IMPLEMENTACIÓN DE DOS *NODOS GRID* BASADOS EN *CLUSTERS* E INTEGRADOS A GRID COLOMBIA A TRAVÉS DE RENATA UTILIZANDO SOFTWARE LIBRE

JORGE IVÁN MEZA MARTÍNEZ

UNIVERSIDAD AUTÓNOMA DE MANIZALES
FACULTAD DE INGENIERÍA
MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE
MANIZALES
2011

IMPLEMENTACIÓN DE DOS *NODOS GRID* BASADOS EN *CLUSTERS* E INTEGRADOS A GRID COLOMBIA A TRAVÉS DE RENATA UTILIZANDO SOFTWARE LIBRE

JORGE IVÁN MEZA MARTÍNEZ

TRABAJO DE TÉSIS PARA OPTAR EL TÍTULO DE MSC EN GESTIÓN Y DESARROLLOS PROYECTOS DE SOFTWARE

ASESORA
ANA LORENA URIBE HURTADO
INGENIERA DE SISTEMAS, MSC.
DOCENTE ACADÉMICO

UNIVERSIDAD AUTÓNOMA DE MANIZALES
FACULTAD DE INGENIERÍA
MAESTRÍA EN GESTIÓN Y DESARROLLO DE PROYECTOS DE SOFTWARE
MANIZALES
2011

CONTENIDO

Pág.

INTRODUCCIÓN....14
1. REFERENTE CONTEXTUAL....17
1.1 PLANTEAMIENTO DEL PROBLEMA....17
1.2 ANTECEDENTES....18
1.3 JUSTIFICACIÓN....22
1.4 DELIMITACIÓN DEL ÁREA PROBLEMÁTICA....25
1.5 OBJETIVOS....27
1.5.1 Objetivo General....27
1.5.2 Objetivo Específico....27
1.6 RESULTADOS ESPERADOS....28
2. ESTRATEGIA METODOLÓGICA....30
2.1 METODOLOGÍA....30
2.2 PRUEBAS....32
2.2.1 Nivel de cluster....33
2.2.2 Nivel de grid....33
2.3 PRESUPUESTO....34
3. DESARROLLO....37
3.1 REFERENTE TEÓRICO....37
3.1.1 Software libre....37
3.1.2 El sistema operativo GNU/Linux....43
3.1.3 Sistemas distribuidos....48
3.1.4 Clusters....52
3.1.5 Condor....56
3.1.6 Mallas computacionales o grids....61
3.1.7 Middleware o capa de abstracción intermedia....68
3.1.8 Open Science Grid (OSG)....69
3.1.9 Grid Colombia....72
3.1.10 Redes académicas de alta velocidad....76
3.1.10.1 Red Nacional Académica de Tecnología Avanzada (RENATA)....78
3.2 PERSPECTIVA GENERAL DEL DESARROLLO....82
3.3 INVENTARIO DE LA INFRAESTRUCTURA TECNOLÓGICA UTILIZADA....88
3.4 ARQUITECTURA DE LOS COMPONENTES....88
3.4.1 Nivel de cluster....89
3.4.2 Nivel de grid....95
3.4.2.1 Componentes centralizados....100
3.4.2.2 Componentes locales....102
3.4.2.2.1 Nodo grid pequeño....102

3.4.2.2.2 Nodo grid mediano....104
3.5 IMPLEMENTACIÓN DEL NIVEL DE CLUSTER....105
3.5.1 Instalación y configuración del cluster....106
3.5.2 Listas de verificación para la instalación y configuración....112
3.5.3 Solución de problemas para la instalación y configuración....113
3.5.4 Administración y uso del cluster....114
3.5.4.1 Administración general del cluster....114
3.5.4.2 Universos del cluster....116
3.5.4.3 Envío de múltiples subtrabajos al cluster....117
3.5.5 Solución de problemas para la administración y uso....119
3.6 IMPLEMENTACIÓN DEL NIVEL DE GRID....120
3.6.1 Instalación y configuración del nodo grid....120
3.6.1.1 Acerca de los certificados....121
3.6.1.2 Instalación y configuración de los componentes....124
3.6.1.2.1 Compute Element (CE)....125
3.6.1.2.1.1 Procedimiento....125
3.6.1.2.1.2 Solución de problemas....128
3.6.1.2.2 Cliente Grid (UI)....129
3.6.1.2.3 Grid User Management System (GUMS)....130
3.6.1.3 Pruebas de conexión y uso básicas....132
3.6.1.4 Configuración de firewalls....133
3.6.2 Administración y uso del nodo grid....134
3.6.2.1 Cargar las variables de ambiente....134
3.6.2.2 Gestión de los intermediarios de los certificados....135
3.6.2.3 Transferencia de archivos al nodo grid....136
3.6.2.4 Envío de trabajos al nodo grid....136
3.6.3 Solución de problemas para la administración y uso del nodo grid....139
3.7 SOFTWARE INSTALADO....141
3.8 ANÁLISIS DE RESULTADOS....143
4. CONCLUSIONES....151
5. RECOMENDACIONES....154
BIBLIOGRAFÍA....159
ANEXOS....161
ANEXO I: PRUEBAS FUNCIONALES DE LOS NODOS....161
Nodo Universidad Autónoma de Manizales (UAM)....161
Nivel de cluster....161
Verificar conectividad con el nodo principal del cluster....161
Verificar conectividad SSH con el nodo principal del cluster....161
Verificar conectividad con los nodos trabajadores del cluster....162
Verificar conectividad SSH con los nodos trabajadores del cluster....163
Verificar el estado del sistema de archivos compartido (NFS) en el cluster....164
Verificar el pool de nodos trabajadores del cluster Condor....166

Verificar el soporte para Java en los nodos trabajadores del cluster.....167
Verificar la cola de trabajos del cluster Condor.....169
Probar el envío de trabajos al universo Vanilla del cluster.....169
Probar el envío de trabajos al universo Standard del cluster.....170
Probar el envío de trabajos al universo Java del cluster.....174
Nivel de grid.....176
Verificar conectividad con entre el Cliente Grid y el Compute Element.....176
Cargar y verificar las variables de ambiente del Cliente Grid.....177
Crear y verificar el intermediario del certificado.....177
Verificar la autenticación con el Nodo Grid.....178
Probar el envío de trabajos al Nodo Grid con las herramientas de Globus.....178
Utilizando el manejador de trabajos Fork.....178
Utilizando el manejador de trabajos de Condor.....178
Probar el envío de trabajos al Nodo Grid utilizando Condor-G.....178
Probar la transferencia de archivos entre el Cliente Grid y el Nodo Grid.....180
Transferir el archivo del Cliente Grid al Nodo Grid.....180
Transferir el archivo de regreso, desde el Nodo Grid hacia el Cliente Grid. 180
Comparar el archivo transmitido inicialmente con el finalmente recibido. 181
Destruir y verificar el intermediario del certificado.....181
Nodo Universidad del Rosario (UR).....182
Nivel de cluster.....182
Verificar conectividad con el nodo principal del cluster.....182
Verificar conectividad SSH con el nodo principal del cluster.....182
Verificar conectividad con los nodos trabajadores del cluster.....182
Verificar conectividad SSH con los nodos trabajadores del cluster.....184
Verificar el estado del sistema de archivos compartido (NFS) en el cluster....185
Verificar el pool de nodos trabajadores del cluster Condor.....186
Verificar el soporte para Java en los nodos trabajadores del cluster.....187
Verificar la cola de trabajos del cluster Condor.....188
Probar el envío de trabajos al universo Vanilla del cluster.....188
Probar el envío de trabajos al universo Standard del cluster.....190
Probar el envío de trabajos al universo Java del cluster.....193
Nivel de grid.....196
Verificar conectividad con entre el Cliente Grid y el Compute Element.....196
Cargar y verificar las variables de ambiente del Cliente Grid.....196
Crear y verificar el intermediario del certificado.....197
Verificar la autenticación con el Nodo Grid.....197
Probar el envío de trabajos al Nodo Grid con las herramientas de Globus.....197
Utilizando el manejador de trabajos Fork.....197
Utilizando el manejador de trabajos de Condor.....198
Probar el envío de trabajos al Nodo Grid utilizando Condor-G.....198

Probar la transferencia de archivos entre el Cliente Grid y el Nodo Grid.....200
Transferir el archivo del Cliente Grid al Nodo Grid.....200
Transferir el archivo de regreso, desde el Nodo Grid hacia el Cliente Grid. 200
Comparar el archivo transmitido inicialmente con el finalmente recibido. 200
Destruir y verificar el intermediario del certificado.....200

LISTA DE GRÁFICAS

Pág.

Gráfica 1: Arquitectura típica de Condor.....58
Gráfica 2: Topología de la red RENATA.....80
Gráfica 3: Arquitectura de cluster implementada en la Universidad Autónoma de Manizales 94
Gráfica 4: Arquitectura de cluster implementada en la Universidad del Rosario.....94
Gráfica 5: Arquitectura de los componentes de un Nodo Grid pequeño.....103
Gráfica 6: Arquitectura de los componentes de un Nodo Grid mediano.....105

RESUMEN

Este trabajo describe el desarrollo del proyecto GridUAM que constituye un esfuerzo de la Universidad Autónoma de Manizales para implementar su propia infraestructura computacional de alto desempeño para brindar soporte a los proyectos académicos y de investigación de sus propias facultades y centros de investigación de la región.

El desarrollo del proyecto se realiza en dos fases principales. La primera de ellas consiste en implementar un nivel de *cluster* que se encargará de establecer la infraestructura, tanto de hardware como software, necesaria para el procesamiento de trabajos que requieran de alto rendimiento. Para esto se utiliza a Condor[1] como *framework* HTC[2]. En el proyecto se utilizaron recursos de hardware dedicados, sin embargo será factible en un futuro integrar al nodo recursos con una dedicación de uso parcial.

La segunda fase consiste en implementar el nivel de *grid* que permite conectar los recursos computacionales de la fase anterior con las demás iniciativas nacionales a través del proyecto Grid Colombia[3]. De esta manera es posible que los

[1] Condor - High Throughput Computing (HTC). Página web. [en línea] <http://www.cs.wisc.edu/condor/> [Consultado en julio 10 de 2011]
[2] Computación de alto rendimiento (*High Throughput Computing*).
[3] Grid Colombia. Página web. [en línea] <http://gridcolombia.org/> [Consultado en julio 10 de 2011]

investigadores asociados al nodo puedan tener acceso a recursos computacionales externos para la ejecución de sus trabajos así como el poder facilitar el uso de los recursos computacionales propios de la Universidad asociados al nodo para la ejecución de trabajos por parte de los demás integrantes de la comunidad de Grid Colombia. Para la implementación de este fase se aprovecha la infraestructura y el apoyo por Open Science Grid[4] quienes dan soporte a la *grid* de Estados Unidos de Norte América.

Cada una de estas fases principales es analizada desde las perspectivas de los usuarios administradores quienes se encargan de la instalación, configuración y pruebas de la infraestructura y desde la perspectiva del usuario final quienes se encargan de la administración y uso de los servicios que esta infraestructura *cluster*/*grid* les provee.

En una fase adicional, transversal al desarrollo del proyecto, se realiza la documentación exhaustiva del diseño e implementación de las dos fases principales en el wiki del proyecto[5]. Esto con el fin de hacer explícito el conocimiento adquirido durante el proyecto y de presentarlo de manera que otras instituciones interesadas en recorrer el mismo camino puedan aprovecharlo e

[4] Open Science Grid. Página web. [en línea] <http://www.opensciencegrid.org/> [Consultado en julio 10 de 2011]

[5] Sitio wiki del proyecto GridUAM. Página web. [en línea] <http://griduam.esencial.co/> [Consultado en julio 10 de 2011]

implementar con él, sus propias infraestructuras *grid* destinando un menor tiempo y esfuerzo para esto.

Palabras clave

cluster, *grid*, grid colombia, open science grid, osg, globus, htc, high-throughput computing, free software

ABSTRACT

This paper describes the development of the GridUAM's project that constitutes an effort of the Autonoma of Manizales University to implement its own high throughput computing infrastructure to provide support to the academic and research projects from their own faculties and research centers in the region.

The development of the project is done in two main phases. The first one consists in the implementation of a cluster level that will establish the infrastructure, both hardware and software, required to process high throughput jobs. To acquire this was used Condor[6] as the HTC[7] framework. The project used dedicated hardware resources, however it's feasible in the future integrate resources with a partial hardware use dedication to the node.

The second phase consists in the implementation of the grid level that connects the local computational resources from the previous stage with the national initiative through the Grid Colombia[8] project. This makes possible for researchers associated to the node access external computing resources provided by Grid Colombia's partners for the execution of their jobs. In the same way, facilitate the

[6] Condor - High Throughput Computing (HTC). Web page. [on line] <http://www.cs.wisc.edu/condor/> [Consulted on 07-10-2011]

[7] *High Throughput Computing*

[8] Grid Colombia. Web page. [on line] <http://gridcolombia.org/> [Consulted on 07-10-2011]

use of local computational resources for the execution of remote jobs by other members of the Grid Colombia's community. To achieve this phase the project draws in the infrastructure and guidance of Open Science Grid[9] who supports the main grid infraestructure in North America.

Each of these major phases are analyzed from the perspectives of the administrator users who are responsible for installation, configuration and testing of the infrastructure and from the end users perspective who are responsible for the administration and use of the services provided by the cluster/grid deployed infrastructure.

An additional cross-project phase is responsible for the documentation of the design and implementation of the two main phases. It is performed in the project's wiki site[10]. The main objective of this final stage is to make explicit the knowledge obtained during the project's development and present it in a way that other institutions interested in walk over the same path can take advantage of it to implement its own grid infrastructure devoting less time an effort for this.

[9] Open Science Grid. Web site. [on line] <http://www.opensciencegrid.org/> [Consulted on 07-10-2011]
[10] Project's wiki site. Web page. [on line] <http://griduam.esencial.co/> [Consulted on 07-10-2011]

Keywords

cluster, grid, grid colombia, open science grid, osg, globus, htc, high-throughput computing, free software

INTRODUCCIÓN

Este proyecto implementa una infraestructura física y lógica para el despliegue de dos nodos *grid* -uno en la Universidad Autónoma de Manizales y otro en la Universidad del Rosario en Bogotá- con los cuales se fortalece la infraestructura de cómputo de esas universidades aprovechando las ventajas de las tecnologías de computación distribuida, permitiéndoles a la academia y a las demás instituciones de investigación asociadas acceder a una mayor capacidad de cómputo y almacenamiento en comparación con el disponible de manera individual en cada una de las instituciones. Esto a su vez les permite enfrentarse a problemas de investigación cada vez más elaborados y resolverlos en una menor cantidad de tiempo.

Cada uno de estos nodos *grid* se basa en un *cluster* computacional sobre el cual se soportará la infraestructura de *grid*. Para la implementación del nivel de *cluster* se utilizó el software Condor[11], desarrollado por la Universidad de Wisconsin[12] el cual es ampliamente utilizado para soportar procesos de computación de alto rendimiento[13]. Para la implementación del nivel de *grid* se utiliza el software

[11] Condor - High Throughput Computing (HTC). Página web. [en línea] <http://www.cs.wisc.edu/condor/> [Consultado en julio 10 de 2011]

[12] University of Wisconsin Madison. Página web. [en línea] <http://www.cs.wisc.edu/> [Consultado en julio 10 de 2011]

[13] High Throughput Computing (HTC). [en línea] <http://www.cs.wisc.edu/condor/htc.html> [Consultado

establecido por Open Science Grid[14] (OSG) quienes se encargan de mantener una infraestructura de *grid* significativa en Norte América destinada al ámbito académico e investigativo a través de sistemas distribuidos abiertos. Este software actualmente utiliza de manera interna al Globus Toolkit[15] como *middleware* o capa intermedia y su utilización fue convenida con el proyecto Grid Colombia[16] con quienes se trabaja muy de cerca durante la implementación de este segundo nivel, ya que en trabajo conjunto con ellos se garantiza la interacción de los nodos *grid* implementados con los demás nodos a nivel nacional e internacional.

En términos de hardware, el proyecto utiliza los equipos de cómputo provistos por el proyecto *Implantación de un sistema de Learning Management System (LMS) sobre una infraestructura* cluster *usando RENATA*[17] aprobado por Colciencias, del cual se desprende este proyecto como un objetivo específico definido por los patrocinadores del proyecto.

Para este proyecto de computación distribuida la capacidad y eficiencia de la

---

en julio 10 de 2011]

[14] Open Science Grid. Página web. [en línea] <http://www.opensciencegrid.org/> [Consultado en julio 10 de 2011]

[15] Globus Toolkit. Página web. [en línea] <http://www.globus.org/toolkit/> [Consultado en julio 10 de 2011]

[16] Grid Colombia. Página web. [en línea] <http://GRIDcolombia.org/> [Consultado en julio 10 de 2011]

[17] Proyecto Learning Management System (LMS) sobre una infraestructura *cluster* usando RENATA. [en línea] <http://www.renata.edu.co/index.php/salud/1304-learning-management-system-lms-sobre-una-infraestructura-cluster-usando-renata.html> [consultado en marzo 20 de 2011].

comunicación entre sus diferentes partes constitutivas es un requisito indispensable. Para satisfacer esta necesidad de conectividad se aprovecha la infraestructura de la Red Nacional Académica de Tecnología Avanzada (RENATA)[18] que provee una red de alta velocidad para el ámbito académico y de investigación, y a la cual pertenecen las dos universidades asociadas en este proyecto.

Como resultado final del proyecto no solo se realiza la implementación de los dos nodos *grid* dando soporte a los diferentes proyectos de computación de alto rendimiento en las instituciones, sino que se estructura el conocimiento obtenido durante el desarrollo del proyecto para que otras universidades y centros de investigación con iniciativas similares puedan realizar sus propias implementaciones aprovechando los logros y experiencia obtenidos. Así mismo se espera que sirva como base académica al interior de programas de pregrado y postgrado de la Universidad Autónoma de Manizales para la desarrollo de cursos y proyectos que aprovechen esta infraestructura de cómputo de alto desempeño y permitan complementarla a través de la innovación en el desarrollo de software y tecnologías útiles para resolver los problemas de la región.

[18] Red Nacional Académica de Tecnología Avanzada (RENATA). Página web. [en línea] <http://www.renata.edu.co/> [Consultado en febrero 14 de 2011].

1. REFERENTE CONTEXTUAL

1.1 PLANTEAMIENTO DEL PROBLEMA

Actualmente las universidades Autónoma de Manizales y del Rosario se encuentran desarrollando conjuntamente el proyecto de Colciencias *Implantación de un sistema de Learning Management System (LMS) sobre una infraestructura cluster usando RENATA*, a través del cual se realiza la implantación de un software de enseñanza sobre una plataforma de *cluster* propietaria llamada Maat-G. Para hacer esto se han desplegado máquinas físicas en cada una de las universidades sobre las cuales se soporta el sistema geográficamente distribuido.

Este proyecto requiere que se complemente con un valor agregado que articule su infraestructura de hardware con la de la *grid* nacional, permitiéndole a las universidades mencionadas utilizar la capacidad computacional (procesamiento y almacenamiento) de otras las universidades participantes y compartir con ellas la suya propia.

Para hacer esto el presente proyecto se relaciona con el proyecto Grid Colombia, quienes fueron los responsables de brindar las especificaciones básicas necesarias para permitir la integración de los nodos regionales con la

infraestructura nacional. Así mismo, se aprovecha a RENATA como infraestructura de red para interconectar los nodos implementados -entre ellos mismos y a nivel nacional- gracias a su alta velocidad y confiabilidad en la transmisión de datos.

1.2 ANTECEDENTES

El concepto de computación distribuida aparece casi tan pronto como el componente de comunicaciones se involucra en el ámbito computacional. A través de él se solventan las limitaciones de capacidad de cómputo impuestas por el aislamiento, aprovechando los recursos de los equipos en red, especialmente optimizando los desperdiciados (*idle*). Esto facilita a los científicos e investigadores ejecutar grandes tareas en términos de recursos computacionales (facilidad de cómputo y almacenamiento) que no podrían ser cubiertos por su capacidad local y realizarlos en una menor cantidad de tiempo.

La computación en racimo o *cluster* tiene sus orígenes a finales de la década de los 50 y principios de los 60. La publicación de la ley de Amdahl[19] en 1967 por Gene Amdahl demuestra matemáticamente las ventajas de la paralelización de las tareas creando el sustento formal para la computación multiprocesador y en

[19] Gene Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", AFIPS Conference Proceedings, (30), pp. 483-485, 1967.

cluster. Estas tecnologías en su momento emergentes, se vieron fortalecidas por la creación de las redes basadas en la conmutación de paquetes gracias a las cuales se dio inicio el proyecto ARPANET[20] el cual terminó convirtiéndose en Internet.

En los años siguientes, tecnologías como la invención del protocolo TCP/IP, la madurez del sistema operativo UNIX, la masificación de las conexiones de red de alta velocidad a nivel mundial y la paralelización, permitieron que la tecnología de clusterización pudiera ofrecer la creación de "supercomputadores" conformados por unidades de cómputo heterogéneas mas pequeñas funcionando al unísono bajo complejas reglas de sincronización.

Durante la última década del siglo 20 junto con la masificación de Internet, evoluciona el concepto de computación distribuida al permitir que se realice la agrupación de los recursos distribuidos sin que estos se encuentren sujetos a un único control centralizado, es decir, se llevaron los conceptos de integración y uso colectivo de recursos planteados por la clusterización a un nuevo nivel mas allá de los límites administrativos de las instituciones. Para esto se debieron desarrollar interfaces mas complejas, mecanismos de autorización y autenticación que permitieran a los usuarios el acceso a la información con alta granularidad, capas

[20] Advanced Research Projects Agency Network (ARPANET).

intermedias (*middleware*) que realizaran la abstracción sobre las diferentes plataformas y arquitecturas, sistemas de archivos distribuidos, gestores de trabajos distribuidos con mejores controles para la concurrencia y la recuperación antes fallos, entre otras muchas tecnologías. A esta nueva tecnología se le conoció como computación en malla o *grid*.

A nivel mundial existen varias iniciativas de computación en *grid* de las cuales se destacan dos de ellas a nivel científico. La European Grid Infraestructure (EGI)[21] que agrupa las iniciativas de los países europeos[22] y la Open Science Grid (OSG) de Norte América la cual fue creada inicialmente para soportar el análisis de los datos obtenidos por el Large Hadron Collider (LHC)[23]. Estos proyectos utilizan respectivamente gLite[24] y Globus Toolkit[25] como sus capas intermedias de abstracción o *middlewares*.

A nivel latinoamericano tanto la Grid Europea como la norteamericana han apoyado iniciativas en varios países, por ejemplo la Grid Europea tiene presencia en Colombia a través del proyecto GISELA[26] con la Universidad de los Andes[27] y

[21] EGI, European Grid Infraestructure. [en línea] <http://www.egi.eu/> [Consultado en julio 3 de 2011]
[22] EGI, Resource infrastructure providers. [en línea] <http://www.egi.eu/infrastructure/Resource-providers/index.html> [consultado en julio 3 de 2011].
[23] LHC, Large Hadron Collider. [en línea] <http://lhc.web.cern.ch/lhc/> [Consultado en julio 3 de 2011]
[24] GLITE, Lightweight Middleware for Grid Computing. [en línea] <http://glite.cern.ch/> [Consultado en julio 3 de 2011]
[25] Globus Toolkit. [en línea] <http://www.globus.org/toolkit/> [Consultado en junio 24 de 2011]
[26] GISELA, Grid Initiatives for e-Science virtual communities in Europe and Latin America. [en línea] <http://www.gisela-grid.eu/> [Consultado en julio 4 de 2011]
[27] Universidad de los Andes, Colombia. Página web. [en línea] <http://www.uniandes.edu.co/>

OSG cuenta con una vasta presencia en las universidades del Brasil. A pesar de esto en el año 2009, Colombia aún no había elegido una infraestructura para desarrollar su *grid* a nivel nacional en conjunto con las instituciones académicas y los centros de investigación. En este contexto surge el proyecto Grid Colombia que con el apoyo de Colciencias y RENATA elige el modelo de Open Science Grid como base para desarrollar la infraestructura *grid* de Colombia.

Actualmente a Grid Colombia pertenecen 25 instituciones[28] agrupadas en cinco regiones que cubren el territorio nacional, de las cuales 13 grupos de investigación[29] ya se encuentran planeando o desarrollando proyectos relacionados con la *grid* nacional.

Las entidades interesadas en desarrollar su infraestructura de computación de alto desempeño y de relacionarla a nivel nacional mediante la *grid* nacional, se deberán asociar a través de Grid Colombia y proveer sus propios recursos -hardware, software y humanos- para realizar la implementación de sus nodos.

Es aquí cuando surge la necesidad de desarrollar este proyecto que realiza la implementación tangible de los nodos *grid* para las universidades relacionadas.

[Consultado en julio 3 de 2011]

[28] GRIDCOLOMBIA, Instituciones conectadas. [en línea] <http://gridcolombia.org/index.php?option=com_content&view=article&id=60&Itemid=77> [Consultado en julio 20 de 2011]

[29] GRIDCOLOMBIA, Grupos de investigación. [en línea] <http://gridcolombia.org/index.php?option=com_content&view=article&id=62&Itemid=78> [Consultado en julio 20 de 2011]

Este se basa en la documentación que OSG ha desarrollado según sus propios intereses y los estándares que Grid Colombia ha determinado para garantizar la conectividad general entre los nodos, sin embargo actualmente debido a lo reciente de estas decisiones, no se cuenta con una especificación o documentación precisa y adaptada a las necesidades nacionales, es por esto que además de los objetivos relacionados con la definición, instalación, configuración, uso y administración de los componentes se considera al objetivo de la documentación como de vital importancia para el proyecto ya que se espera que este se integre con la documentación de Grid Colombia para beneficiar a las instituciones que pertenecen al proyecto a nivel nacional.

1.3 JUSTIFICACIÓN

La motivación inicial para la realización del proyecto surge de la necesidad de su implementación como objetivo específico del proyecto de Colciencias *Implantación de un sistema de Learning Management (LMS) sobre una infraestructura cluster utilizando RENATA* desarrollado por parte de las universidades Autónoma de Manizales y del Rosario. El proyecto mencionado realiza el despliegue de una infraestructura de hardware en cada una de las universidades sobre la cual implanta un software educativo en cluster mediante la

utilización de la plataforma Maat-Knowledge. El presente proyecto aprovecha su infraestructura base de hardware implementando dos clusters de alto rendimiento sobre los cuales se implementan dos nodos grid que permiten exponer esta capacidad de cómputo a otras universidades y centros de investigación a través de la grid nacional. Esto se realiza utilizando software libre y software liberado bajo licencias compatibles.

Gracias al nivel de *cluster* de alto rendimiento que provee el proyecto, las instituciones involucradas optimizan el uso de sus recursos computacionales (procesamiento y almacenamiento) permitiéndoles la posibilidad de desarrollar proyectos de investigación que requieran altas capacidades de cómputo que excedan las de máquinas individuales existentes en sus centros de cómputo.

El nivel de *grid* le permite a las universidades involucradas en el proyecto compartir estos recursos con sus homólogos a través de la *grid* nacional, es decir, los usuarios investigadores de estas universidades pueden aprovechar los recursos de alto desempeño de otras universidades además de los suyos propios y están en capacidad de ofrecer a investigadores de universidades e instituciones del país, utilizar los recursos computacionales que en el proyecto se ofrezcan de manera transparente[30]. De esta manera los investigadores estarán en la

[30] En este contexto *transparente* indica que los recursos o sistemas remotos se acceden o utilizan de la misma manera que se hace con los locales.

capacidad de iniciar proyectos cuyos requisitos computacionales sean mucho mayores a los instalados localmente en sus instituciones.

El desarrollo de proyectos que requieran del procesamiento de imágenes, edición de vídeo, manipulación de altos volúmenes de información, transferencia de datos a altas velocidades, videoconferencias, entornos colaborativos y procesamiento de alto desempeño entre otras características computacionalmente intensivas, como es el caso de los desarrollados en biotecnología, se verán ampliamente beneficiados por los resultados de este proyecto.

Al respecto, la Universidad Autónoma de Manizales ya se encuentra desarrollando tres nuevos proyectos con los cuales se espera explotar las potencialidades que ofrecerá la *grid* en las áreas de procesamiento de señales, procesamiento de imágenes médicas y la realización de cálculos matemáticos para la definición de procesos mecánicos.

Se espera que esta colaboración entre las universidades, centros de investigación y comunidades científicas que se encuentran conectadas a través de RENATA trascienda el uso de la infraestructura computacional compartida y fomente la generación de alianzas y el desarrollo de proyectos de investigación conjuntos que tengan un impacto positivo en la sociedad, fortaleciendo la competitividad y

mejorando la productividad de la región. De igual manera se espera que la documentación y experiencia obtenidas durante el desarrollo de este proyecto sirva para orientar a otras comunidades académicas en la implantación de sus propios *clusters* y nodos *grid* que pertenezcan a RENATA, fortaleciendo y complementando la iniciativa Grid de Colombia.

Una vez terminado el proyecto la Universidad Autónoma de Manizales tendrá implantada una infraestructura de cómputo de alto rendimiento que soportará la implementación de cursos y proyectos en los niveles de pregrado y postgrado en los que se optimicen y aprovechen las ventajas que esta infraestructura provee (punto de vista del usuario), se enseñe a administrar este tipo de infraestructura distribuida (punto de vista del administrador) y se implementen nuevas e innovadoras soluciones aplicadas de software que se basen en esta infraestructura como recurso de bajo nivel (punto de vista del desarrollador).

1.4 DELIMITACIÓN DEL ÁREA PROBLEMÁTICA

El presente proyecto se enfoca en proveer una infraestructura fundamental de *grid* para los nodos de las universidades Autónoma de Manizales y del Rosario en Bogotá a partir de los recursos de hardware provistos por el proyecto

Implantación de un sistema de Learning Management (LMS) sobre una infraestructura cluster *utilizando RENATA* con el cual se relaciona este proyecto.

Para hacer esto se tienen en cuenta dos niveles de asociación de la capacidad de cómputo: un primer nivel constituye el cluster local y un segundo nivel constituye la *grid* como tal. La implementación de estos niveles se realiza teniendo en cuenta las especificaciones y lineamientos del proyecto Grid Colombia y Open Science Grid (OSG) para garantizar la interoperabilidad de los nodos locales con el *grid* nacional colombiano.

Se utiliza la infraestructura de RENATA como red física para la comunicación entre los dos nodos *grid* y la *grid* de alcance nacional. Las dos Universidades involucradas en este proyecto pertenecen a RENATA a través de conexión con las redes regionales que les corresponden. Es responsabilidad de cada una de las universidades proveer al proyecto el acceso a este recurso.

Para la gestión del conocimiento obtenido durante el desarrollo del proyecto se utilizan herramientas web 2.0 como *blogs* o *wikis* que permiten plasmar la información de manera útil y flexible para cumplir con los objetivos del proyecto, facilitando además la utilización de este conocimiento como base para realizar capacitaciones y replicar sus resultados en otras instituciones interesadas en este

tipo de iniciativas.

La integración de aplicaciones al nodo *grid* así como el desarrollo de las mismas mediante el uso de *middlewares* y paradigmas de paralelización que exceden los alcances de este proyecto y constituyen la siguiente etapa propuesta a explotarse sobre la infraestructura obtenida una vez finalizado este proyecto.

1.5 OBJETIVOS

1.5.1 Objetivo General

Implementar dos nodos *grid* basados en clusters integrados a Grid Colombia a través de la Red Nacional Académica de Tecnología Avanzada (RENATA) utilizando software libre.

1.5.2 Objetivo Específico

1. Identificar la infraestructura tecnológica disponible en las universidades para conformar los *nodos grid*.
2. Diseñar la arquitectura de los *nodos grid* con base en los lineamientos

definidos por Grid Colombia.

3. Implementar dos clusters basados en la infraestructura y tecnologías establecidas.
4. Determinar los componentes de los *nodos grid* que deben ser implementados localmente y cuáles deben ser suministrados por el proveedor de la *grid*.
5. Implementar dos *nodos grid* basados en la infraestructura de clusters con las tecnologías establecidas.
6. Documentar los procesos de implementación de los *clusters* y *nodos grid* de tal manera que la comunidad académica de Renata pueda reproducirlos.

1.6 RESULTADOS ESPERADOS

Con el desarrollo del presente proyecto se obtendrán los siguientes resultados.

1. Inventario de la infraestructura tecnológica disponible en cada universidad para la implementación de los nodos *grid*.
2. Modelo de arquitectura de los nodos *grid* para cada uno de los niveles: *cluster* y *grid*, diferenciando cuales componentes será provistos por las instituciones y cuáles serán provistos a nivel general por Grid Colombia.

3. Documentación relacionada con la instalación, configuración, administración y uso del *cluster* en la institución.
4. Documentación relacionada con la instalación, configuración, administración y uso del nodo *grid* en la institución.

La documentación del proyecto se encontrará disponible en Internet para el público en general a través de herramientas web 2.0.

2. ESTRATEGIA METODOLÓGICA

2.1 METODOLOGÍA

No se identificó la existencia de una metodología reconocida o estándar para la implementación de este tipo de sistemas distribuidos. Después de analizar el problema a solucionarse se propone dividir el desarrollo del proyecto en tres niveles independientes.

1. **Nivel de *cluster*** (basado en la infraestructura física de los servidores y la red).
2. **Nivel de *grid*** (basado en el nivel de *cluster*).
3. **Nivel de documentación** (perpendicular a los dos niveles anteriores).

Los dos primeros niveles se subdividen en una **etapa conceptual** en la que se establecen, diseñan y documentan los contenidos, arquitecturas y tecnologías requeridas para el desarrollo de la siguiente etapa que corresponde con la **etapa de implementación**, durante la cual se realiza la instalación, desarrollo, configuración y documentación del software que sea necesario para cumplir con los requisitos de cada uno de los niveles.

La **etapa de implementación** del proyecto se desarrolla a su vez en tres fases diferentes. La primera de estará destinada a realizar las **pruebas, experimentos y ajustes de configuración** que se consideren necesarios para cumplir con los requerimientos de implementación planteados durante la etapa conceptual. Esta primera fase se realizará sobre un conjunto de máquinas virtuales preparadas para este fin.

La segunda fase consiste en **materializar la implementación** del nivel en la infraestructura de producción gracias a las conclusiones, experiencias y valores de configuración obtenidos en durante la fase inmediatamente anterior.

La fase final de la etapa de implementación de cada nivel corresponde con la ejecución de las **pruebas funcionales y de conectividad** que garanticen el correcto funcionamiento de la infraestructura física y de los componentes de software distribuidos recién instalados.

Ya que la implementación de los diferentes niveles del proyecto debe ser compatible con las especificaciones del proyecto Grid Colombia las cuales se basan principalmente sobre la especificación de Open Science Grid (OSG), se toman a estas dos como orígenes de información base para el desarrollo de la

implementación del proyecto, la cual se revisará, experimentará y adaptará a las necesidades específicas de los nodos regionales de las universidades involucradas.

Para formalizar la gestión del conocimiento generado durante el proyecto se utilizan herramientas web 2.0 como un *blog* para el registro y publicación de las actividades y eventos realizados, y un *wiki* para la documentación formal de las conclusiones y logros encontrados durante cada una de los niveles del proyecto.

2.2 PRUEBAS

Se diseñaron una serie de verificaciones para comprobar el funcionamiento de los principales elementos de los nodos de acuerdo con los niveles de implementación del *cluster* y *grid*. Estas pruebas se realizaron de manera independiente en los dos nodos (Universidad Autónoma de Manizales y Universidad del Rosario) incluidos en el proyecto.

Para consultar por extensión las pruebas realizadas y las correspondientes respuestas obtenidas referirse al Anexo I: *Pruebas funcionales de los Nodos*. A continuación se relacionan las verificaciones realizadas para cada uno de los

niveles del nodo.

2.2.1 Nivel de *cluster*

1. Verificar conectividad con el nodo principal del *cluster*
2. Verificar conectividad SSH con el nodo principal del *cluster*
3. Verificar conectividad con los nodos trabajadores del *cluster*
4. Verificar conectividad SSH con los nodos trabajadores del *cluster*
5. Verificar el estado del sistema de archivos compartido (NFS) en el *cluster*
6. Verificar el pool de nodos trabajadores del *cluster* Condor
7. Verificar el soporte para Java en los nodos trabajadores del *Cluster*
8. Verificar la cola de trabajos del *cluster* Condor
9. Probar el envío de trabajos al universo Vanilla del c*luster*
10. Probar el envío de trabajos al universo Standard del c*luster*
11. Probar el envío de trabajos al universo Java del c*luster*

2.2.2 Nivel de *grid*

1. Verificar conectividad entre el Cliente Grid y el Compute Element

2. Cargar y verificar las variables de ambiente del Cliente Grid
3. Crear y verificar el intermediario del certificado
4. Verificar la autenticación con el nodo *grid*
5. Probar el envío de trabajos al nodo *grid* con las herramientas de Globus
    a) Utilizando el manejador de trabajos fork
    b) Utilizando el manejador de trabajos de Condor
6. Probar el envío de trabajos al nodo *grid* utilizando Condor-G
7. Probar la transferencia de archivos entre el Cliente Grid y el nodo *grid*
    a) Transferir el archivo del Cliente Grid al nodo *grid*
    b) Transferir el archivo de regreso, desde el nodo *grid* hacia el Cliente Grid
    c) Comparar el archivo transmitido inicialmente con el finalmente recibido
8. Destruir y verificar el intermediario del certificado

2.3 PRESUPUESTO

Rubro	Descripción	Justificación	Valor
Hardware	Servidores nodo UR[31]	Equipos de cómputo requeridos para implementar el *cluster*/nodo *grid* de la Universidad del Rosario.	$20.000.000
	Servidores nodo UAM[32]	Equipos de cómputo requeridos para implementar el *cluster*/nodo *grid* de la Universidad Autónoma de Manizales (UAM).	$20.000.000
	Switch para nodo UAM	Equipos de red necesarios para realizar la conexión del nodo UAM.	$2.000.000

[31] Universidad del Rosario.
[32] Universidad Autónoma de Manizales.

	Computador portátil	Equipo móvil para el desarrollo de la implementación, pruebas, documentación y capacitaciones del proyecto.	$1.700.000
		TOTAL DEL RUBRO	**$43.700.000**
Software	Ninguno	No se requieren de licencias comerciales.	$0
		TOTAL DEL RUBRO	**$0**
Servicios técnicos	Hosting del sitio web del proyecto por dos años	Se requiere para el almacenamiento remoto y publicación en internet de los contenidos 2.0 del proyecto.	$700.000
		TOTAL DEL RUBRO	**$700.000**
Materiales	Papelería, fotocopias, empastes, tinta, CD/DVD, USB	Elementos de oficina requeridos durante el desarrollo del proyecto.	$600.000
		TOTAL DEL RUBRO	**$600.000**
Servicios personales	Ing. Jorge Iván Meza	Responsable del proyecto	$12.600.000
	Ing. Ana Lorena Uribe	Asesora técnica del proyecto	$18.000.000
		TOTAL DEL RUBRO	**$30.600.000**
Viajes, viáticos y asistencias	Asistencia curso 1 dictado por OSG	Asistencia al curso de computación en *grid* (1) en Bucaramanga 03/2010 – 5 días	$1.920.000
	Asistencia curso 2 dictado por GridColombia	Asistencia al curso de computación en *grid* (2) en Manizales 07/2010 – 3 días	$530.000
	Asistencia reunión GridColombia	Asistencia a reunión con Grid Colombia para establecer los requerimientos necesarios para una implementación compatible del nodo *grid* en Bogotá 01/2011 – 2 días	$610.000
	Asistencia clausura proyecto GridColombia	Asistencia a la ceremonia de clausura del proyecto Grid Colombia en Bogotá 03/2011 – 2 días	$610.000
	Asistencia OSG School 2011 dictado por OSG	Asistencia al *OSG School* previa al 6CCC en Manizales 05/2011 – 2 días	$330.000
	Realización del curso Implementación de Clusters	Realización del curso Implementación de *clusters* para las universidades de la región dictado en Manizales 07/2010 – 3 días	$415.000
	Realización del taller de computación en Grid en el 6CCC	Realización del taller de Computación en *grid* durante el sexto congreso colombiano de computación 05/2011 – 2 días	$490.000
		TOTAL DEL RUBRO	**$4.905.000**
		TOTAL GENERAL	**$80.505.000**

3. DESARROLLO

3.1 REFERENTE TEÓRICO

A continuación se ubicarán los conceptos teóricos relacionados con las temáticas abordadas por el proyecto. Estas serán desarrolladas de manera progresiva abarcado los conceptos generales del software libre, los diferentes tipos de sistemas distribuidos involucrados, las tecnologías utilizadas y las entidades relacionadas con la implementación del proyecto.

3.1.1 Software libre.

La primera de estas temáticas corresponde con el software libre[33] ya que la totalidad del software utilizado durante el diseño y la implementación del proyecto pertenece a este tipo de licenciamiento y esto hace que sea importante conocer sus alcances para determinar que tipos de restricciones o limitaciones legales pueden existir con el uso, desarrollo o modificación de este software.

[33] GNU. Definición del software libre. [en línea] <http://www.gnu.org/philosophy/free-sw.es.html> [Consultado el 10 de abril de 2010]

El *software libre* es un movimiento político y social iniciado por Richard Stallman[34] en 1983, a través del cual se intenta defender los derechos que el movimiento considera, los usuarios deberían tener sobre el software en general. Estos derechos deberían incluir la posibilidad de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software según sus propias necesidades.

Para esto plantean que es necesario contar con cuatro libertades esenciales.

1. La libertad de ejecutar el programa, para cualquier propósito.
2. La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera.
3. La libertad de redistribuir copias para que pueda ayudar al prójimo.
4. La libertad de distribuir copias de sus versiones modificadas a terceros.

De estas libertades se infiere que para la Fundación del Software Libre[35] es necesario que se tenga acceso al código fuente del software para considerarlo libre bajo sus parámetros, sin embargo hace notar que esta condición por si sola no es suficiente y en esto difieren significativamente del movimiento *open source*[36]

---

[34] STALLMAN, Richard. Página personal. [en línea] <http://stallman.org/> [Consultado el 10 de abril de 2010]

[35] FSF - Free Software Foundation. Página web. [en línea] <http://www.fsf.org/> [Consultado el 10 de abril de 2010]

[36] GNU. Why open source misses the point of Free Software. [en línea] <http://www.gnu.org/philosophy/open-source-misses-the-point.html> [Consultado el 10 de abril de 2010]

el cual permite que se establezcan restricciones sobre lo que es permitido hacer con el código fuente recibido.

Para garantizar la libertad del software publicado por los desarrolladores adeptos al movimiento, se proponen una serie de licencias[37] *copyleft*[38].

- Licencia Pública General de GNU[39].
- Licencia Pública General Reducida de GNU[40].
- Licencia Pública General de Affero de GNU[41].
- Licencia de Documentación Pública de GNU[42].

De estas licencias las mas utilizadas para el software libre son la GPL y la LGPL mientras que la FDL se utiliza para cubrir los materiales de documentación como manuales, tutoriales y presentaciones libres.

Entre las dos primeras licencias mencionadas existe una diferencia fundamental

---

37 GNU. Licencias. [en linea] <http://www.gnu.org/licenses/licenses.es.html> [Consultado el 10 de abril de 2010]

38 Copyleft, tipo de licenciamiento que protege al creador como su autor mas no restringe los derechos del usuario que accede a la obra.
GNU, Copyleft. [en línea] <http://www.gnu.org/copyleft/copyleft.es.html> [Consultado el 10 de abril de 2010]

39 GNU. GPL. [en línea] <http://www.gnu.org/licenses/gpl.html> [Consultado el 10 de abril de 2010]

40 GNU. LGPL. [en línea] <http://www.gnu.org/copyleft/lgpl.html> [Consultado el 10 de abril de 2010]

41 GNU. AGPL. [en línea] <http://www.gnu.org/licenses/agpl.html> [Consultado el 10 de abril de 2010]

42 GNU. FDL. [en línea] <http://www.gnu.org/copyleft/fdl.es.html> [Consultado el 10 de abril de 2010]

que es muy importante tenerla en cuenta durante la utilización y desarrollo de software libre. La primera de ellas obliga inexorablemente a que los productos derivados que se basen en software GPL deban hacer público su código fuente bajo la misma licencia GPL, es decir, garantiza la perpetuidad de las cuatro libertades esenciales del software para con los usuarios mencionadas anteriormente. La segunda de estas licencias, la LGPL, establece un licenciamiento menos restrictivo al permitir que el software que aproveche librerías LGPL mediante el uso de enlace dinámico o compartido (no derivado o enlace estático) pueda ser licenciado posteriormente mediante cualquier otro tipo de licenciamiento.

Existen otras licencias de *software libre* llamadas *permisivas* ya que estas no son tan estrictas con el trabajo derivado basado en ellas (*copyleft*) como las licencias GNU permitiendo el uso de su código fuente en software no libre, siendo entonces no necesariamente compatibles con la licencia GPL mencionada anteriormente, y que son de amplio uso en la actualidad. Entre ellas[43] cabe notar las siguientes.

1. La **licencia BSD** (*Berkeley Software Distribution*) desarrollada por la Universidad de California[44]. Fue utilizada inicialmente para permitirle a la

[43] Listado alfabético de licencias aprobadas por la Open Source Initiative (OSI). [en línea] <http://www.opensource.org/licenses/alphabetical> [Consultado el 4 de mayo de 2011].

[44] University of California. Página web. [en línea] <http://www.universityofcalifornia.edu/> [Consultado el 3 de mayo de 2011]

Universidad de California realizar aportes al código fuente de Unix propiedad de los Laboratorios Bell de AT&T durante los años setenta y ochenta. Posteriormente ha sido adoptada y adaptada para por muchos desarrolladores de software, generando incluso problemas por modificaciones inapropiadas[45]. Su contenido puede consultarse en el sitio web de Open Source Initiative[46].

2. La **licencia MIT** (*Massachusetts Institute of Technology*) desarrollada por el instituto con el mismo nombre[47]. Desarrollada inicialmente en la década de los ochenta para la publicación del sistema de ventanas X también conocido como X11. Es muy similar a la licencia BSD en cuanto a sus efectos. Su contenido puede consultarse en el sitio web de Open Source Initiative[48].

3. La **licencia Apache** desarrollada por la Apache Software Foundation (ASF)[49] para la publicación de todos sus proyectos entre los que se incluye

[45] El problema de la licencia BSD. [en línea] <http://www.gnu.org/philosophy/bsd.es.html> [Consultado el 3 de mayo de 2011].
[46] Contenido de la licencia BSD 2. [en línea] <http://www.opensource.org/licenses/bsd-license.php> [Consultado el 3 de mayo de 2011].
[47] Massachusetts Institute of Technology. [en línea] <http://web.mit.edu/> [Consultado el 4 de mayo de 2011].
[48] Contenido de la licencia MIT. [en línea] <http://www.opensource.org/licenses/mit-license.php> [Consultado el 4 de mayo de 2011].
[49] Apache Software Foundation. [en línea] <http://www.apache.org/foundation/> [Consultado el 4 de mayo de 2011].

el servidor de páginas web[50] mas utilizado a nivel mundial. Su contenido puede consultarse en el sitio web de Open Source Initiative[51].

4. La **licencia pública de Mozilla** (MPL) desarrollada originalmente por Netscape Communications Corportation y ahora a cargo de la Fundación Mozilla[52]. Se utiliza principalmente para la publicación de los productos de la Fundación, sin embargo ha sido ampliamente utilizada por otros desarrolladores de manera directa o como base para sus propias licencias. Su contenido puede consultarse en el sitio web de Open Source Initiative[53].

5. Las licencias de **Creative Commons**[54] están basadas en la licencia GPL sin embargo no están pensadas para licenciar software sino apoyar la distribución y el uso de contenidos libres. El propietario de la obra puede decidir las condiciones[55] en las que los usuarios podrán acceder y utilizar sus contenidos según las atribuciones, los fines y la posibilidad de crear derivados de la obra original.

---

[50] The Apache HTTP Server Project. [en línea] <http://httpd.apache.org/> [Consultado el 4 de mayo de 2011].

[51] Contenido de la licencia Apache 2.0. [en línea] <http://www.opensource.org/licenses/Apache-2.0> [Consultado el 4 de mayo de 2011].

[52] Fundación Mozilla. [en línea] <https://www.mozilla.org/> [Consultado el 4 de mayo de 2011].

[53] Contenido de la licencia pública de Mozilla 1.1. [en línea] <http://www.opensource.org/licenses/MPL-1.1> [consultado el 4 de mayo de 2011].

[54] Creative Commons es una organización no gubernamental sin ánimo de lucro que desarrolla planes para reducir las barreras legales de la creatividad por medio de nuevas legislaciones y tecnologías. [en línea] <http://creativecommons.org/> [consultado el 4 de mayo de 2011].

[55] Tipos de licencias Creative Commons en Colombia. [en línea]. <http://co.creativecommons.org/tipos-de-licencias/> [consultado el 4 de mayo de 2011].

Uno de los muchos paquetes de software libre que se encuentran involucrados en el proyecto y el cual es pieza clave para implementar su desarrollo es el sistema operativo GNU/Linux el cual se describe a continuación.

3.1.2 El sistema operativo GNU/Linux.

El sistema operativo GNU/Linux es uno de los ejemplos más claros de software libre desarrollado y mantenido por la comunidad. Su desarrollo fue iniciado por Richard Stallman en 1983 a través de la Free Software Foundation[56] con el objetivo de desarrollar un sistema operativo compatible con Unix[57] enteramente compuesto por software libre mediante el licenciamiento GPL de GNU[58].

Posteriormente se adoptó el *kernel*[59] Linux[60] desarrollado por Linus Torvalds[61] a partir de 1991.

El GNU/Linux, algunas veces llamado Linux de manera incorrecta, ha tenido gran

[56] Fundación del Software Libre. Página web. [en línea] <http://www.fsf.org/> [Consultado el 10 de abril de 2010]
[57] UNIX, Sistema operativo portable, multitarea y multiusuario desarrollado inicialmente por los laboratorios Bell de AT&T. [en línea] <http://es.wikipedia.org/wiki/Unix> [Consultado el 10 de abril de 2010]
[58] GNU Operative System. Página web. [en línea] <http://www.gnu.org/> [Consultado el 10 de junio de 2011]
[59] Núcleo del sistema operativo encargar de gestionar el acceso de bajo nivel a los recursos del hardware.
[60] The Linux Kernel Archives. [en línea] <http://kernel.org/> [Consultado el 10 de abril de 2010]
[61] Ingeniero de software finlandés nacido en diciembre de 1969 y desarrollador del kernel Linux basado en Minix creado a su vez por Andrew Tanenbaum.

acogida a nivel mundial por la comunidad técnica, especialmente en el uso de servidores y entre ellos los web (basados en el protocolo HTTP[62]), segmento del mercado que lidera.

Gracias a la diversidad de herramientas y aplicativos que es posible obtener, tanto de manera libre como comercial para este sistema operativo, se ha convertido en una opción ideal para usuarios con conocimientos técnicos avanzados, desarrolladores de software, administradores de sistemas y la academia, teniendo un extenso uso en servidores de múltiples propósitos. En los últimos años ha venido ganando terreno incluso, en los escritorios de los usuarios menos experimentados, equipos móviles, sistemas empotrados y videoconsolas.

Debido a esto ha surgido una gran cantidad de distribuciones basadas en este sistema operativo que son mantenidas por comunidades independientes y que hacen énfasis en características particulares según el enfoque y finalidad particular que se les ha imprimido.

Las principales distribuciones de GNU/Linux que son mantenidas actualmente son las siguientes.

[62] Protocolo de Transferencia de Hipertextos (*Hypertext Transfer Protocol*).

- Debian[63].
    - Ubuntu[64] con sus variaciones Kubuntu/Edubuntu/XUbuntu y Mint.
    - Knoppix[65].
- RedHat[66].
    - Fedora[67].
    - Scientific Linux[68].
    - CentOS[69].
- Slackware[70].
- SUSE[71].
    - OpenSUSE[72].

Para la implementación de este proyecto se ha elegido **Scientific Linux 5.4** como sistema operativo para los servidores que conformarán los servidores del *cluster* y por ende del nodo *grid* a implementarse, ya que este es uno de los sistemas

---

63 Debian, The universal operative system. Página web. [en línea] <http://www.debian.org/> [Consultado el 10 de abril de 2010]

64 Ubuntu, Linux for human beings. Página web. [en línea] <http://www.ubuntu.com/> [Consultado el 10 de abril de 2010]

65 Knoppix. Página web. [en línea] <http://www.knoppix.net/> [Consultado el 10 de abril de 2010]

66 RedHat. Página web. [en línea] <http://www.redhat.com/> [Consultado el 10 de abril de 2010]

67 Fedora. Página web. [en línea] <http://fedoraproject.org/> [Consultado el 10 de abril de 2010]

68 Scientific Linux. Página web. [en línea] <https://www.scientificlinux.org/> [Consultado el 10 de abril de 2010]

69 CentOS. Página web. [en línea] <http://centos.org/> [Consultado el 10 de abril de 2010]

70 Slackware. Página web. [en línea] <http://www.slackware.com/> [Consultado el 10 de abril de 2010]

71 SUSE. Página web. [en línea] <http://www.suse.com/> [Consultado el 10 de abril de 2010]

72 OpenSUSE. Página web. [en línea] <http://www.opensuse.org/> [Consultado el 10 de abril de 2010]

operativos sugeridos tanto por el Globus Toolkit[73] para Open Science Grid (OSG)[74] como por gLite lo cual permitirá utilizar esta misma infraestructura para realizar una interacción con la *grid* europea si así se desea en un futuro próximo.

A continuación se describen brevemente los servicios y funcionalidades provistas directamente por el sistema operativo utilizadas durante el desarrollo del proyecto.

1. El **Secure Shell** (SSH) se basa en la librería de cifrado OpenSSL y permite la implementación de comunicaciones seguras (cifradas) entre servidores, tanto para acceder a la línea de comando remota (reemplazar a `telnet`) y transferir archivos (con comandos como `sftp` y `scp`) como para realizar túneles y enrutar a través de ellos a servicios intrínsecamente inseguros. Una facilidad interesante que permite este servicio es el establecimiento de relaciones de confianza entre equipos mediante el uso de llaves privadas y públicas. Este servicio se utiliza para la administración remota de los equipos y la transferencia de archivos durante las etapas de instalación y configuración del proyecto.

2. El **Network File System** (NFS) es un servicio del nivel de aplicación que

[73] Capa de software intermedia (o *middleware*) para la construcción de GRIDS desarrollada por la Globus Alliance.

[74] Open Science Grid. Cluster Getting started. Machine, operative system characteristics. [en línea] https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/ClusterGettingStarted#Machine_operating_system_charact [Consultado el 10 de abril de 2010]

permite crear un sistema de archivos distribuido a través de una red de ámbito local, permitiéndoles a distintos servidores acceder a un mismo sistema de archivos a través de las mismas interfaces de alto nivel que las utilizadas para acceder al sistema de archivos local. Este servicio se utiliza para centralizar en un único servidor los archivos comunes a todos los demás servidores agilizando el proceso de actualización del software.

3. El **HyperText Transfer Protocol** (HTTP) constituye el servicio de páginas web. Para su implementación se utiliza el servidor Apache. Este servicio se utiliza para la publicación de la documentación generada por el proyecto y para dar soporte a los servicios web que utiliza el nivel de *grid*.

4. El **Network Time Protocol** (NTP) es un protocolo de Internet que permite sincronizar los relojes de múltiples servidores permitiendo un control exacto del tiempo para las transacciones distribuidas. Este servicio es necesario para la implementación del nivel de *grid*.

5. Las facilidades de **Firewall** (basada en *iptables*[75]) y **TCP Wrappers** permiten establecer de manera flexible restricciones de seguridad en cuanto al acceso al servidor y a los servicios que se ejecutan en él según

[75] Netfilter, *framework* para el filtrado de paquetes para los *kernels* 2,4,x y 2,6,x. [en línea] <http://www.netfilter.org/> [Consultado el 20 de julio de 2011]

los requerimientos específicos del sistema.

Una vez establecido el nivel del sistema operativo se desarrollan sobre él los niveles de *cluster* y *grid* propios del proyecto. Estos desde un punto de vista general constituyen un sistema distribuido para el soporte de la computación en paralelo. A continuación se revisan estos conceptos.

3.1.3 Sistemas distribuidos.

Un sistema distribuido se basa en un conjunto de equipos de cómputo (conformados por hardware y software) que se encuentran separados físicamente y se encuentran interconectados entre sí por medio de una red de datos. El acceso a este sistema por parte del usuario se realiza de manera transparente, es decir, el usuario no debe diferenciar si los recursos que utiliza se encuentran disponibles de manera local o remota.

Estas son las características principales de los sistemas distribuidos, claves en el desarrollo del proyecto.

1. **Fiabilidad, eficiencia y disponibilidad**. El sistema debe contar con

mecanismos para garantizar la prestación exitosa de sus servicios con un desempeño aceptable según sus características y la finalidad de sus servicios.

2. **Flexibilidad y capacidad de crecimiento (escalabilidad)**. Hace referencia a la capacidad que tiene el sistema para mejorar su infraestructura y a la vez adaptarse a ella para seguir prestando un servicio eficiente. El crecimiento puede darse básicamente por generarse mejoras en los elementos computacionales existentes o por la adición de nuevos elementos a la configuración.

3. **Seguridad**. Debido a su naturaleza descentralizada, la capacidad de garantizar que los actores que interactúan con él son realmente quienes dicen ser, que los mensajes enviados y recibidos son transmitidos de manera confiable y que los recursos son accedidos y utilizados de la manera correcta es de principal importancia.

4. **Transparencia para el usuario**. El nivel de abstracción que debe proveen los sistemas distribuidos permite ocultar a los usuarios y desarrolladores la separación entre sus componentes, haciendo que este se perciba como un único bloque. Se identifican las siguientes

formas básicas de transparencia en este tipo de sistemas.

- **Transparencia de acceso**. El acceso a los objetos de información por parte del usuario se realiza de manera independiente si estos son locales o remotos.

- **Transparencia de localización**. Permite el acceso a los objetos de información sin necesidad de conocer su ubicación física.

- **Transparencia de concurrencia**. Varios procesos que operen de manera concurrente pueden acceder a los objetos de información que se encuentren compartidos sin que se generen conflictos o interferencias.

- **Transparencia de replicación**. Permite el despliegue de copias o réplicas de múltiples instancias de los objetos de información distribuidos a lo largo del sistema para incrementar el desempeño de la aplicación sin que el usuario deba gestionarlas o conocer su información específica.

- **Transparencia de fallos**. El sistema manejará adecuadamente

los problemas presentados (hardware y software) de manera que los procesos ejecutados pueda completar sus tareas e informar adecuadamente de lo sucedido.

- **Transparencia de migración**. Permite el movimiento de los objetos de información entre los componentes del sistema sin que esto afecte su desempeño o requiera un conocimiento adicional por parte del usuario.

- **Transparencia de prestaciones**. La configuración del sistema se podrá adaptar para manipular su desempeño según las variaciones de la carga.

- **Transparencia de escalado**. Permite expandir el sistema sin que ello implique un cambio en la estructura física del sistema o en sus algoritmos.

5. **Heterogeneidad en sus componentes de hardware y software**. Los componentes del sistema no corresponden necesariamente a las mismas tecnologías o arquitecturas.

6. **Basado en las redes de datos**. Las redes de datos, especialmente de alta velocidad, interconectan los componentes distribuidos geográficamente del sistema.

El proyecto se fundamenta en la implementación de dos tipos de estos sistemas distribuidos, los *clusters* que permitirán la ejecución de tareas que requieran de alto desempeño y capacidad computacional, y las *grids* para conectar a los primeros y permitir la interacción con otras *grids* para proveer y consumir sus recursos de cómputo.

3.1.4 *Clusters*.

Un *cluster* es un conjunto de equipos de cómputo (no necesariamente con hardware y software homogéneos) unidos a través de una red de datos de alta velocidad que interactúan para el desarrollo de una actividad específica en común y que son vistos como una única máquina de grandes prestaciones por el usuario final permitiéndole desarrollar tareas que devenguen una gran capacidad de recursos.

En términos generales los *clusters* se clasifican de la siguiente manera según la

finalidad computacional que potencian.

1. **Alto rendimiento** (HPCC – *High performance computing clusters*): se especializan en la ejecución de tareas que requieren grandes cantidades de recursos computacionales como memoria, procesamiento o almacenamiento.

2. **Alta disponibilidad** (HA – *High availability*): se especializan en proveer la mayor disponibilidad y confiabilidad del servicio posible. Su funcionamiento se basa en la detección y tolerancia a fallos tanto desde el software como del hardware.

3. **Alto volúmen** (HT – *High throughput*): se especializan en la ejecución de tareas a gran velocidad, es decir, procesar exitosamente la mayor cantidad de tareas en el menor tiempo posible.

Un sistema de *cluster* requiere de los siguientes componentes básicos.

1. **Nodos**. Son equipos con capacidad de cómputo que pueden o no tener el mismo tipo de hardware. Se especializan según las actividades que realizan al interior del *cluster*: el ***nodo principal*** interactúa directamente

con el usuario para recibir sus requerimientos y enviar las respuestas del *cluster*, los ***nodos de trabajo*** se distribuyen el procesamiento de las tareas del *cluster* y los ***nodos de almacenamiento*** se encargan de la persistencia de la información, tanto de los datos de entrada como de salida de la ejecución de las tareas. La especialización de los nodos depende específicamente del tipo de *cluster* y del software distribuido que se utilice.

2. **Red de datos de alta velocidad**. Interconecta tanto a los nodos del *cluster* como a los clientes con el *cluster*. Comúnmente la conexión entre estos componentes se separa físicamente.

3. **Sistemas operativos**. Administran los recursos de los nodos y permiten la instalación del software que conforma el *cluster*.

4. ***Middleware***. Esta capa intermedia de software permite la implementación del *cluster* desde un punto de vista lógico. Interactúa entre el sistema operativo (estándar) y las aplicaciones distribuidas. Este componente es específico para cada uno de los tipos de *cluster*.

5. **Aplicaciones distribuidas**. Software de alto nivel que hace uso del cluster gracias a la abstracción provista por el *middleware*.

Existen múltiples paquetes de software para la implementación de *cluster*, entre los mas conocidos se encuentran los siguientes.

1. Mosix[76]. Su vertiente GPL -OpenMosix[77-] fue descontinuada el 1 de marzo de 2008 y fue retomada por el proyecto LinuxMPI[78].
2. OSCAR[79] para *clusters* tipo Beowulf[80].
3. Condor[81].
4. Solaris Cluster[82].
5. Rocks Clusters[83].
6. BOINC[84].
7. Veritas Cluster Server[85] (comercial).

---

[76] Mosix – cluster and multi-cluster management. Sitio web. [en línea] <http://www.mosix.org/> [Consultado el 10 de abril de 2010]

[77] OpenMosix. Sitio web. [En línea] <http://openmosix.sourceforge.net/> [Consultado el 10 de abril de 2010]

[78] LinuxMPI. Sitio web. [En línea] <http://linuxpmi.org/> [Consultado el 18 de julio de 2010]

[79] OSCAR – Open Source Cluster Application Resources. Sitio web. [En línea] <http://oscar.openclustergroup.org/> [Consultado el 18 de julio de 2010]

[80] Beowulf. Sitio web. [En línea] <http://www.beowulf.org/> [Consultado el 18 de julio de 2010]

[81] Condor – High throughput computing. Sitio web. [en línea] <http://www.cs.wisc.edu/condor/> [Consultado el 10 de abril de 2010]

[82] Solaris Cluster. Sitio web. [en línea] <http://www.sun.com/software/solaris/cluster/> [Consultado el 10 de abril de 2010]

[83] Rocks Clusters. Sitio web. [en línea] <http://www.rocksclusters.org/> [Consultado el 10 de abril de 2010]

[84] BOINC – Berkeley Open Infraestructure for Network Computing. Sitio web. [en línea] <http://boinc.berkeley.edu/> [Consultado el 10 de abril de 2010]

[85] Symantec. Veritas Cluster Server. [en línea] <http://www.symantec.com/business/cluster-server> [Consultado el 10 de abril de 2010]

Los *clusters* implementados como parte del desarrollo del proyecto pertenecen a la tipología de alto rendimiento ya que con ellos se pretende optimizar el uso de los recursos computacionales (procesamiento y almacenamiento) de las universidades permitiéndole a los usuarios la ejecución de trabajos que requieran cantidades significativas de recursos computacionales. Los *clusters* se implementarán utilizando Condor como sistema gestor de cargas de trabajo especializado para tareas de cálculo intensivo.

3.1.5 Condor.

El proyecto Condor es desarrollado por la Universidad de Wisconsin[86] en Madison (Estados Unidos) y su objetivo principal es desarrollar un manejador de cargas de trabajo que provea soporte para la computación oportunista de alto desempeño.

Esto lo hace a través de una arquitectura de tres componentes principales.

1. Un **nodo de administración** centralizado (*head*) el cual se encarga de recibir las solicitudes de ejecución de trabajos, determinar cual nodo trabajador es el idóneo para ejecutarlas, enviar los trabajos encolados a los

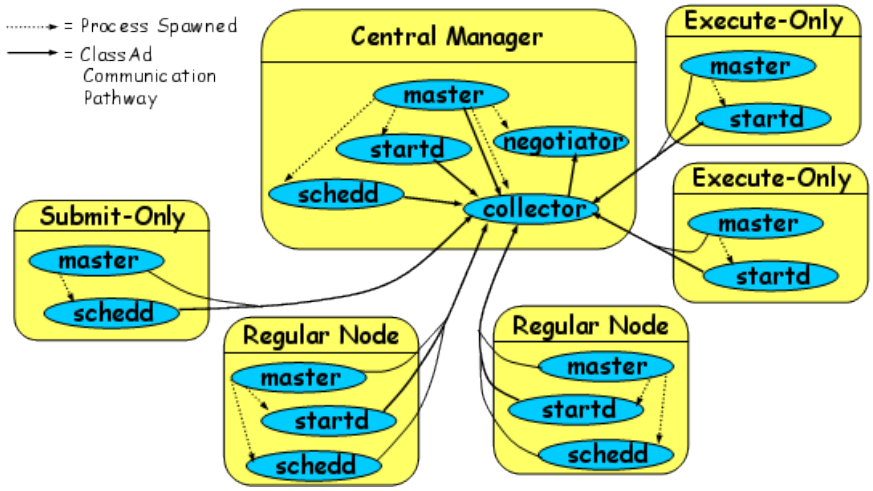
[86] University of Wisconsin. Página web. [en línea] <http://www.cs.wisc.edu/> [Consultado el 10 de junio de 2011].

nodos trabajadores y consolidar los estados y respuestas recibidas por parte de los nodos trabajadores.

2. Los **nodos de ejecución o trabajadores** (*worker nodes*) se encargan de ejecutar físicamente los trabajos enviados por los usuarios y se encuentran coordinados por el nodo de administración.

3. Los **nodos de envío de trabajos** (*submit nodes*) son los únicos componentes del nodo que se encuentran autorizados para que a través de ellos los usuarios envíen trabajos al *cluster*.

Existen otros tipos opcionales de nodos como el servidor de puntos de revisión (*checkpoint*) o los nodos de almacenamiento que no son incluidos durante el desarrollo de este proyecto.



Gráfica 1: Arquitectura típica de Condor. Tomada de http://www.cs.wisc.edu/condor/tutorials/scotland-admin-tutorial-2003-10-23/

A nivel de software, Condor implementa los roles de los componentes mencionados anteriormente mediante la especificación de nueve demonios.

1. `condor_master` realiza la gestión (iniciar, reiniciar y detener) sobre los otros demonios y se ejecuta en todos los servidores del *cluster*.
2. `condor_collector` y `condor_negotiator` se ejecutan en el nodo de administración únicamente y se encargan respectivamente de mantener el estado de todos los nodos y determinar cuales recursos están disponibles para la ejecución de los trabajos.
3. `condor_startd` se ejecuta en los nodos trabajadores y maneja las características de los recursos del nodo para conciliar los requerimientos de

los trabajos recibidos.

4. `condor_starter` es iniciado por `condor_startd` y se encarga de establecer el entorno para la ejecución de los trabajos remotos y realizar su seguimiento.
5. `condor_schedd` se ejecuta en las máquinas de envío de trabajos y se encarga de manejar las colas de trabajos.
6. `condor_shadow` se ejecuta en las máquinas de envío de trabajos y se encarga de manejar a los trabajos en ejecución ¿remota?.
7. `condor_ckpt_server` (opcional) se encarga de administrar (almacenar y recuperar) los archivos de revisión (*checkpoint files*).
8. `condor_kbdd_daemon` (opcional) se encarga de determinar en ciertas plataformas cuando hay o no actividad de entrada del usuario (teclado y ratón).

Funcionalmente Condor divide en los siguientes universos los diferentes tipos de trabajos que pueden ejecutarse en el *cluster*.

1. Universo `vanilla`, se utiliza para ejecutar *scripts*, comandos del sistema operativo y aplicaciones en general de las cuales no se tenga acceso al código fuente. Su funcionalidad es limitada y no cuenta con soporte para puntos de revisión.

2. Universo `standard`, requiere que las aplicaciones sobre las cuales se basan los trabajos a ejecutarse en el *cluster* se recompilen con las librerías de Condor. Cuenta con soporte para puntos de revisión, invocación a llamados remotos e incluye limitaciones técnicas (`fork` y `socket` entre otras).
3. Universo `java`[87], permite la ejecución de aplicaciones basadas en la Máquina Virtual de Java (JVM) sobre el *cluster*.
4. Universo `parallel`[88] permite la ejecución de aplicaciones en paralelo en *cluster*, incluyendo a aplicaciones basadas en MPI.
5. Universo `vm`[89] permite la ejecución de máquinas virtuales basadas en VMWare, XEN y KVM como trabajos completos.
6. Universo `grid`[90] permite la ejecución de trabajos en el nodo *grid* a través del encapsulamiento de las interfaces provistas por el Globus Toolkit.

El ámbito del alcance de este proyecto incluye a los universos `vanilla`,

---

87 Java Applications en Condor. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.6/2_8Java_Applications.html> [Consultado el 15 de abril de 2011].

88 Parallel Applications en Condor. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.6/2_9Parallel_Applications.html> [Consultado el 15 de abril de 2011].

89 Virtual Machine Applications en Condor. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.6/2_11Virtual_Machine.html> [Consultado el 15 de abril de 2011].

90 The Grid Universe en Condor. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.5/5_3Grid_Universe.html> [Consultado el 15 de abril de 2011].

`standard`, `java` y `grid`; los universos `parallel` y `vm`, así como otras características que ofrece Condor como es el caso de DAGMan[91] para determinar el orden del flujo en que se ejecutan los trabajos, quedan enunciadas para ser incluidas en una posible continuación del proyecto.

Mediante el universo `grid`, el nivel de *cluster* se articula con el siguiente nivel a implementarse que corresponde con la *grid*. Esto se realiza a través del Globus Toolkit que actúa como capa intermedia y que es utilizado por la infraestructura de Open Science Grid (OSG) sobre la cual se basa la interacción con Grid Colombia.

3.1.6 Mallas computacionales o *grids*.

Las mallas computacionales permiten dar un paso más allá en la abstracción de los sistemas distribuidos con relación a los *clusters* ya que permiten agrupar de manera distribuida los recursos computacionales ubicados geográficamente de manera dispersa y en diferentes límites administrativos para compartir y maximizar su uso de manera transparente, segura, eficiente y confiable.

[91] DAGMan Applications en Condor. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.6/2_10DAGMan_Applications.html> [Consultado el 15 de abril de 2011].

Su arquitectura puede ser centralizada cuando hay ciertos niveles jerárquicos entre sus componentes o descentralizada cuando no existen relaciones previas entre sus componentes, como es el caso de las aplicaciones *Peer-to-Peer* (P2P)[92].

En términos generales, las *grids* permiten utilizar y compartir los recursos de cómputo de los equipos de una organización y trascender los límites físicos y administrativos de éstas permitiéndoles a su vez articularse con *grids* de otras organizaciones (*clusters* de *clusters*). Esto ha permitido no sólo aprovechar mejor los recursos locales sino también el aprovechar, mediante convenios o contratos, el poder de cómputo existente en lugares remotos, por este motivo, este tipo de sistemas distribuidos ha tenido buena acogida tanto en el ámbito industrial como en el académico al interior de las universidades y los centros de investigación.

Sus características generales son las siguientes.

- **Ambientes heterogéneos**. Los sistemas que se unen a la *grid* pueden ser de diferentes características (sistemas operativos, arquitecturas, plataformas, etc.) y es responsabilidad de la misma el proveer las interfaces de homogenización que se requieran.

[92] P2P (*peer-to-peer*). Redes de sistemas o aplicaciones que presentan conjuntamente comportamientos de cliente y de servidor interactúando directamente entre sí sin la necesidad de servidores adicionales que actúen como mediadores.

- **Utilización de *middlewares* para la comunicación de alto nivel**. Estas capas de traducción intermedia facilitan la abstracción entre los componentes, simplificando las interfaces entre los niveles propietarios con el nivel de aplicación, permitiendo así la interacción entre ambientes heterogéneos.

- **Capacidad de balanceo de cargas**. Mejora el aprovechamiento de los recursos al derivar las solicitudes de los usuarios a componentes que presenten recursos libres en un momento específico del tiempo, optimizando el desempeño general del sistema.

- **Control y tolerancia a fallos**. Provee mecanismos para detectar y manejar los problemas (de hardware y software) que sucedan en el sistema. Ante un eventual problema permite que los procesos terminen completamente su ciclo de ser posible, e informen adecuadamente de lo sucedido para que el sistema global realice las acciones que se consideren pertinentes para garantizar el cumplimiento exitoso de la totalidad de las solicitudes de los usuarios.

- **Altos niveles de seguridad**. Garantizar el resguardo de la información y el acceso a los recursos disponibles de manera segura mediante el uso de

certificados que permitan validar la autenticación de las máquinas que hacen parte de la *grid* y de los usuarios que acceden a sus recursos.

- **Uso intensivo de la red (alta velocidad)**. Las redes de datos sirven de unión entre los componentes dispersos de la *grid*. Su desempeño y seguridad impactan directamente en el desempeño y disponibilidad general del sistema.

- **Concurrencia y soporte para gran cantidad de usuarios**. El sistema debe atender de manera simultánea a una gran cantidad de usuarios sin que esto afecte su desempeño general. Estos usuarios a su vez podrán ejecutar varios procesos en la *grid* sin que se produzcan interferencias entre ellos.

- **Transparencia en el acceso de los recursos**. Los recursos de la *grid* son ubicados y accedidos de una manera única y normalizada sin que sea necesario conocer su ubicación física o detalles técnicos específicos por parte de los usuarios o de los programas.

- **Alta escalabilidad**. El sistema podrá ser alterado para mejorar su capacidad de infraestructura sin que esto afecte negativamente su

desempeño.

Los componentes básicos de la *grid* son similares a los descritos en el nivel de *cluster* con varias mejoras ya que este nivel superior requiere de una mayor infraestructura para la verificación y control de permisos, el balanceo de cargas, y la comunicación entre diferentes ámbitos administrativos y ambientes heterogéneos.

1. Un **Nodo Central** gestiona la actividad de los recursos *grid*, verifica la autenticación de sus diferentes componentes y recibe las solicitudes de trabajos por parte de los usuarios que posteriormente serán delegadas para su ejecución. Este nodo comúnmente se conoce como el Elemento Computacional (*Compute Element*).

2. Los **Nodos Trabajadores** (*Worker Nodes*) ejecutan finalmente los trabajos solicitados por los usuarios y retornan los resultados al Elemento Computacional para que este se los entre a los usuarios solicitantes. Los trabajos recibidos por el nodo *grid* pueden ser enviados a *clusters* para su ejecución, tal y como se realiza en este proyecto, en el cual se utilizan los *clusters* basados en Condor implementados durante el primer nivel del proyecto.

3. **Nodos de autenticación** gestionan la identificación fidedigna de los usuarios y los demás componentes del nodo *grid*.

4. Los **Nodos Cliente** permiten a los usuarios autorizados enviar tareas para que se ejecuten en el nodo *grid*.

5. Los **Nodos de Almacenamiento** permiten disponer de grandes capacidades de almacenamiento para aplicaciones y datos al interior de los nodos *grid*. Esta característica se hace necesaria a medida que se hace mas compleja la arquitectura del nodo, en caso contrario su necesidad puede suplirse con el uso de un sistema de archivos compartido como NFS.

6. Los **Nodos para la administración de las Organizaciones Virtuales** son los encargados de realizar la gestión y control de las organizaciones virtuales de la *grid* y de los recursos que pertenecen a estas.

En términos generales la *grid* computacional se encarga de realizar las siguientes gestiones sobre los elementos de cómputo.

1. **Gestión de los recursos distribuidos (equipos de cómputo)**. Provee el

registro, seguridad y control de los componentes de la *grid*.

2. **Mantenimiento de los servicios de información**. Provee el descubrimiento y monitoreo de los recursos distribuidos.

3. **Gestión de los datos**. Facilita el transporte y replicación de la información entre los diferentes componentes del sistema según se requiera.

A nivel de gestión, las *grid* son mas complejas y elaboradas que los *clusters*, ya que las primeras pueden agrupar recursos sin importar ningún tipo de límites administrativos, es decir, pueden relacionar recursos que se encuentren dispersos de manera global y sean administrados por personas o entidades distintas.

Para facilitar la gestión de los recursos a gran escala se establecieron las **Organizaciones Virtuales (VOs)** las cuales agrupan recursos (usuarios e infraestructura) al rededor de objetivos comunes, tal y como es el caso de la VO de Biotecnología que se espera crear en la ciudad de Manizales en el corto plazo a partir de la infraestructura implementada por este proyecto.

Este tipo de agrupamientos son de carácter lógico ya que no requieren de ningún tipo de modificación física o de ubicación geográfica para su implementación,

dinámicos puesto que son flexibles y su constitución puede variar a través del tiempo, y sobre ellos se pueden establecer reglas administrativas que implementen políticas de acceso y seguridad a los recursos que involucran.

3.1.7 *Middleware* o capa de abstracción intermedia.

El *middleware* es una capa especial de software que se implementa en los sistemas computacionales para proveer una abstracción entre dos componentes (que pueden ser por ejemplo hardware, aplicaciones de software, plataformas o arquitecturas) heterogéneos y permitirles interactuar de manera transparente, es decir, sin que cada uno de estos componentes requiera modificaciones drásticas en su implementación para adecuarse a los demás tipos de componentes con los que espera interactuar.

Un caso ampliamente conocido es la pila de TCP/IP que se encuentra inmersa en prácticamente cualquier sistema operativo y permite que los computadores interactúen a través de redes como Internet sin importar el hardware o el software que utilicen.

En el ámbito de las *grids*, en la actualidad existen varios tipos de *middleware* entre

los cuales se destacan dos de ellos.

1. **Globus Toolkit**[93], desarrollado por Globus Alliance[94] es utilizado por software de mayor nivel como OpenPBS[95], Open Science Grid[4], Condor[8] y Oracle Grid Engine[96]. Es el *middleware* utilizado por la infraestructura de *grid* de Estados Unidos.

2. **gLite**[97], desarrollado por la comisión europea a través de la iniciativa EGEE[98] para la creación de una infraestructura multidisciplinaria de computación en grid para el aprovechamiento científico. Es el *middleware* utilizado por la infraestructura de *grid* de la Comunidad Europea.

3.1.8 Open Science Grid (OSG).

---

93 Globus Toolkit. Sitio web. [en línea] <http://www.globus.org/toolkit/> [Consultado el 10 de abril de 2010]

94 Globus Alliance. Sitio web. [en línea] <http://www.globus.org/> [Consultado el 10 de abril de 2010]

95 OpenPBS – Portable Batch System. Sitio web. [en línea] <http://www.pbsworks.com/> [Consultado el 10 de abril de 2010]

4 Open Science Grid - A national, distributed computing grid for data-intensive research. Sitio web. [en línea] <http://www.opensciencegrid.org/> [Consultado el 10 de abril de 2010]

8 Condor – High throughput computing. Sitio web. [en línea] <http://www.cs.wisc.edu/condor/> [Consultado el 10 de abril de 2010]

96 Oracle. Oracle Engine. Sitio web. [en línea] <http://www.sun.com/software/sge/> [Consultado el 10 de abril de 2010]

97 gLite – Lightweight middleware for Grid Computing. Sitio web. [en línea] <http://glite.web.cern.ch/glite/> [Consultado el 10 de abril de 2010]

98 EGEE – Enabling s for E-sciencE. Sitio web. [en línea] <http://www.eu-egee.org/> [Consultado el 10 de abril de 2010]

OSG[99] es administrada por la Open Science Grid Consortuim y reúne a proveedores de servicios y recursos, investigadores y centros de cómputo de varios países, especialmente de Estados Unidos y Brasil. Fue creada originalmente por el Departamento de Energía y la Fundación para la Ciencia Nacional (NSF)[100] de los Estados Unidos para dar soporte a los requerimientos de cómputo generados por el análisis de la información proveniente del Gran Colisionador de Hadrones (LHC)[101] y posteriormente ha sido adoptada por instituciones académicas y de investigación de diversa índole[102].

Su misión consiste en ayudar a satisfacer la creciente necesidad de capacidad de cómputo y almacenamiento de los investigadores, especialmente para la colaboración científica que requiere de computación de alto rendimiento.

Como tal la Open Science Grid no posee o controla recursos propios pero provee el acceso a recursos de almacenamiento y procesamiento en una *grid* internacional gracias a que da soporte a los proveedores de recursos y las instituciones científicas que los utilizan. Así mismo provee el software (a través del

---

[99] OSG – Open Science Grid Consortuim. Sitio web. [en línea] <http://www.opensciencegrid.org/> [Consultado el 10 de abril de 2010]

[100] NSF – National Science Foundation. Sitio web. [en línea] <http://www.nsf.gov/> [Consultado el 10 de abril de 2010]

[101] CERN. LHC – Large Hadron Collider. Sitio web. [en línea] <http://lhc.web.cern.ch/lhc/> [Consultado el 10 de abril de 2010]

[102] Organizaciones Virtuales (VO) registradas actualmente en OSG. [en línea] <http://www.opensciencegrid.org/VO_List> [Consultado el 25 de julio de 2011].

Virtual Data Toolkit[103] que facilita su instalación) y los servicios necesarios para acceder a estos recursos compartidos.

Las siguientes son las características de un trabajo idóneo para aprovechar al máximo las facilidades que provee la *grid* de OSG[104].

- Las aplicaciones están basadas en GNU/Linux con arquitecturas x86 (32 bits) o x86_64 (64 bits).
- La aplicación es de un solo hilo o multihilo pero no requiere del paso de mensajes.
- El tiempo de ejecución máximo de la aplicación está entre 1 hora y 24 horas.
- La aplicación maneja correctamente el ser terminada inesperadamente para posteriormente ser restaurada nuevamente.
- La aplicación se encuentra construida a partir de software que no requiere contactar con un servidor de licencias.
- Los problemas científicos pueden ser descritos con un flujo consistente y ordenado de subtrabajos.
- La solución del problema radica en la ejecución de un número muy grande

[103] The Virtual Data Toolkit (VDT). [en línea] <http://vdt.cs.wisc.edu/> [Consultado el 25 de julio de 2011]
[104] Succeeding on the OSG. [en línea] <https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/WhatIsOSG> [Consultado el 25 de julio de 2011].

de trabajos pequeños en lugar de unos pocos trabajos muy grandes.

Para la implementación de este proyecto se utiliza la infraestructura provista por el Open Science Grid Consortuim que es la mas utilizada por la mayoría de las *grids* desarrolladas en el seno de las comunidades técnicas y de investigación del continente Americano, incluida la organización Grid Colombia con la cual este proyecto se integra para formar parte de la infraestructura de *grid* a nivel nacional.

3.1.9 Grid Colombia.

Es una iniciativa[105] que articula a las universidades y centros de investigación colombianos con el fin de constituir una *grid* de escala nacional en Colombia que reúna los *clusters* y nodos *grid* de las universidades e instituciones de investigación, haciendo uso de las redes de tecnología avanzada a través del logro de los siguientes objetivos.

1. Conexión de *clusters* y *grids* inicialmente independientes a través de RENATA.
2. Fomentar el uso de *grids* como herramientas para solucionar problemas que necesiten alto nivel de cómputo.

[105] Grid Colombia. Página web. [en línea] <http://gridcolombia.org/> [Consultado el 21 de julio de 2011].

3. Participar en proyectos a nivel nacional e internacional.
4. Promover la colaboración de grupos de interés, estudiantes y docentes.
5. Desarrollar aplicaciones de alta carga computacional para resolver problemas de alto impacto para el país.

La visión de este proyecto se encuentra enmarcada entre los siguientes elementos.

- Constituirse en la primera alternativa en computación distribuida en el país.
- Todas las universidades conectadas a RENATA que estén desarrollando proyectos de computación distribuida hagan parte de Grid Colombia.
- Ofrecer soluciones y aplicaciones útiles en el contexto colombiano.
- Colombia sea visible, reconocida y certificada como una alternativa de *grid* a nivel internacional.

Para su desarrollo se han establecido diferentes frentes de trabajo que abordan el proyecto desde sus múltiples perspectivas[106].

- **Infraestructura**: encargado de preparar, diseñar, implementar y configurar una *grid* de Cómputo Nacional que permita agrupar recursos

[106] Tomado de "Fuerza de trabajo Iniciativa de Grid Nacional – Grid Colombia".

computacionales de las instituciones participantes en el proyecto, la interconexión de los nodos se realizará a través de la Red Nacional de Tecnología Avanzada RENATA.

- **Formación y capacitación**: encargada de fomentar el desarrollo de la formación de estudiantes, docentes y especialistas en computación de alto desempeño, además para la integración de un servicio de cómputo avanzado interdisciplinario en Grid Colombia. Esta formación incluye capacitación en las tecnologías de *clusters* y *grids*, programación concurrente y paralela, y desarrollo de aplicaciones.

- **Promoción y divulgación**: el objetivo es plantear y realizar estrategias de promoción y divulgación del proyecto Grid Colombia para difundir y ampliar el uso de las tecnologías de grilla y obtener un reconocimiento nacional e internacional.

- **Seguridad**: su función es proponer y analizar los aspectos concernientes a la seguridad informática para la implantación de una infraestructura de computación en *grid* a nivel nacional, garantizando el cumplimiento de estándares para esta tecnología.

Actualmente las siguientes instituciones académicas[107] se encuentran participando en el proyecto Grid Colombia a través de sus respectivos grupos de investigación[108] y utilizando sus respectivas Redes Académicas Regionales que interconectan sus nodos *grid* para crear la gran *grid* nacional.

1. Politécnico Gran Colombiano (RUMBO).
2. Pontificia Universidad Javeriana (RUMBO).
3. Universidad Antonio Nariño (RUMBO).
4. Universidad Católica de Colombia (RUMBO).
5. Universidad de Los Andes (RUMBO).
6. Universidad Distrital Francisco José de Caldas (RUMBO).
7. Universidad Nacional de Colombia, Bogotá (RUMBO).
8. Centro de Investigaciones de Café -CENICAFÉ- (RADAR).
9. Universidad Autónoma de Manizales (RADAR).
10. Universidad de Antioquia (RUANA).
11. Universidad de Caldas (RADAR).
12. Universidad de Medellín (RUANA).
13. Universidad Pontificia Bolivariana de Medellín (RUANA).
14. Universidad Tecnológica de Pereira (RADAR).

[107] GRID COLOMBIA. Instituciones conectadas. [en línea] <http://gridcolombia.org/index.php?option=com_content&view=article&id=60&Itemid=77> [Consultado el 25 de julio de 2011]

[108] GRID COLOMBIA. Grupos de investigación. [en línea] <http://gridcolombia.org/index.php?option=com_content&view=article&id=62&Itemid=78> [Consultado el 25 de julio de 2011]

15. Universidad Autónoma de Bucaramanga (UNIRED).
16. Universidad Cooperativa de Colombia en Bucaramanga (UNIRED).
17. Universidad Industrial de Santander (UNIRED).
18. Centro Internacional de Agricultura Tropical -CIAT- (RUAV).
19. Universidad Autónoma de Occidente (RUAV).
20. Universidad del Valle (RUAV).
21. Universidad Javeriana de Cali (RUAV).
22. Universidad Autónoma del Caribe (RUTA).
23. Universidad del Atlántico (RUTA).
24. Universidad del Norte (RUTA).
25. Universidad Tecnológica de Bolivar (RUTA).

3.1.10 Redes académicas de alta velocidad.

Son redes de alto desempeño cuyo fin es el de interconectar a las comunidades universitarias, académicas y científicas de todos los países para fortalecer el desarrollo de la ciencia y la tecnología, garantizando la comunicación y colaboración efectiva a través de medios que no hubieran sido posibles a través de las redes convencionales.

En Colombia, el origen de estas redes surge naturalmente de la asociación de las universidades o entidades científicas cercanas que luego se constituyen en nodo. Posteriormente estos nodos se unen a través de un anillo de alta velocidad y a su vez este anillo se une a las redes de alta velocidad del mundo como Internet2[109], GÉANT2[110] y CLARA[111].

En Colombia hay 103 instituciones interconectadas a la Red Nacional Académica de Tecnología Avanzada (RENATA) a través de 8 redes regionales[112] que las agrupan. Para el eje cafetero la red regional es RADAR y reúne a 18 universidades e instituciones del centro-occidente del país.

Estas redes y permiten la interconexión con otras redes académicas de alta velocidad internacionales a través de RENATA, lo cual es necesario para interactuar con la infraestructura la de la *grid* nacional y aprovechar también la infraestructura internacional que provee Open Science Grid (OSG).

---

[109] Internet2. Página web. [en línea] <http://www.internet2.edu/> [Consultado el 10 de abril de 2010]
[110] GÉANT2. Página web. [en línea] <http://www.geant2.net/> [Consultado el 10 de abril de 2010]
[111] Cooperación Latino Americana de Redes Avanzadas. Página web. [en línea] <http://www.redclara.net/> [Consultado el 10 de abril de 2010]
[112] Redes Académicas Regionales. Página web. [en línea] <http://www.renata.edu.co/index.php/redes-academicas-regionales.html> [Consultado el 10 de abril de 2010]

3.1.10.1 Red Nacional Académica de Tecnología Avanzada (RENATA)[113].

Es la red académica de alto desempeño que se interconecta con las redes regionales en los diferentes sectores de Colombia. Su valor estriba en el fortalecimiento de la comunicación y colaboración entre la comunidad académica y los centros de investigación de Colombia y el mundo a través de los principios de colaboración, innovación, desarrollo tecnológico y calidad del servicio.

Es administrada por una corporación que lleva su mismo nombre y de la cual son miembros las Redes Académicas Regionales, el Ministerio de Educación, el Ministerio de Tecnologías de la Información y las Comunicaciones y Colciencias.

Sus objetivos son los siguientes.

1. Ofrecer un servicio de conectividad de alta calidad usando tecnologías avanzadas.
2. Consolidar una red nacional con el mayor número de instituciones académicas y de investigación que hagan uso efectivo de la red.
3. Ofrecer servicios que faciliten y promuevan el intercambio eficiente de información y comunicaciones, así como el trabajo colaborativo entre las

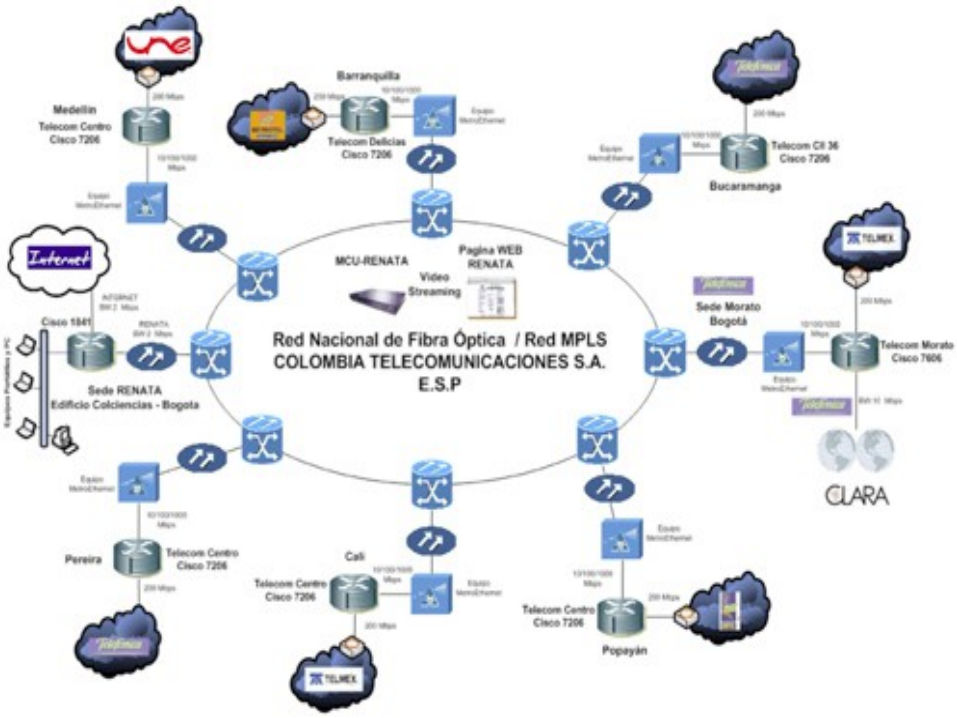
[113] RENATA, Quienes somos ? [en línea] <http://www.renata.edu.co/index.php/quienes-somos-identidad-y-objetivos-de-renata.html> [Consultado el 10 de abril de 2010]

instituciones nacionales e internacionales.

4. Estimular la ejecución de proyectos de educación, investigación y desarrollo que contribuyan a la competitividad y el progreso del país.
5. Desarrollar acciones y alianzas que contribuyan al desarrollo y sostenibilidad de RENATA.

Técnicamente su infraestructura[114] está basada en una topología de estrella jerárquica donde el punto central es la sede Morato de Colombia Telecomunicaciones en Bogotá, los puntos de la estrella los conforman los nodos principales de las Redes Académicas Regionales de las ciudades de Cali, Barranquilla, Medellín, Bucaramanga, Pereira (Eje Cafetero), Popayán y Bogotá, en donde se interconectan a cada uno de los operadores locales que manejan las redes metropolitanas de las universidades. Actualmente cada nodo cuenta con 200Mbps de ancho de banda hacia en nodo central (Morato-Bogotá).

[114] RENATA. Características técnicas. [en línea] <http://www.renata.edu.co/index.php/caracteristicas-tecnicas-de-la-red.html> [Consultado el 10 de abril de 2010]



Gráfica 2: Topología de la red RENATA. Tomada de *http://www.renata.edu.co/index.php/caracteristicas-tecnicas-de-la-red.html*

Actualmente RENATA ofrece los siguientes servicios[115] a sus instituciones miembros:

1. Conectividad.
2. Videoconferencia.
3. Servicio de transmisión de eventos (s*treaming*).
4. Oficina virtual.

[115] RENATA, Nuestros servicios. [en línea] <http://www.renata.edu.co/index.php/nuestros-servicios.html> [Consultado el 10 de abril de 2010]

5. Recursos e-ciencia.
6. Formación.
7. Gestión para la colaboración.
8. Promoción para el desarrollo de proyectos.
9. Información, difusión y divulgación.

El potencial de este tipo de redes académicas de alta velocidad permitirán implementar proyectos de desarrollo científico y cultural que no hubieran sido posibles a través de las redes de datos convencionales. Entre los proyectos más destacados[116] que se podrán implementar se cuenta con los siguientes:

- Acceso a recursos a distancia como instrumentación remota, robots, telescopios, microscopios, equipos de medición y laboratorios virtuales.
- Procesamiento masivo y distribuido de datos a través de *clusters*, servicios de caché, espejos, supercomputación y creación de *grids*.
- Demostraciones técnicas utilizando recursos que están en diferentes sitios y simular ambientes y modelos virtuales.
- Compartir recursos como bibliotecas digitales, sistemas de indexación audiovisual, directorios digitales, manejadores de contenido y bases de

[116] RENATA, Posiblidades que brindan las redes avanzadas. [en línea] <http://www.renata.edu.co/index.php/quienes-somos-identidad-y-objetivos-de-renata.html?start=4> [Consultado el 10 de abril de 2010]

datos digitales.

- Comunicaciones presenciales integradas de vídeo, voz sobre protocolos IP, datos, Videotecas para el servicio de Vídeo *On Demand* (VOD)[117], videoconferencia, colaboración interactiva, televisión de alta definición, y *streaming*[118].
- Creación de espacios virtuales como teleinmersión (entorno virtual compartido en tiempo real, realidad virtual) permitiendo reproducir la realidad por medio de recursos informáticos.

3.2 PERSPECTIVA GENERAL DEL DESARROLLO

Metodológicamente el proyecto se dividió en tres niveles a través de los cuales se lograron los objetivos del proyecto de manera incremental.

En el **nivel de *cluster*** se determinó el hardware con que se contaba en cada Universidad para la implementación del nodo. Se realizó la instalación y configuración básica de dicha infraestructura, se determinó la arquitectura del

[117] *Video On Demand* (VOD). Este servicio le permite a los usuarios acceder a contenidos multimediales (imágenes, audio y vídeo) de manera personalizada permiténdole elegir el contenido y el momento exacto de su presentación. Bajo esta arquitectura se fundamentan los servicios de pague-por-ver.

[118] *Streaming*. Tecnología de distribución de datos a través de redes como Internet que permite a los usuarios acceder a recursos de audio y vídeo sin interrupciones mediante el establecimiento de un flujo continuo de datos entre el servidor y el cliente, almacenando sólo un fragmento de la información en un *buffer* de este último. Gracias a esta arquitectura, permite la publicación de recursos multimedia de gran tamaño y ser consumidos por clientes de menores prestaciones.

cluster de acuerdo con el hardware y el software a utilizarse.Finalmente se llevó a cabo la instalación, configuración y pruebas de Condor como la plataforma elegida de código abierto para la paralelización distribuida de tareas computacionales intensivas que requieran de alto rendimiento. Se complementó este nivel con la documentación de las principales tareas de uso y administración del *cluster*, necesarias para la gestión de tareas (desde el punto de vista del usuario) y realizar su mantenimiento operativo para garantizar su correcto funcionamiento (desde el punto de vista del administrador).

Para el desarrollo de este nivel se participó en el OSG School desarrollado en la ciudad de Bucaramanga en el mes de febrero de 2010 en el cual ingenieros de Open Science Grid compartieron con la comunidad de Grid Colombia los lineamientos básicos de instalación y configuración de la infraestructura de software para los nodos. Adicionalmente se utilizó ampliamente la documentación provista por Open Science Grid en sus wikis la cual fue analizada, probada, apropiada y complementada según los requerimientos propios del proyecto.

Para la experimentación en la etapa de pruebas se hizo un amplio uso de máquinas virtuales basadas en KVM[119] que se ejecutaban sobre un servidor destinado para tal fin con un procesador AMD de cuatro núcleos y 4GB de

[119] Kernel Based Virtual Machine. Página web. [en línea] <www.linux-kvm.org/page/Main_Page> [Consultado en octubre 1 de 2011]

memoria RAM. Una vez estables y documentados los procedimientos fueron implementados en los servidores de producción ubicados en las dos Universidades que hicieron parte del proyecto.

Una vez terminada la implementación de este nivel se realizó un curso en julio de 2010 con representantes de las universidades y centros de investigación de la región para compartir con ellas el conocimiento adquirido por el proyecto hasta ese momento y facilitarles la implementación (instalación, configuración, uso y administración) de *clusters* al interior de sus instituciones.

Posterior al nivel de *cluster* se implementó el **nivel de *grid***. En él se determinó la arquitectura y los componentes que debería tener el nodo *grid* de acuerdo con los objetivos del proyecto, el hardware existente y los lineamientos provistos por Open Science Grid.

Para el desarrollo de este nivel se participó en el OSG School 2011 desarrollada en el mes de mayo en la ciudad de Manizales y se mantuvo un continuo contacto con las personas de apoyo de Open Science Grid mediante el correo electrónico y la lista de OSG-AMERICAS[120]. De igual modo se hizo amplio uso de la documentación expuesta en los wikis de Open Science Grid.

[120] Lista de correo electrónico de OSG para la coordinación de tareas con los nodos *grid* de sur américa. [en línea] <http://www.opensciencegrid.org/Consortium_Mailing_Lists> [Consultado en octubre 1 de 2011]

Durante el desarrollo de este nivel se trabajo de cerca con los integrantes del proyecto Grid Colombia con quienes se compartieron conocimientos y experiencias relacionadas con la implementación de la *grid* en los nodos de las instituciones a través de múltiples reuniones tanto virtuales como presenciales en la ciudad de Bogotá.

En este nivel se definieron procedimientos con Grid Colombia y Open Science Grid para la generación, manipulación e instalación de los certificados de seguridad necesarios para autenticar a los nodos y permitirles ser parte de la *grid*, así como el autenticar a los usuarios y permitirles acceder a los servicios de los nodos *grid*.

La etapa de pruebas se desarrolló inicialmente en los servidores en producción debido a la necesidad de contarse con los certificados de seguridad específicos al *hostname* real de los servidores para autenticarse como parte de la *grid*, sin embargo posteriormente se determinó que estos podían ser utilizados en máquinas de prueba que tuvieran temporalmente el mismo *hostname* y se continuó utilizando el esquema de máquinas virtuales utilizado en el nivel de *cluster*. Una vez estables y documentados los procedimientos fueron implementados por completo en los servidores de producción ubicados en las dos

Universidades que hicieron parte del proyecto.

Una vez terminada la implementación de este nivel se realizaron múltiples pruebas de conectividad y funcionamiento a nivel interno (con los nodos propios) y a nivel externo (con los nodos de otras instituciones de Grid Colombia). Se complementó este nivel con la documentación de las principales tareas de uso y administración del nodo *grid*, necesarias para la gestión de tareas (desde el punto de vista del usuario) y realizar su mantenimiento operativo para garantizar su correcto funcionamiento (desde el punto de vista del administrador).

Como cierre del desarrollo del nivel de *grid* se dictó un taller de computación de alto desempeño con *grids* en el Sexto Congreso Colombiano de Computación (6CCC) que se llevó a cabo en el mes de Mayo de 2011, en el cual se presentaron los logros del proyecto a la comunidad académica y se mostró de manera práctica la gestión de trabajos en el nodo *grid* desde la perspectiva del usuario final.

El **nivel de Documentación** se realizó de manera perpendicular al desarrollo de los dos niveles descritos anteriormente. La documentación del proyecto se formalizó en un wiki[121] que puede consultarse en línea desde Internet y que continente la totalidad de los conocimientos obtenidos durante el desarrollo del

[121] Sitio wiki del Proyecto. Página web. [en línea]. <http://griduam.esencial.co/wiki/> [Consultado en octubre 1 de 2011]

proyecto, dispuestos en secciones haciendo énfasis en la separación entre las actividades de instalación y configuración, y las de administración y uso para facilitar el acceso a la información a los usuarios de la *grid* (investigadores) y a los administradores quienes se consideran dos perfiles muy diferentes.

Adicionalmente a los objetivos del proyecto se ha trabajado de cerca con el grupo de investigación de Ingeniería de Software[122] de la Universidad Autónoma de Manizales en la instalación, configuración y prueba distintos paquetes de software que puedan ser aprovechados en el nivel de *cluster* o de *grid*, especialmente los enfocados en el área de Bioinformática y que puedan dar soporte a la elaboración de proyectos conjuntos con el Centro Nacional de Investigaciones de Café (CENICAFÉ)[123] y otras instituciones dedicadas a la investigación en esta área del conocimiento. Así mismo se trabaja de manera activa en la implementación de la Organización Virtual de Biotecnología a nivel nacional, iniciativa de la cual es líder la Universidad Autónoma de Manizales.

[122] Grupo de Investigación Ingeniería del Software, Universidad Autónoma de Manizales. Página web. [en línea]. <http://www.autonoma.edu.co/web/sitios/cmsimple/unidades/gruposinvestigacion/GIngSoftware/> [Consultado en octubre 1 de 2011]

[123] Centro Nacional de Investigaciones (CENICAFE). Página web. [en línea] <www.cenicafe.org/> [Consultado en octubre 1 de 2011]

3.3 INVENTARIO DE LA INFRAESTRUCTURA TECNOLÓGICA UTILIZADA

En este apartado se detallan los aspectos técnicos de la configuración de hardware de los servidores utilizados en los dos nodos *grid*. Esta información se encuentra en linea en el wiki del proyecto, por este motivo se proporcionan los enlaces requeridos para acceder a esta información.

1. Universidad Autónoma de Manizales[124]
2. Universidad del Rosario[125]

3.4 ARQUITECTURA DE LOS COMPONENTES

A continuación se detallan las arquitecturas de los componentes del proyecto estratificadas en los dos niveles que lo componen: *cluster* y *grid*.

---

[124] Inventario de la infraestructura tecnológica utilizada en la Universidad Autónoma de Manizales. Wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/proyecto:objetivos:infraestructura_hardware#nodo_universidad_autonoma_de_manizales> [Consultado en octubre 1 de 2011]

[125] Inventario de la infraestructura tecnológica utilizada en la Universidad del Rosario. Wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/proyecto:objetivos:infraestructura_hardware#nodo_universidad_del_rosario> [Consultado en octubre 1 de 2011]

3.4.1 Nivel de *cluster*

Condor describe los siguientes roles[126] que pueden ser implementados por las máquinas que pertenecen al nodo.

1. El **Administrador Central o Nodo Cabeza (*head*)** es la máquina de entrada al *cluster*, su misión es la de recoger toda la información de las actividades que realizan los demás equipos y de garantizar la negociación de entre recursos y solicitudes de recursos producto de la ejecución de las tareas. Estos dos servicios comúnmente se alojan en un único equipo que se encarga de estas tareas y de gestionar las solicitudes de los usuarios.

2. El rol de **Ejecución** es implementado por los **Nodos Trabajadores (*Worker Nodes*)** y son los encargados de finalmente realizar la ejecución de los trabajos solicitados por los usuarios del *cluster*. Se recomienda que este rol sea implementado en equipos con suficientes características técnicas (especialmente capacidad de procesamiento, memoria y almacenamiento) de acuerdo con los requerimientos de hardware de los trabajos enviados.

[126] The different roles a machine can play. Página web. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.0/3_1Introduction.html#SECTION00411000000000000000> [Consultado en octubre 1 de 2011]

3. El rol de **Envío de Trabajos** debe ser implementado por los **Nodos Cliente (*Submitter Nodes*)** del *cluster*, es decir, por cualquier equipo que necesite estar facultado para enviar (*submit*) trabajos al cluster (como lo son los equipos de los investigadores por ejemplo). La implementación de este rol evita que los usuarios que envían trabajos al *cluster* requieran obligatoriamente acceso directo (a través de SSH por ejemplo) al Nodo Cabeza.

4. El rol de **Servidor de Puntos de Verificación (*Checkpoint Server*)** centraliza la información de los *puntos de verificación*[127] sucedidos en el *cluster*, los cuales corresponden con la información del estado (*snapshot*) de un trabajo específico la cual puede ser migrada o continuada posteriormente. La presencia de este rol no hace parte de la distribución binaria estándar de Condor, es opcional, requiere de una amplia capacidad de almacenamiento y sólo debe ser implementado en ambientes de muy alto tráfico de trabajos.

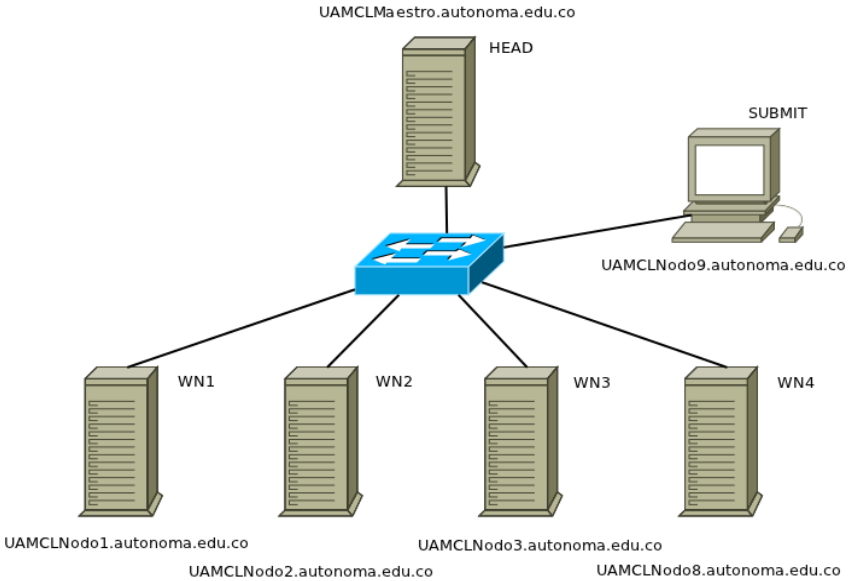
Para la implementación del nivel de *cluster* en el proyecto se eligió la siguiente arquitectura de componentes para los dos nodos de acuerdo con el hardware

[127] Condor's Checkpoint Mechanism. Página web. [en línea] <http://www.cs.wisc.edu/condor/manual/v7.0/4_2Condor_s_Checkpoint.html> [Consultado en octubre 1 de 2011]

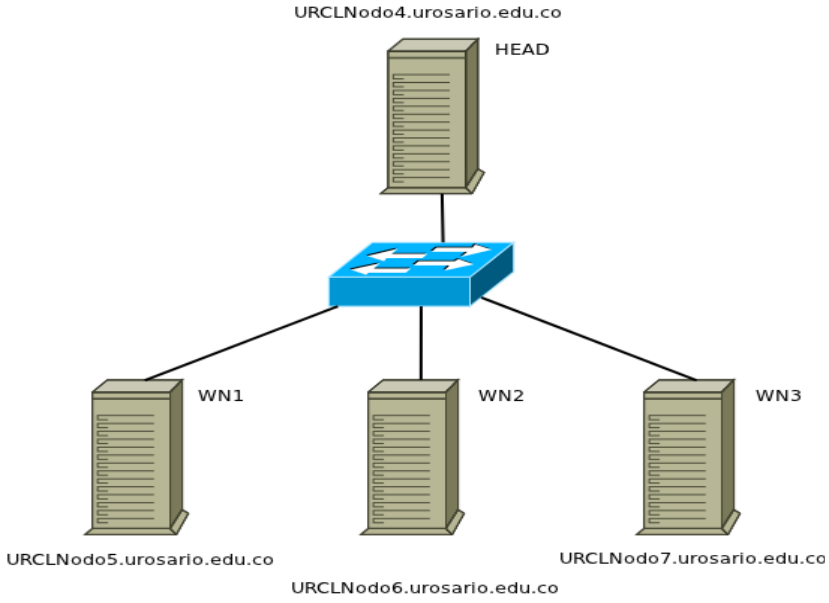
desplegado en cada una de las universidades.

En ambos sitios se utilizó el equipo con mejor configuración de hardware como nodo principal para encargarse de las actividades de coordinación del *cluster* (*head*) y los tres equipos restantes se destinaron para implementarse como nodos trabajadores (*wn*). Adicionalmente en la Universidad Autónoma de Manizales se utilizó un servidor adicional, originalmente destinado como servidor web, para que adicionalmente desempeñe el rol de nodo trabajador y temporalmente un equipo adicional que se configuró como equipo cliente (*submit*).

No se realizaron instalaciones de mas equipos cliente en los entornos de las Universidades ya que durante el desarrollo del proyecto no se establecieron usuarios de la infraestructura que explícitamente los requirieran. Con la instalación del cliente temporal (en el nodo9) se verificó el procedimiento y la funcionalidad entre estos y el *cluster*. El rol del servidor de puntos de verificación no fue tenido en cuenta para la arquitectura de este proyecto ya que el tráfico esperado en el *cluster* a corto y mediano plazo no será el suficiente para sacar provecho de este componente.



Gráfica 3: Arquitectura de cluster implementada en la Universidad Autónoma de Manizales



Gráfica 4: Arquitectura de cluster implementada en la Universidad del Rosario

3.4.2 Nivel de *grid*

Para la implementación de este nivel la elección de tecnologías, software y componentes que realizó el proyecto estuvo ampliamente ligada a las convenciones y estándares establecidos con el proyecto Grid Colombia con el cual este proyecto se relaciona para formar parte de la *grid* nacional. Es importante recordar que el proyecto Grid Colombia eligió a Open Science Grid como base técnica para la implementación de la *grid* colombiana.

Se identifican los siguientes componentes que hacen parte funcional de una *grid*.

1. El ***Compute Element*** **(CE)**[128] constituye la puerta inicial de la *grid* para los usuarios interesados en enviar trabajos al nodo. Este componente provee los siguientes servicios.

    a) ***Grid Resource Allocation Manager*** **(GRAM)**[129] este servicio es provisto por Globus y le permite al CE acceder al sistema de lotes (*batch system*) para solicitar la ejecución de trabajos en la *grid*. Adicionalmente permite

[128] Compute Element (CE). Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/OverviewOfServicesInOSG#Compute_Element_CE> [Consultado en octubre 1 de 2011]

[129] GRAM en The Globus Alliance. Página web. [en linea] <http://dev.globus.org/wiki/GRAM> [Consultado en octubre 1 de 2011]

la funcionalidad del manejador de trabajos *fork* que los ejecuta directamente en el equipo CE.

b) ***Grid File Transfer Protocol*** **(GridFTP)**[130] permite la transferencia de archivos entre usuarios y equipos autenticados en la *grid*.

c) **GRAM-WS (GRAM basado en Web Services)**[131] este servicio es obsoleto y no se recomienda que se utilice en producción.

d) **Squid**[132] es un servicio opcional de proxy/caché web que permite optimizar el acceso de la red por parte de los nodos trabajadores en condiciones especiales (como la instalación de paquetes) estableciendo restricciones de seguridad.

2. El ***Storage Element*** **(SE)**[133] permite el manejo (lectura y escritura) de grandes volúmenes de información al interior de la *grid* utilizando el protocolo *Storage Resource Manager* (SRM)[134] que actúa como *middleware*

[130] GridFTP en The Globus Alliance. Página web. [en línea] <http://www.globus.org/toolkit/data/gridftp/> [Consultado en octubre 1 de 2011]
[131] GRAM-WS en The Globus Alliance. Página web. [en línea] <http://globus.org/toolkit/docs/3.2/gram/ws/> [Consultado en octubre 1 de 2011]
[132] What is Squid? Página web. [en línea] <http://www.squid-cache.org/Intro/> [Consultado en octubre 1 de 2011]
[133] Data Storage and Management in the OSG. Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/Storage> [Consultado en octubre 1 de 2011]
[134] Storage Resource Management (SRM) Working Group. Página web. [en línea] <https://sdm.lbl.gov/srm-wg/> [Consultado en octubre 3 de 2011]

para proveer el acceso uniforme a fuentes de almacenamiento heterogéneas en la *grid*.

3. El ***OSG Client*** **(UI)** incluye todas las herramientas de linea de comando que necesita el usuario final (investigador) para interactuar con el nodo *grid*: manejo de certificados de usuario, consulta de información y gestión de trabajos.

4. El ***Grid User Management System*** **(GUMS)**[135] permite establecer la relación entre las credenciales de identificación de un usuario *grid* (DNs[136]) y la información de la cuenta de usuario local (UNIX o Kerberos) en los servidores del nodo. Es particularmente apropiado para el control centralizado de políticas de uso en ambientes con múltiples *gatekeepers*[137]. Si no se utiliza este servicio es posible usar el `edg-mkgridmap` el cual cumple una función similar, local y a menor escala, de manera análoga a la relación entre el servicio DNS y el archivo /etc/hosts en los sistemas operativos basados en UNIX.

---

[135] GUMS. Página web. [en línea] <https://www.racf.bnl.gov/Facility/GUMS/1.4/> [Consultado en octubre 1 de 2011]

[136] Distinguished Names. Página web. [en línea] <http://msdn.microsoft.com/en-us/library/windows/desktop/aa366101(v=vs.85).aspx> [Consultado en octubre 1 de 2011]

[137] Un *gatekeeper* es una capa de abstracción que facilita la transformación y ejecución de comandos enviados por clientes remotos los cuales son ejecutados en los sistemas de lotes locales del nodo *grid*. Actualmente los CE de OSG utilizan GRAM de Globus pero se espera que esto se migre en un futuro cercano a CREAM de EMI.

5. El ***VO Membership Service*** **(VOMS)**[138] es un sistema que maneja en tiempo real la información de membresía de los usuarios en una *Virtual Organization* (VO)[139] las cuales son agrupaciones lógicas de recursos (humanos y computacionales) que optimizan su uso compartido mediante el establecimiento de políticas de acceso.

6. El ***Virtual Organization Management Registration Service*** **(VOMRS)**[140] un conjunto de servicios de alto nivel que facilitan la administración del VOMS.

7. El ***Berkeley Database Information Index*** **(BDII)**[141] provee información del nodo *grid* acerca de la topología del sitio y el estado de los servicios, a través de una interfaz LDAP.

8. El ***Workflow Management System*** **(WMS)** permite la implementación de un servicio de flujo de trabajo al interior de la infraestructura de OSG. Los

[138] Virtual Organization Membership Service. Página web. [en línea] <http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html> [Consultado en octubre 1 de 2011]
[139] Virtual Organizations en OSG. Página web. [en línea] <http://www.opensciencegrid.org/About/Learn_About_Us/OSG_Organization/VOs> [Consultado en octubre 1 de 2011]
[140] Virtual Organization Management Registration Service (VOMRS). Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/VomrsInstallGuide> [Consultado en octubre 3 de 2011]
[141] CEMON / BDII Server. Página web. [en línea] <http://is.grid.iu.edu/documentation.html> [Consultado en octubre 3 de 2011]

sistemas mas utilizados para este fin son GlideinWMS[142] y PanDA[143].

9. El ***Grid Operation Center*** **(GOC)**[144] es un punto centralizado de soporte para las operaciones de la *grid* como servicios de monitoreo del desempeño de la *grid* en tiempo real, seguimiento a problemas, soporte a usuarios, desarrolladores y administradores, mantenimiento de servicios, proveer respuestas ante incidentes de seguridad y mantener la información de los repositorios.

10. Una **Entidad Certificadora (CA)** es el elemento encargado de gestionar los certificados[145] de un sistema basado en la infraestructura de llave pública (PKI): emitirlos, verificar su autenticidad (mediante la firma digital) y revocarlos.

11. El **Sistema de Lotes**[146] (*batch system*) o **Manejador de Trabajos** (*job*

[142] GlideinWMS. Página web. [en línea] <http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/index.html> [Consultado en octubre 3 de 2011]

[143] PanDA en OSG. Página web. [en línea] <http://www.opensciencegrid.org/PANDA> [Consultado en octubre 3 de 2011]
Pilot factory – a Condor-based system for scalable Pilot Job generation in the Panda WMS framework. Página web. [en línea] <http://iopscience.iop.org/1742-6596/219/6/062041> [Consultado en octubre 3 de 2011]

[144] What is the GOC ? Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/Operations/TheGOC> [Consultado en octubre 3 de 2011]

[145] What is a certificate ? Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/CertificateWhatIs> [Consultado en octubre 3 de 2011]

[146] About Batch Systems in OSG. Página web. [en línea] <https://twiki.grid.iu.edu/bin/view/Documentation/AboutBatchSystems> [Consultado en octubre 3 de

manager) permite gestionar la ejecución de los múltiples trabajos enviados al nodo *grid*. OSG soporta actualmente los siguientes: Fork[147] (Globus), Condor[148], PBS[149], SGE[150] y LSF[151].

3.4.2.1 Componentes centralizados

Se estableció que los siguientes componentes de la *grid* fueran instalados a nivel central por el proyecto Grid Colombia ya que estos prestan servicios requeridos a nivel general por todos los nodos *grid* de las instituciones Colombianas.

1. ***VO Membership Service* (VOMS)** y ***Virtual Organization Management Registration Service* (VOMRS).** Los servicios de gestión de VOs se encuentran centralizados pero pueden ser administrados remotamente por los usuarios encargados en las regiones.

2011]

147 *Fork* permite la ejecución de trabajos remotos únicamente en el equipo que sirve como *gatekeeper*. Es útil para pruebas de funcionamiento o instalación de software en el nodo. Probablemente sea contraproducente que los usuarios finales lo utilicen.

148 *Condor* es un software para la ejecución de trabajos que requiere de alto rendimiento (HTC) desarrollado por la Universidad de Wisconsin y utilizado en la implementación del nivel de *cluster* de este proyecto.

149 *Portable Batch System* (PBS). Página web. [en línea] <http://www.pbsworks.com/> [Consultado en octubre 3 de 2011]

150 *Oracle Grid Engine* (OGE) antes *Sun Grid Engine* (SGE). Página web. [en línea] <http://www.oracle.com/technetwork/oem/grid-engine-166852.html> [Consultado en octubre 3 de 2011]

151 *Load Sharing Facility* (LSF). Página web. [en línea] <http://www.platform.com/workload-management/high-performance-computing> [Consultado en octubre 3 de 2011]

2. ***Berkeley Database Information Index*** **(BDII).** Tanto el manejo de la información como la gestión y el monitoreo de la *grid* se realizan inicialmente por completo en el nivel central.

3. ***Workflow Management System*** **(WMS).** La implementación de flujos de trabajo se está experimentando inicialmente en el nivel central pero muy probablemente sea puesta implementada también en los nodos *grid* de las instituciones una vez que su instalación, configuración y uso estén completamente establecidos.

4. ***Grid Operation Center*** **(GOC).** Actualmente se utiliza el GOC de OSG sin embargo se ha discutido la posibilidad de implementar un GOC a nivel nacional para todas las instituciones de Colombia, inclusive se ha discutido la opción de implementar uno general que brinde los servicios de apoyo a todos los nodos *grid* de Latinoamérica.

5. **Entidad Certificadora (CA).** Grid Colombia cuenta con su entidad certificadora propia lo que nos faculta como país para realizar la gestión de nuestros propios certificados.

3.4.2.2 Componentes locales

Se establecieron dos posibles arquitecturas para los componentes a instalarse en los nodos *grid* de las instituciones de acuerdo con el tamaño de los sitios y el tráfico estimado para estos.

Para la implementación de este proyecto se eligió utilizar la arquitectura de nodo pequeño ya que inicialmente se contará con muy bajo tráfico y con recursos de hardware limitados. Es importante hacer notar que los nodos *grid* implementados bajo esta arquitectura podrán ser modificados con los componentes adicionales incluidos para el tamaño mediano y soportar así un mayor tráfico.

3.4.2.2.1 Nodo *grid* pequeño

Para los nodos *grid* con baja cantidad de trabajos por unidad de tiempo, el proyecto sugiere instalar los siguientes componentes.

1. Un *Compute Element* (CE) que implementa GRAM y GridFTP.

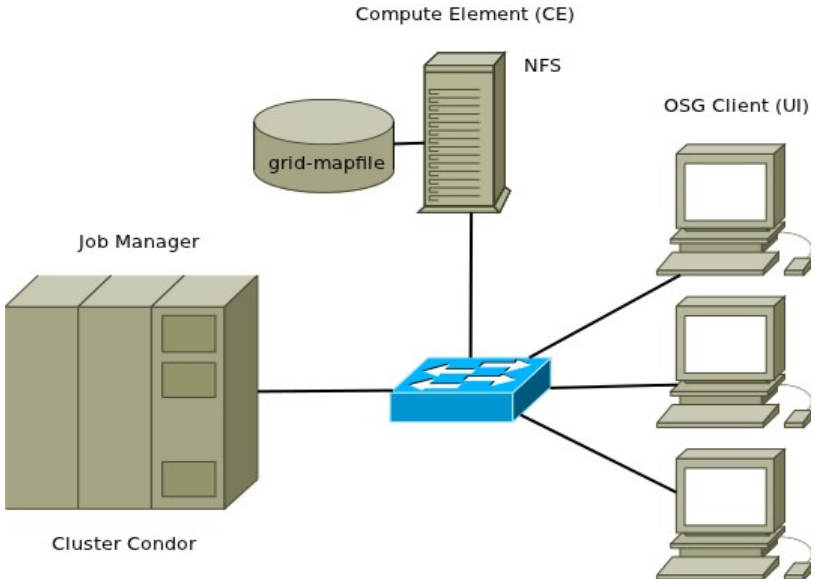
2. Para la gestión del almacenamiento de la información implementar un

sistema de archivos compartido (NFS).

3. Tantos *OSG Client* (UI) como sean necesarios según los usuarios que requieran enviar trabajos al nodo *grid*.

4. Para la gestión de usuarios utilizar el `grid-mapfile` y controlar su configuración mediante `edg-mkgridmap`.

5. Como *Manejador de Trabajos* utilizar el *cluster* Condor implementado en el nivel anterior.



Gráfica 5: Arquitectura de los componentes de un Nodo Grid pequeño

3.4.2.2.2 Nodo *grid* mediano

Para los nodos *grid* con moderada o alta cantidad de trabajos por unidad de tiempo, el proyecto sugiere instalar los siguientes componentes.

1. Un *Compute Element* (CE) que implementa GRAM.

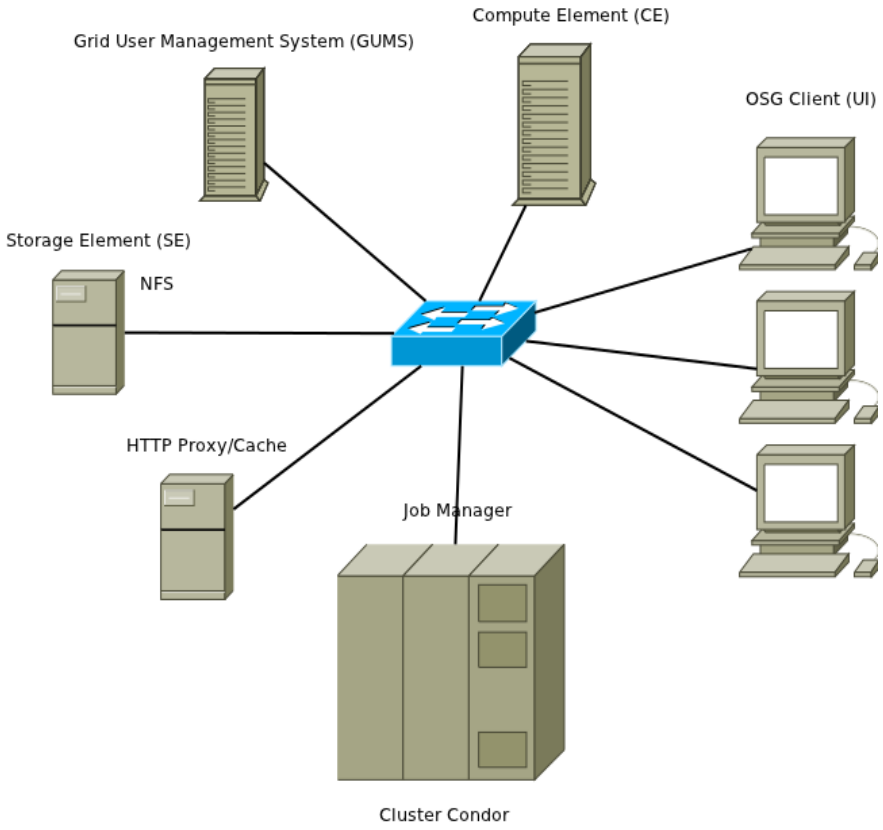
2. Un *Servidor Caché/Proxy* (Squid) que permita la optimización del ancho de banda y el control de acceso de los *nodos trabajadores* a la red.

3. Un *Storage Element* (SE) para la gestión del almacenamiento de la información. Este elemento integra el protocolo GridFTP y un sistema de archivos compartido al interior del nodo *grid*.

4. Tantos *OSG Client* (UI) como sean necesarios según los usuarios que requieran enviar trabajos al nodo *grid*.

5. Un *Grid User Management System* (GUMS) para la gestión de usuarios.

6. Como *Manejador de Trabajos* utilizar el *cluster* Condor implementado en el nivel anterior.



Gráfica 6: Arquitectura de los componentes de un Nodo Grid mediano

3.5 IMPLEMENTACIÓN DEL NIVEL DE CLUSTER

En términos generales, la implementación del nivel de *cluster* se divide en dos

partes. La primera de ellas consiste en la instalación y configuración de los componentes, y la segunda consiste en la administración y uso del mismo.

3.5.1 Instalación y configuración del *cluster*

Para la instalación y configuración del *cluster*[152] se realizaron las etapas de implementación descritas a continuación. La documentación de las etapas relacionadas con la instalación de los servidores incluye pasos adicionales para realizarse en máquinas físicas (despliegue) y máquinas virtuales bajo Virtualbox (usado en los cursos de capacitación dictados) y KVM (usado durante la etapa de pruebas y experimentación).

1. **Instalación del sistema operativo básico[153]**. En este etapa inicial se crean las máquinas virtuales que se van a utilizar o se eligen los servidor físicos donde se realizará el despliegue, y se les instala el sistema operativo básico. Para esto se eligió Scientific Linux[154] en su versión 5.4 el cual al ser derivado de RedHat Linux[155] es totalmente compatible con el software de

[152] Instalación y configuración del cluster utilizando Condor y software libre. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:instalacion_configuracion> [Consultado en octubre 10 de 2011]

[153] Instalación del sistema operativo básico. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:instalacionosbase> [Consultado en octubre 10 de 2011]

[154] Scientific Linux. Página web. [en línea] <http://www.scientificlinux.org/> [Consultado en octubre 10 de 2011]

[155] Redhat Enterprise Linux (RHEL). Página web. [en linea] <http://www.redhat.com/rhel/> [Consultado

cluster de Condor y posteriormente con el software de *grid* de Open Science Grid (OSG).

2. **Configuración básica de los servidores**. En esta etapa se realizan los ajustes iniciales a la configuración del sistema operativo para establecer el *cluster*. Estos procedimientos se realizan tanto en el servidor que actuará como elemento principal[156] (*head*) como en los servidores que actuarán como nodos trabajadores[157] (*worker nodes*).

    a) Desactivar el inicio automático del sistema de ventanas.

    b) Desactivar los servicios de *firewall* y Yum (actualización de software).

    c) Configurar el archivo de `/etc/hosts` con la información de los servidores involucrados en el *cluster*.

    d) Configurar el *hostname* de cada uno de los servidores.

    e) Configurar la dirección IP de cada uno de los servidores.

    f) Configurar los servidores de DNS que utilizarán los servidores del *cluster*.

---

en octubre 10 de 2011]

[156] Configuración básica del nodo c-head. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:configuracion_c-head> [Consultado en octubre 10 de 2011]

[157] Creación y configuración básica de los nodos trabajadores. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:configuracion_c-wn> [Consultado en octubre 10 de 2011]

3. **Configuración del servicio de Secure Shell (SSH) entre los nodos**[158]. Este ajuste establece relaciones de confianza entre los servidores del *cluster* permitiendo que los usuarios (incluyendo a `root`) autenticados en el servidor principal (*head*) puedan acceder a otros servidores del *cluster* a través del protocolo SSH sin necesidad de autenticarse nuevamente.

4. **Configuración del servicio de Network FileSystem (NFS) entre los nodos**[159]. Este ajuste permite establecer una serie de directorios únicos entre los servidores del *cluster* compartidos a través de la red local. Entre estos servidores de incluyen las cuentas de los usuarios (`/home`) y la ubicación donde se instalará el software de Condor.

5. **Configuración de las cuentas de usuario del *cluster***[160]. Durante este paso se sincronizan las cuentas de usuario del *cluster* para que todos los servidores que lo conforman cuenten con iguales usuarios con iguales UID/GID[161].

---

[158] Configuración del servicio de SSH entre los nodos. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:configuracion_ssh> [Consultado en octubre 10 de 2011]

[159] Configuración del servicio de NFS entre los nodos. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:configuracion_nfs> [Consultado en octubre 10 de 2011]

[160] Configuración de las cuentas de usuario del cluster. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:configuracion_usuarios> [Consultado en octubre 10 de 2011]

[161] Identificadores de Usuario y Grupo del sistema operativo respectivamente.

6. **Instalación del software de Condor**. Este procedimiento se realiza por separado en el servidor principal[162] del *cluster* y en los nodos trabajadores[163]. En términos generales se realizan los siguientes pasos.

    a) Establecer las ubicaciones necesarias para la instalación en los directorios compartidos con NFS.

    b) Descomprimir e instalar Condor.

    c) Ajustar la configuración de Condor.

    d) Crear los archivos de configuración para los diferentes roles de los servidores del *cluster* según la arquitectura elegida.

    e) Establecer una contraseña del *cluster* para autenticar la adición de nuevos servidores.

    f) Crear los directorios de registro centralizados.

    g) Configurar el inicio automático de Condor.

    h) Añadir los comandos de Condor al ambiente del usuario.

[162] Instalación de Condor en el nodo principal. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:instalacion_condor_head> [Consultado en octubre 10 de 2011]

[163] Instalación de Condor en los nodos trabajadores. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:instalacion_condor_wn> [Consultado en octubre 10 de 2011]

7. **Asegurar el *cluster***. Durante esta etapa final se realizan ajustes para restringir el acceso a los recursos del *cluster* desde los siguientes frentes.

    a) Configuración del *firewall*[164] con `iptables` tanto del nodo principal como de los nodos trabajadores.

    b) Restringir el acceso a los servicios según el origen de la petición[165] mediante el uso de *tcpwrapper*[166].

    c) Restringir los equipos que se pueden conectar o enviar trabajos al cluster[167] mediante ajustes a la configuración de seguridad de Condor.

Adicionalmente se documentaron dos procedimientos de configuración adicionales necesarios para el mantenimiento general del *cluster*.

8. **Agregar nuevos nodos al *cluster***.

---

[164] Configurar el firewall. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:asegurar_firewall> [Consultado en octubre 10 de 2011]

[165] Restringir el acceso a los servicios según el origen de la petición. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:asegurar_origen_servicios> [Consultado en octubre 10 de 2011]

[166] ITSO: TCP Wrappers overview. Página web. [en línea] <http://itso.iu.edu/TCP_Wrappers> [Consultado en octubre 10 de 2011]

[167] Restringir los equipos que se pueden conectar o enviar trabajos al cluster. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:asegurar_conexion_submit> [Consultado en octubre 10 de 2011]

a) Agregar nodos trabajadores[168].

b) Agregar nodos cliente para el envío de trabajos. Debido a que el software cliente del *cluster* se instala en el escritorio de los usuarios interesados en enviar trabajos, se realizó la documentación de varias alternativas de instalación de este software.

- A partir de un nodo trabajador[169].
- Utilizando Ubuntu Linux y el paquete .tgz de Condor[170].
- Utilizando Ubuntu Linux y el paquete .deb de Condor[171].
- Utilizando Scientifc Linux y el paquete .tgz de Condor[172].

9. **Permitir a los nodos trabajadores acceder a Internet a través del nodo**

[168] Agregar nuevos nodos trabajadores al cluster. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:agregar_nodos> [Consultado en octubre 10 de 2011]

[169] Convertir un nodo trabajador a un nodo de envío de trabajos. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:agregar_nodos_submitter_wn_convertir> [Consultado en octubre 10 de 2011]

[170] Agregar nodos de envío de trabajos al cluster utilizando Ubuntu Linux y el paquete .tgz. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:agregar_nodos_submitter_ubuntu_tgz> [Consultado en octubre 10 de 2011]

[171] Agregar nodos de envío de trabajos al cluster utilizando Ubuntu Linux y el paquete .deb. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:agregar_nodos_submitter_ubuntu_deb> [Consultado en octubre 10 de 2011]

[172] Agregar nodos de envío de trabajos al cluster utilizando Scientific Linux y el paquete .tgz. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:agregar_nodos_submitter_sl_tgz> [Consultado en octubre 10 de 2011]

principal[173]. Este procedimiento permite a los nodos trabajadores del *cluster* acceder a Internet a través del nodo principal mediante la configuración de este último como enrutador utilizando `iptables`.

3.5.2 Listas de verificación para la instalación y configuración

Para facilitar y fomentar la replicación de los *cluster* y la enseñanza de estos procedimientos se decidió crear listas de verificación de las principales actividades a desarrollarse permitiendo establecer la relación entre los pasos necesarios de la implementación y la documentación contenida en el wiki del proyecto.

En esta etapa de instalación y configuración del *cluster* se establecieron las siguientes listas de verificación.

1. Instalación y configuración de un Cluster con Condor[174].

[173] Permitir a los nodos trabajadores acceder a Internet a través del nodo principal. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:acceso_internet_nodos> [Consultado en octubre 10 de 2011]

[174] Lista de verificación para la instalación y configuración de un Cluster con Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:checklist:instalacion_configuracion_cluster> [Consultado en octubre 10 de 2011]

2. Agregar un nuevo nodo trabajador a un Cluster con Condor[175].

3.5.3 Solución de problemas para la instalación y configuración

Los siguientes fueron los principales problemas encontrados durante el desarrollo de la implementación y configuración del nivel de *cluster*.

1. **Valor incorrecto en el tamaño máximo de la memoria del montón (*heap*) para la máquina virtual de Java**[176]. Este problema sucede por un error de funcionamiento entre Condor y la máquina virtual de SUN para GNU/Linux.

2. **Instalación de Condor en una arquitectura incorrecta**[177]. Este error sucede cuando se instala la distribución de Condor con la arquitectura incorrecta con respecto a la del servidor.

3. **Propietario incorrecto del archivo de contraseñas**[178]. Este error sucede

[175] Lista de verificación para agregar un nuevo nodo trabajador a un Cluster con Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:checklist:agregar_nodo_trabajador> [Consultado en octubre 10 de 2011]

[176] Valor incorrecto en el tamaño máximo de la memoria del heap para la máquina virtual de Java. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:java-java_maxheap_argument> [Consultado en octubre 10 de 2011]

[177] Instalación de Condor en una arquitectura incorrecta. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:instalacion-arquitectura-incorrecta> [Consultado en octubre 10 de 2011]

[178] Propietario incorrecto del archivo de contraseñas. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:propietario-incorrecto-sec_password_file>

cuando el archivo con la información de la contraseña del *cluster* tiene un propietario o permisos incorrectos impidiendo que los nodos se agreguen al *pool* de recursos.

4. **No es posible encontrar condor_config durante el inicio automático**[179]. Este error sucede cuando no sea ha establecido correctamente la ubicación del archivo de configuración de Condor (`condor_config`) en una de las ubicaciones donde él lo espera encontrar.

3.5.4 Administración y uso del *cluster*

En ese capítulo se realiza la documentación de las principales actividades relacionadas con el uso del *cluster*, el envío de trabajos a sus diferentes universos y su correspondiente gestión.

3.5.4.1 Administración general del cluster

1. **Uso básico del *cluster***[180]. En esta sección se relacionan las principales

[Consultado en octubre 10 de 2011]

[179] No es posible encontrar condor_config durante el inicio automático. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:no-condor_config> [Consultado en octubre 10 de 2011]

[180] Uso básico del cluster. Sitio wiki del proyecto. [en línea]

actividades y comandos relacionados que un usuario del *cluster* debe conocer para interactuar con él.

- Verificar el *pool* de servidores.
- Enviar un trabajo simple.
- Verificar la cola de trabajos.
- Consultar el estado de la ejecución de un trabajo.
- Consultar la información de los trabajos completados.
- Remover trabajos de la cola de trabajos.
- Determinar el tiempo de finalización de los trabajos enviados.

2. **Uso básico del *cluster* sin un sistema de archivos compartido**[181]. En esta sección se documentan las opciones de configuración de un archivo de envío de trabajos (*submit*) de Condor necesarias para su ejecución en ambientes donde no exista un sistema de archivos compartido. Estos conceptos son especialmente importantes para la posterior ejecución de trabajos en el nivel de *grid* ya que en este nivel los trabajos se ejecutan en

<http://griduam.esencial.co/wiki/cluster:uso_basico> [Consultado en octubre 10 de 2011]

[181] Uso básico del cluster sin un sistema de archivos compartido. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:uso_sin_nfs> [Consultado en octubre 10 de 2011]

nodos (o *sites*) externos con los cuales no es técnicamente posible contar con sistemas de archivo compartidos.

3. **Determinar el estado y los posibles problemas de un trabajo en ejecución**[182]. En esta sección se analizan las herramientas que provee Condor y que permiten la consulta del estado de un trabajo que ha sido enviado al *cluster* y el motivo por el cual se presentan problemas con su ejecución.

3.5.4.2 Universos del cluster

Condor define varios universos los cuales corresponden a los ambientes de ejecución de trabajos que este soporta. La implementación de este proyecto se concentró en el uso y documentación de los tres universos principales del *cluster*.

1. **Universo Vanilla**[183]. Este universo provee el ambiente de ejecución de trabajos mas simple del *cluster*. No incluye puntos de verificación (*checkpoints*) así que es necesario repetir el trabajo completo en caso de que sea necesario migrar de nodo su ejecución. Este universo se utiliza

[182] Determinar el estado y los problemas de los trabajos enviados al cluster. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:determinar_estado_trabajo> [Consultado en octubre 10 de 2011]

[183] Enviar trabajos al universo Vanilla. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:universo_vanilla> [Consultado en octubre 10 de 2011]

para realizar trabajos cuyo ejecutable no pueda ser reenlazado (*relinked*) con las librerías de Condor como por ejemplo *scripts*, comandos del sistema operativo y aplicaciones de terceros.

2. **Universo Standard**[184]. Este universo provee un ambiente de ejecución de trabajos mas elaborado que el anterior, incluye puntos de verificación (*checkpoints*) lo que facilita la recuperación y migración de trabajos a otro nodo trabajador. Incluye algunas restricciones para la implementación de los ejecutables y estos deberán reenlazarse (*relinked*) con las librerías de Condor para aprovechar este universo.

3. **Universo Java**[185]. Este universo permite la ejecución de trabajos basados en aplicaciones Java en el *cluster* de alto desempeño.

3.5.4.3 Envío de múltiples subtrabajos al cluster

Una de las múltiples ventajas que ofrece el nivel de *cluster* es el de ejecutar múltiples subtrabajos a partir de un trabajo original, esto permite implementar de

[184] Enviar trabajos al universo Standard. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:universo_standard> [Consultado en octubre 10 de 2011]
[185] Enviar trabajos al universo Java. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:universo_java> [Consultado en octubre 10 de 2011]

manera rápida una estrategia de procesamiento por partes en la que el problema original se divide en pequeñas partes que son procesadas de manera independiente en el *cluster* y sus resultados se consolidan para obtener la respuesta final del trabajo.

En el desarrollo del proyecto se analizó el mecanismo que provee para la sintaxis del archivo de envío de trabajos (*submit*) de Condor desde dos perspectivas.

1. **Subtrabajos con argumentos diferentes**[186]. Esta aproximación permite especificar la información general del trabajo y posteriormente especificar argumentos de entrada específicos para cada uno de los subtrabajos, los cuales se utilizan principalmente para parametrizar el subconjunto de la tarea completa que cada subtrabajo va desarrollar.

2. **Subtrabajos con múltiples directorios de trabajo**[187]. Otra aproximación permite especificar directorios de trabajo independientes para cada subtrabajo evitando así cualquier tipo de conflicto o confusión. Esto es útil en los casos en que cada subtrabajo obtiene sus parámetros

[186] Envío de trabajos al cluster con múltiples argumentos. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:multiples_trabajos_varios_queue> [Consultado en octubre 10 de 2011]

[187] Envío de trabajos al cluster con múltiples directorios de trabajo. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:multiples_trabajos_varios_initdir> [Consultado en octubre 10 de 2011]

(configuración) e información a procesar desde archivos de datos.

3.5.5 Solución de problemas para la administración y uso

Los siguientes fueron los problemas principales encontrados durante el desarrollo de la administración y uso del nivel de *cluster*.

1. **Problemas con el soporte de Java en los nodos trabajadores**[188]. Este problema sucede debido a la interpretación incorrecta del parámetro `JAVA_MAXHEAP_ARGUMENT` en la configuración de Java en Condor.

2. **Failed to execute '[...]/execute/[...]/condor_exec.exe': Exec format error**[189]. Este error sucede cuando el ejecutable del trabajo que se ha enviado al *cluster* no cuenta con el permiso de ejecución o no cuenta con el correspondiente *shebang*[190] en caso de ser un *script*.

---

[188] Problemas con el soporte de Java en los nodos trabajadores (`JAVA_MAXHEAP_ARGUMENT`). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:soporte_java_maxheap> [Consultado en octubre 10 de 2011]

[189] Failed to execute '[...]/execute/[...]/condor_exec.exe': Exec format error. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/cluster:problemas:condorexec_format_error> [Consultado en octubre 10 de 2011]

[190] *Shebang* o *hashbang* corresponde con la primera línea de un *script* en UNIX y especifica el programa bajo el cual se deberá ejecutar su contenido. Por ejemplo, para ejecutar el *script* bajo un *shell* típico se deberá utilizar el siguiente: `#!/bin/sh`

3.6 IMPLEMENTACIÓN DEL NIVEL DE GRID

En términos generales, la implementación del nivel de *grid* se divide también en dos partes. La primera de ellas consiste en la instalación y configuración de los componentes, y la segunda consiste en la administración y uso del mismo.

3.6.1 Instalación y configuración del nodo *grid*

La instalación y configuración del nivel de *grid* es mas compleja que el nivel anterior ya que se cuenta con mas componentes. Por este motivo sus contenidos se han divido en las siguientes secciones.

1. Acerca de los certificados.
2. Instalación y configuración de los componentes.
3. Pruebas de conexión y uso básicas.
4. Configuración de *firewalls*.

3.6.1.1 Acerca de los certificados

Los certificados son documentos digitales basados en la arquitectura de llave pública (criptografía asimétrica) y deben ser firmados por una entidad certificadora (CA) autorizada, se utilizan para identificar de manera fidedigna a los usuarios, recursos y servicios de la .

Al utilizar la criptografía de llave pública, se generarán dos tipos de llaves: la llave pública que como su nombre lo indica debe hacerse pública y la llave privada la cual se encuentra protegida con una contraseña (*passphrase*) adicional y debe resguardarse en secreto. El cifrado de información y la verificación de autorización se realizan utilizando la llave pública mientras que el descifrado y firma digital son efectuados utilizando la llave privada.

Durante el desarrollo del proyecto se utilizaron dos tipos de certificados.

1. Los **certificados de usuario** permiten identificar a un usuario en la *grid* para definir el nivel de acceso que tenga a los recursos (como Organizaciones Virtuales por ejemplo).

Las siguientes son los procedimientos asociados a la manipulación de los certificados de usuario en un nodo *grid*.

a) **Solicitar el certificado**[191]. Este paso se subdivide en dos etapas: en la primera de ellas es necesario solicitar la aprobación de la solicitud ante Grid Colombia, para esto se debe enviar un correo con ciertas características. Una vez obtenida la respuesta favorable, se debe solicitar formalmente el certificado ante Open Science Grid. Para hacer esto es necesario llenar un formato en un sitio web.

b) **Obtener el certificado**[192]. Una vez generado el certificado por parte de OSG, el usuario puede descargarlo e instalarlo en el navegador web. Posteriormente podrá exportarlo en formato PKCS #12[193] para poder autenticarse en los sitios web de administración provistos por OSG.

c) **Instalar el certificado en el *Compute Element* (CE) o OSG Client (UI)**[194]. Una vez exportado el certificado puede ser instalado en el

[191] Solicitar el certificado de usuario. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:certificados:usuario_solicitar> [Consultado en octubre 10 de 2011]
[192] Obtener el certificado de usuario. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:certificados:usuario_obtener> [Consultado en octubre 10 de 2011]
[193] PKCS #12: Personal Information Exchange Syntax Standard. Sitio web. [en línea] <http://www.rsa.com/rsalabs/node.asp?id=2138> [Consultado en octubre 10 de 2011]
[194] Instalar el certificado de usuario en el servidor. Sitio web. [en línea]

elemento *grid* para su uso. Para hacer esto antes debe ser convertido del formato PKCS #12 al formato apropiado que separa la llave pública (`userkey.pem`) de la llave privada (`usercert.pem`).

d) **Registrar la membresía del usuario en una Organización Virtual (VO)**[195]. En este paso, el usuario propietario del certificado se registra en la Organización Virtual a la que desea pertenecer. Este procedimiento sólo debe realizarse una única vez por cada una de las VOs a las cuales se registra.

2. Un segundo tipo de certificados son los **certificados de máquina** que permiten identificar a una máquina autorizada para hacer parte de la *grid*.

Las siguientes son los procedimientos asociados a la manipulación de los certificados de máquina en un nodo *grid*.

a) **Solicitar el certificado**[196]. Para hacer esto es necesario que el responsable de la institución envíe un correo a Grid Colombia con la solicitud del certificado y la información de la correspondiente

<http://griduam.esencial.co/wiki/grid:certificados:usuario_instalar> [Consultado en octubre 10 de 2011]

[195] Registrar la membresía a la VO. Sitio web. [en línea] <http://griduam.esencial.co/wiki/grid:certificados:usuario_registrar> [Consultado en octubre 10 de 2011]

[196] Solicitar el certificado de máquina. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:certificados:host_solicitar> [Consultado en octubre 10 de 2011]

máquina.

b) **Instalar el certificado**[197]. Una vez recibido el certificado este se debe instalar en una ubicación y condiciones específicas para su correcto funcionamiento.

Con respecto a los procedimientos asociados a los certificados de usuario se incluye una lista de verificación[198] para facilitar su proceso de creación.

3.6.1.2 Instalación y configuración de los componentes

Los componentes *grid* instalados y configurados durante el desarrollo del proyecto corresponden a los mostrados a continuación, a los cuales se les crearon listas de verificación para facilitar su implementación y seguimiento.

1. Compute Element (CE)[199].

[197] Instalación de los certificados del CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:certificados_host> [Consultado en octubre 10 de 2011]

[198] Lista de verificación para gestionar los certificados. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:checklist:certificado_usuario_gestionar> [Consultado en octubre 10 de 2011]

[199] Lista de verificación para la instalación y configuración de un Compute Element. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:checklist:instalacion_configuracion_computeelement> [Consultado en octubre 10 de 2011]

2. Cliente Grid (UI)[200].

3. Grid User Management System (GUMS)[201].

3.6.1.2.1 Compute Element (CE)[202]

El CE se considera la cabeza visible del nodo *grid*, este componente realiza las tareas de gestión del nodo incluyendo el procesamiento de las solicitudes de trabajos enviadas por los usuarios y la interacción con el nivel de *cluster*.

3.6.1.2.1.1 Procedimiento

A continuación se relacionan las principales actividades asociadas al procedimiento de instalación y configuración del CE.

---

[200] Lista de verificación para la instalación y configuración de un Cliente Grid (UI). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:checklist:instalacion_configuracion_clientegrid> [Consultado en octubre 10 de 2011]

[201] Lista de verificación para la instalación y configuración del Grid User Management System (GUMS). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:checklist:instalacion_configuracion_gums> [Consultado en octubre 10 de 2011]

[202] Instalación del Compute Element (CE). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion> [Consultado en octubre 10 de 2011]

1. **Instalar y configurar los servicios de red**[203] requeridos para su funcionamiento.

2. **Instalar el gestor de paquetes**[204]. El gestor de paquetes (`pacman`) se encarga de manejar las dependencias y de descargar los archivos necesarios para suplirlas durante la instalación de los componentes *grid*.

3. **Instalar los certificados de máquina**[205].

4. **Descargar e instalar el software**[206]. En esta etapa se descargan los paquete requeridos para la instalación del CE utilizando el gestor de paquetes, se realizan las tareas de post instalación y se crea el usuario bajo el cual se ejecutarán por defecto los trabajos *grid*.

5. **Establecer los directorios compartidos**[207]. En esta etapa se definen y

---

[203] Instalación de los servicios de red del Computer Element. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:servicios> [Consultado en octubre 10 de 2011]

[204] Instalación del gestor de paquetes (pacman). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:pacman> [Consultado en octubre 10 de 2011]

[205] Instalación de los certificados del CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:certificados_host> [Consultado en octubre 10 de 2011]

[206] Descarga e instalación del software del CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:instalacion_software> [Consultado en octubre 10 de 2011]

[207] Definición de los directorios compartidos del CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:directorios_nfs> [Consultado en octubre 10 de 2011]

comparten las ubicaciones comunes entre los componentes *grid*. De estas ubicaciones se destacan un sitio común para la ubicación de las aplicaciones y otro para la ubicación de los datos generados por estas.

6. **Actualizar la configuración general**[208]. En esta etapa se realizan los ajustes de configuración del CE según la disposición del nodo *grid*.

7. **Configurar el manejo de usuarios**. En esta etapa se configura el método a través del cual el nodo *grid* dará soporte a la gestión de los usuarios. Esto se puede realizar mediante el uso del `grid-mapfile`[209] o mediante el uso de un GUMS[210].

8. **Agregar el nodo *grid* a la VO de GCEDU**[211]. En esta etapa se realizan los pasos necesarios para dar soporte a la VO GCEDU en el nodo *grid*.

---

[208] Configuración general del CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:configuracion> [Consultado en octubre 10 de 2011]

[209] Configuración del archivo gridmap en el CE. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:configuracion_gridmapfile> [Consultado en octubre 10 de 2011]

[210] Configuración CE para utilizar un GUMS específico. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:gums:configuracion_ce> [Consultado en octubre 10 de 2011]

[211] Configuración del Nodo Grid en la VO GCEDU. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:ce:instalacion:configuracion_gcedu> [Consultado en octubre 10 de 2011]

3.6.1.2.1.2 Solución de problemas

Los siguientes fueron los problemas principales encontrados durante la instalación y configuración del CE.

1. **Instalación de los paquetes *Globus-Base-Info* en una arquitectura de 64 bits**[212]. Durante la instalación se obtuvieron mensajes informativos acerca de la imposibilidad de instalar algunos paquetes debido a la plataforma, sin embargo esta situación parece ser inocua.

2. **Mensaje "*GSS authentication failure*" durante la autenticación de usuarios**[213]. Este error sucede cuando el servidor no puede identificarse de la misma manera como consta en su certificado de máquina.

3. **Mensaje "*passwd_cache::cache_uid(): getpwnam("osgedu") failed: user not found*"**[214]. Este error sucede cuando no existe el usuario bajo el

[212] Instalación de los paquetes Globus-Base-Info en una arquitectura de 64 bits. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:instalacion_paquetes_globusbaseinfo> [Consultado en octubre 10 de 2011]

[213] GSS authentication failure durante la autenticación de usuarios. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:gss_authentication_failure> [Consultado en octubre 10 de 2011]

[214] passwd_cache::cache_uid(): getpwnam("osgedu") failed: user not found. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:failed_user_not_found> [Consultado en octubre 10 de 2011]

cual se ejecutan los procesos *grid* (`osgedu` en este caso).

4. **Mensaje "*Error initializing GAHP*"**[215]. Este error sucede cuando no se encuentra instalado Java o no se pudo encontrar la información de las entidades certificadoras (CA).

5. **Problemas con el DNS reverso**[216]. Este error sucede por un problema de configuración del servicio DNS el cual no permite la consulta reversa del dominio del CE, es decir, obtener correctamente su dirección IP a partir de su FQDN[217].

3.6.1.2.2 Cliente Grid (UI)[218]

El UI es el componente cliente que permite la interacción con el nodo *grid* enviando trabajos y recibiendo su respuesta desde el escritorio del usuario investigador. La instalación de este componente requiere que se cuente con

[215] Error initializing GAHP. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:failed_initialize_gahp> [Consultado en octubre 10 de 2011]
[216] Problemas con el DNS reverso. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:reverse_dns> [Consultado en octubre 10 de 2011]
[217] Full Qualified Domain Name
[218] Instalación del Cliente de la Grid. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:cliente:instalacion> [Consultado en octubre 10 de 2011]

GNU/Linux (preferiblemente de la vertiente RedHat) como sistema operativo, sin embargo el equipo no formará parte constitutiva del nodo *grid*.

El procedimiento de instalación de este componente se describe a continuación.

1. Establecer la ubicación donde se instalará el software.
2. Descargar e instalar los archivos de la distribución de software del componente utilizando el manejador de paquetes `pacman`.
3. Realizar las tareas de post instalación.
4. Descargar los certificados asociados a las Autoridades Certificadoras (CA).
5. Activar el inicio de los servicios requeridos incluyendo a Condor (cliente).
6. Establecer la configuración de los puertos a utilizarse para interactuar con la *grid*.
7. Agregar la VO GCEDU a la configuración del cliente.
8. Ejecutar el software del cliente.

3.6.1.2.3 Grid User Management System (GUMS)[219]

El GUMS permite una gestión avanzada de los usuarios del nodo *grid* mas allá de

[219] Instalación y configuración del Grid User Management System (GUMS). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:gums:instalacion_configuracion> [Consultado en octubre 10 de 2011]

lo que la configuración del `grid-mapfile` lo permite en primera instancia. Este componente no se utiliza en la implementación del proyecto, sin embargo su instalación y configuración en el nodo fueron experimentados y documentados como avance para una posible segunda etapa del proyecto.

El procedimiento de instalación[220] de este componente se describe a continuación.

1. Establecer la ubicación donde se instalará el software.
2. Descargar e instalar los archivos de la distribución de software del componente utilizando el manejador de paquetes `pacman`.
3. Realizar las tareas de post instalación.
4. Modificar la contraseña del administrador de la base de datos (MySQL).
5. Descargar los certificados asociados a las Autoridades Certificadoras (CA).
6. Activar el inicio de los servicios requeridos.
7. Iniciar el servicio.

Para la configuración inicial del servicio[221] es necesario realizar los siguientes pasos.

---

[220] Instalación del Grid User Management System (GUMS). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:gums:instalacion> [Consultado en octubre 10 de 2011]

[221] Configuración del Grid User Management System (GUMS). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:gums:configuracion> [Consultado en octubre 10 de 2011]

1. Crear usuarios administradores del componente.
2. Actualizar la configuración por defecto incluyendo los cambios sugeridos por la plantilla de OSG.
3. Probar el funcionamiento del GUMS con la nueva configuración accediendo a él a través de su interfaz web.

Finalmente es necesario actualizar la configuración del Compute Element (CE) que va a utilizar el componente GUMS[222] en lugar del `grid-mapfile`, para hacer esto se deben realizar los siguientes pasos.

1. Editar la configuración del CE modificando el método de autorización y complementando la ubicación del servicio GUMS a utilizarse.
2. Verificar la sintaxis de la nueva configuración.
3. Reiniciar los servicios del CE para que los cambios sean tenidos en cuenta.

3.6.1.3 Pruebas de conexión y uso básicas[223]

En esta sección se realizan las pruebas de conexión y funcionamiento del nodo

[222] Configuración CE para utilizar un GUMS específico. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:gums:configuracion_ce?&#actualizar_la_configuracion> [Consultado en octubre 10 de 2011]

[223] Pruebas de conexión al Nodo Grid. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:pruebas:conexion> [Consultado en octubre 10 de 2011]

grid, estas pruebas se pueden realizar desde el mismo CE o desde un UI externo.

Las pruebas incluyen revisiones de los siguientes apartados.

a) Verificaciones internas (sólo desde el mismo CE).
b) Autenticación de usuarios.
c) Transferencia de archivos.
d) Envío de trabajos utilizando las herramientas basadas en Globus y en Condor-G.

3.6.1.4 Configuración de firewalls[224]

En esta sección se proponen reglas basadas en `iptables` para asegurar los servidores en los cuales se ejecutan los diferentes componentes del nodo *grid*.

1. Compute Element (CE)[225].
2. Cliente Grid (UI)[226].

[224] Asegurar el Nodo Grid con firewalls de equipo. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:seguridad:asegurar_firewall> [Consultado en octubre 10 de 2011]
[225] Firewall para el Compute Element (CE). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:seguridad:firewall_ce> [Consultado en octubre 10 de 2011]
[226] Firewall para el Cliente Grid (UI). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:seguridad:firewall_cliente> [Consultado en octubre 10 de 2011]

3. Grid User Management System (GUMS)[227].

3.6.2 Administración y uso del nodo *grid*

En ese capítulo se realiza la documentación de las principales actividades relacionadas con el uso del nodo *grid*, el manejo de intermediarios para los certificados de usuario, el envío de trabajos y su correspondiente gestión.

3.6.2.1 Cargar las variables de ambiente[228]

Este simple paso permite cargar las variables de ambiente relacionadas con las herramientas *grid* en el *shell* del usuario. Una vez hecho esto es posible ejecutar los comandos descritos a continuación.

Es posible que este paso sea innecesario si el administrador del nodo ha ubicado estas variables en `/etc/profile.d` previamente por lo cual son cargadas automáticamente con el registro del usuario en el sistema.

---

[227] Firewall para el Grid User Management System (GUMS). Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:seguridad:firewall_gums> [Consultado en octubre 10 de 2011]

[228] Cargar las variables de ambiente de las herramientas de línea de comando. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:admnistracion_uso:cargar_variables_ambiente> [Consultado en octubre 10 de 2011]

3.6.2.2 Gestión de los intermediarios de los certificados[229]

Los certificados de usuario son esenciales para que el usuario pueda interactuar con la *grid* ya que gracias a ellos se puede autenticar como un usuario reconocido y autorizado para enviar trabajos. Sin embargo los certificados de usuario no son utilizados directamente. Por el contrario, antes de cada sesión de interacción entre el usuario y la *grid* se debe crear un intermediario (*proxy*) del certificado de dicho usuario que se asemeja a una autorización firmada por el usuario para acceder a la *grid* en su nombre por un periodo de tiempo limitado (usualmente 12 horas).

Estos intermediarios temporales de usuario pueden ser:

1. Creados a partir un certificado de usuario válido.
2. Consultados para verificar su estado y tiempo de validez.
3. Destruidos si se desea inhabilitarlos.

[229] Gestión de los intermediarios de los certificados. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:admnistracion_uso:gestion_proxy_certificados> [Consultado en octubre 10 de 2011]

3.6.2.3 Transferencia de archivos al nodo grid[230]

La transferencia de archivos hacia y desde el nodo *grid* se realiza utilizando el protocolo GSIFTP (GridFTP) y la herramienta `globus-url-copy` provista por el middleware Globus. Esta herramienta permite el uso de diversos protocolos además de GSIFTP como HTTPS, HTTP, FTP y FILE, siendo los dos primeros en mencionarse los mas utilizados.

Este procedimiento permite llevar al nodo la información requerida para la posterior ejecución de los trabajos, como lo son los datos de entrada o los archivos del paquete de software para ser instalado, y permite también extraer la información resultante de la ejecución del trabajo para ser analizada en el escritorio del usuario que la envió a procesar.

3.6.2.4 Envío de trabajos al nodo grid[231]

En este capítulo se describen las diferentes herramientas que existen para el

---

230 Transferencia de archivos al Nodo Grid. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:admnistracion_uso:transferencia_archivos> [Consultado en octubre 10 de 2011]

231 Envío de trabajos al Nodo Grid. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:admnistracion_uso:envio_trabajos> [Consultado en octubre 10 de 2011]

envío de trabajos a un nodo *grid* por parte de un usuario. El primer grupo de estas herramientas es provisto por Globus el cual es el *middleware* de la *grid*.

a) `globusrun`. Es el comando mas genérico de los analizados en esta sección, permite la ejecución de trabajos definidos directamente en lenguaje RSL[232].

b) `globus-job-run`. Permite la ejecución síncrona de trabajos en el nodo *grid*, es decir, la ejecución de este comando termina al momento de recibir respuesta de la terminación de la ejecución de la tarea por parte del recurso grid a la cual esta fue entregada .

c) `globus-job-submit`. Permite la ejecución asíncrona o en lotes (*batch*) de trabajos en el nodo *grid*, es decir, la ejecución de este comando termina tan pronto es contactado al nodo *grid* con el trabajo requerido y es responsabilidad del usuario estar verificando el estado de la tarea y el obtener el resultado final de la misma cuando esta termine. Este comando no permite el envío de archivos locales por lo que estos deberán ser transferidos con anterioridad.

---

[232] Globus Resource Specification Language (RSL): lenguaje común para la descripción de recursos en Globus. Sitio web. [en línea] <http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html> [Consultado en octubre 10 de 2011]

La flexibilidad de los comandos descritos anteriormente es limitada, por este motivo se incluyó en el proyecto el uso de Condor-G como una capa adicional para el envío de trabajos al nodo *grid*.

Condor-G[233] permite utilizar archivos de envío de trabajos (*submit*) con una sintaxis similar a la utilizada anteriormente con Condor en el nivel de *cluster* pero haciendo referencia ahora al universo *Grid*.

Se analizaron los siguientes casos de envío de trabajos utilizando Condor-G.

a) **Envío simple**. Ejecuta un *script* o programa que ya se encuentra instalado en el nodo *grid*.

b) **Envío de un programa local**. Ejecuta un *script* o programa que se encuentra originalmente en el equipo cliente, este será transmitido hacia el nodo *grid* para su correspondiente ejecución y posteriormente el resultado final será traído de vuelta al cliente para su análisis.

c) **Con entradas y salidas**. Consiste en una versión mas elaborada y

[233] Envío de trabajos al Nodo Grid con las herramientas de Condor-G. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:admnistracion_uso:envio_trabajos_condorg> [Consultado en octubre 10 de 2011]

cercana al uso diario en la realidad, permite enviar trabajos desde el Cliente que lleven consigo los archivos de entrada que servirán como materia prima para el procesamiento de la tarea y traigan de vuelta a los archivos de salida generados durante el proceso.

d) **Con múltiples subtrabajos**. Este caso es una modificación del caso anterior en el que se procesan múltiples archivos de entrada utilizando el mismo ejecutable.

3.6.3 Solución de problemas para la administración y uso del nodo *grid*

Los siguientes fueron los problemas principales encontrados durante el desarrollo de la administración y uso del nivel de *grid*.

a) **Intentar enviar trabajos al nodo *grid* por parte de un usuario no autorizado**[234]. Este problema sucede cuando un usuario no autorizado (no existe referencia de él en el `grid-mapfile`) intenta enviar un trabajo a un

[234] Intentar enviar trabajos a la Grid por parte de un usuario no autorizado. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:envio_usuario_no_autorizado> [Consultado en octubre 10 de 2011]

nodo *grid*.

b) **Problemas de conexión con el servidor: Error Code 12**[235]. Este error sucede cuando no se puede contactar físicamente al *gatekeeper* del nodo *grid* solicitado o cuando su certificado de máquina se ha vencido.

c) **Problemas para establecer la comunicación entre el cliente y el servidor: Error Code 74**[236]. Este error sucede cuando no se puede completar exitosamente la comunicación entre el cliente y el *gatekeeper*. Usualmente es un problema de especificación de puertos y/o configuración del *firewall*.

Para la depuración de los problemas relacionados con la conexión de red, especialmente los que involucran filtros de paquetes que limitan la conectividad a través de ciertos puertos, se recomienda realizar la verificación de las conexiones utilizando `netcat`[237].

---

[235] Problemas de conexión con el servidor: Error Code 12. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:error_code_12> [Consultado en octubre 10 de 2011]

[236] Problemas para establecer la comunicación entre el cliente y el servidor: Error Code 74. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:error_code_74> [Consultado en octubre 10 de 2011]

[237] Verificar la conectividad de puertos entre Cliente y Nodo Grid. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/grid:problemas:verificar_conectividad_puertos> [Consultado en octubre 10 de 2011]

3.7 SOFTWARE INSTALADO

Como valor agregado para el proyecto, durante su ejecución se ha trabajado en la instalación, configuración y prueba de diferentes paquetes de software (especialmente relacionados con Bioinformática) que puedan tomar provecho de los niveles de *cluster* y *grid*.

Tal es el caso de **InterProScan**[238] y **Octave**[239] del cual se envían trabajos utilizando las facilidades de Condor[240].

Para el nivel de *grid* la instalación del software es mas elaborada ya que no sólo deberá instalarse en una ubicación específica sino que deberá hacerse desde *scripts* que puedan ser invocados como trabajos *grid*. Para probar este concepto se decidió realizar un piloto con el software de John The Ripper[241] y su invocación utilizando Condor para el cual se desarrollaron los siguientes pasos.

---

238 Instalación y configuración inicial de InterProScan. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/interproscan:instalacion> [Consultado en octubre 10 de 2011]

239 Instalación y configuración de Octave. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/octave:instalacion_configuracion> [Consultado en octubre 10 de 2011]

240 Envío de trabajos de Octave con Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/octave:envio_trabajos_condor> [Consultado en octubre 10 de 2011]

241 John the Ripper password cracker. Sitio web. [en linea] <http://www.openwall.com/john/> [Consultado en octubre 10 de 2011]

1. **Instalación y configuración**[242]. Esto se realiza mediante la creación de un *script* de instalación que descarga, desempaqueta e instala el software, y es enviado al nodo *grid* mediante un trabajo basado en Condor-G.

2. **Creación del archivo de contraseñas**[243]. En este paso se crea el archivo de contraseñas a verificarse el cual constituye el archivo de entrada para el trabajo *grid* próximo a crearse.

3. **Creación del archivo de envío del trabajo con Condor**[244]. En este paso se crea el archivo de envío de trabajo que encapsulará la solicitud de procesamiento para ser enviada al nodo *grid* a través de Condor-G. Este procedimiento se puede realizar manualmente[245] o de manera automatizada[246] mediante el uso de una aplicación ayudante desarrollada en Java que puede ser descargada del sitio wiki del proyecto.

---

[242] Instalación y configuración de John The Ripper. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/jtr:instalacion_configuracion> [Consultado en octubre 10 de 2011]

[243] Creación del archivo de contraseñas para John The Ripper. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/jtr:creacion_archivo_contrasenas> [Consultado en octubre 10 de 2011]

[244] Generación del archivo de trabajo de John para Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/jtr:creacion_archivo_submit> [Consultado en octubre 10 de 2011]

[245] Creación manual del archivo de especificación de trabajos de John para Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/jtr:creacion_manual> [Consultado en octubre 10 de 2011]

[246] Creación asistida del archivo de especificación de trabajos de John para Condor. Sitio wiki del proyecto. [en línea] <http://griduam.esencial.co/wiki/jtr:creacion_asistida> [Consultado en octubre 10 de 2011]

3.8 ANÁLISIS DE RESULTADOS

La elección de la metodología elegida permitió realizar la implementación del proyecto en dos grandes etapas: la correspondiente al nivel de *cluster* y posteriormente al nivel de *grid*. Este último basa su funcionalidad en el sistema de procesamiento por lotes provisto por el nivel de *cluster*, haciendo que estos niveles sean ordenados y complementarios.

El desarrollo del nivel de *cluster* se realizó en aproximadamente seis meses ya que su implementación dependió directa y exclusivamente por el desarrollador del proyecto sin necesidad de interactuar con entidades externas como sucedió posteriormente con el nivel de *grid*. Como fuentes de información se utilizaron la documentación del proyecto Condor la cual es extensa y muy completa, y se tuvo en cuenta la documentación provista por Open Science Grid para este nivel, sin embargo no fue utilizada como base directa para la implementación del nivel ya que se eligió una aproximación mas flexible basada en el paquete TGZ de Condor (en lugar del RPM) para permitir la creación de un estándar propio de ubicaciones y con ello lograr una instalación que permitiera mantener distintas versiones instaladas (aunque una sola activa) y poder intercambiar entre ellas. Así mismo el uso de esta aproximación permitió el compartir una única instalación de binarios

de Condor entre los nodos del *cluster* facilitando los procedimientos de administración y mantenimiento del software.

El uso de máquinas virtuales destinadas a la experimentación de los procedimientos que una vez probados y verificados eran implementados en los servidores en producción facilitó en gran medida el desarrollo del proyecto ya que gracias a la posibilidad de clonar máquinas virtuales fue posible disminuir los tiempos de preparación de las máquinas para la experimentación, así como disminuir muchas de las demoras y restricciones que hubiera impuesto el uso de hardware físico.

Basado en la implementación y el conocimiento generado durante esta primera etapa, el proyecto dictó un curso de tres días en el cual se instruyó a representantes de las universidades de la región en la instalación, configuración, administración y uso de *clusters* basados en Condor.

La implementación del nivel de *cluster* le permitió al grupo de investigación de la Universidad Autónoma de Manizales iniciar la experimentación con software de Bioinformática con el cual se espera apoyar la investigación y el desarrollo en general de este sector mediante el apoyo a la infraestructura de hardware y software de alto desempeño. Al respecto, actualmente la Universidad se

encuentra en proceso de establecer un convenio interinstitucional con CENICAFÉ.

El desarrollo del siguiente nivel -*grid*- requirió de una mayor interacción con entidades y proyectos externos como lo fue el caso de Grid Colombia con quien se convinieron estándares y procedimientos para garantizar la unión de los nodos implementados a la *grid* nacional. Con el proyecto Grid Colombia fue necesario además gestionar los elementos de la infraestructura de seguridad del nodo ya que ellos conformaron la entidad certificadora encargada de emitir los certificados para las máquinas y los usuarios que deseen hacer parte de la *grid* Colombiana. El tiempo de implementación de este nivel fue mas prolongado debido a que Grid Colombia también era un proyecto en desarrollo así que en múltiples ocasiones no se tenían claros o aún no se habían implementado algunos de los elementos necesarios para continuar con la implementación del nivel. Al respecto se trabajó de la mano con los integrantes de este proyecto en la definición e implementación de los elementos y procedimientos involucrados en el proyecto.

Otra entidad con la cual se debió mantener una amplia interacción fue con Open Science Grid quienes rigen la tecnología del *grid* norteamericano y sobre el cual eligió Grid Colombia basar la infraestructura *grid* del país. Con ellos se mantuvo contacto frecuente mediante la participación en los diferentes cursos presenciales que se hicieron en Colombia, la participación en foros y la lista de correo

electrónico para Latinoamérica, y el contacto directo con las principales personas involucradas en la articulación de los nodos *grid* Colombianos. Open Science Grid expone de manera libre una base de conocimiento a través de su wiki en el cual se detalla la información y procedimientos que sirvieron como base para la implementación del nivel de *grid* del proyecto, sin embargo fue necesaria una amplia revisión y experimentación de esta información para apropiarla de manera adecuada en el proyecto y extraer el conocimiento requerido para la implementación final de los nodos *grid* y generar la documentación precisa y explícita que se incluyó en el wiki del proyecto.

Con respecto al resultado final de la implementación de los dos niveles se identifica que ambos tienen como fin el mismo: ejecutar trabajos que requieren de alto desempeño. La diferencia entre la finalidad de estos dos niveles radica en el ámbito de los mismos. Mientras que el nivel de *cluster* permite utilizar las máquinas de un sitio específico, el nivel de *grid* permite trascender los límites administrativos e interactuar con recursos externos al sitio propio, permitiendo utilizar recursos de otros nodos *grid* asociados de acuerdo con las políticas y normas de uso establecidas.

Un concepto importante que surge con el nivel de *grid* es el de Organización Virtual (VO) el cual constituye una estructura lógica que agrupa los recursos, tanto

de hardware como software y humanos, de diferentes nodos *grid* en torno a fines explícitos y compartidos. Actualmente los nodos *grid* implementados pertenecen a la OSGEDU (VO común para los usuarios de OSG), GCVO/GCEDU (Grid Colombia) y GCBIO la cual se constituye en la VO para Biotecnología de Colombia y se encuentra liderada por el grupo de investigación de la Universidad Autónoma de Manizales.

La elección de un wiki como herramienta web 2.0 para gestionar la información producida por el proyecto fue la adecuada ya que no sólo permitió la edición rápida de esta información sino que permitó la relación inteligente entre los diferentes conceptos técnicos a medida en que el proyecto se iba desarrollando, constituyendo al wiki del proyecto como referencia imprescindible para la implementación de los nodos *grid* de varias universidades e instituciones de investigación en Colombia.

En términos generales, durante el desarrollo del proyecto se tuvieron los siguientes problemas críticos que impactaron negativamente el cronograma estimado.

- Durante la finalización de el nivel de *cluster* y el inicio del nivel de *grid* no se contaba con suficiente información para iniciar la etapa de experimentación

de este nivel. Fueron necesarias varias reuniones con Grid Colombia y contactos con Open Science Grid para establecer claramente los lineamientos sobre los cuales se estructuraría este nivel.

- Una vez iniciada la experimentación del nivel de *grid* y se hizo evidente la necesidad de los certificados tanto para los usuarios que van a utilizar el nodo *grid* como de los servidores que lo van a conformar, hubo un largo periodo entre la solicitud de estos certificados y la recepción de los mismos lo cual detuvo al proyecto por algunas semanas.

- Durante las etapas iniciales de la implementación del nivel de *grid* se presentaron problemas con la ejecución de los trabajos debido a discrepancias entre el *hostname* reportado por el *Compute Element* y el establecido en su certificado de máquina. Al respecto se concluyó que debe tenerse especial cuidado con la definición del archivo `/etc/hosts` (o del servicio DNS en su defecto) ya que la presencia de alias para un servidor específico puede generar este tipo de problemas.

- Durante varias semanas se presentaron problemas de conectividad con la red de la Universidad Autónoma de Manizales y algunos sitios incluyendo a los de Open Science Grid requeridos para la descarga del software *grid*.

Todo se debió problemas en la configuración del *router* del proveedor de acceso (ISP) que une a esta Universidad con Internet y RENATA. Como paliativo para esta situación fue necesaria la descarga del software desde otras redes y realizar la copia física de estos datos en el *Compute Element* de la Universidad mediante dispositivos USB o transmisiones SSH que tomaban varias horas debido a su gran tamaño.

- Durante la realización de las pruebas finales de conectividad y ejecución remota de trabajos en el nivel de *grid* se identificaron problemas en la resolución del *DNS inverso* para los dominios de los *Compute Element* en ambas universidades, lo que impedía que se enviaran y recibieran trabajos remotos desde y hacía estos. Una vez solucionado este problema por las correspondientes dependencias de Informática de las Universidades, la conectividad remota de los nodos fue completa.

- Durante las pruebas de ejecución de trabajos *grid* en los nodos se identificó que los trabajos enviados utilizando las herramientas de Condor-G entre el UI y el CE eran exitosos pero que si se envían desde el mismo CE ya no lo son. Aparentemente el trabajo es recibido pero queda siendo infinitamente procesado por la cola de trabajos de Condor. Se realizaron pruebas con los diferentes nodos *grid* funcionales de otras universidades y se obtuvo el

mismo resultado. Se contactó a Grid Colombia al respecto quienes consideraron que este comportamiento era esperado.

4. CONCLUSIONES

Tanto el nivel de *cluster* como el de *grid* se especializan en la ejecución de trabajos de alto desempeño, la diferencia entre estos es el ámbito de los recursos utilizados así que los recursos gestionados por el *cluster* son locales mientras que los recursos gestionados por la *grid* pueden incluir remotos además de los locales, trascendiendo las fronteras administrativas de las diferentes organizaciones que lo conformen.

La implementación del nivel de *cluster* del proyecto se basó en el software Condor desarrollado por la Universidad de Wisconsin, sin embargo existen otros sistemas manejadores de lotes como LSF, PBS y SGE que pueden ser integrados de manera paralela en la infraestructura de los nodos y eventualmente integrados al nivel de *grid* si se cuenta con el soporte apropiado por parte de OSG. De manera similar, el nivel de *grid* fue desarrollado utilizando los lineamientos de OSG debido a que Grid Colombia lo adoptó como *grid* nacional, sin embargo es posible también incorporar los nodos a la *grid* europea mediante la instalación y configuración del software adecuado.

No todo el software puede ser ejecutado en un *cluster* o en una *grid*, el software

deberá haber sido desarrollado de manera apropiada para ser ejecutado en este tipo de ambientes. Al respecto se identifican tres grupos funcionales de aplicaciones.

1. **Software no apto**. En este grupo se encuentran las aplicaciones que no están habilitadas para sacar provecho de esta infraestructura. De este grupo se destacan las aplicaciones que cuentan con una interfaz de usuario a través de la cual requieren interacción para lograr su cometido, por ejemplo las aplicaciones de ofimática.

2. **Software que puede dividir su entrada**. En este grupo se encuentran las aplicaciones que no fueron desarrolladas explícitamente para trabajar en ninguno de los niveles pero que pueden procesar la información de entrada por partes que posteriormente puede ser consolidada para obtener el resultado final. Esta información, así como todos sus parámetros de configuración deberán ser transmitidos a través de la línea de comando. Este tipo de aplicaciones puede aprovechar la ejecución de tareas por Condor del nivel de *cluster* y por ende el nivel de *grid*.

3. **Software diseñado para *cluster*/*grid***. En este grupo se encuentran las aplicaciones que fueron diseñadas explícitamente para aprovechar estos

niveles. Son implementadas utilizando tecnologías como MPI o Globus para interactuar con la infraestructura distribuida.

De manera análoga, no toda la ejecución del software en un *cluster* o *grid* obtendrá una ventaja significativa en su desempeño ya que las capas adicionales de *middleware* que deben ejecutarse imponen una sobrecarga significativa que en ciertas condiciones puede aumentar el consumo de recursos y tiempo con respecto a realizar la misma ejecución en un equipo independiente. Los contextos con mayores probabilidades de obtener una ganancia al ser ejecutados en un *cluster* o *grid* son los que tienen altos requerimientos de hardware como memoria, procesamiento o espacio en disco, y son susceptibles de dividir una tarea grande en múltiples tareas pequeñas sin dependencias que pueden ser ejecutadas de manera simultánea en los nodos trabajadores.

A esta última clasificación pertenecen la mayoría de las herramientas relacionadas con la Bioinformática y que han sido analizadas al interior del grupo de investigación de la Universidad Autónoma de Manizales, por este motivo se está estudiando la posibilidad de llevar a los nodos a estas aplicaciones susceptibles de configurarse o adaptarse a las tecnologías existentes, aprovechando de manera óptima los niveles de *cluster* y *grid* implementados.

5. RECOMENDACIONES

Las siguientes son las recomendaciones que se realizan en torno a este proyecto que servirán para ampliarlo y fortalecerlo en pro de la comunidad académica y de investigación de la región.

1. La utilidad de los nodos se mide en función de su capacidad (procesamiento y almacenamiento), por este motivo es importante escalarlo continuamente mediante la adición de nuevos nodos trabajadores o la actualización de los existentes.

2. Al igual que en todos los proyectos de informática, la seguridad es un tema primordial que debe ser tenido en cuenta con especial cuidado. A pesar de que en este proyecto se tuvieron en cuenta medidas básicas para resguardar la seguridad de los servidores es posible fortalecer aún mas sus mecanismos pasivos y activos para garantizar al máximo posible el correcto funcionamiento y privacidad de la información que en ellos se procesa. De manera análoga, es muy conveniente involucrar también herramientas para la administración y la medición del desempeño de los recursos involucrados en los nodos.

3. Durante los últimos meses del proyecto se participó en la creación de la Organización Virtual en Bioinformática la cual hará uso extensivo de los nodos implementados. Al respecto es necesario fortalecer esta VO y en el corto plazo establecer, instalar y experimentar con el software requerido por los investigadores de manera que aprovechen los niveles de *cluster* y *grid* ya existentes.

4. Con respecto al software se propone el desarrollo de nuevas aplicaciones basadas en diferentes *middlewares* y la adaptación de aplicaciones existentes para que tomen ventaja del paralelismo presente en la infraestructura, así como la adaptación de aplicaciones existentes para que saquen provecho del *cluster* y la *grid*.

5. El desarrollo del proyecto incluyó la implementación de Condor en el nivel de *cluster*, sin embargo es posible integrar otros manejadores de colas (*batch systems*) a la infraestructura para complementarla y apoyar el uso y desarrollo de otras aplicaciones.

6. El proyecto cubre además de los temas de instalación y configuración, una parte importante de la administración y uso de la infraestructura

implementada, sin embargo esta es una mínima parte si se compara con las características potencialmente aprovechables del software que ya se encuentra instalado. Por este motivo, se propone profundizar en el aprendizaje, documentación y difusión que estas tecnologías brindan para la academia y la investigación.

7. La implementación del proyecto aborda el problema desde la visión de la ingeniería de sistemas ya que el objetivo principal es la preparación de la infraestructura de hardware y software de alto desempeño. Una vez logrado ese objetivo se propone tener en cuenta también la visión del usuario investigador que va a utilizar esa infraestructura para desarrollar su trabajo técnico ajeno a la informática. Para este tipo de usuarios se deberán desarrollar interfaces de interacción con los nodos que le permitan realizar su cometido evitándole retrasos y dificultades propias de la tecnología y que no pertenecen a su área del saber.

8. El siguiente paso a las tecnologías de *cluster* y *grid* ya desarrolladas durante este proyecto, corresponde con la búsqueda de integración de los nodos a la nube (*cloud*).

9. El resultado del proyecto así como las tecnologías sobre las cuales se

fundamente, se encuentran en profundos y continuos cambios constantes. Por ejemplo, en estos momentos OSG se encuentra migrando los repositorios de paquetes del nivel de *grid* de `pacman` a RPM y muy probablemente actualice la versión de Globus actual (versión 2) a la 4 o 5, o inclusive migre a utilizar CREAM como *Compute Element*. Por este motivo es necesario que las instituciones inviertan recursos en la actualización y desarrollo continuo de los nodos. Se recomienda que cada universidad cuente con al menos un responsable administrativo y un responsable técnico con tiempo y recursos suficientes para mantener los nodos funcionales y a la vanguardia tecnológica, pues al fin y al cabo esto es lo que diferencia a la *grid* de simples equipos interconectados.

10. Como consecuencia del punto anterior, es necesario mantener actualizado el sitio wiki con los cambios y procedimientos realizados para mantener la dinámica del conocimiento generado por el proyecto y que ha sido reconocido y aprovechado por otras instituciones académicas y de investigación en el país.

11. Finalmente se invita a la comunidad académica y de investigación en general a desarrollar proyectos en los diferentes campos del conocimiento que tomen ventaja de la infraestructura de hardware y software provista por

este proyecto y fruto de dos años de intensa implementación.

BIBLIOGRAFÍA

CASTRO Harold, Chacón Jorge, Díaz César, González Enrique, Zuluaga Jorge. "COLOMBIA: SOPORTE PARA INVESTIGACIONES DE AVANZADA" [En línea] <http://www.unired.edu.co/eventos/unired/2009/noticias/%20colombia.pdf> [consultado en 10 de abril de 2010]

COLOURIS, George. Sistemas distribuidos: conceptos y diseño. 3Era. edición. Addison Wesley, 2001. 744 p. ISBN 8478290494.

DÍAZ Cesar. "Fuerza de trabajo Iniciativa de *grid* Nacional – Colombia" [En línea] <http://www.saber.ula.ve/bitstream/123456789/16017/1/cdiaz.pdf> [consultado en 10 de mayo de 2010]

FERREIRA *et al*, Introduction to Grid Computing with Globus. Redbooks (IBM). 2003.

FOSTER, A Globus Toolkit Primer, Or, Everything You Wanted to Know about Globus, but Were Afraid To Ask. 2005. [En línea] <http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf> [consultado en 1 de mayo de 2010]

FOSTER, C. Kesselman. "The : Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999. Capítulo 2. [En línea] <http://www.globus.org/alliance/publications/papers/chapter2.pdf> [consultado en 1 de mayo de 2010]

FOSTER, C. Kesselman. "The Physiology of the : An Open Grid Services Architecture for Distributed Systems Integration.". Open Grid Service Infrastructure WG, Global Forum. 2002. [En línea] <http://www.globus.org/alliance/publications/papers/ogsa.pdf> [consultado en 1 de mayo de 2010]

GONZÁLEZ, Enrique. " COLOMBIA: COMPUTACIÓN DE ALTO DESEMPEÑO PARA EL DESARROLLO CIENTÍFICO Y TECNOLÓGICO DE COLOMBIA". [En línea] <http://www.cudi.edu.mx/aplicaciones/dias_cudi/09_10_13/-Colombia-2009.pdf> [consultado en 1 de mayo de 2010]

LEZAMA LUGO, Ada. Modelado de dispositivos para un sistema de seguridad implementando tecnología Jini. México, 2001, Capítulo 2. [En línea] <http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/lezama_l_a/capitulo_2.html> [consultado en 1 de mayo de 2010]

MOORE, Gordon. "Cramming more components onto integrated circuits". En Electronics Magazine, Página 4. (1965). [en línea] <ftp://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf> [consultado el 10 de abril de 2010]

RENATA, Quienes somos ? [En línea] <http://www.renata.edu.co/index.php/quienes-somos-identidad-y-objetivos-de-renata.html> [citado en 10 de abril de 2010]

RENATA, "Infraestructura de RENATA". [En línea] <http://www.renata.edu.co/index.php/infraestructura.html> [consultado en 10 de abril de 2010]

TANENBAUM, Andrews. Sistemas Distribuidos: principios y paradigmas. 1Era. Edición. Prentice Hall, 2008. 704 p. ISBN 9789702612803.

OSG, Blueprint. [En línea] Marzo 4 de 2011. <http://osg-docdb.opensciencegrid.org/0000/000018/012/OSG%20Blueprint%20v2.0.pdf> [consultado en 10 de octubre de 2011]

OSG, Open Science Grid Architecture . [En línea] Febrero 10 de 2009. <http://osg-docdb.opensciencegrid.org/0009/000966/001/OSG-Architecture-V2.pdf> [consultado en 10 de octubre de 2011]

OSG, Technical Documentation. [En línea] <https://twiki..iu.edu/twiki/bin/view/Documentation/WebHome> [consultado en 10 de abril de 2010]

PORDES et al, The Open Science Grid Status and Architecture. 2008. [En línea] <http://iopscience.iop.org/1742-6596/119/5/052028> [consultado en 10 de octubre de 2011]

WISCONSIN, Universidad de. Condor Manuals. [En línea] <http://www.cs.wisc.edu/condor/manual/> [consultado el 15 de octubre de 2011]

ANEXOS

ANEXO I: PRUEBAS FUNCIONALES DE LOS NODOS

Nodo Universidad Autónoma de Manizales (UAM)

Nivel de cluster

Verificar conectividad con el nodo principal del *cluster*.

```
$ ping -c 5 ce.autonoma.edu.co

     PING ce.autonoma.edu.co (200.21.104.84) 56(84) bytes of
     data.
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_req=1 ttl=57 time=321 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_req=2 ttl=57 time=351 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_req=3 ttl=57 time=378 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_req=4 ttl=57 time=392 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_req=5 ttl=57 time=467 ms

     --- ce.autonoma.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4000ms
     rtt min/avg/max/mdev = 321.494/382.164/467.374/48.978 ms
```

Verificar conectividad SSH con el nodo principal del *cluster*

```
$ ssh jimezam@ce.autonoma.edu.co

     jimezam@ce.autonoma.edu.co's password:
```

```
     Last login: Tue Aug  2 10:58:49 2011 from
     190.254.142.132

[jimezam@ce ~]$ uname -a

     Linux ce.autonoma.edu.co 2.6.18-238.1.1.el5xen #1 SMP
     Tue Jan 18 19:15:11 EST 2011 x86_64 x86_64 x86_64
     GNU/Linux
```

Verificar conectividad con los nodos trabajadores del *cluster*.

```
$ ping -c 5 nodo1

     PING UAMCLNodo1.autonoma.edu.co (190.15.30.201) 56(84)
     bytes of data.
     64 bytes from UAMCLNodo1.autonoma.edu.co
     (190.15.30.201): icmp_seq=1 ttl=64 time=0.229 ms
     64 bytes from UAMCLNodo1.autonoma.edu.co
     (190.15.30.201): icmp_seq=2 ttl=64 time=0.214 ms
     64 bytes from UAMCLNodo1.autonoma.edu.co
     (190.15.30.201): icmp_seq=3 ttl=64 time=0.211 ms
     64 bytes from UAMCLNodo1.autonoma.edu.co
     (190.15.30.201): icmp_seq=4 ttl=64 time=0.215 ms
     64 bytes from UAMCLNodo1.autonoma.edu.co
     (190.15.30.201): icmp_seq=5 ttl=64 time=0.218 ms

     --- UAMCLNodo1.autonoma.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4000ms
     rtt min/avg/max/mdev = 0.211/0.217/0.229/0.014 ms

$ ping -c 5 nodo2

     PING UAMCLNodo2.autonoma.edu.co (190.15.30.202) 56(84)
     bytes of data.
     64 bytes from UAMCLNodo2.autonoma.edu.co
     (190.15.30.202): icmp_seq=1 ttl=64 time=0.227 ms
     64 bytes from UAMCLNodo2.autonoma.edu.co
     (190.15.30.202): icmp_seq=2 ttl=64 time=0.214 ms
     64 bytes from UAMCLNodo2.autonoma.edu.co
     (190.15.30.202): icmp_seq=3 ttl=64 time=0.204 ms
```

```
     64 bytes from UAMCLNodo2.autonoma.edu.co
     (190.15.30.202): icmp_seq=4 ttl=64 time=0.216 ms
     64 bytes from UAMCLNodo2.autonoma.edu.co
     (190.15.30.202): icmp_seq=5 ttl=64 time=0.212 ms

     --- UAMCLNodo2.autonoma.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4001ms
     rtt min/avg/max/mdev = 0.204/0.214/0.227/0.017 ms

$ ping -c 5 nodo3

     PING UAMCLNodo3.autonoma.edu.co (190.15.30.203) 56(84)
     bytes of data.
     64 bytes from UAMCLNodo3.autonoma.edu.co
     (190.15.30.203): icmp_seq=1 ttl=64 time=0.234 ms
     64 bytes from UAMCLNodo3.autonoma.edu.co
     (190.15.30.203): icmp_seq=2 ttl=64 time=0.221 ms
     64 bytes from UAMCLNodo3.autonoma.edu.co
     (190.15.30.203): icmp_seq=3 ttl=64 time=0.230 ms
     64 bytes from UAMCLNodo3.autonoma.edu.co
     (190.15.30.203): icmp_seq=4 ttl=64 time=0.660 ms
     64 bytes from UAMCLNodo3.autonoma.edu.co
     (190.15.30.203): icmp_seq=5 ttl=64 time=0.217 ms

     --- UAMCLNodo3.autonoma.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4004ms
     rtt min/avg/max/mdev = 0.217/0.312/0.660/0.174 ms
```

Verificar conectividad SSH con los nodos trabajadores del *cluster*

```
# ssh nodo1

     Last login: Thu Jul 28 14:47:27 2011

[root@UAMCLNodo1 ~]# uname -a

     Linux UAMCLNodo1.autonoma.edu.co 2.6.18-164.15.1.el5xen
     #1 SMP Tue Mar 16 19:05:55 EDT 2010 x86_64 x86_64 x86_64
     GNU/Linux
```

```
# ssh nodo2

     Last login: Wed Jul 27 16:43:42 2011 from
     ce.autonoma.edu.co

[root@UAMCLNodo2 ~]# uname -a

     Linux UAMCLNodo2.autonoma.edu.co 2.6.18-194.3.1.el5xen
     #1 SMP Fri May 7 02:05:32 EDT 2010 x86_64 x86_64 x86_64
     GNU/Linux

# ssh nodo3

     Last login: Thu Jul 28 17:57:17 2011 from
     ce.autonoma.edu.co

[root@UAMCLNodo3 ~]# uname -a

     Linux UAMCLNodo3.autonoma.edu.co 2.6.18-164.15.1.el5 #1
     SMP Tue Mar 16 18:44:51 EDT 2010 x86_64 x86_64 x86_64
     GNU/Linux
```

Verificar el estado del sistema de archivos compartido (NFS) en el *cluster*

```
# exportfs -v

     /exports/interproscan
     190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
     ck,anonuid=65534,anongid=65534)
     /exports/condor
     190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
     ck,anonuid=65534,anongid=65534)
     /exports/octave
     190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
     ck,anonuid=65534,anongid=65534)
     /exports/blast
     190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
     ck,anonuid=65534,anongid=65534)
     /exports/home
     190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
```

```
      ck,anonuid=65534,anongid=65534)
      /exports/osg
      190.15.30.192/28(rw,wdelay,no_root_squash,no_subtree_che
      ck,anonuid=65534,anongid=65534)

UAMCLNodo1# mount | grep UAMCLMaestro

      UAMCLMaestro:/exports/condor on /nfs/condor type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/interproscan on /nfs/interproscan
      type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/blast on /nfs/blast type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/octave on /nfs/octave type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/home on /nfs/home type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/osg on /nfs/osg type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)

UAMCLNodo2# mount | grep UAMCLMaestro

      UAMCLMaestro:/exports/condor on /nfs/condor type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/interproscan on /nfs/interproscan
      type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/blast on /nfs/blast type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
      UAMCLMaestro:/exports/octave on /nfs/octave type nfs
      (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
      .15.30.200)
```

```
     UAMCLMaestro:/exports/home on /nfs/home type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
     UAMCLMaestro:/exports/osg on /nfs/osg type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)

UAMCLNodo3# mount | grep UAMCLMaestro

     UAMCLMaestro:/exports/condor on /nfs/condor type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
     UAMCLMaestro:/exports/blast on /nfs/blast type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
     UAMCLMaestro:/exports/interproscan on /nfs/interproscan
     type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
     UAMCLMaestro:/exports/home on /nfs/home type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
     UAMCLMaestro:/exports/osg on /nfs/osg type nfs
     (rw,noatime,udp,bg,intr,wsize=32768,rsize=32768,addr=190
     .15.30.200)
```

Verificar el pool de nodos trabajadores del *cluster* Condor

```
# condor_status

     Name               OpSys      Arch   State     Activity
     LoadAv Mem   ActvtyTime

     slot1@UAMCLNodo1.a LINUX      X86_64 Unclaimed Idle
     1124.000  3912 26+18:59:14
     slot2@UAMCLNodo1.a LINUX      X86_64 Unclaimed Idle
     1.000  3912  0+02:50:06
     slot3@UAMCLNodo1.a LINUX      X86_64 Unclaimed Idle
     1.000  3912 26+19:07:37
     slot4@UAMCLNodo1.a LINUX      X86_64 Unclaimed Idle
     1.000  3912 26+19:07:37
```

```
slot1@UAMCLNodo2.a LINUX      X86_64 Unclaimed Idle
1.000  3912  0+02:50:05
slot2@UAMCLNodo2.a LINUX      X86_64 Unclaimed Idle
1.000  3912 26+19:07:23
slot3@UAMCLNodo2.a LINUX      X86_64 Unclaimed Idle
1.000  3912 26+19:07:24
slot4@UAMCLNodo2.a LINUX      X86_64 Unclaimed Idle
1287.010  3912 26+19:07:25
slot1@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  0+03:05:05
slot2@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:22
slot3@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:23
slot4@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:24
slot5@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:25
slot6@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:26
slot7@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:26
slot8@UAMCLNodo3.a LINUX      X86_64 Unclaimed Idle
0.000  4022  4+19:08:19
                     Total Owner Claimed Unclaimed
Matched Preempting Backfill

        X86_64/LINUX    16     0       0        16
0          0        0

               Total    16     0       0        16
0          0        0
```

Verificar el soporte para Java en los nodos trabajadores del *cluster*

```
# condor_status -java

     Name               JavaVendor Ver    State     Activity
     LoadAv Mem   ActvtyTime

     slot1@UAMCLNodo1.a Sun Micros 1.6.0_ Unclaimed Idle
```

```
1124.000  3912 26+19:34:14
slot2@UAMCLNodo1.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912  0+03:25:06
slot3@UAMCLNodo1.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912 26+19:42:37
slot4@UAMCLNodo1.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912 26+19:42:37
slot1@UAMCLNodo2.a Sun Micros 1.6.0_ Unclaimed Idle
1286.220  3912  0+00:00:02
slot2@UAMCLNodo2.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912 26+19:42:23
slot3@UAMCLNodo2.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912 26+19:42:24
slot4@UAMCLNodo2.a Sun Micros 1.6.0_ Unclaimed Idle
1.000  3912 26+19:42:25
slot1@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  0+03:45:05
slot2@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:22
slot3@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:23
slot4@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:24
slot5@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:25
slot6@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:26
slot7@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:26
slot8@UAMCLNodo3.a Sun Micros 1.6.0_ Unclaimed Idle
0.000  4022  4+19:48:19
                     Total Owner Claimed Unclaimed
Matched Preempting Backfill

        X86_64/LINUX    16     0       0        16
0          0        0

               Total    16     0       0        16
0          0        0
```

Verificar la cola de trabajos del *cluster* Condor

```
# condor_q

     -- Submitter: ce.autonoma.edu.co :
     <190.15.30.200:9486> : ce.autonoma.edu.co
      ID      OWNER            SUBMITTED     RUN_TIME ST PRI
     SIZE CMD

     0 jobs; 0 idle, 0 running, 0 held
```

Probar el envío de trabajos al universo Vanilla del *cluster*

```
$ vi script.sh

     #!/bin/sh
     /bin/hostname
     /usr/bin/id

$ chmod +x script.sh

$ vi demo_vanilla.submit

     executable              = script.sh
     universe                = vanilla
     log                     = _demo_vanilla.log
     output                  = _demo_vanilla.out
     error                   = _demo_vanilla.err
     should_transfer_files   = YES
     when_to_transfer_output = ON_EXIT
     queue

$ condor_submit demo_vanilla.submit
     Submitting job(s).
     Logging submit event(s).
     1 job(s) submitted to cluster 2397.

$ ls -l

     total 16
```

```
     -rw-r--r-- 1 jimezam jimezam   0 Aug  2 11:31
     _demo_vanilla.err
     -rw-rw-r-- 1 jimezam jimezam 611 Aug  2 11:31
     _demo_vanilla.log
     -rw-r--r-- 1 jimezam jimezam  81 Aug  2 11:31
     _demo_vanilla.out
     -rw-rw-r-- 1 jimezam jimezam 273 Aug  2 11:30
     demo_vanilla.submit
     -rwxrwxr-x 1 jimezam jimezam  37 Aug  2 11:26 script.sh

$ cat _demo_vanilla.log

     000 (2397.000.000) 08/02 11:31:19 Job submitted from
     host: <190.15.30.200:9486>
     ...
     001 (2397.000.000) 08/02 11:31:33 Job executing on host:
     <190.15.30.202:9960>
     ...
     005 (2397.000.000) 08/02 11:31:45 Job terminated.
          (1) Normal termination (return value 0)
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          81  -  Run Bytes Sent By Job
          37  -  Run Bytes Received By Job
          81  -  Total Bytes Sent By Job
          37  -  Total Bytes Received By Job
     ...

$ cat _demo_vanilla.out

     UAMCLNodo2.autonoma.edu.co
     uid=509(jimezam) gid=509(jimezam) groups=509(jimezam)
```

Probar el envío de trabajos al universo Standard del *cluster*

```
$ vi HelloCluster.c

     #include <stdio.h>
     #include <unistd.h>

     int main(void)
     {
          char hn[256];
          int control = gethostname(hn, sizeof(hn));

          if(control == -1)
          {
               perror("gethostname");
               return 1;
          }

          printf("Hello cluster, my name is: %s.\n\n", hn);

          return 0;
     }

$ gcc -c HelloCluster.c

$ condor_compile gcc HelloCluster.o -o HelloCluster

     LINKING FOR CONDOR : /usr/bin/ld -L/opt/condor/lib
     -Bstatic --eh-frame-hdr -m elf_x86_64 --hash-style=gnu
     -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o
     HelloCluster /opt/condor/lib/condor_rt0.o
     /usr/lib/gcc/x86_64-redhat-
     linux/4.1.2/../../../../lib64/crti.o
     /usr/lib/gcc/x86_64-redhat-linux/4.1.2/crtbeginT.o
     -L/opt/condor/lib -L/usr/lib/gcc/x86_64-redhat-
     linux/4.1.2 -L/usr/lib/gcc/x86_64-redhat-linux/4.1.2
     -L/usr/lib/gcc/x86_64-redhat-
     linux/4.1.2/../../../../lib64 -L/lib/../lib64
     -L/usr/lib/../lib64 HelloCluster.o
     /opt/condor/lib/libcondorsyscall.a
     /opt/condor/lib/libcondor_z.a
     /opt/condor/lib/libcomp_libstdc++.a
     /opt/condor/lib/libcomp_libgcc.a
     /opt/condor/lib/libcomp_libgcc_eh.a --as-needed --no-as-
```

```
      needed -lcondor_c -lcondor_nss_files -lcondor_nss_dns
      -lcondor_resolv -lcondor_c -lcondor_nss_files
      -lcondor_nss_dns -lcondor_resolv -lcondor_c
      /opt/condor/lib/libcomp_libgcc.a
      /opt/condor/lib/libcomp_libgcc_eh.a --as-needed --no-as-
      needed /usr/lib/gcc/x86_64-redhat-
      linux/4.1.2/crtend.o /usr/lib/gcc/x86_64-redhat-
      linux/4.1.2/../../../../lib64/crtn.o
      /opt/condor/lib/libcondorsyscall.a(condor_file_agent.o):
      In function `CondorFileAgent::open(char const*, int,
      int)':
      /
      home/condor/execute/dir_9339/userdir/src/condor_ckpt/con
      dor_file_agent.cpp:106: warning: the use of `tmpnam' is
      dangerous, better use `mkstemp'
      /opt/condor/lib/libcondorsyscall.a(special_stubs.o): In
      function `condor_gethostbyaddr':
      /
      home/condor/execute/dir_9339/userdir/src/condor_syscall_
      lib/special_stubs.cpp:201: warning: Using
      'gethostbyaddr' in statically linked applications
      requires at runtime the shared libraries from the glibc
      version used for linking
      /opt/condor/lib/libcondorsyscall.a(special_stubs.o): In
      function `condor_gethostbyname':
      /
      home/condor/execute/dir_9339/userdir/src/condor_syscall_
      lib/special_stubs.cpp:194: warning: Using
      'gethostbyname' in statically linked applications
      requires at runtime the shared libraries from the glibc
      version used for linking
      /opt/condor/lib/libcondorsyscall.a(sock.o): In function
      `Sock::getportbyserv(char*)':
      /
      home/condor/execute/dir_9339/userdir/src/condor_io/sock.
      cpp:233: warning: Using 'getservbyname' in statically
      linked applications requires at runtime the shared
      libraries from the glibc version used for linking

$ vi HelloCluster.submit

      Executable     = HelloCluster
```

```
    Universe       = standard
    Output         = _HelloCluster.out
    Error          = _HelloCluster.err
    Log            = _HelloCluster.log
    Queue

$ condor_submit HelloCluster.submit

    Submitting job(s).
    Logging submit event(s).
    1 job(s) submitted to cluster 2398.

$ ls -l

    total 5368
    -rwxrwxr-x 1 jimezam jimezam 5461107 Aug  2 11:45
    HelloCluster
    -rw-rw-r-- 1 jimezam jimezam     323 Aug  2 11:35
    HelloCluster.c
    -rw-rw-r-- 1 jimezam jimezam       0 Aug  2 11:47
    _HelloCluster.err
    -rw-rw-r-- 1 jimezam jimezam     625 Aug  2 11:48
    _HelloCluster.log
    -rw-rw-r-- 1 jimezam jimezam    1760 Aug  2 11:35
    HelloCluster.o
    -rw-rw-r-- 1 jimezam jimezam      56 Aug  2 11:48
    _HelloCluster.out
    -rw-rw-r-- 1 jimezam jimezam     167 Aug  2 11:47
    HelloCluster.submit

$ cat _HelloCluster.log

    000 (2398.000.000) 08/02 11:47:59 Job submitted from
    host: <190.15.30.200:9486>
    ...
    001 (2398.000.000) 08/02 11:48:31 Job executing on host:
    <190.15.30.202:9960>
    ...
    005 (2398.000.000) 08/02 11:48:42 Job terminated.
         (1) Normal termination (return value 0)
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
    Usage
```

```
          Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          1053  -  Run Bytes Sent By Job
          5462125  -  Run Bytes Received By Job
          1053  -  Total Bytes Sent By Job
          5462125  -  Total Bytes Received By Job
     ...

$ cat _HelloCluster.out

     Hello cluster, my name is: UAMCLNodo2.autonoma.edu.co.
```

Probar el envío de trabajos al universo Java del *cluster*

```
$ vi HelloClusterJava.java

     public class HelloClusterJava
     {
          public static void main(String args[])
          {
               String hostname = "*** desconocido *** ";

               try
               {
                    java.net.InetAddress localMachine =

          java.net.InetAddress.getLocalHost();
                    hostname = localMachine.getHostName();
               }
               catch (java.net.UnknownHostException e)
               {
                    hostname = e.getMessage();
               }

               System.out.println("Hello cluster, my name is:
     " +                                 hostname + ".\n\n");
          }
```

```
    }

$ javac HelloClusterJava.java

$ vi HelloClusterJava.submit

    executable = HelloClusterJava.class
    arguments  = HelloClusterJava
    universe   = java
    log        = _HelloClusterJava.log
    output     = _HelloClusterJava.out
    error      = _HelloClusterJava.err
    queue

$ condor_submit HelloClusterJava.submit

$ ls -l

    total 20
    -rw-rw-r-- 1 jimezam jimezam 959 Aug  2 11:54
    HelloClusterJava.class
    -rw-rw-r-- 1 jimezam jimezam   0 Aug  2 12:00
    _HelloClusterJava.err
    -rw-rw-r-- 1 jimezam jimezam 504 Aug  2 11:54
    HelloClusterJava.java
    -rw-rw-r-- 1 jimezam jimezam 607 Aug  2 12:00
    _HelloClusterJava.log
    -rw-rw-r-- 1 jimezam jimezam  57 Aug  2 12:00
    _HelloClusterJava.out
    -rw-rw-r-- 1 jimezam jimezam 195 Aug  2 11:59
    HelloClusterJava.submit

$ cat _HelloClusterJava.log

    000 (2399.000.000) 08/02 11:59:44 Job submitted from
    host: <190.15.30.200:9486>
    ...
    001 (2399.000.000) 08/02 12:00:02 Job executing on host:
    <190.15.30.202:9960>
    ...
    005 (2399.000.000) 08/02 12:00:14 Job terminated.
         (1) Normal termination (return value 0)
```

```
          Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
     Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          0  -  Run Bytes Sent By Job
          0  -  Run Bytes Received By Job
          0  -  Total Bytes Sent By Job
          0  -  Total Bytes Received By Job
     ...

$ cat _HelloClusterJava.out

     Hello cluster, my name is: UAMCLNodo2.autonoma.edu.co.
```

Nivel de grid

Verificar conectividad con entre el Cliente Grid y el *Compute Element*.

```
$ ping -c5 ce.autonoma.edu.co

     PING ce.autonoma.edu.co (200.21.104.84) 56(84) bytes of
     data.
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_seq=1 ttl=64 time=0.172 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_seq=2 ttl=64 time=0.198 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_seq=3 ttl=64 time=0.170 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_seq=4 ttl=64 time=0.174 ms
     64 bytes from ce.autonoma.edu.co (200.21.104.84):
     icmp_seq=5 ttl=64 time=0.179 ms

     --- ce.autonoma.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4001ms
```

```
rtt min/avg/max/mdev = 0.170/0.178/0.198/0.017 ms
```

Cargar y verificar las variables de ambiente del Cliente Grid.

```
$ source /opt/osg-client/setup.sh

$ echo $GLOBUS_TCP_PORT_RANGE

     20000,22000

$ echo $GLOBUS_TCP_SOURCE_RANGE

     23000,25000
```

Crear y verificar el intermediario del certificado.

```
$ grid-proxy-init

     Your identity: /DC=org/DC=doegrids/OU=People/CN=Jorge
     Ivan Meza Martinez
     Enter Grid pass phrase for this identity:
     Creating proxy .....................................
     Done
     Your proxy is valid until: Wed Aug  3 05:44:01 2011

$ grid-proxy-info

     subject  : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez/CN=
     issuer   : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez
     identity : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez
     type     : Proxy draft (pre-RFC) compliant impersonation
     proxy
     strength : 512 bits
     path     : /tmp/x509up_u509
     timeleft : 11:59:50
```

Verificar la autenticación con el Nodo Grid.

```
$ globusrun -a -r ce.autonoma.edu.co

     GRAM Authentication test successful
```

Probar el envío de trabajos al Nodo Grid con las herramientas de Globus.

Utilizando el manejador de trabajos Fork.

```
$ globus-job-run ce.autonoma.edu.co/jobmanager-fork
/bin/hostname

     ce.autonoma.edu.co
```

Utilizando el manejador de trabajos de Condor.

```
$ globus-job-run ce.autonoma.edu.co/jobmanager-condor
/bin/hostname

     UAMCLNodo2.autonoma.edu.co
```

Probar el envío de trabajos al Nodo Grid utilizando Condor-G.

```
$ vi test_condorg.submit

     universe      = grid
     grid_resource = gt2 ce.autonoma.edu.co/jobmanager-condor
     executable    = /bin/hostname
     output        = _test.out.$(Cluster).$(Process)
     error         = _test.err.$(Cluster).$(Process)
     log           = _test.log
     queue

$ condor_submit test_condorg.submit

     Submitting job(s).
     Logging submit event(s).
```

```
     1 job(s) submitted to cluster 54.

$ ls -l

     total 12
     -rw-r--r-- 1 jimezam jimezam  237 Aug  2 17:52
     test_condorg.submit
     -rw-r--r-- 1 jimezam jimezam    0 Aug  2 17:56
     _test.err.54.0
     -rw-r--r-- 1 jimezam jimezam 1062 Aug  2 17:56 _test.log
     -rw-r--r-- 1 jimezam jimezam   27 Aug  2 17:56
     _test.out.54.0

$ cat _test.log

     000 (054.000.000) 08/02 17:53:03 Job submitted from
     host: <190.15.30.205:43293>
     ...
     017 (054.000.000) 08/02 17:53:12 Job submitted to Globus
         RM-Contact: ce.autonoma.edu.co/jobmanager-condor
         JM-Contact:
     https://ce.autonoma.edu.co:20886/30846/1312325614/
         Can-Restart-JM: 1
     ...
     027 (054.000.000) 08/02 17:53:13 Job submitted to grid
     resource
         GridResource: gt2 ce.autonoma.edu.co/jobmanager-
     condor
         GridJobId: gt2 ce.autonoma.edu.co/jobmanager-condor
     https://ce.autonoma.edu.co:20886/30846/1312325614/
     ...
     001 (054.000.000) 08/02 17:55:47 Job executing on host:
     gt2 ce.autonoma.edu.co/jobmanager-condor
     ...
     005 (054.000.000) 08/02 17:56:26 Job terminated.
          (1) Normal termination (return value 0)
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
```

```
                Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          0  -  Run Bytes Sent By Job
          0  -  Run Bytes Received By Job
          0  -  Total Bytes Sent By Job
          0  -  Total Bytes Received By Job
     ...

$ cat _test.out.54.0

     UAMCLNodo2.autonoma.edu.co
```

Probar la transferencia de archivos entre el Cliente Grid y el Nodo Grid.

Transferir el archivo del Cliente Grid al Nodo Grid.

```
$ globus-url-copy -v file:///etc/services
gsiftp://ce.autonoma.edu.co/tmp/ARCHIVO_REMOTO

     Source: file:///etc/
     Dest:   gsiftp://ce.autonoma.edu.co/tmp/
       services  ->  ARCHIVO_REMOTO
```

Transferir el archivo de regreso, desde el Nodo Grid hacia el Cliente Grid.

```
$ globus-url-copy -v
gsiftp://ce.autonoma.edu.co/tmp/ARCHIVO_REMOTO file://$
{PWD}/ARCHIVO_LOCAL

     Source: gsiftp://ce.autonoma.edu.co/tmp/
     Dest:   file:///home/jimezam/jobs/filetransfer/
       ARCHIVO_REMOTO  ->  ARCHIVO_LOCAL

$ ls -l

     total 360
     -rw-r--r-- 1 jimezam jimezam 362031 Aug  2 18:06
     ARCHIVO_LOCAL
```

Comparar el archivo transmitido inicialmente con el finalmente recibido.

```
$ diff -rcs /etc/services  ARCHIVO_LOCAL
     Files /etc/services and ARCHIVO_LOCAL are identical
```

Destruir y verificar el intermediario del certificado.

```
$ grid-proxy-destroy

$ grid-proxy-info

     ERROR: Couldn't find a valid proxy.
     Use -debug for further information.
```

Nodo Universidad del Rosario (UR)

Nivel de cluster

Verificar conectividad con el nodo principal del *cluster*.

```
$ ping -c 5 ce.urosario.edu.co

     PING ce.urosario.edu.co (201.234.181.15) 56(84) bytes of
     data.

     --- ce.urosario.edu.co ping statistics ---
     5 packets transmitted, 0 received, 100% packet loss,
     time 3999ms
```

Nota: el *firewall* de la Universidad del Rosario no permite el paso de paquetes ICMP hasta el CE, por lo tanto el `ping` fallará siempre.

Verificar conectividad SSH con el nodo principal del *cluster*

```
$ ssh jimezam@ce.urosario.edu.co

     jimezam@ce.urosario.edu.co's password:
     Last login: Thu Jun 23 14:46:08 2011 from 190.7.145.35

[jimezam@ce ~]$ uname -a

     Linux ce.urosario.edu.co 2.6.18-238.1.1.el5xen #1 SMP
     Tue Jan 18 19:15:11 EST 2011 x86_64 x86_64 x86_64
     GNU/Linux
```

Verificar conectividad con los nodos trabajadores del *cluster*.

```
$ ping -c 5 nodo5

     PING URCLNodo5 (10.10.11.13) 56(84) bytes of data.
     64 bytes from URCLNodo5 (10.10.11.13): icmp_seq=1 ttl=64
     time=0.126 ms
```

```
    64 bytes from URCLNodo5 (10.10.11.13): icmp_seq=2 ttl=64
    time=0.112 ms
    64 bytes from URCLNodo5 (10.10.11.13): icmp_seq=3 ttl=64
    time=0.111 ms
    64 bytes from URCLNodo5 (10.10.11.13): icmp_seq=4 ttl=64
    time=0.112 ms
    64 bytes from URCLNodo5 (10.10.11.13): icmp_seq=5 ttl=64
    time=0.110 ms

    --- URCLNodo5 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time
    3998ms
    rtt min/avg/max/mdev = 0.110/0.114/0.126/0.009 ms

$ ping -c 5 nodo6

    PING URCLNodo6 (10.10.11.14) 56(84) bytes of data.
    64 bytes from URCLNodo6 (10.10.11.14): icmp_seq=1 ttl=64
    time=0.138 ms
    64 bytes from URCLNodo6 (10.10.11.14): icmp_seq=2 ttl=64
    time=0.114 ms
    64 bytes from URCLNodo6 (10.10.11.14): icmp_seq=3 ttl=64
    time=0.115 ms
    64 bytes from URCLNodo6 (10.10.11.14): icmp_seq=4 ttl=64
    time=0.112 ms
    64 bytes from URCLNodo6 (10.10.11.14): icmp_seq=5 ttl=64
    time=0.111 ms

    --- URCLNodo6 ping statistics ---
    5 packets transmitted, 5 received, 0% packet loss, time
    3998ms
    rtt min/avg/max/mdev = 0.111/0.118/0.138/0.010 ms

$ ping -c 5 nodo7

    PING URCLNodo7 (10.10.11.12) 56(84) bytes of data.
    64 bytes from URCLNodo7 (10.10.11.12): icmp_seq=1 ttl=64
    time=3.48 ms
    64 bytes from URCLNodo7 (10.10.11.12): icmp_seq=2 ttl=64
    time=0.112 ms
    64 bytes from URCLNodo7 (10.10.11.12): icmp_seq=3 ttl=64
    time=0.112 ms
```

```
64 bytes from URCLNodo7 (10.10.11.12): icmp_seq=4 ttl=64
time=0.112 ms
64 bytes from URCLNodo7 (10.10.11.12): icmp_seq=5 ttl=64
time=0.113 ms

--- URCLNodo7 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
4003ms
rtt min/avg/max/mdev = 0.112/0.787/3.488/1.350 ms
```

Verificar conectividad SSH con los nodos trabajadores del *cluster*

```
# ssh nodo5

     Last login: Thu Jun 30 20:45:43 2011 from urclnodo6

[root@URCLNodo5 ~]# uname -a

     Linux URCLNodo5.urosario.edu.co 2.6.18-238.1.1.el5 #1
     SMP Tue Jan 18 18:49:37 EST 2011 x86_64 x86_64 x86_64
     GNU/Linux

# ssh nodo6

     Last login: Thu Jun 30 21:48:01 2011 from
     ce.urosario.edu.co

[root@URCLNodo6 ~]# uname -a

     Linux URCLNodo6.urosario.edu.co 2.6.18-238.1.1.el5 #1
     SMP Tue Jan 18 18:49:37 EST 2011 x86_64 x86_64 x86_64
     GNU/Linux

# ssh nodo7

     Last login: Thu Jun 30 21:46:27 2011 from urclnodo6

[root@URCLNodo7 ~]# uname -a

     Linux URCLNodo7.urosario.edu.co 2.6.18-238.1.1.el5 #1
     SMP Tue Jan 18 18:49:37 EST 2011 x86_64 x86_64 x86_64
```

```
     GNU/Linux
```

Verificar el estado del sistema de archivos compartido (NFS) en el *cluster*

```
# exportfs -v

     /opt/octave/3.2.4
     10.10.11.0/24(rw,async,wdelay,no_root_squash,no_subtree_
     check,anonuid=65534,anongid=65534)
     /exports/condor
     10.10.11.0/24(rw,async,wdelay,no_root_squash,no_subtree_
     check,anonuid=65534,anongid=65534)
     /exports/home
     10.10.11.0/24(rw,async,wdelay,no_root_squash,no_subtree_
     check,anonuid=65534,anongid=65534)
     /exports/osg
     10.10.11.0/24(rw,async,wdelay,no_root_squash,no_subtree_
     check,anonuid=65534,anongid=65534)

URCLNodo5# mount | grep URCLMaestro

     URCLMaestro:/exports/condor on /nfs/condor type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/home on /nfs/home type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/osg on /nfs/osg type nfs
     (rw,noatime,bg,intr,addr=10.10.11.15)
     URCLMaestro:/exports/octave on /nfs/octave type nfs
     (rw,noatime,bg,intr,addr=10.10.11.15)

URCLNodo6# mount | grep URCLMaestro

     URCLMaestro:/exports/condor on /nfs/condor type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/home on /nfs/home type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/osg on /nfs/osg type nfs
```

```
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/octave on /nfs/octave type nfs
     (rw,noatime,bg,intr,addr=10.10.11.15)

URCLNodo7# mount | grep URCLMaestro

     URCLMaestro:/exports/octave on /nfs/octave type nfs
     (rw,noatime,bg,addr=10.10.11.15)
     URCLMaestro:/exports/condor on /nfs/condor type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/home on /nfs/home type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
     URCLMaestro:/exports/osg on /nfs/osg type nfs
     (rw,noatime,rsize=8192,wsize=8192,bg,intr,addr=10.10.11.
     15)
```

Verificar el pool de nodos trabajadores del *cluster* Condor

```
# condor_status

     Name               OpSys      Arch   State     Activity
     LoadAv Mem   ActvtyTime

     slot1@URCLNodo5.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+01:30:04
     slot2@URCLNodo5.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+09:30:16
     slot3@URCLNodo5.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+09:30:17
     slot4@URCLNodo5.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+09:30:17
     slot1@URCLNodo6.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+01:30:04
     slot2@URCLNodo6.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+09:30:16
     slot3@URCLNodo6.ur LINUX      X86_64 Unclaimed Idle
     0.000  4010  0+09:30:17
     slot4@URCLNodo6.ur LINUX      X86_64 Unclaimed Idle
```

```
0.000  4010  0+09:30:18
slot1@URCLNodo7.ur LINUX      X86_64 Unclaimed Idle
0.230  4010  0+00:00:04
slot2@URCLNodo7.ur LINUX      X86_64 Unclaimed Idle
0.000  4010  0+00:00:05
slot3@URCLNodo7.ur LINUX      X86_64 Unclaimed Idle
0.000  4010  0+00:00:06
slot4@URCLNodo7.ur LINUX      X86_64 Unclaimed Idle
0.000  4010  0+00:00:07
                     Total Owner Claimed Unclaimed
Matched Preempting Backfill

        X86_64/LINUX    12     0       0         12
0          0        0

               Total    12     0       0         12
0          0        0
```

Verificar el soporte para Java en los nodos trabajadores del *cluster*

```
# condor_status -java

     Name               JavaVendor Ver    State     Activity
     LoadAv Mem   ActvtyTime

     slot1@URCLNodo5.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+01:30:04
     slot2@URCLNodo5.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:16
     slot3@URCLNodo5.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:17
     slot4@URCLNodo5.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:17
     slot1@URCLNodo6.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+01:30:04
     slot2@URCLNodo6.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:16
     slot3@URCLNodo6.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:17
     slot4@URCLNodo6.ur Sun Micros 1.6.0_ Unclaimed Idle
     0.000  4010  0+09:30:18
```

```
slot1@URCLNodo7.ur Sun Micros 1.6.0_ Unclaimed Idle
0.230  4010  0+00:00:04
slot2@URCLNodo7.ur Sun Micros 1.6.0_ Unclaimed Idle
0.000  4010  0+00:00:05
slot3@URCLNodo7.ur Sun Micros 1.6.0_ Unclaimed Idle
0.000  4010  0+00:00:06
slot4@URCLNodo7.ur Sun Micros 1.6.0_ Unclaimed Idle
0.000  4010  0+00:00:07
                      Total Owner Claimed Unclaimed
Matched Preempting Backfill

        X86_64/LINUX    12     0       0        12
0          0        0

               Total    12     0       0        12
0          0        0
```

Verificar la cola de trabajos del *cluster* Condor

```
# condor_q

     -- Submitter: ce.urosario.edu.co : <10.10.11.15:9869> :
     ce.urosario.edu.co
      ID      OWNER            SUBMITTED     RUN_TIME ST PRI
     SIZE CMD

     0 jobs; 0 idle, 0 running, 0 held
```

Probar el envío de trabajos al universo Vanilla del *cluster*

```
$ vi script.sh

     #!/bin/sh
     /bin/hostname
     /usr/bin/id

$ chmod +x script.sh

$ vi demo_vanilla.submit
```

```
executable              = script.sh
universe                = vanilla
log                     = _demo_vanilla.log
output                  = _demo_vanilla.out
error                   = _demo_vanilla.err
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
queue

$ condor_submit demo_vanilla.submit

    Submitting job(s).
    Logging submit event(s).
    1 job(s) submitted to cluster 8073.

$ ls -l

    total 16
    -rw-r--r-- 1 jimezam jimezam   0 Aug  2 16:59
    _demo_vanilla.err
    -rw-r--r-- 1 jimezam jimezam 609 Aug  2 16:59
    _demo_vanilla.log
    -rw-r--r-- 1 jimezam jimezam 136 Aug  2 16:59
    _demo_vanilla.out
    -rw-r--r-- 1 jimezam jimezam 280 Aug  2 16:59
    demo_vanilla.submit
    -rwxr-xr-x 1 jimezam jimezam  38 Aug  2 16:57 script.sh

$ cat _demo_vanilla.log

    000 (8073.000.000) 08/02 16:59:10 Job submitted from
    host: <10.10.11.15:9869>
    ...
    001 (8073.000.000) 08/02 16:59:24 Job executing on host:
    <10.10.11.12:9577>
    ...
    005 (8073.000.000) 08/02 16:59:24 Job terminated.
         (1) Normal termination (return value 0)
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
    Usage
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
    Usage
```

```
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
          Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          136  -  Run Bytes Sent By Job
          38  -  Run Bytes Received By Job
          136  -  Total Bytes Sent By Job
          38  -  Total Bytes Received By Job
     ...

$ cat _demo_vanilla.out

     URCLNodo7.urosario.edu.co
     uid=509(jimezam) gid=509(jimezam) groups=509(jimezam)
     context=root:system_r:unconfined_t:SystemLow-SystemHigh
```

Probar el envío de trabajos al universo Standard del *cluster*

```
$ vi HelloCluster.c

     #include <stdio.h>
     #include <unistd.h>

     int main(void)
     {
          char hn[256];
          int control = gethostname(hn, sizeof(hn));

          if(control == -1)
          {
               perror("gethostname");
               return 1;
          }

          printf("Hello cluster, my name is: %s.\n\n", hn);

          return 0;
     }

$ gcc -c HelloCluster.c
```

```
$ condor_compile gcc HelloCluster.o -o HelloCluster

    LINKING FOR CONDOR : /usr/bin/ld
    -L/opt/condor/current/lib -Bstatic --eh-frame-hdr -m
    elf_x86_64 --hash-style=gnu -dynamic-linker /lib64/ld-
    linux-x86-64.so.2 -o HelloCluster
    /opt/condor/current/lib/condor_rt0.o
    /usr/lib/gcc/x86_64-redhat-
    linux/4.1.2/../../../../lib64/crti.o
    /usr/lib/gcc/x86_64-redhat-linux/4.1.2/crtbeginT.o
    -L/opt/condor/current/lib -L/usr/lib/gcc/x86_64-redhat-
    linux/4.1.2 -L/usr/lib/gcc/x86_64-redhat-linux/4.1.2
    -L/usr/lib/gcc/x86_64-redhat-
    linux/4.1.2/../../../../lib64 -L/lib/../lib64
    -L/usr/lib/../lib64 HelloCluster.o
    /opt/condor/current/lib/libcondorsyscall.a
    /opt/condor/current/lib/libcondor_z.a
    /opt/condor/current/lib/libcomp_libstdc++.a
    /opt/condor/current/lib/libcomp_libgcc.a
    /opt/condor/current/lib/libcomp_libgcc_eh.a --as-needed
    --no-as-needed -lcondor_c -lcondor_nss_files
    -lcondor_nss_dns -lcondor_resolv -lcondor_c
    -lcondor_nss_files -lcondor_nss_dns -lcondor_resolv
    -lcondor_c /opt/condor/current/lib/libcomp_libgcc.a
    /opt/condor/current/lib/libcomp_libgcc_eh.a --as-needed
    --no-as-needed /usr/lib/gcc/x86_64-redhat-
    linux/4.1.2/crtend.o /usr/lib/gcc/x86_64-redhat-
    linux/4.1.2/../../../../lib64/crtn.o
    /
    opt/condor/current/lib/libcondorsyscall.a(condor_file_ag
    ent.o): In function `CondorFileAgent::open(char const*,
    int, int)':
    /
    home/condor/execute/dir_9339/userdir/src/condor_ckpt/con
    dor_file_agent.cpp:106: warning: the use of `tmpnam' is
    dangerous, better use `mkstemp'
    /
    opt/condor/current/lib/libcondorsyscall.a(special_stubs.
    o): In function `condor_gethostbyaddr':
    /
    home/condor/execute/dir_9339/userdir/src/condor_syscall_
    lib/special_stubs.cpp:201: warning: Using
```

```
     'gethostbyaddr' in statically linked applications
     requires at runtime the shared libraries from the glibc
     version used for linking
     /
     opt/condor/current/lib/libcondorsyscall.a(special_stubs.
     o): In function `condor_gethostbyname':
     /
     home/condor/execute/dir_9339/userdir/src/condor_syscall_
     lib/special_stubs.cpp:194: warning: Using
     'gethostbyname' in statically linked applications
     requires at runtime the shared libraries from the glibc
     version used for linking
     /opt/condor/current/lib/libcondorsyscall.a(sock.o): In
     function `Sock::getportbyserv(char*)':
     /
     home/condor/execute/dir_9339/userdir/src/condor_io/sock.
     cpp:233: warning: Using 'getservbyname' in statically
     linked applications requires at runtime the shared
     libraries from the glibc version used for linking

$ vi HelloCluster.submit

     Executable     = HelloCluster
     Universe       = standard
     Output         = _HelloCluster.out
     Error          = _HelloCluster.err
     Log            = _HelloCluster.log
     Queue

$ condor_submit HelloCluster.submit

     Submitting job(s).
     Logging submit event(s).
     1 job(s) submitted to cluster 8074.

$ ls -l

     total 5368
     -rwxr-xr-x 1 jimezam jimezam 5461107 Aug  2 17:01
     HelloCluster
     -rw-r--r-- 1 jimezam jimezam     279 Aug  2 17:00
     HelloCluster.c
```

```
     -rw-r--r-- 1 jimezam jimezam       0 Aug  2 17:01
     _HelloCluster.err
     -rw-r--r-- 1 jimezam jimezam     621 Aug  2 17:02
     _HelloCluster.log
     -rw-r--r-- 1 jimezam jimezam    1760 Aug  2 17:01
     HelloCluster.o
     -rw-r--r-- 1 jimezam jimezam      55 Aug  2 17:02
     _HelloCluster.out
     -rw-r--r-- 1 jimezam jimezam     172 Aug  2 17:01
     HelloCluster.submit

$ cat _HelloCluster.log

     000 (8074.000.000) 08/02 17:01:56 Job submitted from
     host: <10.10.11.15:9869>
     ...
     001 (8074.000.000) 08/02 17:02:06 Job executing on host:
     <10.10.11.12:9577>
     ...
     005 (8074.000.000) 08/02 17:02:06 Job terminated.
          (1) Normal termination (return value 0)
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          1048  -  Run Bytes Sent By Job
          5462130  -  Run Bytes Received By Job
          1048  -  Total Bytes Sent By Job
          5462130  -  Total Bytes Received By Job
     ...

$ cat _HelloCluster.out

     Hello cluster, my name is: URCLNodo7.urosario.edu.co.
```

Probar el envío de trabajos al universo Java del *cluster*

```
$ vi HelloClusterJava.java

     public class HelloClusterJava
     {
          public static void main(String args[])
          {
               String hostname = "*** desconocido *** ";

               try
               {
                    java.net.InetAddress localMachine =

          java.net.InetAddress.getLocalHost();
                    hostname = localMachine.getHostName();
               }
               catch (java.net.UnknownHostException e)
               {
                    hostname = e.getMessage();
               }

               System.out.println("Hello cluster, my name is:
     " +                                  hostname + ".\n\n");
          }
     }

$ javac HelloClusterJava.java

$ vi HelloClusterJava.submit

     executable = HelloClusterJava.class
     arguments  = HelloClusterJava
     universe   = java
     log        = _HelloClusterJava.log
     output     = _HelloClusterJava.out
     error      = _HelloClusterJava.err
     queue

$ condor_submit HelloClusterJava.submit

Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 8075.
```

```
$ ls -l

     total 20
     -rw-r--r-- 1 jimezam jimezam 959 Aug  2 17:04
     HelloClusterJava.class
     -rw-r--r-- 1 jimezam jimezam   0 Aug  2 17:04
     _HelloClusterJava.err
     -rw-r--r-- 1 jimezam jimezam 453 Aug  2 17:03
     HelloClusterJava.java
     -rw-r--r-- 1 jimezam jimezam 603 Aug  2 17:04
     _HelloClusterJava.log
     -rw-r--r-- 1 jimezam jimezam  56 Aug  2 17:04
     _HelloClusterJava.out
     -rw-r--r-- 1 jimezam jimezam 202 Aug  2 17:04
     HelloClusterJava.submit

$ cat _HelloClusterJava.log

     000 (8075.000.000) 08/02 17:04:16 Job submitted from
     host: <10.10.11.15:9869>
     ...
     001 (8075.000.000) 08/02 17:04:27 Job executing on host:
     <10.10.11.12:9577>
     ...
     005 (8075.000.000) 08/02 17:04:28 Job terminated.
          (1) Normal termination (return value 0)
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
     Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total
     Remote Usage
               Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
     Usage
          0  -  Run Bytes Sent By Job
          0  -  Run Bytes Received By Job
          0  -  Total Bytes Sent By Job
          0  -  Total Bytes Received By Job
     ...

$ cat _HelloClusterJava.out
```

```
Hello cluster, my name is: URCLNodo7.urosario.edu.co.
```

Nivel de grid

Verificar conectividad con entre el Cliente Grid y el *Compute Element.*

```
$ ping -c5 ce.urosario.edu.co

     PING ce.urosario.edu.co (10.10.11.15) 56(84) bytes of
     data.
     64 bytes from ce.urosario.edu.co (10.10.11.15):
     icmp_seq=1 ttl=64 time=0.120 ms
     64 bytes from ce.urosario.edu.co (10.10.11.15):
     icmp_seq=2 ttl=64 time=0.112 ms
     64 bytes from ce.urosario.edu.co (10.10.11.15):
     icmp_seq=3 ttl=64 time=0.092 ms
     64 bytes from ce.urosario.edu.co (10.10.11.15):
     icmp_seq=4 ttl=64 time=0.112 ms
     64 bytes from ce.urosario.edu.co (10.10.11.15):
     icmp_seq=5 ttl=64 time=0.110 ms

     --- ce.urosario.edu.co ping statistics ---
     5 packets transmitted, 5 received, 0% packet loss, time
     4001ms
     rtt min/avg/max/mdev = 0.092/0.109/0.120/0.011 ms
```

Cargar y verificar las variables de ambiente del Cliente Grid.

```
$ source /opt/osg-client/setup.sh

$ echo $GLOBUS_TCP_PORT_RANGE

     20000,22000

$ echo $GLOBUS_TCP_SOURCE_RANGE

     23000,25000
```

Crear y verificar el intermediario del certificado.

```
$ grid-proxy-init

     Your identity: /DC=org/DC=doegrids/OU=People/CN=Jorge
     Ivan Meza Martinez
     Enter Grid pass phrase for this identity:
     Creating proxy ..................................
     Done
     Your proxy is valid until: Wed Aug  2 20:21:01 2011

$ grid-proxy-info

     subject  : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez/CN=
     issuer   : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez
     identity : /DC=org/DC=doegrids/OU=People/CN=Jorge Ivan
     Meza Martinez
     type     : Proxy draft (pre-RFC) compliant impersonation
     proxy
     strength : 512 bits
     path     : /tmp/x509up_u509
     timeleft : 11:49:30
```

Verificar la autenticación con el Nodo Grid.

```
$ globusrun -a -r ce.urosario.edu.co

     GRAM Authentication test successful
```

Probar el envío de trabajos al Nodo Grid con las herramientas de Globus.

Utilizando el manejador de trabajos Fork.

```
$ globus-job-run ce.urosario.edu.co/jobmanager-fork
/bin/hostname

     ce.urosario.edu.co
```

Utilizando el manejador de trabajos de Condor.

```
$ globus-job-run ce.urosario.edu.co/jobmanager-condor
/bin/hostname

     URCLNodo7.urosario.edu.co
```

Probar el envío de trabajos al Nodo Grid utilizando Condor-G.

```
$ vi test_condorg.submit

     universe      = grid
     grid_resource = gt2 ce.urosario.edu.co/jobmanager-condor
     executable    = /bin/hostname
     output        = _test.out.$(Cluster).$(Process)
     error         = _test.err.$(Cluster).$(Process)
     log           = _test.log
     queue

$ condor_submit test_condorg.submit

     Submitting job(s).
     Logging submit event(s).
     1 job(s) submitted to cluster 8078.

$ ls -l

     total 12
     -rw-r--r-- 1 jimezam jimezam  237 Aug  2 20:24
     test_condorg.submit
     -rw-r--r-- 1 jimezam jimezam    0 Aug  2 20:28
     _test.err.8078.0
     -rw-r--r-- 1 jimezam jimezam 1062 Aug  2 20:28 _test.log
     -rw-r--r-- 1 jimezam jimezam   27 Aug  2 20:28
     _test.out.8078.0

$ cat _test.log

     000 (8078.000.000) 08/02 20:53:03 Job submitted from
```

```
    host: <10.10.11.13:43290>
    ...
    017 (8078 .000.000) 08/02 20:53:12 Job submitted to
    Globus
        RM-Contact: ce.urosario.edu.co/jobmanager-condor
        JM-Contact:
    https://ce.urosario.edu.co:20887/30586/1316525210/
        Can-Restart-JM: 1
    ...
    027 (8078.000.000) 08/02 20:53:13 Job submitted to grid
    resource
        GridResource: gt2 ce.urosario.edu.co/jobmanager-
    condor
        GridJobId: gt2 ce.urosario.edu.co/jobmanager-condor
    https://ce.urosario.edu.co:20887/30586/1316525210/
    ...
    001 (8078.000.000) 08/02 20:55:47 Job executing on host:
    gt2 ce.urosario.edu.co/jobmanager-condor
    ...
    005 (8078.000.000) 08/02 20:56:26 Job terminated.
         (1) Normal termination (return value 0)
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Remote
    Usage
              Usr 0 00:00:00, Sys 0 00:00:00  -  Run Local
    Usage
              Usr 0 00:00:00, Sys 0 00:00:00  -  Total
    Remote Usage
              Usr 0 00:00:00, Sys 0 00:00:00  -  Total Local
    Usage
         0  -  Run Bytes Sent By Job
         0  -  Run Bytes Received By Job
         0  -  Total Bytes Sent By Job
         0  -  Total Bytes Received By Job
    ...

$ cat _test.out.8078.0

    UAMCLNodo2.urosario.edu.co
```

Probar la transferencia de archivos entre el Cliente Grid y el Nodo Grid.

Transferir el archivo del Cliente Grid al Nodo Grid.

```
$ globus-url-copy -v file:///etc/services
gsiftp://ce.urosario.edu.co/tmp/ARCHIVO_REMOTO

     Source: file:///etc/
     Dest:   gsiftp://ce.urosario.edu.co/tmp/
       services  ->  ARCHIVO_REMOTO
```

Transferir el archivo de regreso, desde el Nodo Grid hacia el Cliente Grid.

```
$ globus-url-copy -v
gsiftp://ce.urosario.edu.co/tmp/ARCHIVO_REMOTO file://$
{PWD}/ARCHIVO_LOCAL

     Source: gsiftp://ce.urosario.edu.co/tmp/
     Dest:   file:///home/jimezam/jobs/filetransfer/
       ARCHIVO_REMOTO  ->  ARCHIVO_LOCAL

$ ls -l

     total 360
     -rw-r--r-- 1 jimezam jimezam 362031 Aug  2 21:06
     ARCHIVO_LOCAL
```

Comparar el archivo transmitido inicialmente con el finalmente recibido.

```
$ diff -rcs /etc/services  ARCHIVO_LOCAL

     Files /etc/services and ARCHIVO_LOCAL are identical
```

Destruir y verificar el intermediario del certificado.

```
$ grid-proxy-destroy
```

```
$ grid-proxy-info

     ERROR: Couldn't find a valid proxy.
     Use -debug for further information.
```