ARQUITECTURAS Y MODELOS DE PROGRAMACIÓN EN COMPUTACIÓN GRID

Grid Computing Architectures and Programming Models

RESUMEN
El crecimiento permanente en computación paralela y sus arquitecturas en hardware y software, han generado nuevos retos para los desarrolladores de software. La computación Grid cada vez toma más relevancia en los problemas que requieren soluciones de multiprocesamiento y grandes cargas de trabajo; en consecuencia los modelos de programación deben adaptarse a estos desarrollos para que aprovechen y soporten las características de paralelismo, distribución, transparencia, heterogeneidad, rendimiento y procesamiento.
En este artículo se analizan los principales modelos de programación para computación Grid, así como el uso de librerías y frameworks que pueden ser adaptados sobre estas arquitecturas.

PALABRAS CLAVES: Computación grid, modelos de programación paralela, OGSA, programación grid.

ABSTRACT
The permanent growth in parallel computing and their hardware and software architectures has generated new challenges for the software developers. The Grid Computing takes more relevance in problems that require solutions on multiprocessing schemes or great load balancing problems; consequently the programming models must adapt to these developments so that they take advantage of and they support the characteristics of parallelism, distribution, transparency, heterogeneity and performance..
This paper describes the main programming models and standards for Grid computing are analyzed, as well as some examples in the use of packages, libraries and frameworks are detailed that can be adapted in the computing described it.
.

KEYWORDS: *Grid Computing, Grid Programming, OGSA, Parallel programming models*

GUSTAVO A. ISAZA
Ingeniero de Sistemas y Computación, Ph.D. (Estudiante)
Esp. Software para Redes
Profesor Auxiliar
Universidad de Caldas
gustavo.isaza@ucaldas.edu.co

NESTOR DARIO DUQUE MENDEZ
Ph.D. (c)
M.Sc en Ingeniería de sistemas
Profesor Asociado
Universidad Nacional de Colombia
Sede Manizales
ndduqueme@unal.edu.co

1. INTRODUCCIÓN

El principal objetivo de la programación Grid es el estudio de los modelos de programación, herramientas y métodos que soporten el desarrollo eficiente y portable de aplicaciones y algoritmos con alto desempeño en ambientes paralelos. [1]
Si bien hay un crecimiento importante en las herramientas para escalar la computación grid, los retos para los programadores son cada vez más interesantes y aquí es donde se puede evidenciar una debilidad en los lenguajes y estándares de programación.
Es necesario por lo tanto identificar dónde están las debilidades de los modelos de programación, qué nuevas capacidades se requieren y cuál es su mejor ambiente de implementación, si directamente en los lenguajes, en las herramientas o en algunas funciones extendidas. Los modelos de programación grid, no solamente deben soportar los principios de paralelismo, rendimiento, heterogeneidad, seguridad, sino también deben evolucionar sus técnicas de compilación, depuración y monitoreo.
Es evidente que las aplicaciones de Grid se tornan en un ambiente de operación heterogéneo y dinámico, estos sistemas corren sobre diferentes tipos de recursos que pueden cambiar en tiempo de ejecución, además deben invocar la tolerancia a fallos, redundancia, seguridad, balanceo de cargas, entre otras acciones como reconfiguración el sistema.
La evolución de la computación Grid se evidencia en la madurez de los estándares, el estándar de facto más relevante es el *Globus Project*, en el cual se definen arquitecturas del GRID, niveles de accesos, servicios, requerimientos, entre otros.
En este artículo se presenta una introducción a los fundamentos de programación para ambientes grid, tratando desde los elementos incorporados a los lenguajes, los ambientes middleware, así como las herramientas y librerías que pueden utilizarse para estas tecnologías, en el numeral 2 se hace una introducción a la computación grid, sus arquitecturas y servicios, en el

Fecha de Recepción: 24 Agosto de 2007
Fecha de Aceptación: 30 Octubre de 2007

aparte 3 se presentan los principales modelos de programación y requerimientos para estos ambientes y posteriormente se hace una breve reflexión del tema aquí expuesto.

2. CONTEXTO DE LA COMPUTACIÓN GRID

La tecnología de Grid permite usar en forma coordinada diferentes recursos que no están controlados por un ente centralizado, es un ambiente distribuido, por lo general heterogéneo en arquitecturas, sistemas operativos, aplicaciones, redes.[2]

El Grid Computing Information Centre[1] define un sistema Grid como un "tipo de sistema paralelo y distribuido que permite compartir, seleccionar y reunir recursos autónomos geográficamente distribuidos en forma dinámica y en tiempo de ejecución, dependiendo de su disponibilidad, capacidad, desempeño, costo y calidad de servicio requerida por sus usuarios". El concepto de Grid se refiere a la coordinación de recursos para ser compartidos y resolver problemas en un ambiente dinámico, heterogéneo en diferentes organizaciones.

El conjunto de recursos individuales y organizacionales que se definen bajo un conjunto de reglas para compartir es llamado *Organización Virtual (VO)*. [2] Algunos autores, precursores de la computación grid, proponen la existencia de estas organizaciones como el pilar de este ambiente. Los retos de la computación Grid pueden resumirse en características como: Gestión de recursos heterogéneos e interoperabilidad, descubrimiento, selección, reserva, asignación, gestión y monitoreo de recursos, garantías de rendimiento, gestión de fallos, optimización de algoritmos distribuidos, control de acceso a datos remotos, seguridad y autenticación, optimización de modelos de programación, entre otros.

En general las arquitecturas Grid se definen por capas donde cada nivel define sus propios recursos, servicios y protocolos, permitiendo escalar y abstraer la implementación de los mismos.

El proyecto Globus es una iniciativa de I + D cuyo principal objetivo es permitir la aplicación de las tecnologías de Grid en la ciencia, la arquitectura de *Globus ToolKit* está basada en estándares abiertos y está estructurada en capas; en la capa superior se gestionan los servicios de grandes volúmenes de datos, aquí aparece el modelo de Grid Service que es realmente una adaptación de los Servicios Web a modelos de computación grid. [3][5]

[1] http://www.gridcomputing.com/

Existen dos componentes disponibles que permiten desarrollar aplicaciones basadas en Globus, el API y el Kit de desarrolladores. [6]
El API está implementado en C, sin embargo provee un desarrollo eficiente a través de un modelo conocido como CoG (Commodity Grid), el cual soporta tecnologías implementadas en Java, Phyton, Web services, CORBA, Java Server Pages, Perl y Matlab.

3. MODELOS DE PROGRAMACIÓN GRID

Hasta hace unos pocos años se había enfocado el problema de la programación grid sobre los modelos tradicionales de programación paralela, sin embargo, es relevante considerar que puede ser insuficiente e incluso subutilizado aplicar solo éstas técnicas para los retos de la computación grid.[3]

Los estándares de programación grid deben entonces permitir crear aplicaciones que cumplan con requerimientos tales como: usabilidad, soporte para ambientes heterogéneos, portabilidad, interoperabilidad, confidencialidad y seguridad, tolerancia a fallos, entre otros, un evento tan simple como el "acceso al estado" de un recurso puede influir significativamente sobre el modelo de programación, el simple hecho de gestionar recursos compartidos basados en modelos de memoria compartida o memoria distribuida genera un modelo diferente sobre como paralelizar un algoritmo.

El estado del arte actual ofrece diferentes paradigmas como modelos de paso de mensajes (MPICH), modelos de memoria compartida (OpenMP), modelos RPC y RMI (GridRPC), modelos peer-to-peer (JXTA), modelos basados en componentes (XCAT) y modelos orientados a servicios (GSFL). [4] En los siguientes apartados se describen algunas de estás técnicas y modelos.

3.1 Modelos Orientados a Memoria Compartida (OpenMP) y Memoria distribuida (MPI)

OpenMP es una librería que soporta programación paralela en ambientes de memoria compartida. [14] Estos recursos pueden ser usados en diferentes lenguajes tales como Fortran, C, C++. OpenMP permite la ejecución paralela de código, la definición de datos compartidos y la sincronización de procesos [13]. Una aplicación desarrollada con OpenMP inicializa su entorno de ejecución con un solo hilo activo que se conoce como maestro, este a su vez ejecuta una porción de código antes del primer fragmento paralelo, sobre esta API la construcción de un ambiente paralelo se logra mediante directivas de concurrencia.
MPI (Message Passing Interface) es un estándar de programación paralela que define un modelo de librerías de paso de mensajes en un ambiente de memoria distribuida. [16][17] El entorno más utilizado para computación grid es MPICH-G2 que es realmente una

implementación y variante de MPI creado para estos ambientes. Este estándar usa el servicio de Globus Toolkit y permite acoplar múltiples máquinas, potencialmente con diferentes arquitecturas para correr aplicaciones MPI; MPICH-G2 automáticamente convierte los datos en mensajes que son enviados entre máquinas heterogéneas y multiprotocolo. [18] En MPI el número de procesos se asignan antes de que se ejecute la aplicación, además, no se generan nuevos procesos mientras el programa está en ejecución.

La principal diferencia entre los modelos de programación de memoria compartida y los de paso de mensajes (memoria distribuida) es fundamentalmente el número de procesos e hilos activos [1], en el caso de paso de mensajes todos los procesos están activos durante la ejecución de la aplicación, en el modelo de memoria compartida, hay un solo hilo activo al inicio y fin de la aplicación; pero durante la ejecución de la misma, el número de hilos puede cambiar dinámicamente. [21]

3.2 Modelos Peer-To-Peer

Los modelos basados en sistemas P2P (Peer-to-Peer) pueden ser clasificados en dos grupos, los *sistemas para compartir archivos* (tipo Gnutella, Ares, LimeWire, Napster, entre otros) que permiten crear un espacio global de nombres, un modelo de cache de archivos y un servicio de directorios para compartir recursos en ambientes distribuidos sin que haya un control centralizado, y los sistemas para compartir ciclos de CPU en un ambiente subutilizado controlado por un sistema central el cual distribuye trabajo en componentes más pequeños de contribución (caso Seti-AtHome, Entropía, Parabon entre otros). [19]

SUN Microsystems ha lanzado el paquete JXTRA en un ambiente de dominio público, este conjunto de APIS es una herramienta para construir sistemas P2P basado en el modelo de *PeerGroups* que es realmente una colección distribuida de personas que desean colaborar de una u otra forma [19]. JXTA incluye protocolos y servicios para descubrir, registrarse y compartir recursos. Los protocolos P2P están basados en mensajes XML los cuales permiten conectividad desde computadores hasta PDAs y dispositivos móviles. Los diferentes nodos (Peers) cooperan para enrutar mensajes permitiendo una conectividad convergente sin tener que entender la complejidad de los protocolos y topologías de la red. [20] En el siguiente código se presenta un ejemplo de inicialización de la plataforma JXTA y descubrimiento de servicios.[2]

```
/**
 * Inicializar  un servicio JXTA
 */
public class StartJXTA {
  static PeerGroup netPeerGroup = null;
  static PeerGroupAdvertisement groupAdvertisement = null;
            private DiscoveryService discovery;
            private PipeService pipe;
            public StartJXTA() { }
            public static void main(String args[]) {
                        StartJXTA myapp = new StartJXTA();
                        System.out.println ("Iniciando Jxta ....");
                        myapp.startJxta();
                        System.out.println ("Hasta pronto....");
                        System.exit(0);
            }
private void startJxta() {
 try {
            // crear e Iniciar el NetPeerGroup
            netPeerGroup =
            PeerGroupFactory.newNetPeerGroup();
 } catch (PeerGroupException e) {
  // Si no puede instanciar el Grupo genera error de salida
 System.out.println("Error al crear Grupo Peer");
 e.printStackTrace();
 System.exit(1);
}
// Obtener el anuncio del grupo
            groupAdvertisement =
            netPeerGroup.getPeerGroupAdvertisement();
// Obtener el discovery y el servicio PIPE
.
.
```

3.3 Modelos basados en RPC

3.3.1 RPC Grid

GridRPC es un modelo y conjunto de API's para computación Grid basado en RPC, permite desarrollar sobre un ambiente paralelo y provee ejecución para ambientes de alto desempeño. [12] Como en muchos de los modelos basados en RPC, el API usa la interfaz basada en IDL (Interface Definition Language), diferentes modelos de programación de tareas paralelas en la grid y estilos asincrónicos. El modelo de seguridad de GridRPC depende del Grid Security de GSI.

3.3.2 Java RMI (Remote Method Invocation)

RMI (*Remote Method Invocation*) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas máquinas virtuales de Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas. Es un modelo basado en RPC pero restringido a sistemas con Máquina Virtual de Java (JVM). Java RMI provee un ambiente de programación de alto nivel para desarrollo de aplicaciones de computación grid, sin embargo los problemas de interoperabilidad que se presentan por las restricciones de usar JVM's ha hecho que el estándar no sea el más posicionado en la computación grid.

[2] Taylor, Ian J. From P2P to Web Services and Grid http://www.p2pgridbook.com/index.html

3.4 Modelos basados en Objetos, Componentes, FrameWorks

Las principales alternativas que se utilizan para la programación distribuida han sido hasta el momento:

- *Sockets:* Uno de los inconvenientes es que requiere una implementación costosa, en términos de usar APIS complejas y tediosas para el programador.
- *Remote Procedure Calls (**RPC**):* No soporta objetos explícitamente.
- *Microsoft Distributed Component Object Model (**DCOM**):* Menos maduro, menos portable y además propietario.
- *Java Remote Method Invocation (**RMI**):* Debe ser Java – To – Java.
- *Common Object Request Broker Architecture (**CORBA**):* Multiplataforma, multilenguaje, restricciones en los puertos de comunicación.[11]
- *Web Services:* Multiplataforma, multilenguaje, No tiene restricciones de puertos por usar servicios sobre protocolos http estándar.

Algunos de los lenguajes ofrecen un conjunto de librerías que facilitan el desarrollo de aplicaciones distribuidas, en está sección se nombrarán algunas de estas técnicas que pueden usarse sobre ambientes de computación Grid.

3.5 Modelos Orientados a Servicios

3.5.1 JINI

JINI es una arquitectura orientada a servicios que provee mecanismos para que múltiples equipos interconectados puedan compartir recursos sin procesos previos de planificación y configuración o puesta a punto de protocolos de redes, opera en un ambiente de comunidad donde cada cliente incorpora a los servicios interfaces y controladores para que se distribuya eficientemente carga de trabajo y tareas. JINI ofrece procedimientos de descubrimiento y registro para incorporarse al grid que se requiera, es una arquitectura distribuida, no existe un control centralizado.

3.5.2 OGSA

Los modelos orientados a servicios están escalando a través del estándar OGSA (Open Grid Services Architecture), este modelo presenta una arquitectura basada en tecnologías de Web Services, soportada en estándares como XML, WSDL, SOAP, UDDI.
El componente básico de la arquitectura es el estándar OGSI (Open Grid Service Infrastructure), que es realmente una infraestructura de software estándar fundada en Servicios Web y utilizada para proveer la máxima interoperabilidad a través de los componentes de la arquitectura OGSA.

Como se puede observar en el framework de la Figura 1 [7] cada componente de la arquitectura OGSA ofrece un conjunto de servicios e interfaces para interactuar con los demás elementos del marco permitiendo incorporar de manera transparente y eficiente nuevos protocolos y aplicaciones. Este framework utiliza una arquitectura basada en Web Services descritos por WSDL (Web Services Description Languague), representando elementos con XML e intercambiando mensajes con SOAP (Simple Object Access Protocol). [7]

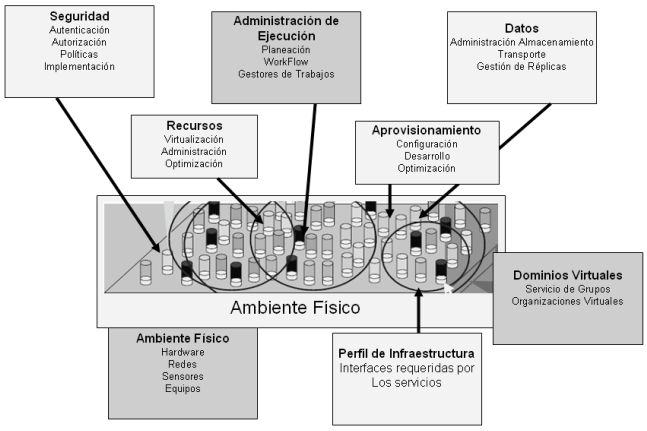

Figura 1. Framework OGSA

Un Web Service puede utilizarse para construir recursos identificados y localizados a través del protocolo UDDI, pueden ser descritos a través del lenguaje WSDL que describe un conjunto de servicios y puede comunicarse por mensajes a través del protocolo SOAP.
Existen otros mecanismos que permiten ejercer una gestión integral de los servicios Grid como: [8][9]

- *"Factory"* se utiliza para crear de forma dinámica instancias de los Grid Services, códigos de Grid Services ejecutables y requerimientos
- *"Registry"* es una interfaz que permite crear un conjunto de instancias para registrar los servicios en la Grid
- *"Handle"* gestiona los servicios después de estar registrados
- *"Discovery"* es la interfaz que facilita a los usuarios obtener información sobre los servicios ofrecidos
- *"Life Cycle"* maneja los estados de las instancias de los Grid Services
- *"Service Data"* Es el conjunto de información estructurada que se asocia las instancias
- *"Notification"* mecanismo que permite enviar información de estados y cambios (notificaciones).

Basándose en el estándar OGSI (Open Grid Service Infrastructure), una instancia de un servicio Grid es un Web Services que conforma un conjunto de convenciones expresadas por WSDL a través de sus interfaces, extensiones y comportamientos, estas especificaciones definen como los servicios grid son nombrados y referenciados, qué interfaces y comportamientos son comunes a todos los servicios grid,

cómo especificar interfaces adicionales, comportamientos y sus extensiones.

Un requerimiento importante en el momento en que se definen los servicios es la habilidad de describir los conceptos usando un modelo OGSI que combinado con WSDL, genera el estándar GWSDL (Grid Web Service Description Language). Una declaración de un *Servicio de Datos* es un mecanismo para expresar información disponible del estado de un servicio Grid a través de un esquema bien definido. Estos elementos están descritos a través de una estructura bien formada, un ejemplo de esta declaración usando el estándar GWSDL es el siguiente: [10]

```
<wsdl:definitions xmlns:tns="abc"
targetNamespace="mynamespace">
        <gwsdl:portType name="AbstractSearchEngine">
                <wsdl:operation name="BUSCAR" />
                 --------------------
                <sd:serviceData
name="URLCacheada" type="tns: cachedURLType"
                mutability="mutable" nilable="true",
maxOccurs="1"  minOccurs="0"
                modifiable="true"/>
        </gwsdl:portType>
</wsdl:definitions>
```

Existen otras formas de implementar estos recursos con ambientes tradicionales basados en C, C++ y C#, incluso .NET; estos entornos permiten alojar modelos para el OGSI. [15]

3.6 Otros modelos de programación Grid

3.6.1 Portals

Este modelo puede ser entendido como una técnica para proveer sistemas distribuidos sobre una interfaz basada en Web, en estas arquitecturas se evidencia la presencia de modelos por niveles o capas, la capa 1 corresponde a los clientes, la capa 2 (intermedia) los servidores y la capa 3 el repositorio de objetos, bases de datos y otros servicios. La comunicación entre estos niveles puede ser a través de http e incluso otros servicios como RMI, y la evolución hacia Web Services a través de XML y WSDL.

Un ejemplo es el Grid Portal Toolkit (GridPort), este ambiente permite el rápido desarrollo de altas funcionalidades de portales grid que simplifican el uso de servicios de bajo nivel. Provee un amplio conjunto de interfaces y servicios en el nivel de portal el cual provee acceso a un amplio rango de servicios grid e información, incluyendo el Globus Toolkit, el Grid Portal Information Repository (GPIR) entre otros. Este modelo hace parte del proyecto de *Portal Open Grid Computing Enviroments (OGCE)* [23]
GridPort provee servicios para los *"portales usuario"* tales como:

- Gestión de cuentas de usuario a través del modelo de tecnologías de autenticación grid
- Actualización dinámica de visualización, operación, carga y encolado de trabajos
- Comandos de ejecución de recursos grid
- Gestión de archivos y datos
- Procesar, clasificar y programar trabajos por lotes, entre otros.

3.6.2 Linda

Es un lenguaje de programación paralelo basado en C y Fortran con capacidades para crear programas que corren en diferentes plataformas, trabajo sobre un modelo de memoria virtual compartida conocida como *"Espacio de Tuplas"* el cual provee mecanismos de comunicación y sincronización de procesos independientes de la plataforma.[22] En estos modelos partes distintas de los datos pueden coexistir en diferentes procesadores, pero los elementos del proceso se visualizan como una única memoria global, es realmente una implementación del modelo *distributed virtual shared memory* (DVSM) [24]

4. CONCLUSIONES Y TRABAJOS FUTUROS

Las plataformas computacionales, como la computación Grid, serán cada vez más avanzadas, potentes, dinámicas y complejas, incrementando sustancialmente el número de componentes y recursos dispersos geográficamente.
Los modelos y herramientas de programación Grid requieren adaptarse a estándares abiertos y soportar necesariamente ambientes totalmente heterogéneos, interoperabilidad, confiabilidad y desempeño.

Es evidente la convergencia natural de los servicios de la computación Grid y la evolución de las arquitecturas basadas en Web Services. Esta sinergia es importante dada la madurez de estos estándares y la adaptación de los mismos a la computación paralela y distribuida sobre tecnologías Web.

Sigue siendo importante mantener y evolucionar los modelos basados en paso de mensajes (MPI) y los modelos basados en memoria compartida (OpenMP) por su gran potencialidad y estabilidad, y optimizar su incorporación expedita hacia las tecnologías de grid computing.

Por el momento nos encontramos aplicando técnicas de computación Grid para resolver problemas de optimización de firmas en Sistemas de Detección de Intrusos Distribuidos, sin embargo es importante que se utilicen y evidencien más iniciativas de investigación utilizando tecnologías paralelas y distribuidas.

5. BIBLIOGRAFÍA

[1] C. Lee, T. Domenico. *"Grid Programming Models: Current Tools, Issues and Directions"*. Computer Systems Research Department DEIS The Aerospace

Corporation, P.O. Box 92957 Universit ´a della Calabria. CA USA 87036 Rende, CS Italy. pp. 3-6. May 2002.

[2] I. Foster, C. Kesselman S. Tuecke. *"The Anatomy of the Grid. Enabling Scalable Virtual Organizations"*. pp. 2,3. October 2003

[3] I. Foster; Argonne & U.Chicago (Editor), Kishimoto, Fujitsu (Editor),A. Savva, Fujitsu (Editor), D. Berry, NeSC,A. Djaoui, CCLRC-RAL,A. Grimshaw, UVa B. Horn, IBM *"The Open Grid Services Architecture, Version 1.0"*. PP. 14-15. January 2005

[4] C. Lee, G. Lee, D. Talia. *"Grid Programming Models: Current Tools, Issues and Directions. Grid Computing –Making the Global Infrastructure a Reality"*. PP. 16-18 2003

[5] L. Ferreira, V. Berstis, J. Armstrong, otros. *"Introduction to Grid Computing with Globus. IBM Red Books"*. PP. 157-159. Septiembre 2003

[6] L. Ferreira,,V. Berstis,J. Armstrong, otros. *"Introduction to Grid Computing with Globus. IBM Red Books"*. PP. 142-143. Septiembre 2003

[7] I. Foster; Argonne & U.Chicago (Editor), Kishimoto, Fujitsu (Editor),A. Savva, Fujitsu (Editor), D. Berry, NeSC,A. Djaoui, CCLRC-RAL,A. Grimshaw, UVa B. Horn, IBM "*The Open Grid Services Architecture"*, Version 1.0. PP. 14-28. Enero 2005

[8] I. Foster, C. Kesselman, J.M. Nick, S. Tuecke. *"The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration*". PP. 13-15. 2003

[9] D. Gannon, R. Ananthakrishnan, S. Krishnan, M. Govindaraju, L. Ramakrishnan, A. Slominski. "*Grid Web Services and Application Factories"*. Department of Computer Science. Indiana University. PP. 6-8. 2003

[10] J. Joshy. "*A developer's overview of OGSI and OGSI-based grid computing Get an in-depth look at the Open Grid Service Infrastructure"*. Software Engineer, IBM. 2005

[11] H. Jacobser., et al. High performance corba working group.http://www.omg.org/realtime/workinggroups/high performance corba.html, 2001.

[12] V. Getov, G. von Laszewski, M. Philippsen, and I. Foster. *"Multi-Paradigm Communications in Java for Grid Computing"*. Communication of the ACM, pages 118–125, October 2001.

[13] OpenMP Consortium. OpenMP C and C++ Application. Program Interface, Version 1.0, 2007. http://www.compunity.org/index.php

[14] C. Lee; T. Domenico. *"Grid Programming Models: Current Tools, Issues and Directions. Computer Systems"*. Research Department DEIS The Aerospace Corporation, P.O. Box 92957 Universit ´a della Calabria. CA USA 87036 Rende, CS Italy.PP 5-8. 2002.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. *"Grid Services for Distributed System Integration"*. IEEE Computer,pages 37–46, June 2002.

[16] Message Passing Interface Forum. "*MPI: A Message Passing Interface Standard"*, June 1995. www.mpi-forum.org/.

[17] Message Passing Interface Forum. *MPI-2: "Extension to the Message Passing Interface"*, July 1997. www.mpiforum.org/

[18] MPICH-G2. http://www3.niu.edu/mpi/ Diciembre 2005

[19] G. Dennis, R. Bramley, G. Fox. *"Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications"*. Department of Computer Science, Indiana University. PP. 18-20. 2003

[20] O. F. Rana, V. S. Getov, E. Sharakan, S. Newhouse, and R. Allan. "*Building Grid Services with Jini and JXTA"*, February 2002. GGF2 Working Document.

[21] W. Vambenepe, *"Web Services Distributed Management: Management using Web Services"* Part 2, OASIS Committee Draft http://docs.oasis-open.org/wsdm/2004/12/cd-wsdm-muws-part2-1.0.pdf. , December 2004.

[22] R. Bjornson, e otros. "*Grid Computing & Linda Programming Model : an alternative to web-service interface"* / Rob Bjornson, Andrew Sherman Dr. Dobb's Journal, Boulder, .PP16-17,20,22,24. 2004

[23] M. Thomas, et al. *"The Grid Portal Toolkit"*. *http://gridport.net/main/*, 2006.

[24] N. Carriero. Carriero and D. Gelerntern. *"Application experience with Linda"*. In Proc. ACM/SIGPLAN Symposium on Parallel Programmig, 1988.