MAGOS: Simplificación del Desarrollo de Aplicaciones *Datagrid*

MAGOS: Simplifying Datagrid Application Development

Claudia L. Jiménez Guarín[1], Marcela Cañón Vargas[1], Juan D. Cardona Marín[1], Vladimir López Borja[1]

[1] *Universidad de Los Andes, Depto. de Ing. de Sistemas y Computación, Bogotá, Colombia*

{ cjimenez, mar-cano, dav-card, v-lopez } @uniandes.edu.co

Resumen

Este trabajo presenta MAGOS *– Middleware Architecture for Grid Oriented Services.* MAGOS *ofrece al desarrollador de aplicaciones orientadas a servicios (SOA) la posibilidad de describir de forma declarativa los requerimientos no funcionales que espera de un grid, indicando los recursos necesarios, características de privacidad, replicación de servicios y de fuentes de datos.* MAGOS *se encarga de ubicar y asignar los recursos en el grid*[2]*, instala las aplicaciones y servicios y construye un catálogo que permite su posterior ubicación. Finalmente, permite orquestar dinámicamente aplicaciones existentes de acuerdo con su disponibilidad real y con las condiciones del grid en el momento de ejecución.* MAGOS *reduce la curva de aprendizaje para el desarrollador SOA y facilita la instalación y la cooperación de aplicaciones autónomas en el grid*[3]*.*

Palabras clave: aplicaciones orientadas a servicios, computación en malla, mallas de datos, middleware, Globus

Abstract

MAGOS *– Middleware Architecture for Grid Oriented Services, offers to the developer of service oriented applications a descriptive way to declare the non functional requirements for installing and executing grid-enabled applications. The developer indicates the required resources, privacy features and replication requirements on services and data sources.* MAGOS *installs and locates complex services using a catalog. Finally, it offers a workflow definition functionality, which allows orchestrating existing applications in agreement with its actual availability on the grid and to the actual grid load.* MAGOS *helps reducing the learning curve for the SOA developer to interact with a grid, installing and running applications in a good context for cooperation between autonomous, heterogeneous datagrid applications.*

Keywords: *grid services, grid computing, datagrid, middleware,* Globus.

1. Introducción

La infraestructura de cómputo en malla (*grid computing)* [8][9][13][22] ha sido uno de los grandes logros de la computación distribuida y paralela en los últimos tiempos. Su utilización se ha dado principalmente en el contexto de la investigación, donde grandes proyectos han logrado ampliar de forma virtualmente infinita su capacidad de almacenamiento y de cálculo. Entre aquellos proyectos orientados a la gestión de datos podemos mencionar [12][18][19][20][23].

Las mallas de cómputo (*grid*) concebidas para la manipulación de datos, *datagrid* [15][21][24][30], ofrecen una posibilidad interesante en el caso de aplicaciones que, siendo independientes y autónomas en su desarrollo, necesitan compartir servicios o información. Esta situación es particularmente interesante en el caso de entidades de un sector, en el cual las necesidades de cooperación son relevantes, pero cada uno quiere guardar su independencia frente a su disponibilidad, ubicación, tecnología, servicios expuestos o restricciones de confidencialidad. La promesa de esta tecnología es lograr escalabilidad en proporciones importantes, más allá de la capacidad que una sola entidad podría posiblemente adquirir por sí

[2] Los autores utilizamos en este trabajo terminología técnica en inglés, por ser las de uso corriente en nuestro medio. Las palabras y siglas *grid, datagrid, middleware, workflow, api, deploy, browsing, script*, por nombrar las más frecuentes, son dejadas en ese idioma de forma intencional.

[3] Este proyecto se inscribe en el marco del proyecto Ecos-Colciencias "Desarrollo de una infraestructura de GRID para cálculo y datos - código C07M02"

misma dado el nivel de inversión requerido. La capacidad buscada puede ser potencia de cálculo, espacio de almacenamiento o disponibilidad de aplicaciones [4] [24].

Este es un escenario interesante en muchos sectores de países en desarrollo, como el sector gobierno, educación, salud, por nombrar apenas algunos. Encontrar casos exitosos de uso de tecnología de *grid* en sectores de gestión del sector salud [1] [10] nos anima en búsqueda de soluciones efectivas que puedan ser adoptadas por las entidades colombianas. A diferencia de los casos expuestos en dichos trabajos, donde usualmente el sistema de salud es regulado y ofrecido por el estado, el sistema de salud colombiano es mixto, con una porción significativa del sector privado. Cada institución puede establecer sus criterios, dentro de un marco legal, lo cual permite un grado de autonomía y heterogeneidad que se ajustan a convertirse en un caso real en el cual se aplica nuestra propuesta: Simplificar el esfuerzo necesario para lograr el uso de la tecnología de grid por parte de aplicaciones heterogéneas y autónomas, que cooperan en un *datagrid*.

En el estado actual de la tecnología hay un sinnúmero de aplicaciones que usan de forma intensiva un *grid*, con fines principalmente de almacenamiento o de cómputo. Sin embargo, el uso de *grid* aún es lejano a una inmensa masa de desarrolladores de aplicaciones en el sector productivo. La curva de aprendizaje, así como la infinidad de detalles que permiten el uso real de un *grid*, aleja a muchos de ellos. Las aplicaciones desarrolladas, al momento de ser instaladas quedan profundamente acopladas con la tecnología *grid* que se utiliza y el conocimiento es difícilmente transmisible. Estos son los problemas que originaron el trabajo que se presenta en este artículo.

Este trabajo presenta **MAGOS, *Middleware Architecture for Grid Oriented Services***, una herramienta que facilita a desarrolladores de aplicaciones orientadas a servicios el uso de un *datagrid*, *m*ediante un proceso basado en la descripción declarativa de requerimientos no funcionales. MAGOS simplifica al desarrollador los detalles de la infraestructura tecnológica, le permite utilizar aplicaciones ya instaladas y cooperar con terceros garantizando seguridad y confidencialidad. El aporte principal de MAGOS es facilitar los procesos de descripción, instalación, ubicación, replicación y ejecución de aplicaciones que exponen servicios en un *datagrid*, de forma que respetan la heterogeneidad y autonomía de cada una de esas aplicaciones.

MAGOS está desarrollado sobre las tecnologías *Toolkit v4* (GT4) [26], y OGSA-DAI [31], estándares propuestos por *Globus Consortium*. OGSA-DAI permite el uso de servicios básicos de datos sobre GT4. MAGOS permite utilizar fuentes de datos implementadas sobre tecnología relacional, XML (eXtensible Markup Language) nativas, multimedia y archivos. Considera la vinculación de clientes móviles al *datagrid* y el uso de tecnología de apoyo en los servicios no funcionales como autorización, privacidad y confidencialidad.

En la parte 2 de este artículo se presenta el enfoque MAGOS para simplificar la tarea del desarrollador y se compara con trabajos de investigación que se ocupan de problemas similares. La parte 3 presenta las operaciones ofrecidas por MAGOS al desarrollador de aplicaciones. La parte 4 presenta la arquitectura computacional de MAGOS. La parte 5 presenta la comparación entre la instalación y orquestación de aplicaciones utilizando el enfoque MAGOS y el proceso que se debe llevar a cabo usualmente sobre GT4. Finalmente se presentan los resultados, trabajo futuro y conclusiones del trabajo.

2. MAGOS: el *datagrid* al alcance del desarrollador de aplicaciones orientadas a servicios

La tecnología *grid* actualmente está fundamentada en la concepción de las aplicaciones como grupos de servicios, basada en SOA (*Service Oriented Architecture* [28]) y compatible con marco de desarrollo de servicios *Web*. Actualmente las aplicaciones en arquitecturas orientadas a servicios no son monolíticas, parten del principio de cooperación e integración con otros servicios y/o sistemas.

La arquitectura OGSA (*Open Grid Services Architecture*) [25], propuesta por Globus, ofrece un marco de referencia para el desarrollo, instalación y uso de dichos servicios en el contexto de un *grid*. Para utilizar una aplicación intensiva en datos sobre un *datagrid* hay un largo camino por recorrer: Su instalación, así como su puesta efectiva en marcha, exige una comprensión detallada tanto de la tecnología Globus, como de los elementos de comunicación, de localización y uso de recursos y de fuentes de datos.

El aporte fundamental de MAGOS está en reducir este esfuerzo, en un paso adelante en la promesa de "conectarse al *grid* tan fácilmente como lo hace a la red de servicios de energía eléctrica o de teléfonos" [8]. MAGOS permite descubrir y orquestar los servicios instalados en el *grid* manteniendo su individualidad y autonomía. De esta forma, MAGOS ofrece la posibilidad real de un funcionamiento incremental de sistemas complejos sobre el *grid*, conformando un *workflow* de servicios.

Actualmente hay en curso muchos esfuerzos similares en cuanto a facilitar el uso del *grid* por parte de desarrolladores. En cuanto a proyectos que están desarrollando actualmente herramientas de *workflow*, podemos mencionar el proyecto WEEP (*Workflow*

Enactment Engine Project) de la Universidad de Viena [14]. Con respecto a lo presentado en nuestro trabajo, encontramos relevante destacar los siguientes aspectos: WEEP implementa un nuevo motor de *workflow* conforme a la especificación del lenguaje estándar `bpel` [29]; tiene un componente de invocación dinámica a nivel de operaciones, el cual requiere conocer la localización del archivo `wsdl` de un servicio para poder invocar alguna de sus operaciones y uno de sus objetivos es implementar una *api* para invocar de manera segura el servicio mismo del motor.

Otro proyecto que involucra definición de *workflow* sobre un *grid* es CoG Workflow desarrollado por Commodity Grid (CoG) [20] como parte de su *framework* (Java CoG kit). Allí se trabaja en el proyecto *Java CoG Kit Karajan Workflow*. Este proyecto consiste en el desarrollo de un nuevo motor *workflow* con un lenguaje propio y que es orientado a *jobs*.

En cuanto a propuestas de herramientas de administración, se puede mencionar *Introduce - Grid Service Authoring Toolkit* [11], realizado actualmente por el Departamento de Informática Biomédica de *Ohio State University*, en el contexto del desarrollo de un *datagrid* de diagnóstico de cáncer. Es un ambiente de desarrollo (IDE) que permite automatizar muchas de las tareas de *deploy* y acceso al *grid*. Esta herramienta se enfoca en la creación, modificación y *deploy* de servicios tipados, ya que entre otras razones, se debe asegurar que los servicios estén bien definidos para que haya interoperabilidad sintáctica entre ellos.

En cuanto a trabajos similares en manejo de replicación, podemos citar el realizado actualmente por el Instituto de Informática Academia de Ciencias de Eslovaquia, que se ocupa de sincronización de réplicas basados en servicios OGSA-DAI [6]. OGSA-DAI está desarrollando también mecanismos de replicación relacionales para grandes bases de datos heterogéneas [5][6].

Si bien en estas propuestas se enfrentan problemas similares, MAGOS presenta una aproximación que las integra y las complementa en un solo producto utilizando, en la medida de lo posible, tecnología estándar existente. De esta forma, a diferencia de CoG, MAGOS es totalmente orientado a servicios y se basa en el uso de `bpel`. A diferencia de WEEP, MAGOS integra un motor de *workflow* ya existente. MAGOS provee invocación dinámica de las aplicaciones, de forma que permite que la localización del servicio a invocar sea descubierta en el momento mismo de ejecución, teniendo en cuenta la carga del *grid* para ese instante. MAGOS permite que el usuario no requiera conocer la localización real del servicio para lograr su reutilización, inclusive para servicios desarrollados por terceros.

MAGOS ofrece al desarrollador los conceptos de **aplicación** y **servicio.** Una **aplicación** constituye una unidad funcional completa, autónoma, que tiene sentido por sí misma y cumple una tarea concreta desde el punto de vista de una organización. Una aplicación está compuesta de **servicios**, que son unidades funcionales que cumplen una tarea concreta, pero no necesariamente son una unidad funcional de negocio. MAGOS permite al desarrollador describir, de una forma muy sencilla, su aplicación, así como los requerimientos no funcionales que espera sean suplidos por el *grid*. A partir de esa descripción, MAGOS instala y permite utilizar tanto la aplicación como los servicios asociados.

A diferencia de Introduce, MAGOS no espera que el desarrollador se involucre con un IDE diferente al que usualmente utiliza, sino que complete su aplicación, describa los requerimientos de instalación y no requiera otro conocimiento.

MAGOS maneja los requerimientos no funcionales que los servicios *grid* demandan y se ocupa de la replicación de servicios y datos, así como del balanceo de carga en momento de la ejecución. MAGOS propone un esquema de replicación que es tolerante a fallas, que es sensible a replicación de datos o de servicios y que permite la inclusión de aspectos de confidencialidad y autorización sobre estos elementos.

MAGOS permite la integración de elementos de *workflow* y aspectos no funcionales y orquestación en momento de ejecución en una sola herramienta, utilizada de manera descriptiva por el desarrollador de aplicaciones. Esto permite concebir a MAGOS como un *middleware* de servicio sobre el cual se pueden apoyar aplicaciones de más alto nivel, aprovechando estas características. MAGOS está trabajando en la inclusión de servicios propios a aplicaciones multimedia y con clientes móviles, así como en la integración de servicios de manejo semántico de los esquemas de datos de la información que alberga el *datagrid*. La descripción de estos elementos está fuera del alcance de este artículo.

3. MAGOS: Cómo interactuar de forma sencilla con el *grid* para orquestar aplicaciones orientadas a datos, heterogéneas, autónomas

A continuación se describen las operaciones que MAGOS ofrece al desarrollador de aplicaciones. Se tienen cuatro puntos de interacción principales, que constituyen sus casos de uso:

- La definición de una nueva fuente de datos, si es necesario
- La declaración e instalación de una aplicación en el *grid*. Se especifica el propietario, servicios ofrecidos,

visibilidad de dichos servicios y esquema de replicación deseado. Este paso registra la aplicación en un catálogo MAGOS, para su gestión y ubicación posteriores.

- Los servicios instalados tienen un tipo, según su funcionalidad desde el punto de vista de los datos que maneja sobre los recursos utilizados: `Create`, `Query`, `Update` y `Compute`. Un servicio de tipo `Create`, se encarga únicamente de crear e inicializar una fuente de datos. Un servicio de tipo `Query`, interactúa con una fuente de datos para consultar información sin modificarla. Un servicio de tipo `Update` modifica la información de la fuente de datos. Un servicio de tipo `Compute` no hace uso de fuentes de datos.

- Búsqueda de servicios registrados en MAGOS, *browsing* sobre el catálogo acorde con los requerimientos de confidencialidad de los servicios expuestos, ubicación de los servicios y fuentes de datos en el *grid* para su ejecución.

- Orquestación en momento de ejecución de servicios ubicados en un *workflow* que permite la ejecución efectiva de los mismos de acuerdo con la carga y disponibilidad actual de los recursos del *grid*.

A continuación se presenta la descripción de cada una de estas funcionalidades.

3.1. Creación de fuentes de datos

Para crear un nuevo servicio de datos utilizando MAGOS, es necesario describir los parámetros requeridos por la fuente de datos que incluyen: tecnología requerida, proveedor y el *script* de creación del esquema. MAGOS se encarga de ubicar los nodos del *grid* que satisfacen estos requerimientos. En la Figura 1, se presenta un ejemplo de descripción de una aplicación tipo `Create`.

3.2. Instalación de aplicaciones

El descriptor que permite instalar una aplicación en MAGOS incluye la descripción de cada uno de los servicios que la componen y sus respectivas dependencias. Algunas opciones de esta descripción varían dependiendo del tipo de servicio. Se indican los recursos necesarios para correr adecuadamente el servicio, los requerimientos de replicación y de confidencialidad.

En el archivo de descripción de un servicio tipo `Create`, se indica el nombre de la aplicación, un identificador del descriptor, las características de la fuente de datos donde se crearán los datos incluyendo el `path` del *script* de creación. Se incluye también el número de réplicas de los datos descritos en el *script*.

```
<application>
  <name> crearBDPagos </name>
  <service>
    <id>pagosCreacion</id>
    <name>ServiciosCreacionPagos</name>
    <scope>0</scope>
    <BDArgument>
      <type>Relational</type>
      <vendor>MySQL</vendor>
      <version>4.0</version>
      <storageCapacity >10 </storageCapacity>
      <user> dbuser </user>
      <password> dbpassword</password>
      <pathScript>
            /home/user/data/script.sql
      </pathScript>
    </BDArgument>
    <replicate>
      <data min="2" max="2"/>
    </replicate>
    <type>Create</type>
  </service>
...
</application>
```

Figura 1. Ejemplo de descripción de un servicio tipo `Create` en MAGOS

En el archivo de descripción de un servicio tipo `Query` o `Update`, se indica el nombre de la aplicación, un identificador del descriptor, el `path` donde se encuentra el archivo `gar` que se va a instalar, la fuente de datos a la cual se va a conectar el servicio (este nombre es ubicado por el desarrollador previamente en el catálogo MAGOS), y el número de réplicas del servicio. Cabe anotar que un servicio de este tipo, en su instalación no describe réplica a nivel de datos.

```
<application>
  <name> pagos </name>
  <mainService>pagosRegistro</mainService>
  <requirements>
   <memoryUsage unit=MB>512</memoryUsage>
   <processUsage unit=GB>2.4</processUsage>
  </requirements>
  (...)
  <service>
   <id>pagosRegistro</id>
   <name> ServiciosRegistroPagos </name>
   <pathGar>
host.uniandes.edu.co/sed/ServRegPagos.gar
   </pathGar>
   <scope> 0 </scope>
   <depends> verificarAfiliacion </depends>
   <datasource> fuenteDatosPagos </datasource>
   <replicate>
     <service min="2" max="2" />
   </replicate>
   <type> Query </type>
  </service>
  <service>
   <id> verificarAfiliacion</id>
   <name> ServicioVerifAfiliacion</name>
   <pathGar>
host.uniandes.edu.co/sermed/VerificarAfiliacion.gar
   </pathGar>
   <scope> 0 </scope>
   <datasource>fuenteDeDatosPagos</datasource>
   <replicate>
     <service min="2" max="2" />
   </replicate>
   <type> QueryQuery </type>
  </service>
...
```

Figura 2. Ejemplo de descripción de un servicio tipo `Query` en MAGOS

La figura 2 muestra un ejemplo de descriptor de un servicio `Query`, donde se involucra un servicio dependiente (`verificarAfiliacion`).

3.3. Servicios de ubicación y exploración del catálogo MAGOS

En MAGOS el desarrollador cuenta con un catálogo de servicios y aplicaciones que va conformando a medida que estos se instalan. Este catálogo permite realizar búsquedas de aplicaciones que se encuentran disponibles para un cierto usuario en un momento dado y ver las dependencias entre sus servicios. La localización en el *grid* de esos recursos es transparente para quien consulte el catálogo [3].

3.4. Herramientas ejecución y orquestación de aplicaciones que cooperan con servicios instalados en el *grid*

El desarrollador debe definir su proceso de negocio en lenguaje `bpel`, utlizando ActiveBPEL[17]. Una vez definido el proceso, selecciona y asigna las aplicaciones disponibles usando el catálogo MAGOS para cada una de las actividades del *workflow*. MAGOS integra el motor de *workflow* con los servicios de ubicación, para hacer la orquestación y ejecución de la aplicación. El alcance de la expresividad en la definición de procesos en MAGOS es el ofrecido por ActiveBPEL.

En MAGOS las aplicaciones sólo conocen el nodo donde se ejecuta un servicio en el momento mismo de la ejecución. Esto permite que MAGOS provea la mejor opción para ejecución en las condiciones del *grid* en ese momento. El servicio de localización es el encargado de elegir el nodo y el de orquestación lo liga al código y pasa el control de ejecución al cliente en correspondencia directa con el servicio *grid*. MAGOS no interviene durante la ejecución del servicio: Es posible que entre una ejecución y otra del mismo *workflow* se orqueste la aplicación de forma diferente, bien sea por disponibilidad de las réplicas o por carga del *grid*. De esta forma, en el momento de ejecución, la eficiencia del proceso depende de la carga del grid y del código propio a la aplicación, no de MAGOS o de sus componentes de arquitectura.

De esta manera, MAGOS permite que las aplicaciones instaladas puedan cooperar siguiendo un flujo coherente de negocio sin necesidad de programar servicios adicionales. MAGOS permite que las definiciones de procesos realizadas en el *workflow* sean persistentes para su posterior consulta y ejecución.

4. Arquitectura computacional de MAGOS

MAGOS está construido en una arquitectura multicapas, orientadas a la realización de cada una de las tareas principales anteriormente descritas. La arquitectura de MAGOS es mostrada en la figura 3 y se describe con más detalle a continuación.

4.1. Capa de servicios: MAGOS-SERVICE

MAGOS-SERVICE se encarga de instalar cada servicio sobre el *grid* de forma independiente de otros servicios y siguiendo su descripción no funcional, según lo descrito en la parte 3. Garantiza la asignación de recursos a los servicios dependiendo de su tipo:

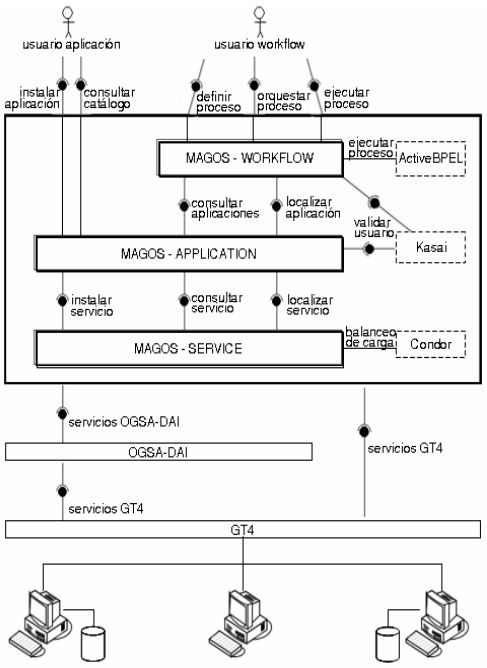

Figura 3. Arquitectura computacional de MAGOS

Si el servicio es de tipo `Create`, se buscan los servicios de datos que cumplan con los requerimientos de la fuente de datos. Es decir, si es necesario que la fuente esté replicada, MAGOS busca los servicios de datos para asignar y registrar la réplica principal con sus respectivas réplicas secundarias. El anterior paso, incluye la ejecución del *script* que se encarga de la inicialización de la fuente de datos.

Por otro lado, si el servicio es de tipo `Update`, `Query` o `Compute` y requiere replicación MAGOS-SERVICE busca los nodos *grid* disponibles para instalar el servicio y por último registrar la localización de dichas réplicas. En el momento de ejecución, ubica los servicios instalados en el *grid*. Si dichos servicios fueron instalados con replicación, se efectúa un balanceo de carga, mediante el uso de Condor [7], para seleccionar la mejor opción disponible. Por último, si un servicio usa una fuente de datos y es de tipo `Update`, la replicación ubica la réplica principal y se encarga de la sincronización de las dependientes. En caso de no tener disponibilidad de la réplica principal, de forma dinámica cambia esta característica hacia alguna de las réplicas disponibles [16]. De esta forma, MAGOS-SERVICE es la responsable de los

requerimientos no funcionales de disponibilidad y transparencia a la ubicación.

MAGOS-SERVICE es un servicio grid que permite instalar y ejecutar un servicio. Se seleccionan los nodos necesarios de acuerdo con los requerimientos, para luego efectuar un *deploy* remoto. En la ejecución de un servicio se realiza balanceo de carga que selecciona el nodo más apropiado. La replicación de datos usa OGSA-DAI, realizando las operaciones de actualización sobre la réplica principal y se propaga la operación hacia las réplicas dependientes.

4.2. Capa de instalación y registro de aplicaciones: MAGOS-APPLICATION

MAGOS-APPLICATION se encarga de instalar aplicaciones sobre un *grid*. Una aplicación está compuesta de uno o más servicios, cuyas dependencias son descritas en un archivo XML. Cada servicio describe sus requerimientos específicos y es instalado de manera independiente en el *grid*. La aplicación sólo queda instalada si es exitosa la instalación completa de todos sus servicios.

Como resultado de la instalación y registro de una aplicación, MAGOS-APPLICATION la almacena en un catálogo, conservando la información de las dependencias entre servicios. El catálogo también almacena información sobre el ámbito de los servicios instalados: `público`, que permite que todo usuario pueda visualizar estos servicios; `autorizado`, que habilita su uso sólo a algunos usuarios; y `privado`, que restringe su uso al propietario.

En el momento de ejecución, MAGOS-APPLICATION ubica el esquema de servicios y solicita a MAGOS-SERVICE la localización de las mejores opciones de ejecución para cada uno de ellos. Obtiene la `uri` de cada servicio componente y los asocia en tiempo real en las máquinas correspondientes. Finalmente, entrega la `uri` del servicio principal para proceder a la invocación y ejecución.

4.3. Capa de gestión de cooperación y ejecución de aplicaciones: MAGOS-WORKFLOW

MAGOS-WORKFLOW se encarga de orquestar un conjunto de aplicaciones instaladas en el *grid* para que lleven a cabo un conjunto de tareas que representan un proceso de negocio, tal como fue descrito en el numeral 3.4. MAGOS-WORKFLOW expone un servicio grid con operaciones que permiten orquestar y ejecutar un proceso. Este servicio es accedido por un cliente que provee una interacción simple, permitiendo (1) consultar y enlazar actividades definidas en el proceso, con aplicaciones existentes en el *grid* y (2) ejecutar el proceso. Para lograr que las actividades definidas en el proceso efectivamente ejecuten un servicio grid, se adicionó una funcionalidad al motor ActiveBPEL. Esta nueva funcionalidad es un `Dispatcher` invocado por la actividad, y que, usando reflexión, invoca al servicio grid correspondiente y devuelve la respuesta al motor para continuar con la ejecución. [2] Para facilitar la asignación de aplicaciones a las tareas del proceso `bpel`, MAGOS-WORKFLOW utiliza la documentación `javadoc` de la aplicación, generada en formato XML, para presentar de forma automática una descripción completa de los servicios disponibles. De esta forma, se ofrece al desarrollador del *workflow* más información que la encontrada habitualmente en el WSDL de descripción del servicio.

MAGOS utiliza el *framework* Kazai [27] para manejar el requerimiento no funcional de seguridad.

5. Puesta en marcha de una aplicación *datagrid* utilizando MAGOS

A continuación se muestran aspectos relevantes relacionados con la puesta en marcha de una aplicación sencilla sobre el *grid*, una vez esta ha sido desarrollada. Esta descripción de ejemplo no tiene en cuenta el caso en que se desee utilizar servicios ya instalados en el *grid*, que hagan parte de otras aplicaciones, que hayan sido instaladas por el mismo desarrollador o por terceros, escenario previsto en MAGOS.

Primero se muestra el proceso sobre GT4 y OGSA-DAI, y posteriormente se presenta el proceso de desarrollo utilizando MAGOS:

Creación de un servicio de datos (DataService)

GT4 - OGSA-DAI
Crear un script de creación de un esquema de datos para una tecnología específica (por ejemplo, bases de datos relacionales).
Buscar un servicio de datos de OGSA-DAI (`DataService`) que cumpla los requerimientos de tipo de fuente de datos, proveedor, versión, capacidad de almacenamiento. Para esto, puede hacer uso del *IndexService* de Globus que permite ver los servicios instalados incluyendo los servicios de datos. Si no existe un `DataService` adecuado, debe crear uno.
Crear un cliente para comunicarse con ese `DataService` para enviarle el script de creación.
Modificar sus servicios para que apunten a ese *DataService*. Entonces, el software de aplicación indica de manera explícita la URI de la máquina seleccionada
MAGOS
Generar un *script* de creación de un esquema de datos para una tecnología específica.
Crear un archivo XML que describe los

requerimientos no funcionales de la aplicación (ver figuras 1 y 2).

Incluir en el descriptor XML los requerimientos de tecnología datos a utilizar (v. gr., un SMBD en particular como MySQL 4) e indicar cuál es el *script* de creación.

MAGOS busca el recurso y ejecuta el *script*

Instalación de un servicio

GT4 - OGSA-DAI

Recorrer los nodos del *grid* para verificar requerimientos de hardware que necesite el servicio (procesador, memoria, etc).

En cada nodo adecuado, tomar el archivo `gar` correspondiente al servicio, copiarlo en el nodo y hacer *deploy* de su servicio (el desarrollador debe conocer los comandos necesarios para poder realizar estas tareas). Esto lo debe hacer por cada uno de los servicios que desee instalar.

MAGOS

Incluir en el archivo XML los recursos que espera del *grid*, sus requerimientos no funcionales de replicación y confidencialidad y la ubicación del archivo `gar`.

MAGOS instala el software como indica el desarrollador

Manejo de Replicación

GT4 - OGSA-DAI

Definir un esquema de replicación y el proceso para lograrlo, pues GT4 no incluye manejo de réplicas a nivel de servicios. Para poder instalar un servicio con réplicas, debe repetir los pasos de instalación descritos tantas veces como réplicas necesite.

Si requiere réplica a nivel de datos, debe repetir los pasos para creación de esquema de datos anteriormente descritos, tantas veces como réplicas de datos necesite y además definir también un esquema para este tipo de replicación, porque OGSA-DAI no incluye este requerimiento.

Debe tener en cuenta la disponibilidad de los servicios replicados en los nodos. Si un nodo se cae, la lógica del servicio debe buscar una réplica en otro nodo que se encuentre disponible, luego estos aspectos quedan totalmente acoplados en el código fuente de la aplicación

MAGOS

Indicar en el archivo XML cuantas réplicas de datos requiere y cuantas de servicios en el archivo de instalación.

MAGOS instala el software como indica el desarrollador

MAGOS se ocupa de la ubicación de réplicas disponibles y realiza tareas básicas de balanceo de carga

Manejo de autorización y seguridad

GT4 - OGSA-DAI

Definir el esquema de seguridad. Si los servicios son privados, autorizados o públicos, el desarrollador debe idearse alguna manera para no permitir que usuarios no autorizados accedan a ellos. Por ejemplo, debe desarrollar dichos mecanismos o debe integrar tecnología que lo implemente, con la consecuente curva de aprendizaje adicional.

MAGOS

Indicar en el archivo XML el esquema de seguridad para la aplicación y sus servicios.

MAGOS se encarga de autenticar y autorizar a los usuarios para el uso o consulta de servicios.

Manejo de servicios complejos y dependencias

GT4 - OGSA-DAI

Si alguno de los servicios requiere de otro, el desarrollador debe instalarlos todos y posteriormente debe especificar de manera explícita en el software de aplicación, las referencias que tengan unos a otros, para que el servicio que los invoca funcione correctamente. El software de aplicación queda acoplado a dichas referencias

MAGOS

El desarrollador no interviene en el proceso.

MAGOS instala y ubica en momento de ejecución los servicios dependientes, según la carga del *grid* en ese momento.

Consulta de servicios instalados

GT4 - OGSA-DAI

GT4 provee el *IndexService*, que muestra los servicios que se encuentran dentro del contenedor. El desarrollador debe conocer la manera de invocar al *IndexService* y la manera de extraer la información que necesita. El *IndexService* no permite ver las dependencias entre servicios

MAGOS

Ubica los servicios instalados y sus dependencias, teniendo en cuenta su nivel de seguridad, utilizando el catálogo MAGOS.

Manejo de la distribución de carga

GT4 - OGSA-DAI

Para la ejecución de sus servicios, el desarrollador

ha cableado ya sus servicios y referencias a un nodo específico del *grid*, sin importar si este nodo esta sobrecargado o no. Es decir, en la ejecución, no se tiene distribución de carga, a menos que el desarrollador la implemente dentro de la lógica del servicio apoyándose en alguna herramienta de *scheduling* que se lo permita. El desarrollador debe aprender e incorporar dicha herramienta.

MAGOS

El desarrollador no interviene en el proceso.

MAGOS integra herramientas de *scheduling* que permiten no sobrecargar un nodo sino distribuir lo mas uniforme posible la carga sobre el *grid*.

Manejo de workflow de aplicaciones

GT4 - OGSA-DAI

En el caso de requerir que un servicio coopere con otro para realizar una acción más grande o un proceso de negocio, debe realizar otro servicio que los incluya.

Debe instalar el servicio integrador siguiendo los pasos de instalación anteriormente descritos y debe conocer de antemano la localización de los servicios que va a invocar. De esta manera, el desarrollador tendría muchas dependencias entre muchos servicios, lo cual lleva a no poder diferenciar entre manejar dependencias entre servicios y orquestar servicios

MAGOS

Definir el proceso `bpel`.

Orquestar las actividades definidas en el proceso `bpel` con las aplicaciones existentes en el *grid*, de acuerdo con la información respectiva ofrecida por MAGOS.

Ejecutar el proceso.

Como puede observarse en la comparación, MAGOS propone una forma sencilla de describir requerimientos frente al *grid*, permitiendo al desarrollador concentrarse en cómo quiere instalar y usar su aplicación.

6. Resultados y trabajo futuro

MAGOS está construido sobre GT4 4.0.3, OGSA-DAI 2.2, Java 1.5. Usa ActiveBPEL 3.0, Condor 6.8.4, Kazai 1.1.0. MAGOS cuenta actualmente con todas las funcionalidades y características descritas.

El entorno de desarrollo actual es un *cluster* basado en Scientific Linux. La Universidad de los Andes está actualmente instalando una infraestructura de *grid* que permite integrar varios *cluster* existentes. El siguiente paso es instalar MAGOS en esta infraestructura, para iniciar el proceso de uso de MAGOS en la instalación de aplicaciones de alto desempeño por parte de desarrolladores no expertos en tecnología *grid*.

Se está desarrollando actualmente un modelo de instalación, ejecución y control de flujos de datos multimedia, que permiten incluir servicios de *streaming* e indexación como parte de los servicios básicos de MAGOS. También se está incluyendo en la arquitectura la conexión al *grid* de clientes móviles multimedia que utilizan tecnología SVG y SMIL. En cuanto a la gestión de datos como parte central de la cooperación entre entidades heterogéneas y autónomas en el *grid*, se está integrando en MAGOS servicios básicos que permiten la declaración y validación de esquemas XML y su utilización en fuentes de datos nativas XML sobre el *grid*. En un futuro esperamos incluir la integración herramientas que faciliten validación e intercambio de documentos complejos en el *datagrid*.

7. Conclusiones

MAGOS es una propuesta orientada a simplificar la tarea de utilizar un *datagrid*, tanto para quien desarrolla aplicaciones orientadas por servicios como para quien quiere que su aplicación coopere con otras ya instaladas allí. Para esto, MAGOS ofrece un mecanismo para describir, de manera declarativa, la forma como se quiere usar un *grid* para instalar y ejecutar una aplicación compuesta de servicios complejos. Permite establecer flujos de ejecución de negocio, que utilizan aplicaciones ya instaladas, bien sean propias o autorizadas, de forma que entidades autónomas, heterogéneas y dispersas pueden cooperar fácilmente. MAGOS integra *middleware* existente que facilita ofrecer sus servicios siguiendo plataformas estándar y desarrolla los componentes que permiten la instalación, ubicación, composición, orquestación, replicación, y ejecución de servicios *grid*, en un entorno que garantiza confidencialidad y autorización.

MAGOS ofrece al desarrollador SOA la posibilidad de describir los recursos que espera del *grid* para la ejecución correcta de su aplicación, indicando aspectos de replicación y autorización. Ofrece el concepto de aplicación como integrador de servicios complejos, propios o de terceros. Realiza el *deploy* de las aplicaciones de forma transparente para el desarrollador, a quien se ofrecen herramientas de navegación y *workflow* para orquestar y ejecutar las aplicaciones, teniendo en cuenta la carga real del *grid* en ese momento.

La estrategia de solución de MAGOS es coherente con muchos trabajos que se desarrollan actualmente en el área, utilizando la tecnología abierta y estándar. Su principal aporte está en el desacoplamiento de las aplicaciones con respecto a la ubicación y replicación

de servicios, lo cual permite la autonomía de las entidades que publican los servicios y la ejecución adecuada según la carga del grid en momento de ejecución. MAGOS permite la orquestación de aplicaciones heterogéneas, autónomas que cooperan en un *grid* de datos.

Referencias

[1] BILYKH, Iryna. et all. "¿Can GRID services provide answers to the challenges of national health information sharing?". Proc. of the 2003 Conference of the Centre for Advanced Studies on Collaborative Research. 2003.

[2] CAÑÓN VARGAS, Marcela. "Cooperación entre aplicaciones autónomas y heterogéneas Grid-enabled – Proyecto Magos". Tesis de Maestría – Univ. de los Andes. Director: Claudia Jiménez. A publicarse Agosto 2007.

[3] CARDONA MARÍN, Juan David. "Catálogo de servicios para aplicaciones Grid-enabled – Proyecto Magos". Tesis de Maestría – Univ. de los Andes. Director: Claudia Jiménez. A publicarse Agosto 2007

[4] CASTRO, Harold. "Grid Computing: promesa de los sistemas distribuidos". Revista Sistemas ACIS, edición 98, Oct-Dic 2006.

[5] CHEN, Yin. "Transaction-based Grid data replication using OGSA-DAI". Feb 2007. [En línea] Disponible: http://www.gridminer.org/dialogue/slides/chen.ppt. Consulta: Abr 2007.

[6] CIGLAN, Marek. HLUCHY, Ladislav. "Replica synchronization based on OGSA-DAI services". Feb 2007. [En línea] Disponible: http://www.gridminer.org/dialogue/slides/ciglan.ppt. Consulta: Abr 2007.

[7] CONDOR Team - Univ. of Wisconsin–Madison. Manual de Condor version 6.8.4. [En línea] Disponible: http://www.cs.wisc.edu/condor/manual/v6.8.4/condor-V6_8_4-Manual.pdf. Consulta: Feb 2007

[8] FOSTER, Ian. "What is the Grid? A Three Point Checklist". [En línea] Disponible: http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf. Consulta: Abr 2007

[9] FOSTER, Ian. KESSELMAN, Carl. "The Grid 2: Blueprint for a New Computing Infrastructure". The Elsevier Series in Grid Computing. Morgan Kaufmann. 2nd. Edition. 2003.

[10] FREUND, Joerg. et all. "Health-e-Child : An Integrated Biomedical Platform for Grid-Based Paediatrics". [En línea] Disponible: http://www.caip.rutgers.edu/~comanici/Papers/Health-e-Child_HG06.pdf. Consulta: Abr 2007.

[11] HASTINGS, Shannon y otros. "Introduce: Grid Service Authoring Toolkit". Feb 2007. [En línea] Disponible: http://www.gridminer.org/dialogue/slides/langella2.ppt. Third DIALOGUE Workshop: Towards the Next Generation of Data Grid Software. Consulta: Abr 2007.

[12] HEY, Tony. TREFETHEN, Anne E. "UK e-Science Programme: Next Generation Grid Applications". Int. J. of High Performance Computing Applications, Vol. 18, No. 3, 285-291 (2004) [En línea] Disponible: http://hpc.sagepub.com/cgi/content/abstract/18/3/285. Consulta: Abr 2007.

[13] JACOB, Bart. BROWN, Michael. FUKUI, Kentaro. TRIVEDI, Nihar. IBM. "Introduction to Grid Computing". IBM Redbooks. Dic 2005. [En línea] Disponible: http://www.redbooks.ibm.com/abstracts/sg246778.html

[14] JANCIAK, Ivan. "Workflow Enactment Engine and OGSA-DAI". Feb 2007. [En línea] Disponible: http://www.gridminer.org/dialogue/slides/janciak.ppt. Third DIALOGUE Workshop: Towards the Next Generation of Data Grid Software. Consulta: Abr 2007.

[15] JIMÉNEZ GUARÍN, Claudia Lucía. "Datagrid: Infraestructura de computación en malla para integración de aplicaciones e información". Revista Sistemas ACIS, edición 99, Ene-Mar 2007. pp. 58-67

[16] LÓPEZ BORJA, Vladimir. "Infraestructura para la instalación y ejecución de servicios Grid sin intervención del desarrollador – Proyecto Magos". Tesis de Maestría – Univ. de los Andes. Director: Claudia Jiménez. A publicarse Agosto 2007

[17] Proyecto ActiveEndpoints. [En línea] Disponible: http://www.active-endpoints.com. Consulta: Abr 2007.

[18] Proyecto BeInGrid. [En línea] Disponible: http://www.beingrid.com/. Consulta: Abr 2007.

[19] Proyecto BioInfoGrid. [En línea] Disponible: http://www.bioinfogrid.eu/. Consulta: Abr 2007.

[20] Proyecto Commodity Grid. [En línea] Disponible: http://wiki.cogkit.org/index.php/Main_Page. Consulta: Abr 2007.

[21] Proyecto DataGrid – CERN. [En línea] Disponible: http://eu-datagrid.web.cern.ch/eu-datagrid/ Consulta: Abr 2007.

[22] Proyecto EELA - E-Infrastructure shared between Europe and Latin America. [En línea] Disponible: http://www.eu-eela.org/ Consulta: Abr 2007.

[23] Proyecto EGEE - Enabling Grids for E-sciencE. [En línea] Disponible: http://www.eu-egee.org/ Consulta: Abr 2007.

[24] Proyecto EGEE. "An Introduction to DataGrid". [En línea] Disponible: http://web.datagrid.cnr.it/LearnMore/LearnMore4.jsp. Consulta: Abr 2007.

[25] Proyecto Globus. "Towards Open Grid Services Architecture". [En línea] Disponible: http://www.globus.org/ogsa/. Consulta: Abr 2007.

[26] Proyecto Globus. Globus Toolkit. [En línea] Disponible http://www.globus.org/toolkit/ Consulta: Jun 2007

[27] Proyecto Kazai. [En línea] Disponible: http://kasai.manentiasoftware.com/ Consulta: Abr 2007

[28] Proyecto Oasis. "SOA, Service Oriented Architecture". [En línea] Disponible: http://www.oasis-open.org/committees/tc_cat.php?cat=soa. Consulta: May 2007.

[29] Proyecto Oasis. "Web Services Business Process Execution Language 2.0". OASIS Standard. [En línea] Disponible: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html. Consulta: Jun 2007.

[30] Proyecto Teragrid. [En línea] Disponible: http://www.teragrid.org/. Consulta: Abr 2007.

[31] Univ. of Edinburgh. Proyecto OGSA-DAI. [En línea] Disponible: http://www.ogsadai.org/. Consulta: Abr 2007.