Arquitectura de almacenamiento masivo de datos en la infraestructura Grid usando el middlewareglite

Data mass-storage architecture in a Grid infrastructure using glite middleware

Iván Fernando Gómez Pedraza
Ingeniero de Sistemas. Investigador de la Universidad Industrial de Santander. Bucaramanga, Colombia. Contacto: *ivar.gomez@gmail.com*

Carlos Alberto Varela Garzón
Ingeniero de Sistemas. Investigador de la Universidad Industrial de Santander. Bucaramanga, Colombia. Contacto: *carlosbebeto86@gmail.com*

Henry Arguello Fuentes
Ingeniero eléctrico, doctor en Electrical and Computer Engineering. Docente de la Universidad Industrial de Santander. Bucaramanga, Colombia.
Contacto: *henarfu@uis.edu.co*

Juan Carlos Escobar Ramírez
Ingeniero de sistemas. Investigador de la Universidad Industrial de Santander. Bucaramanga, Colombia. Contacto: *juanca.es@gmail.com*

Fecha de recepción: 10 de noviembre de 2011
Fecha de aceptación: 28 de agosto de 2012
Clasificación del artículo: Investigación
Financiamiento: Universidad Industrial de Santander

Palabras clave: banco de datos, computación Grid, dispositivos de almacenamiento, HPC, IGALC.

Key words: data bank, Grid computing, storage devices, HPC, IGALC.

RESUMEN

En la actualidad, el incremento de las investigaciones en las universidades consumen altos recursos de procesamiento y almacenamiento, generando la necesidad de tener una infraestructura de supercomputación. Este trabajo tiene como objetivo dar una solución a los problemas de almacenamiento que surgen en los diferentes grupos de investigación de una universidad, tal como la Universidad Industrial de Santander (Colombia) entidad que financia este proyecto. Implementando una infraestructura de almacenamiento masivo usando un sistema de archivos compatible con el middleware de EGEE (The Enabling Gridsfor E-science), aprovechando el uso del espacio de disco libre en nodos de trabajo, ofreciendo así una solución a un bajo costo. Además, se presenta la implementación de un clúster con los servicios fundamentales para Grid y así poder integrarlo a la infraestructura intercontinental de EELA-2 (E-science Grid facility for Europe and Latin America), estableciendo una plataforma distribuida de procesamiento y almacenamiento, para el uso de la comunidad científica de una universidad.

ABSTRACT

Nowadays, the increase in research projects at universities requires high-computing and mass-storage equipment, creating the need for having a supercomputing infrastructure. This work attempts to find a solution to the storage problems that arise in the different research groups at universities. Massive-storage infrastructures, which use a file system compatible with EGEE middleware, are implemented, taking advantage of the storage space left by working nodes. This should offer a very low-price solution. Additionally, a cluster with fundamental Grid services is created and thus integrated to the intercontinental infrastructure of EELA-2, establishing a platform for distributed computing and storage intended for the scientific community at Universidad Industrial de Santander.

* * *

1. INTRODUCCIÓN

Los retos planteados por la ciencia han crecido a un ritmo tal que la tecnología no está preparada para afrontarlos. De un trabajo empírico y teórico la ciencia ha pasado, en la actualidad, a un proceso de simulación y de manipulación de datos en donde son necesarias miles de horas de CPU, Petabytes de almacenamiento y Gigabytes por segundo en capacidad de comunicación [1]. Gracias al desarrollo de los sistemas operativos y la supercomputación se ha creado una nueva perspectiva, a partir de los denominados Clústeres, los cuales permiten la integración de máquinas independientes que se interconectan en un solo conjunto, dando al usuario la apariencia de un solo supercomputador, pero con ciertas limitaciones, ya que no permite una plataforma heterogénea y compartir recursos en un entorno distribuido.

Es ahí donde aparece una nueva propuesta llamada Grid computacional [2] como respuesta a las necesidades de la e-ciencia, la cual se apoya de manera intensiva en recursos computacionales, pero donde dicho apoyo se realiza en entornos altamente distribuidos. La Grid permite compartir recursos entre grupos que estén distribuidos geográficamente o que no pertenezcan a una misma organización y, de esta forma, poder hacer que todos los participantes obtengan una experiencia satisfactoria a la hora de hacer investigación, sin sufrir las consecuencias de una baja capacidad computacional propia.

El entorno local se caracteriza por poseer recursos limitados para la ejecución de las investigaciones, por lo tanto, un sitio Grid constituye una de las alternativas que más se ajusta al entorno, ya que brinda a las investigaciones capacidades considerables, aprovechando de una mejor manera los recursos que este ofrece sin la necesidad de grandes inversiones.

Actualmente, la Universidad Industrial de Santander cuenta con diferentes grupos de investigación, que poseen aplicaciones que se caracterizan por la necesidad de realizar gran cantidad de cálculos para el procesamiento de datos, arrojando una considerable suma de resultados que deben ser almacenados en los discos duros para su posterior análisis y para el desarrollo de aplicaciones futuras. Por lo tanto, el presente proyecto busca implementar un prototipo de una infraestructura de almacenamiento masivo de datos, a partir de los componentes fundamentales del middleware gLite [3], en una sala de supercomputación en donde se brindará este servicio.

El presente artículo está organizado de la siguiente forma: los primeros seis numerales dan una breve descripción del proyecto y sus autores, en el numeral 7 se describe el marco teórico donde se encuentran las diferentes tecnologías que han sido desarrolladas para prestar el servicio de almacenamiento masivo de datos, profundizando en la arquitectura Grid computacional y el middleware gLite; luego, en el siguiente numeral, se explicará la arquitectura del prototipo de almacenamiento creada, con sus respectivas pruebas, y por último se presentan las conclusiones y recomendaciones para futuros trabajos de investigación.

2. ESTADO DEL ARTE

2.1 Middleware gLite

El middleware es el software que organiza e integra los servicios en un Grid, consiste en un conjunto de servicios que implementan el acceso uniforme a los datos y recursos, la autenticación y autorización, el manejo de la información y la planificación de la asignación de los recursos. También se encuentran los servicios que permiten obtener la información de un recurso en particular y gestionarlo controlando el acceso, arranque de procesos, gestión, monitorización y auditoría [4]. Este middleware ha tenido una evolución a través de los años como se puede observar en la figura 1.

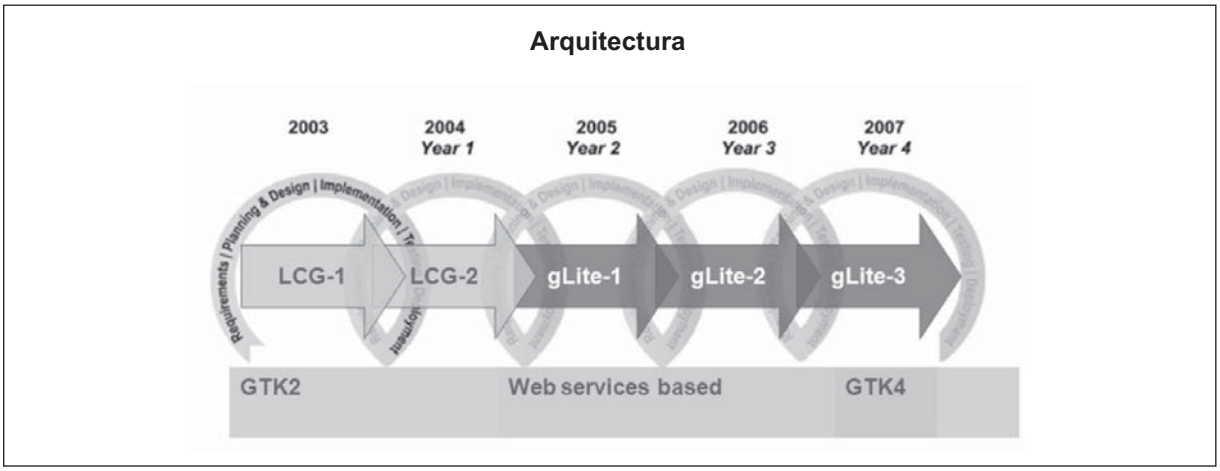

Figura 1. Línea de Evolución del Middleware gLite
Fuente disponible en: http://glite.web.cern.ch/glite/

2.1.1 Componentes del Middleware gLite

Se mostrará una a continuación parte de la arquitectura y de los servicios que la componen:

User Interface (UI): la interfaz de usuario (UI) es el punto de acceso al Grid, ésta puede ser cualquier máquina donde los usuarios tienen una cuenta personal y donde su certificado de usuario está instalado. Desde una interfaz de usuario, un usuario puede ser autenticado y autorizado para utilizar los recursos, y pueden acceder a las funcionalidades que ofrecen los sistemas de Información, Workload y Data Manager.

Computing Element (CE): el Computing Element (CE) [5] es el servicio que representa un recurso de cómputo. Su principal funcionalidad es la gestión de trabajos (envío de trabajos, control de trabajos). El CE puede ser utilizado por un cliente genérico, un usuario final interactúa directamente con el ComputingElement, o el Workload Manager, el cual envía un determinado trabajo a un Computing Element adecuado encontrado mediante un proceso de emparejamiento**.**

Storage Element (SE): es el servicio que permite a un usuario o una aplicación almacenar datos para una futura consulta o recuperación de los mismos, para ser usados en otras aplicaciones o para mejorar la ya existente. Un Storage Element (SE) proporciona acceso uniforme a los recursos de almacenamiento de datos. El SE puede controlar servidores de disco simple, grandes arreglos de disco o cinta, basados en los sistemas de almacenamiento masivo (MSS).

Existen varios tipos de elementos de almacenamiento, los cuales pueden soportar diferentes protocolos de acceso de datos e interfaces; como el GSIFTP (un GSI-seguro FTP) [6] que es el protocolo utilizado para la transferencia completa de archivos, mientras que el acceso local y remoto a archivos se lleva a cabo usando el RFIO [7].

Workload Management (WMS): su propósito es el de aceptar y satisfacer las solicitudes de gestión de trabajos procedentes de los clientes. Para luego asignarlo al CE más adecuado para su ejecución, teniendo en cuenta las necesidades y preferencias expresadas en la descripción de trabajo, también registra su estado y recupera su salida [8].

2.2 Manejo de Datos en gLite

2.2.1 Protocolos de Canal de datos

Los protocolos de acceso a datos que soporta gLite 3.1 se pueden observar en la tabla 1:

GSIFTP: el GSIFTP es soportado por cada SE de Grid y es, por lo tanto, el principal protocolo de transferencia de archivos en gLite 3.1. El protocolo GSIFTP básicamente ofrece la funcionalidad de FTP, como en la transferencia de archivos, pero incrementado para soportar la seguridad GSI. Este protocolo es responsable de la transferencia de archivos segura, rápida y eficiente, a y desde, los elementos de Almacenamiento, proporcionando control por parte de un tercero de la transferencia de datos, así como también, la transferencia de datos de flujos paralelos [9].

Tabla 1. Protocolos de canal de Datos en gLite.

Protocolo	Tipo	GSI secure	Descripción	Opcional
GSIFTP	File transfer	Si	FTP-like	No
Gsidcap	File I/O	Si	Acceso a archivos remoto	Si
Insecure RFIO	File I/O	No	Acceso a archivos remoto	Si
Secure RFIO (gsirfio)	File I/O	Si	Acceso a archivos remoto	Si

Fuente: Manual de usuario gLite 3

El protocolo de acceso GSI dCache (gsidcap): el protocolo gsidcap es una versión segura de la GSI del protocolo de acceso dCache, dcap. Siendo la GSI segura, el gsidcap se puede usar para el acceso a archivos remotos dentro del sitio [10].

El protocolo de Entrada/Salida de Archivo Remoto (RFIO): este protocolo se divide en dos tipos, el RFIO inseguro y el RFIO seguro (gsirfio). El primero permite el acceso a sistemas de archivo en cinta, como CASTOR (CERN Advanced Storage manager) y, por consiguiente, sólo se puede usar para acceder a datos desde cualquier WN dentro de la Red de Área Local (LAN) y sólo se puede autenticar a través del UID y GID. Por otro lado, el RFIO seguro se puede usar para el acceso de archivos a cualquier almacenamiento de red remota y también desde una UI [11].

2.3 Almacenamiento en Grid

El aumento de la potencia de cálculo ha creado la oportunidad para las nuevas simulaciones científicas, las cuales son más precisas y complejas conduciendo a nuevos conocimientos científicos. Del mismo modo, los experimentos con grandes volúmenes deben generar cada vez mayor cantidad de datos los cuales tienen que ser destinados a los discos de almacenamiento. Para realizar este tipo de procesos en Grid es necesario utilizar una interfaz que administre los recursos que están almacenados y en Grid se utiliza la interfaz que se explica a continuación.

2.3.1 Interfaz del administrador de recursos de almacenamiento (SRM)

El administrador de recursos de almacenamiento (SRM) se ha diseñado como la única interfaz (a través del protocolo SRM correspondiente) para la gestión de los recursos de almacenamiento de disco y cinta.

Eventualmente, cualquier tipo de elemento de almacenamiento ofrecerá una interfaz SRM que ocultará la complejidad de los recursos, y le permitirá al usuario consultar archivos, bloquearlos por un período de duración específico y reservar espacio para nuevos registros. Detrás de esta interfaz, el SE tendrá una política definida del sitio específico, según la cual, los archivos se migrarán de disco a cinta, se les permitirá a los usuarios leer y escribir. Los SRMs también serán capaces de encargarse de la solicitud de transferencias de un SRM a otro (copia para un tercero). Es importante recalcar que el protocolo SRM es un protocolo de manejo de almacenamiento y no un protocolo de acceso o transferencia de archivo. Para estas tareas, la aplicación cliente accederá directamente al acceso de archivo o al servidor de transferencia apropiado [12].

2.3.2 Tipos de elementos de almacenamiento

Existen diferentes tipos de SEs en gLite 3.1

SE clásico: consiste en un servidor con un solo disco físico o un arreglo de disco. El SE clásico sólo puede ser llenado por una VO y no soporta la interfaz SRM por tanto está desapareciendo [9].

dCache Disk pool manager: consiste en un servidor dCache y uno o más nodos pool. El servidor representa el único punto de acceso al SE y presenta archivos en el arreglo de discos bajo un sólo árbol de sistema de archivo virtual. Los nodos se pueden añadir dinámicamente al arreglo de discos. Éste presenta una interfaz SRM que permite superar las limitaciones del SE Clásico [13].

Disk pool manager de LCG: es la alternativa ligera de LCG para dCache. Es fácil de instalar y

aunque no es tan versátil como dCache, ofrece toda la funcionalidad que requieren los sitios pequeños. Los discos se pueden añadir dinámicamente el arreglo de discos en cualquier momento. Tal como en dCache, un sistema de archivo virtual oculta la complejidad de la arquitectura del pool de disco [14]. También presenta una interfaz SRM. Además, soporta la asignación de cuota de disco por VO. Debido a esto, una vez que se despliega el DPM, este remplaza al SE clásico. Los SEs clásicos se están convirtiendo en DPMs con un solo disco en el pool.

CASTOR: consiste en un sistema de almacenamiento masivo de cintas. Su nombre proviene de CERN Advanced Storage manager. Tiene un sistema de archivos virtual que abstrae las complicaciones existentes del manejo de discos y cintas. Soporta únicamente el protocolo inseguro RFIO, por lo cual, sólo puede ser accedido localmente dentro de la misma red local del SE. CASTOR cuenta con un diseño modular con una base de datos central para el manejo de información de los archivos, los cinco módulos que los componen son: el Stager, el Servidor de Nombres, cintas de almacenamiento, una base de datos y el cliente [15].

2.4 Sistemas de archivos

El tratamiento de datos es uno de los aspectos más importantes que debe considerar cualquier middleware Grid, debe cubrir aspectos como: la localización, transparencia y transferencia de archivos, así como la seguridad implicada en todo el manejo de datos. Se pueden clasificar en sistemas de archivos distribuidos y en paralelo [16].

Sistemas de archivos distribuidos: con la aparición de las redes de interconexión, surgió la necesidad de compartir datos entre diferentes máquinas, debido a la economía o la naturaleza de algunas aplicaciones, situación que originó la aparición de los sistemas de archivos distribuidos (DFS), estos son una implementación distribuida del clásico modelo de tiempo compartido de un sistema de archivos, donde varios usuarios comparten archivos y almacenan recursos [17].

Sistema de archivos en paralelo: en los sistemas de archivos distribuidos, los archivos se almacena en un servidor, y el ancho de banda de acceso a un archivo se encuentra limitado por el acceso a un único servidor. Este problema es originado por el desequilibrio existente entre el tiempo de cómputo y el tiempo de E/S (Entrada/Salida) [18]. La anterior situación se soluciona con los sistemas de archivos paralelos, ya que estos utilizan paralelismo en el sistema de E/S con la distribución de los datos de un archivo entre diferentes dispositivos o servidores. Esto evita los cuellos de botella en los procesos de E/S, ya que se accede de forma paralela a un archivo, mejorando el rendimiento al hacer un mejor uso del ancho de banda del sistema total.

2.5 Tipos de sistemas de archivo

GLUTERFS: es un sistema de archivos para cluster de alto rendimiento que soporta múltiples tipos de datos, compatible POSIX, escalable y altamente distribuido, puede almacenar petabytes de datos cubriendo necesidades de cluster de alto rendimiento [19].

PVFS2: es una implementación open source de un sistema de archivos paralelo, desarrollado específicamente para clustersBeoWulf y para sistemas operativos Linux. Es un sistema de archivos paralelo que permite a las aplicaciones paralelas y seriales almacenar y recuperar datos desde de un conjunto de nodos o servidores de I/O unidos a través de la red, el diagrama del sistema PVFS se puede observar en la figura 2. El éxito de los PVFS es el alto rendimiento que tiene cuando se escriben archivos de gran tamaño [20].

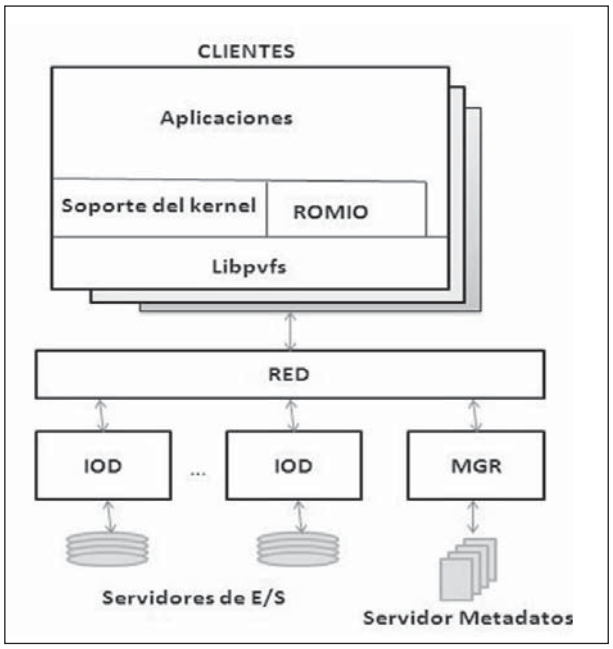

Figura 2. Diagrama del sistema PVFS.
Fuente: The Parallel Virtual File System for High Performance Computing Clusters, 2002.

Como se observa en la figura 2, las máquinas que integran el sistema PVFS pueden tomar uno o más de los siguientes roles:

1) Servidor de metadatos: aquí se mantiene la información de los archivos y directorios que están almacenados en el sistema PVFS, tal como permisos, dueños, localización de los datos.

2) Servidor de E/S: es en estos servidores en donde se almacenan los datos, pueden existir varios de estos.

3) Soporte para el kernel: provee la funcionalidad para montar sistemas PVFS en los nodos Linux, lo que permite a los programas existentes acceder a los datos almacenados en PVFS sin modificaciones.

Lustre: es un sistema de archivos para clúster iniciado por Carnegie Mellon University en el año 1999, proponiendo una arquitectura basada en objetos. Cuenta con tres elementos básicos: el cliente, que es usado para acceder al sistema de archivos; el servidor de almacenamiento de objetos (OSS), que provee el servicio I/O; y servidores de metadatos (MDS), los cuales manejan los nombres y directorios dentro del sistema de archivos [21]**.** El total de almacenamiento manejado por un OSS es separado en volúmenes, la capacidad de cada volumen varia de 2 a 8 terabytes. Cada OSS está encargado de manejar múltiples Object Storage Target (OST), uno por cada volumen.

3. METODOLOGÍA

El trabajo que se presenta en este artículo consistió en el desarrollo de un sistema de almacenamiento que se acoplara a los recursos existentes en el campus universitario, a través del uso del sistema de archivosPVFS2.

Actualmente, la Universidad Industrial de Santander hace parte del proyecto IGALC, ofreciendo el servicio de procesamiento a los grupos de investigación, lo cual hace necesario la implementación de una infraestructura de almacenamiento. Este trabajo propone la implementación de una infraestructura de almacenamiento masivo basada en una estrategia recursiva utilizando los nodos de trabajo como discos servidores de PVFS2, la cual es factible dado el poco uso de éste durante las labores de cálculo intensivo. Esta estrategia ahorra grandes sumas de dinero en la compra de nuevos equipos.

3.1 Implementación del sitio

Esta arquitectura consta de 10 nodos locales y 4 externos, como se puede observar en la figura 3. Dos de la Universidad Federal de Rio de Janeiro, otro de la UFF Latin American and Caribbean Catch all Grid Certification Authority, en Brasil y uno del CIEMAT (Centro de Investigaciones

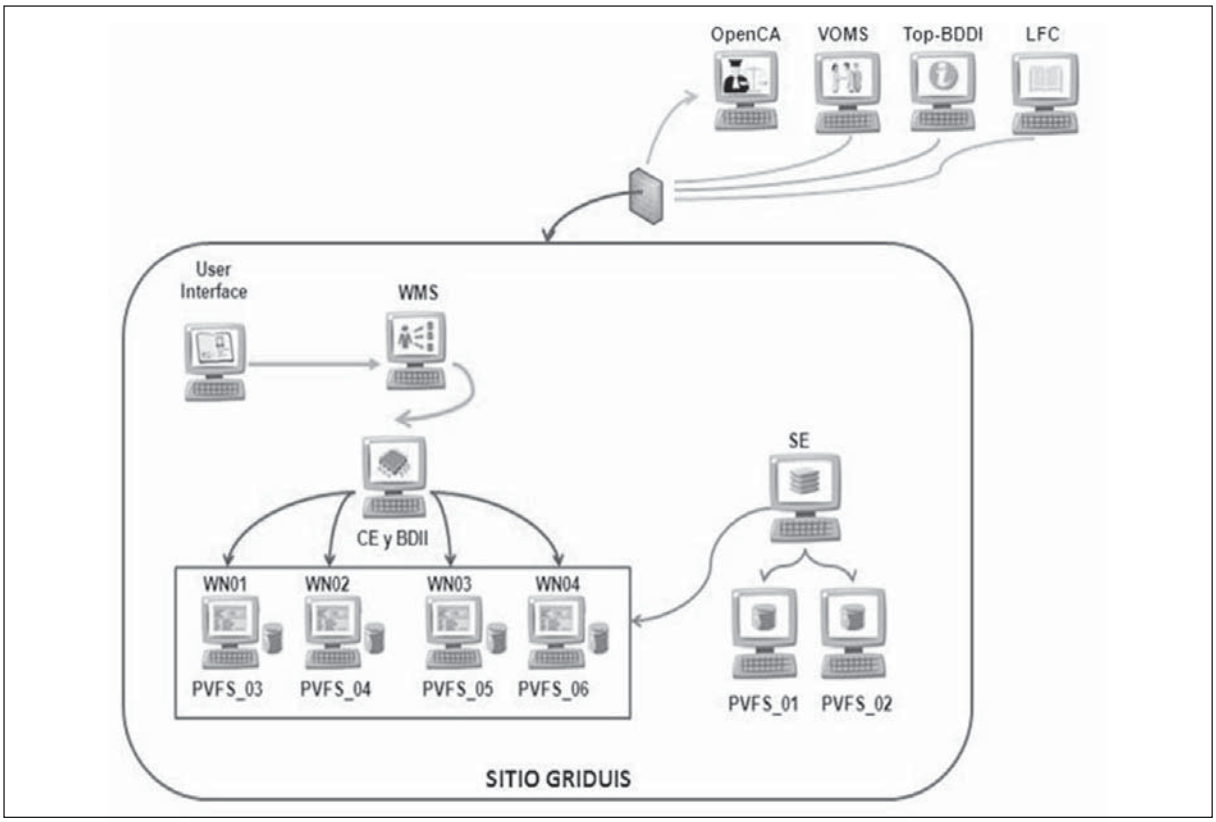

Figura 3. Estructura general del Sitio.
Fuente: elaboración propia

Energéticas, Medo Ambientales y Tecnológicas), en España.

Cada uno de ellos conforma uno de los componentes del middleware gLite para establecer una infraestructura distribuida en la Universidad Industrial de Santander. Por lo tanto, la interacción entre cada uno de estos nodos es vital a la hora de almacenar y procesar datos.

3.1 Implementación de la arquitectura de almacenamiento

La unidad primaria para el manejo de datos en Grid, al igual que en la informática tradicional, es el archivo; en un entorno Grid, los archivos pueden estar replicados en diferentes lugares. Debido a que todas las réplicas deben ser coherentes, los archivos no pueden modificarse después de su creación, solo leerlos y eliminarlos. Idealmente, los usuarios no necesitan saber dónde está ubicado un archivo, ya que este servicio se encarga de manejar las tareas de almacenamiento, transacción, control e información de los datos a usar en los trabajos. Para esto, se soporta en elementos de almacenamiento (storageelement), los cuales son servidores o dispositivos con capacidad de almacenamiento masivo, el FileCatalogocatálogo de archivos permite la ubicación de determinado archivo que necesite un trabajo y protocolos especiales de transferencia de archivos.

El elemento de almacenamiento (SE) implementado fue el Disk Pool Manage (DPM), el cual se apoya en un catálogo de archivos llamado Logical File Catalog (LFC) y los protocolos de trasferen-

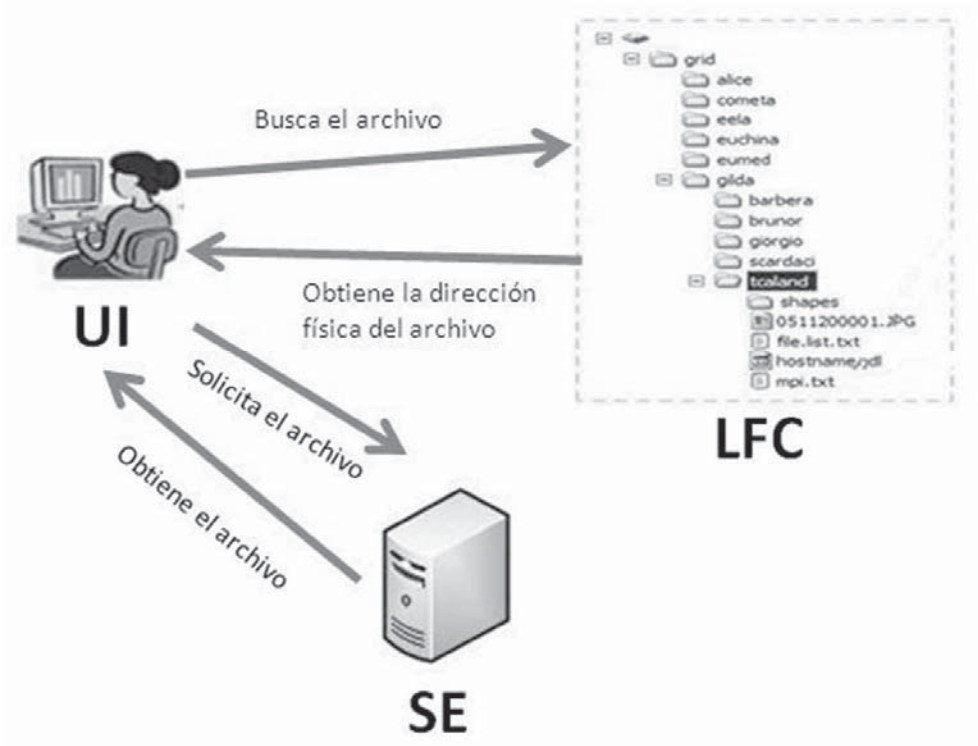

Figura 4. Envío y consulta de datos desde el SE y la UI.
Fuente: elaboración propia

cia y acceso de archivos usados fueron GridFTP y RFIO. Para esta implementación se usaron en total ocho equipos que están ubicados en la UIS, seis de ellos son los discos de almacenamiento, otro es el SE y un último equipo que presta el servicio de catálogo de archivos LFC que se encuentra en Brasil.

El procedimiento general que realiza un usuario para consultar y transferir datos desde el SE a la UI se muestra en la figura 4.

Desde la UI el usuario accede al LFC a través de línea de comandos para obtener la dirección física del archivo, una vez obtenida, el usuario actúa directamente con el SE y por medio de los protocolos de transferencia recibe el archivo para ser visto desde la UI.

El funcionamiento interno del sistema de almacenamiento se explica más detalladamente en la figura 5.

- El cliente SRM pregunta al servidor SRM por espacio disponible o por el archivo.
- El servidor SRM pregunta al SE (head node) por espacio disponible o por el archivo.
- EL Se (head node) notifica al servidor SRM la disponibilidad de espacio o la dirección del archivo.
- El servidor SRM comunica al cliente SRM la información.
- El cliente SRM interactúa con el SE (head node).

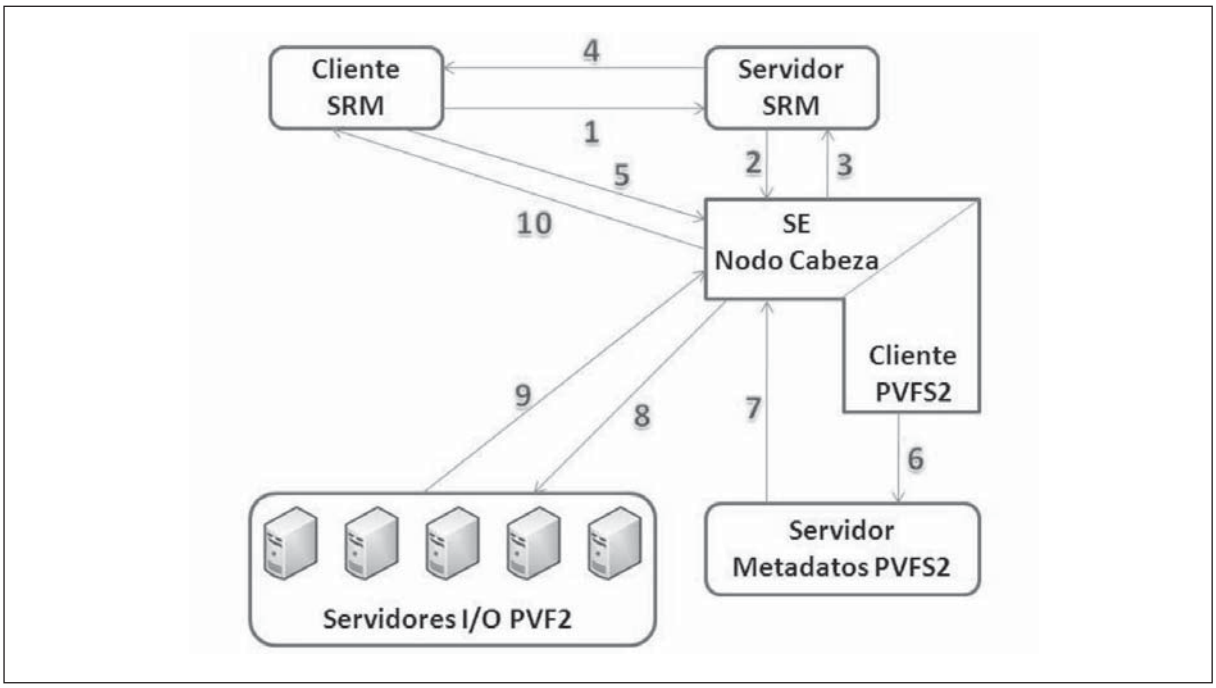

Figura 5. Funcionamiento de la arquitectura de almacenamiento
Fuente: elaboración propia

- El cliente PVFS2 pregunta a su servidor de metadatos por espacio disponible o la ubicación del archivo.
- El servidor de metadatos notifica al SE (head node), si es posible almacenar, o devuelve la dirección física del archivo.
- El SE (head node) accede directamente al servidor PVFS2 para leer o escribir archivos.
- Los servidores PVFS2 devuelven al SE el archivo buscado o la dirección donde queda almacenado.
- Finalmente, el cliente SRM obtiene el archivo o la dirección donde quedó almacenado.

4. PRUEBAS Y RESULTADOS

Las pruebas de almacenamiento fueron evaluadas en equipos con las siguientes características: CPU Intel Pentium 4, 3.2 GHz, 2 GB de memoria RAM, 160 GB de disco duro; todas las máquinas conectadas por una red Gigabit.

4.1 Pruebas de lectura y escritura de archivos

Esta prueba consistió en medir el tiempo de lectura y escritura de archivos entre 5 y 50 GB, se hizo en comparación de una implementación con ext3 y la nueva con PVFS2, con el fin de observar la pérdida o ganancia de desempeño percibida por el usuario, al momento del almacenamiento. Se compara con ext3 porque es un sistema de archivos con estructura simple, es el más usado en distribuciones Linux y ofrece un excelente desempeño. Las pruebas de escritura se realizaron descuerdo al siguiente orden:

1. Se accede a la UI por medio de protocolo SSH y se genera un certificado proxy temporal para poder interactuar con las partes de la Grid.

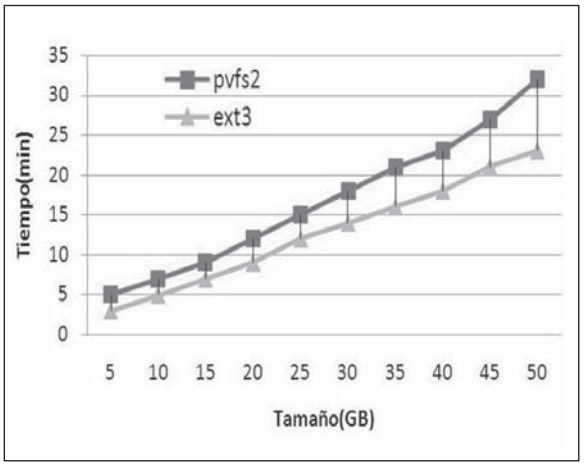

Figura 6. Tiempo de escritura de archivos.
Fuente: elaboración propia

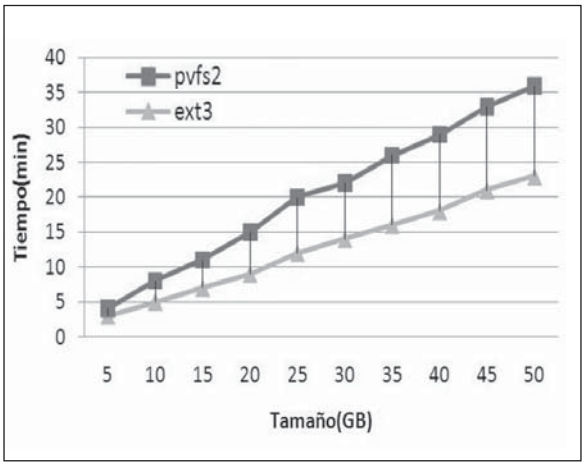

Figura 7. Tiempo de lectura de archivos.
Fuente: elaboración propia

2. En el home de la cuenta de usuario se crea el archivo con el siguiente comando:

   timedd if=/dev/zero of=test5gb count=5000 bs=1M

3. Se envía el archivo para que sea almacenado en el SE de la siguiente manera:

   lcg-cr -d se.uis.edu.co -l lfn://Grid/prod.vo.eu-eela.eu/UIS/test5gb –v file:///home/carlos/test5gb

4. Se captura la salida del comando anterior, y se obtiene el tiempo que tarda en crear el archivo en el SE. Los resultados obtenidos del tiempo de escritura de un archivo de acuerdo a su tamaño se pueden observar en la figura 6.

A continuación, se muestra la secuencia que se realizó para la prueba de lectura:

1. Se accede a la UI por medio de protocolo SSH y se genera un certificado proxy temporal para poder interactuar con las partes de la Grid.

2. Se observa el archivo que se desea traer desde el SE y se obtiene la dirección lógica del LFC con el siguiente comando:

   lfc-ls /Grid/prod.vo.eu-eela.eu/UIS

3. Una vez obtenido el nombre lógico, se procede a hacer la transferencia de la siguiente manera:

   timelcg-cp lfn://Grid/prod.vo.eu-eela.eu/UIS/test5gb file:///home/carlos/test5gb

4. Se captura el tiempo que nos muestra la salida del comando. Los resultados obtenidos del tiempo de lectura de acuerdo a su tamaño se pueden observar en la figura 7.

En las figuras 6 y 7 se observa el tiempo que tarda cada archivo en ejecutar la secuencia vista en la figura 5. La nueva implementación con el sistema de archivos PVFS2 ofrece un rendimiento eficiente, no muy distante del sistema de archivos ext3 en el momento de la escritura, sin embargo, en la lectura de datos la gráfica muestra una diferencia de desempeño alrededor de (60%) a favor de ext3.

4.2 Pruebas de tráfico de la red

Esta prueba consistió en medir el tráfico de la red en los discos servidores de PVFS2 y en el SE, que es el nodo cabeza, en el mismo instante que

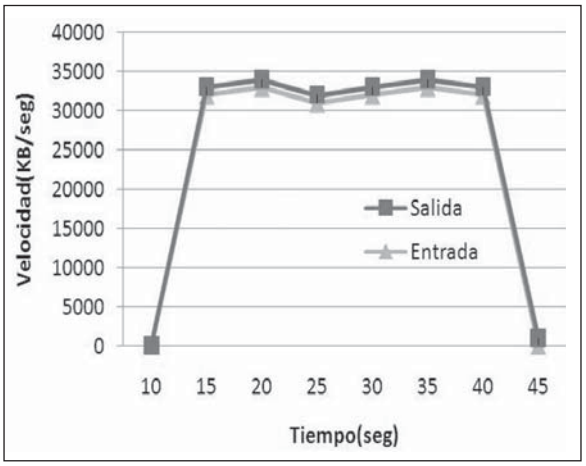

Figura 8. Tráfico de red en él SE (head node).
Fuente: elaboración propia

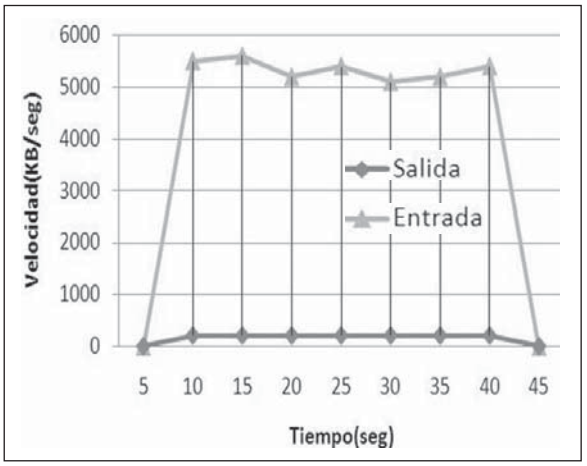

Figura 9. Tráfico de red PVFS2 server.
Fuente: elaboración propia

se hace la escritura de un archivo de 1 GB, esto se lleva a cabo para conocer el uso de red de los diferentes equipos que conforman el sistema de almacenamiento.

En la figura 8 se muestra que la red del nodo cabeza (SE) se satura, tanto en la entrada como en la salida. Mientras en la figura 9, la red de entrada se satura una sexta parte de su capacidad máxima.

La relación entre la tasa de ancho de banda consumido de la salida de SE y la entrada del servidor PVFS2 está dada por:

$$TePVFS2 = TsSE / N \text{ [1]}$$

$$TePVFS2 = 32683,12/ 6 = 5447,02 \text{ [2]},$$

Donde TePVFS2 corresponde al ancho de banda de entrada consumido durante la creación de un archivo en el servidor PVFS2, TsSe es el ancho de banda de salida consumido por el SE head node y N es el número de discos servidores PVFS2.

Para este caso el ancho de banda consumido durante la transferencia, es una sexta parte (1/6) de su capacidad total, lo cual permite el normal desarrollo de otras actividades de red. Además de esto, el ancho de banda consumido en cada servidor PVFS2 disminuye en la medida que se aumente el número de estos.

4.3 Envío de trabajos con datos de entrada y de salida

Esta prueba consistió en el envío dos trabajos: uno de ellos con la necesidad de un fuente de datos de entrada, en este caso, datos traídos desde el SE; y otro con datos de salida que serán almacenados en el SE. Esta prueba se hace para observar el buen funcionamiento del sistema y ver si cumple con el servicio que se desea prestar a los grupos de investigación interesados en ejecutar sus aplicaciones en la Grid.

4.4 Envío de trabajos con datos de salida

El siguiente trabajo consistió en tres scripts: JobEscribeEnSE.*jdl* que es usado para describir los trabajos a ejecutar en la Grid, scriptQueHaceAlgo.sh es un script que se encarga de llenar un archivo de texto y *scripQueRegistraElArchivo.sh* que es el encargado de registrar el archivo en el SE. A continuación se muestra la secuencia de los pasos que se hicieron para esta prueba:

1. Se accede a la UI por medio de protocolo SSH y se genera un certificado proxy temporal con su respectivo tiempo de ejecución del trabajo.
2. Se crean los scripts: *JobEscribeEnSE.jdl*, *scriptQueHaceAlgo.sh* y *scriptQueHaceAlgo.sh*.
3. Se hace el envío del trabajo y se espera a que termine la ejecución.
4. Finalmente, se solicita el resultado del trabajo, el cual devuelve la dirección física y lógica del archivo que se creó en el SE.

4.5 Envío de trabajos con datos de entrada

Para el envío de este trabajo fue necesario la creación de dos scripts: InputData.jdl que es el script principal reconocido por la Grid y scripInput.sh que es el encargado de crear el ambiente necesario en el WN para traer el archivo desde el SE. Estas pruebas se realizaron de acuerdo al siguiente proceso:

1. Se accede a la UI por medio de protocolo SSH y se genera un certificado proxy temporal con su respectiva delegación de recurso para poder ejecutar el trabajo.
2. Se crea el archivo *ResultadosFinal.txt* en el SE, el cual será usado en la ejecución del trabajo.
3. Se crean los scripts: *InputData.jdl* y *scripInput.sh*
4. Se hace el envío del trabajo y se espera a que termine la ejecución.
5. Finalmente, se solicita el resultado del trabajo, el cual devuelve los datos de salida del trabajo ejecutado.
6. En esta prueba el 100% de los datos que fueron enviados a procesar se recuperaron con su respectiva respuesta.

4.6 Recuperación de archivos

Esta prueba consistió en la alteración del funcionamiento del sistema de almacenamiento, simulando una caída de energía eléctrica en todo el sistema. Se hace con el fin de saber cuál es el comportamiento de la arquitectura y ver qué tan difícil es la recuperación de archivos en el sistema, ya que no se cuenta con una UPS que respalde el flujo de electricidad.

Se suspende el servicio de electricidad. En el momento que regresa la electricidad, los equipos se encienden automáticamente, en los discos servidores PVFS2 es necesario iniciar el servicio, mientras que en el SE es necesario iniciar el PVFS2 cliente, de esta manera el servicio está disponible nuevamente.

En esta prueba se encontró que los archivos que han sido almacenados antes del corte de electricidad no son alterados y simplemente con iniciar los servicios PVFS2 están disponibles. Sin embargo, los archivos que han sido transferidos en el momento del corte son irrecuperables y la única forma de recuperarlos es volviendo a iniciar la transferencia de mismo.

5. CONCLUSIONES

El sistema de almacenamiento masivo es funcional y ofrece un componente adicional de escalabilidad, permitiendo así la integración de nuevos componentes en la medida que sean necesarios. La implementación presentada en este trabajo permite aumentar la capacidad de almacenamiento del sistema, utilizando recursos existentes de la universidad, sin incurrir en la adquisición de nuevos equipos.

El hacer uso de equipos dedicados a múltiples actividades como servidores PVFS2, no se obstruye el desarrollo normal de otras actividades de

red, ya que ésta no utiliza toda su capacidad. La implementación de sistemas de archivos como PVFS, permite que el proceso de almacenamiento en Grid sea transparente para el usuario, dando la apariencia de que los archivos son almacenados en un único disco.

En la implementación de la arquitectura de almacenamiento se usaron únicamente herramientas de software libre, minimizando los costos del proyecto al no tener que incurrir en el pago de licencias de uso privativo.

REFERENCIAS

[1] B. Jacob, M. Brown, K. Fukui and N. Trivedi, *Introduction to Grid computing*, IBM Redbooks, December, 2005.

[2] F. Berman, G. Fox, and A. Hey, *Grid Computing, making the global infrastructure a reality*, Ed Wiley, March, 2003, Chapter 1 and 2.

[3] V. Sipkova, and V. Tran, "Glite Lightweight Middleware for Grid Computing", in *The 2nd International Workshop on Grid Computing for Complex Problems (GCCP'2006)*, pp. 228-241, April, 2006.

[4] *GLite: La nueva generación de Middleware para el Grid de EGEE*, (15 Sept. 2008), [Online]. Available: http://www.eu-egee.org/.

[5] S. Burke, S. Campana, A. Delgado, F. Donno, P. Méndez, R. Santinelli, and A. Sciaba, *Manual de usuariogLite 3*, CERN, March, 2006, Chapter 3.2.3.

[6] *Overview of the Grid Security Infrastructure,* (20 Jan 2009), [Online]. Available: http://www-unix.globus.org/security/overview.html.

[7] P. Fuhrmann, (20 Jan 2009), *Dcache, the overview*, [Online]. Available: http://www.dcache.org/manuals/dcachewhitepaper-light.pdf.

[8] S. Ulrich, and A. Heiss, *First experience with the InfiniBandinterconnect, sciencedirect*, Ed Karlsruhe: Alemania, 2004.

[9] S. Burke, S. Campana, A. Delgado, F. Donno, P. Méndez, R. Santinelli, and A. Sciaba, *Manual de usuariogLite 3*, CERN, March 2006, Chapter 7.

[10] GSIFTP, (12 Jan 2010), *GSIFTP Tools for the Data Grid*, [Online]. Available: http://www.globus.org/toolkit/docs/2.4/dataGrid/deliverables/gsiftp-tools.html.

[11] RFIO, (2 Feb 2010), *Remote File Input/Output*, [Online]. Available: http://doc.in2p3.fr/doc/public/products/rfio/rfio.html.

[12] S. Burke, S. Campana, A. Delgado, F. Donno, P. Méndez, R. Santinelli, and A. Sciaba, *Manual de usuario gLite 3.1*, CERN, October, 2008, Chapter 7.

[13] Dcache, (3 Feb 2010), *Dcache, Theoverview*, [Online], Available: http://www.dcache.org/.

[14] CERN, (3 Feb 2010), *DPM Admin Guide*, [Online], Available: https://twiki.cern.ch/twiki/bin/view/LCG/DpmAdminGuide.

[15] CERN, (3 Feb 2010), *Castor - cern advanced storage manager*, [Online]. Available: http://castor.web.cern.ch/castor/.

[16] J. Joshy and F. Craig, *Grid Computing*, IBM Press, April, 2003.

[17] A. S. Tanenbaum, *Sistemas Operativos Distribuidos*, Prentice Hall Hispanoamérica S.A, 1st Ed, Septiembre, 1998, Capítulo 5.

[18] Y.N. Patt, *The I/O subsystem: A candidate for improvement*, April, 1994, Chapter 15-16.

[19] M. Barrios, T. Jones, S. Kinnane, M. Landzettel, S. Al-Safran, J. Stevens, C.Stone, C. Thomas, and U. Troppens, *Sizing and Tuning GPFS,* IBM Red Book, July, 1999.

[20] M. Kashyap, J. Hsieh, C. Stanton and R. Ali, *The Parallel Virtual File System for High Performance Computing Clusters*, July, 2002.

[21] Lustre, (3 Feb 2010), *Lustre File Systems,* [Online], Available: http://wiki.lustre.org/index.php.