.1 CARACTERIZACIÓN DE LA ESPECIFICACIÓN DRMAA

Distributed Resource Management Application API (DRMAA), es la especificación de un estándar desarrollada por el *Global Grid Forum*, con el objetivo de que las aplicaciones que necesitan de computación de alto desempeño para su ejecución, puedan ser desarrolladas sin centrarse en un sistema administrador de recursos distribuidos determinado, por el contrario, se busca que las aplicaciones se puedan migrar entre infraestructuras de supercomputación, sin la necesidad de realizar cambios en el código fuente, brindándole de esta forma portabilidad a las aplicaciones.

En la figura 5-1, se representa la forma general en que DRMAA interactúa con los recursos hardware de una infraestructura computacional cualquiera. Así, un usuario que tiene su aplicación realizada bajo el estándar DRMAA, puede enviarla a cualquier sistema administrador de recursos, que cuente con una implementación del estándar, y este se encargara de la interacción con las máquinas que se encargan del procesamiento final.

Figura 5-1, Componentes de la especificación DRMAA



Fuente: Autor

En el desarrollo de una implementación del estándar DRMAA para un sistema administrador de recursos distribuidos determinado, es necesario tener en cuenta unas ciertas partes, que están especificadas por el *Open Grid Forum* en la documentación del estándar DRMAA.

Las siguientes son las partes de una implementación DRMAA

- DRMS (Distributed Resource Management System)

  Se denomina DRMS a cualquier sistema que soporte el concepto de ejecución de tareas en recursos computacionales distribuidos, con la ayuda de un software planificador central. Algunos ejemplos DRMS serían los sistemas multiprocesador controlados por un sistema operativo planificador,

sistemas Clusters con múltiples maquinas controladas por un planificador central, sistemas Grid o sistemas Cloud con un concepto de trabajos.

- Implementación DRMAA (Librería DRMAA)
  La librería DRMAA, es el resultado del desarrollo de la implementación del estándar DRMAA, para la vinculación del comportamiento funcional de un DRMS bajo dicho estándar.

- Aplicación (DRMAA-based)
  La aplicación es el software utilizado por la implementación del DRMAA para establecer de forma estandarizada la comunicación y el acceso a múltiples sistemas DRM.

- Host de sumisión
  En cada DRMS debe haber un recurso computacional encargado de ejecutar la aplicación DRMAA-based. Este host a su vez también puede funcionar como un recurso de ejecución de trabajos.

- Host de ejecución
  Es el recurso computacional disponible en el sistema DRM en el que se realiza el procesamiento de los trabajos enviados por el administrador de trabajos.

- Job
  El trabajo computacional que desean realizar los usuarios en el sistema DRM mediante la implementación DRMAA. Este trabajo es enviado desde el host de sumisión por medio de la aplicación DRMAA-based a múltiples host de ejecución.

5.1.1 Características de una implementación DRMAA

Para el desarrollo de una implementación del estándar DRMAA, sin importar el sistema administrador de recursos distribuidos para el que va a ser desarrollado se deben tener en cuenta las siguientes características:

- Portabilidad

  El desarrollo de la implementación DRMAA, debe ser realizado por medio de módulos que puedan ser intercambiados por el usuario en tiempo de ejecución. Es decir, la aplicación que utiliza el estándar DRMAA, debe poder ejecutarse en cualquier DRMS, que cuente con la implementación del estándar, tan solo con la configuración de un conjunto de variables de entorno para el DRMS específico con que se desee trabajar en ese momento.

- Thread Safety

  Los autores esperan que los desarrolladores de la librería DRMAA proporcionen *multithread*. Por ello es recomendable que la librería DRMAA sea *thread-safety*, y permita aplicaciones multihilo que usen la librería DRMAA sin una sincronización explicita entre los hilos de la aplicación.

  Se recomienda que los implementadores de DRMAA califiquen sus implementaciones como *thread-safe* de acuerdo con el criterio anterior. Los desarrolladores de implementaciones que no sean *thread-safe* deberían documentar todas las interfaces “inseguras” y proporcionar una lista de interfaces y sus dependencias con las rutinas externas “inseguras”. Sin embargo, antes de que una aplicación multihilo pueda usar cualquier interfaz DRMAA la rutina de inicialización de la librería debe ser llamada por un único hilo, normalmente el principal, y del mismo modo, la desconexión de la librería será realizada por un único hilo.

- Sincronización

  La implementación DRMAA debe gestionar el asincronismo del inicio y la terminación de los trabajos, de la misma forma a como trabajan los procesos UNIX, esto mediante una llamada de espera al trabajo que se desee.

- Entorno de la aplicación distribuida

  En la implementación DRMAA se especifican mecanismos para el envío, monitoreo y control de trabajos y la recolección de los resultados de la ejecución de los trabajos. Idealmente, las implementaciones DRMAA y las aplicaciones distribuidas no necesariamente están configuradas para un particular entorno DRMS o una política DRMS específica. Para facilitar el desarrollo donde éste no esté completamente acoplado, las categorías de trabajos y la especificación nativa debe ser usada para abstraer o agregar las políticas específicas con simples cadenas que serán interpretadas por las implementaciones DRMAA.

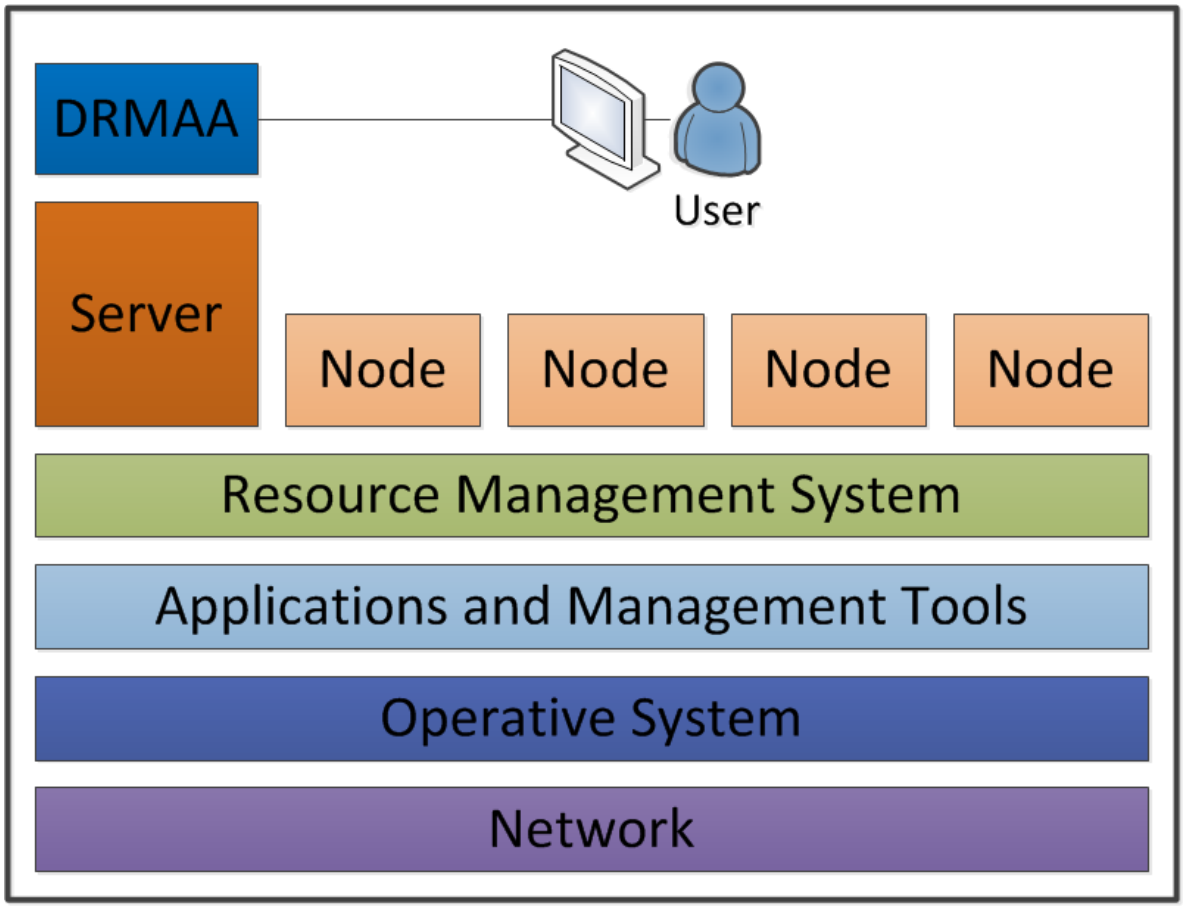
5.1.2 Arquitectura DRMAA

Toda implementación DRMAA sigue la misma arquitectura indiferente del tipo de DRMS en el que se encuentre integrado, por tanto el API DRMAA debe estar instalado en el servidor encargado de administrar los recursos del DRMS.

En la figura 5-2, se representa la arquitectura general de un Cluster en el que se ha montado un sistema administrador de recursos, que cuenta con la implementación del estándar DRMAA. Entre las capas que lo componen se encuentra a más bajo nivel la red de comunicación entre los componentes del Cluster; en el siguiente nivel está el sistema operativo que es el mismo para todos; sigue a capa de herramientas de administración y herramientas en general que

son instaladas en el Cluster para poder realizar cálculo avanzado; sobre estas capas se encuentra el sistema administrador de recursos distribuidos que se encarga de distribuir y planificar las tareas enviadas desde el servidor a los nodos de cómputo; por último se encuentra la implementación del estándar DRMAA que es instalada únicamente en el servidor y se encarga básicamente de hacer la migración de la aplicación para que esta se ejecute sobre la plataforma sin realizar cambios el código de la aplicación.

Figura 5-2, Arquitectura DRMAA



Fuente: Autor

5.1.3 Funcionamiento del estándar DRMAA

Antes de especificar las funciones más importantes que ofrece el estándar DRMAA, es importante explicar el concepto de sesión. Toda implementación de una aplicación que se realice bajo el API DRMAA debe estar dentro de una sesión DRMAA. Es decir, cualquier llamada a un método o función del API que no se encuentre dentro de una sesión, previamente iniciada, no será tratada y se enviará un mensaje de error. Por lo tanto, todas las llamadas DRMAA deberán estar ubicadas entre las llamadas de inicio y de fin de sesión. El motivo de la existencia de sesiones es permitir la independencia de los elementos que se encuentren dentro de una misma sesión del resto de elementos semejantes que se encuentren en ejecución.

La especificación del API DRMAA se divide en cuatro grandes grupos de rutinas:

- INIT y EXIT

  Estas rutinas son las encargadas de inicializar y finalizar la comunicación con los diferentes DRMS, inicializando la librería API DRMAA creando una sesión DRMAA. La rutina encargada de iniciar la sesión, inicializa las estructuras de datos necesarias en una sesión de DRMAA, mientras que la llamada encargada de finalizar la sesión libera las estructuras de datos, dejando la aplicación como se encontraba antes del inicio de la sesión.

- Gestión de las plantillas del trabajo

  Permiten la manipulación de entidades de definición, es decir, patrones de trabajos. De esta forma, permite establecer parámetros tales como el fichero ejecutable, sus argumentos, flujos de salida estándar, variable de entorno y directorio de trabajo local. DRMAA permite extender estas rutinas

a rutinas propias y especificas del gestor de recursos a utilizar. Dependiendo del lenguaje de programación estas rutinas pueden ser un conjunto de definiciones, funciones y constantes declaradas en un módulo, o pueden constituir una clase base a partir de la cual se pueden extender todas las clases hijas que se deseen.

El conjunto de atributos del trabajo es el siguiente:

- Comando remoto de ejecución.
- Comando remoto de entrada de parámetros, un vector de parámetros.
- Estado de sumisión del trabajo.
- Ambiente del trabajo, un vector de parámetros.
- Directorio del trabajo.
- Categoría del trabajo.
- Especificación nativa.
- Entrada y salida estándar, y flujo de errores.
- Lista de correo para informar del estado y la finalización del trabajo, un vector de parámetros.
- Correo de supresión.
- Tiempo de inicio del trabajo.
- Nombre del trabajo a ser usado por job submission.

- Envío de trabajos

Estas rutinas son el eje fundamental dentro de la API DRMAA, se encargan de enviar los trabajos para su ejecución. Permiten el envío de un trabajo independiente, así como el envío de un conjunto de trabajos. Tanto en un caso como en el otro, el API DRMAA devuelve un identificador único que

representa al trabajo enviado al DRMS para su ejecución dentro de la sesión DRMAA.

- Monitorización y control de trabajos

  En este grupo se aglomeran todas aquellas rutinas que se utilizan para controlar y sincronizar trabajos así como para monitorizar su estado. Las rutinas de control permiten parar la ejecución, reenviar o matar tanto un trabajo específico como la totalidad de trabajos que estén en ejecución en una determinada sesión. Por otro lado, las rutinas de sincronización permiten esperar a la finalización de ejecución de un trabajo determinado, de un conjunto de trabajos o de todos los trabajos de la sesión. Finalmente, las llamadas de monitorización devuelven el estado actual de un determinado trabajo.

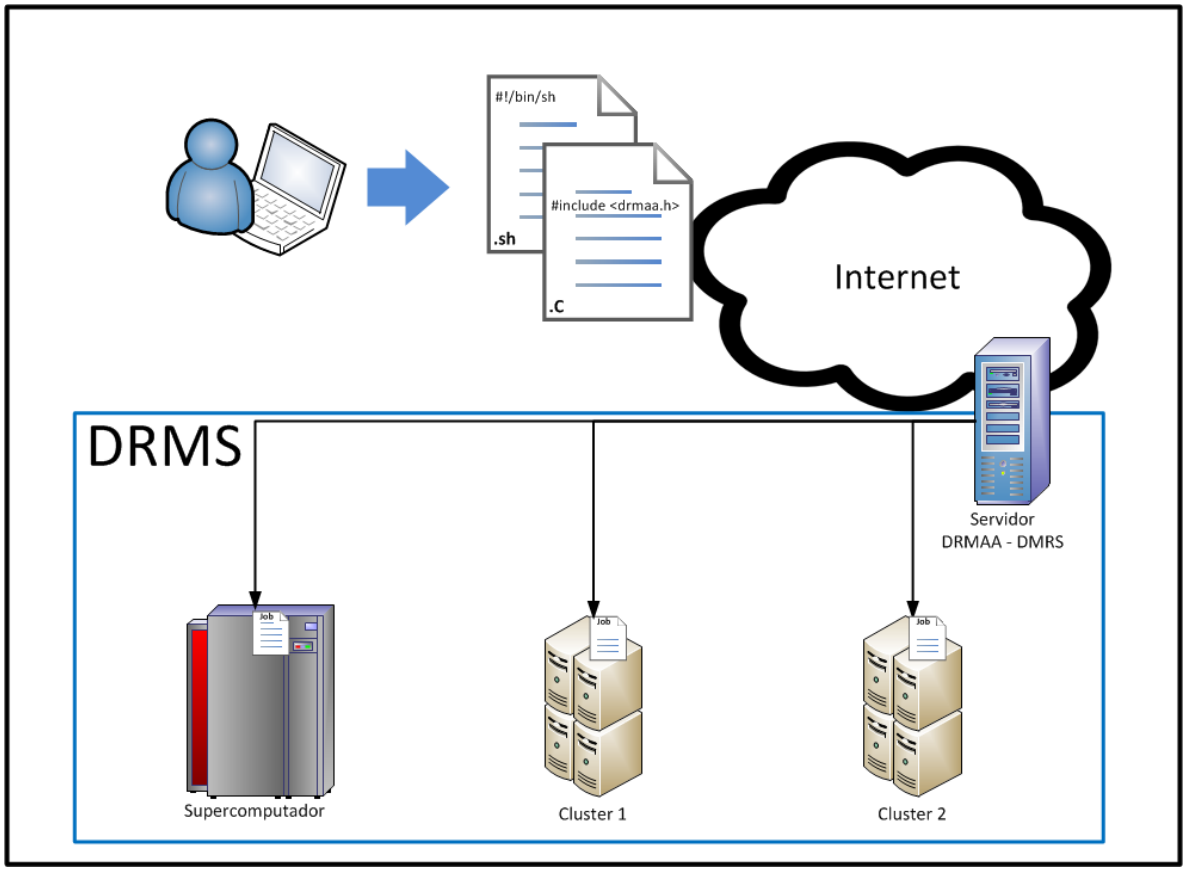
  Un trabajo remoto puede estar en uno de los siguientes estados:

  - Sistema en espera.
  - Usuario en espera.
  - Sistema y usuario en espera simultáneamente.
  - Encolado.
  - Sistema suspendido.
  - Usuario suspendido.
  - Sistema y usuario suspendidos simultáneamente.
  - Ejecutándose.
  - Finalizado correctamente.

  Los trabajos rechazados no se encuentran en ningún estado debido a que no se les asigna ningún tipo de identificador.

Esta especificación permite el desarrollo de programas a través de un estándar facilitando así el despliegue de aplicaciones en DRMS. Además, el uso del API DRMAA permite desarrollar aplicaciones independientemente del gestor de recursos utilizado. También proporciona al desarrollador rutinas para la gestión de los trabajos en ejecución, posibilitando de esta manera la implementación de aplicaciones compatibles con los perfiles de ejecución típicos de Grid, como aplicaciones de alta productividad o aplicaciones maestro-esclavo. Y lo más importante, facilita el desarrollo de aplicaciones que se desacoplen de la infraestructura DRMS donde se vayan a ejecutar.

Figura 5-3, Funcionamiento del estándar DRMAA



Fuente: Autor

En la figura 5-3, se muestra el funcionamiento de un ambiente de computación avanzada en el que se utiliza un sistema administrador de recursos distribuidos que cuenta con la implementación del estándar DRMAA para su funcionamiento.

5.2 DRMAA CON OAR (BEST EFFORT)

Como se mencionó anteriormente, una infraestructura de cómputo avanzado, está integrada por varios Clustes, que a su vez, pueden estar compuestos por un gran número de recursos computacionales, dispuestos para que los usuarios ejecuten aplicaciones que buscan explotar el gran poder de cálculo con que cuenta una plataforma de este tipo.

Los usuarios que generalmente acceden a recursos Cluster, son de tipo académico, científico o empresarial, todos con el objetivo de sacarle el mayor provecho posible a la ejecución de tareas en un ambiente con arquitectura distribuida. Buscando una mayor organización al momento de utilizar los recursos de la plataforma, los usuarios deben estar clasificados en atención a la importancia del trabajo que se encuentren realizando, esto para poder definir unas políticas de prioridad y acceso a determinadas máquinas. Dada la gran cantidad de usuarios con que se puede llegar a contar, y la competencia por el acceso a los recursos, se hace necesaria la implementación de un software planificador de tareas que ordene la forma en que se utiliza la infraestructura.

La administración del acceso y control de los trabajos recae en el planificador de tareas, cuyo principal objetivo en cada aplicación paralela que se va a ejecutar sobre la plataforma, es la elección de un conjunto de recursos que se adecue a las necesidades de dicha aplicación. Dicho de otra forma, el planificador de tareas busca seleccionar entre los recursos disponibles, el conjunto de ellos con los que la aplicación paralela podrá obtener un rendimiento máximo.

Todo trabajo que se desee ejecutar en la infraestructura de cómputo deberá recorrer una serie de pasos que se describen a continuación.

- **Creación**

  La creación del trabajo, consiste en la solicitud de recursos dentro de la plataforma por parte del usuario, para la ejecución de una aplicación.

- **Envío**

  En este paso el trabajo es enviado a los recursos que han sido asignados previamente para su ejecución.

- **Ejecución**

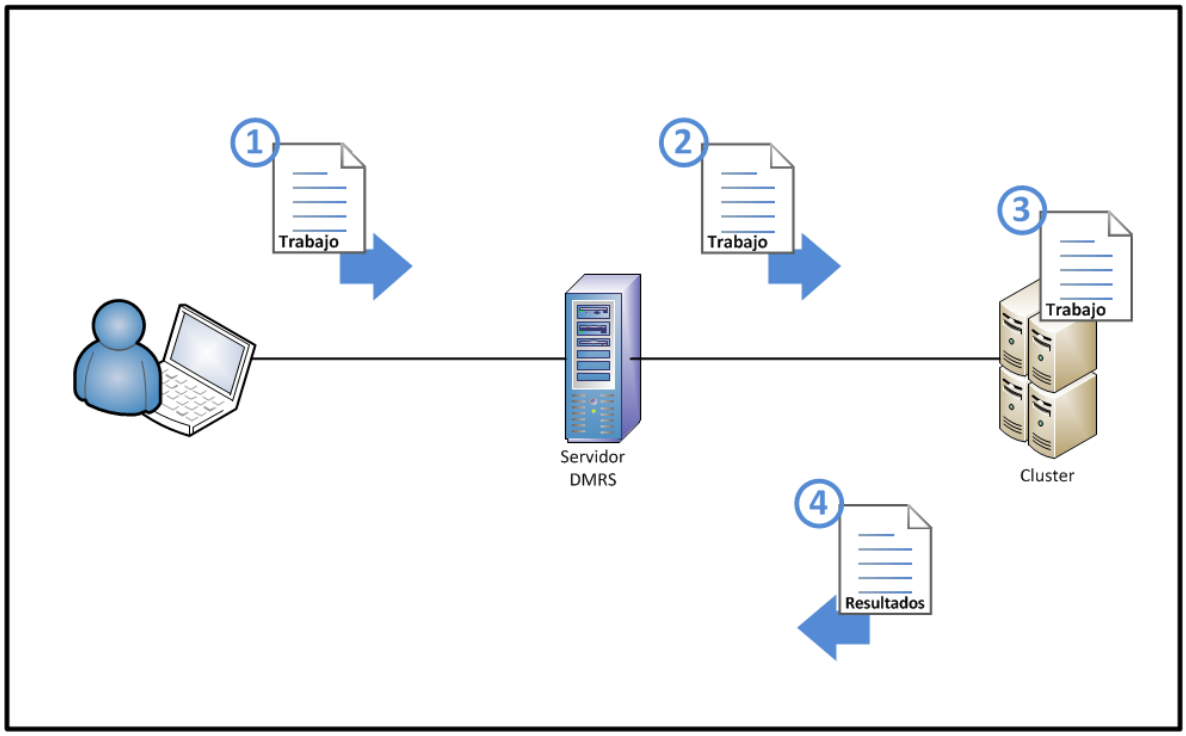
  En este punto, el conjunto de tareas incluidas en el trabajo, ha sido distribuido para que se ejecute en los recursos asignados.

- **Finalización**

  Al terminar las tareas que se ejecutan en cada uno de los recursos computacionales asignados para dicha labor, estos son liberados y quedan a la espera de nuevas tareas para su ejecución.

El la figura 5-4, se puede observar el flujo que debe seguir cualquier trabajo para su exitosa culminación dentro de la plataforma. Así, en el paso 1, el usuario hace el envío de la aplicación hacia el servidor, para que el planificador de tareas cree el trabajo en la plataforma; en el paso 2, el planificador se encarga de enviar el trabajo creado previamente a los recursos de cálculo; en el paso 3, las tareas del trabajo son ejecutadas; por último, en el paso 4, el trabajo es finalizado y los resultados son enviados de regreso al usuario.

Figura 5-4, Pasos de un trabajo para su ejecución



Fuente: Autor

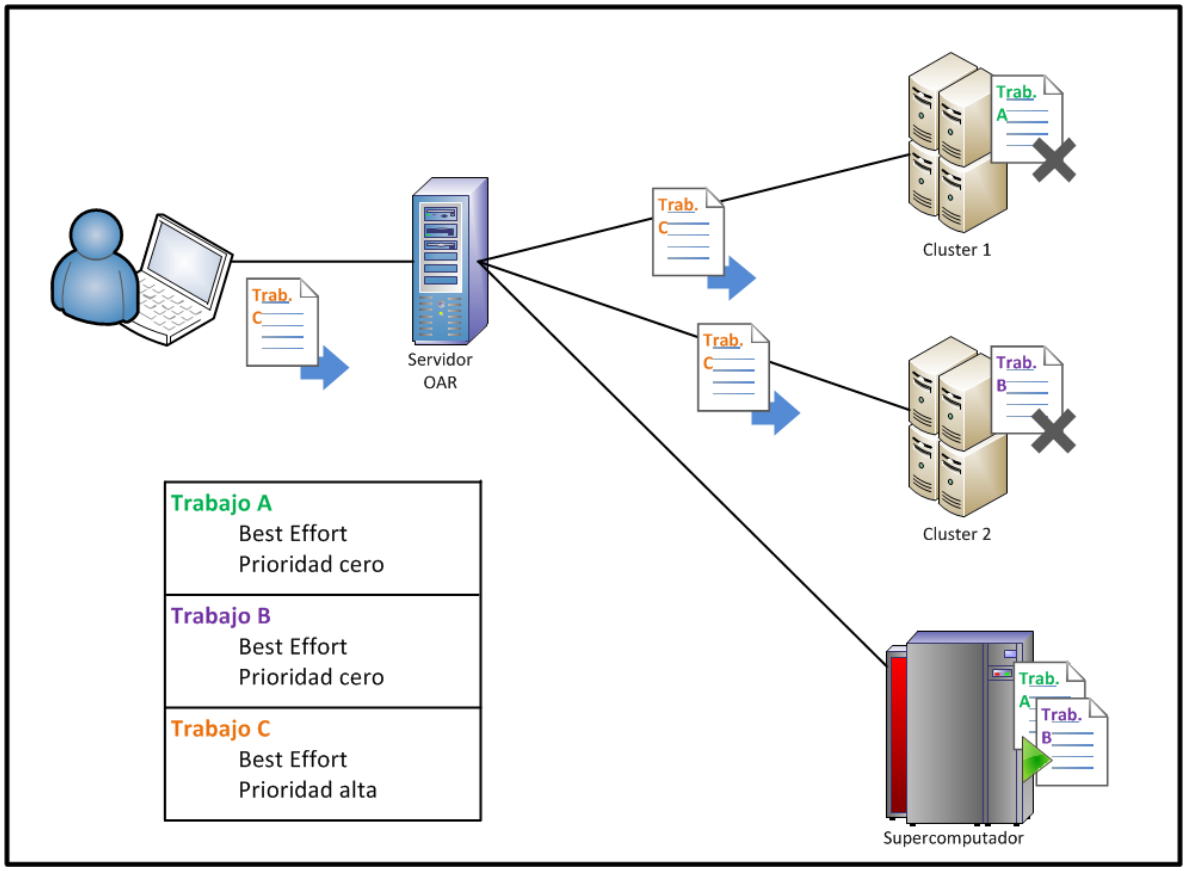
OAR, al igual que cualquier otro sistema planificador de tareas provee funcionalidades con las que se envían, suspenden, cancelan y monitorean los trabajos, con el fin de lograr el uso óptimo de los recursos y minimizar el tiempo de acceso a los mismos. La planificación de las tareas, depende en gran medida de las características de la aplicación y los requerimientos computacionales que esta tiene para su ejecución, dependiendo de esto, OAR encola el trabajo para que en el momento en que los recursos solicitados sean liberados, la aplicación pueda ser enviada para su ejecución.

5.2.1 Best Effort

En una plataforma de computación avanzada, habitualmente se están ejecutando cientos de tareas al tiempo, algunas de ellas con mayor importancia que otras. Entre estas, se puede contar con aplicaciones que buscan aprovechar los

recursos subutilizados de la infraestructura, como es el caso del proyecto *SETI@home*. Este tipo de aplicaciones son enviadas bajo el concepto de *Best Effort*, por el cual, las aplicaciones que se dividen en grupos de tareas y que se pueden migrar fácilmente entre los recursos de la plataforma, son enviadas para su ejecución con la prioridad mínima, así, si el planificador de recursos recibe una aplicación con mayor prioridad procede a liberar estos recursos para asignárselos a la nueva aplicación.

Figura 5-5, Funcionamiento de Best Effort



Fuente: Autor

En la figura5-5, se aprecia una plataforma que está totalmente ocupada con la ejecución de dos trabajos A y B. Las tareas de estos dos trabajos que se están

ejecutando en los Clusters 1 y 2, son canceladas para darle paso a la nueva aplicación (trabajo 3), que necesita ocupar los recursos de estos dos Clusters y cuenta con una prioridad mayor para el uso de la plataforma.

5.2.2 Best Effort bajo el estándar DRMAA

Actualmente hay múltiples planificadores de tareas que han sido creados cumpliendo con las funciones básicas pero a los que se les ha ido agregando características que los diferencian de los demás y con las que buscan suplir necesidades inherentes al proyecto por el cual se han desarrollado. Una de las principales diferencias entre los planificadores de tareas son los comandos que se deben utilizar para trabajar con ellos. Aunque para los administradores de las plataformas es de gran importancia la forma en cómo trabaja el planificador de tareas, para el usuario final, el que tan solo ve la infraestructura como un medio para la obtención de sus resultados, es poco o nada importante el cómo se administre al ejecución de su aplicación, él tan solo desea que su aplicación se ejecute correctamente y en tiempos óptimos.

Dadas las diferencias que existen entre los diferentes planificadores de trabajos el Global Grid Forum pretende estandarizar la utilización de los sistemas administradores de recursos bajo el DRMAA, esto con el fin de poder migrar aplicaciones entre plataformas sin tener que reconstruirlas de nuevo para que se adapten al cambio de planificador.

En el caso de OAR, es necesario tener en cuenta el enfoque Best Effort, para el desarrollo de la implementación del estándar DRMAA, esto se hace agregando excepciones en los módulos, con el objetivo de poder obtener el tipo de trabajo que se está ejecutando, y la prioridad que esta aplicación puede tener sobre los recursos que está utilizando. Estas modificaciones en el estándar DRMAA, son

con el fin de que las aplicaciones que se ejecutan mediante el API, puedan también aprovechar las ventajas del enfoque Best Effort, con que cuenta OAR.

5.3 DRMAA CON SLURM EN GRID Y CLOUD COMPUTING

Como ya fue mencionado anteriormente, *Simple Linux Utility for Resource Management* (SLURM), es un sistema administrador de recursos distribuidos que trabaja a nivel de Cluster. SLURM, cumple con las funcionalidades de un administrador y planificador de tareas.

SLURM, actualmente se encuentra en la versión 2.5, pero esta versión aún no está soportada por la especificación DRMAA. La implementación del estándar DRMAA para SLURM, se encuentra en la versión 1.0.6, y brinda soporte para la versión 2.3 de SLURM desde la versión 1.0.4 de la implementación DRMAA. Estas implementaciones del estándar han sido desarrolladas mediante librerías escritas en el lenguaje de programación C.

En la implementación que ya se encuentra desarrollada para el funcionamiento de SLURM bajo las directivas del estándar DRMAA, se encuentran los siguientes archivos, cada uno con funciones incluidas dentro de sí.

En el archivo drmaa.c, se pueden encontrar las funciones que permiten la creación de una sesión y una plantilla de trabajo nuevas. Por medio de este archivo, se obtiene la información relacionada con los datos de la implementación del estándar, como es el caso de la versión, y el sistema administrador de recursos distribuidos sobre el que está implementado el estándar. Aquí, también se puede encontrar el tratamiento de algunos errores que se pueden presentar al momento de hacer la planificación de recursos con SLURM.

Tabla 5-1, Funciones incluidas en el achivo drmaa.c

drmaa.c
slurmdrmaa_new_session
slurmdrmaa_new_job_template
slurmdrmaa_get_contact
slurmdrmaa_get_version
slurmdrmaa_get_DRM_system
slurmdrmaa_get_DRMAA_implementation
slurmdrmaa_get_attribute_names
slurmdrmaa_get_vector_attribute_names
slurmdrmaa_wifexited
slurmdrmaa_wexitstatus
slurmdrmaa_wifsignaled
slurmdrmaa_wtermsig
slurmdrmaa_wcoredump
slurmdrmaa_wifaborted

La administración de los trabajos se realiza mediante el archivo job.c, por medio del cual se puede administrar el envío de trabajos a los recursos Cluster, adicionalmente se realiza un control de los trabajos.

Tabla 5-2, Funciones incluidas en el achivo job.c

job.c
slurmdrmaa_job_control
slurmdrmaa_job_update_status
slurmdrmaa_job_on_missing

slurmdrmaa_job_new
slurmdrmaa_job_create_req
slurmdrmaa_job_create

Las funciones incluidas en el archivo sesión.c, se encargan de los trabajos o grupos de trabajos desde e punto de vista de la sesión creada para ellos.

Tabla 5-3, Funciones incluidas en el achivo session.c

session.c
slurmdrmaa_session _new
slurmdrmaa_session _run_job
slurmdrmaa_session _run_bulk
slurmdrmaa_session _new_job

Por último en el archivo util.c, se cuenta con algunas funciones de utilidades adicionales para la especificación del estándar DRMAA.

Tabla 5-4, Funciones incluidas en el achivo util.c

util.c
slurmdrmaa_datetime_parse
slurmdrmaa_mail_type_parse
slurmdrmaa_init_job_desc
slurmdrmaa_free_job_desc
slurmdrmaa_add_attribute
slurmdrmaa_parse_additional_attr
slurmdrmaa_parse_native

Como se puede apreciar, el desarrollo de una implementación del estándar DRMAA para un sistema administrador de recursos distribuidos, es un trabajo que se debe planificar para ser realizado en etapas, en las que en cada una de ellas, el grupo de desarrolladores, basándose en las funcionalidades ya implementadas, realiza el estudio para adicionar alguna propiedad del sistema administrador de recursos distribuidos o para mejorar las características de una funcionalidad ya implementada en el estándar.

5.3.1 SLURM y Grid Computing

Sabiendo que el principal objetivo de la computación Grid, se centra en la unión de todo tipo de recursos computacionales, como lo son unidades de cálculo y almacenamiento. Esta unión se debe realizar teniendo presente las políticas de seguridad, las herramientas implementadas para la administración interna de las diferentes infraestructuras unidas a la Grid y cualquier política autónoma de gestión de usuarios o trabajos que se tenga establecida en cada entidad que se ha unido a la Grid.

Aunque SLURM trabaja a nivel de Cluster, con las funcionalidades incluidas en la actual implementación del estándar DRMAA, se obtiene una opción básica de crear aplicaciones que puedan ser portadas a otros sistemas administradores de recursos distribuidos, esto sin importar que trabajen a un nivel de Grid o a un nivel de Cluster.

En este orden de ideas, SLURM, pude interrelacionarse con sistemas administradores de recursos distribuidos, que se encuentren trabajando a nivel Grid de dos formas.

La primera sería estableciendo una capa Grid en la que SLURM actuaría como el administrador y planificador de recursos y tareas en cada uno de los Clusters que

se unen a la infraestructura Grid, quedando por sobre SLURM la capa Grid. En este caso la implementación del estándar DRMAA debe ser desarrollada para el sistema administrador de recursos distribuidos que se encuentre en la capa Grid.

La segunda es el caso en el que se desarrolla una completa implementación del estándar DRMAA para SLURM, contemplando todas las funcionalidades con las que dispone SLURM, junto a esa implementación, están las implementaciones que se deben realizar para todos los sistemas administradores de recursos distribuidos con los que se desea que SLURM interactúe. Aunque puede pensarse en que el trabajo de desarrollar una implementación para cada sistema por separado, es un trabajo bastante tedioso se debe aclarar que la implementación del estándar ya se ha comenzado a trabajar para algunos sistemas por los equipos de desarrolladores que utilizan cada herramienta. El objetivo del Global Grid Forum, es que en trabajos separados se vayan desarrollando las implementaciones del estándar DRMAA para cada uno de los sistemas administradores de recursos distribuidos y así poder desarrollar aplicaciones que se puedan migrar entre plataformas sin tener que asumir grandes cambios sobre la aplicación.

5.3.2 SLURM y Cloud Computing

Como ya se mencionó anteriormente, la computación en la nube propone básicamente el mismo modelo de agrupamiento de recursos distribuidos, y conectados a internet, con el fin de ofrecer una diversidad de recursos computacionales que se muestran al usuario final de forma simplificada, tal como lo hace la computación Grid.

La computación en la nube se rige y sigue el ámbito comercial. La idea, básicamente se centra en que grandes empresas, con una gran disponibilidad de recursos computacionales, permiten la utilización de una parte de sus recursos a

terceros obteniendo en contraprestación una retribución monetaria por el uso de dichos recursos.

En este modelo comercial del préstamo de recursos computacionales, estos pertenecen a una única empresa que es la encargada del ofrecimiento, mantenimiento y administración de la plataforma, limitando el acceso de los usuarios a los recursos, únicamente a la utilización de los mismos, sin la posibilidad de conocer el proceso interno del funcionamiento de la plataforma, como si es posible en un ambiente de computación Grid.

Se puede apreciar, que aunque el concepto de agrupamiento de recursos es esencialmente el mismo para computación Grid y computación en la nube, para este último esquema, los recursos son provistos por una única organización, que se encarga de todos los procesos de administración y organización de los mismos. Dada la autonomía de cada organización, cada una de ellas puede proponer su propia implementación de computación en la nube, y por ende no es fácil determinar si los recursos están sujetos o no a un control centralizado, y si se utilizan protocolos o estándares abiertos.

Por lo anterior, el desarrollo de una implementación del estándar DRMAA, para un ambiente de computación en la nube, se encuentra muy ligado a la disposición de la organización y el interés que esta tenga para que sus recursos puedan ser accedidos de forma estandarizada sin afectar sus políticas de funcionamiento y seguridad.

5.4 INTEGRACIÓN DE SLURM CON CIGRI

En el modelo de integración de los recursos de cómputo avanzado de la Universidad Industrial de Santander, se trabajó con tres herramientas que cumplen

la labor de administración de recursos y planificación de tareas, una de ellas trabaja a nivel Grid (CiGri), y dos a nivel de Cluster (OAR y SLURM).

Dado que la integración de las herramientas CiGri y OAR, ya se encuentra implementada mediante un API, que cumple con las características del estilo de arquitectura *Representational State Transfer* (REST), se procedió a realizar el desarrollo de un API para la integración de SLURM con CiGri, que de igual forma cumpla las restricciones que controlan el funcionamiento de los elementos de la arquitectura REST.

5.4.1 Representational State Transfer (REST)

Para el desarrollo del RestAPI, se define el siguiente conjunto de características que se deben cumplir en el desarrollo del API.

- Cliente Servidor

  Es necesario separar las cosas que pertenecen al servidor de las cosas que le pertenecen al cliente. Debido a esta separación es posible desarrollar los componentes por separado, mejorar la portabilidad de la interfaz y también mejora la escalabilidad.

- Sin estado

  Toda petición que realiza el cliente al servidor, debe contener toda la información necesaria para que pueda ser entendida por el servidor, sin la necesidad de consultar algún dato adicional almacenado previamente en la comunicación. EL cliente es el encargado de mantener guardado el estado de la sesión.

- Caché

Las respuestas a una petición deben poder ser etiquetadas como respuestas cacheables o no cacheable. Si una respuesta es cacheable, entonces al cliente cache se le da permiso para reutilizar la respuesta más tarde si se hace una petición equivalente.

- Interfaz uniforme

  La principal característica que distingue a REST del resto de estilos de arquitecturas de red es el énfasis de usar una interfaz uniforme entre los componentes. Aplicando los principios de generalidad de la ingeniería del software a los componentes de la interfaz, se simplifica la arquitectura del sistema global y la visibilidad de interacciones se mejora. Las implementaciones se separan de los servicios que proporcionan, lo que anima al desarrollo independiente.

- Sistema de capas

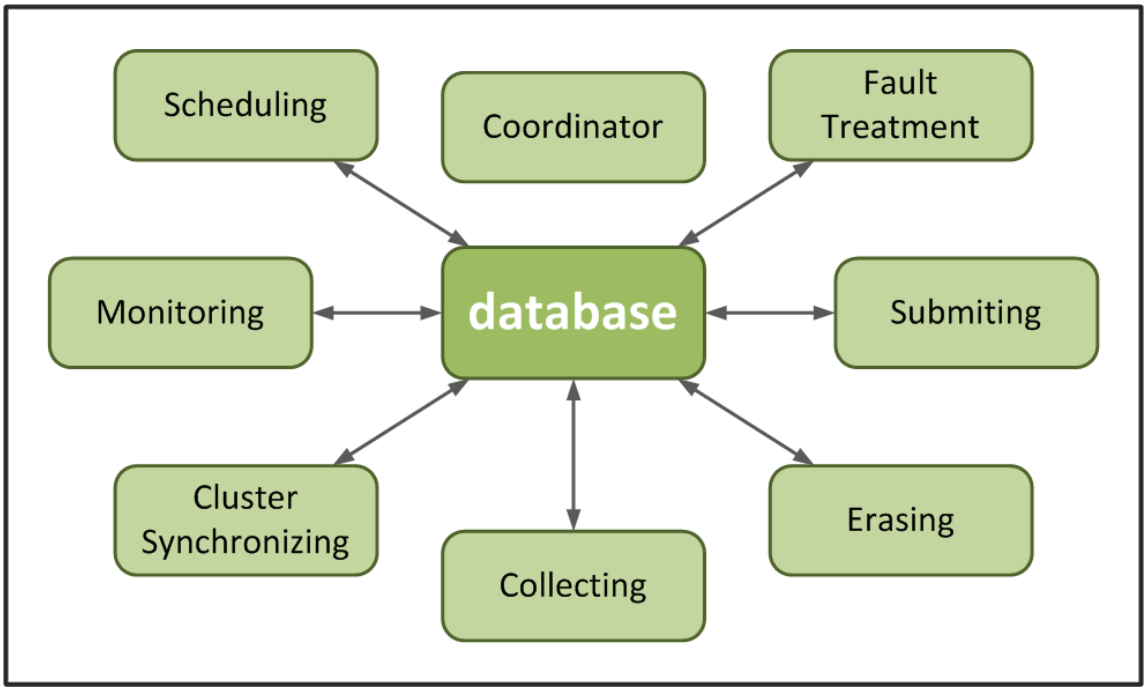
  Para poder mejorar el comportamiento de la escalabilidad en Internet, se añade la restricción del sistema de capas. Este sistema permite tener una arquitectura compuesta por capas jerárquicas, limitando el comportamiento de los componentes porque no pueden “ver” más allá de la capa con la que está interactuando.

- Código bajo demanda

  Esta característica es opcional y consiste en permitir a los clientes tener la funcionalidad de descargar y ejecutar código en forma de applets y scripts. Esto simplifica el lado del cliente porque reduce el número de funcionalidades que tiene que tener implementadas al crearse. Las funcionalidades se pueden descargar posteriormente aumentando así la extensibilidad del sistema.

5.4.2 Características de CiGri

Figura 5-6, Módulos CiGri



Fuente: Y. Georgiou, R. Olivier y N. Capi, «Evaluations of the lightweight grid CIGRI upon the Grid5000 platform,» 2007.

CiGri, básicamente está compuesto por un servidor que se comunica con todos los planificadores de tareas de los Clusters, mediante el envío de trabajos con la prioridad mínima, bajo el enfoque Best Effort. Como se puede apreciar en la figura 5-6, CiGri está basado en una arquitectura modular, en donde cada uno de los módulos se encarga de realizar una tarea en particular, interactuando con la base de datos. Todo el sistema es gestionado por un módulo central, que se encarga de coordinar el funcionamiento del resto de módulos para que realicen sus correspondientes tareas.

Tabla 5-5, Archivos de CiGri

/usr/local/share/cigri/		
api	cigri-api.rb	
	config.ru	
	launch_api.sh.in	
bin	griddel.rb	
	gridevents.rb	
	gridstat.rb	
	gridsub.rb	
lib	cigri-clientlib.rb	cigri-logger.rb
	cigri-clusterlib.rb	cigri.rb
	cigri-colombolib.rb	cigri-restclientlib.rb
	cigri-conflib.rb	cigri-runnerlib.rb
	cigri-eventlib.rb	cigri-scheduler-fifo.rb
	cigri-exception.rb	jdl-parser.rb
	cigri-iolib.rb	rack_debugger.rb
	cigri-joblib.rb	version.rb
modules	almighty.rb	
	meta-scheduler.rb	
	nikita.rb	
	runner.rb	
	updator.rb	
sbin	newcluster	

Dado que actualmente en el INIRIA GForge, se está trabajando en el desarrollo de la versión 3 de CiGri, y que esta versión va a presentar grandes cambios con respecto a la versión que de momento se encuentra en producción, se decidió trabajar con la última versión aunque sea susceptible a cambios en la versión final que sea lanzada como la versión estable del software. En la tabla 5-5, se puede

ver el directorio en el que es instalado CiGri3, junto con sus subdirectorios y los archivos que la componen para su funcionamiento.

Los archivos que se han intervenido para que CiGri pudiera interactuar con sistema planificador de tareas diferente de OAR son los siguientes:

- cigri-clientlib.rb
- cigri-clusterlib.rb
- cigri-logger.rb
- cigri-restclientlib.rb

6 CONCLUSIONES

- Se ofrece una especificación para la administración y gestión de trabajos en ejecución basada en DRMAA, adaptándolas a la infraestructura de cómputo de la UIS, teniendo en cuenta las características propias de la plataforma de prueba – GridUIS-2 – pero extensibles a otros sistemas de gran escala que integren recursos de cómputo de alto rendimiento.

- El análisis de las dos versiones de la especificación del estándar DRMAA, permitió reconocer en este, facilita la portabilidad de aplicaciones distribuidas en diferentes DRMS (Sistema Administrador de Recursos Distribuidos). En la versión 2 de esta especificación, el principal avance que es el incremento en el número de excepciones que se pueden tratar con el estándar, permitiendo que en las diferentes implementaciones del estándar, se pueda tratar una mayor cantidad de características inherentes a cada DRMS.

- El esquema en el tratamiento de los trabajos que se ejecutan mediante el estándar DRMAA, se debe contemplar el nivel de prioridad de cada uno de ellos, para tratar los casos en que los trabajos deban ser ubicados de acuerdo a las características de la aplicación. Igualmente hay que observar los casos en que una tarea con baja prioridad, deba ser reubicada en otro recurso computacional, para permitir que una tarea enviada por un usuario con mayor prioridad sobre dicho recurso, pueda ejecutarse de forma inmediata.

- Teniendo en cuenta el modelo de servicio que se observa en la computación en la nube, se puede considerar que un sistema que soporte

este tipo de computación esta fundamentalmente soportado por una Grid + Web Services (W.S.). La especificación estudiada y propuesta por este proyecto, permite ofrecer Infraestructura como Servicio (IaaS), sin embargo, es necesario acoplar este nivel a plataformas de acceso, dadas por los W.S., a través de portales, como es el caso del Science Gateway. [15] [16] [17]

- Para la arquitectura de GridUIS-2, y las herramientas que se encuentran en funcionamiento y en etapa de prueba dentro de la plataforma, la creación de un API siguiendo el estilo de arquitectura REST (Representation State Trasnfer), permite mejorar los tiempos de integración dado que CiGri se encuentra trabajando con este estilo de arquitectura para la interacción entre el servidor CiGri y los planificadores de tareas de los Clusters.

7 RECOMENDACIONES

- Se recomienda que partiendo del API, especificado para la conexión de CiGri con el planificador de tareas SLURM, dotar a este API, de todas las funciones que le permitan a los usuarios obtener el mejor rendimiento de esta herramienta por medio del envío de trabajos a través de CiGri.

- Es necesario que una vez la especificación del estándar DRMAA para CiGri este implementada, se desarrolle el API, que permita el envío de trabajos, que se comportan como grupos de tareas independientes las unas de las otras, para garantizar su interoperabilidad.

- Dado que la plataforma de computo avanzado GridUIS-2, tiene características que la definen como una plataforma de computación Grid y en procura de poder ofrecer servicios Cloud es necesario definir Web Service integrando la plataforma con Science Gateway.

- Debido al continuo desarrollo que se realiza sobre la herramienta planificadora de tareas SLURM, es necesario trabajar conjuntamente con su grupo de desarrolladores, para poder tener una implementación del estándar DRMAA que evolucione en conjunto con la herramienta.

8 BIBLIOGRAFÍA

[1] A. J. Burgarín Diz, Fronteras de la Computación, illustrated ed., Ediciones Díaz de Santos, 2002.

[9] Bull, «Extreme computing, SLURM guide,» BULL CEDOC, 2010.

[20] Bull, Architect of an open world, [En línea]. Available: http://www.bull.com/.

[4] C. C. Ruiz Sanabria, Análisis e implementación de una infraestructura de cálculo distribuid en la red universitaria, Bucaramanga, 2009.

[26] CiGri, [En línea]. Available: http://cigri.imag.fr/.

[15] GISELA Science Gateways, [En línea]. Available: http://gisela-gw.ct.infn.it/.

[22] Grid5000, [En línea]. Available: https://www.grid5000.fr/.

[21] Grupo de Supercomputación y Cálculo Científico SC3 UIS, [En línea]. Available: http://sc3.uis.edu.co/.

[11] I. Foster y C. Kesselmas, The Grid: Blueprint for a New Computing Infrastructure, New York: Morgan Kaufmann Publishers - Elsevier Inc., 2004.

[10] I. Foster, C. Kesselmas, J. M. Nick y S. Tuecke, «Grid Services for Distributed Systems Integration,» *IEEE Computer,* vol. 35, 2002.

[27] INRIA Forge, [En línea]. Available: https://gforge.inria.fr/projects/cigri/.

[2] K. Dowd y C. R. Severance, High Performance Computing, Sebastopol: O'Reilly & Associates, 1998.

[7] L. Grandinetti, High performance computing and grids in action, IOS Press, 2008.

[5] M. C. Mantilla Serrano y S. A. Orostegui Prada, Análisis y diseño de una

estrategia de interacción entre recursos distribuidos de una infraestructura de cómputo en redes heterogéneas, Bucaramanga, 2011.

[3] M. Colobran Huguet y E. M. Galindo, Administración de sistemas operativos en red, Barcelona: Editorial UOC, 2008.

[8] M. Jette y M. Grondona, «SLURM: Simple Linux Utility for Resource Management,» de *ClusterWorld Conference and Expo*, 2003.

[6] N. Capit y J. Emeras, «OAR-DOCUMENTATION-USER,» 16 Noviembre 2012. [En línea]. Available: http://oar.imag.fr/sources/2.5/docs/documentation/OAR-DOCUMENTATION-USER/. [Último acceso: 21 Enero 2013].

[23] OAR, [En línea]. Available: http://oar.imag.fr/.

[14] P. Mell y T. Grance, «The NIST Definition of Cloud Computing,» Septiembre 2011.

[13] R. Jamet y G. Huard, «Task aggregation in CiGri, a grid management middleware».

[16] Science Gateways Portal, [En línea]. Available: http://www.sciencegateway.org/.

[24] Slurm Workload Manager, [En línea]. Available: http://www.schedmd.com/slurmdocs/.

[25] SLURM: A Highly Scalable Resource Manager, [En línea]. Available: https://computing.llnl.gov/linux/slurm/.

[19] Supercomputación y Cálculo Científico UIS, [En línea]. Available: http://grid.uis.edu.co/.

[18] Universidad Industrial de Santander, [En línea]. Available: http://www.uis.edu.co/.

[17] XSEDE Science Gateway, [En línea]. Available:

https://www.xsede.org/gateways-overview/.

[12] Y. Georgiou, R. Olivier y N. Capi, «Evaluations of the lightweight grid CIGRI upon the Grid5000 platform,» 2007.

9 ANEXOS

A. Participación en el CLCAR 2012 (Conferencia Latinoamericana de Computación de Alto Rendimiento), con la presentación de un Paper llamado: HETEROGENEOUS RESOURCES AND PLATFORMS INTEGRATED WITH CIGRI: A LIGHTWEIGHT GRID SOLUTION, el cual fue elegido como parte del evento y consignado en las memorias del mismo.

Heterogeneous Resources and Platforms Integrated with CiGri: a Lightweight Grid Solution

Integración de Recursos Heterogéneos y Múltiples Plataformas Usando Estrategias de Grid Ligeros con CiGri

Díaz González, J., Orostegui Prada, S., Barrios Hernández, C.
Unidad de Supercomputación y Cálculo Científico
Universidad Industrial de Santander, Bucaramanga, Colombia
http://sc3.uis.edu.co/

Abstract

Due to growth of the processing and storage demand, companies and universities choose to increase computing capability and performance adding new resources. Often, involving interconnection with other platforms or gathering elements of the infrastructure already in place, generating large-scale architectures, composed by heterogeneous resources with different negative consequences (low fault tolerance, administration problems, bottlenecks and poor transparency). This paper shows a strategy proposed to increase performance and computing capacity in the context of a service of supercomputing and scientific computing in the Universidad Industrial de Santander in Bucaramanga, Colombia, using already campus resources in different levels with external resources provided by other institutions and consortiums.

Resumen

Tras la creciente demanda de procesamiento y almacenamiento, las empresas y universidades que prestan este servicio, optan por incrementar su capacidad de cómputo mediante la adición de nuevos recursos a las infraestructuras ya establecidas, generando arquitecturas de gran escala compuestas por recursos heterogéneos. Esta adición de recursos normalmente presenta diferentes dificultades (baja tolerancia a fallos, problemas de administración, poca transparencia). Este artículo presenta una estrategia propuesta para incrementar la capacidad y desempeño computacional en el contexto de un servicio de supercomputación y cálculo científico en la Universidad Industrial de Santander en Bucaramanga, Colombia, usando recursos ya existentes en diferentes niveles con recursos externos de otras instituciones y consorcios.

1 INTRODUCCIÓN Y CONTEXTO.

Las infraestructuras de computación y cálculo avanzado ubicadas en las universidades, proveen una amplia variedad de servicios, que ofrecen un gran soporte a la investigación y al desarrollo de las actividades académicas.

Estas infraestructuras se caracterizan por su escalabilidad y por la heterogeneidad de sus componentes, pues se crean por medio de la interconexión de diferentes recursos entre los que se puede contar con clusters, máquinas paralelas, dispositivos de captura y almacenamiento de datos, cluster ligeros (conformados principalmente por salas de cómputo), entre otros que se integran en una Grid [1].

Aunque en las universidades existen numerosos grupos de investigación con recursos de cómputo dispuestos especialmente para sus proyectos, estos suelen quedar cortos en su capacidad de procesamiento a medida que los cálculos van aumentando en tamaño y complejidad, por lo que se busca una integración de los equipos ubicados en cada uno de los laboratorio para incrementar la capacidad de cálculo de la universidad en general. [2]

A su vez, uno de los principales objetivos en la integración de los cluster distribuidos entre los laboratorios de las universidades, consiste en tener una mejor administración del total de los recursos, con lo que se busca que todos los usuarios tengan una forma sencilla de utilizar la plataforma sin la necesidad de pensar en las trabas que presenta la administración.

Los proyectos realizados conjuntamente entre los grupos investigativos dentro de las universidades, permiten compartir recursos de cómputo de cada uno de ellos, con el fin de establecer estructuras computacionales que suplan las múltiples necesidades. La unión de estos recursos se puede ver afectada por alguna de las siguientes implicaciones:

- Middleware y ambientes diferentes.
- Herramientas de administración diferentes.
- Usos diferentes.
- Diferencia en el hardware.
- Disponibilidad variable.

Existen proyectos que buscan atacar estas implicaciones como lo es el proyecto CIMENT[1], de la Universidad Joseph Fourier de Grenoble[2], Francia, y cuyo principal objetivo, es ayudar al desarrollo de proyectos de la comunidad académica de investigadores en computación de alto rendimiento.

La infraestructura de cómputo avanzado de los diferentes proyectos en la Universidad Industrial de Santander[3], que se han agregado a GridUIS-2[4], tienen recursos y herramientas de administración diferentes, por lo que se tiene una infraestructura completamente heterogénea que dificulta la administración global de todos los recursos.

Teniendo en cuenta la experiencia del proyecto CIMENT, se plantea una estrategia de integración de los recursos de GridUIS-2, garantizando su interacción con otras Grid, como es el caso de Grid Colombia[5], GISELA[6] y Grid SCALA.

[1] https://ciment.ujf-grenoble.fr/ [6]
[2] http://www.ujf-grenoble.fr/ [7]
[3] http://www.uis.edu.co/ [8]
[4] http://grid.uis.edu.co/ [9]
[5] http://www.gridcolombia.org/ [10]

A continuación se presenta una descripción de la infraestructura, los mecanismos de administración, los problemas presentados en los clusters heterogéneos y los mecanismos en estudio para dar solución a esta problemática en la infraestructura GridUIS-2, que se presenta en la siguiente sección.

2 GRIDUIS-2.

La Universidad Industrial de Santander cuenta con múltiples grupos de investigación que adelantan trabajos en diversas áreas del conocimiento, algunos de ellos realizando simulaciones que demandan grandes capacidades de procesamiento y almacenamiento. Estos centros investigativos buscan suplir dichas necesidades mediante la implementación de estructuras de cómputo con arquitecturas heterogéneas, acorde a la naturaleza de sus proyectos.

GridUIS-2, es la plataforma de computación distribuida de la Universidad Industrial de Santander, que soporta las actividades científicas y académicas que implican el uso de computación de alto rendimiento. En su fase actual, la plataforma está compuesta por diferentes recursos computacionales distribuidos físicamente dentro del campus principal, y en el Parque Tecnológico de Guatiguará (PTG), Piedecuesta, Colombia.

Entre las diferentes soluciones que se establecieron para el desarrollo de GridUIS-2, está la principal unidad de cálculo científico denominada GUANE-1, unida a los clusters implementados por los diferentes grupos de investigación.

Como una opción de apoyo y buscando la optimización en el uso de los recursos computacionales dispuestos en el campus universitario, se tiene implementado un cluster ligero mediante la herramienta Computemode[7], que aprovecha el tiempo ocioso de los computadores de escritorio utilizados habitualmente para apoyar el desarrollo de las clases. [3]

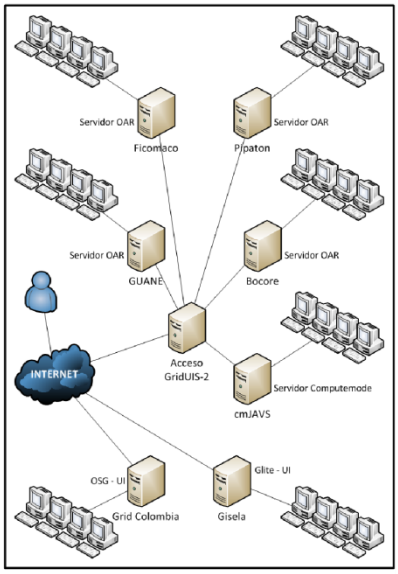


Figura 1: Estructura GridUIS-2.

Adicional a las plataformas estructuradas por los grupos de investigación de la universidad y las soluciones cluster ya mencionadas, GridUIS-2 se encuentra unida a los proyectos GRID Colombia y GISELA, como se muestra en la figura 1. Estos proyectos permiten la interacción de GridUIS-2

[6] http://www.gisela-grid.eu/ [11]

[7] http://computemode.imag.fr/ [12]

con recursos de cómputo de alto rendimiento distribuidos geográficamente a nivel nacional e internacional.

A continuación, en la tabla 1 se presentan algunas características de los clusters de GridUIS-2, como el número de nodos en cada cluster, el número de core's de cada nodo y el número de tarjetas Nvidia incluidas en cada nodo y utilizadas para procesamiento cuda.

Cluster	Número de Nodos	Core's / nodo	GPU's / nodo
GUANE	8	5 (8) 3 (12)	8
Ficomaco	3	4	1
Pipaton	6	2	- - -
Bocore	6	2	- - -
cmJAVS	22	2	- - -
Grid Colombia	2	2	- - -
GISELA	3	2	- - -

Tabla 1: Componentes de GridUIS-2.

2.1 ARQUITECTURA.

GridUIS-2, une recursos de cómputo de diferentes centros de investigación, por lo que algunas de las arquitecturas presentes en la plataforma se diferencian unas de otras.

En general la arquitectura de los cluster incluidos en la plataforma GridUIS-2, como se muestra en la figura 2, utiliza un nivel de red al que están directamente conectados todos los nodos. Cada cluster funciona con un sistema operativo que es el mismo para cada uno de los nodos dentro del cluster, pero que en algunos casos se diferencia entre los clusters de la plataforma. La distribución de las aplicaciones en caso de ser necesario está a cargo del middleware. Por último la administración de la disponibilidad y el acceso a los recursos de la plataforma se realiza por medio de un sistema manejador de colas. [4]

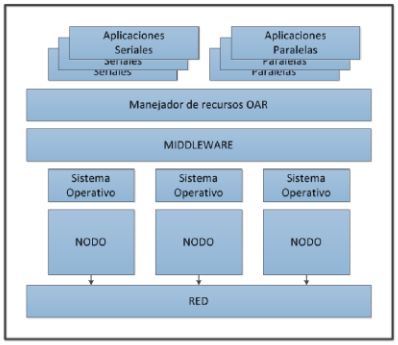


Figura 2: Arquitectura GridUIS-2

2.2 ASPECTOS ADMINISTRATIVOS Y DE CONFIGURACIÓN.

Dadas las características heterogéneas de las soluciones de computación de alto rendimiento implementadas en los diferentes centros investigativos de la universidad, y las necesidades inherentes a los trabajos que se realizan en cada uno de ellos, se hace necesaria la utilización de una considerable variedad de software.

Para la administración de los recursos de cada cluster es necesario un sistema manejador de colas que permita el ingreso a los recursos de una forma ordenada. Entre los sistemas que realizan este tipo de tareas se encuentra PBS[8], Torque[9], LSF[10], SLURM[11], OAR[12], entre otros.

[8] http://www.pbsworks.com/ [13]
[9] http://www.adaptivecomputing.com/products/open-source/torque/ [14]
[10] http://www.platform.com/workload-management/high-performance-computing/lp [15]
[11] http://www.schedmd.com/slurmdocs/ [16]
[12] http://oar.imag.fr/ [17]

En el caso de GridUIS-2, OAR es el sistema manejador de colas utilizado principalmente en la mayoría de los clusters. Actualmente se esta implementando SLURM como manejador de recursos en algunos de los clusters de la plataforma.

OAR es un sistema manejador de colas basado en una base de datos (MySQL o PostgreSQL), un lenguaje de script (Perl), y una herramienta opcional de administración escalable (Taktuk). Este sistema está compuesto por módulos que interactúan únicamente con la base de datos y se ejecutan como programas independientes. Así formalmente, no hay un API, el sistema está definido completamente por el esquema de la base de datos. Este enfoque facilita el desarrollo de módulos independientes, en los que cada uno de ellos puede ser desarrollado en cualquier lenguaje que tenga acceso a una librería de base de datos.

SLURM es un sistema de código abierto, administrador de los recursos clusters altamente escalable, con tolerancia a fallas y organizador de trabajos para pequeños y grandes clusters Linux. Es un sistema relativamente autónomo y no requiere modificaciones en el kernel.

Para el despliegue de imágenes sobre los nodos de GridUIS-2, se utiliza la herramienta denominada Kadeploy[13], que se encuentra integrada con OAR. El despliegue de imágenes no se puede ejecutar sobre clusters oportunistas, ya que necesita de una configuración especial en la tabla de particiones para poder funcionar correctamente.

Para darle provecho a los tiempos libres de los computadores se implementó la herramienta Computemode, la cual nos permite agregar recursos a un cluster ligero de forma no intrusiva. El manejador de colas utilizado por esta herramienta es OAR y aunque se tiene estrategia de tolerancia a fallas no es confiable. Su uso es solo para procesos académicos.

Un servidor LDAP[14] (*Lightweight Directory Access Protocol*), herramienta ampliamente utilizada en ambientes distribuidos, que ofrece la opción de centralizar las cuentas de los usuarios utilizando una base de datos en forma de árbol, que permite la especificación de perfiles y la determinación de los niveles de interacción de los usuarios en la plataforma. Estas cuentas de usuario permiten el uso autorizado de los servicios de procesamiento y cálculo científico en GridUIS-2, además del uso de otros servicios como la wiki.

Protocolo NFS (*Network File System*), por medio del cual se permite la réplica de los espacios de trabajo en toda la infraestructura.

Adicional a estas herramientas se tiene el software necesario para la administración y funcionamiento de los proyectos Grid Colombia y GISELA. Cada uno de estos proyectos necesita de una configuración especial para la administración y uso de estas plataformas.

La heterogeneidad de los clusters, presenta un problema en la integración de los recursos bajo una estrategia que simplifique los procesos de administración, y facilite el uso de la plataforma por parte de la comunidad, que se ve restringida a usar estos arreglos de computadores de forma separada.

[13] http://kadeploy.imag.fr/ [18]

[14] http://www.openldap.org/ [19]

Establecido el panorama de la infraestructura GridUIS-2, se puede percibir la separación que existe entre los recursos, dados los diferentes servicios ofrecidos en cada infraestructura por separado, a demás de la heterogeneidad de los dispositivos.

3 ESTRATEGIA DE SOLUCIÓN.

CIGRI[15] es una estrategia de computación a gran escala, basada en el concepto de Grid ligera, que busca aprovechar el tiempo de ocio de los clusters y cuyas principales características son su simplicidad, la escalabilidad y los mecanismos de tolerancia a fallas.

Para el funcionamiento de CIGRI no es necesario la instalación de software adicional en los clusters, pues el sistema se compone de un servidor central (servidor CIGRI) que se encarga de la comunicación con los batch scheduler de cada cluster, a los que realiza el envío de los trabajos, actuando como un usuario más del cluster, y con un mínimo de prioridad en sus trabajos. [5]

Actualmente CIGRI únicamente soporta trabajos denominados "Bag of Task (BoT)", que son grupos de tareas enviados a cada cluster por separado, y que ejecutan una parte específica del trabajo.

Dada la posibilidad de paradas inesperadas en la ejecución de los trabajos debido a fallas en un nodo o a la demanda de los recursos por parte de un usuario, CIGRI cuenta con un mecanismo de tolerancia a fallas que establece unos puntos de control (checkpoint), para que en caso de una parada abrupta del proceso, se pueda retomar el trabajo desde un punto determinado.

CIGRI, funciona como un usuario más de la plataforma, que recibe los trabajos de los usuarios y los redirige a los clusters, por lo que de este modo cada usuario ya no tendrá inconvenientes con la necesidad de escoger los recursos sobre los cuales se ejecutarán sus aplicaciones, pues en ese caso tan solo tendrían que dirigirse los trabajos a CIGRI, y de ahí se encolarían a la plataforma.

Actualmente CIGRI tan solo puede integrar a los clusters que utilizan como batch scheduler a OAR.

Dado el hecho de que en GridUIS-2, se cuenta con varios clusters que utilizan el manejador de colas OAR, se presenta a CIGRI como una estrategia válida para la integración de algunos de los recursos disponibles en esta Infraestructura.

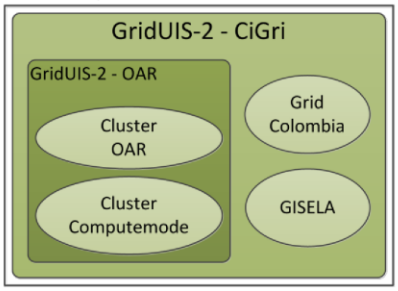


Figura 3: Integración de clusters bajo CIGRI.

Como ya se mencionó anteriormente y debido a que algunos de los proyectos vinculados a GridUIS-2, utilizan sistemas administradores de recursos diferentes de OAR, de momento no es posible la integración de estos recursos bajo el enfoque de Grid ligera CIGRI.

[15] http://cigri.imag.fr/ [20]

Entre los proyectos que utilizan sistemas administradores de recurso diferentes de OAR se tiene Grid Colombia, que utiliza Globus[16] y Condor[17] para la administración de los recursos de la plataforma. Por otra parte también está el proyecto GISELA que utiliza gLite[18] como middleware. Otro de los inconvenientes presentados con estos proyectos es la necesidad de un usuario debidamente identificado ante cada uno de las organizaciones para poder hacer uso de los recursos, ya que CIGRI funciona sin la estricta identificación de los usuarios como si lo hacen estos proyectos. Por lo anterior es necesario establecer una estrategia que permita la conexión de los usuarios a los proyectos externos a través de CIGRI de una manera transparente.

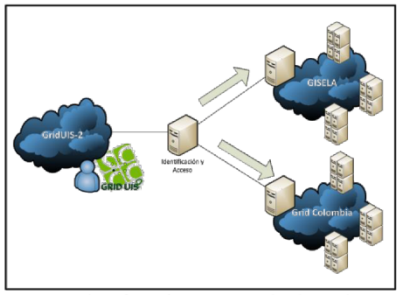


Figura 4: Identificación y accesos desde GridUIS-2

En la figura 4, se representa la forma en que se pretende realizar la validación y el ingreso de los usuarios de GridUIS-2, a las plataformas de cómputo Grid Colombia y/o GISELA, siempre y cuando cuenten con los respectivos certificados de usuario que los acredite y les permita el uso de éstas plataformas.

[16] http://www.globus.org/ [21]
[17] http://research.cs.wisc.edu/condor/ [22]
[18] http://glite.cern.ch/ [23]

4 CONCLUSIONES Y TRABAJO FUTURO.

Teniendo en cuenta que varios de los clusters que actualmente pertenecen a GridUIS-2, utilizan el manejador de colas OAR, para la administración de los recursos y los trabajos, se postula a CIGRI, como la mejor opción para la integración de la infraestructura, bajo una herramienta que permite centralizar la administración de los recursos, y ofrecer un uso de la plataforma transparente para la comunidad de usuarios.

Dado que de momento CiGri es capaz de integrar únicamente los clusters que trabajan con el batch scheduler OAR, actualmente se están realizando estudios que permitan determinar los mecanismos necesarios para la integración de las plataformas que utilizan sistemas de administración de recursos como SLURM, PBS, Condor y otros.

En cuanto a los proyectos Grid Colombia y GISELA, se está evaluando la forma en que los usuarios puedan utilizar sus certificados de cada uno de estos proyectos para poder conectarse a través de CIGRI de forma segura sin la necesidad de realizar autentificaciones adicionales.

Entre las pruebas que se realizarán al terminar de integrar los recursos de cómputo bajo el enfoque de grid ligera CiGri, se evaluará con principal atención la confiabilidad del mecanismo de tolerancia a fallas incluido en este sistema.

Aunque el proyecto se encuentra actualmente en fase de desarrollo e implementación, al finalizar la integración se espera que los usuarios puedan acceder de una forma simple a los recursos distribuidos de la plataforma sin preocupase por

como moverse entre las diferentes estructuras de cómputo de GridUIS-2. Al tiempo se pretende simplificar la administración de la plataforma en general, por medio de la centralización del sistema manejador de los recursos y los trabajos.

5 AGRADECIMIENTOS

Los autores agradecen a Yiannis Georgieu, de Bull Inc. por sus contribuciones.

Los resultados de los experimentos presentados en esta publicación fueron obtenidos usando la plataforma GridUIS-2, desarrollada por el Servicio de Cómputo de Alto Rendimiento y Cálculo Científico de la Universidad Industrial de Santander (UIS). Esta acción es soportada por la Vicerrectoría de Investigación y Extensión de la UIS (VIE-UIS) y diferentes grupos de investigación de la universidad. (http://grid.uis.edu.co)

6 REFERENCIAS.

[1] I. Foster y C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, New York, United States of America: Morgan Kaufmann Publishers - Elsevier Inc., 2004.

[2] I. Foster, C. Kesselman, J. M. Nick y S. Tuecke, Grid Services for Distributed Systems Integration, United States of America: IEEE Computer Volume 35, 2002.

[3] M. Mantilla Serrano, S. Orostegui Prada, R. Uribe Espinosa, C. Ruiz Sanabria, J. Escobar Ramirez y C. Barrios Hernandez, «Strategies to Fault Tolerance and hierarchical networks interaction for Lightweight Clustering,» 2010.

[4] S. Orostegui Prada, R. Uribe Espinosa, M. Mantilla Serrano, I. Gómez Pedraza, C. Varela Garzón, A. Lobo Figueroa, L. Camargo Forero, J. Escobar Ramírez, J. Chacón Velasco, C. Barrios Hernández y D. Sierra Bueno, «GridUIS-2: Lightweight and Dedicated Architectures Integration on a Platform to High Performance and Scientific Computing,» 2011.

[5] Y. Georgiou, O. Richard y N. Capit, «Evaluations of the lightweight grid CIGRI upon the Grid5000 platform,» 2007.

[6] «CIMENT,» [En línea]. Available: https://ciment.ujf-grenoble.fr/. [Último acceso: Julio 2012].

[7] «Université Joseph Fourier,» [En línea]. Available: http://www.ujf-grenoble.fr/. [Último acceso: Junio 2012].

[8] «Universidad Industrial de Santander,» [En línea]. Available: http://www.uis.edu.co/. [Último acceso: Junio 2012].

[9] «GridUIS-2,» [En línea]. Available: http://grid.uis.edu.co/. [Último acceso: Agosto 2012].

[10] «Grid Colombia,» [En línea]. Available: http://www.gridcolombia.org/. [Último acceso: Junio 2012].

[11] «GISELA,» [En línea]. Available: http://www.gisela-grid.eu/. [Último acceso: Junio 2012].

[12] «Computemode,» [En línea]. Available: http://computemode.imag.fr/. [Último acceso: Julio 2012].

[13] «PBS Works,» [En línea]. Available: http://www.pbsworks.com/. [Último acceso: Julio 2012].

[14] «Adaptive Computing,» [En línea]. Available: http://www.adaptivecomputing.com/products/open-source/torque/. [Último acceso:

Julio 2012].

[15] «Platform Computing,» [En línea]. Available: http://www.platform.com/workload-management/high-performance-computing/lp. [Último acceso: Julio 2012].

[16] «SLURM,» [En línea]. Available: http://www.schedmd.com/slurmdocs/. [Último acceso: Agosto 2012].

[17] «OAR,» [En línea]. Available: http://oar.imag.fr/. [Último acceso: Agosto 2012].

[18] «KADEPLOY,» [En línea]. Available: http://kadeploy.imag.fr/. [Último acceso: Julio 2012].

[19] «Open LDAP,» [En línea]. Available: http://www.openldap.org/. [Último acceso: Junio 2012].

[20] «CiGri,» [En línea]. Available: http://cigri.imag.fr/. [Último acceso: Agosto 2012].

[21] «Globus,» [En línea]. Available: http://www.globus.org/. [Último acceso: Julio 2012].

[22] «Condor High Throughput Computing,» [En línea]. Available: http://research.cs.wisc.edu/condor/. [Último acceso: Julio 2012].

[23] «gLite - Lightweight Middleware for Grid Computing,» [En línea]. Available: http://glite.cern.ch/. [Último acceso: Julio 2012].