Planificación Dinámica sobre Entornos Grid

Tesis presentada para obtener grado de Doctor en Ciencias Informáticas

Autor: **Mg. Mario Leandro Bertogna**
Director: **Ing. De Giusti Armando**
Co-Director: **Dr. Naiouf Marcelo**

Facultad de Informática UNLP, Mayo 2010.

Agradecimientos

En primer término quiero agradecer a mi familia, mi señora Ximena y mis hijas Lara y Eva por haber tenido la paciencia y haberme dado el apoyo necesario para terminar esta tesis. Muchos fueron las horas y fines de semana que no estuve presente, los cuales espero compensar.

A Armando De Giusti y Marcelo Naiouf por las oportunidades de crecimiento y apoyo incondicional que me brindaron durante el desarrollo de la tesis. A Tito por la guía en cuanto a relaciones institucionales y trabajo profesional, y a Marcelo por sus sugerencias y correcciones técnicas en relación a los trabajos realizados.

A mis compañeros de la Universidad Nacional del Comahue: Rodolfo del Castillo por el espacio que me brindo para llevar a cabo la experimentación y sus consejos. A Eduardo Grosclaude y Francisco López Luro por las discusiones técnicas y el apoyo en la implementación de las soluciones, siempre dispuestos a brindar ayuda en forma generosa. Por último, pero no menos importante, a Laura Sánchez y Jorgelina Giorgetti no solamente por la contención durante el doctorado, sino desde 1990 cuando comencé la carrera de licenciatura en informática siempre fueron guía y apoyo en mis pasos profesionales más relevantes.

Tabla de Contenidos

Indice de Figuras

Indice de Tablas

Indice de Código

Resumen

El objetivo de esta Tesis es el análisis para la gestión de entornos virtuales de manera eficiente. Se realizó una optimización para el middleware de planificación en forma dinámica sobre entornos de computación Grid, siendo la meta a alcanzar la asignación y utilización óptima de recursos para la ejecución coordinada de tareas.

Se investigó en particular la interacción entre servicios Grid y la problemática de la distribución de tareas en meta-organizaciones con requerimientos de calidad de servicio no trivial, estableciendo una relación entre la distribución de tareas y las necesidades locales pertenecientes a organizaciones virtuales.

En primer término, se realizaron experiencias sobre la problemática de aplicaciones en entornos multicluster y luego se analizó como la infraestructura Grid pueden impactar en el diseño de este tipo de aplicaciones. Se propuso e implementó una arquitectura para la gestión de entornos virtuales, logrando un máximo aprovechamiento de entornos Grid utilizando clusters de computadoras como recurso. Por último se formuló una optimización para algoritmos de planificación, donde en base a la distribución geográfica y las características de la aplicación que se debe ejecutar, se generan automáticamente entornos virtuales de ejecución utilizando el conjunto de máquinas más adecuado. Con esta optimización se logra en el mejor de los casos un incremento del 20% en el tiempo total de ejecución del sistema.

Motivaciones y Objetivos

El objetivo general de esta Tesis es investigar la administración y realizar mejoras en el middleware de planificación en forma dinámica sobre entornos de computación Grid, siendo la meta a alcanzar la asignación y utilización óptima de recursos para la ejecución coordinada de tareas.

Los temas de investigación derivados se enmarcan en la administración de recursos en entornos Grid; esto involucra temas de procesamiento distribuido, comunicación y coordinación de recursos, buscando optimizar aspectos de adaptación en la planificación y ejecución de tareas en forma dinámica a su entorno.

En particular se busca investigar la interacción entre servicios Grid y la problemática de la distribución de tareas en meta-organizaciones con requerimientos de calidad de servicio no trivial, estableciendo una relación entre la distribución de tareas entre las necesidades locales y de las organizaciones virtuales que las contienen.

El ámbito de experimentación consta de equipamiento y recursos distribuidos geográficamente en la Universidad Nacional del Comahue y la Universidad Nacional de La Plata, así como también en otras universidades latinoamericanas y europeas.

Como dominio objetivo para la aplicación de las tareas de investigación interesa la clase de problemas relacionados con el tratamiento masivo de datos y procesamiento del mismo, aplicaciones paralelas del tipo SIMD y problemas con requerimientos de tiempo real o uso coordinado de recursos como el procesamiento de visualización científica remota.

Los resultados obtenidos constan de una arquitectura e implementación sobre un banco de pruebas multi-geográfico y una propuesta de optimización del proceso de asignación y adaptación al entorno dinámico de espacios virtuales. La propuesta fue probada de manera efectiva a través de un simulador de eventos, modelando un entorno multicluter.

1. Motivaciones

La tecnología Grid ha logrado insertarse en el mundo científico, debido al proceso de evolución en los últimos años. El viejo deseo de compartir recursos e información que llevó a la interconexión de equipos y aplicaciones, cobra un nuevo impulso con la posibilidad de cumplir este deseo también a nivel global, entre distintas organizaciones en forma cooperativa y coordinada. Grid posibilita la creación de sistemas que aprovechan esta capacidad dando lugar a un avance en el ámbito de los sistemas distribuidos. Si bien en algunos lugares esta tecnología ha comenzado a usarse, lejos está aún de lograr su madurez, restando soluciones a complejos problemas; uno de estos problemas es la administración eficiente de recursos.

En los entornos computacionales tradicionales, la administración de recursos es un problema resuelto. Los administradores de recursos como planificadores, coordinadores de flujos de trabajo y sistemas operativos se encuentran presentes en la mayoría de las plataformas, y están diseñados para ejecutarse con la premisa de que poseen el completo control de los recursos que disponen. De esta manera son capaces de implementar políticas y mecanismos para hacer un uso efectivo y eficiente de los mismos.

Esta premisa no se aplica a entornos Grid. Se deben desarrollar métodos para administrar y coordinar recursos a través de distintos dominios administrativos, teniendo en cuenta la heterogeneidad, la falta casi total del control y las inevitables diferencias en las políticas aplicables a cada uno de los recursos involucrados.

El proceso de planificación en entornos Grid consta en general de tres fases bien definidas:

- Primera fase, descubrimiento de los recursos. Se basa en determinar qué recursos están disponibles, si se dispone de acceso y cumple con los requerimientos mínimos para la ejecución de las tareas.

- Segunda fase, selección del sistema. Dado un grupo de posibles recursos se debe obtener información y realizar toma de decisiones sobre los datos obtenidos.

- Tercera fase, ejecución de las tareas. Involucra el envío de las tareas y la reserva de recursos, preparación de las tareas, ejecución, monitoreo y proceso de finalización.

Cada uno de estos puntos plantea desafíos y de los trabajos que se están llevando a cabo pocos encaran este tipo de soluciones desde un punto de vista global, en búsqueda de una mejora integral en la coordinación de las tareas que involucra el proceso.

A medida que la tecnología Grid maduró y los entornos en producción incrementaron su tamaño y aumentó la complejidad de la interacción entre los distintos recursos, los tiempos de respuesta necesarios para satisfacer en forma óptima los requerimientos de ejecución deberán disminuir considerablemente. Por esto, siempre es de gran interés poder optimizar el proceso de administración disminuyendo tiempos y automatizando la toma de decisiones en cada una de las fases mencionadas.

2. Experiencias realizadas

Con el fin de poder adquirir los conocimientos necesarios y realizar distintas experiencias ejercitando los conceptos a describir en esta tesis se realizaron trabajos en conjunto y pasantías en las siguientes instituciones y proyectos:

- Universidad Nacional de la Plata: trabajos en conjunto para la interconexión de clusters y gestión de recursos.
- Universidad Autónoma de Barcelona: pasantía donde se expuso el tema del presente trabajo y se realizaron contribuciones.
- Centro de Investigaciones Básicas y Aplicadas de Bahía Blanca: experiencias de interconexión con dispositivos especiales y clúster a través de portales Grid.
- Universidad Nacional del Sur: experiencias de interconexión de recursos.
- Universidad Nacional del Cuyo y Universidad Nacional de Buenos Aires: pruebas de interconexión de recursos y asignación de tareas en cluster no dedicados.
- Proyecto Cyted Grid: intercambio de conocimientos y sincronización de infraestructura.

3. Contenido

El desarrollo del presente trabajo abarcó varios años y fue dividido en etapas con el fin de lograr el objetivo propuesto. A continuación se describen cada una de estas etapas que pueden verse reflejadas a lo largo de los distintos capítulos.

Capitulo 1

En el primer capítulo se desarrollan conceptos básicos de Grid y sus usos. Se analiza en profundidad su arquitectura, virtualización de recursos y servicios. Más alineado al objetivo del trabajo se detalla la problemática de la gestión de recursos, alternativas, soluciones y se realiza una descripción de los entornos virtuales siendo este el ámbito de desarrollo de aporte doctoral.

Capitulo 2

En el segundo capítulo, con una perspectiva técnica, se desarrolla la evolución de las aplicaciones paralelas de entornos cluster a multicluster. Se describe la ejecución remota en Grid. Se detallan las experiencias realizadas con universidades en el exterior, centros de investigación nacionales y empresas privadas, realizando migraciones de aplicaciones paralelas a Grid. Por último se realiza la caracterización de su comportamiento extendiendo la metodología de asignación eficiente de recursos multicluster a Grid.
Aporte: Se logro extender el modelo de caracterización de aplicaciones multiclusters a entornos Grid bajo un paradigma de programación orientada a servicios con mínimo impacto en las aplicaciones.

Capitulo 3

Una vez caracterizado el comportamiento de las aplicaciones paralelas dentro del dominio de estudio, en el capitulo tres se extiende el concepto de multicluters a espacios virtuales. Se propone una arquitectura y se desarrolla un marco de trabajo (*framework*) para realizar el envío de trabajos, analizarlos para la toma de decisión con respecto a la localización de corrida y ejecución de políticas de ejecución.
Aporte: Se diseño e implementó una arquitectura para la creación de clusters a demanda basados, creando espacios virtuales de trabajo con máximo rendimiento de los nodos

asignados.

Capitulo 4
En este capítulo se describe la creación e implementación de redes de recursos virtuales dentro de una organización virtual en entornos Grid utilizando la arquitectura definida para el uso de organizaciones virtuales. Los usuarios pueden acceder a los recursos en forma interactiva a través de interfaces web. Se buscó realizar una configuración en forma segura y con mínima intervención de los administradores locales en cada organización física. Se aplica la solución a varios casos de uso.
Aporte: Se implementa un banco de pruebas y se analizan las experiencias realizadas sobre casos de usos reales sobre espacios virtuales usando la arquitectura propuesta.

Capitulo 5
En el quinto capítulo se realiza un análisis y clasificación de algoritmos de planificación en Grid. Para los algoritmos seleccionados, se desarrolla un entorno de simulado, donde se realizan pruebas de concepto. En estas pruebas se estudian alternativas al problema de la planificación y escalabilidad. También se realizan optimizaciones para análisis en tiempo en ejecución y estudios de impacto de migración de tareas.
Aporte: Se propone y analiza factibilidad de la incorporación de algoritmos de selección de clusters para el espacio virtual. Se diseña e implementa un simulador basado en eventos para la prueba de escalabilidad de la solución. Se incorpora la adaptación del espacio virtual logrando optimizar la asignación de nodos de cómputo en un porcentaje significativo.

Capitulo 6
En el sexto capítulo se analiza el impacto de los aportes realizados en los diferentes capítulos sobre nuevas tecnologías, como la migración de sistemas Grid desde entornos científico/académicos a sistemas Cloud en entornos comerciales. Se plantean diferencias y evaluación de evolución en el concepto de meta-planificación.
Aporte: Se analiza las propuestas realizadas en los diferentes capítulos con respecto a la evolución de las tecnologías Grid.

Capitulo 7
Por último se realiza la un raconto de los conocimientos adquiridos durante todo el desarrollo del doctorado culminando con la conclusión del trabajo. Se detallan los aportes del trabajo a través de los proyectos y publicaciones nacionales e internacionales realizadas en los últimos años.

Apéndice A
Resumen de corridas del simulador

Apéndice B y C
Traza y Código del simulador

Capítulo 1 Introducción

La computación Grid es relativamente reciente, pero a pesar de contar con pocos años de desarrollo ha sufrido y sigue sufriendo diversas transformaciones, una evolución que va desde su definición hasta cambios en su implementación. En la primera parte de este capítulo se establece el dominio de la tecnología definiendo usos y componentes de la arquitectura, planteando la complejidad que involucra la coordinación propia de los sistemas distribuidos. Luego, alineado con el foco de esta tesis, se analiza en detalle la planificación de recursos en los sistemas Grid relacionado con la generación de espacios virtuales para la ejecución de aplicaciones. Finalmente se presenta el objetivo y aporte del trabajo doctoral analizando el contenido de la tesis distribuidos en los diferentes capítulos.

1. Computación Grid

El termino Grid [1,2] surge en los años noventa en el mundo académico. Originalmente se propuso para definir un sistema distribuido que proveía servicios de computo a demanda, de forma similar las redes de electricidad o agua. En estos sistemas la fuente de recursos es desconocida, solo se tiene acceso a una interfaz, como los tomas corriente en las casas para acceder a la red. Para los sistemas de computo la propuesta era que se podría acceder a recursos (como por ejemplo de almacenamiento) sin importar físicamente donde estuviesen, y la interfaz sería un cliente con un navegador de internet. Los servicios de cómputo permitirían colaborar compartiendo no solo conjuntos de datos sino también recursos de propósito especializados como telescopios o microscopios electrónicos. Y finalmente las aplicaciones demandantes de estos recursos y servicios aprovecharían el incremento en la capacidad de cómputo para análisis y procesamiento de datos, así como almacenamiento distribuido para visualización colaborativa, etc. Para coordinar el acceso de las aplicaciones a los recursos compartidos se crearon organizaciones multi-institucionales también llamadas organizaciones virtuales (OV). Tomando estas características se definió un sistema Grid como *la capacidad de compartir*

recursos coordinados y solución de problemas de organizaciones virtuales, dinámicas y multi-institucionales.

El compartir que a Grid concierne no es solo el de archivos, como se conoce en la WWW o en los sistemas *peer-to-peer (P2P)*, sino el acceso directo a computadoras, sistemas de almacenamiento y aplicaciones. Este compartir es necesariamente controlado con los proveedores de recursos y consumidores, estableciendo qué se comparte, quién puede acceder y cuáles son las condiciones bajo las que esto ocurre. Luego, las organizaciones virtuales son el conjunto de individuos o instituciones definidos por estas reglas.

Las OVs pueden variar en alcance, tamaño, duración, estructura, distribución y capacidades de compartir recursos. Algunos ejemplos de OVs pueden ser proveedores de servicios de aplicaciones, proveedores de servicios de almacenamiento, proveedores de ciclos de cómputo o una unión transitoria de consultoras al servicio de un proyecto particular. Estos ejemplos muestran la diversidad de aplicaciones que surgen a través de compartir recursos a nivel multi-institucional.

La rápida aceptación es esta tecnología se debe básicamente a la emergente necesidad de principios arquitecturales claros, estándares de software de facto y a la aparición de complejas aplicaciones demandantes de recursos computacionales [18]. Actualmente el concepto más aceptado describe a los sistemas Grid como:

- *Coordinación de recursos que no están sujetos a un control centralizado...*

Un Grid integra y coordina recursos y usuarios que se desenvuelven en distintos dominios administrativos, dentro de la misma empresa o en distintas organizaciones. Con distinto rango de direcciones o políticas, etc.

- *Utilización de estándares abiertos, interfaces y protocolos de propósito general ...*

Un Grid está construido con protocolos multipropósito e interfaces que abarcan desde autenticación, autorización, etc. Es importante que estas interfaces sean abiertas y estandarizadas para que los sistemas puedan integrarse fácilmente.

- *Para entregar calidad de servicio no trivial ...*

Un Grid permite que sus recursos sean coordinados para ser usados entregando diferente calidad de servicio como tiempo de respuesta, disponibilidad, seguridad y asignación de múltiples recursos que combinados tienen mayor capacidad que la suma de las partes.

Los casos de uso en esta tecnología plantean nuevos requerimientos, y por ello se hace necesario el establecimiento y la mantención de mecanismos para compartir recursos altamente flexibles, capaces de reflejar las estructuras colaborativas como cliente-servidor y P2P. Por ejemplo mecanismos de seguridad, como complejos requerimientos de control del uso, acceso a los recursos compartidos, delegación o la negociación de aplicaciones locales y políticas globales. Se necesitan mecanismos básicos para el descubrimiento, aprovisionamiento y gestión de recursos que van desde aplicaciones, archivos y datos a computadoras, sensores y redes, para permitir colaboraciones en tiempo real o de altas prestaciones. Existen distintos modelos de uso como mono o multiusuario, basados en costo o prestaciones y aspectos a solucionar como calidad de servicio, planificación y monitoreo de recursos.

2. Tipos de Grid

De manera ideal, un Grid debe proveer máxima integración de cualquier tipo de recursos de cómputo heterogéneo: unidades de procesamiento, almacenamiento, comunicación, etc. Pero como la tecnología no ha alcanzado ese punto de madurez, las redes Grid generadas hasta el momento son especializadas y generalmente con foco en la integración de determinados tipos de recursos. Actualmente se pueden observar distintos tipos de estas redes como las mencionadas a continuación [3]:

- **Grid computacional:** Grid computacional es una red que tiene poder de procesamiento como principal recurso de cómputo compartido por los demás nodos. Esta es la red Grid más común y es utilizada para realizar procesamiento de altas prestaciones con tareas de alta necesidad de procesamiento.
- **Grid de datos:** Como el Grid computacional el principal recurso a compartir entre sus nodos es la capacidad de almacenamiento. Este tipo de red puede ser considerado como un sistema de almacenamiento masivo construido sobre un gran número de dispositivos de almacenamiento.
- **Grid de redes:** Estas son conocidas como redes Grid o Grid de distribución. Este tipo de Grid tiene como propósito principal proveer tolerancia a fallas o altas prestaciones en los servicios de comunicación. En este sentido, cada nodo Grid actúa como un enrutador de datos entre dos puntos proveyendo almacenamiento de datos y facilidades para acelerar las comunicaciones entre estos puntos. En este punto la WWW puede considerarse una Grid de redes embrionaria que no satisface aún el tercer requerimiento de calidad de servicio de la definición de Grid.

Es sencillo ver cuán útil puede ser para soluciones de altas prestaciones, pero no está limitado a este aspecto. El gran diferenciador es la posibilidad de uso de equipamiento que de otra manera quedaría ocioso o la solución de problemas altamente complejos. Se puede observar que es posible agregar cómputo, almacenamiento y aumentar la capacidad de comunicación como por ejemplo conferencias virtuales. Teniendo esto en cuenta se pueden listar las principales razones del uso de esta tecnología:

- Mejorar la eficiencia y reducir el costo
- Utilizar recursos sub-utilizados
- Permitir la colaboración
- Virtualizar recursos y organizaciones
- Incrementar la capacidad y productividad
- Generar capacidad de procesamiento paralelo
- Soportar sistemas heterogéneos
- Proveer confiabilidad y disponibilidad
- Realizar balance de recursos
- Reducir el tiempo para lograr resultados

Cuando se exponen estas razones se puede demostrar la rápida aceptación de Grid en la academia e institutos de investigación o inclusive en entornos comerciales. Este tipo de tecnología tiene el potencial de cambiar cómo trabaja la gente:

- Una oportunidad para una nueva forma de aprendizaje y colaboración

- Una forma sencilla de presentar y analizar resultados
- Una forma sencilla de conducir discusiones entre pares
- Un solo punto de acceso a recursos distribuidos

3. Aplicaciones Grid

La investigación y la educación superior son los campos donde más se ha percibido el uso de la tecnología Grid. La Tabla 1 muestra algunos ejemplos de áreas de investigación que hacen uso de infraestructura de altas prestaciones y qué tipo de Grid puede tener más desarrollo.

Área de Investigación	Computacional	Datos	Redes
Física de Altas Energías			
Estudios ambientales			
Biología y Genética			
Química			
Materiales			
Astrofísica			
Ciencias Aeroespaciales			
Diseño			
Análisis Económicos			
Medicina			
Acceso remoto a aparatos experimentales			

Tabla 1. Tipos de Grid por Área de Investigación

Realizar pronósticos meteorológicos, cálculo de comportamiento aerodinámico de un avión, el ensamblado del genoma de organismos, el análisis de partículas elementales en un acelerador, virtualizar recursos o analizar patrones en terabytes de datos son algunas de las tareas que hacen uso de cálculo intensivo y gestionan grandes volúmenes de información. Por esto, forman parte del escenario donde la aplicación de la tecnología Grid se hace indispensable.

3.1. Mantenimiento predictivo

El diagnostico de fallas (DF) [4] se encuentra presente en muchos dominios, como por ejemplo medicina, ingeniería, transporte, industria aeroespacial. Sin importar el dominio, los sistemas de diagnóstico de fallas comparten un número de requisitos operativos y diseño. Son dependientes de los datos. El monitoreo y el análisis de sensores y su relación con conocimiento de dominio específico son críticos para la capacidad de predicción. Generalmente requieren complejas interacciones entre muchos agentes y expertos generalmente distribuidos. Deben proveer fuertes aspectos de seguridad o son sensibles al negocio teniendo alta dependencia de los requerimientos. En algunos contextos, la determinación de umbrales e intervalos permite la detección de fallos activa y predicción que pueden derivar en acciones preventivas cuando el impacto del mantenimiento se minimiza. En estos casos se dice que el sistema produce “mantenimiento predictivo”.

La adopción de tecnológicas avanzadas de monitoreo y control en el domino de equipamiento aeronáutico ha tenido beneficios significativos. El diagnostico y prognosis de condiciones de motores fue mejorada con el sistema DAME (Distributed Aircraft Maintenance Envornoment). El proyecto provee un banco de pruebas colaborativo e interactivo que soporta acceso remoto para el análisis de vibración y rendimiento de motores para usuarios distribuidos como: ingenieros de mantenimiento, analistas de mantenimiento y expertos del dominio.

Los servicios Grid involucrados en la arquitectura son:

- Servicios de Datos: Este servicio controla la interacción entre los sistemas en los motores y la estación en tierra que establecen los enlaces con los repositorios Grid. Debido a que los aviones aterrizan en distintas partes del mundo, existen distintas replicas del servicio.

- Servicio de Almacenamiento de la Información: El sistema consiste en un sistema de búsqueda de patrones, que usa métodos especiales de búsqueda en los datos provistos por los aviones y los datos históricos. El sistema usa técnicas similares a las de minería de datos. Se ha desarrollado una versión Grid que soporta búsqueda distribuida y obtención de datos masiva, usando método derivados de redes neuronales.

- Servicio de Modelado de motores: Este sistema permite tomar parámetros de un vuelo en particular y correr modelos de ingeniería. El propósito es poder detectar el estado actual de los motores.

- Soporte de razonamiento basado en casos: La herramienta de razonamiento basado en casos mejora el conocimiento y captura posible fallas de manera procedural. Otro uso es la gestión del manejo del flujo de tareas predicativas. Esto flujo de trabajo es coordinado por el sistema Grid estableciendo la secuencia de pasos necesarios en los distintos nodos.

- Servicio de la interfaz de mantenimiento: Este servicio coordina la interacción entre todos los expertos responsables del análisis de los datos y toma de decisión. Este servicio se ejecuta en forma autónoma capturando, chequeando y notificando a los distintos usuarios que los datos han sido recibidos. Grid incorpora los conocimientos de organización virtual bajo las cuales las operaciones son realizadas, invocación de servicios y operaciones de cómputo intensivas, uso de almacenamiento masivo, simulaciones y gestión de flujo de trabajo.

3.2. Access Grid

Access Grid [5] es una clase de aplicaciones Grid que intenta dar soporte a comunicación en tiempo real combinando aplicaciones, acceso a almacenamiento. El proyecto Access Grid creció con la red de investigación de los Laboratorios Argonne en Chicago, donde desarrolló entornos de colaboración basados en dos supuestos: el primero, que el ancho de banda entre los laboratorios con el tiempo se iba a incrementar dejando de ser un limitante para la integración y el segundo que las colaboraciones científicas involucrarían a grupos de personas en cada institución, no solo individuos.

Actualmente la aplicación involucra grandes pantallas multimedia, entornos de software interactivo, interfaces a *middleware* Grid y entornos de visualización remota. La mayoría de las universidades e institutos de investigación poseen recursos de cómputo avanzado, como por ejemplo equipos de altas prestaciones, almacenamiento masivo de datos y servicios de visualización, pero estos recursos no siempre están disponibles para la gente que los necesita.

La arquitectura involucra cuatro tipos de servicios Grid:

- Seguridad: provee un solo punto de autenticación para el dominio, manejo de claves, certificado de autenticación y canal encriptado.
- Servicios de persistencia: permiten al usuario crear proyectos de espacio persistente donde los objetos críticos como datos, flujos multimedia, aplicaciones o notas pueden ser almacenados, visualizados o manipulados.
- La *metáfora espacial (Conjunto de habitaciones virtuales)*, se expande con un conjunto de interconexiones y sofisticados mecanismos de descubrimiento y navegación basada en registración y combinación de servicios.
- Reproducción: sustenta la ejecución de aplicaciones existentes en un nodo compartido.

El objetivo es poder a través de un portal de internet con interfaces simples, enviar y compartir tareas o documentación. El material de estudio en proceso o elaborado puede ser analizado en línea varios equipos distribuidos por el mundo colaborando en tiempo real.

3.3. Visualización Científica

Los investigadores requieren grandes cálculos en distintos dominios desde simulaciones de clima a nivel global, experimentos físicos, medicina, simulaciones moleculares como fases de transformación sólidos-liquido-gaseoso, etc. Todos estos casos de uso entran dentro de la categoría de cómputo intensivo [6]. Debido a que estos fenómenos son experimentalmente no reproducibles, la mayoría de estos proyectos de investigación con sistemas complejos por primera vez son posibles gracias a la tecnología de visualización y simulación numérica usando tecnología de altas prestaciones.

La tecnología de altas prestaciones incluye grupos de máquinas con redes de baja latencia, supercomputadores, y de cómputo dedicados que proveen buena capacidad de computo pero con alto costo. Aparte de los costos de compra, estos dispositivos generalmente tienen alto costo de mantenimiento y tienden a volverse obsoletos en un tiempo relativamente corto, debiendo ser reemplazados por otros a medida que la investigación avanza.

Para satisfacer los requerimientos que este tipo de aplicaciones plantea se debe proveer alta capacidad de cómputo (en el orden de teraflops) a bajo costo. Lograr un costo accesible permite que los equipos puedan ser renovados sin causar impacto negativo en los experimentos. La computación Grid provee en este sentido la capacidad de integrar recursos dentro de la estructura de cómputo virtual, que puede estar constituida de múltiples grupos de máquinas de bajo costo. Estas máquinas están coordinadas por planificadores locales y de la organización virtual llamados *meta planificadores*.

Al convertirse en un entorno distribuido la seguridad es un tema primordial: la privacidad de los datos y la autenticación debe garantizarse a toda transacción realizada en el sistema. La gestión desde el punto de vista de mantenimiento debe ser moderada para no tener influencia en el costo del sistema. Desde el punto de vista del investigador el sistema debería poder convertirse en una aplicación de uso cotidiano ocultando la complejidad de la ejecución de la aplicación utilizando interfaces comunes. Los sistemas Grid intentan solucionar esta tarea proveyendo puntos únicos de acceso y credenciales de autenticación a través del uso de portales. Estos portales simplifican las cuestiones de interfaz permitiendo el acceso desde simples navegadores de Internet utilizando. Estos portales invocan capas intermedias para poder comunicarse con los gestores de recursos y poder ejecutar en forma coordinada las tareas seleccionadas.

4. Arquitectura Grid

Como factor común de cada una de las aplicaciones que se ejecutan sobre la tecnología Grid se dispone de una arquitectura estándar. En la definición de esta arquitectura, surgió la necesidad de elegir una capa de software intermedia en donde basar la misma [7]. Por ejemplo cuando en la arquitectura se define la interfaz con un método específico, debe haber una forma estandarizada de poder invocar el método para que la arquitectura sea utilizada como un estándar. Esta base, en teoría, puede ser cualquier capa intermedia de software de sistemas distribuidos como CORBA, RMI o RPC. Por estas razones se eligió una arquitectura de servicios *web* como la tecnología de preferencia para implementar esta capa de software. Si bien esta elección cubre los requerimientos de estandarización, los servicios *web* no mantienen el estado entre invocaciones, una desventaja para este tipo de tecnología. Existen soluciones para este problema pero no se cuenta con una manera estándar de hacerlo.

Todos estos servicios interactúan entre ellos constantemente. Con la interacción entre los servicios y pudiendo ser estos provistos por distintos prestadores, la única forma en que pueden coexistir es con el uso de estándares. Por ejemplo, el servicio de gestión de tareas puede consultar al servicio de descubrimiento de recursos para encontrar los dispositivos con los que coinciden los requerimientos de la tarea. Estos deben ser capaces de conectarse y establecer comunicación con el uso de protocolos conocidos por las dos entidades.

Un entorno Grid generalmente consiste en diferentes componentes. Algunos de ellos son:

- Arquitectura orientada a servicio: el sistema está diseñado para gestionar aplicaciones basadas en servicios a través de protocolos estándar. El software incluye protocolos de servicios y librerías. Los desarrolladores pueden usar estos servicios y librerías para implementar sistemas rápidamente.
- Infraestructura de servicios: los sistemas Grid incluyen servicios predefinidos para acceso, monitoreo, gestión y control de acceso a los elementos de la infraestructura computacional y datos.
- Servicios Web. Se hace uso de protocolos de servicios estándar y mecanismos de descripción, descubrimiento, acceso, autenticación y autorización. Estos componentes de software se encuentran en contenedores que almacenan los servicios y generalmente están desarrollados en Java, C o Python.
- Seguridad: Un subsistema de seguridad basado también en estándares es el encargado de la protección, autenticación, delegación y autorización.

- Estándares: En la mayoría de los casos las implementaciones son estándares o especificaciones adoptadas por distintas organizaciones para facilitar la construcción, operación, reusabilidad de componentes y el uso de herramientas estándares.
- Herramientas relacionadas: las herramientas seleccionadas en la mayoría de los casos intentan satisfacer las necesidades de los usuarios finales. Como se ha descripto en la aplicaciones estos generalmente no son más que navegadores de internet con funcionalidades especiales para la visualización de la información.

Un aspecto fundamental en las aplicaciones Grid es la estandarización. Esta define una interfaz común para cada tipo de servicio. Por ejemplo, si se observa Internet, una de las razones por las cuales es tan usada es porque la mayoría de las aplicaciones usan los estándares HTML, http, etc. El Open Grid Services Architecture (OGSA), desarrollado por The Global Grid Forum [20], intenta definir una arquitectura común, estandarizada y abierta para aplicaciones Grid. El objetivo de OGSA es estandarizar la gran mayoría de los servicios que comúnmente se encuentran en una aplicación Grid, especificando un conjunto de interfaces estándar para estos servicios: para esto, se deben identificar los servicios más utilizados en este tipo de aplicaciones.

4.1. Mecanismos estándar e interfaces

En los ejemplos y los componentes de la arquitectura mencionada anteriormente, la interacción se realiza a través del intercambio de mensajes por la red con el fin de realizar alguna tarea. Estos sistemas distribuidos utilizan protocolos en la estandarización como por ejemplo la descripción de interfaces de servicio, intercambio de mensajes, y validación de permisos. Esta estandarización facilita la construcción y entendimiento de los componentes individuales, la interoperabilidad entre las diferentes implantaciones de la misma interfaz, el compartir de los componentes y el desarrollo de herramientas reusables para ayudar al desarrollo de aplicaciones.

Los servicios web proveen esta flexibilidad, extensión y amplia adopción de mecanismos de XML para describir, descubrir e invocar servicios en la red; además estos protocolos son los recomendados para interacción con bajo acoplamiento y son robustos para un sistema distribuido.

Web Services Resource Framework (WSRF) es una especificación desarrollada por OASIS [19]. WSRF ha sido construido para expresar la relación entre recursos que mantienen el estado y los servicios web. Es un conjunto de especificaciones que definen la traducción de un servicio web de recurso en términos de la especificación de intercambios de mensaje y definiciones XML. Esta definición permite al programador declarar e implementar asociaciones entre servicios web y uno o más recursos con estado, y se describe a través de una asociación entre la descripción de un servicio web y la vista del estado del recurso.

La relación entre OGSA y WSRF se encuentra en que OGSA es la arquitectura y WSRF es la especificación sobre la cual se construye la infraestructura. WSRF provee la definición de un servicio con estado como la arquitectura (OGSA) lo necesita [8]. Como se ve en la Figura 1 WSRF específica el servicio con estado.

4.2. Infraestructura

Mientras que la aplicación se encuentra a un nivel de abstracción superior (como por ejemplo la ejecución de una aplicación paralela para la búsqueda de genes), a un nivel inferior se necesita la gestión de la infraestructura, entendida como computadoras, sistemas de almacenamiento e instrumental. La arquitectura orientada a servicios utilizada para las aplicaciones también puede ser usada para manejar los elementos de esta infraestructura, siempre que las interfaces se definan de manera apropiada.

Las infraestructuras Grid implementan servicios web para la gestión de infraestructura de elementos computacionales y actividades de ejecución. Un ejemplo de esto en Globus Toolkit es el servicio de ejecución remota GRAM (*Grid resource allocation and management service*), para el manejo de información RFT (*Reliable File Transfer*) o de control remoto de instrumentos GTPC (Grid *TeleControl Protocol*). Estos componentes permiten gestionar interacciones confiables y seguras, y proveen las bases para construir infraestructuras de mayor tamaño.

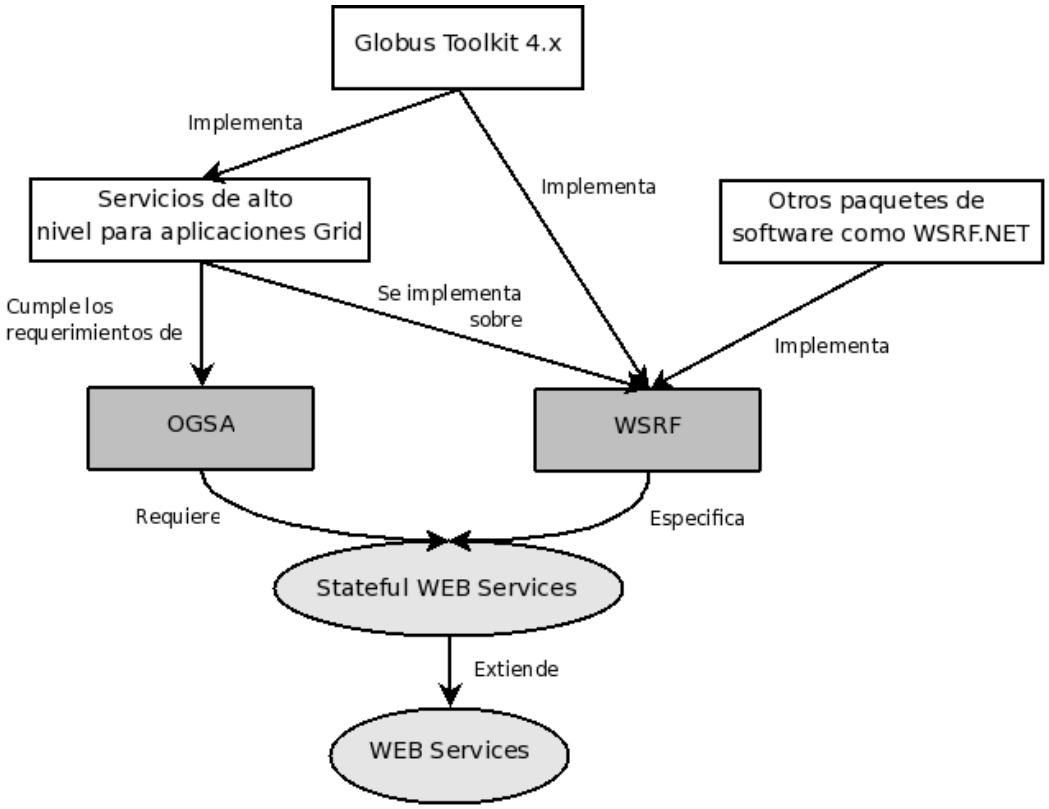


Figura 1. Diagrama por capas de OGSA, GT4, WSRF y Servicios WEB

4.3. Mecanismos de monitoreo y descubrimiento

Monitoreo y descubrimiento son dos funciones vitales en un sistema distribuido, en especial si el sistema se encuentra diseminado en distintas ubicaciones donde una sola persona no dispone de todo el conocimientos de los componentes. El monitoreo permite detectar y diagnosticar la mayoría de los problemas en estos contextos, mientras que el descubrimiento permite identificar recursos o servicios con las propiedades requeridas. Ambas tareas requieren la habilidad de obtener información de fuentes de información múltiple y distribuida.

Generalmente esta funcionalidad se encuentra presente en la infraestructura de los sistemas Grid. Características fundamentales de estos sistemas son la estandarización de la información basada en asociaciones de propiedades de los recursos descritos en lenguaje XML con entidades de la red que pueden ser accedidos con mecanismos de consulta o subscripción. Estos mecanismos se implementan usando el protocolo WSRF y WS-Notification [34] estándar para el servicio de notificación.

También es común contar con un mecanismo de agregación para obtener información de estado de las fuentes de información. Estos servicios de agregación pueden ser configurados para involucrar miles de elementos que pueden afectar el funcionamiento del sistema agrupándose en forma jerárquica por recursos o locación geográfica. Generalmente pueden ser accedidos por interfaz de navegación web o directamente en lenguaje XML, haciendo factible la función de monitoreo y la consulta por aplicaciones para funciones de descubrimiento.

4.4. Seguridad

La seguridad es importante en cualquier sistema de computación, pero cobra mayor importancia en un sistema distribuido con los recursos y usuarios distribuidos. Los administradores querrán ejercer control sobre quien puede invocar servicios a pesar de que estos no sean los propietarios de los recursos sino las organizaciones virtuales establecidas para la gestión de recursos compartidos. Ejercer el control puede contemplar acciones que aseguren políticas o funciones de auditoría. Cuando de diseñan mecanismos se debe tener en mente no solo la protección de la comunicación sino también las posibles intrusiones a los sistemas de cómputo. Una solución a estos inconvenientes siempre debe combinar métodos para la identificación, aplicar políticas y hacer seguimiento de las acciones.

La infraestructura Grid provee estos mecanismos, y a bajo nivel los componentes de seguridad estandarizados son el uso de credenciales y protocolos de protección de mensajes, autenticación, delegación y autorización. Uno de los protocolos que provee este tipo de seguridad es el X.509 con credenciales pública y privada. Estos protocolos se implementaron para permitir que dos entidades se validen sus credenciales mutuamente y establezcan un canal seguro de comunicación para la protección de mensajes. Además permite crear y transportar la delegación de credenciales para permitir que un componente actúe en representación de otro por un periodo de tiempo limitado.

4.5. Datos

Un tema común en las aplicaciones Grid es la necesidad de gestionar, proveer acceso e integrar grandes volúmenes de datos de uno o más ubicaciones geográficas. Este problema con los datos es engorroso y complejo y ningún software por si solo es capaz de dar una solución. En los sistemas Grid se incluyen distintos componentes y mecanismos para poder en conjunto dar distintos tipos de soluciones. Se proveen librerías para movimientos de dispositivos de almacenamiento en forma confiable, segura y de alta prestación. Generalmente son extensiones del protocolo de transferencias de archivos como FTP. También se encuentran disponibles servicios de replicación y localización de réplicas, que pueden accede a gran número de archivos y conjunto de datos en forma paralela. También se encuentra disponible un servicio de consulta de

integración de datos, que combina fuentes de datos como bases de datos relacionales y fuentes de datos XML permitiendo la consulta en forma paralela.

4.6. Coreografía

Un tema común a todas las aplicaciones Grid es la necesidad de coordinar las actividades de las diferentes ubicaciones. Las actividades a ser coordinadas pueden ser tareas computacionales, transferencias de información, operación de instrumentos, o monitoreo y control de operaciones, servicios o recursos físicos.

Según la perspectiva de coordinación que se ve en los sistemas Grid, se debe contemplar muchos aspectos por lo que un solo sistema no puede dar solución a todos los requerimientos. Los distintos sistemas Grid proveen distintas soluciones como por ejemplo facilidades para el desarrollo de coordinadores de alto nivel o meta-planificadores para la gestión de tareas o DAGs (Direct Acyclic Graphs) para la coordinación de tareas poco acopladas e interfaces con sistemas de librerías paralelas como MPI.

5. Planificación de Recursos

Dentro de una arquitectura orientada a servicios los elementos de que se han mencionado en la sección anterior son características del sistema que pueden ser tomados como recursos. Pueden ser servicios computacionales que ofrece una computadora como conexión a la red, espacio de almacenamiento o capacidad de cómputo. O servicios de virtualización como una base de datos, transferencia de archivos o simuladores, que pueden diferir en la función pero dan una vista consistente al usuario final de la manera en que se acceden desde la red. Por esto, podemos mantener el término de *gestión de recursos* para describir los aspectos del proceso para localizar determinados tipos de recursos, organizarlos para su uso o monitorear su estado.

La gestión de recursos es un problema ampliamente estudiado. Los gestores de recursos como los planificadores por lotes, motores de flujo de trabajo y sistema operativos han sido implementados en la mayoría de los sistemas de cómputo eficientemente. Estos gestores de recursos son diseñados y operados bajo la premisa de que poseen completo control de los recursos que administran y así implementar mecanismos y políticas para el correcto accionar de los mismos. Esta suposición no se cumple en los sistemas Grid: se deben desarrollar métodos para la gestión del sistema comprendiendo diferentes dominios administrativos, con heterogeneidad de recursos, falta de control y diferentes políticas [9].

Aún cuando los recursos son del mismo tipo, como por ejemplo grupos de máquinas para altas prestaciones, estos grupos de máquinas pueden no poseer la misma configuración de software, gestión de recursos local, requerimientos administrativos, etc. Por estos motivos, uno de los problemas a solucionar es poder salvar el problema de heterogeneidad. Incluso más problemático es que las distintas organizaciones operen sus recursos bajo diferentes políticas, el requerimiento de los usuarios y de los proveedores del recurso puede ser inconsistente o conflictuar. A esto se agrega el hecho de que en un sistema Grid se requiere el uso concurrente de múltiples recursos necesitando una estructura para la coordinación de múltiples dominios.

La planificación en Grid se define como el proceso de tomar decisiones de planificación sobre distintos recursos en diferentes dominios administrativos. Este proceso involucra la búsqueda en distintos dominios de una máquina o el envío de una tarea sobe un sitio geográfico o múltiples sitios. Se define como tarea cualquier software que necesite un recurso, desde ancho de banda hasta un conjunto de aplicaciones.

En general se puede diferenciar entre en planificador Grid y un planificador de recursos locales en que este último es responsable de la planificación y gestión de recursos en un solo sitio. Otra diferencia es que el planificador Grid no es dueño de los recursos y por lo tanto no tiene control sobre ellos. El planificador Grid tampoco tiene el control de los trabajos que se encuentran en la cola o se encuentran corriendo sobre el recurso, por lo que decisiones globales entre coordinación de trabajo no tienen sentido.

Los pasos para la planificación en sistemas Grid son tres:
- El descubrimiento de los recursos, que generalmente involucran una lista y ubicación de los posibles recursos.
- Obtención de la información para poder realizar filtros sobre estos y elegir el mejor
- La ejecución de la tarea que incluye las distintas etapas desde la configuración del entorno hasta la limpieza de los archivos temporales.

5.1. Búsqueda de recursos

El primer paso en cualquier planificación involucra la búsqueda de qué recursos están disponibles para un determinado usuario. La fase de descubrimiento implica seleccionar un conjunto de recursos para ser estudiados en la fase de selección del sistema. Como resultado de los pasos en la búsqueda se tendrá una lista con todos los recursos candidatos para ser seleccionados.

- **Autorización:** El primer paso es determinar el conjunto de recursos a que el usuario a ejecutar el trabajo tiene acceso. La principal diferencia en los sistemas Grid es la cantidad de recursos. Se puede realizar fácilmente accesos remotos pero es difícil realizar su administración: por ejemplo es sencillo con los sistemas actuales saber dónde se encuentran los recursos pero es complejo saber a cuál de ellos se tiene derechos de acceso. La opción más común es llevar una lista con un mapeo de los usuarios globales a cada uno de los sistemas locales.

- **Definición de requerimientos de la aplicación:** Para continuar en el descubrimiento de recursos, el usuario debe especificar el conjunto de requerimientos mínimo para encontrar los recursos adecuados. El conjunto de requerimientos varía ampliamente entre tareas. Puede contener información estática, por ejemplo sistema operativo, hardware para el que los binarios del código se encuentran compilados, o arquitecturas especificas. También se pueden especificar información dinámica como requerimientos de memoria, conectividad, o espacio de almacenamiento. Actualmente esta información se especifica como parte de la línea de comando cuando se ejecuta la tarea, por ejemplo en planificadores como PBS[10], LSF[11] o ClassAd[12]. En un sistema Grid la solución se complica por el hecho de que los requerimientos cambiarán según el sistema que se asigne; por ejemplo dependiendo de la arquitectura pueden llegar a cambiar los requerimientos de memoria o las librerías necesarias.

- **Filtrado según requerimientos:** Una vez que se tienen los requerimientos provistos por el usuario se deben filtrar los recursos que puedan cumplir los requisitos. Los sistemas Grid proveen información sobre los datos estáticos y dinámicos de los diferentes recursos. La mayoría de estos datos se actualiza cada un tiempo predeterminado o deben realizarse consultas a demanda. Debido a que los recursos se encuentran distribuidos, obtener el último estado de los recursos puede resultar costoso en términos de tiempo. La mayoría de los sistemas incorpora una funcionalidad de consultas distribuida para paralelizar este proceso siendo este un potencial cuello de botella en la selección de recursos.

5.2. Selección del sistema

Dado un grupo de posibles recursos todos los cuales cumplen requerimientos mínimos para la tarea, se debe seleccionar cuáles de ellos son los más adecuados. Esta selección generalmente se realiza en dos pasos: obtención de información y toma de decisión.

- **Compilación de información dinámica:** Para poder hacer la mejor combinación entre la tarea y los recursos se necesita disponer de información dinámica (por ejemplo sobre los gestores de planificación local). En los sistemas Grid también se suma información de las políticas que se llevan adelante en los distintos dominios administrativos y cómo estas de implementan. Generalmente en los sitios se cuenta con límites para el consumo de almacenamiento, tiempo, etc.

- **Selección del sistema:** Con la información dinámica disponible el próximo paso es decidir qué recurso usar. Uno de los aportes del presente trabajo impacta directamente sobre este punto buscando una solución que optimice el uso de los recursos.

5.3. Ejecución del Trabajo

En la ejecución de una tarea se realiza la configuración, ejecución, monitoreo y finalmente la limpieza de archivos temporales en el recurso que se acaba de usar.

- **Reserva de recursos:** Para hacer mejor uso del sistema, los recursos deben ser reservados con anticipación. Dependiendo de los recursos estas tareas pueden ser complejas y no siempre es posible realizarlas de manera automática. Además la reserva puede expirar y tener un costo asociado. Pero el punto de mayor importancia en la reserva del recurso es el soporte que proveen en cada sitio y esto no está implementado en la mayoría de los recursos disponibles.

- **Ejecución de la tarea:** Una vez que el recurso o el conjunto de recursos ha sido elegido, la tarea debe ser enviada y ejecutada remotamente. Si bien esto tiene complejidad técnica, es uno de los temas en donde se ha incorporado mayor automatización y puede llegar a ser sólo la invocación de un comando conteniendo gran cantidad de otros archivos. Para los sistemas Grid es necesario contar en este punto con estándares para que la ejecución de las tareas pueda realizarse sobre los distintos recursos siendo estos heterogéneos.

- **Preparación de la tarea:** Para la preparación del entorno generalmente se deben realizar pasos de configuración, movimiento de datos, o de ejecutables. En los sistemas Grid esto puede involucrar gran cantidad de sitios siendo el problema de escalabilidad el de mayor importancia.

- **Monitoreo del Progreso:** Dependiendo de la aplicación y el tiempo de ejecución, el usuario puede monitorear el progreso en la ejecución. Este es un proceso habitual en cualquier sistema de cómputo y generalmente se verifica realizando una consulta sobre el recurso que se está utilizando en ese momento. En los sistemas Grid el monitoreo cobra mayor importancia: por ejemplo, si una tarea no avanza en su ejecución lo suficiente puede que se encuentren disponibles otros recursos que se adapten más a las necesidades de la aplicación y donde ésta pueda ejecutarse más eficientemente. Una vez detectados, se verifica el costo de la migración debiendo volver al paso de selección del sistema del proceso de planificación.

- **Finalización de la tarea:** Cuando la tarea finaliza los usuarios debe ser notificados. Generalmente los comandos de ejecución en los sistemas paralelos tiene un sistema de aviso por correo electrónico. También en este tipo de sistemas que involucra gran cantidad de recursos detectar cuándo la aplicación no ha terminado correctamente es un tema aún en investigación. Finalmente una vez que la tarea ha terminado correctamente, los archivos generados deben ser rescatados de los distintos sitios y se debe llevar en cuenta la información temporaria para poder eliminar cualquier rastro de la ejecución.

6. Espacios de Trabajo Virtuales

Como se ha descrito en la sección anterior, la interacción se basa en el mapeo de los trabajos a los recursos, con la suposición de que el entorno de ejecución está en su mayor parte configurado y sus características son independientes de la infraestructura del sistema Grid. Esta suposición es verdadera para un número importante de usuarios en Grid, pero no lo es tanto cuando aplicaciones o usuarios varían drásticamente sus requerimientos tratando de usar los mismos recursos.

Si se toma en cuenta esta característica, podemos definir el término de *espacio virtual* como un contenedor que puede ser automáticamente montado sobre los recursos existentes y proveer en el entorno de ejecución adecuado. De esta manera las tareas se mapean a los espacios de trabajo y los espacios de trabajo son mapeados a los recursos Grid.

Las máquinas virtuales (MV) [13] les permiten a los usuarios crear entornos de ejecución configurados con los requerimientos de sistema operativo, software y políticas de acceso, todo esto montado sobre una capa de abstracción de software llamada *hypervisor*. Además de esto, el estado de la MV puede ser serializada en imágenes, permitiendo al usuario realizar pausas, apagarlas y retomar la actividad en otro momento o lugar desacoplando la generación de la imagen de su ejecución permitiendo su migración. Las MV también permiten realizar implementación de políticas de ejecución, restringiendo asignación de memoria, tamaño de disco pudiendo y capacidad de uso de procesador, estas características pueden ser administradas en tiempo de ejecución. Por todas estas razones, las máquinas virtuales son una solución ideal para la creación de espacios de

trabajo. La configuración de la imagen de la MV refleja los requerimientos de software del espacio de trabajo mientras que el hypervisor asegura que las políticas de las propiedades de hardware sean respetadas.

Las MV tienen la ventaja de la flexibilidad y rapidez de despliegue. La flexibilidad proviene del propio concepto de virtualización, que provee una abstracción de la representación del estado que puede ser desplegado en cualquier lugar donde esté presente el hypervisor. En los hypervisores actuales este despliegue puede tomar solo unos segundos, que es despreciable comparado con la sobrecarga de las herramientas de los sistemas Grid. El hypervisor permite controlar y asegurar políticas de detalle fino pero esto es también su punto débil ya que es dependiente que el sistema provea un hypervisor y que este sea compatible para que esta solución sea factible.

Los espacios de trabajos implementados por esta vía pueden ser agrupados en diferentes topologías. Se puede construir un cluster virtual, implementado en términos de múltiples imágenes representando recursos especializados como nodos de cómputo o cabeza del grupo de máquinas. En este caso algunas de las máquinas como los nodos especializados en cómputo deberán tener la misma configuración optimizando el tiempo de desarrollo y despliegue de la solución.

La descripción de este tipo de espacios de trabajo debe contener suficiente información para crear el entorno representado por el espacio de trabajo. Esta información es de dos tipos:

- La descripción de los paquetes o los datos que son necesarios obtener de fuentes externas.
- La información para el despliegue de la solución que necesita ser interpretada y configurada en tiempo de ejecución, como por ejemplo el tipo de conexión a la red desde la MV.

La cantidad de información que debe ser descargada y el tiempo para la configuración de despliegue depende de la implementación del espacio de trabajo, si esta imagen esta pre-configurada el tiempo se reduce considerablemente. De esta manera existen distintos métodos para encarar esta problemática, desde tener un conjunto de imágenes disponibles para armar el espacio de trabajo hasta tener una infraestructura para poder generar las imágenes en el momento en que fuese necesario.

Los metadatos a tener en cuenta para la generación de este tipo de entornos son múltiples. Los módulos del espacio de trabajo pueden ser implementados por particiones de las MV o discos que forman una imagen de MV. El servicio de despliegue debe ser capaz de resolver cómo obtener estas particiones, verificar su integridad e instanciar la MV tal que la imagen correcta sea cargada en el dispositivo específico. Para completar la instanciación también es necesario describir la información de despliegue (por ejemplo cómo proveer conectividad y almacenamiento). Para reflejar el hecho de que se pueden tener múltiples máquinas virtuales sobre una máquina física es necesario tener capacidad de definir un multiplexado de interfaces de red sobre una interfaz virtual y de esta manera acceder a redes privadas virtuales, servicios de red como NAT o definición de direcciones de internet o redes locales.

Si bien los metadatos describen las partes que comprenden las MV, estas no lo incluyen. Las imágenes, así como también las descripciones, se encuentran en la infraestructura de la organización virtual. Estos espacios de trabajo pueden ser compartidos, copiados o definidos incrementalmente como agregación de múltiples espacios de trabajo adecuándose a diferentes requerimientos. Siguiendo con la arquitectura orientada a servicios, se deberán proveer los servicios adecuados soportando las operaciones que permitan la generación y configuración de los espacios de trabajo. Algunos ejemplos de estas infraestructura pueden versen en [14,15,16,17].

Capítulo 2 Aplicaciones Paralelas a Grid

Este capítulo tiene como objetivo poder evaluar cómo afecta un entorno Grid a la interconexión de clusters dedicados a través de Internet. En primer lugar se seleccionaron aplicaciones paralelas, se realizaron experiencias de adaptación al entorno Grid y luego se efectuaron búsqueda de patrones de conexión y predicciones en el comportamiento de transferencia de datos sobre clusters y el conjunto interconectado mediante Internet. Por último, se realizó una optimización en la asignación de equipamiento disponible en los distintos clusters para lograr un rendimiento efectivo del poder de cómputo del conjunto. A través del trabajo se pudo comprobar que los entornos Grid no alteran el tipo de métricas empleadas para estimar aplicaciones paralelas, ya que la carga extra de su ejecución, para tiempos de procesamiento suficientemente extensos, no es significativa. Sí, en cambio, se torna necesario adaptar las aplicaciones a un nuevo modelo de implementación basado en servicios.

1. Ejecución remota sobre entornos Grid

Para lograr un entendimiento más profundo de las diferencias de ejecución remota sobre un sistema Grid y las aplicaciones paralelas típicas se realizó un estudio de las capas de software que intervienen en la ejecución remota en este tipo de sistemas. Además se realizó un estudio de la sobrecarga que cada una de estas capas genera en tiempo de ejecución sobre el sistema.

Las herramientas de ejecución remota en los entornos Grid [26] proveen funcionalidad para la iniciación, monitoreo, administración, planificación y coordinación de computación remota. El software utilizado para esta tesis es Globus Toolkit 4 (GT4), el cual provee una interfaz llamada GRAM (Grid Resource Allocation and Management) para la ejecución remota de tareas.

GRAM está diseñado para trabajar en conjunto con el servicio de delegación (que forma parte del sistema de seguridad de Globus) y con el servicio de RFT (Reliable File Transfer), componente del sistema de transferencia de información. Cada uno de estos servicios provee APIs para acceso a su funcionalidad, que puede ser implementada en los lenguajes C, Java y Python.

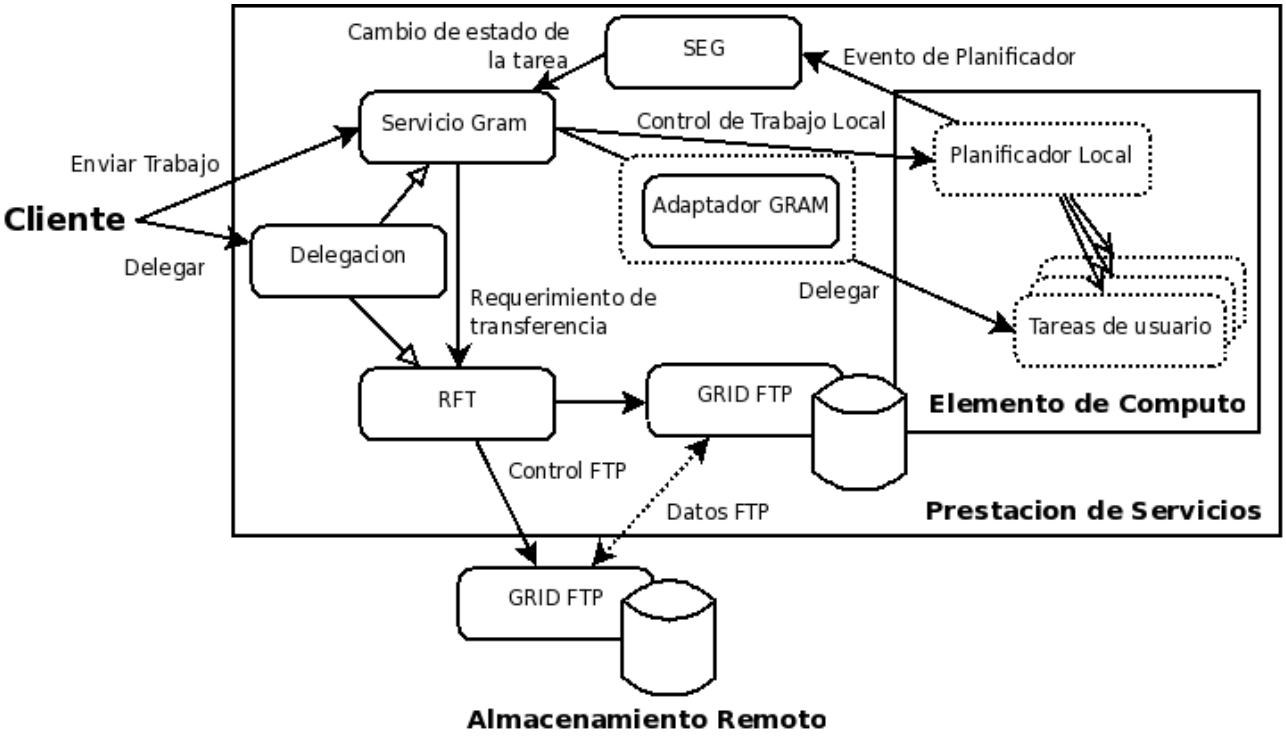


Figura 2. Arquitectura de ejecución remota en Globus Toolkit 4

En la Figura 2 se pueden ver cada uno de los módulos que intervienen en la ejecución remota de tareas. En primer término, de izquierda a derecha, se observan las acciones concernientes al cliente, que son el envío de la tarea y la delegación de la credencial para el esquema de seguridad, de acuerdo al protocolo X.509. En el servidor este mensaje es recibido por los módulos antes mencionados, comenzando con la coordinación, el pasaje de datos necesarios para el inicio de ejecución de las tareas, la comunicación con los planificadores locales y el monitor de tareas.

En la Figura 3 se describe la secuencia de acciones en la ejecución de las tareas, donde participan seis actores. Tal como en la figura anterior, las acciones comienzan por el cliente. Las primeras acciones (TA) corresponden a establecer las tareas en lo relativo a la seguridad y configuración. Una vez que el usuario ha sido validado, puede comenzar con la creación remota de la tarea conectándose al servicio GRAM (TB); éste almacena las credenciales e inicia la transferencia de archivos necesarios durante la ejecución de la tarea (TC). Algunas tareas dentro de este conjunto se muestran en rayado oblicuo indicando que la tarea a ejecutar tiene requerimientos de transferencia de datos para iniciar la ejecución, por lo cual se debe iniciar la transferencia comunicándose con el servicio RFT.

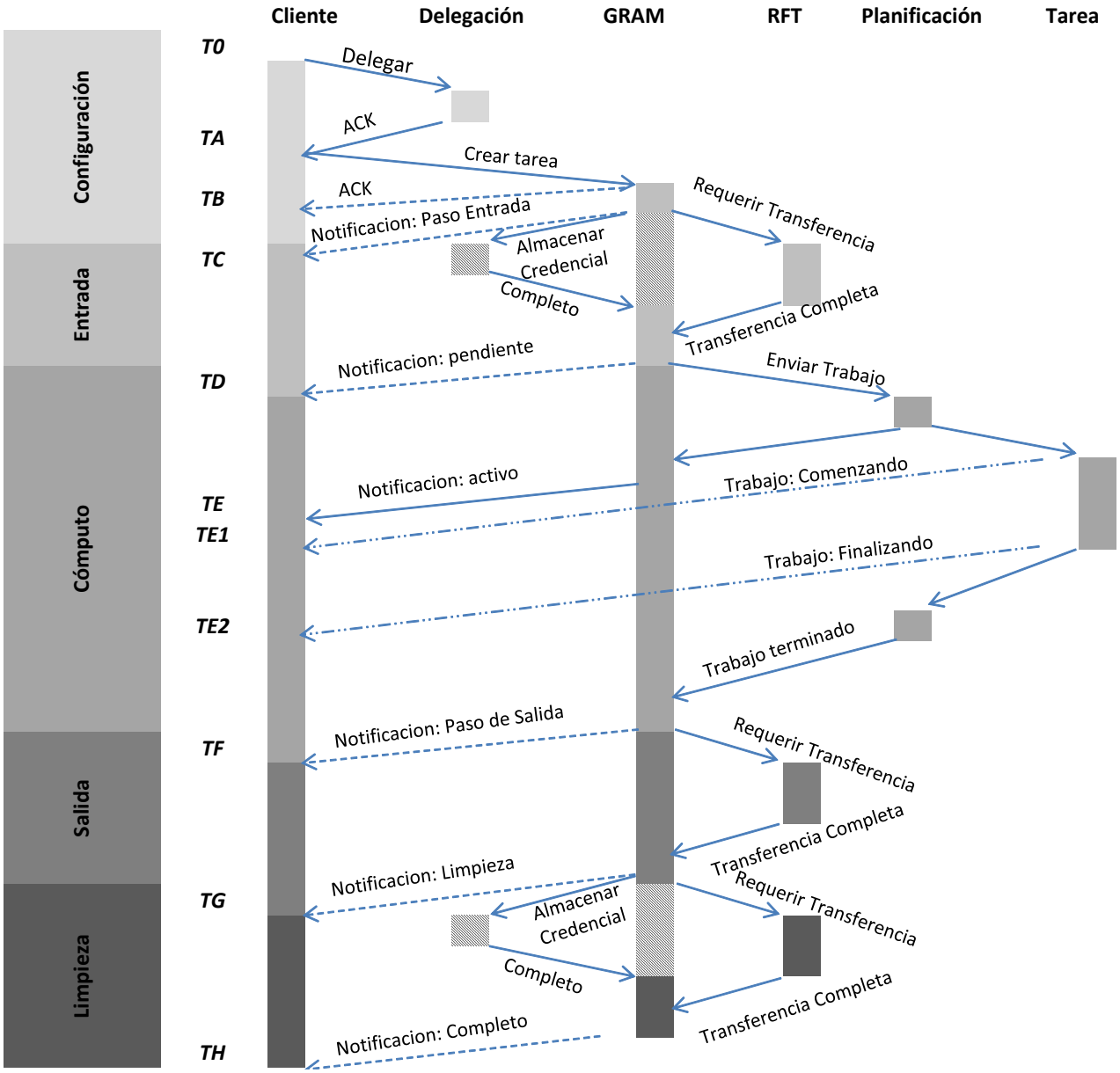


Figura 3. Diagrama de Secuencias en la ejecución de tareas remotas

Una vez que los datos se encuentran en la máquina remota y se puede iniciar la ejecución (TD), comienza la secuencia, donde la tarea se ejecuta efectivamente comunicándose con el planificador local (TE). Finalizadas las tareas, se envía la notificación de terminación y se obtienen los datos de salida (TF-TG), limpiándose el almacenamiento temporal que haya sido usado en la ejecución.

1.1. Tiempos del Servidor en ejecución remota

Con el código involucrado en la ejecución remota de tareas en entornos Grid se realizó una depuración de la ejecución con la herramienta Log4j [30], un esquema del lenguaje Java para la depuración de ejecución de aplicaciones. A modo de ejemplo, el resultado de este análisis para la ejecución de tareas en un servidor remoto de la Universidad del Comahue se presenta en la Tabla 2. Como resultado se puede observar que la carga del middleware (independiente de la tarea realizada) es de menos de un minuto.

Segs	Actividad	Tarea
39	Inicia la validación del Archivo	**Configuración**
39	Realiza el Mapeo del Usuario Grid al usuario local	
39	Carga el archivo de descripción con los comandos	
40	PROCESSING INTERNAL STATE: -- None --	
40	PROCESSING INTERNAL STATE: -- Start --	**Transferencia de datos de entrada**
40	PROCESSING INTERNAL STATE: -- StageIn --	
40	PROCESSING INTERNAL STATE: -- Submit --	**Cómputo**
42	PROCESSING INTERNAL STATE: -- WaitingForStateChanges --	
42	PROCESSING INTERNAL STATE: -- OpenStdout --	**Transferencia datos de resultado**
43	PROCESSING INTERNAL STATE: -- OpenStderr --	
43	PROCESSING INTERNAL STATE: -- WaitingForStateChanges --	
43	PROCESSING INTERNAL STATE: -- MergeStdout --	
43	PROCESSING INTERNAL STATE: -- StageOut --	
43	PROCESSING INTERNAL STATE: -- CleanUp --	**Limpieza y Cierre**
43	PROCESSING INTERNAL STATE: -- FileCleanUp --	
43	PROCESSING INTERNAL STATE: -- CacheCleanUp --	
44	PROCESSING INTERNAL STATE: -- Done --	

Tabla 2. Tiempos de ejecución de funcionalidad de servicio

Se puede concluir según la Tabla 2 que se tiene un costo independiente de la tarea de 40” de inicialización. La tarea que se está analizando no tiene transferencia de archivos de salida ni entrada y el solo la escritura en pantalla de una línea de chequeo. Para cómputos que pueden durar varias horas no presenta una sobrecarga significativa.

2. Entorno de Experimentación

Una vez que se obtuvo un conocimiento necesario sobre la ejecución de tareas en el entorno Grid y se encontraron los puntos de carga del software intermedio, se procedió al análisis de cómo las aplicaciones paralelas podrían ser adaptadas. La siguiente etapa no solo pretende ejecutar las aplicaciones en forma remota en un cluster, sino que puedan ser distribuidas en forma coordinada en múltiples clusters heterogéneos distribuidos en el entorno Grid. Como entorno de experimentación se seleccionaron dos aplicaciones típicamente paralelas y se creó un banco de pruebas distribuido en distintos ubicaciones geográficas con cuatro clusters.

2.1. Aplicaciones

Uno de los tipos de aplicaciones paralelas habituales son las relacionadas con aplicaciones graficas o de manejo de video. POV-Ray (Persistence Of Vision Ray Tracer) es considerado habitualmente una referencia en su género. Sin embargo, su capacidad de procesamiento paralelo no se obtiene a partir de la versión original del software sino aplicando una modificación al mismo. El segundo programa analizado fue Tachyon, que incorpora el procesamiento paralelo desde el diseño. Este factor sumado a su mayor simplicidad fueron los factores determinantes para su elección, a pesar de que no es un programa tan completo como POV-Ray especialmente en cuanto al poder del lenguaje de descripción de escenas y capacidad de *parsing*.

Se utilizó la versión 0.97 de Tachyon, (la última disponible), que ofrece una gran variedad de opciones de compilación para diferentes plataformas. Tachyon es parte del conjunto de librerías de SpecCPU y SpecMPI estándares para realizar pruebas de rendimiento en equipos multi y mono procesador. En la plataforma Linux, Tachyon puede ser compilado en versión secuencial, en versión multihilo para arquitectura SMP con memoria compartida, y además está implementado sobre MPI, tanto con soporte MPICH como con LAM.

Tachyon maneja un lenguaje de descripción de escenas que ofrece la posibilidad de utilizar primitivas que describen objetos geométricos básicos, ejecutar *antialiasing (minimización de la distorsión visual)* y mapeo de texturas. Es capaz de leer un conjunto de datos representando un camino para la cámara virtual que visualiza la escena, y crear así una secuencia de imágenes con la que se puede construir una animación. Hay varios parámetros que influyen sobre la cantidad de esfuerzo computacional que se llevará a cabo durante el *rendering* de una escena dada, y que pueden modificarse fácilmente mediante las directivas que describen la escena. Por ejemplo, el tamaño o resolución de la imagen; la profundidad de recursión o la cantidad de niveles de reflexión y refracción que se tendrán en cuenta en el proceso de *raytracing;* y el nivel de *antialiasing*, relacionado con la cantidad de radios extras que generará el *raytracer* para elevar la calidad de la imagen computada. Las variaciones sobre estos parámetros sirvieron para generar diferentes pruebas y elevar el trabajo computacional del cluster para una escena a un nivel fácilmente observable mediante las herramientas de monitoreo. Al trabajar en modo paralelo, Tachyon efectúa de forma transparente la descomposición espacial de las escenas mediante una malla rectangular, para distribuir el trabajo entre los nodos del cluster.

La otra aplicación paralela utilizada para implementar la transformación a entornos Grid es el desarrollo de un modelo de transmisión sináptica de neuronas. La resonancia estocástica es un fenómeno que juega un papel fundamental al detectar débiles señales periódicas en presencia de ruido. Este fenómeno es de importancia en neurofisiología, en el estudio de la transmisión sináptica entre neuronas (TSN), siendo las neuronas modeladas por un sistema biestable, con dos estados estables: de disparo y de inactividad. El proceso ya fue simulado mediante un anillo de osciladores armónicos sobre-amortiguados con transmisión unidireccional, que operando en un régimen de resonancia estocástica puede comportarse como una unidad de memoria de corto alcance sostenido por ruido. Este trabajo es parte del resultado de las investigaciones que llevan adelante el grupo de Sistemas Complejos de la Universidad Nacional de

General Sarmiento [27][28].

2.2. Arquitectura del sistema de Cómputo

Para los distintos experimentos se montaron cuatro clusters, y sobre ellos se compiló e instalaron las dos aplicaciones antes mencionadas. El objetivo fue la caracterización de estos grupos de máquinas como prestadoras de servicios dentro de una organización virtual. El banco de prueba está implementado sobre clusters, dos de los cuales se encontraban instalados físicamente en el departamento de Ciencias de la Computación de la Universidad Nacional del Comahue, el tercero en el CRIBABB de Bahía Blanca y el cuarto en una empresa privada de la ciudad de Neuquén. Los clusters de la UNC son similares en cuanto a características de procesamiento: en todos los casos los equipos son Pentium III de 500 MHz, con memoria de 128MB. Cada cluster cuenta con cuatro equipos, incluido el master. Los equipos de la UNC difieren de aquellos con que se cuenta en la empresa ya que estos últimos son cinco máquinas Pentium IV de 3.06 GHz con tecnología Hyper-Threading y memoria de 1 GB. La Tabla 3 resume estas características de hardware.

En cuanto al software, en la configuración de los clusters de la UNC se utilizó el software Oscar 4.1, un asistente en la instalación y mantenimiento de clusters de computadoras. Para la instalación se seleccionaron librerías MPICH 1.2.7, un software C3 para la gestión del cluster y planificadores de cluster Open PBS. Todas las máquinas de la UNC se configuraron con la distribución de Linux Fedora Core 4. Gracias a la funcionalidad del software Oscar, sólo se debió configurar el nodo master, y desde allí generar imágenes en forma automática e instalar el resto de los nodos workers. En el cluster de CRIBABB la instalación fue sin ayuda de herramientas de gestión de clusters lográndose la misma configuración final que el cluster de la UNC con MPICH 1.2.7. En cuanto al cluster de la empresa, se instaló sistema operativo Fedora Core 5 y se instalaron las librerías MPICH, en dos versiones: 2 y 1.2.7. Con la versión 2 de MPICH se encontraron problemas en la ejecución desde Grid, ya que en esta versión cambia su implementación, por lo que las pruebas fueron realizadas con versión 1.2.7. En todos los casos la red local es de 100Mbps. Cada cluster tiene una computadora que actúa como procesador frontal (G1, G2, M y Router CDF) para el acceso y administración de sus nodos. En cada cluster, los nodos se encuentran interconectados por una red Ethernet de 10/100 Mbps.

Cluster	***Master***			***Workers***		
	CPU	**Memoria**	**Sistema Operativo**	**CPU**	**Memoria**	**Sistema Operativo**
G3	PIV 2.2	256Mb Ram	Fedora Core 4	PIII 500 Mhz	128 Mb Ram	Fedora Core 4
G4	PIV 2.2	256Mb Ram	Fedora Core 4	PIII 500 Mhz	128 Mb Ram	Fedora Core 4
CDF	PIV 3.0	1Gb Ram	CentOS 4	PIV 3.0 MHz	1Gb Ram	Fedora Core 5
CRIBBAB	PIV 3.0	1GB Ram	Fedora Core 3	Celeron 333 MHz	128MB Ram	Linux RH9

Tabla 3. Configuración de equipos

El Grid se implementó a partir de los tres procesadores frontales de cada cluster, para lo cual se les instaló el software Globus Toolkit 4.0. Como se observa en la Figura 4, los nodos Grid G1 y G2 están conectados localmente mediante una red Ethernet, con direcciones IP privadas. El acceso de esta red a Internet se realiza a través de un

dispositivo NAT, el cual también cumple la función de corta fuegos.

Por medio de Internet se conectan G1 y G2 con el nodo M de CRIBABB y Router CDF de la empresa CDF. Uno de los inconvenientes observados en esta topología se produjo por el esquema de seguridad utilizado por Globus al ejecutar tareas remotas, debido a las diferentes direcciones IP (pública y privada) utilizadas por la aplicación por la presencia del NAT. Para solucionar este problema se implementó la propuesta de Von Welch [31] para cortafuegos, pero no dio los resultados esperados debido a que no se logro controlar la creación de puertos efímeros en los canales de comunicación. Se optó por configurar una red privada virtual (VPN) seleccionándose OpenVPN como software de soporte para la comunicación entre los nodos remotos. Para todas las pruebas se utilizó IPv4. Para evitar conflictos en la delegación de credenciales se deshabilitó el protocolo IPv6.

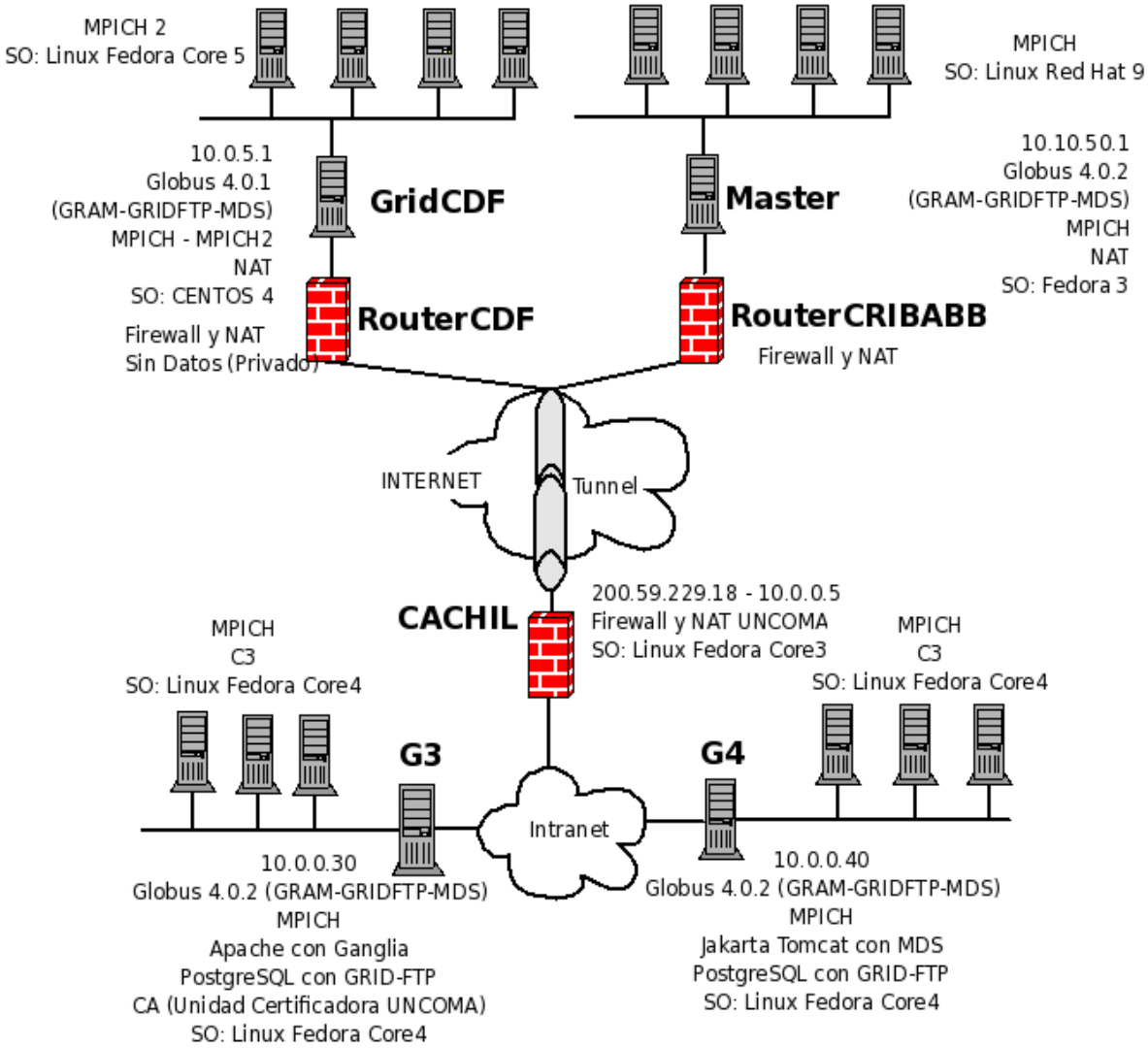


Figura 4. Arquitectura del Sistema de Computo

En el aspecto de monitoreo se vinculó el servicio MDS con los datos de Ganglia (Figura 5); con este fin se utilizó la propiedad GLUE provista por el Globus Toolkit. Esta obtiene información de distintas fuentes como el planificador, o el sistema de información del

cluster (por ejemplo Ganglia [32], que es el utilizado en este trabajo o Hawkeye monitor sistema Condor [33]). Esto une en una sola salida estándar la propiedad del recurso. En la Figura 5 se ve un ejemplo del sistema MDS donde se muestran los distintos servicios provistos por los clusters con la palabra GRAM y los servicios de transferencia de archivos con la palabra RFT. Las informaciones obtenidas de los servicios de ejecución serían la totalidad de la información provista por Ganglia. De esta manera, bajo una misma interfaz, obtenemos los datos del sistema Grid y de los clusters en forma particular desde una interfaz centralizada.

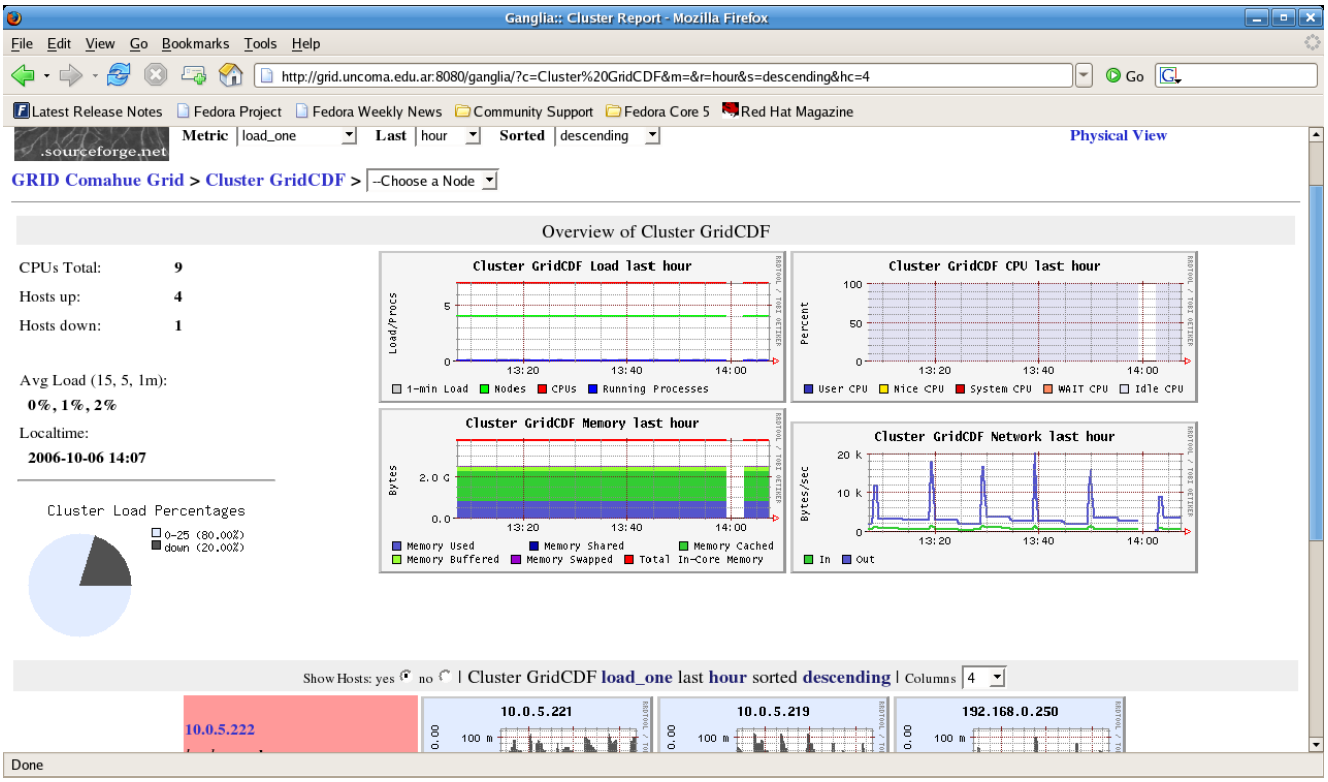


Figura 5. Interfaz de control centralizada. Uso de software Ganglia

3. Adaptación de la Aplicación Paralela

3.1. Transformación a Grid Aplicación Gráfica

El funcionamiento de la aplicación gráfica es básicamente el procesamiento de una especificación de una secuencia de imágenes o cuadros. El tratamiento de cada uno de estos cuadros se puede paralelizar sobre los nodos de los clusters; por ejemplo, si se desean generar 128 cuadros de animación de una escena, el proceso de ejecución debe generar el archivo con formato XML con los trabajos para cada cluster y enviar a ejecución a un cluster los primeros 64 cuadros y al mismo tiempo enviar al segundo cluster los próximos 64 cuadros. Una vez que ambos finalizan su ejecución, se combinan los frames para formar archivos en formato de video MPEG-1, y éstos se recuperan desde cada uno de los clusters combinándolos en uno solo para luego publicarlo en Internet. En primer término se realizaron ejecuciones generando el total de cuadros del movimiento de la cámara en cada uno de los clusters para observar su comportamiento.

En la Figura 6 se ven los tiempos de ejecución para diferentes cantidad de procesos se encuentran distribuidos sobre el eje de las abscisas. En los clusters G1 y G2 solo se llegan con valores hasta 8 procesos ya que incrementando esta cantidad las máquinas detenían su procesamiento por falta de memoria. Si se analiza el gráfico, se puede observar que en los clusters del Comahue, G1 y G2, existen mínimos en los tiempos de procesamiento cuando se ejecutan con cinco procesos y para el cluster del CRIBABB el mínimo se obtiene con seis procesos, en ambos casos un proceso más que la cantidad de máquinas del cluster. Esto se debe a la forma en que implementa la paralelización la aplicación Tachyon. Este examen previo permitió seleccionar la cantidad adecuada de procesos con los que se ejecutaran las corridas con los clusters sincronizados y utilizar de manera eficiente los equipos asignados.

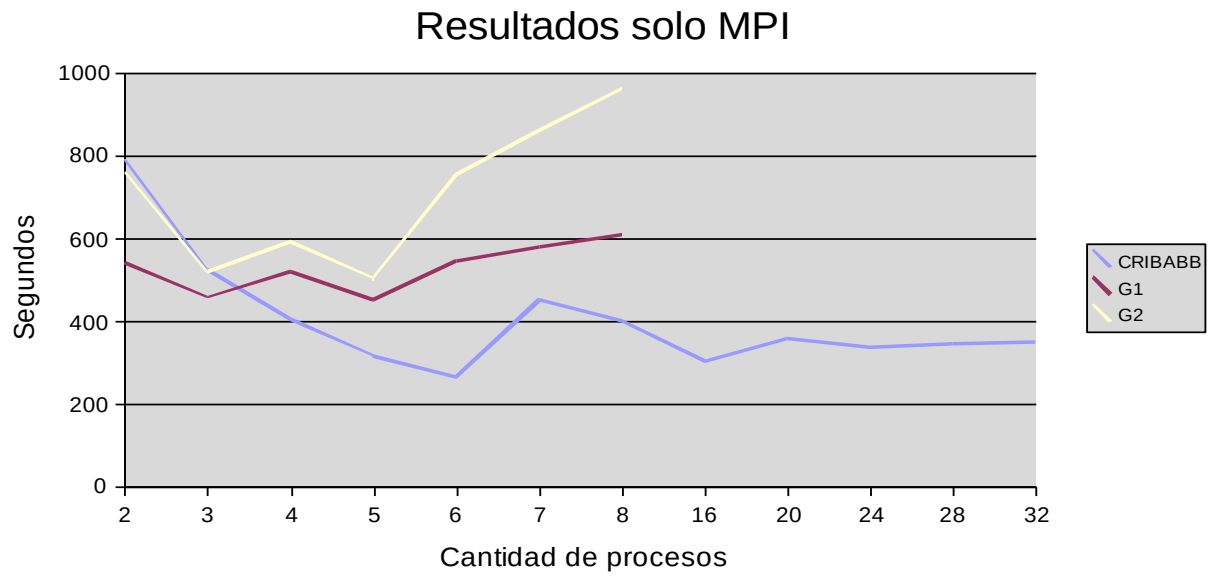


Figura 6. Análisis de asignación optima de equipos para procesamiento Tachyon

Para la ejecución de los clusters se tomó el archivo de control de movimiento de cámara y se lo dividió de distintas maneras realizando ejecuciones en forma paralela sobre los diferentes clusters. Con este fin se conformó un archivo de especificación de trabajo de múltiples tareas y transferencia de archivos, en lenguaje XML, según el lenguaje de especificación del Globus Toolkit. Los resultados obtenidos en las ejecuciones pueden verse en la Figura 7 Las tres primeras columnas muestran los mejores tiempos de los clusters sin la intervención del software de Grid; las dos columnas siguientes, muestran los tiempos de ejecuciones paralelas entre dos y tres clusters respectivamente. En las ejecuciones de los clusters en conjunto los tiempos disminuyen. En la ejecución de los clusters CRIBABB-G1, los dos más veloces, el tiempo resultado es casi un promedio de los dos mejores tiempos. En cambio si agregamos el tercer cluster y ejecutamos en forma paralela, si bien la diferencia de tiempo decrece sensiblemente es menor que se realizara el promedio de los tres. Es necesario remarcar que con esta experiencia no se buscó establecer análisis de performance entre las distintas modalidades de ejecución, sino que el objetivo es establecer la factibilidad del uso de la tecnología Grid, tomando en cuenta la sobrecarga que añade este software intermedio sobre entornos heterogéneos dispersos geográficamente.

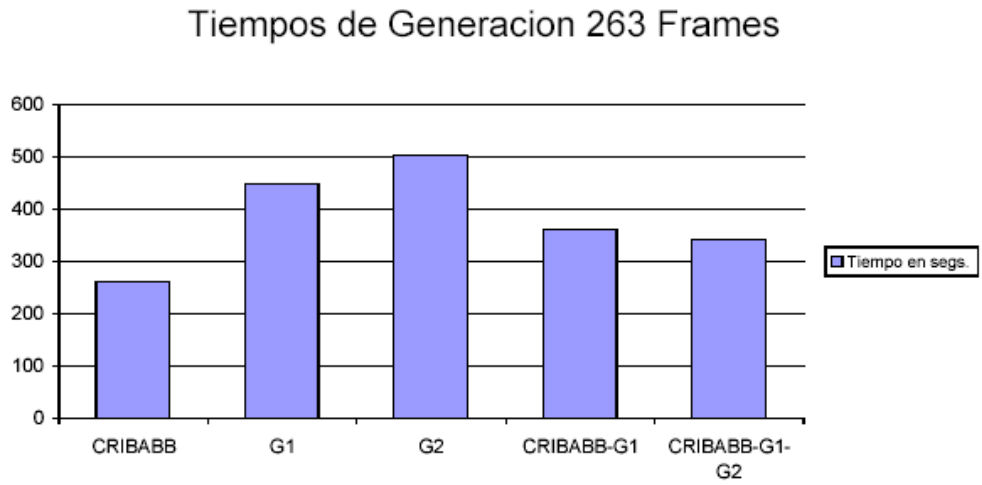


Figura 7. Tiempos de ejecución promedio según combinaciones en el banco de prueba.

En esta primera etapa no se contempló en los tiempos de ejecución el impacto de la latencia de red en la transferencia de datos y tareas; si se tomó la sobrecarga en los nodos del Grid debido a la ejecución de Globus y la capacidad de procesamiento del hardware instalado. En segundo término, esta etapa del trabajo permitió concluir que existen umbrales definidos para los que una aplicación paralela distribuida en un entorno Grid tiene rendimiento óptimo. Estos umbrales óptimos dependen del tipo de conexión a la red interna y externa, además de la capacidad de procesamiento de los nodos de cómputo y la combinación de clusters del banco de prueba asignados al procesamiento de la aplicación. Esta combinación también afecta a la distribución de la aplicación entre los distintos nodos Grid.

3.2. Transformación a Grid de una Aplicación Simulador

Como paso siguiente a las conclusiones obtenidas de las experiencias de modificar la aplicación gráfica, el objetivo fue encontrar una metodología para determinar los umbrales óptimos de eficiencia además de distribuir una aplicación paralela en un entorno Grid.

En los entornos Grid habitualmente se utilizan dos formas de ejecutar aplicaciones paralelas. La primera es compilar las librerías MPICH con dispositivo Globus, lo que permite cambiar las máquinas del entorno local a un entorno de servidores Grid. Esto implica tener gran cantidad de servidores Globus accedidos desde Internet. La otra opción es usar un servidor Globus por cluster y publicar al cluster como un servicio de ese nodo.

Se seleccionó la segunda alternativa, ya que con la configuración de la red y las características de los equipos de los que se dispone, sería imposible instalar el software en cada uno de ellos, debido a que no cumplen con los requisitos mínimos para la instalación del software Grid. Otra razón de peso es el número de clusters dedicados ya instalados; si se desea incorporar estas instalaciones a Grid se deberían realizar grandes modificaciones a la infraestructura limitando el número de clusters disponibles en los sistemas Grid.

Otro tipo de solución en ambiente multicluster, que se puede ver en la Figura 8 sobre el sector derecho, es la de tener nodos especiales dentro del cluster con distintas jerarquías. Se evalúa la comunicación en los clusters y entre ellos, y según estos datos el master envía las tareas a ejecutar a un sub master a través de un nodo de comunicación (Comunication Manager) con librerías modificadas de MPI. Como se puede observar existen dos niveles de workers: los que están en un primer nivel y son coordinados por el master principal, y los que están en un segundo nivel, coordinados por el sub master. Tomar esta solución para entornos Grid nos lleva a la primera opción mencionada, donde todos los nodos son parte del entorno Grid.

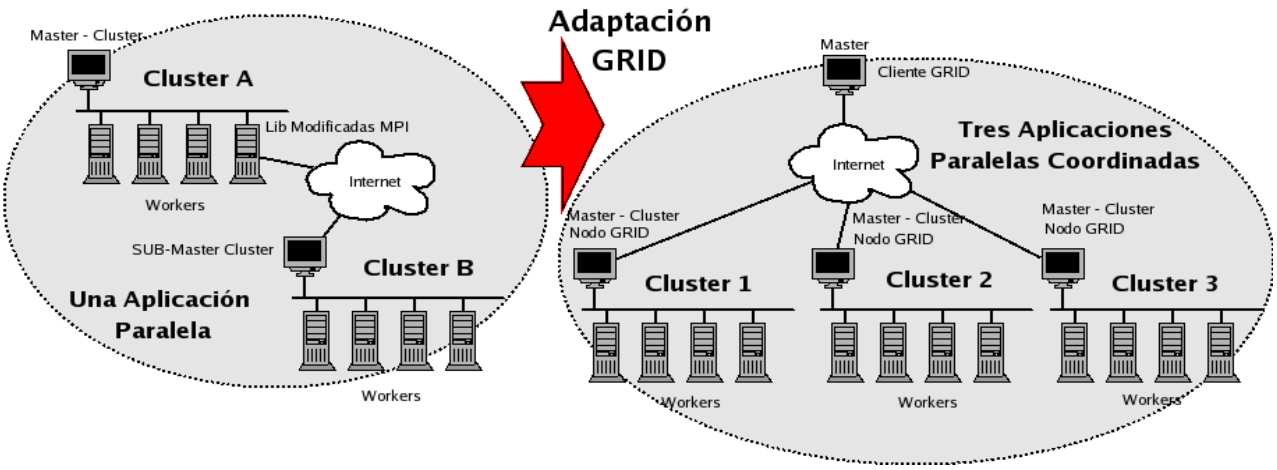


Figura 8. Adaptación Entorno MultiCluster a Entorno Grid

Esto motivó que se decidiera explorar nuevas alternativas. En Grid la primera consideración fue acerca del modelo de programación. El modelo utilizado para desarrollo es el llamado SOA (Service Oriented Architecture, o arquitectura orientada a servicios). En SOA, generalmente los servicios se publican en un directorio, el cliente los descubre, elige a cuál invocar según algún criterio y accede directamente al nodo que los provee. Bajo esta metodología, se decidió que los servicios publicados fueran los prestados por los nodos Grid, siendo el servicio en cuestión el poder de procesamiento de los clusters.

En esta etapa del trabajo, el servicio que presta el cluster es una aplicación que simula una malla de sensores. Se le envía un valor aleatorio al servicio, y éste genera un evento en la red de sensores devolviendo la matriz resultante. Para una red de 30 sensores, el tamaño de la matriz resultante es de aproximadamente 2MB. Con la metodología orientada a servicios, el proceso completo puede ser paralelizado en los clusters y sólo ser devuelta una matriz de 7KB, para luego vincular las matrices resultado de los distintos clusters, y de esta manera optimizar el tiempo de *turn-around*. Esta optimización no fue adoptada ya que en estas condiciones los datos transferidos serían mínimos y no tendría sentido realizar la medición del ancho de banda.

Cada uno de estos nodos se ejecuta en forma independiente. Cada nodo Grid, al inicializarse, se registra en un servicio de monitoreo y descubrimiento llamado MDS (Monitoring and Discovery System) provisto por el software Globus y quedando en espera para que sea solicitada una ejecución remota. Cuando uno de los modulos cliente invoca

una simulación, el nodo Master consulta una estructura de directorio en donde los nodos Grid están registrados, y según la información de ancho de banda y carga de los clusters realiza la asignación de tareas. Esta asignación no es más que la ejecución de la aplicación en los distintos clusters, a través de procesos concurrentes en el nodo master.

Una vez obtenidos estos resultados el master realiza la ponderación final y la deja a disposición del cliente. Todos estos eventos están reflejados en el diagrama de secuencias en la Figura 9.

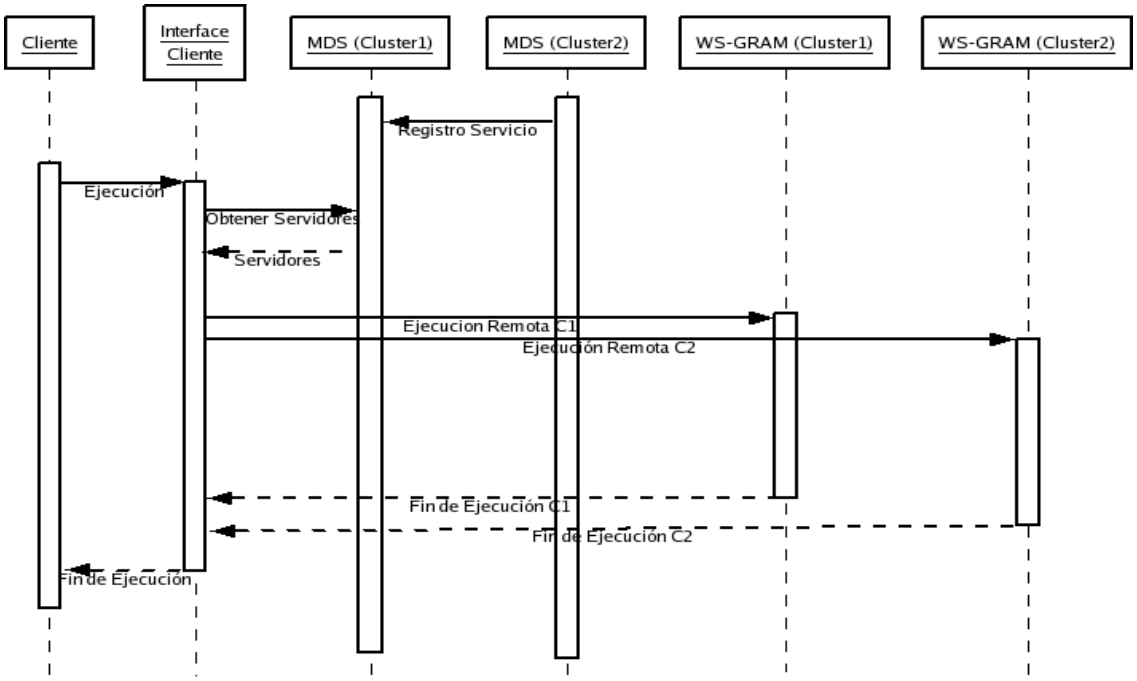


Figura 9. Diagrama de secuencia de ejecución remota

3.2.1.1. Predicción Intra Clusters

Para lograr la caracterización de la performance de cada uno de los clusters se tomó como guía el trabajo realizado en [29], el cual es una evolución del trabajo hecho manualmente en la experimentación de la aplicación gráfica. En este caso según el patrón de comunicaciones entre los clusters y su poder de cómputo, se puede realizar una estimación de la cantidad de workers necesarios para una ejecución eficiente.

Las formulas descriptas en [29] son:

- La performance disponible (*AvPerf*) es la sumatoria de la performance de cada worker ejecutando en forma aislada.
- La performance estimada (*EstPerf*) es el mínimo valor entre la performance disponible y la máxima performance de la red local.
- La eficiencia estimada (*EstEficiency*) para cada cluster es el cociente entre la performace estimada y la performance disponible.
- El speedup estimado (*EstSpeedup*) es el cociente entre la performance estimada y la performance estimada del cluster local.

Para un nodo de cómputo ejecutándose en un procesador el tiempo de computo (T_{Cpt}) se define como la relación entre el número de operaciones de la aplicación (Oper) y el

rendimiento del procesador (Perf): T_{Cpt}=Oper/Perf. El tiempo de comunicación (T_{Comm}) es la relación entre la cantidad de datos transferidos entre el nodo de cómputo y la máquina servidora, en un esquema *master-worker,* (Comm) y la capacidad de procesamiento de la red (T_{Put}): T_{Comm}=N*Comm/T_{Put}. El rendimiento máximo es el rendimiento que puede ser obtenido cuando T_{Cpt} >= T_{Comm} (E. 1). En un entorno de múltiples clusters así como en Grid hay dos niveles de comunicación: intra-cluster e inter-clusters. Para calcular el rendimiento máximo intra-cluster (MaxPerfintra) es necesario considerar el rendimiento de la red de área local y para calcular el rendimiento máximo de un inter-cluster (MaxPerfinter) el rendimiento promedio de Internet entre los dos clusters El rendimiento disponible (AvPerf) es la suma entre los nodos de computo ejecutando por separado la tarea de la aplicación. El rendimiento estimado (EstPerf) que puede proveer un cluster al entrono Grid es el valor mínimo entre AvPerf, MaxPerfLAN and MaxPerfInet (E.2). La eficiencia estimada para cada cluster es la relación entre EstPerf and AvPerf mientras la aceleración estimada (Speedup) es la relación entre EstPerf del cluster y EstPerf del cluster (EstPerfLC) (E. 3).

T_{Cpt} >=T_{Comm} → MaxPerf = Perf <= Oper T_{Put} /Comm *(E. 1)*

EstPerf = min(AvPerf , MaxPerf LAN , MaxPerf Inet) *(E. 2)*

EstimatedEfficiency = EstPerf / AvPerf; EstimatedSpeedup = EstPerf/ EstPerf LC *(E. 3)*

Código 1. Formulas de estimación de eficiencia inter-clusters

En la Tabla 4 se pueden observar los valores medidos en cada worker y los del cluster en su conjunto. Como resultado de aplicar las fórmulas de performance al conjunto Grid UNC-CDF se obtiene la Tabla 5 con los datos de cada uno de los clusters.

	Cluster	Tiempo Procesamiento por tarea	Tarea por Segundo
G3	Onode31	1163 segs.	0.000860
	Onode32	1142 segs.	0.000876
	Onode33	1143 segs.	0.000875
	Performance Disponible		**0.00261**
	Total Tiempo incluido Master y Comunicación (10 tareas)	4068 segs.	**0.00246**
G4	Onode41	1177 segs.	0.0008496
	Onode42	1150 segs.	0.0008696
	Onode43	1141 segs.	0.0008764
	Performance Disponible		**0.00259561**
	Tiempo incluido Master y Comunicación (10 tareas)	4480 segs.	**0.00223214**
CDF	Cluster219	50.16 segs.	0.01994

Cluster	Tiempo Procesamiento por tarea	Tarea por Segundo
Cluster220	50.66 segs.	0.01974
Cluster221	50.15 segs.	0.01994
Cluster222	50.31 segs.	0.01988
Performance Disponible		**0.07949**
Total Tiempo incluido Master y Comunicación (10 tareas)	210 segs.	**0.04762**

Tabla 4. Valores sobre los que se realizaron los cálculos finales

3.2.1.2. Predicción Inter Clusters

Para la predicción de la utilización del ancho de banda por Internet, en primer término se intentó usar las librerías implementadas en la UAB. Por lo complejo del código y la falta de documentación resultó difícil implementarlas como solución para la prueba inter cluster, por lo que se optó por realizar una búsqueda encontrándose gran cantidad de herramientas disponibles en Internet. Resultados de esta prueba pueden ser observados en la Tabla 5.

Cluster	Nro.Comp	LAN TPUT (Mbytes/sec)	AvPerf (10-3 tareas por sec)	EstPerf	Estimate Efficiency	Experimental Speedup	Precision
G3	3	10	2.2	1	100.00%	0.86	86.00%
G4	3	10	2.4	1	100.00%	0.9417	94.00%
GridCDF	4	10	79	1	100.00%	0.98	98.00%

Tabla 5. Tiempos de ejecución de funcionalidad de servicio

La herramienta seleccionada, por su sencillez de instalación y su funcionalidad, fue Netperf [31]. Esta funcionó correctamente mientras las direcciones IP fueran públicas, pero una vez que se implementó una red privada virtual entre la Universidad del Comahue y la empresa CDF, la medición del ancho de banda consumido resultó constantemente igual al máximo disponible, por más que la red estuviera congestionada. Luego de intensas pruebas se llegó a la conclusión de que al tener la VPN funcionando sobre interfaces virtuales, la medición de utilización que este software realizaba era la de la conexión al buffer del dispositivo virtual de la VPN, y no la real de las interfaces físicas de la red. En segundo lugar, y teniendo este problema en mente, se usó la herramienta Iperf [25], diseñada para medir consumos de ancho de banda con protocolos TCP y UDP. En este caso se disponía de una opción *prefill-buffer*, que completaba el buffer del dispositivo antes de comenzar la transmisión de la información, y lo que se mide es el flujo de datos a partir del momento en que el buffer se completa. Como resultado de estas mediciones se obtuvo el gráfico de la Figura 10. En el mismo se pueden observar sobre el eje de las abscisas las fechas en que fueron realizadas las mediciones, hechas cada diez minutos por un lapso de 30 segs.

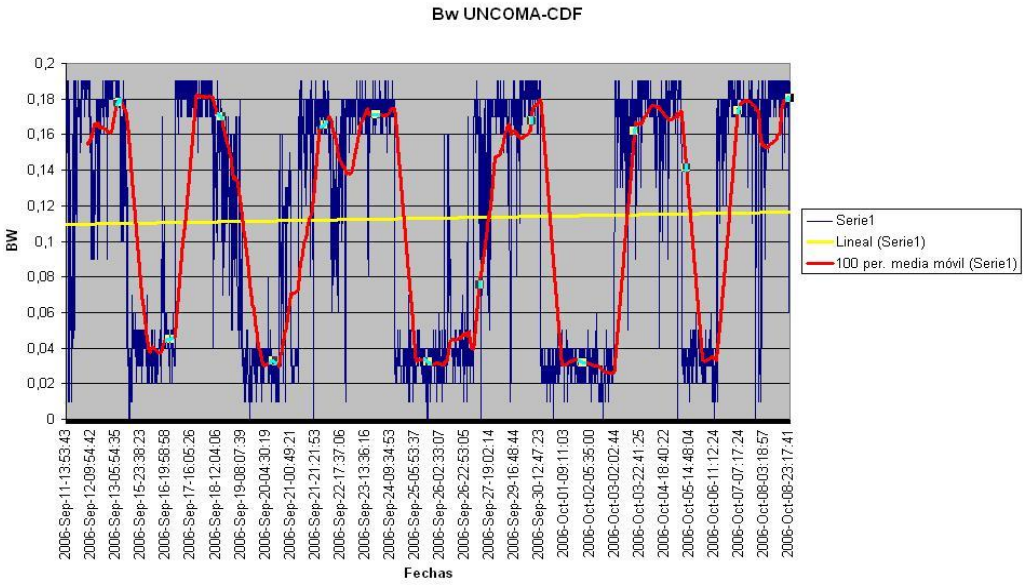


Figura 10. Mediciones de ancho de banda Comahue - CDF

La conexión a Internet en la UNC es simétrica de 256Kbps y la conexión de la empresa es asimétrica con 512Kbps de descarga y 190Kbps de subida a Internet. Como el nodo coordinador se encuentra en la Universidad del Comahue y uno de los clusters está en la empresa CDF, las mediciones de tráfico se realizaron sobre el canal saliente desde la empresa hacia la UNC, para medir el vínculo más débil.

Se realizó un estudio para observar si era conveniente determinar la cantidad de nodos de cómputo en los clusters dependiendo de si la conexión a Internet se encontraba en un valle o un pico (Figuras 11 y 12). Con las cotas impuestas en los valores que se ven en los dos siguientes gráficos se cubre un 80% de los valores totales medidos.

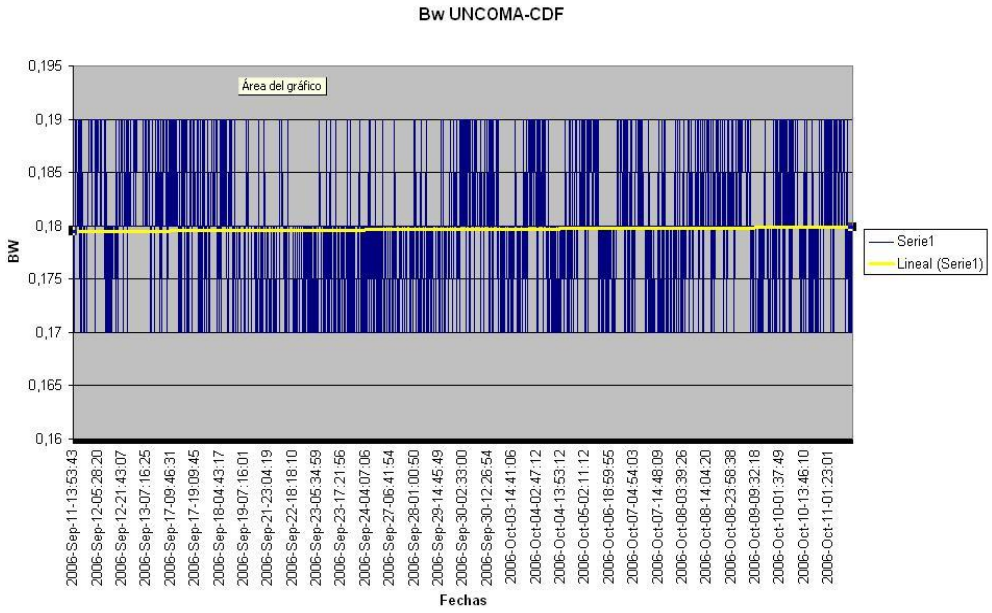


Figura 11. Anchos de Banda limitando solo los valores de los picos. (170 Kbits – 190Kbits)

En la Figura 10 se pueden observar tres tipos de líneas. La primera, desplazándose sobre el eje y, representa los valores obtenidos en esas fechas. La siguiente sobre esta es una media obtenida cada 100 valores. Con este análisis se intenta estimar la frecuencia de picos y valles sobre los valores obtenidos. Por último, una línea horizontal corresponde a una regresión lineal para realizar un ajuste de los valores. Sobre este último análisis se obtiene un valor cercano a los 0,11 Mbps de promedio para el uso de la conexión entre los laboratorios.

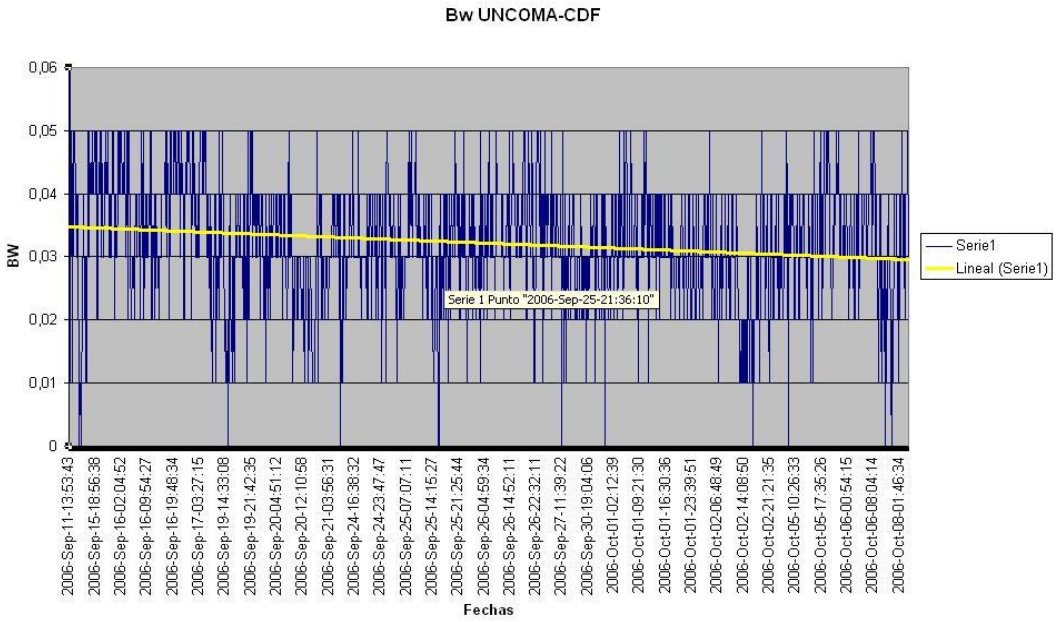


Figura 12. Anchos de Banda limitando solo los valores de los valles. (0 Kbits – 0.5Kbits)

Sobre estos datos y aplicando las fórmulas de [29] se obtiene la Tabla 6. En la columna Internet Tput, línea correspondiente al cluster GridCDF, se puede observar el cambio de valor con respecto a la ejecución por Internet. En la fila de G4 el valor no cambia, ya que este cluster se encuentra en una Intranet de la facultad y el impacto de la conexión de red es mínimo.

Cluster	Nro. Comp	Internet TPUT (Mbits/sec)	AvPerf (10-3 tareas por sec)	EstPerf	Estimate Efficiency	Experimental Speedup	Experimental Efficiency	Precision
G3	3		2.2	1	100.00%	0.86	86%	86.00%
G4	3	93.7	2.4	1	100.00%	0.7661	77%	76.00%
GridCDF	4	0.11	79	0.5949	59%	0.5990	***40%***	99.00%
Total	**10**		**83.6**	**2.59**	**86%**	**2.23**	**68%**	**86%**

Tabla 6. Mediciones Intranet-Internet

Los datos experimentales de performance obtenidos sobre los clusters GridCDF y G4 incluyen la sobrecarga del middleware Grid. Desde una máquina coordinadora se ejecuta en forma remota el cálculo sobre el cluster, y cuando éste termina se recupera la matriz resultante. La descripción en formato XML de la tarea ejecutada se observa en el Código 2.

```
<job>
  <executable>/usr/RedOsciladores/redPar</executable>
  <directory>${GLOBUS_USER_HOME}</directory>
  <stdout>${GLOBUS_USER_HOME}/stdoutRP2</stdout>
  <stderr>${GLOBUS_USER_HOME}/stderrRP2</stderr>
  <count>5</count>
  <jobType>mpi</jobType>
  <fileStageOut>
    <transfer>
      <sourceUrl>file:///home/mlbeg3/red.dat</sourceUrl>
      <destinationUrl>gsiftp://g4.uncoma.edu.ar/tmp/red2.dat</destinationUrl>
    </transfer>
  </fileStageOut>
</job>
```

Tag executable. Archivo ejecutable compilado con librerías MPI
Tag directory. Directorio de trabajo
Tag std. Archivo de salida stdout y stderr
Tag count. Cantidad de procesos a ejecutar sobre el cluster.
Tag jobType. Especificación para la ejecución con librerias MPI
Tag fileStageOut. Tareas de transferencia de archivos, una vez terminada la ejecución del procesamiento.

```
globusrun-ws -submit -s -F gridcdf.uncoma.edu.ar -f redparSCop_CDF.xml
```

Código 2. Comando de ejecución sobre el nodo Grid GridCDF del archivo XML.

3.2.1.3. Ajuste de la Aplicación

Como se puede observar en la Tabla 6, la eficiencia lograda en el cluster GridCDF es significativamente baja. Esto se debe a la diferencia de poder de cálculo que existe entre las máquinas de los distintos clusters y el ancho de banda disponible en promedio en la conexión a Internet.

Para llevar el uso de los recursos por sobre un umbral aceptable, se realizaron los cálculos reduciendo la cantidad de nodos de computo disponibles en el cluster GridCDF. Esto queda reflejado en la Tabla 7, donde la cantidad de tareas en el cluster GridCDF se reduce a 39.

Con esta reducción en la capacidad de trabajo y sobre los cálculos en relación al ancho de banda de internet en 110Kbits promedio, se realizaron nuevamente las mediciones, arrojando un incremento en la eficiencia. Esto se debió en parte a que la conexión de Internet en ese momento se encontraba en un pico de disponibilidad de ancho de banda y el promedio de los picos se encuentra en los 180Kbps por segundo.

Cluster	Nro. Comp	Internet TPUT (Mbits/sec)	AvPerf (10-3 tareas por sec)	EstPerf	Estimate Efficiency	Experimental Speedup	Experimental Efficiency	Precision
G3	3		2.2	1	100.00%	0.86	86%	86.00%
G4	3	93.7	2.4	1	100.00%	0.9417	94%	94.00%
GridCDF	2	0.11	***39***	0.4936	82%	0.4855	97%	98.00%
Total	**8**		**83.6**	**2.49**	**94%**	**2.29**	**92%**	**93%**

Tabla 6. Mediciones Intranet-Internet con reducción de workers

Como prueba final se realizó la ejecución de una simulación paralela combinando el poder de cálculo de los tres clusters. Para esta prueba se incrementó el número de tareas a 100, con el fin de observar la escalabilidad de la solución. Sobre el total de tareas se realizó una división proporcional, donde los clusters de la UNC ejecutaron cada uno 3 tareas, y en los dos nodos de cómputo de la empresa CDF se ejecutaron 94 tareas. Esto arrojó un resultado aproximado de 20' en la ejecución de cada cluster.

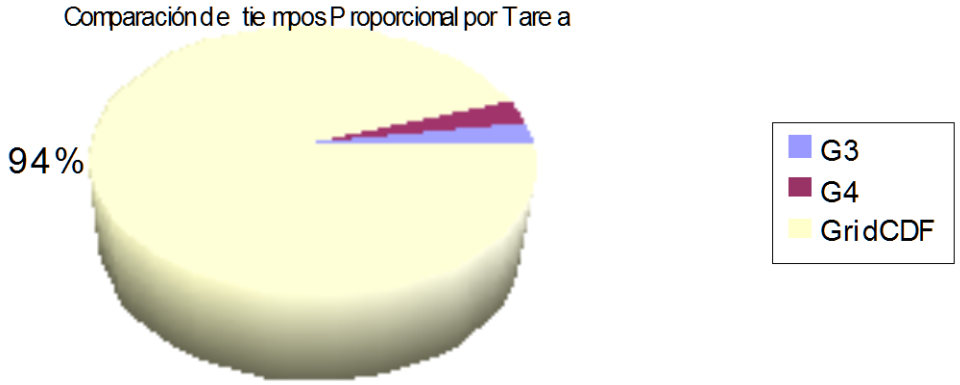


Figura 13. Proporcional en la ejecución de tareas

4. Resumen

Según las experiencias realizadas la incorporación del middleware de Grid no altera en absoluto el desarrollo de aplicaciones paralelas de tipo master-worker, por ejemplo las analizadas en este capítulo. Dados tiempos de ejecución suficientemente prolongados y según lo expresado en la sección 2.1, la sobrecarga de unos segundos no influye de manera determinante sobre el tiempo total de la ejecución.

Por otro lado, la incorporación de middleware Grid sí altera el diseño de la aplicación paralela. En este trabajo se debió modificar la lógica de trabajo de multicluster según lo expresado en [21, 22, 23, 24] por la de un modelo orientado a servicios, la cual es más acorde al uso eficiente del middleware Grid. Este modelo es un caso especial de [21] donde cada nodo de cómputo del cluster de primer nivel es un nodo *Communication Manager*.

Otra de las conclusiones es la observación de la necesidad de contar con la implementación de clusters a demanda para no tener que pasar por el tipo de modificaciones de diseño observadas en este trabajo. La necesidad surge de poder

vincular los distintos clusters que se encuentran detrás de nodos Grid en una red privada virtual para que puedan trabajar en conjunto bajo el esquema de seguridad de la organización virtual. Esta asignación debería poder realizarse en forma automática en base a modelos matemáticos con los parámetros de red capacidad de cómputo y características de la aplicación para poder realizar una distribución de equipos de manera eficiente.

Capítulo 3. Meta-planificación en entornos Virtuales

La integración de recursos heterogéneos en distintos dominios administrativos, convierte tareas de control rutinario en gestión altamente compleja. Esto aumenta aún más si los recursos involucrados en la operación de diferentes organizaciones se encuentran coordinados. El objetivo de este capítulo es proponer una arquitectura basada en políticas de gestión para simplificar labores en un entorno de ejecución de altas prestaciones sobre espacios virtuales. Las políticas de gestión, son capaces de reflejar objetivos de negocio con alto nivel de abstracción, y pueden ser transformadas automáticamente en reglas entendibles por sistemas de cómputo, siendo estas distribuidas en los componentes de gestión correspondientes a los distintos nodos de la organización virtual. Para lograr este objetivo se define un marco de trabajo que incorpora la capacidad de crear y gestionar entornos virtuales en base a requerimientos de ejecución. Este marco de trabajo puede ser extendido para implementar distintos modelos de planificación e incorporar diferentes modelos de clusters. Por último se presenta un ejemplo de gestor de tareas y se presentan componentes para implementar la arquitectura propuesta.

1. Gestión Basada en Políticas

En la mayoría de las organizaciones, las redes de cómputo y los recursos de información, son puntos críticos para lograr una óptima administración en la infraestructura. Con su evolución, estos recursos poseen funcionalidad cada vez más diversa; por ejemplo, tipos de servicio, protocolos de acceso, heterogeneidad de software y plataformas, llevando a aumentar la complejidad en su configuración y gestión. Para mantener la infraestructura, se debe contar con personal de diferentes perfiles y alto nivel de capacitación. Esto se ve claramente en los entornos Grid: el tener un entorno que combina diferentes dominios administrativos, lleva a que la heterogeneidad de recursos crezca, y la complejidad se

multiplique si los mismos deben cooperar para lograr un fin común.

En algún punto, es necesario que los sistemas sean más fáciles de administrar, simplificando tediosas tareas administrativas. La situación ideal es que no necesiten dichas tareas, como asignación de sistemas de archivos, coordinación o planificación de recursos de cómputo. De todas maneras la disciplina de gestión de sistemas no desaparecerá, pero si reducirá su complejidad con nuevas áreas aún en desarrollo. Las áreas de investigación más representativas sobre aspectos de administración automáticos son los sistemas autonómicos o sistemas de gestión basados en políticas.

El paradigma de los sistemas autonómicos [61] modela el sistema nervioso, el cual debe tener un mecanismo donde la adaptación a los cambios es la variable esencial. Estos cambios disparan acciones en el comportamiento del sistema de cómputo de manera tal, que el sistema debe ser llevado a un estado “normal” con respecto al entorno. El estado de equilibrio es una condición necesaria para la supervivencia del organismo. En el caso de un sistema de computo autonómico, se puede ver a la supervivencia como la habilidad del sistema para recobrarse ante fallas, reconfigurarse a medida que se requiera y mantener el nivel de rendimiento de operaciones siempre cercano a un nivel óptimo. El ambiente externo (como un ataque) o interno (excesivo uso de CPU por ejemplo), tienen impacto en el equilibrio del sistema. El sistema autonómico requiere sensores que monitoreen los cambios en el entorno interno y externo, y estos reaccionan ante los cambios para mantener el equilibrio. Los cambios son censados y deben ser analizados para determinar si alguna variable ha salido de los límites establecidos. Si un umbral ha sido superado, se procede a la ejecución de algún conjunto de tareas planeadas con anterioridad para que el sistema vuelva a un equilibrio dentro del nuevo entorno. Este esquema requiere inteligencia para seleccionar las tareas correctas de un gran conjunto de posibles comportamientos. Finalmente, el motor de neuronas ejecuta el cambio requerido.

Otro paradigma con las mismas bases que los sistemas autonómicos es el de gestión de cómputo basado en políticas. Una política es un conjunto de reglas o instrucciones que determinan la ejecución de operaciones, y el ejemplo más común es el de administración de redes basadas en políticas. Las políticas expresan la vista de gestión de cómo la red puede ser usada por aplicaciones o usuarios desde el punto de vista de sus servicios y puede aplicarse a un área de la red local, así como también múltiples dominios. Este es un cambio en la manera que las redes son configuradas y los recursos son asignados: en lugar de hacer hincapié en los dispositivos, un sistema basado en políticas hace foco en los usuarios y aplicaciones. Esto se implementa ocultando al administrador de la red el mapeo de los usuarios a los dispositivos; en cambio, se asigna este mapeo a entidades de la red que generen dinámicamente esta dependencia basadas, por ejemplo, en la relación de usuarios y el tráfico de red que generan.

Las redes basadas en políticas, complementan y extienden los métodos actuales de gestión. Permiten a los operadores de red gestionar los recursos basados en las necesidades del negocio y asegurando el rendimiento predecible para aplicaciones de misión critica. También, simplifican las operaciones de la red, desde la configuración de dispositivos para la provisión de nuevos servicios haciendo la red más productiva, hasta el control centralizado de los recursos, haciéndola más segura y orientada al tráfico. Permiten a los operadores de la red expresar los objetivos de negocio como un conjunto

de reglas o políticas, que son aplicadas a lo largo de la red. Las redes basadas en políticas permiten que estas reglas sean definidas de manera centralizada pero implementadas de manera distribuida. Este tipo de arquitectura hace posible la implementación de las reglas en dominios como grupos o áreas geográficas determinadas. Facilita y automatiza tareas que deben ser operadas manualmente como la configuración de dispositivos de red para priorizar aplicaciones específicas.

Como resultado de la automatización estos sistemas permiten implementar funciones de calidad de servicio (QoS) que requieren complejas tareas de configuración. Haciendo uso de la integración con servicios de consulta disponibles en la intranet o internet, el sistema puede correlacionar información de los usuarios, aplicaciones y características de red para asegurar que las reglas son aplicadas en forma coherente.

La importancia de este concepto dentro de la tesis es el hecho que se puede extender a organizaciones de entornos Grid, donde un aspecto clave para el correcto funcionamiento del sistema es la interconexión de recursos informáticos y la gestión coordinada de los mismos. La aplicación de estas reglas tiene un especial impacto en la gestión de los planificadores y sus colas de trabajo. Las políticas podrán clasificar roles de usuario, servicios, tiempo de servicio, clases de recursos, etc. Estas políticas son interpretadas y almacenadas automáticamente en los distintos niveles de las organizaciones virtuales y físicas, para ser aplicadas en diferentes puntos críticos de la red siendo además suficientemente flexible para incluir futuros requerimientos de calidad de servicio y nuevos dispositivos.

El nivel de mayor impacto para la gestión de recursos en los sistemas Grid se encuentra en la capa de planificación. Esta coordina recursos e intercambia información sin importar su ubicación física. Cuando alguna aplicación utiliza el sistema Grid y realiza requerimientos de hardware para su ejecución, esta capa es la encargada de buscar, seleccionar e instanciar los recursos apropiados en donde realizar su despliegue. En este capítulo se realiza un mapeo y extensión de la arquitectura descrita en la RFC 2753 [35]. Esta RFC detalla la especificación de un marco de trabajo para la provisión de control basado en políticas sobre las decisiones de control de admisión.

El componente que interviene de forma activa en este proceso es el planificador de Grid o metaplanificador. Este permite acceder a los recursos distribuidos en distintas organizaciones físicas y comunicarse con los distintos administradores de recursos locales. Los administradores de recursos se convierten de esta manera en los proveedores de servicio y el metaplanificador los coordina y utiliza, en base a criterios preestablecidos. Otro aspecto a tener en cuenta es la gestión de los recursos de cada organización física. Aún persisten problemas cuando distintos requerimientos compiten por los mismos recursos dentro de la organización virtual. Poder garantizar un correcto control de la utilización de recursos y su disponibilidad en un entorno adecuado para las aplicaciones es una tarea difícil y generalmente no llevarlo adelante en forma automática conlleva a un uso incorrecto o bajo aprovechamiento.

Una de las alternativas para maximizar la utilización de recursos, es la implementación de máquinas virtuales. Las máquinas virtuales ofrecen la posibilidad de instanciar entornos de trabajo pre-configurados e independientes; tienen la capacidad de administrar y limitar el uso de procesadores, memoria y disco, además de la capacidad de migrar a otra

máquina física el entorno completo si fuera necesario. Las implementaciones actuales de máquinas virtuales proveen una performance similar a la obtenida por los sistemas físicos.

El objetivo de este capítulo es definir un marco de trabajo y describir una implementación para simplificar la gestión de entornos virtuales evolucionando conceptos de gestión de redes basadas en políticas [37, 38, 39, 40, 41] y sistemas de gestión autónomos [36]. Algunos de los temas abordados por estos trabajos son aspectos de calidad en servicios, control de admisión, control de congestión, balance de carga, etc. Cada uno de estos aspectos puede ser combinado a través de políticas de alto nivel, reglas de negocio en un esquema centralizado y simple.

La contribución se centra en la extensión e implementación de un metaplanificador y un administrador de recursos. En este caso los recursos son clusters de máquinas virtuales. El metaplanificador según el requerimiento de una aplicación y a través de heurísticas de manera automática selecciona equipos físicos del conjunto de servidores disponibles en el Grid y los conecta a una red virtual. El administrador de recursos local en una organización física monitorea y adapta en forma dinámica la carga de trabajo sobre los equipos físicos para optimizar el uso de los mismos. Finalmente se describirá la arquitectura en la que está incluido el metaplanificador y se detallan las conclusiones del capítulo.

2. Elementos Arquitecturales

Las políticas representan los objetivos de negocio de la organización; lo que interesa en esta tesis son los objetivos de la organización virtual. Estas políticas deben tener algún mecanismo para la toma de decisión incorporando las organizaciones locales, dueñas de los recursos físicos, siendo éstos los nodos donde se ejecutarán finalmente las tareas.

Se debe realizar una traducción entre los objetivos de negocio y su implementación en el sistema. Los acuerdos de calidad de servicio o Service Level Agreement (SLA), y sus objetivos o métricas (Service Level Objectives, o SLOs), son usados para especificar servicios que la infraestructura proveerá a un cliente determinado. Los SLAs generalmente se expresan en terminología de negocios de alto nivel y los SLOs son métricas más especificas que dan el soporte a las SLAs. Estos dos niveles de descripción de servicios de red deben ser traducidos a un nivel de implementación lo más genéricos posible para poder ser implementados en los distintos dispositivos.

Los principales elementos en la arquitectura [47] son el PEP (Policy Enforcement Point), que define la entidad donde se aplican las políticas, y el PDP, que es la entidad donde se decide qué políticas aplicar. La Figura 14 muestra una configuración simple de estos dos elementos, en este ejemplo el PEP es un componente en una red de datos y el PDP es una entidad que pudiera existir en un servidor de políticas.

El PEP representa un componente que se ejecuta en nodos donde las políticas pueden ser aplicadas. La toma de decisión se realiza principalmente en el PDP. El PDP hace uso de mecanismos adicionales y protocolos logrando funcionalidades extras como autenticación de usuarios, almacenamientos de políticas y estadísticas. La interacción comienza con el PEP, que recibe una notificación o un mensaje que requiere una decisión

sobre las políticas establecidas. Dado este evento, el PEP formula el requerimiento (que debe contener los elementos necesarios) para una decisión al PDP. El PDP retorna una decisión descrita como reglas y el PEP luego las ejecuta. El PDP generalmente adjunta información adicional, relacionada con aspectos de seguridad. El PDP puede ocasionalmente contactar servidores externos, por ejemplo para compartir configuraciones, autenticación de usuarios o base de datos estadísticas; por este motivo los protocolos deben ser estándares.

La especificación de políticas implementadas en el PDP es de particular importancia. Debido al número de campos necesarios para lograr una decisión, el número de políticas aplicables puede ser potencialmente importante y complejo. Deben ser contemplados mecanismos para gestionar la cantidad y calidad de políticas que pudieran ser aplicables al mismo requerimiento, ya que existe la posibilidad de un conflicto entre las mismas.

Con el objetivo de gestionar la complejidad del punto de toma de decisión y para asegurar su coherencia y una ejecución consistente, el lenguaje debe asegurar un mapeo no ambiguo del requerimiento a la política en acción. También debe permitir la especificación de secuencia o prioridad en que las diferentes reglas pueden ser aplicadas. En algunos casos una configuración con un solo PDP no alcanza a ser suficiente, por ejemplo, para aplicar control de políticas locales además de las políticas emanadas del PDP. Una opción es que el PEP disponga de un PDP local para optimizar los tiempos de comunicación y minimizar el dominio de políticas.

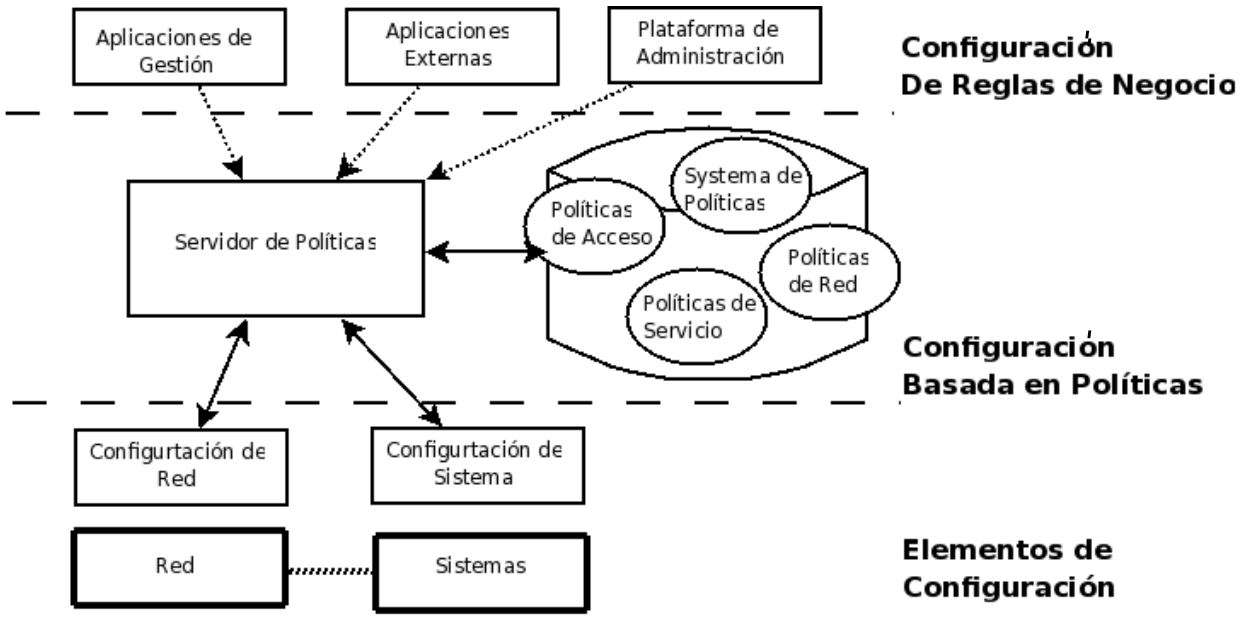


Figura 14. Jerarquía de Control en sistemas basados en políticas.

El servidor de políticas centralizado puede ser responsable de la decisión de políticas en representación de los componentes del entorno del dominio administrativo. Por otro lado las políticas que dependen de información y condiciones locales de una entidad particular, más dinámicas, pueden ser mejor implementadas por un servidor más cercano a la organización local (LPDP). Si estuviera disponible, el PEP podría usar primero el LPDP

para tomar una decisión local. Esta toma de decisión parcial y el requerimiento de política original se envían luego al PDP que toma la decisión final, pudiendo superponerse a la decisión del LPDP. El PDP es la autoridad final y el PEP debe ser quien siempre lleve a cabo la tarea indicada. Una vez que se ha establecido una relación entre el PDP y el PEP, este último es quien tiene la responsabilidad de notificar si se ha llevado a cabo la tarea y si el requerimiento ya no está vigente.

Siguiendo el esquema de la Figura 14, bajo el modelo de control descripto, el módulo de PEP en el nodo debe seguir los siguientes pasos para la toma de decisión:

1. Cuando un evento local o mensaje invoca al PEP para una decisión de política, el PEP crea un requerimiento que incluye la información de mensaje o estado local que describe el requerimiento de control de admisión. Además, el requerimiento debe contener los elementos para la toma de decisión previamente especificados.
2. El PEP puede consultar una base de datos de configuración local para identificar un conjunto de elementos de políticas (*conjunto A*) que pueden ser evaluados localmente. La configuración local especifica el tipo de elementos que pueden ser evaluados localmente. El PEP envía el requerimiento con el *conjunto A* al LDPD y obtiene los resultados, llamados *resultados parciales* pudiéndolos expresar como *D(A)*.
3. El PEP envía el requerimiento con todos los elementos y *D(A)* al PDP. El PDP aplica las políticas basadas en todos los elementos de las políticas y el requerimiento obtiene una decisión que podemos expresar como *D(Q)*. Luego combina el resultado con los resultados parciales *D(A)* y logra una decisión final. En el caso de que la función *D(Q)* tuviera un resultado de falla por no poder determinar el resultado sobre el conjunto de elementos del requerimiento este resultado será enviado el PEP.
4. El PDP retorna la decisión final, obtenida en el paso 3 al PEP. También se contempla el envío de los resultados de notificación en forma asincrónica al PEP para cambiar una decisión previa o generar un nuevo mensaje de política de error o advertencia.

3. Adaptación al entorno Grid

La arquitectura básica propuesta en la RFC 2753 establece que la interacción entre los componentes comienza en el PEP. El PEP recibe una notificación o un mensaje que requiere una toma de decisión. Dado este evento, el PEP formula el requerimiento para una decisión y se lo envía al PDP. Este esquema se basa en la suposición de que en la mayoría de los casos se encuentran dentro de un escenario en donde el dominio administrativo posee un número limitado de recursos. Si se replica este esquema en distintos dominios y se combina la solución bajo un esquema común como proponen los sistemas Grid, la solución tiene dificultades para poder tomar decisiones de manera centralizada para la gran cantidad de recursos disponibles en este tipo de sistemas.

Las organizaciones virtuales agregan otro nivel de abstracción en la toma de decisiones: se tiene un nivel local de toma de decisiones (la organización que posee los recursos físicos) y un meta-nivel que es la organización virtual que se genera dinámicamente. Estos dos niveles se superponen y pueden llegar a generar conflictos en las políticas que cada una aplica, por lo que es un desafío importante lograr que estos dos niveles trabajen de manera eficiente. Otro de los elementos arquitecturales que describe la RFC

es la extensión del PDP con un representante local llamado LPDP. El documento presenta al LPDP cercano a los dispositivos de red; esto significa que se encuentra próximo al PEP. Este usará al PDP para optimizar el proceso de toma de decisión, primero se tomará una acción de manera local y se enviará un requerimiento al PDP quien tomara finalmente una decisión. Esta decisión podrá validar o corregir la decisión del LPDP. El PDP es la autoridad final que determina el comportamiento del PEP y este deberá implementar esta última decisión.

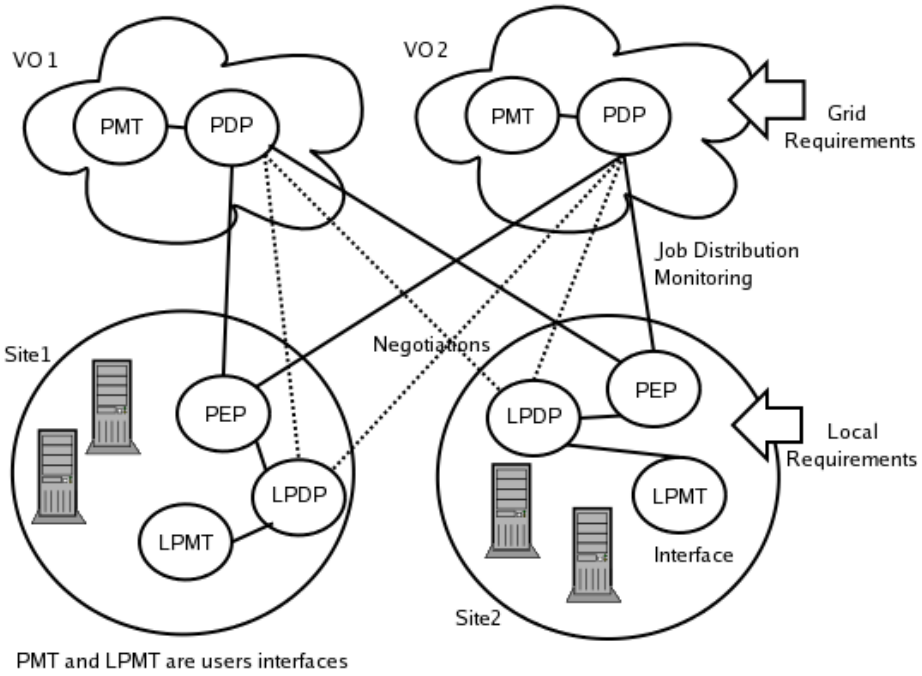


Figura 15. Adaptación RFC 2753 para arquitecturas Grid

Se puede extender este comportamiento a los entornos Grid como muestra la Figura 15. Si se asume que el LPDP es un representante de la organización local implementando sus políticas, pero al mismo tiempo partícipe de la organización virtual Grid, se deberá enviar el requerimiento al PDP general. Las políticas se definirán y almacenarán en ambos niveles. Se analizarán primero a nivel local y luego a nivel global pero este último tendrá más prioridad, si existiese algún conflicto entre las reglas. Un ejemplo de esta interacción es la autenticación de usuarios, en ambos casos los usuarios son distintos, sin embargo a nivel global la política realiza un mapeo a los usuarios locales pudiendo los primeros trabajar en usuarios habilitados en el sistema físico. En algunos casos se podrá filtrar el requerimiento al PDP global si el contrato de participación en la organización virtual así lo habilitara. Los diagramas de secuencias con requerimientos a nivel de organización virtual y organización local se pueden observar en la Figura 16.

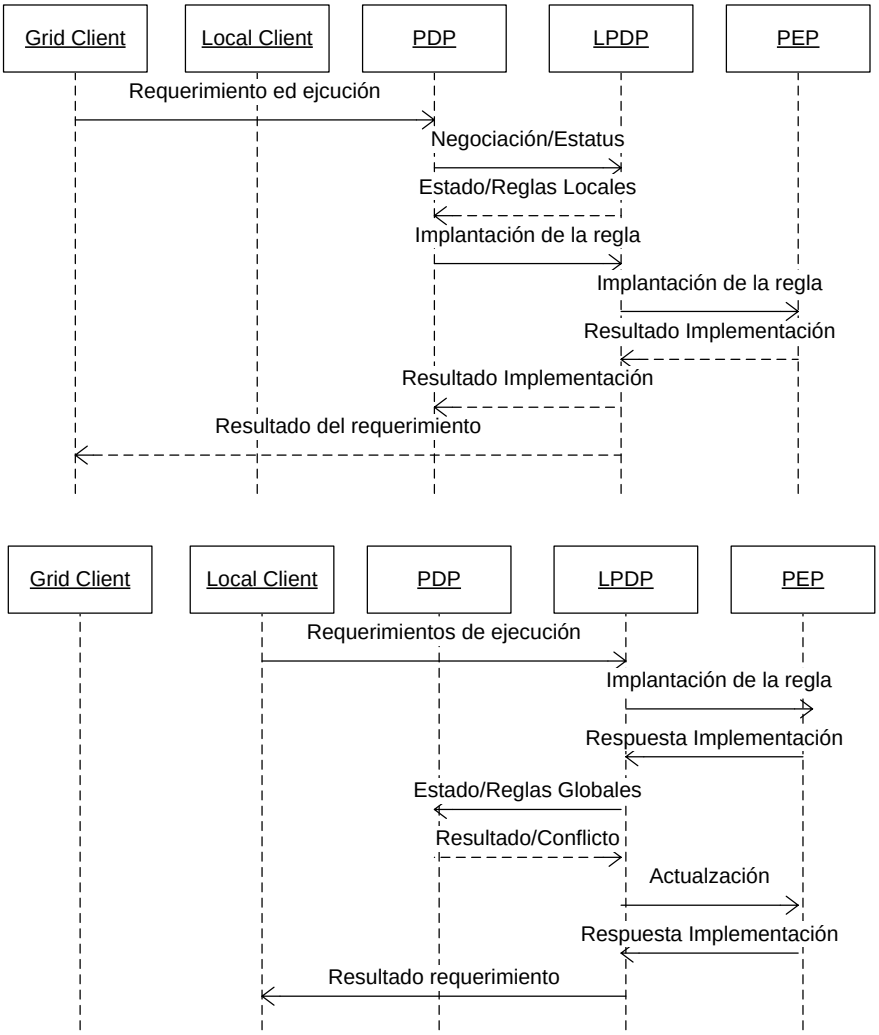


Figura 16. Diagrama de secuencias, requerimientos a nivel de (a) Organización Virtual y (b) Local

4. Implementación

La arquitectura que se propone en esta tesis tiene por objetivo la creación de redes de recursos virtuales dentro de entornos Grid [40]. Los usuarios pueden acceder a los recursos en forma interactiva a través de interfaces web. Se busca realizar una configuración en forma segura y con mínima intervención de los administradores locales en cada organización física.

Existen diferentes publicaciones relacionadas con esta problemática tomando como caso de uso entornos de altas prestaciones, cada uno enfatizando diferentes aspectos de la solución. El proyecto Virtual Workspaces [16] pone mayor importancia en la definición del espacio virtual dentro de un entorno Grid. Su caso de uso son los clusters de máquinas virtuales dentro de una red local. Cluster on Demand [62] implementa el empaquetado de un planificador de cluster para obtener subconjuntos de un cluster físico a través de la

asignación dinámica de direcciones de red. El proyecto VioCluster [63] se relaciona en gran medida con este trabajo, salvo por la diferencia de que no hace mención a Grid en la configuración de las redes virtuales, o descubrimiento dinámico de máquinas candidatas para instanciar máquinas virtuales, sino que hace hincapié en la negociación automática de dominios de administración según políticas preestablecidas relacionados con conceptos autonómicos. Por último, el proyecto In-VIGO[64] donde el nivel de abstracción es mucho mayor, permite a las aplicaciones hacer uso de entornos virtuales a través de servicios Grid.

Desde el punto de vista de diseño, la arquitectura se divide conceptualmente en tres capas. En la primera, llamada capa de acceso, los clientes ingresan al sistema. La segunda es la capa de gestión, que controla el acceso y la creación de los recursos definidos en el sistema; este nivel se puede mapear al rol del PDP. Finalmente, la capa de recursos se relaciona con la instanciación de los recursos, en este caso máquinas virtuales sobre equipos físicos; en este nivel puede ser instalado el PEP. Los clientes acceden al servicio de planificación del sistema expresando sus requerimientos a través de un lenguaje basado en XML, que en forma paramétrica determina cuántos nodos virtuales, qué imágenes y cuáles son los requerimientos de ejecución de la aplicación deseada.

Los nodos Grid cuentan con información del estado de los hosts de la planta física. El planificador, basándose en información de configuración y estado de esos servidores físicos, determina la asignación de nodos virtuales a nodos físicos. Si se requirieran más máquinas virtuales de las que un nodo Grid pudiera ofrecer, por no encontrarse disponibles, o debido a su nivel de utilización, se buscará instanciar recursos virtuales en distintos nodos Grid formando una red virtual entre ellos.

Para el caso de uso de cómputo paralelo la asignación llevada a cabo por el planificador instanciará una o más máquinas virtuales por host físico; todas estas máquinas virtuales se encontrarán conectadas en un solo espacio de direcciones, y el resultado devuelto al usuario será un punto de acceso único al conjunto de recursos de la organización.

5. Metaplanificadores para entornos virtuales

Para la implementación, en primer término se realizó el estudio de distintos metaplanificadores disponibles en el mercado eligiéndose a CSF (Community Scheduler Framework -CSF) [52, 65]. Por su característica, permite reutilizar componentes implementados y extenderlo agregando nuevas implementaciones de acuerdo con el modelo de gestión deseado. Este metaplanificador es una implementación *open-source* que consta de distintos servicios: un sistema de colas con mecanismos de planificación adaptables y extensibles, un servicio de tareas y un servicio de reserva, así como también una estructura para el flujo de información desde los nodos y un sistema de adaptación de las tareas a distintos planificadores en cada uno de los recursos físicos (en este caso, clusters de máquinas para cómputo). A continuación se analiza cada uno de estos componentes.

5.1. Servicios de Tareas

El metaplanificador se encuentra entre los Adaptadores de Gestión de Recursos, y usa una combinación del Servicio de Tareas y el Servicio de colas. Las tareas son creadas en

base a un archivo de especificación denominado RSL (Resource Specification Lenguaje). Una vez que una tarea es creada, puede ser ejecutada sobre un recurso específico o enviada al Servicio de Colas, donde se aplican las políticas de planificación.

Los Servicios de Tareas se encuentran para ayudar con las operaciones de control como detener, cancelar, reiniciar el trabajo u obtener información detallada sobre su ejecución. La interacción típica con el servicio de Tareas incluye la creación de la tarea basado en el archivo RSL, enviándolo al servicio de colas y obteniendo luego su estado.

5.2. Servicio de encolado

El servicio de encolado carga y valida la información de configuración. Se usa la información para especificar y definir las diferentes colas. Se puede definir un componente de planificación asociado a una o más colas, y dentro de estos componentes implementar diferentes algoritmos de planificación aplicando distintas políticas.

El servicio de encolado carga las configuraciones de los componentes de planificación y llama a la interfaz de programa del componente utilizando su API. Las políticas de planificación implementadas en los componentes predefinidos son *first-come, first-served* o *round robin*. También pueden implementarse otros algoritmos más complejos utilizando el servicio de Indices.

Por defecto, el servicio de encolado se encuentra desactivado y no hay colas configuradas; antes de que las tareas sean enviadas a este servicio se debe realizar su configuración. Se debe editar la configuración del meta-planificador para especificar el nombre de la cola y el componente de planificación que deba usar. Es provisto un componente por defecto y se asigna a la cola que se ha definido, se puede invocar clases de cliente en Java para crear la cola o se puede utilizar línea de comando para el mismo fin.

5.3. Componentes del Servicio de Encolado

El servicio de encolado permite definir políticas de planificación a nivel de gestión de recursos, así como también el mapeo de las tareas a los gestores de recursos, y provee el marco de planificación para la definición de distintas políticas de organización. Estas políticas están definidas en los componentes del servicio.

La distribución de CSF provee dos componentes:

- El primero es un planificador *first-come first-served o round-robin* que despacha el próximo trabajo de la cola al próximo recurso disponible. Las tareas se despachan según su tiempo de arribo y pueden ser mapeados a un gestor de recursos basado en una reserva previa o según un nombre de cluster especificado dentro del archivo de configuración de la tarea. Si no se específica ninguno de estos parámetros, el planificador selecciona automáticamente un recurso basado en la estrategia *round-robin* de la lista de recursos definida por el servicio de índices.
- El segundo componente es un acelerador, que previene que un gran número de tareas se agrupen en un gestor de recursos. Chequea los trabajos en cada cola durante cada ciclo de planificación y verifica el número de trabajos despachados a cada cluster. El servicio se asegura que los gestores de tareas no se sobrecarguen y no envíen tareas a un gestor donde puede haber una espera significativa y donde otro gestor pudiese despacharlo en menor tiempo.

Para extender el metaplanificador, se deben extender los siguientes métodos:

Interfaz	Descripción
SchedInit()	El método de inicialización es llamado una vez antes de cargar el *plug-in*
SchedOrder()	Este método ordena todas las tareas del vector de trabajos. EL planificador por defecto se basa en los tiempos de arribo. Este método se invoca cada vez que una nueva tarea llega a la cola
SchedMatch()	El método de combinación selecciona las tareas ingresadas con los gestores de recurso. Por defecto el planificador crea una asignación inicial que puede ser modificada. Este método se llama durante cada ciclo de planificación.
SchedPost()	Este método se llama antes de que el ciclo de planificación termine y después de que la decisión de enviar la tarea ha sido tomada para limpiar la información temporal que fue generada durante la planificación. Este método se invoca cada ciclo de planificación por el servicio de cola una vez que se ha llamado a l método `schedMatch()`

Tabla 7. Métodos interfaz para el planificador

5.4. Servicio Adaptador de Gestión de Recursos

El metaplanificador acepta requerimientos descriptos por el lenguaje de especificación de requerimientos y los traduce a un formato comprensible por el planificador del cluster local. Esta traducción se realiza por un adaptador (RM-Adapter), y un solo adaptador sirve para múltiples planificadores locales si estos corren la misma versión del planificador.

5.5. Servicio de Reserva

El servicio de reserva permite, por parte de los usuarios, reservar los recursos bajo el control de un gestor que garantiza que los recursos estarán disponibles para la ejecución de áreas en un tiempo futuro. La reserva puede ser realizada por cantidad de CPUs, tiempo de comienzo o finalización, nombre de usuario, nombre del servidor o tipo de servidor. El servicio de reserva soporta reservas de cualquier tipo de recurso incluso servidores, licencias, o ancho de banda de red.

Una vez que se ha configurado el servicio, la reserva puede ser realizada usando el servicio de reserva. La tarea puede ser creada usando un identificador de reserva, y utilizar un documento de acuerdo, que puede ser encontrado en el sistema Grid. Los valores en el acuerdo incluyen: requerimiento, servidor, CPU, usuario, tiempo de comienzo y final.

5.6. Políticas de Planificación usando proveedores de índices.

El servicio de índices obtiene datos de estado desde diferentes recursos. Las políticas avanzadas de planificación deben considerar los atributos de los recursos y la carga que en ese momento tiene cada recurso. También se debe considerar la información de los planificadores locales y gestores de recurso. El servicio de índice del sistema Grid recolecta estos datos y los asocia al servicio de datos. También provee información estática asociada a los servicios instalados como nombre, localización y lista de control de acceso (ACL). El servicio gestión es el responsable de obtener los datos dinámicamente. El sistema Globus usa el servicio de provisión de información (MDS) para obtener información de las tareas y recursos de cada cluster en el sistema. El servicio

obtiene la información y la agrega proveyendo servicios de notificación para informar al usuario cuando cambia el estado de algún servicio Grid.

Una de las partes más importantes del esquema es la obtención de información sobre las áreas y recursos; esta información servirá para la toma de decisiones y la verificación de los acuerdos de servicio. La información se obtiene de los distintos nodos Grid de servicios como el sistema de Monitoreo y Descubrimiento (MDS) componente de Globus Toolkit V4 [48]. Este componente puede compilar la información de los servicios de monitoreo y descubrimiento de los distintos nodos del sistema. El esquema MDS requiere que los recursos se registren en el servicio de agregación, que periódicamente obtiene la información del estado de cada uno de estos recursos. En los clusters se dispone de una interfaz capaz de lograr una imagen uniforme llamada *GLUE* [49] que obtiene información de los planificadores de clusters e información del sistema, como por ejemplo Ganglia [50] or Hawkeye [51], y la envía al planificador de la organización virtual.

La tarea de registro se hace automáticamente con la publicación del acuerdo de servicio; esto muestra que el servicio de agregación se encuentra en el PDP. Basada en esta información el metaplanificador podrá tomar la decisión sobre la ubicación donde enviar la tarea. De esta manera se minimiza el acoplamiento de componentes y se logra la abstracción de alto nivel correspondiente a tareas Grid y recursos, para que el PEP sólo deba disponer de adaptadores para los sistemas de planificación locales como OpenPBS [10], SGE [54], LSF [11]. De esta manera se logra la combinación de alcance de políticas locales y globales.

5.7. Acuerdos de Servicio

Se ha definido un estándar para la negociación de acuerdos de servicio Grid y dispositivos de gestión llamado WS-Agreement [42, 43, 45, 46, 47]. Se usa este protocolo en ambientes que no están sujetos a control centralizado, como el que se describe en este capítulo. La negociación se logra a través de un método interactivo entre el requirente del servicio y el proveedor del mismo.

El método que requiere contiene una descripción de la reserva incluyendo el tipo de recurso y el tiempo que estima necesitar dicho recurso; en la Figura 17 se puede ver una estructura del acuerdo de WS-Agreement. El sistema gestor del recurso puede responder de manera positiva, o negativa, si es incapaz de satisfacer el requerimiento. Se contempla que pueda realizar una contra-propuesta con parámetros alternativos. La respuesta positiva indica que la reserva es posible pero también puede contener modificaciones sobre los parámetros originales, y contiene un identificador de reserva y una duración del requerimiento. Si el servicio requirente acepta la propuesta se completa la fase llamando al servicio de reserva con el identificador antes de que finalice el tiempo de duración del requerimiento. En la segunda fase se debe coordinar con el sistema de reserva, que implica comenzar negociaciones con múltiples dominios y gestores locales para lograr satisfacer el pedido.

Al usar WS-Agreement como modelo para representar los recursos y definir el servicio que prestará cada uno, el proveedor del servicio enviará el acuerdo al PDP local y al PDP de la organización virtual. En este escenario, el acuerdo define una lista de aplicaciones a ser ejecutadas y el software del entorno a ser ejecutado, cuando se realice la invocación del servicio. Al consumidor del servicio se le garantiza el servicio en término de número

de nodos, memoria o tiempo permitido de ejecución. La aplicación que requiere el servicio debe contemplar el nombre de la aplicación a ser ejecutada y los archivos de entrada y salida, además del numero de nodos y otro recursos necesarios para la ejecución.

Para enviar la tarea, el servicio consumidor requiere el documento acuerdo del PDP, selecciona el nombre de la aplicación y provee el URL de los archivos. Este documento se envía al PDP, que realiza un análisis, negociando la decisión con otros PDPs de la organización virtual a la que pertenece el recurso y decide si aceptar o rechazar el pedido. Esto depende de la cantidad de tareas que se encuentran en espera a ser procesadas o el nivel de carga del recurso.

Figura 17. Esquema de acuerdo

6. Adaptación de CSF a espacios virtuales

Conceptualmente, CSF se adapta naturalmente a la propuesta de la RFC 2753. Los módulos del metaplanificador, colas, reserva y tareas, pueden cumplir el rol del PDP; el adaptador de recursos locales o RM Adapter cumple el rol del LPDP, y por último las máquinas virtuales cumplen el rol de PEP. Así como también el servicio de reserva cumple con el estándar de WS-Agreement. Un esquema de esta propuesta puede observarse en la Figura 18.

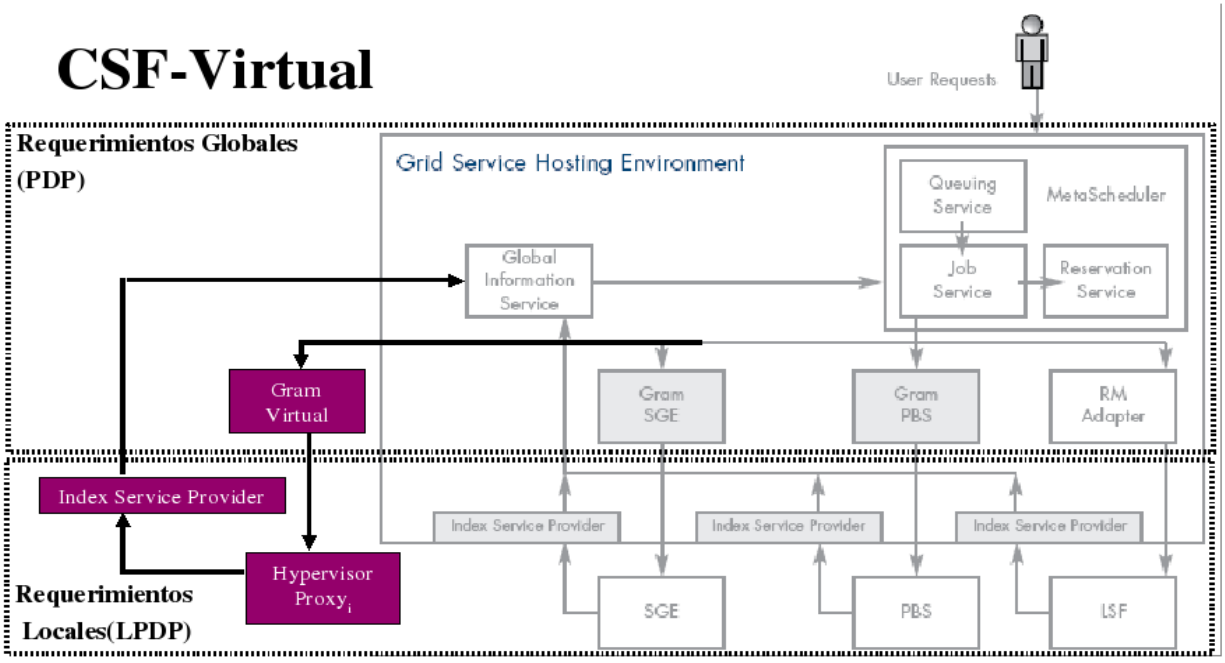


Figura 18. Arquitectura extendida de CSF

Los módulos griseados son parte del marco de trabajo. Los módulos extendidos en esta tesis son las cajas rectangulares “GRAM Virtual”, es el encargado de obtener el

equipamiento solicitado por los módulos “Job Service” y “Hypervisor Proxy”, con tareas de administrador local. A diferencia de la propuesta del marco original, puede haber varios módulos “Hypervisor Proxy” coordinados trabajando en una ejecución bajo “GRAM Virtual”. Los rectángulos con bordes punteados indican la relación con la RFC antes mencionada.

Una diferencia importante que surge en la extensión y las clases provistas del framework es que los adaptadores de recursos, como los planificadores locales, se encuentran definidos antes de hacer el requerimiento de procesadores por parte de la aplicaciones, y si bien cada adaptador de recursos puede acceder a más de un recurso dentro de la Grid, este acceso siempre es de uno por vez. En el caso del adaptador de máquinas virtuales los recursos son dinámicos y con características heterogéneas, y además por definición si la cantidad de procesadores no se logra satisfacer en un recurso físico se deberá completar el requerimiento complementándolo con otros recursos disponibles en la Grid. Esto genera múltiples combinaciones y alternativas que son necesarias evaluar para obtener una solución aceptable.

Otra diferencia en la extensión del esquema propuesto en CSF es el análisis estático que realizan los planificadores de cluster: una vez que las tareas fueron asignadas a sus máquinas físicas, estas se ejecutan allí sin cambios. El uso de máquinas virtuales otorga mayor libertad de gestión en tiempo de ejecución, y según la magnitud de requerimientos se puede modificar la cantidad de memoria asignada, cantidad de procesadores; incluso se puede migrar la máquina virtual completa para un mejor aprovechamiento del cluster. Toda esta inteligencia debe ser incorporada en el módulo de gestión de recursos locales ya que los planificadores habituales (como los que se ven en la Figura 18), SGE o PBS, generalmente son solo sistemas de colas; estas características se adaptan perfectamente en el esquema de puntos de toma de decisión local.

6.1. Gestión de máquinas Virtuales

El módulo de planificación local o administrador de recursos es un servicio disponible en cada nodo Grid, y posee las facultades de crear y administrar dinámicamente los recursos de la organización local. Debido a la variedad de recursos disponibles y a la complejidad de cada uno, se utilizó un protocolo de administración uniforme que se encuentra especificado en el proyecto OASIS WSDM (Web Services Distributed Management) [55] y se utilizó la implementación Apache Muse. Este protocolo basado en servicios web permite disponer de un solo punto de acceso para un conjunto de tareas administrativas y realizar la integración entre los distintos dispositivos gracias al uso de estándares.

En el caso de este trabajo los recursos involucrados en la gestión de cluster virtual son switchs virtuales, máquinas virtuales y conexiones SSH. Todos estos recursos tienen su interfaz publicada a través de WSDM. Las distintas instancias de cada recurso son registradas y accedidas desde el módulo de gestión de recursos, quien según la información obtenida del entorno, ajusta dinámicamente la asignación de recursos virtuales y físicos. Para el acceso a las conexiones SSH y switches virtuales, los servicios web se implementan ejecutando comandos propios del sistema. En el caso de máquinas virtuales este aspecto es diferente porque, por definición, la arquitectura permite el uso de distintas distribuciones de máquinas virtuales, total o parcialmente virtualizadas como por ejemplo Xen[56], KVM[57] o Qemu[58]. En este caso se utiliza la librería Libvirt para

interactuar con las capacidades de cada máquina, poder interconectarlas y hacer un uso coordinado de las mismas.

Cada vez que llega el requerimiento de una aplicación al planificador global, este realiza consultas a los administradores de recursos de cada organización física; cada uno de ellos, según las políticas definidas en cada organización, ofrece determinados recursos con restricciones horarias o de calidad de servicios. Los módulos de administradores locales deberán hacer cumplir las restricciones impuestas por las políticas para hacer un uso racional de los recursos.

7. Experiencias realizadas sobre la gestión de entornos virtuales

Para la realización de las experiencias del gestor de recursos locales se instaló un cluster con máquinas PIV de 3.06 Ghz con tecnología HT y 1GB RAM, y Intel Core 2 Duo, 1.86Ghz por núcleo. Cada máquina cuenta con sistema operativo Linux distribución CentOS 5, y kernel 2.6.18-8.1.3.el5xen. Una de ellas dispone de middleware Grid Globus 4.0.4[8] y WSDM Apache Muse 2.0.2.

Las experiencias que se realizaron tuvieron como objetivo ver la factibilidad de administración dinámica de máquinas virtuales durante la ejecución de una aplicación paralela del cluster a través del metaplanificador. Las pruebas consistieron en el envío de tareas a grupos de máquinas remotas y una vez allí, modificar parámetros de los entornos virtuales como memoria y cantidad de procesadores asignados, así como también con la migración de los mismos. De esta manera se logró determinar el impacto de los cambios en tiempo de ejecución sobre las aplicaciones paralelas. Para la experiencia se utilizó una aplicación de simulación con uso intensivo de CPU y baja entrada/salida y el modelo de paralelismo master-worker.

Se utilizaron las variables memoria y cantidad de procesadores debido a que son dos de los parámetros que generalmente se conocen a priori y son parte del acuerdo de servicio en la ejecución de la aplicación. En esta experiencia se sometió a la ejecución de un nodo de ejecución un stress de memoria. La ejecución de la aplicación comienza con un requerimiento de uso de memoria de 256MB y sin que la ejecución se detenga desde el PEP se disminuye la cantidad a intervalos regulares. En la Figura 19 se puede observar la reacción de la máquina virtual a esa modificación durante dos corridas de la aplicación. La primera corrida simbolizada por un círculo con una letra “a”, es una corrida sin modificación; en la siguiente, la máquina virtual comienza con 256MB Ram y luego se le restringe a un mínimo donde debe comenzar a descargar datos a disco, esto también se refleja en el uso de CPU donde la mayor parte de los procesos son esperas de entrada/salida. Una vez que se observó este comportamiento se modificó nuevamente la memoria, ampliándola a 96MB donde la máquina retomó su normal funcionamiento. En este caso podemos ver que la máquina virtual comenzó con 256MB Ram pero solo eran necesarios 96MB para un correcto funcionamiento, encontrando de esta manera el requerimiento inferior para la ejecución.

Para las pruebas de asignación de procesadores se verificó que el porcentaje de utilización fue variando proporcionalmente procesadores asignados y los procesos que en ellos se ejecutaban. Esto cobra importancia cuando se intenta asignar mayor prioridad a trabajos ejecutados en distintas máquinas virtuales o realizar consideraciones especiales

como por ejemplo variar la cantidad de memoria por máquinas virtuales o asignar a un proceso una CPU al 100% y a otros dos a un 50% cada uno.

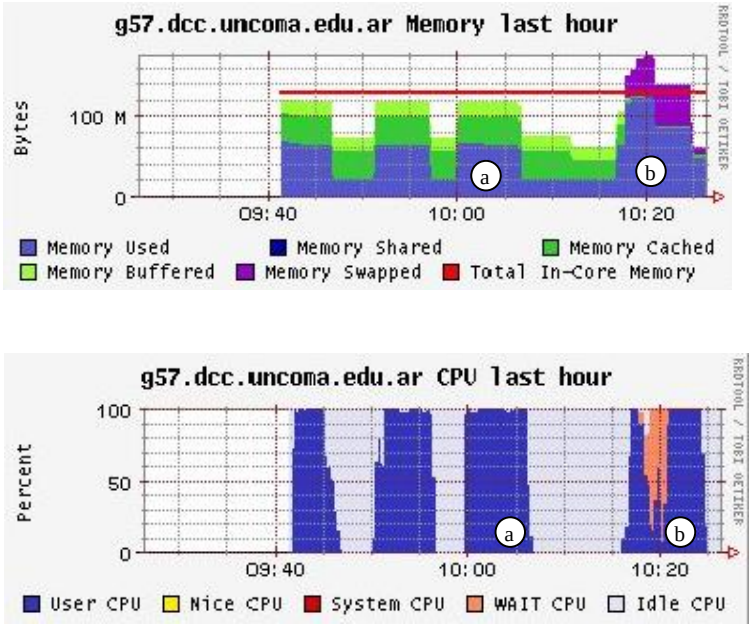
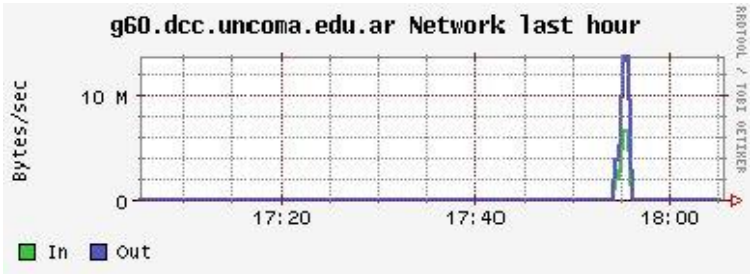


Figura 19. Modificación parámetro de memoria

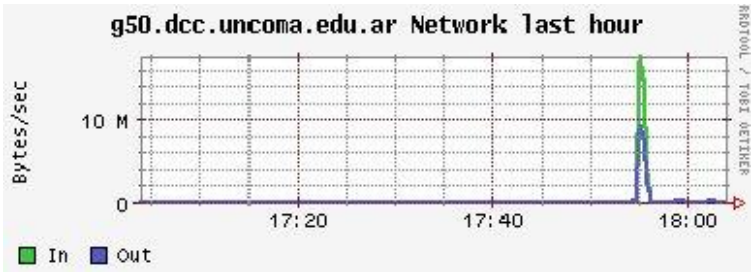
Otro caso donde se probó la utilidad del LPDP fue con la configuración de un cluster de equipos con doble procesador, unos dedicados a satisfacer requerimientos de la organización local y otros a los requerimientos de la organización virtual. Sin embargo los requerimientos locales tienen picos de trabajo en determinadas fechas u horas y en el resto del tiempo son esporádicos. Dejar sólo dos procesadores disponibles para satisfacer los requerimientos de Grid y otros ociosos sería costoso. Por este motivo, por política de la organización local, el administrador de recursos ofrece cuatro procesadores a la organización virtual, pero si llega algún requerimiento local tendrá prioridad sobre los procesos de la organización virtual. Si relacionamos estos conceptos con los de la Figura 19, donde cada máquina tiene un costo económico asociado, el costo será menor en el cluster en donde no necesariamente estarán todas las máquinas disponibles todo el tiempo por las políticas de la organización local.

Para el caso de uso mencionado un ejemplo es el siguiente: en una de las máquinas se encuentra ejecutando una aplicación local a la organización, con dos procesos, cada uno utilizando su CPU asignada al 100%. Al entrar un requerimiento desde la organización virtual y como el administrador de recursos ofreció cuatro procesadores se han generado cuatro procesos sobre dos máquinas virtuales en una única máquina física, ya que la primera está siendo ocupada por el requerimiento local, y se han asignado dos procesos a cada procesador, cada uno de ellos usando la CPU al 50%. Cuando la aplicación local termina, el administrador de recursos LPDP toma la decisión de migrar y el PEP migra automáticamente una de las máquinas virtuales con el requerimiento global, sin detener la ejecución de la aplicación. La máquina virtual se encarga de mantener toda la información y enviarla a la nueva máquina física reasignando nuevamente los procesadores, quedando cada uno con el uso de la CPU al 100%. En el caso de que las

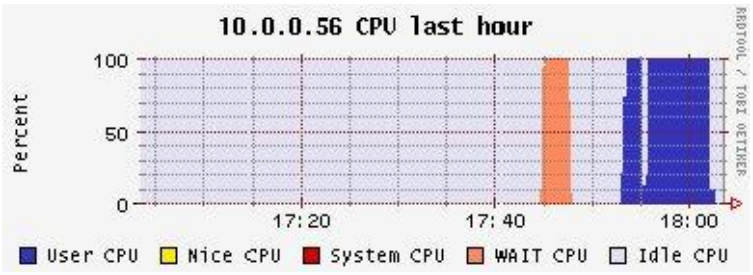
políticas hubieran sido fijas, una vez terminada la aplicación local, el equipamiento asignado a esas tareas hubiera quedado ocioso. De esta manera se obtiene una mejora en el aprovechamiento del equipamiento solo con adaptar las políticas.



(a) Transferencia de I/O Física Origen



(b) Transferencia de I/O Física Destino



(c) Ejecución durante la migración en la Máquina Virtual

Figura 20. Proceso de migración máquina virtual

En la Figura 20 se muestra un ejemplo de proceso de migración de máquinas virtuales [60]. En la Figura (20.a) se encuentra el tráfico de red de entrada y salida de la máquina física origen, en la Figura (20.b) el tráfico de red de entrada y salida en la máquina física receptora. En la tercera Figura de la serie se ve el proceso de simulación de la máquina virtual; como se aprecia, éste en ningún momento se interrumpe solo se atenúa unos segundos cuando migra definitivamente, no durante todo el proceso de transferencia de estado. El proceso de transferencia se hace en forma paralela a su ejecución y el impacto en la performance es mínimo.

8. Resumen

Debido a la gran cantidad de recursos de hardware y software que componen los sistemas Grid, cada uno con diferentes características y complejidades, se torna imperioso simplificar y automatizar su administración. En este capítulo se presentó una arquitectura con dos niveles de planificación, el primero a nivel de meta-organización, con

el objetivo de generar clusters a demanda basado en requerimientos de aplicaciones y el segundo a nivel de organización local, gestionando los recursos del cluster en forma dinámica para lograr un máximo aprovechamiento de los recursos ofrecidos. Como recursos para generar el entorno virtual se definieron máquinas virtuales que interconectadas forman un cluster homogéneo entre organizaciones.

Se ha presentado la extensión de un metaplanificador Grid, detallando sus componentes, y haciendo hincapié en los agregados para poder generar laboratorios remotos de máquinas virtuales (dentro de un esquema de administración basado en políticas).

Para el administrador local de recursos se ha verificado a través de experimentación los beneficios del uso de máquinas virtuales para lograr un máximo aprovechamiento de los clusters pudiendo modificar de manera homogénea parámetros de ejecución como memoria y cantidad de procesadores en forma dinámica y migrar máquinas virtuales reasignando el espacio de máquinas físicas sin tener que detener la aplicación que está corriendo en el cluster.

El aporte de este capítulo con la generalización de la RFC 2753 a entornos Grid más la extensión del meta-planificador CSF para la generación de espacios virtuales, permitieron concluir que ésta tecnología es factible de aplicarse en espacios Grid, donde existe la necesidad de adaptación y control del entorno. En este sentido redunda en beneficios importantes para la calidad de administración y una mejor utilización de recursos.

Capítulo 4. Gestión en entornos Virtuales[1]

La disponibilidad e integración de recursos tecnológicos dispersos geográficamente es uno de los desafíos actualmente más importantes. La búsqueda de este tipo de soluciones trae aparejado incrementos en los requerimientos de software y hardware, así como también en la complejidad de su administración debido a la heterogeneidad de los recursos y la necesidad de mantener una postura de seguridad en ambientes distribuidos. Este capítulo propone una solución a estos aspectos implementando la arquitectura de gestión, propuesta en el capítulo anterior. La misma utiliza dispositivos virtuales para lograr un marco de trabajo homogéneo, permitiendo al mismo tiempo una configuración dinámica y flexible, dentro de una organización virtual sobre un entorno Grid. Con foco en la interconexión de recursos de cómputo, de esta manera asegura un espacio de trabajo con estándares y protocolos abiertos bajo estrictas normas de seguridad.

1. Casos de Uso

La solución que se presenta admite tres casos de usos, o aplicaciones, de propósitos diferentes:

- **Laboratorio Remoto**. En este caso de uso, la incorporación de nodos Grid permite aumentar y gestionar de manera transparente distintos recursos pertenecientes a dominios administrativos diferentes, como si fueran pertenecientes a una red local.
- **Computación Paralela**. En ambientes Grid existen dos formas diferentes de realizar cómputo paralelo. La primera es considerar a los clusters como un servicio Grid; la segunda es ver a cada nodo Grid como un worker del cluster, usando bibliotecas Mpich-G2 para la implementación del cómputo paralelo. La primera

[1] Los trabajos implementados en este capítulo, se realizaron con la colaboración del Lic. Eduardo Grosclaude y el Lic. Francisco Lopez Luro.

forma excluye la posibilidad de sincronizar a workers individuales en distintos clusters. Este trabajo posibilita la sincronización de los workers de distintos nodos Grid como en la segunda opción, sin tener que pasar a través de las múltiples capas a las que nos obligan los servicios Web, y de esta manera mejorar las prestaciones, además de obtener una mejora al tener múltiples nodos locales trabajando en conjunto.

- **Workflows Grid**. Esta alternativa no fue implementada como prueba en esta tesis, pero es válida como caso de uso. Utilizando el acceso mediante servicios Web se puede acceder a la red de recursos virtuales como cualquier otro recurso del Grid. Esta solución facilita la configuración y la coordinación de los equipos de cómputo, permitiendo utilizar la red privada virtual (VPN) sobre los elementos como una componente o servicio y componiéndola en un flujo de trabajo.

A continuación se describen los módulos que componen la solución y analizará la implementación sobre los dos casos de uso seleccionados; el primero, de redes para laboratorios remotos, y el segundo, un entorno de computación paralela.

2. Arquitectura

La arquitectura propuesta se divide, desde el punto de vista de aspectos de diseño, en tres instancias o zonas donde se han tomado ciertas decisiones independientes. La primera, de acceso, donde se definen los clientes que pueden acceder al sistema; la segunda, de gestión del sistema donde se realiza la toma de decisiones (PDP), conteniendo la definición del control de acceso y la creación de los recursos; y la tercera, la de implementación de los recursos físicos y virtuales, donde se aplican las políticas de ejecución (PEP).

Un diagrama de esta arquitectura en la Figura 21 muestra a los clientes que a través de internet acceden a una organización virtual, compuesta de cuatro nodos Grid en dos organizaciones físicas A y B. Las mismas poseen distintos tipos de recursos disponibles para el Grid, pero tienen en común clusters dedicados que permiten la instanciación de máquinas virtuales.

La solución presentada hace aparecer un espacio donde pueden crearse redes virtuales, independientes, cuya administración es totalmente separada de las redes físicas que las soportan, sin tener que ejecutar complejas tareas de coordinación entre los administradores locales. El espacio virtual realiza tareas similares a una solución de red privada virtual con recursos virtuales, por esto se llamará V2PN.

2.1. Zona de clientes

Para la primera instancia se consideran tres tipos de clientes:

- El cliente por defecto de Grid, que en la implementación Globus es el cliente WS_GRAM. Permite acceder a los recursos de uno o más nodos Grid mediante línea de comandos. Este cliente se utiliza en la gestión de los recursos en los distintos nodos Grid.
- Un cliente de web, que no posee ningún sistema o bibliotecas propias de Grid para poder ejecutar las tareas. Este cliente va a ser el usuario final del sistema, que no debería conocer cuestiones de implementación, y su espacio de trabajo es un espacio virtual.

- Un cliente de acceso seguro a terminales remotas utilizando el modelo de seguridad Grid. Es un usuario especial del primer tipo que podría ser el docente o un administrador del entorno Grid.

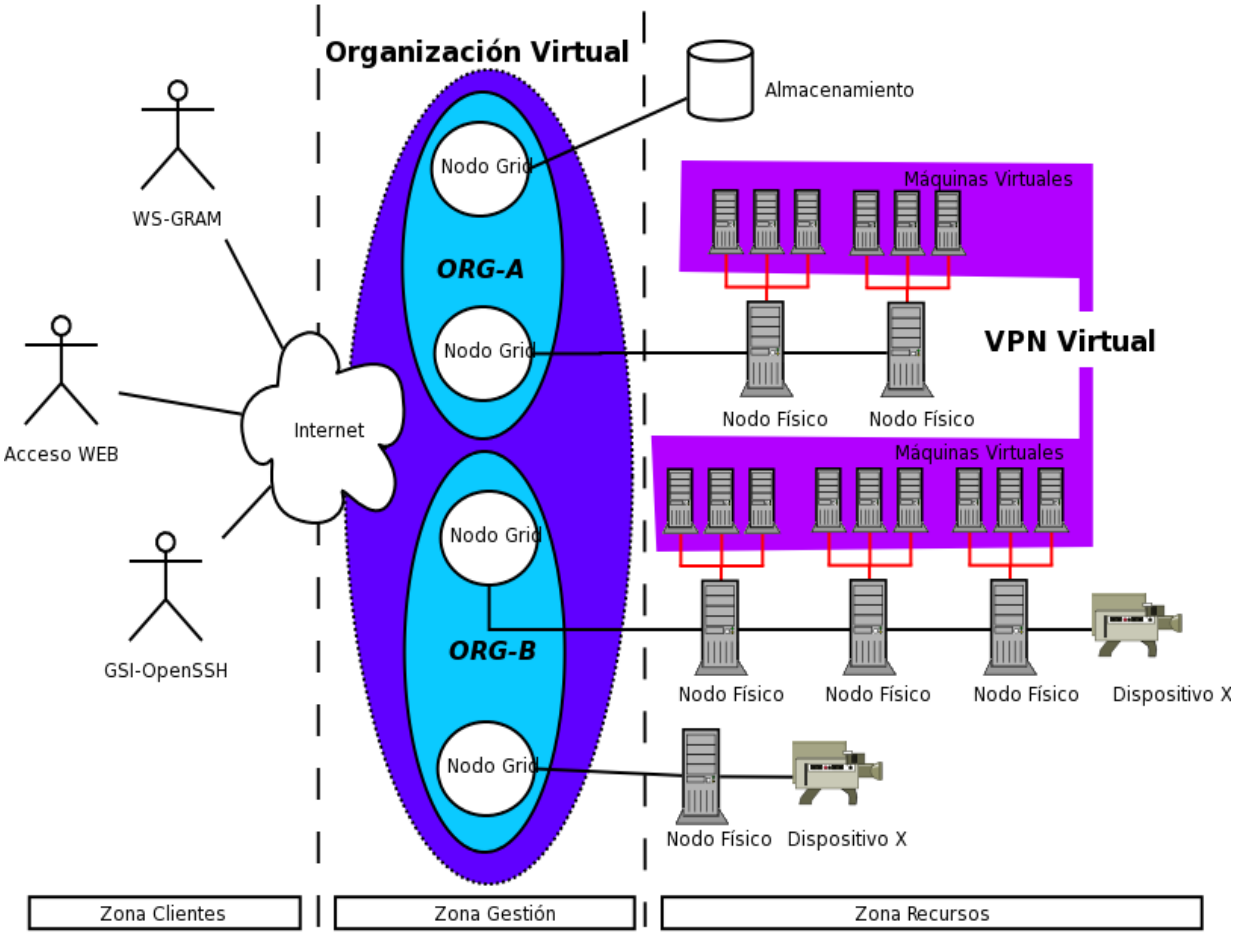


Figura 21. Arquitectura de laboratorio

2.2. Zona de Gestión

- En la segunda instancia se encuentran los módulos de gestión de la plataforma Grid. Para el presente trabajo se utilizará el middleware Globus Toolkit en su versión 4.0.2. Estos nodos permiten el vínculo con el espacio de direcciones privadas de los clusters locales. La ejecución de tareas en este espacio se realiza siempre a través de estos nodos. En el ámbito administrado por estos nodos Grid se podrán ejecutar archivos de comandos para trabajar sobre las máquinas físicas y virtuales.
- Para el acceso de los usuarios finales y como enlace entre los nodos virtuales de la red privada en los nodos Grid, se dispondrá de un elemento proxy a los efectos de multiplexar los accesos. Este proxy permite acceder a través de un nodo Grid a las máquinas virtuales bajo su administración, por ejemplo, con el protocolo SSH, sin que sea necesario proporcionar complejos parámetros. En nuestra solución, este elemento no constituye un servicio Grid.

2.3. Zona Recursos

En la tercera instancia se encuentran los nodos físicos y virtuales del sistema. Los nodos físicos quedan dedicados a los servicios requeridos desde el nodo Grid. Cada uno de estos nodos físicos podrá instanciar una o más máquinas virtuales. En esta implementación utilizamos la tecnología de virtualización Xen para la creación de máquinas virtuales (un ejemplo de configuración puede observarse en el Código 3). Los mecanismos de acceso implementados permiten interconectar también máquinas UML, QEMU o simuladores de dispositivos de interconexión como Dynamips (CISCO 7200 Simulator) [66]

```
<maquina_virtual>
<host>cluster_maq1</host>
<usuario>root</usuario>
<vm_class>xen</vm_class>
<vm_args>
<kernel>/boot/vmlinuz-2.6.17-1.2187_FC5xenU</kernel>
<memory>64</memory>
<name>maq1</name>
<network>gateway=10.0.5.1,dns1=10.0.5.1</network>
<vifs>
<vif0>ip=10.0.5.99,netmask=255.255.255.0,lan=xenbr0</vif0>
<vif1>lan=xenbr1</vif1>
</vifs>
<disk>/root/imagenes/FedoraCore5_francho</disk>
<root>/dev/sda1</root>
<extra_kernel>ro,selinux=0,hostname=francho,3</extra_kernel> </vm_args>
</maquina_virtual>
```

Código 3. Ejemplo de parámetros del servicio iniciar máquina virtual

3. Casos de Uso Implementados

Los dos casos seleccionados para la prueba de concepto tienen distintos objetivos, y sus escenarios correspondientes presentan diferentes topologías y requerimientos de interacción.

- En el caso de Laboratorio Remoto, los enlaces tienen atributos específicos, como ancho de banda o confiabilidad, y son una parte integral del funcionamiento del escenario realizado. El acceso es esencialmente interactivo.
- En el caso de Cómputo Paralelo, el interés está puesto en la capacidad de procesamiento efectiva de las máquinas utilizadas. Las tareas se lanzan en modo batch y no son interactivas.

Como consecuencia, ambos tipos de escenario presentan diferentes metas y restricciones de mapeo al conjunto de soporte físico.

3.1. Meta-Planificador

Los nodos Grid cuentan con información de estado de los servidores de la planta física. El planificador, basándose en información de configuración y estado de esos servidores físicos, determinará la asignación o mapeo de nodos virtuales a nodos físicos. Si se requiriesen más máquinas virtuales de las que un nodo Grid pudiera ofrecer, por no encontrarse disponibles o debido a su nivel de utilización, se buscará instanciar recursos virtuales en distintos nodos Grid formando una red virtual entre ellos.

Los requerimientos para la fase de planificación son:

- Capacidad de detectar qué máquinas físicas de la organización virtual disponen de software de virtualización.
- Para el caso de uso de cómputo paralelo, la asignación llevada a cabo por este planificador instanciará no más de una máquina virtual por equipo físico; todas estas máquinas virtuales se encontrarán conectadas en un solo espacio de direcciones; el resultado devuelto al usuario será un punto de acceso único.
- Para el caso de uso de Laboratorio Remoto, el planificador instanciará distintos tipos de máquinas virtuales según lo expresado en el escenario requerido, y tantas instancias por máquina como la memoria de los equipos físicos lo permitan. Estas máquinas también se encontrarán en un solo espacio de direcciones pero el resultado de la operación del planificador serán tantos puntos de accesos como máquinas virtuales haya instanciado.

Además de proponer un mapeo de servidores virtuales a servidores físicos, el planificador debe explicitar los puntos de acceso a los elementos virtuales que se instanciarán. En este trabajo, los puntos de acceso son nombres de dispositivos de caracteres (normalmente ttys) que permiten flujo de datos y control entre el cliente y las máquinas virtuales.

En el caso del Laboratorio Remoto, la topología física puede utilizarse de acuerdo a la topología lógica (mapeando enlaces de ciertas características a vínculos físicos de características similares). En el caso del escenario de Cómputo, no tiene mayor sentido instanciar más de una máquina virtual en una máquina física, y en lo posible todas las máquinas virtuales deben estar alojadas en los nodos de un único cluster físico.

Para que esta diferenciación pudiera ser transparente a la interfaz, el algoritmo de mapeo del planificador debería contar con un nivel de inteligencia considerable, y ser capaz de resolver este tipo de cuestiones por examen de la topología deseada y recursos disponibles. Para mantener el elemento de mapeo en un nivel de complejidad aceptable, optamos por hacer explícito el caso de uso en la descripción del escenario.

3.2. Secuencia de Actividades

En la Figura 22 se muestra un ejemplo de acceso a los servicios del sistema involucrado en la ejecución de una solicitud de veinticinco nodos de cómputo. En este caso el cliente se conecta con un servicio *planificador (PDP)*, invocándolo con las necesidades de cómputo parametrizadas mediante un lenguaje de descripción basado en XML. El resultado de esta etapa específica: cuáles serán los nodos físicos involucrados en el servicio, cuántos nodos virtuales se lanzarán en cada uno y qué clase de máquina virtual soportará a cada nodo virtual. Este resultado es enviado como parámetro en el servicio web correspondiente de cada nodo Grid.

Otros parámetros incluidos en el resultado del planificador son el rango de direcciones de red a utilizar, y los puntos de acceso a la V2PN si hiciera falta un acceso interactivo. Cada uno de los nodos Grid instancia las máquinas y devuelve un punto de acceso para que el cliente pueda acceder a esta red de recursos virtuales.

3.3. Especificación de requerimientos de recursos

El límite del sistema, en el extremo próximo al usuario, es la descripción de los recursos

lógicos que se desea instanciar. Esta descripción es la entrada al sistema, y una vez verificada su factibilidad, debe ser mapeada a recursos lógicos, realizados sobre recursos físicos.

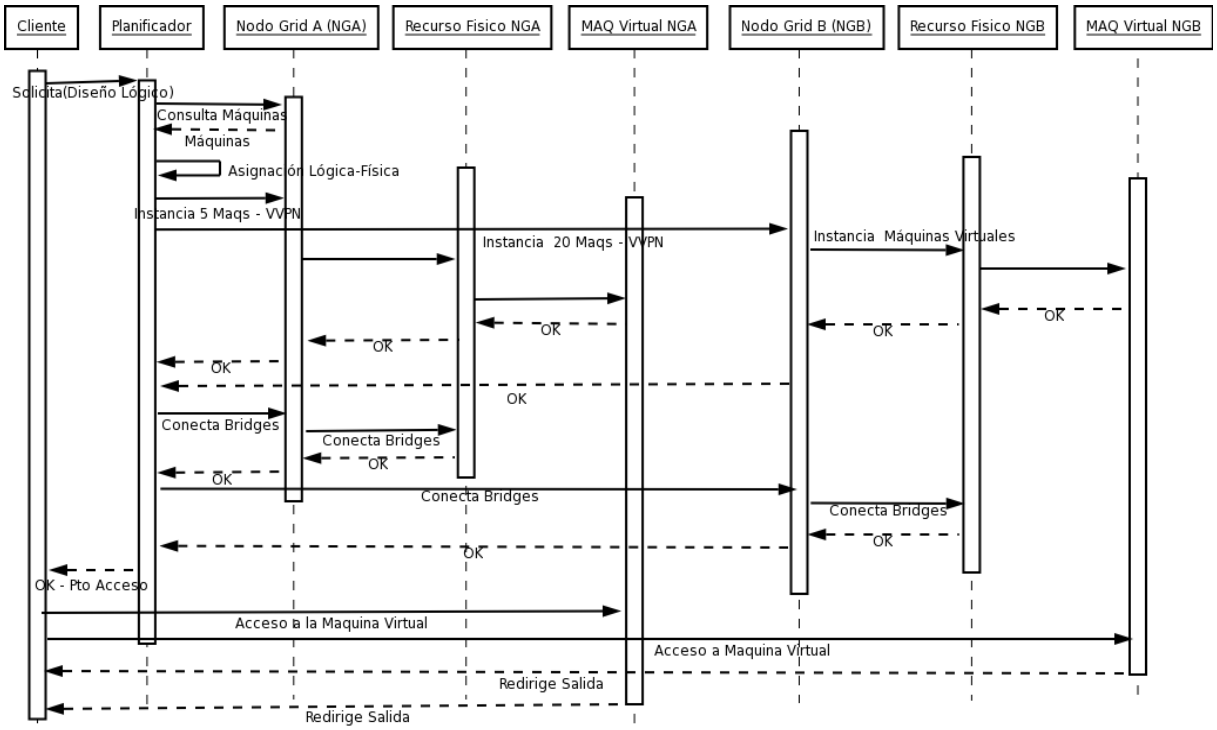


Figura 22. Secuencia de accesos para planificación según modelo de capas

En condiciones ideales, el usuario del sistema contará con una interfaz de edición de requerimientos para producir esta descripción. Por sencillez se ha implementado una interfaz programática reducida, que basta para ejemplificar el uso del sistema. Esta interfaz permite modelar los objetos representados en la Figura 23 (escenario, nodos virtuales y enlaces).

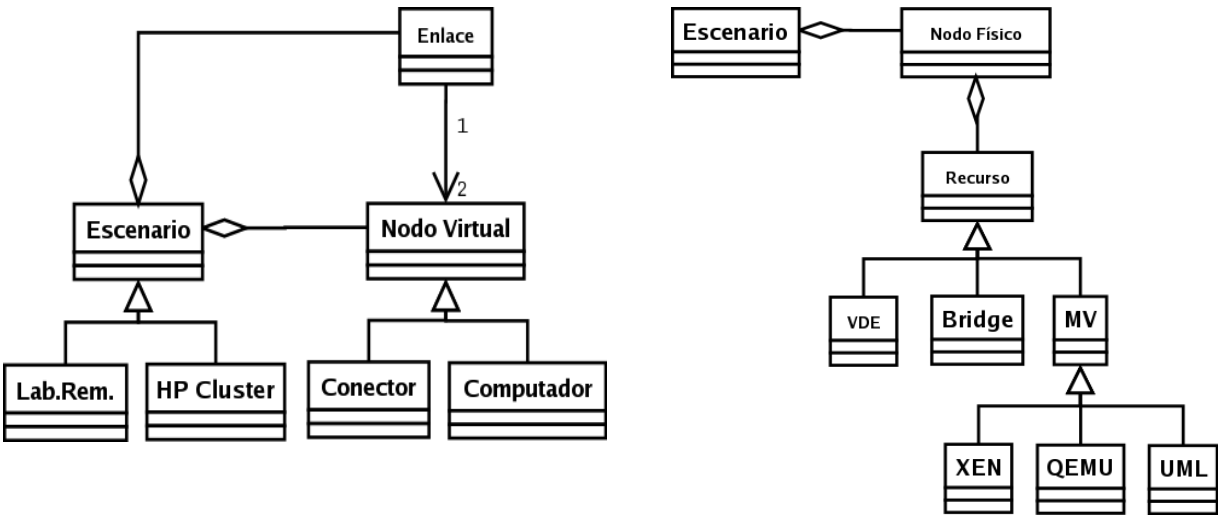


Figura 23. Modelado de objetos representados y Modelado de los elementos de la planta física

Usando un lenguaje de objetos (expresados en un subconjunto de Python), el usuario puede describir un escenario, que es esencialmente un grafo cuyos nodos son los nodos virtuales de la red lógica a implementar, y cuyos arcos son los enlaces que conforman la topología deseada (el Código 4 ejemplifica la creación de un escenario). El planificador resolverá el requerimiento obteniéndose la descripción del escenario a instanciar, donde los elementos virtuales requeridos se han mapeado a elementos de la planta física.

3.4. Mapeo Lógico-Físico

El sistema de Monitoreo y Descubrimiento (MDS4) es un conjunto de servicios web para monitorear y descubrir servicios Grid. Este sistema permite a los usuarios descubrir qué recursos son considerados parte de la organización Virtual y monitorear el estado de dichos recursos.

MDS4 hace uso intenso de XML e interfaces de servicios web para simplificar las tareas de registro y localización de interés. En particular, toda la información obtenida por los servicios de agregación se mantiene en XML y puede ser accedida por consultas Xpath, así como por otros mecanismos de consulta a través de servicios web.

Ejemplo de especificación de diseño lógico	Ejemplo de especificación de diseño lógico
#!/usr/bin/python from red import * e = escenario('prueba2') sw1 = e.add_nodo(switch('Switch_A',16)) pc1 = e.add_nodo(pc('PC_1',128)) pc2 = e.add_nodo(pc('PC_2',256)) e.add_arco(sw1.conectar(pc1)) e.add_arco(sw1.conectar(pc2)) hub2 = e.add_nodo(hub('Hub_B',8)) e.add_arco(hub2.conectar(pc2))	<escenario id="prueba2"> <nodos> <nodo id="Switch_A"> <tipo>switch</tipo> <bocas>16</bocas> <conexiones> <nodo id="PC_1" /> <nodo id="PC_2" /> </conexiones> </nodo> <nodo id="PC_1"> <tipo>pc</tipo> <mem>128</mem> </nodo> <nodo id="PC_2"> <tipo>pc</tipo> <mem>256</mem> </nodo> <nodo id="Hub_B"> <tipo>hub</tipo> <bocas>8</bocas> <conexiones> <nodo id="PC_2" /> </conexiones> </nodo> </nodos> <arcos> <arco id="Switch_A-PC_1"> <nodo1>Switch_A</nodo1> <nodo2>PC_1</nodo2> </arco>

Código 4. Ejemplo de especificación de diseño lógico

En esta implementación, la jerarquía de obtención de información comienza por monitorear los clusters a través del software Ganglia, un sistema distribuido de monitoreo para sistemas de alta performance que puede ser accedido a través de Internet por un navegador o puede exportar sus datos en formato XML a otras aplicaciones. Esta facilidad fue utilizada para vincular los datos de los cluster con cada uno de los nodos Grid. Cada nodo Grid concentra los datos de su cluster, y a la vez un nodo Grid centraliza esta información generando un árbol de información actualizada en segundos. Esta información también puede ser accedida por una interfaz Web, como se ve en las Figuras 24 y 25 o puede ser consultada a través de Xpath.

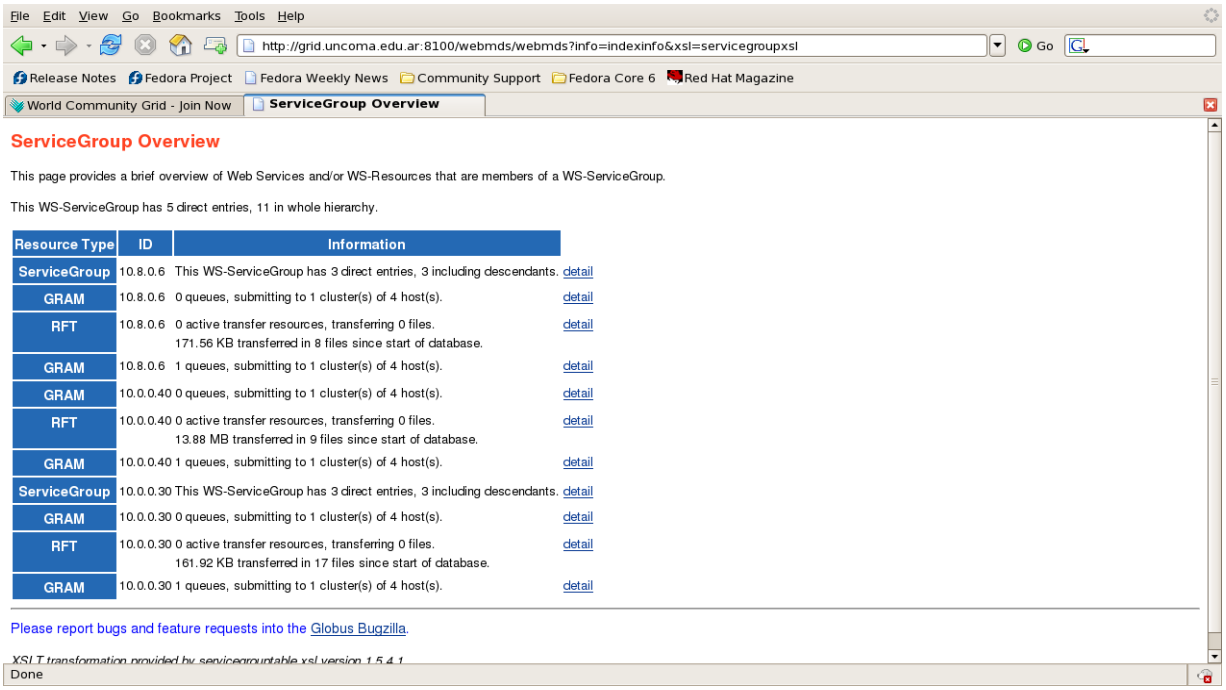


Figura 24. Ejemplo de pantalla de información MDS por Web

Para obtener información sobre las máquinas que están activas en los clusters, el servicio planificador realiza la consulta Xpath al nodo Grid concentrador. Dicha consulta se genera dependiendo de los requerimientos expresados en el diseño lógico y puede contener especificaciones como estado de la memoria o capacidad libre del procesador. Un ejemplo puede verse en el Código 5.

Figura 25. Ejemplo de pantalla de información de Clusters

Consulta XPATH

```
wsrf-query -s https://10.0.0.40:8443/wsrf/services/DefaultIndexService
"//glue:Host[glue:OperatingSystem[contains(@glue:Release,'xen')]]"
```

Resultado

```
<ns1:Host ns1:Name="10.0.5.220" ns1:UniqueID="10.0.5.220"
xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
<ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:CacheL1I="0" ns1:CacheL2="0"
ns1:ClockSpeed="3066" ns1:InstructionSet="x86"/>
<ns1:MainMemory ns1:RAMAvailable="339" ns1:RAMSize="931" ns1:VirtualAvailable="2379"
ns1:VirtualSize="2985"/>
<ns1:OperatingSystem ns1:Name="Linux" ns1:Release="2.6.18-1.2849.fc6xen"/>
<ns1:Architecture ns1:SMPSize="2"/>
<ns1:FileSystem ns1:AvailableSpace="64595" ns1:Name="entire-system" ns1:ReadOnly="false"
ns1:Root="/" ns1:Size="71631"/>
<ns1:NetworkAdapter ns1:IPAddress="10.0.5.220" ns1:InboundIP="true" ns1:MTU="0"
ns1:Name="10.0.5.220" ns1:OutboundIP="true"/>
<ns1:ProcessorLoad ns1:Last15Min="0" ns1:Last1Min="0" ns1:Last5Min="0"/>
</ns1:Host>
…
…
```

Código 5. Consulta XPATH para conocer estado del sistema

3.5. Instanciación de Recursos

Una vez obtenido un mapeo lógico-físico factible, y habiendo previamente comprobado en forma dinámica la disponibilidad efectiva de los recursos, se procede a la creación de las máquinas virtuales en los distintos nodos físicos en cada nodo Grid que sea necesario.

Como muestra la secuencia de la Figura 22, el planificador enviará a cada nodo Grid mediante WS las solicitudes de asignación de recursos virtuales. A su vez cada nodo Grid redirigirá dichas solicitudes a los nodos físicos correspondientes. Se implementaron

scripts en Python para traducir la especificación general de creación de máquinas virtuales en archivos de configuración que instancian dichas máquinas en cada nodo físico. La extensión de estos scripts permitirá de manera transparente el uso de nuevas tecnologías de virtualización de recursos.

Como se ha mencionado para cada máquina o recurso virtual que se crea, se deja disponible un punto de acceso en el nodo Grid más cercano, pero independiente y aislado completamente de la red física real.

3.6. Virtualización de la red

Una vez creadas todas las máquinas del requerimiento lógico del usuario, se necesita interconectarlas de tal manera que se refleje el esquema de conexión requerido, pero ocultándose los detalles de las redes privadas y servidores físicos que hay en cada nodo Grid. Por esta razón lo que se necesita es una red privada entre las máquinas virtuales que allí se ejecuten.

Entre las aplicaciones disponibles que proveen facilidades para la creación de VPNs, se escogió el uso de la herramienta VDE_SWITCH[67], creando algunos scripts a medida para conformar una solución que se ajuste a nuestras necesidades teniendo en cuenta aspectos de performance y seguridad de la información, pero con un mínimo de esfuerzo de administración en los nodos físicos.

La Figura 26 muestra un escenario de conexión de tres máquinas virtuales Xen, dos de ellas alojadas en nodos físicos en una LAN A y la otra en un nodo físico en otra red accesible vía internet a través de los nodos Grid.

Como se puede ver en el gráfico, las dos máquinas de la LAN A se interconectan mediante un "cable" virtual (línea punteada) construido con VDE_SWITCH y usando como mecanismo de transporte entre los dos hosts físicos un túnel TCP establecido con NetCat [69]. Una solución similar se utiliza para conectar una de las máquinas de la LAN A con Host C, pero usando en este caso un túnel cifrado mediante una conexión SSH para asegurar la privacidad de la información que se transmite. El encriptado no supone pérdida de rendimiento puesto que las conexiones a internet utilizadas representan el cuello de botella frente al costo del cifrar la información que viaja por el "cable" virtual.

De esta manera, las tres máquinas virtuales Xen quedan interconectadas entre sí de manera transparente y en un espacio de direcciones privado virtual que no tiene posibilidad de colisión con los demás esquemas de direcciones de los nodos Grid participantes, ya que las tramas de nivel 2 de las máquinas virtuales quedan encapsuladas en el túnel.

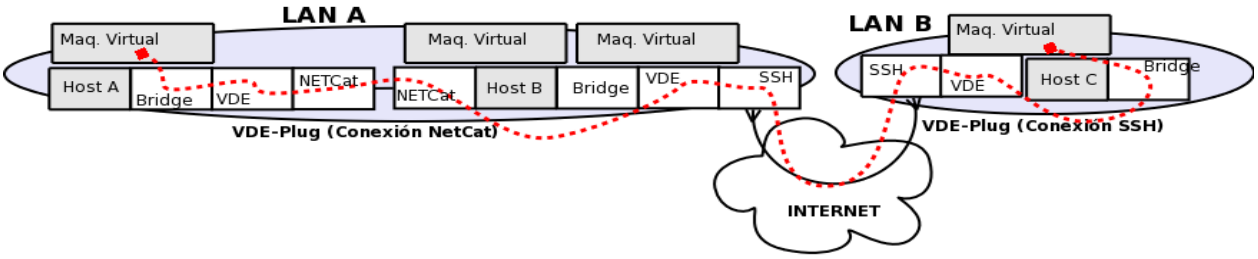


Figura 26. Componentes de la virtualización de la red

3.7. Accesos desde la Web

Si bien son tres los roles de acceso al sistema desde internet, son dos los mecanismos que estos roles utilizan:

- **Acceso Usuario Grid**: El acceso Grid utiliza la infraestructura de seguridad implementada por el software intermedio Globus llamada GSI (Grid Security Infrastructure). GSI provee servicios de autenticación y delegación basados en el estándar X.509 y extensiones del protocolo SSL. Si el cómputo Grid requiere distintos recursos y cada uno de ellos solicita autenticación, como en nuestro caso con el uso coordinado de diferentes nodos Grid, el servicio crea un *proxy*. Este proxy evita volver a ingresar la palabra clave, estableciendo un esquema de *single sign on* sobre la organización virtual. El *proxy* consiste de un certificado y una clave privada.
- **Acceso Usuario Final**: El acceso del usuario final queda disponible una vez que el planificador termina de instanciar las máquinas virtuales y de conectarlas. Mediante la herramienta GNU-Screen se crea un punto de acceso a los recursos en cada nodo Grid. Luego se establece una conexión a través del protocolo SSH al nodo Grid correspondiente especificando el punto de acceso. Por último para facilitar el acceso a través de Internet se genera una página con formato HTML y un archivo de configuración que junto con el applet Java llamado GSI-SSHTerm [68] permiten al usuario final acceder al recurso a través de un navegador de Internet con capacidad Java. Una vez que el usuario final que gestiona la infraestructura ha finalizado el requerimiento lógico y que el planificador haya concretado la solicitud, el usuario tendrá una o más páginas web disponibles para acceder a los recursos de la especificación. Un ejemplo del archivo de configuración se da en el Código 6, donde se encuentran resaltadas las líneas del punto de acceso y el comando de conexión.

4. Implementación de los casos de uso

El entorno de cómputo está compuesto por dos clusters, uno situado en dependencias de la Universidad Nacional del Comahue y el otro localizado en instalaciones facilitadas por la empresa CDF. Ambas ubicaciones están dentro de la ciudad de Neuquén. El cluster de la UNC cuenta con equipos PIV 2.2Mhz y 512Mb Ram y en la empresa existen cinco máquinas Pentium IV de 3.06 GHz con tecnología Hyper-Threading y memoria de 1 GB. La Figura 27 resume estas características de hardware y la disposición de la red donde se monta el espacio virtual.

El software utilizado para la gestión del entorno Grid es Globus Toolkit 4 (GT4), el cual provee:

- Una interfaz llamada GRAM (Grid Resource Allocation and Management) para la ejecución remota de tareas,
- El servicio de RFT (Reliable File Transfer), componente del sistema de transferencia de información,
- El servicio MDS (Monitoring and Discovering Service) para obtener información de los servicios publicados en el Grid.

```
<?xml version="1.0" encoding="UTF-8"?>
<SshToolsConnectionProfile Hostname="labrem.ods.org" Port="22" Username="francholi"
Provider="socket">
  <PreferedCipher Client2Server="blowfish-cbc" Server2Client="blowfish-cbc"/>
  <PreferedMac Client2Server="hmac-sha1" Server2Client="hmac-sha1"/>
  <PreferedCompression Client2Server="none" Server2Client="none"/>
  <PreferedPublicKey Name="ssh-dss"/>
  <PreferedKeyExchange Name="gss-group1-sha1-dZuIebMjgUqaxvbF7hDbAw=="/>
  <OnceAuthenticated value="3"/>
  <AuthenticationMethod Name="password">
    <AuthenticationProperty Name="Username" Value="francholi"/>
  </AuthenticationMethod>
  <ExecuteCommands>screen -S maquinaVirtual_3 </ExecuteCommands>
  <ApplicationProperty Name="sshterm.xForwarding" Value="true"/>
  <ApplicationProperty Name="FONT_SIZE" Value="12"/>
```

Código 6. Parte del archivo de configuración

Para el sistema operativo se optó por las distribuciones Linux CentOS, orientada a servidores, y Fedora Core 6, esta última debido a que provee facilidades para la instalación de máquinas virtuales como Xen. Para este tipo de máquina virtual en particular se utilizó la versión Xen 3.0.3 y en las pruebas de laboratorio de redes se incorporaron además máquinas realizadas sobre QEMU.

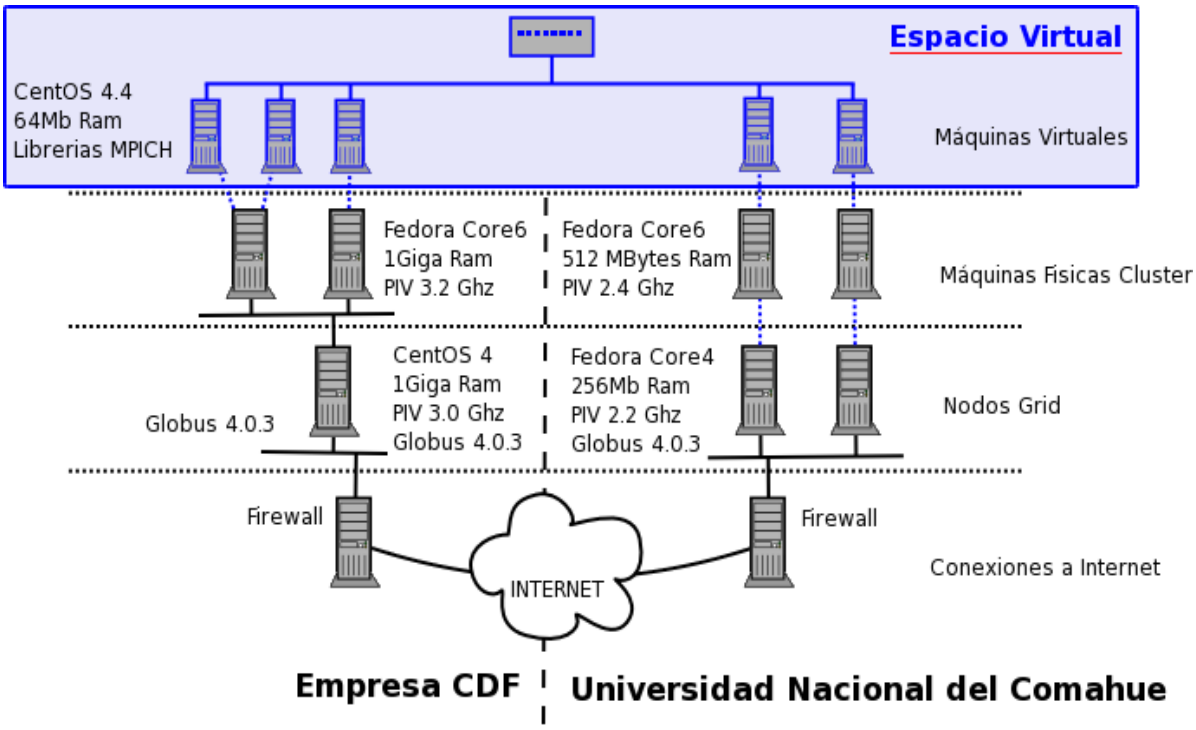


Figura 27. Arquitectura del Caso de Uso

4.1. Implementación: Laboratorio de Redes

Este caso de uso consiste en un escenario de interconexión de computadoras, donde los alumnos ponen en práctica técnicas de configuración y administración de redes. En este caso de uso suponemos que el laboratorio existente en la universidad tiene recursos limitados, haciendo necesaria la incorporación de mayor cantidad de equipos; por este motivo se incorporan las máquinas de una entidad externa (como una empresa) accedidas a través de internet. La empresa dispone de un repositorio de imágenes con las que se puede contar para la realización de una práctica de redes.

El objetivo del caso de uso con respecto a este trabajo es mostrar cómo la infraestructura propuesta permite la instanciación de escenarios de laboratorios virtuales a través de un entorno Grid de manera automática y aprovechando los recursos físicos disponibles en cada organización.

A modo de ejemplo, la práctica consiste en lograr la interconexión de tres nodos, dos situados en una misma LAN conectada a un router, con un tercer nodo en una red diferente también conectada al router. En primera instancia el docente genera una plantilla con el diseño lógico de la práctica sin importar los recursos físicos disponibles en la Universidad. Una vez concluida esta etapa, invoca tantas veces al servicio planificador de la solución como cantidad de grupos de trabajo sean necesarios. El planificador, una vez que se han agotado los recursos del equipamiento de la universidad, comenzará a instanciar máquinas virtuales en la planta física de la empresa, ya que ésta reside dentro de la organización virtual de la universidad y posee servicios publicados para la instanciación y gestión de las máquinas virtuales. El resultado de estas acciones serán los escenarios virtuales en ejecución. Para que los alumnos puedan acceder a los escenarios se han configurado los puntos de acceso y se han generado las páginas en formato HTML bajo un servidor web disponibles para el acceso desde la web.

Los alumnos, mediante un navegador web, acceden a los recursos del laboratorio desde distintos sitios, en forma interactiva bajo un esquema colaborativo. Pueden realizar la práctica propuesta en forma grupal o individual según se requiera. Cabe destacar que el acceso a los laboratorios es transparente a la ubicación física del equipamiento. En las experiencias realizadas se percibieron retardos variables, dependiendo de la capacidad de conexión de los nodos involucrados y de la carga ofrecida a los enlaces en el momento de la realización de las tareas prácticas.

4.2. Implementación: Aplicación Paralela

El objetivo de este caso de uso es comprobar la factibilidad de la solución (y dimensionar su sobrecarga) para la ejecución de aplicaciones paralelas haciendo uso de los beneficios de un entorno Grid sin necesidad de modificar la aplicación. Esta solución es un caso particular del caso de redes. Aquí las máquinas están conectadas bajo un nodo master en una red propia, y los requerimientos de performance son estrictos debido a la naturaleza de las aplicaciones paralelas.

Para este caso de uso se tomó como modelo una aplicación paralela utilizada para modelar la transmisión sináptica de neuronas. Este trabajo es parte del resultado de las investigaciones que lleva adelante el grupo de Sistemas Complejos de la Universidad Nacional de General Sarmiento[21][22]. El servicio que presta el cluster es una aplicación

que simula una malla de sensores: se envía un valor aleatorio al servicio, y éste genera un evento en la red de sensores devolviendo la matriz resultante.

La paralelización reparte una cantidad adecuada de tareas para que los workers colaboren en el cómputo final de la manera más eficiente posible, enviando los datos parciales que van obteniendo al Master. Este los recolecta y sigue adelante con el procesamiento secuencial, aunque debe esperar por todos los resultados que le enviarán los workers. Para una red de 30 sensores, el tamaño de la matriz resultante es de aproximadamente 2MB calculando la totalidad de las comunicaciones.

El software donde se ejecutaron las distintas pruebas cuenta con la librería paralela MPICH 1.2.7p1, en la instalación física y en las imágenes que se ejecutaron sobre las máquinas virtuales. El sistema operativo difiere en distribución con las imágenes generadas, Fedora Core 6 y CentOS respectivamente, pero el manejo de los recursos de memoria, procesamiento y entrada/salida es similar.

Para la ejecución de la aplicación paralela se realizaron tres tipos de pruebas: la primera realiza la ejecución sin usar virtualización en las máquinas del cluster ni en la red de máquinas, y estas medidas permitirán tener una base para medir la sobrecarga de la virtualización en la red y el procesamiento. La siguiente prueba utiliza máquinas virtuales sobre las físicas pero sin virtualizar la red, de esta manera a través de la diferencia con la prueba anterior se determina cuál es la sobrecarga sólo en el procesamiento de aplicaciones. Por último, además de las máquinas se virtualiza la red, y sobre esta virtualización se comparan los dos métodos de conexión, pudiendo así calcular la carga de comunicación.

Como se muestra en la Figura 28, la sobrecarga del worker, medida únicamente en cómputo, es mínima ya que sólo se hace uso intensivo de CPU; en cambio, en las medidas desde el nodo Master, que incluyen la coordinación e intercambio de datos con los workers, los distintos tipos de virtualización aumentan considerablemente el tiempo de cómputo.

4.3. Performance de la red

Para las mediciones del ancho de banda de la interconexión entre los nodos Grid y clusters se utilizó Netperf, una herramienta que mide consumos de ancho de banda con protocolos TCP y UDP generando tráfico en la conexión. Las mediciones se realizaron cada diez minutos generando datos por un lapso de 10 segs sin interrupción.

En la Figura 29 puede observarse claramente la degradación del ancho de banda entre el uso de la red sin virtualización y la conexión atravesando las distintas capas de virtualización, llegando a un 40% de la performance óptima. Este último tipo de medida utilizando el protocolo SSH solamente será necesario en las conexiones a través de Internet donde la seguridad es indispensable. Para una red local, las mediciones nos dan alrededor de un 20% de sobrecarga.

Si bien la diferencia en la Figura 29 entre las máquinas conectadas con Netcat y SSH es notable vemos que en la Figura 28 esta diferencia no es visible porque al trasmitirse mensajes pequeños la carga de los protocolos de comunicación tiene preponderancia por sobre los datos enviados.

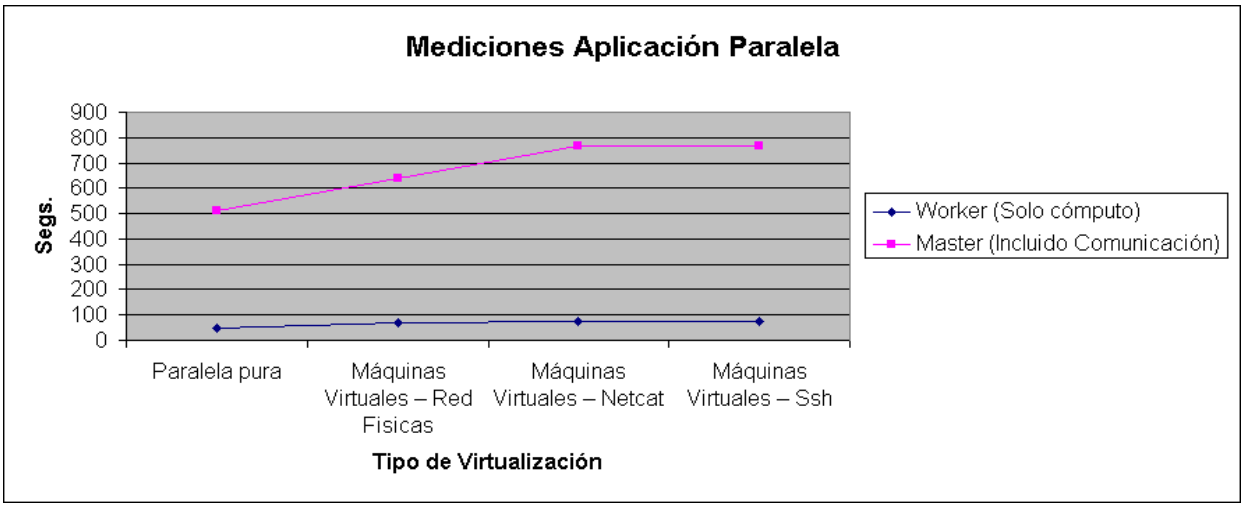


Figura 28. Mediciones de Ancho de Banda de la Aplicación Paralela

5. Resumen

Con este capítulo se muestra que es posible interconectar y utilizar en forma coordinada recursos de cómputo geográficamente dispersos, en forma dinámica y flexible, gracias a un esquema de organización virtual provisto por la infraestructura Grid y a las distintas opciones que provee el uso de máquinas virtuales.

Como aporte del trabajo aparecen aspectos de configuración, acceso y gestión de los recursos.

- En lo relativo a configuración se ha desarrollado un pseudo lenguaje orientado a objetos para el diseño y validación de la especificación lógica.
- Para el acceso se ha desarrollado un modelo de puntos de acceso realizando la redirección de terminales virtuales hasta el usuario final a través de Internet.
- En cuanto a la gestión se ha utilizado la funcionalidad del middleware Grid para permitir la ejecución y transferencia de información entre dominios administrativos diferentes sin la intervención de administradores locales.
- La implementación de casos de uso permitió realizar una experiencia sobre la solución propuesta con distintos tipos de requerimientos. El primero, un laboratorio remoto donde el principal requerimiento es la flexibilidad en la configuración de distintos tipos de escenarios y gran demanda de máquinas virtuales sin requerimientos estrictos de performance. El restante, el uso de una aplicación paralela, donde la configuración de red es simple, con un solo tipo de máquina virtual con requerimientos de performance estrictos.
- La implementación de estos dos casos de uso permitió un análisis detallado destacándose los siguientes aspectos: la configuración del espacio de direcciones propio y separado del espacio de direcciones físico permite independencia y seguridad en el momento de la realización de las prácticas de laboratorio.
- Basándose en plantillas de laboratorios, se pueden instanciar múltiples escenarios

aislados sin incurrir en costos administrativos adicionales para las organizaciones locales, sin limitar los horarios de acceso y prestando servicio a una cantidad mayor de alumnos que usan los laboratorios virtuales.

- La utilización de máquinas virtuales en la implementación de una aplicación paralela facilita la configuración de los clusters con características diferentes como por ejemplo el uso de distintas versiones de librerías paralelas con una mínima intervención del administrador.
- La incorporación del entorno Grid a la configuración de clusters de máquinas permite combinar a demanda gran cantidad de recursos permitiendo un mejor uso de los mismos.

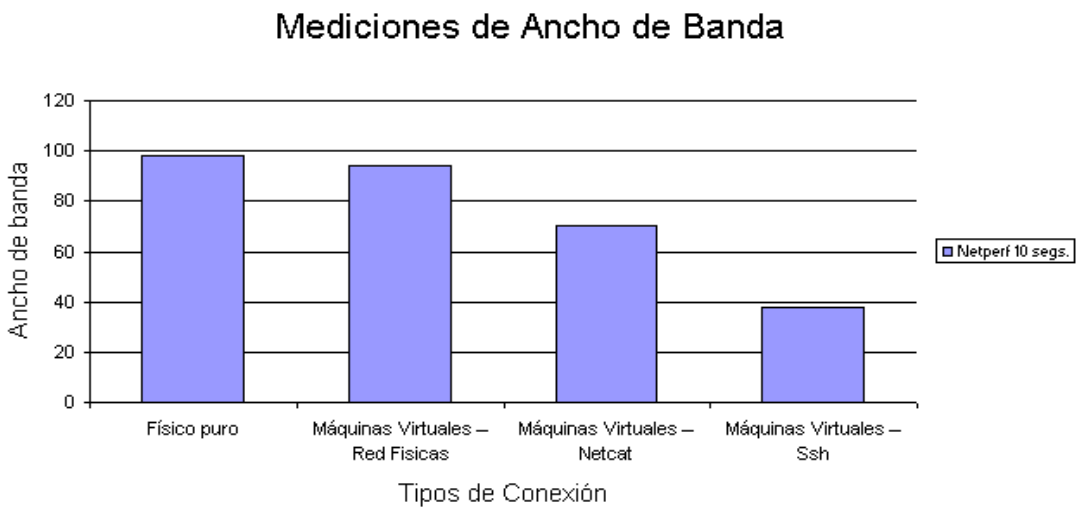


Figura 29. Mediciones de Ancho de Banda

Como se puede comprobar en las pruebas de performance, esta solución no está exenta de una sobrecarga por la virtualización de cómputo de red. La implementación de las máquinas virtuales (por ejemplo, Xen) se ubica por encima del 25% de una solución pura (sin virtualización) y la incorporación de una red virtual agrega un 20% de sobrecarga, lo que debe llevarnos a un análisis más profundo del uso de esta arquitectura ya que el total de sobrecarga es aproximadamente un 50%.

En el caso donde los requerimientos de performance no son específicos y lo importante es la creación de un entorno virtual de trabajo aislado del entorno físico que lo contiene, esta es una solución aceptable que permite hacer un uso eficiente de los recursos disponibles o incorporar elementos de otro modo inaccesibles.

Si la aplicación tuviera restricciones de performance, como en el caso de una aplicación paralela, el análisis del uso de esta aplicación llevaría a una segunda etapa de estudio sobre el comportamiento del uso de la infraestructura (en este caso, la cantidad de cómputo y cantidad de mensajes de coordinación o volumen de transferencias, más la sobrecarga de la virtualización de la red). Por esta razón debería estimarse el compromiso entre aprovechamiento de recursos remotos y consumo de ancho de banda,

es decir, qué cantidad de equipamiento remoto es conveniente incorporar dado un ancho de banda disponible. Este análisis debe contemplar el hecho de que la aplicación paralela no sufriría ninguna modificación al utilizar esta solución; mientras que se debería incurrir en dicho costo si debiera ser "gridificada".

Capítulo 5. Algoritmos de planificación

Para el dinamismo de las organizaciones Grid, los planificadores tradicionales no suelen ser una opción válida. Los planificadores para este tipo de entorno deben contemplar aspectos tales como heterogeneidad de recursos, adaptación dinámica a las tareas y altos costos de comunicación. En este capítulo se presenta una taxonomía de algoritmos de planificación para sistemas distribuidos, se clasifica la arquitectura según la descripción realizada anteriormente y los métodos de planificación según sus principales características. Se realiza una propuesta de selección de equipamiento para la generación dinámica de entornos virtuales basada en la utilización de recursos, y se utiliza información del estado de red para la optimización de los algoritmos seleccionados. Se desarrolla un modelo analítico con uso de teoría de grafos y se realizan estudios de complejidad algorítmica para determinar la factibilidad de la propuesta.

6. Introducción

Para lograr el máximo potencial de los recursos distribuidos, de manera efectiva y eficiente, los algoritmos que utilice el metaplanificador son de fundamental importancia. Desafortunadamente, los algoritmos de planificación de sistemas paralelos y distribuidos tradicionales en general disponen de recursos homogéneos y dedicados, y por lo tanto no logran satisfacer de manera completa los requerimientos de este nuevo entorno [70, 71].

Los algoritmos de planificación han sido estudiados como problemática en los sistemas tradicionales de paralelismo y sistemas distribuidos, como los sistemas de multiprocesadores simétricos (SMP), masivamente paralelos (MPP) o clusters de Workstation (COW). Cada uno de estos algoritmos de planificación tomó en cuenta características propias de los sistemas para lograr máxima eficiencia. En la Tabla 8 se puede observar la evolución de estas características.

Arquitectura	DSM/MPP	COW	GRID
Cronología	Fines de los 70’s	Fines de los 80’s	Mediados de los 90’
Conexión de los sistemas	Bus, switches	LAN, ATM	WAN/Internet
Costo de Interconexión	Muy Bajo / Despreciable	Bajo/ Usualmente no despreciable	Alto/ No despreciable
Heterogeneidad de Conexión	No	Bajo	Alto
Heterogeneidad de Nodos	No	Bajo	Alto
Grupo de Recursos	Predeterminado y Estático	Predeterminado y estático	Indeterminado y dinámico
Política de administración de recursos	Única	Única	Múltiple
Única Imagen del sistema	Si	Si	No

Tabla 8. Características de los sistemas de cómputo paralelos y distribuidos

Si bien se pueden tomar modelos de planificación tradicional, las características propias de los sistemas Grid hace que esta tarea en muchos casos sea difícil de implementar. A continuación se describen en detalle las particularidades de los entornos Grid:

- **Heterogeneidad y Autonomía**

A pesar de que la heterogeneidad no es nueva en los algoritmos de planificación, aún es un gran desafío para su diseño. En los sistemas Grid, debido a que los recursos son distribuidos en múltiples dominios sobre Internet, los nodos de cómputo, datos e interconexión varían entre organizaciones.

En los sistemas paralelos y distribuidos tradicionales, los recursos de computación generalmente están administrados en un punto central. El planificador tiene información completa de las tareas en ejecución y pendientes, y la carga de trabajo de los nodos, además de administrar colas de tareas y el grupo de recursos. De esta manera puede predecir el comportamiento de los recursos y es capaz de asignar tareas de acuerdo a requerimientos de rendimiento. En los sistemas Grid los recursos son autónomos y el planificador no tiene control de los mismos, y no puede influir en las políticas de las organizaciones, por lo que se torna difícil estimar el tiempo de ejecución de cada tarea en los distintos nodos de cómputo.

- **Dinamismo de Rendimiento**

Realizar una planificación depende de las estimaciones de rendimiento que pueden proveer recursos candidatos, especialmente cuando el algoritmo es estático. Los algoritmos Grid trabajan en ambientes dinámicos donde el rendimiento de los recursos disponibles cambia constantemente. El cambio proviene de la autonomía de los recursos de cómputo, y de la competencia de las aplicaciones por estos recursos. En el caso de los sistemas Grid los recursos no son asignados con exclusividad a las aplicaciones Grid.

- **Selección de recurso y Separación de Datos**

En los sistemas tradicionales, el código de ejecución y los datos de entrada/salida se encuentran generalmente en el mismo sitio, o las fuentes de datos y el destino de salida se determinan antes de que la aplicación inicie su ejecución. De esta manera, el costo de la transferencia de datos es mínimo o al menos es constante, y en los algoritmos de planificación rara vez se considera. Los sistemas Grid involucran distintos recursos de cómputo, sitios de almacenamiento y redes de ancho limitado. Los sitios de cómputo son seleccionados por el planificador de acuerdo a modelos de rendimiento y estatus de los recursos. Además, la capacidad de conexión cobra mayor importancia ya que no solo es limitada sino que es compartida por otras tareas que no necesariamente son de la aplicación Grid.

Los desafíos que se han mencionado ponen obstáculos significativos al diseño e implementación eficiente y efectiva de los sistemas de planificación Grid. A pesar de esto, algunos de los conceptos de los planificadores distribuidos tradicionales pueden servir en aspectos puntuales a medida que se desarrollan nuevos planificadores Grid.

7. Taxonomía de algoritmos Grid

En [72] se propone una taxonomía jerárquica de algoritmos de planificación de propósito general en sistemas de cómputo paralelo y distribuido (Figura 30). Debido a que los sistemas Grid son una clase especial de estos sistemas, los algoritmos de planificación Grid se encuentran dentro de un subconjunto de esta taxonomía. A continuación se describe desde la raíz cómo puede ser identificada y se analizan las alternativas que fueron tomadas en consideración para implementar la solución propuesta para esta tesis en la generación de espacios virtuales desde un metaplanificador.

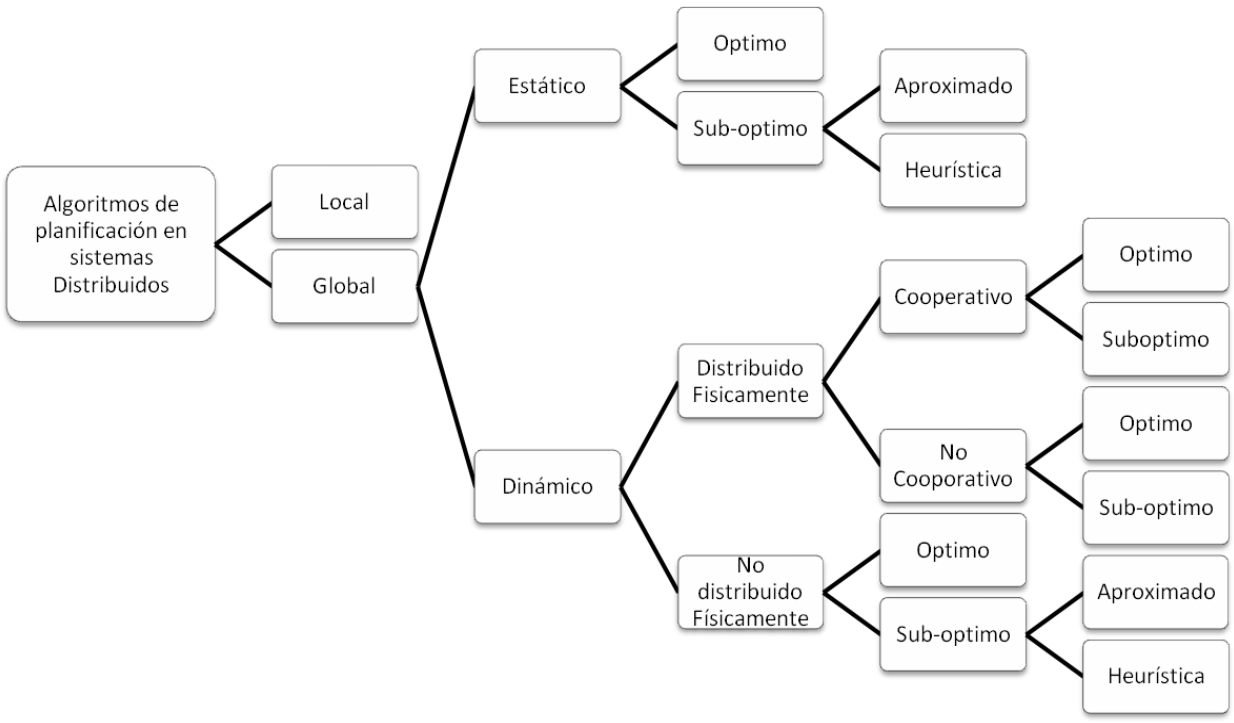


Figura 30. Taxonomía jerarquía de algoritmos de planificación

Locales vs Globales: En el nivel más alto, se encuentra la distinción entre planificación global y local. La planificación local determina como el proceso que reside en un procesador es asignado y ejecutado, mientras que una política de planificación global usa la información del sistema para asignar el proceso a múltiples procesadores y optimizar el sistema en forma general con el objetivo de mejorar el rendimiento, por lo que los sistemas Grid se encuentran en esta segunda categoría.

Estático vs Dinámico: El siguiente nivel en la jerarquía (bajo planificación global) es la elección entre planificación dinámica y estática. La elección indica el tiempo en que se realizan las decisiones de planificación. En el caso de planificación estática, se asume que la información de los recursos así como la de las tareas se encuentra disponible en el momento que se planifica la aplicación. Por el contrario, en el caso de planificación dinámica, la idea básica es realizar la asignación de tareas en el momento de ejecución de la aplicación. Esto es útil cuando es imposible determinar en ejecución, las ramas que puede tomar el código, el número de iteraciones o el modo en que arriban los trabajos en tiempo real.

Optimo vs Sub-óptimo: Cuando toda la información del estado de los recursos y tareas sea conocida, se puede realizar una asignación óptima basada en una función objetivo, como un árbol de expansión mínima o maximización en la utilización de los recursos. Pero debido a la naturaleza NP-Completa de los algoritmos de planificación y la dificultad de hacer suposiciones razonables en los entornos Grid que son condición necesaria para probar la optimalidad de los algoritmos, las líneas de investigación actuales intentan encontrar soluciones sub-optimas, que pueden ser divididas en las siguientes categorías:

Aproximada vs. Heurística: Los algoritmos de aproximación usan modelos computacionales formales, pero en vez de hacer búsquedas sobre la solución completa para una solución óptima, se satisfacen con una solución suficientemente “buena”. En este caso, donde se dispone de una métrica para evaluar la solución, esta técnica se usa para minimizar el tiempo tomado para encontrar una planificación aceptable.

La otra rama de la categoría sub-optima es llamada heurística, esta rama representa la clase de algoritmos que hace suposiciones más realistas sobre un conocimiento de los procesos y conocimiento de carga del sistema. También representa la solución a los problemas de planificación que no pueden dar una solución óptima, sino que sólo requiere el costo más razonable y sólo algunos recursos para realizar las tareas. La evaluación de esta clase de soluciones generalmente se basa en experiencias del mundo real o simulaciones. Los algoritmos con heurísticas se adaptan más a los entornos Grid donde los recursos y aplicaciones son diversos y dinámicos.

Distribuidos vs Centralizados: En los escenarios de planificación dinámicas, la responsabilidad de tomar decisiones de planificación globales son responsabilidad de uno o más planificadores centralizados o compartida por múltiples planificadores distribuidos. En un Grid, pueden haber varias aplicaciones enviadas a planificar o re-planificar simultáneamente. La estrategia centralizada tiene la ventaja de su fácil implementación, pero sufre de problemas de escalabilidad, tolerancia a fallos o la posibilidad de ser el cuello de botella.

Cooperativo vs No-cooperativo: Si se usa un algoritmo de planificación distribuido, el próximo paso es conocer si los nodos que se encuentran involucrados en la planificación de las tareas se encuentran cooperando o son independientes. Si el caso es no-cooperativo, los planificadores actúan como entidades autónomas y toman decisiones según sus propios valores; independientemente de los efectos que estas decisiones pueden tener en el resto del sistema. Algunos ejemplos se dan en planificadores a nivel de aplicación donde los planificadores se encuentran altamente acoplados con aplicaciones especiales tratando de optimizar su funcionamiento.

En el caso cooperativo, es tarea del planificador Grid la responsabilidad de llevar adelante la tarea de planificación, pero junto con el resto de los planificadores trabajan para lograr un objetivo global. Cada planificador Grid tiene la política local de consensuar las decisiones en vez de sólo tener en cuenta el rendimiento local.

En el caso de los entornos Grid la selección por defecto sigue el camino de algoritmos Globales, ya que estos toman la información del sistema para asignar procesos a múltiples procesadores en distintos equipos de computo. En la segunda opción esto depende del algoritmo elegido, en el caso de esta tesis y tomando como referencia el marco de trabajo para la toma de decisiones (capítulo 4), la categorización de los algoritmos que se utilizan es del tipo dinámico, ya que la información se obtiene al arribo de la tarea. La responsabilidad en la toma de decisión para la planificación es compartida entre el metaplanificador y el planificador local, siendo estas características de algoritmos distribuidos y cooperativos.

7.1. Función objetivo

Los dos actores principales en los entornos Grid (los consumidores de los recursos que envían las aplicaciones y los proveedores de recursos que los comparten), generalmente tienen diferentes intenciones cuando se encuentran dentro del entorno Grid. Estas expectativas se presentan en las funciones objetivos. La mayoría de las funciones objetivo en la computación Grid son heredadas de los sistemas de cómputo paralelo y distribuido. Los usuarios Grid se concentran en el rendimiento de las aplicaciones (por ejemplo el costo total de correr una aplicación), mientras que los proveedores de servicio prestan más atención al rendimiento de los recursos que comparten (por ejemplo, el uso en un determinado lapso de tiempo). Así las funciones objetivo pueden ser divididas en centradas en la aplicación o centradas en el recurso.

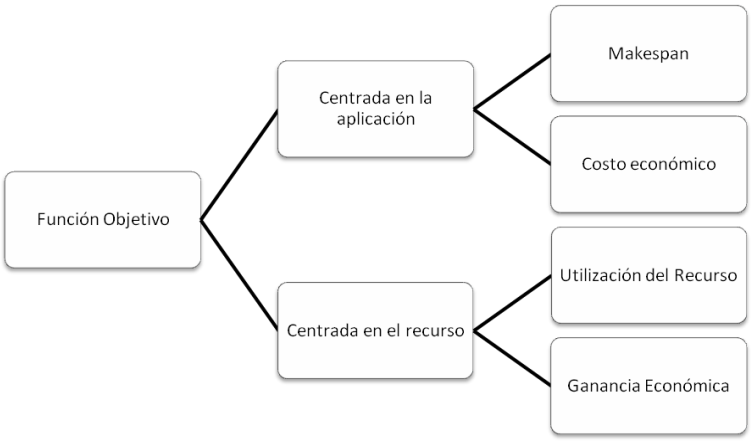


Figura 31. Funciones objetivo

Orientada a Aplicaciones: Los algoritmos que adoptan la función objetivo orientada a las aplicaciones intentan optimizar el rendimiento para cada aplicación. La mayoría de las aplicaciones Grid tiene como objetivo minimizar el tiempo, por ejemplo el *makespan*, que es el tiempo que lleva desde la ejecución de la primera tarea hasta la ejecución de la última. El *makespan* es una medida común para medir algoritmos de planificación. A medida que se introducen modelos económicos, el costo económico que una aplicación necesita pagar por los recursos utilizados también es de interés para los usuarios Grid.

Orientada a Recursos: La función objetivo de algoritmos de planificación orientados a recursos intenta optimizar la performance de los recursos. Sus objetivos están relacionados con la utilización de los recursos, por ejemplo el *throughput* que es la habilidad de los recursos de procesar cierto número de tareas en un periodo de tiempo; o la utilización, que es el porcentaje de tiempo que un recurso está siendo utilizado. Para un recurso multiprocesador, la diferencia en el uso de los procesadores también describe el balance de carga del sistema y disminuye la cantidad de tareas procesadas.

La función objetivo de los algoritmos distribuidos que se desarrollan en esta tesis toma como base el uso de los recursos de cómputo y comunicación disponibles en la red Grid (capítulo 3), por lo que según la calificación se centran en el recurso. Si bien los algoritmos que se describirán en las secciones siguientes, intentan optimizar el uso de los recursos de cómputo y red, la función de optimización puede cambiar de manera sencilla a una ecuación económica siendo esta la función objetivo.

7.2. Planificación adaptativa

Se establece una solución adaptativa al problema de planificación cuando los algoritmos y parámetros para tomar una decisión de planificación cambian significativamente de acuerdo al estado, previo, actual o futuro de los recursos. En la computación Grid la necesidad de contar con planificadores adaptativos proviene de la heterogeneidad de los recursos candidatos, el dinamismo del rendimiento de los recursos y la diversidad de aplicaciones.

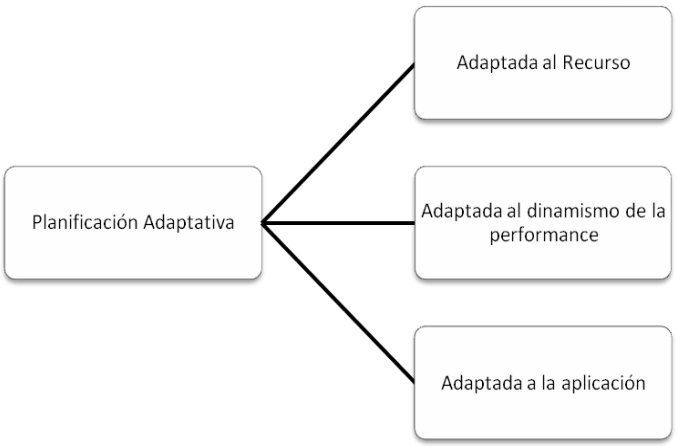


Figura 32. Taxonomía para algoritmos adaptativos en computación Grid

Adaptadas a los recursos: Debido a la heterogeneidad de recursos y diversidad de aplicaciones, descubrir recursos disponibles y seleccionar la aplicación apropiada al subconjunto de los recursos es muy importante para obtener un buen rendimiento o reducir el costo. Por ejemplo, la selección de la ubicación del almacenamiento es afectada por las demoras en la red, o agrupar las aplicaciones dependiendo los anchos de banda de la red o las características de cómputo de los equipos involucrados.

Adaptación al rendimiento: La adaptación al rendimiento de los recursos puede observarse cuando se cambian las políticas de planificación, o también cuando se intenta lograr un balance de carga dependiendo de algún modelo de rendimiento o encontrar un número de recursos adecuado a ser usado.

Adaptación a la aplicación: Para lograr más rendimiento, los planificadores a nivel de aplicación se encuentran integrados a los sistemas y no es fácilmente extensible otros sistemas similares o al entorno. Por tal motivo, este tipo de planificadores es propio de un tipo de aplicación.

Según el modelo de planificación distribuida basado en políticas presentado en el capítulo 4, la interacción entre los planificadores se adapta según los cambios de patrón con la llegada de tareas y capacidad de cómputo de los equipos disponibles. Por lo tanto, según la clasificación que aquí se describe se realiza una planificación adaptativa basada en la adaptación de los recursos.

7.3. Dependencia de las tareas

Cuando se toma en consideración la relación entre las tareas se presentan dos alternativas. La primera, es que las tareas puedan ejecutarse de forma independiente unas de otras y la segunda que posean un orden de dependencia entre ellas. La dependencia tiene un impacto crucial en el diseño de algoritmos de planificación.

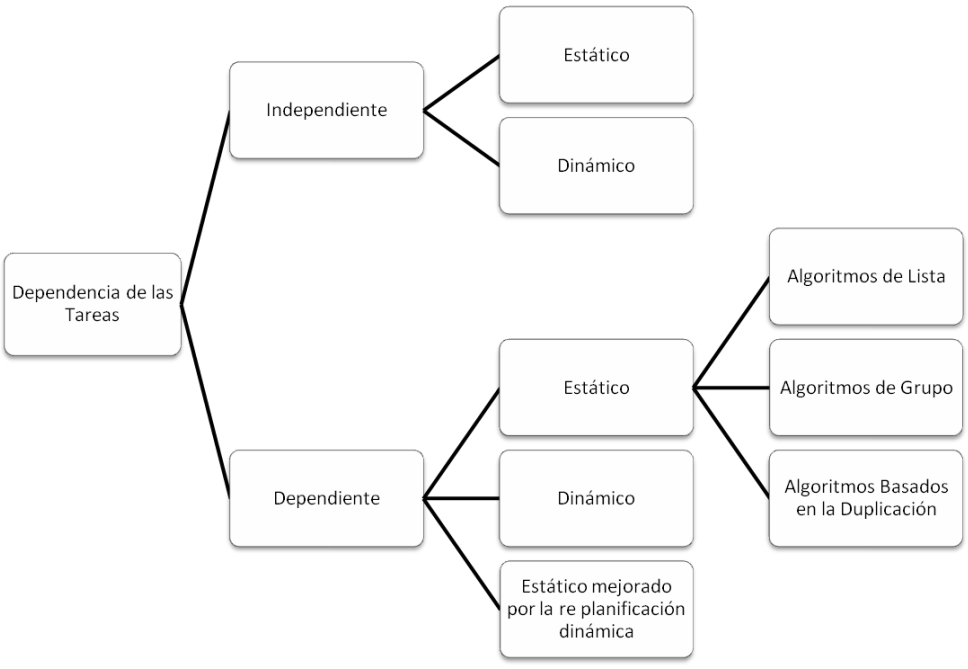


Figura 33. Taxonomía por dependencias de tareas.

Cuando las tareas que componen un trabajo tienen orden de precedencia, se aplica un modelo común llamado grafo directo acíclico (DAG), en el que un nodo representa una tarea y la arista dirigida denota un orden de precedencia entre dos vértices. En algunos casos los pesos pueden ser agregados como costo computacional o costos de comunicación respectivamente. Como la infraestructura Grid madura cada vez más y se hace más compleja, se prevén cada vez más herramientas de soporte de flujo que hacen uso de estos conceptos.

Las aplicaciones que se tienen en consideración para el desarrollo de esta tesis, son las que cumplen con el modelo *master-worker*, por lo que la relación entre las tareas que ejecutan el cómputo es nula. Su asignación es dinámica ya que se relaciona con la carga que tienen en el momento de arribo los recursos seleccionados.

7.4. Métodos no tradicionales

El entorno Grid es el tipo de sistema que se compone de gran número de proveedores de recursos autónomos, ejecutándose en forma concurrente, cambiando dinámicamente e interactuando entre ellos. En la naturaleza y en la sociedad, existen diversos sistemas con las mismas características. Por lo que nuevas metodologías como los sistemas económicos pueden ser incorporados. Algunos de estos conceptos se describen a continuación.

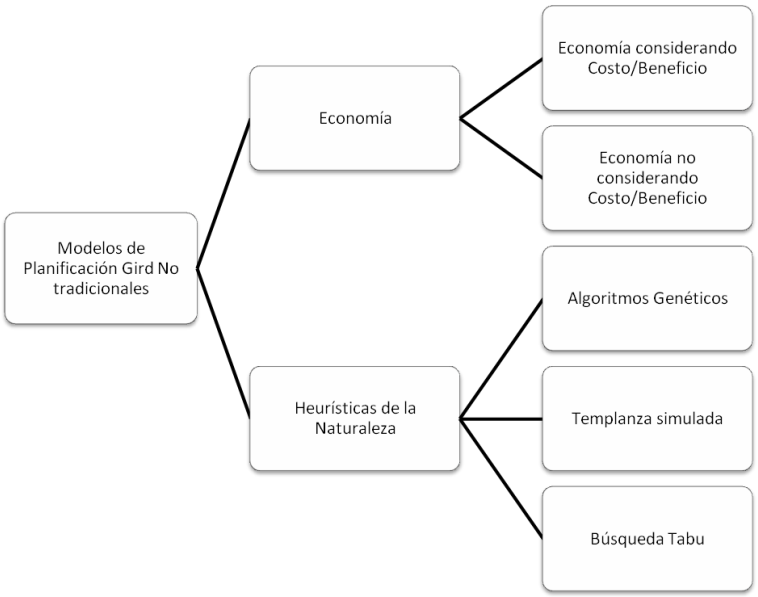


Figura 34. Taxonomía de métodos no tradicionales

El uso de métodos económicos en planificación Grid requiere la interacción de procesos entre proveedores y consumidores de recursos, de manera análoga a la que funcionan los comportamientos de Mercado, Buyya [73] presenta algunos modelos económicos que pueden ser aplicados en entornos Grid como modelos de mercados de productos, modelos de contrato o modelos de compras.

Algunas de las características comunes de las heurísticas de la naturaleza son relacionadas con el no determinismo y la presencia de estructuras paralelas así como

también la adaptabilidad. Las heurísticas clásicas son los algoritmos Genéticos (GA), *Simulated Annealing* (SA) y *Tabu Search* (TS), y otras que son combinaciones de los tres tipos de algoritmo.

En los algoritmos analizados para la implementación de esta tesis se encuentran algunos no tradicionales como pueden ser los que utilizan heurísticas de la naturaleza. Estas heurísticas fueron seleccionadas debido a que en la mayoría de los casos tienden a encontrar soluciones cuasi optimas en un lapso de tiempo razonablemente corto, siendo estos requerimientos fundamentales para determinar la factibilidad del funcionamiento a un algoritmo de planificación.

8. Selección de equipos de cómputo

La necesidad de encontrar un conjunto de máquinas físicas con características definidas, en un tiempo razonablemente corto y que además sea una solución aceptable no es un problema trivial. Recorrer el espacio de búsqueda según alguna variable que puede ser costo o rendimiento y con todas las combinaciones de equipamiento posible no es una solución válida para la planificación, por lo que se trató de optimizar dos parámetros: el primero, la rapidez en la obtención de la solución, y el segundo lo buena que puede ser la solución con respecto al caso óptimo.

Como se ha descrito en la taxonomía, el tipo de estructura de planificación es distribuida cooperativa, ya que la información y toma de decisión es compartida por los componentes de la meta organización y la organización local. Los recursos son el factor clave en la función objetivo, observando que estos trabajen de manera eficiente durante la ejecución de la aplicación y el punto de referencia para adaptar la planificación. En la Figura 35 se describe un ejemplo sobre una red de cómputo de cuatro clusters con distintos tipos de enlace de red.

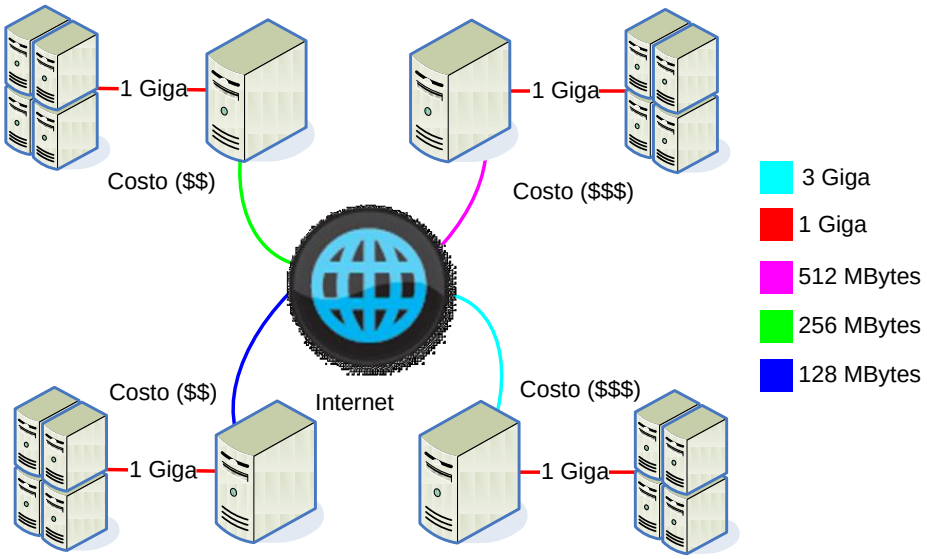


Figura 35. Ejemplo Multicluster en Grid

Una alternativa para trabajar de manera computacional se encuentra en la posibilidad de mapear la red de comunicación y las máquinas disponibles de Grid como un grafo no dirigido. Los equipos disponibles serán representados por nodos, y los vínculos, por aristas. Nodos y aristas tienen un peso o costo asociado. Las aristas reciben menor peso o costo cuanto mayor ancho de banda ofrecen, y a los equipos se les asocia un peso que puede ser función del costo de alquiler del equipamiento, capacidad de procesamiento, o un mix de memoria, almacenamiento, etc. Tomando el ejemplo de la Figura 35, el metaplanificador realiza una consulta según los requerimientos de la aplicación y genera un grafo como el de la Figura 36. Puede observarse que uno de los puntos a tener en cuenta es el tipo de comunicación disponible; si bien en la figura todos los puntos se encuentran conectados, el tipo de comunicación entre dos nodos se realiza con el ancho de banda más limitado, ya que éste será la capacidad disponible en el vínculo.

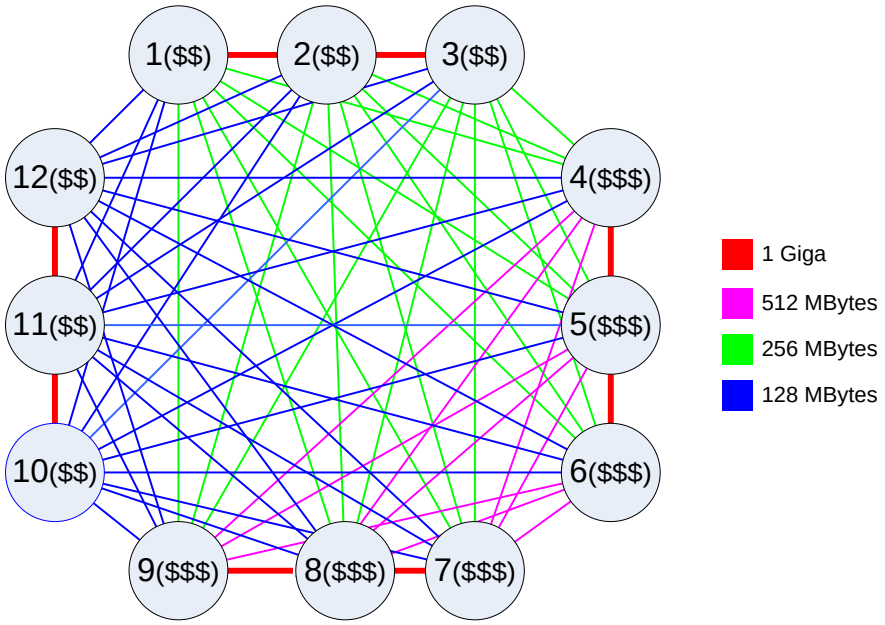


Figura 36. Mapeo de Multicluster a Grafo.

Sobre el grafo se realizan búsquedas basadas en la metodología de *mejora iterativa*. La idea general es comenzar con un solución completa y hacer modificaciones para mejorar su calidad. La forma más sencilla de entender la *mejora iterativa* es imaginar todas las posibles soluciones en una superficie y las diferentes alturas de esa superficie corresponden a la evaluación de la solución de optimización. La idea de mejora iterativa es moverse por ese plano tratando de encontrar los picos más altos, que son las soluciones óptimas al problema. Este tipo de algoritmos siempre tiene en cuenta solo el estado actual, y no lleva información más allá de sus vecinos inmediatos. Los algoritmos de mejora iterativa se dividen en dos grandes clases: *Hill-Climbing*, que siempre trata de mejorar el estado actual, y *Simulated Anealing* en donde el propósito sigue siendo el mismo pero la solución puede empeorar al menos temporariamente [75].

Hill-Climbing busca el camino de menor costo sin mantener un árbol de búsqueda y solo guarda el estado de su última evaluación, lo que mejora el uso de memoria, y según una función de costo determina como dar el siguiente paso. El problema que tiene el algoritmo

es encontrar la presencia de mínimos locales y mesetas (Figura 37): una vez que llegue a un mínimo local, a pesar de no ser una buena solución se detendrá la búsqueda. De la misma manera que si encontrara mesetas, no sabría por donde continuar y podría seguir un camino equivocado. Se puede observar el pseudocódigo del algoritmo *Hill-Climbing* en el listado Código 7.

```
función HILL-CLIMBING( problema) devuelve un estado de solución
entrada: problema, un problema
staticas: actual, un nodo
          próximo, un nodo
actual ← hace-nodo(Estado-Inicial[problema])
bucle hacer
          próximo ← un valor más alto sucesor del actual
          si valor[próximo] < valor[actual] entonces devolver actual
actual ← próximo
fin
```

Código 7. Pseudocódigo del algoritmo Hill-Climbing

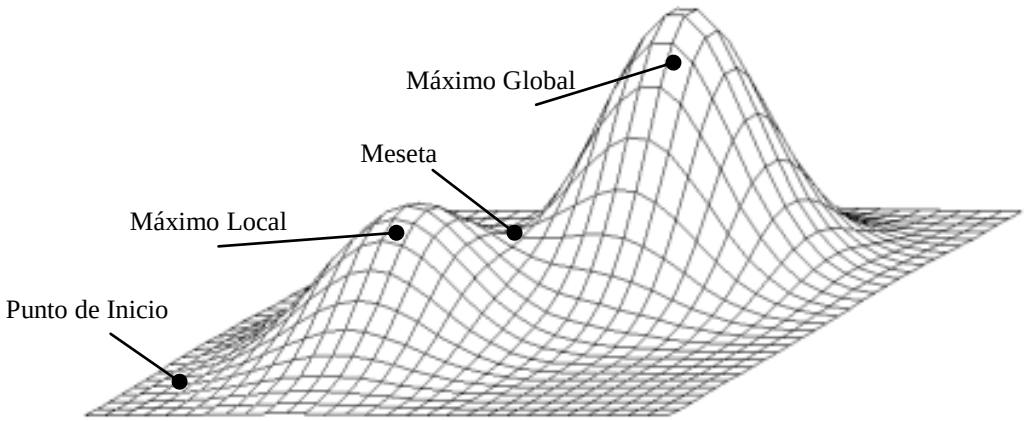


Figura 37. Espacio de todas las posibles soluciones

El algoritmo de templanza simulada (*Simulated Anealing*, SA) es una técnica de búsqueda basada en el proceso de templanza, que es el proceso termal para obtener estados cristalinos de un sólido a baja temperatura. Al comienzo, la temperatura se aumenta para mezclar el sólido. Si la temperatura decrece lentamente las partículas se mezclan localmente, en un estado estable. La teoría de SA postula que si la temperatura bajó lo suficientemente lenta, el sólido puede obtener un equilibrio, que es el estado óptimo. Por analogía, el equilibrio termal es un mapeo a un óptimo, la temperatura es el tiempo total del mapeo (función de costo) y el cambio de temperatura es el proceso de cambio. Si la próxima temperatura es mayor, lo que significa un mapeo peor, el próximo estado es aceptado con cierta probabilidad. Esto se debe a que un estado “peor” provee una manera de escapar a un estado local óptimo que ocurre frecuentemente en una búsqueda local. El pseudocódigo puede observarse en el listado Código 8.

```
función SIMULATED-ANNEALING(problema, planificación) devuelve un estado de solución
entradas: problema, un problema
        planificacion, un mapeo de tiempo a “temperatura”
estática: actual, un nodo
        próximo, un nodo
        T,  una “temperatura” que controla la probabilidad de los pasos
actual← hacer-nodeo(Estado-Inicial [problema])
para t ← 1 a ∞ hacer
        T ← planificar[t]
        si T = 0 entonces retorna actual
        próximo ← un sucesor del actual seleccionado al azar
        ΔE← valor[próximo] - valor[actual]
        si ΔE > 0 entonces actual← próximo
        de otra manera actual← próximo solo con probabilidad e^(ΔE/T)
```

Código 8. Pseudocódigo del algoritmo Simulated-Annealing

En [74] se puede observar que pruebas realizadas con algoritmos de planificación basados en SA tienen mejor tiempo de ejecución estimado que los provistos por los algoritmos voraces (*greedy*). Los algoritmos SA pueden evitar algunos mínimos locales que no pueden ser detectados por otros algoritmos. Cuando se obtiene la media de SA sobre una muestra mayor, la mejora sobre el tiempo total de ejecución es la misma.

Durante el desarrollo de la tesis se probaron distintas alternativas para mejorar la eficiencia de este tipo de algoritmos. Una de ellas es la de mejorar el rendimiento realizando varios inicios en forma aleatoria. Este método se denomina *Hill-Climbing* con *k-recomienzos,* donde *k* es la cantidad de nuevos inicios. Otra variante al algoritmo es perturbar el vector de sentido de acuerdo a una función aleatoria, variante del algoritmo *Simulated Annealing*.

Sin embargo para este tipo de soluciones se desestimaron los movimientos aleatorios debido a las características singulares del problema. Puede observarse que al inicio de la ejecución, geográficamente (agrupando países o zonas) se producen conjuntos con mejor calidad o ancho de banda de comunicación que otras, y luego esto varía según la ejecución de las tareas sobre el entorno Grid (ver Capítulo 2). Esto se traduce en distanciamientos o altos costos en las aristas del grafo por conexiones con ancho de banda pobres, llevando a un particionamiento o zonificación del grafo. En estas particiones se pueden encontrar mínimos locales, ya que los bajos enlaces se pueden asociar a los valles en el espacio de solución y alguna de estas particiones puede coincidir con el mínimo global. Debido a este análisis se optimizó la definición del algoritmo, y si bien se podría seguir interpretando como *k-recomienzos* la idea es que estos no sean aleatorios sino que se utilicen las particiones del grafo. Esta solución no asegura el máximo global pero brinda una buena aproximación a la misma con alto grado de confiabilidad. La interpretación gráfica puede observarse en la Figura 38.

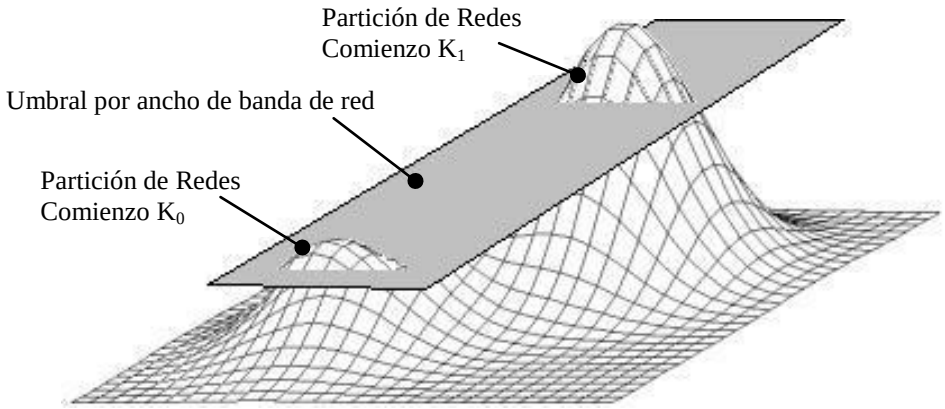


Figura 38. Espacio de todas las posibles soluciones con umbral por enlace de red

8.1. Árboles de Expansión mínima

Luego de realizar el análisis sobre el particionamiento del grafo para el problema de los recomienzos del *Hill-Climbing* optimizando la solución, la necesidad de encontrar una forma de obtener el particionamiento de manera eficiente llevó a la búsqueda de otro algoritmo para tratar el problema. Una alternativa válida es la ejecución de árboles de expansión mínimos (*Minimum Spanning Trees*, MST). Un árbol de expansión mínimo es un subgrafo al que pertenecen todos los vértices del grafo, donde la sumatoria de pesos de las aristas es menor o igual al peso de cualquier otro subgrafo de dicho grafo.

Un subgrafo de un grafo $G = (V, E)$ es cualquier grafo $G' = (V', E')$ tal que $V' \subseteq V$ y $E' \subseteq E$. En particular se considera un grafo no dirigido conexo y sus subgrafos mínimos. El subgrafo mínimo de un grafo conectado se llama árbol de expansión mínima.

Definición (Árbol de expansión mínima): Considere un grafo conexo no dirigido $G = (\mathcal{V}, \varepsilon)$. Un árbol de expansión mínima es un árbol de G, que es un subgrafo, donde $T = (\mathcal{V}', \varepsilon')$, con las siguientes propiedades:

1. $\mathcal{V}' = \mathcal{V}$
2. T es conexo
3. T es aciclico.

Existen varios algoritmos para encontrar el MST de un grafo. El algoritmo de Prim encuentra el árbol de expansión mínima de un grafo $G = (\mathcal{V}, \varepsilon)$ conexo, no dirigido y con peso en sus aristas. El algoritmo construye un árbol de expansión mínima de un grafo seleccionando la aristas una a una y agregando esas aristas al árbol de expansión.

El algoritmo de Prim es una variación del algoritmo de Dijkstra. Para construir un árbol de expansión, el algoritmo construye una secuencia de árboles de expansión $T_0, T_1, \ldots T_{|\mathcal{V}|-1}$, donde cada uno es un subgrafo de G. El algoritmo comienza con un árbol que contiene un vértice seleccionado, $\mathcal{V}_n \in \mathcal{V}$. Esto es, $T_0 = \{\mathcal{V}_n, \varnothing\}$

Dado $T_i = \{ V_i, \mathcal{E}_i \}$, se obtiene el próximo árbol como se describe a continuación. Considere el conjunto de aristas dado por:

$$\kappa_i = \bigcup_{\upsilon \in Vi} A(\upsilon) - \bigcup_{\upsilon \in Vi} B(\upsilon)$$

El conjunto κ_i contiene todas las aristas $\{\upsilon, \omega\}$ tal que exactamente una de las dos υ o ω este en V_i pero no ambas. Seleccionar la arista $\{\upsilon, \omega\} \in \kappa_i$ con el menor peso.

$$C\{\upsilon, \omega\} = \min_{\{\upsilon, \omega\} \in \kappa i} C\{\upsilon', \omega'\}$$

Entonces $T_{i+1} = \{ V_{i+1}, \mathcal{E}_{i+1} \}$, donde $V_{i+1} = V_i \cup \{\upsilon\}$ y $\mathcal{E}_{i+1} = \mathcal{E} \cup \{\{\upsilon, \omega\}\}$. Después de $|V| -1$ pasos se tiene $T_{|V|-1}$ que es el árbol de expansión mínima de G.

Al igual que el algoritmo de *Prim*, el algoritmo de Kruscal construye el árbol de expansión mínima de un grafo agregando las aristas del árbol una a una. En cualquier momento de la ejecución el algoritmo de *Prim* forma exactamente un árbol, pero el algoritmo de *Kruscal* forma un conjunto de árboles. El concepto del algoritmo de *Kruscal* es simple, las aristas son seleccionadas y agregadas al árbol de expansión de manera ordenada en base al peso y cada una es agregada si y solo si no genera ciclos en el grafo.

Se considera un grafo no dirigido $G = \{V, \mathcal{E}\}$. Se puede ver el conjunto de vértices V, como un *conjunto universal* y al conjunto de las aristas, $\mathcal{E}$ como la definición de una *relación de equivalencia* sobre el universo V. En general, una relación de equivalencia particiona un universo en un conjunto de clases de equivalencia. Si el grafo es conexo, solo hay una clase de equivalencia, todos los elementos del universo son equivalentes. Por lo tanto el árbol de expansión es un conjunto mínimo de equivalencias que resultan en una clase de equivalencia.

El algoritmo de *Kruscal* ejecuta $P_0, P_1, \ldots, P_{|V-1|}$, una secuencia de particiones del conjunto de vértices V. La partición inicial consiste en $|V|$ conjuntos de tamaño uno:

$$P_0 = \{\{V_1\},\{V_2\},\ldots,\{V_{|V|}\}\}$$

Cada elemento siguiente de la secuencia se obtiene de su predecesor por la unión de dos elementos de la partición. Por lo que, P_i tiene la forma de

$$P_i = \{ S_0^i, S_1^i, \ldots, S^i_{|V|-1-i} \} \qquad \text{para } 0 \leq i \leq |V| -1$$

Para construir la secuencia de arista en $\mathcal{E}$ se consideran una a una, ordenadas por su peso. Se supone que se ha computado la secuencia hasta P_i, donde se tiene el conjunto de vertices, la próxima arista a ser considerada es $\{\upsilon, \omega\}$. Si υ y ω son ambos miembros

del mismo elemento de la partición P_i, entonces las aristas forman un ciclo, y no son parte del árbol de expansión mínima.

Por otro lado, si υ y ω son miembros de dos elementos de la partición P_i (por ejemplo S_K^i y S_L^i respectivamente), entonces $\{\upsilon, \omega\}$ deben ser una arista en el árbol de expansión mínima. En este caso se calcula P_{i+1} juntando S_K^i y S_L^i. Esto es, se reemplaza S_K^i y S_L^i en P_i por la unión S_K^i U S_L^i.

La Figura 39 ilustra cómo el algoritmo de Kruscal determina el árbol de expansión mínima en un grafo. El algoritmo calcula la siguiente secuencia de particiones:

$P_0 = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}\}$
$P_1 = \{\{a, d\}, \{b\}, \{c\}, \{e\}, \{f\}\}$
$P_2 = \{\{a, d\}, \{b\}, \{c\}, \{e, f\}\}$
$P_3 = \{\{a, d\}, \{b\}, \{c, e, f\}\}$
$P_4 = \{\{a, d, c, e, f\}, \{b\}\}$
$P_5 = \{\{a, d, c, e, f, b\}\}$

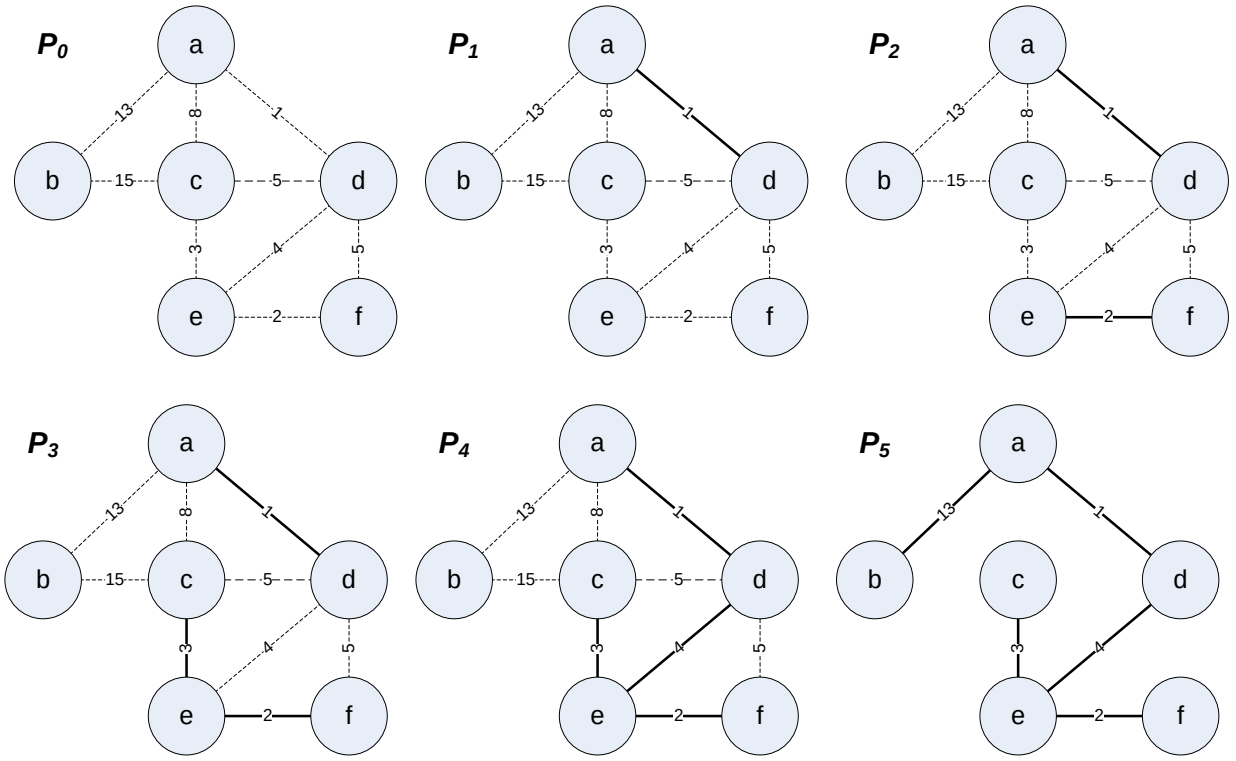


Figura 39. Árbol de expansión mínima según Kruscal

El algoritmo que más se adapta a la solución del problema en tratamiento es el de Kruskal. Este algoritmo ordena las aristas del grafo de menor a mayor peso, va agregando de a una y chequeando que no se formen ciclos en el grafo hasta encontrar la solución. Para esta tesis la elección de este algoritmo se justifica por su metodología de construcción del grafo, ya que inicia su proceso suponiendo tantas particiones como vértices tenga el grafo. La adaptación que se realizó del problema es la suposición de que cada vértice corresponde a una máquina. Cada vez que el algoritmo agrega una arista se

chequea que todas las particiones tengan al menos la cantidad de nodos que requiere la aplicación. Cuando esta condición sea verdadera se tendrá la cantidad de particiones suficientes como para encontrar en forma autocontenida a cada partición una solución para ejecutar el algoritmo *Hill-Climbing con K-recomienzos*.

En la Figura 40 se puede observar el particionamiento según el algoritmo de *Kruskal* sobre el grafo de la Figura 36. La serie de figuras muestra una solución de *Hill-Climbing* con K recomienzos con el conjunto de pasos completo. La función de optimización que se utilizó fue el conjunto de máquinas más rápidas y en la segunda alternativa se optimizó el costo. En cada caso el algoritmo de *Hill-Climbing* realizó 3 reintentos (K_0, K_1, K_2), como se observa en la segunda parte de la Figura 40. El siguiente paso fue buscar la máquina más rápida por partición; cada uno de estos nodos sirve de inicio para el algoritmo de *Hill-Climbing*. Finalmente, luego de aplicar el algoritmo de expansión mínima, las dos alternativas resultantes se pueden observar en la Figura 41, donde el máximo o mínimo local encontrados coinciden con el global.

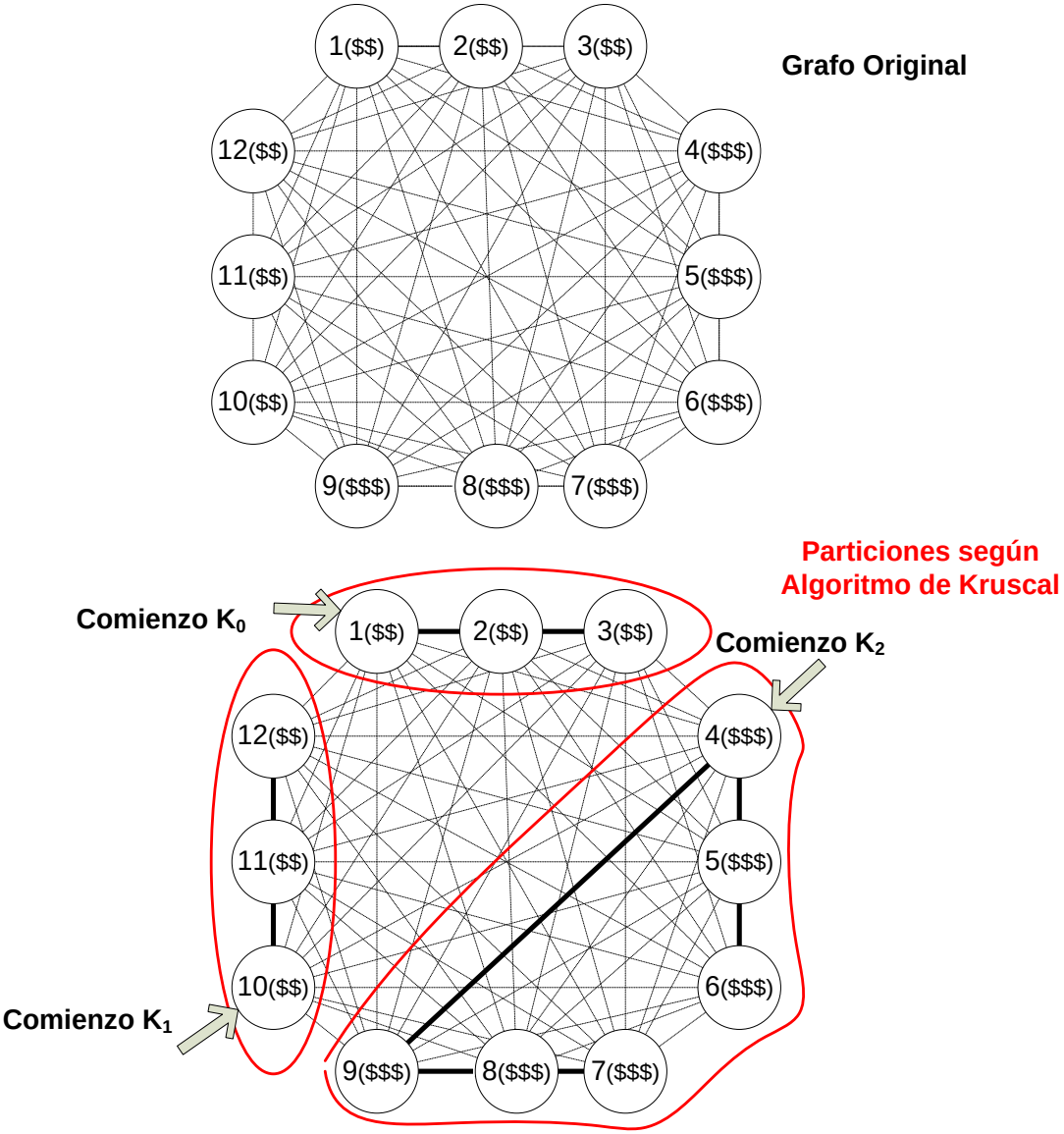


Figura 40. Hill-Climbing con 3 reintentos (K_0, K_1, K_2)

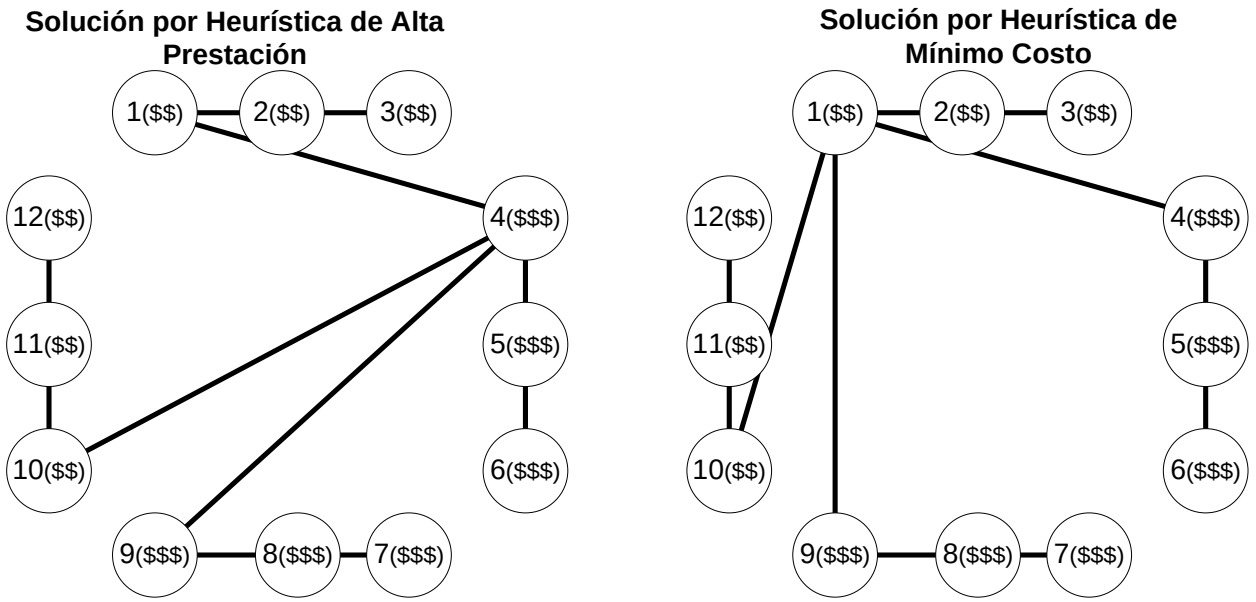


Figura 41. Solución por Heurística de Alta Prestación

8.2. Análisis de complejidad algorítmica

Se realizó un análisis de complejidad del algoritmo de Kruscal debido a la gran cantidad de aristas que se generan para un número pequeño de máquinas.

```
1.  class Algorithms (object):
2.
3.          @staticmethod
4.          def KruscalsAlgorithm(g):
5.                  n = g.numberOfVertices
6.                  reusult = GraphAsLists(n)
7.                  for v in xrange(n):
8.                          result.addVertex(v)
9.                  queue = BinaryHeap(g.numberOfEdges)
10.                 for e in g.edges:
11.                         weight = e.weight
12.                         queue.enqueue(Assiciation(weight, e))
13.                 partition = PartitionAsForest(n)
14.                 while not queue.isEmpty and partition.count >1:
15.                         assoc = queue.value
16.                         e = assoc.value
17.                         n0 = e.v0.number
18.                         n1 = e.v1.number
19.                         s = partition.find(n0)
20.                         t = partition.find(n1)
21.                         if s != t:
22.                                 partition.join(s, t)
23.                                 result.addEdge(n0,n1)
24.                 return result
```

Código 9. Algoritmo de Kruscal.

El algoritmo de Kruskal comienza por crear un grafo para mantener el resultado del árbol de expansión mínima (líneas 5 a 7). Como el árbol de expansión mínima es un grafo con poca cantidad de aristas, para representarlo de manera eficiente, se utiliza una clase que lo almacena como una lista. El grafo contiene |V| vértices pero ninguna arista, por lo que el tiempo de ejecución de las líneas 5 a 7 es de O(|V|).

Luego, todas las aristas del grafo de entrada son insertas una a una en la cola de prioridad (líneas 8 a 11). Como hay |*E*| aristas, el peor caso de ejecución para una sola inserción es de *O(log |E|)*. Por lo que, el peor caso de tiempo de ejecución para inicializar la cola de prioridad es de *O(|V| + |E| log |E|)* cuando se usa lista de adyacencias.

El bucle principal comprende de las líneas 13 a la 22. Este buque se ejecuta al menos |*E*| veces. En cada iteración del bucle, se remueve una de las aristas de la cola de prioridad (líneas 14 y 15). En el peor caso toma *O(log |E|)* tiempo en realizarse.

Entonces, se realizan dos operaciones de búsqueda para determinar los elementos que contienen las dos puntas de una arista dada (líneas 16 a 19). Debido a que la partición contiene al menos |*V*| elementos, el tiempo de ejecución para la operación de búsqueda es de *O(log |V|)*. Si los dos elementos de la partición son distintos, entonces la arista se agrega al árbol y se ejecuta una operación de unión para unir los dos elementos de la partición (líneas 20 a 22). La operación de unión también requiere *O(log|V|)* tiempo en el peor caso. Por lo que, el tiempo total de ejecución del bucle principal es de:

$$O(|E| \log |E| + |E| \log |V|)$$

Y el peor caso del algoritmo de Kruscal es

$$O(|V| + |E| \log |E| + |E| \log |V|) \rightarrow max\ (O(|V|), O(|E| \log |E|), O(|E| \log |V|))$$

Luego, el orden del algoritmo de Kruscal es ***O(n log n)***

9. Resumen

En este capítulo se han presentado características especiales de los algoritmos de planificación Grid, como la capacidad de trabajar en entornos heterogéneos y dinámicos. Se ha realizado una propuesta a través de métodos analíticos sobre cómo puede implementarse, partiendo desde un requerimiento de cómputo, un espacio virtual de manera casi-óptima en base a la información del entorno computacional. Anteriormente en esta tesis se ha planteado el marco de trabajo y la infraestructura del entorno de planificación; pero restaba aun agregar la “inteligencia” para la generación y gestión del entorno en forma autonómica. La categorización a través de una taxonomía de algoritmos distribuidos, y la prueba del método analítico para selección de equipamiento ayudan a definir esta “inteligencia”. Siendo este desarrollo el aporte realizado en este capítulo.

Según una taxonomía de algoritmos distribuidos, el tipo de estructura de planificación es distribuida-cooperativa, ya que la información y toma de decisión es compartida por los componentes de la meta organización y la organización local. Los recursos son el factor clave en la función objetivo observando que estos trabajen de manera eficiente durante la ejecución de la aplicación y el punto de referencia para adaptar la planificación. Como

caso especial y para acotar el alcance del tipo de aplicaciones, se seleccionaron aquellas más comunes en entornos Grid, donde las tareas involucradas son independientes unas de otras.

Una vez que las alternativas de implementación fueron planteadas, se evaluaron distintas posibilidades a través de modelos analíticos y teoría de grafos. Buscando el balance entre efectividad y rapidez se seleccionó el algoritmo *Hill-Climbing* con *K-recomienzos*. En este caso con una sustancial modificación sobre el algoritmo clásico, donde las *K* opciones no son aleatorias, sino una deducción en base a las características particulares del problema utilizando información de enlaces de comunicación. Este aporte se pudo realizar en base a las experiencias llevadas a cabo a lo largo del doctorado trabajando sobre entornos reales distribuidos geográficamente. Por último se realizaron pruebas de complejidad algorítmica sobre el algoritmo MST de Kruscal, ya que éste era el algoritmo más complejo de los propuestos, con un orden de *O(n log n).*

Capítulo 6. Adaptación Dinámica

Una aplicación paralela en un entorno multi-cluster está limitada por el rendimiento de las máquinas en el cluster (*compute-bound*) o por la capacidad de la red (*communication bound*). El rendimiento máximo que puede lograr la aplicación se alcanza por la aplicación en un cluster determinado si el limite esta dado por la capacidad de cálculo. Si la aplicación está limitada por la capacidad de la red, las máquinas estarán ociosas esperando por transferencia de entrada/salida. En este capítulo se describen y se implementan las técnicas utilizadas para optimizar el uso de multi-clusters en Grid, a través de algoritmos de planificación y la utilización de máquinas virtuales. Asimismo se propone una optimización sobre la adaptación dinámica del espacio virtual, con el fin de mejorar el aprovechamiento del ancho de banda y los recursos ociosos disponibles. Se utilizan características de migración en línea y planificación dinámica en la gestión de máquinas virtuales para lograr transformar entornos limitados por la red, a entornos limitados por el cómputo, alcanzando altos rendimientos en el uso del equipamiento disponible. Para lograr un espacio de pruebas configurable, repetible y rápido las ejecuciones de los algoritmos y heurísticas se desarrollaron sobre un simulador, implementado como parte del trabajo doctoral.

10. Introducción

Una aplicación paralela en un entorno multi-cluster está limitada por el rendimiento de las máquinas en el cluster (*compute-bound*) o por la capacidad de la red (*communication bound*). El rendimiento máximo que puede lograr la aplicación se alcanza por la aplicación en un cluster determinado si el limite esta dado por la capacidad de cálculo. Por lo que se puede deducir que si la cantidad de equipos disponibles en el cluster se reduce, la trasferencia de información disminuirá en forma proporcional, pudiendo lograrse un balance entre cómputo y comunicación que satisfaga los umbrales requeridos

de rendimiento. El cluster no estará utilizando todo su equipamiento, pero las máquinas que participen estarán trabajando con su máximo de eficiencia.

En un entorno Grid, existen gran cantidad de combinaciones para la asignación de clusters. Si la aplicación se iniciara en distintos clusters, es decir, se seleccionan diferentes clusters cabecera del conjunto de clusters, el resultado de la ejecución y los tiempos serán distintos en cada caso. Como se ha descrito en el capítulo anterior, con algoritmos adecuados, el espacio de búsqueda se minimiza, y la mejor solución se encuentra de la combinación de equipos, logrando el mayor poder de cómputo. Para hacer esto posible, se realiza una búsqueda analítica sobre el espacio de todas las posibles soluciones mapeadas en un grafo. Esto limitará el número de máquinas de cada cluster de tal manera que el rendimiento pueda ser alcanzado de manera eficiente.

No solo se deben evaluar la comunicación y el poder de cómputo, sino también los equipos cabecera en el cluster. Se define como cabecera por ser el cluster que genera las tareas y las envía a los nodos *worker* de su propio cluster y a los demás. Si la capacidad de conexión o ancho de banda del equipo es menor que la suma de los clusters que se comunican con él, los equipos de los clusters trabajadores estarán ociosos por más que sus redes tengan la capacidad de comunicación suficiente para transferir los datos resultantes de su cómputo. En este caso, se deberá contemplar el fraccionamiento del ancho de banda basado en la capacidad de cómputo del cluster cabecera y calcular el balance computo/comunicación. En la Figura 42, se puede observar al Cluster A como cabecera, con un ancho de banda disponible de 1 Gigabit; el Cluster C también dispone del mismo ancho de banda pero como la conexión es compartida con el Cluster B solo puede usar un 50%. En el caso de esta tesis, el ancho de banda se divide en forma proporcional al poder de cómputo de los clusters, es decir, si el poder de cómputo de los Cluster B y C fuesen iguales la conexión del cluster cabecera se repartiría un 50% para cada uno.

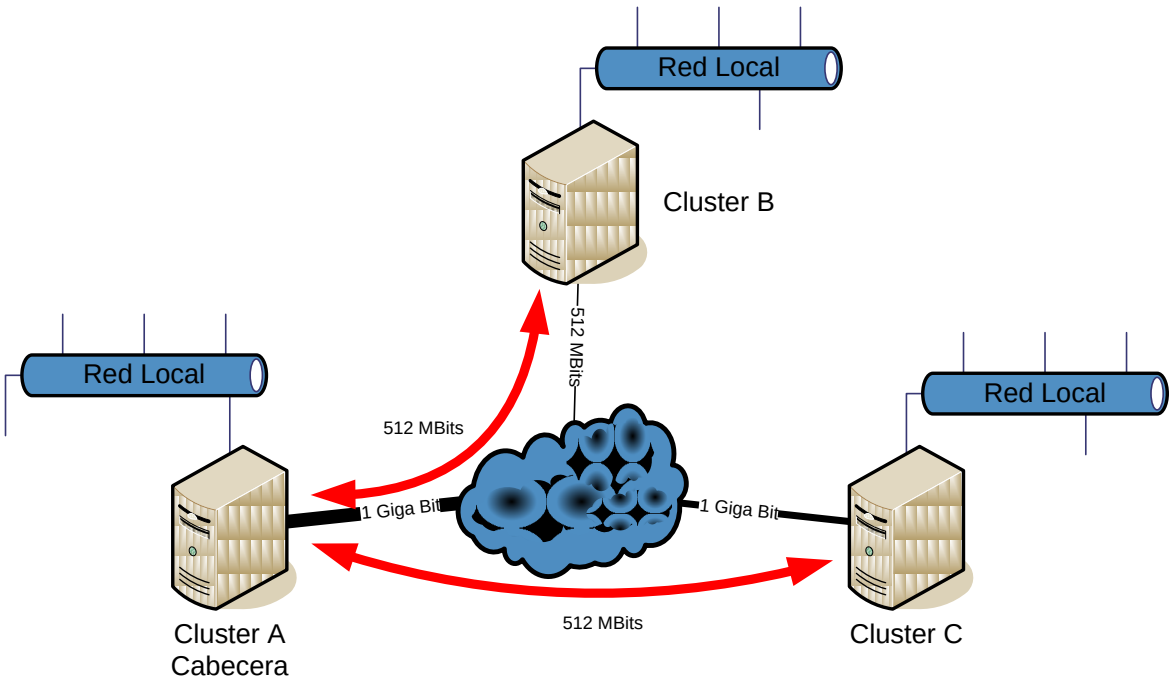


Figura 42. Multiplexión de conexiones de Internet al cluster cabecera

Los algoritmos propuestos en los capítulos anteriores hacen foco en el mantenimiento del rendimiento de los equipos en forma individual, esto es esperable en el uso de clusters, sin embargo también es deseable el aprovechamiento eficiente de todas sus máquinas. Si un cluster puede ejecutar tareas con requerimientos de cómputo limitados pero gran necesidad de transferencia de datos, entonces el enlace de red siempre limitará el poder de cómputo de las pocas máquinas necesarias para realizar la tarea. Un método para mejorar este tipo de problemas es limitar aún más el poder de cómputo y liberar el ancho de banda. Cuando una tarea con menos requerimientos de comunicación arribe, puede ser ejecutada en las máquinas restantes, aprovechando de esta manera el poder de cómputo remanente.

Las propuestas de este capítulo son construir, según métodos analíticos, y analizar clusters sobre máquinas virtuales, asignando ancho de banda a demanda en forma dinámica según el arribo de tareas. En la primera parte se simulan los algoritmos del capítulo anterior y se analiza cuál es el impacto sobre el tiempo total de ejecución. En la segunda parte del capítulo, se optimiza la ejecución adaptando dinámicamente la asignación de equipos. Por ejemplo, cuando una nueva tarea arriba con menores requerimientos de cómputo, las máquinas que se encuentra trabajando en el cluster en ese momento se migran, sin interrumpir su ejecución. Cuando dos o más máquinas virtuales se encuentran en ejecución sobre el mismo procesador físico, el software de virtualización asigna el uso de los procesadores en forma equitativa, por lo que el resultado de poder de cómputo por máquinas es menor así como también la cantidad de datos que envían. El tiempo de finalización de ambos tipos de tareas, las tareas en ejecución y la que recién arriba son conocidos, por lo que el meta-planificador puede calcular el tiempo que se gana por la migración de máquinas virtuales. Si la ganancia en tiempo al incorporar mayor cantidad de máquinas a un nuevo requerimiento es mayor a la perdida de la tarea en ejecución, entonces el meta-planificador limitará los recursos de la tarea en ejecución migrando sus procesos a otras máquinas, dejando así procesadores libres. Si la nueva tarea tiene menores requerimientos de comunicación que las tareas migradas, no solo los nodos físicos que se encontraban trabajando, sino también los nodos que se encontraban aislados pueden comenzar a realizar cómputo con la nueva tarea, mejorando los parámetros de uso del cluster.

11. Modelo de Simulación

El problema de la planificación en Grid puede ser investigado de manera experimental o usando técnicas de simulación. Las ventajas de realizarlo de manera experimental planificando aplicaciones reales sobre recursos reales, es que facilita la comparación de múltiples algoritmos. A pesar de esto, los estudios experimentales de algoritmos de planificación no son muy comunes, ya que difícilmente puedan llevarse a cabo la suficiente cantidad de experimentos para obtener resultados significativos. Además, es difícil explorar la disponibilidad de variaciones en los múltiples parámetros de configuración, sin contar con que el entorno Grid es muy dinámico, haciendo complejo poder obtener resultados repetibles. A raíz de estas dificultades con el método experimental, la simulación es el método más aceptable para investigar de manera efectiva los algoritmos de planificación Grid. El método simulado es configurable, repetible y más rápido, por lo que es el elegido en esta tesis. En este trabajo se toman parámetros como la descripción de los clusters intervinientes, los enlaces de red,

características de los equipos (memoria y tipo de procesador) y las tareas que intervienen en la ejecución, donde se caracteriza su tiempo.

11.1. Ingeniería del simulador

El simulador que se ha desarrollado para ejercitar los conceptos planteados en esta tesis consta de cinco componentes principales. A continuación se describe a cada uno de ellos:

- Componente Simulador: En este componente se ha desarrollado la coordinación de todo el simulador. En primer término se genera el entorno de clusters. Se dispone de una línea por cada equipo en la que se define las características de la máquina y tipo de conexión. Se configuran las distintas opciones para la ejecución del simulador, cantidad de tareas, tipo de tasa de arribo, o algoritmo que se desea usar para la atención de las tareas en el cluster. Para la tasa de arribo se han probado tres opciones: Exponencial, Gaussiana y Uniforme.
- *Componente Heurística Clásica:* Los métodos desarrollados en este componente son los distintos algoritmos clásicos para planificación de sistemas Grid:
    - FIFO No Restringido: En este método (también es llamado balance de carga oportunístico, OLB) se rotan los clusters para la asignación de los requerimientos. .
    - Tiempo Mínimo de Ejecución (MET): asigna a cada tarea al recurso con mejor tiempo esperado de ejecución para esa tarea, sin importar si el recurso está disponible o no en ese momento. La estrategia detrás de MET para cada tarea es darle el mejor grupo de máquinas.
    - Tiempo Mínimo de Finalización (MCT): asigna a cada tarea en orden arbitrario, al recurso con el tiempo de finalización mínimo esperado.

  El algoritmo seleccionado en tiempo de ejecución verifica los distintos clusters y elige el más adecuado, según su política, tomándolo para la ejecución; esto implica en el sistema que los recursos quedan bloqueados durante el tiempo que dura la ejecución.
- *Componente Heurística:* En este componente se desarrollan los métodos relacionados con la propuesta de esta tesis, como el particionamiento de Kruscal, donde se realiza una conversión de los cluster usando grafos para facilitar el manejo de la estructura de datos. Como se ha descripto en el capítulo anterior se encuentran los métodos de búsqueda y particionamiento. Una vez que se encuentran las máquinas adecuadas se dispone de métodos para la división del ancho de banda, enlace de máquinas libres y la conformación del espacio virtual.
- *Componente Estructura de Datos:* En este componente se dispone de los tipos de datos básicos y su respectiva funcionalidad. Los cuatro tipos de datos son los clusters (como conjunto), cluster como un único componente, tareas y procesadores.
    - El tipo de dato Clusters provee funciones como: cuáles equipos están libres, estadísticas, búsqueda de máquinas, migración de máquinas virtuales, consumo de ancho de banda, etc.
    - El tipo de dato Cluster dispone de funciones como asignación de los requerimientos a los equipos del cluster, estima poder de cómputo, bloqueo de máquinas, etc.
    - El tipo de dato Tarea se requiere para la inicialización de los procesos que simulan el trabajo en los clusters y provee para el cómputo de los algoritmos el consumo de memoria y la cantidad de datos de entrada/salida

necesarios. Estos datos surgen de la caracterización de la aplicación que se simula.
    - Por último el tipo procesador, así como la tarea, sirve para definir la característica del recurso y provee la capacidad de procesamiento por tipo de tarea. La tarea y el procesador son tipos de datos del tipo simulador, donde la tarea es el evento a ser atendido, y el procesador es quien lo atiende.
- *Componte Gráfico:* Este componente desarrolla las funciones de conversión de estructura de simulación a estructura de grafo para modelarlo forma visual. Se debe realizar una conversión a una estructura específica y luego indicar características como color, tipo de línea o forma del nodo para poder dibujarlo en pantalla o enviar directamente el gráfico a archivo.

En la Figura 43 se observa la relación de los componentes, el componente Simulador coordina e invoca a los componentes Heurísticas dependiendo de la configuración de código, y estas tres, utilizan el componente que define su estructura y funciones a medida que es necesario manipular los tipos de datos. Finalmente, para mostrar los datos en forma visual, las heurísticas hace uso de las funciones del componente gráfico.

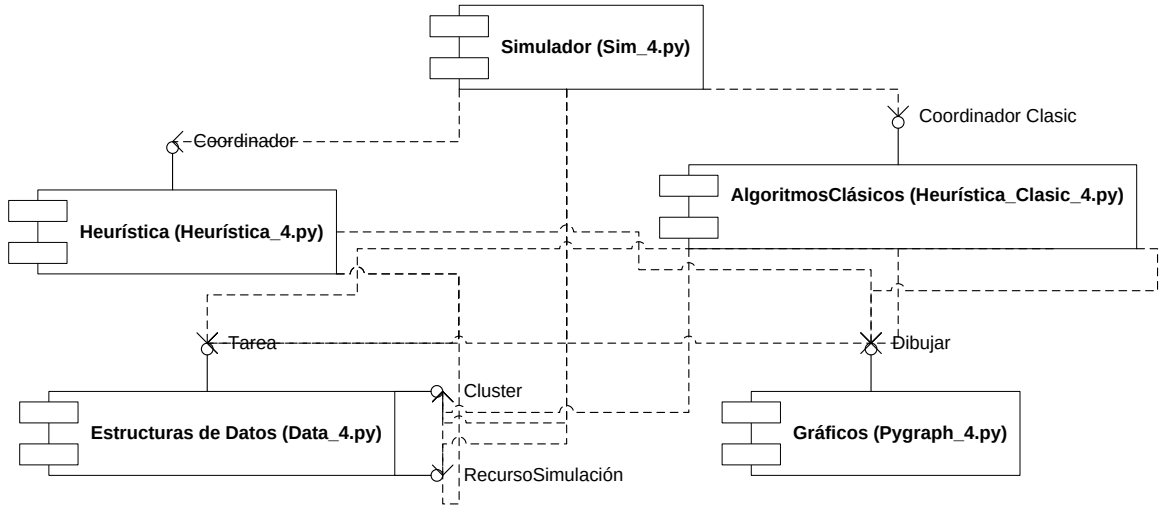


Figura 43. Diagrama de paquetes del Simulador Grid

En la Figura 44, se describe un modelo de flujo del simulador, donde las cuatro primeras tareas se realizan dentro del componente Simulador, utilizando la estructura de datos. Es importante notar que luego del arribo de un requerimiento se generan procesos concurrentes para que este pueda ser atendido y se realicen nuevos arribos. Lo mismo ocurre con la atención de las tareas: una vez que el requerimiento es atendido se crean nuevos procesos, de esta manera se desgranan en tareas y estas son atendidas según la política del algoritmo configurado. Por lo que existen tantos hilos de ejecución como requerimientos y tareas se encuentren en un momento determinado de la ejecución.

Según el algoritmo seleccionado (OLB, MCT, MET, Heurística), se realiza el particionamiento de los clusters y se comparan las mejores alternativas según la función de optimización seleccionada, pudiendo ser costo o eficiencia. Cuando se dispone del

conjunto de máquinas libres seleccionadas, se realiza el mapeo de tarea-procesador y se realiza la asignación para comenzar la simulación. Cada vez que una tarea libera el recurso Procesador, actualiza una serie de indicadores para obtener las diferentes estadísticas. También se dispone de herramientas para registrar la ejecución, con el fin de obtener niveles de trazabilidad sobre los hilos de ejecución. Esta funcionalidad se encuentra parametrizada en la rutina de configuración de la aplicación, categorizada por niveles de información, ejecución o depuración.

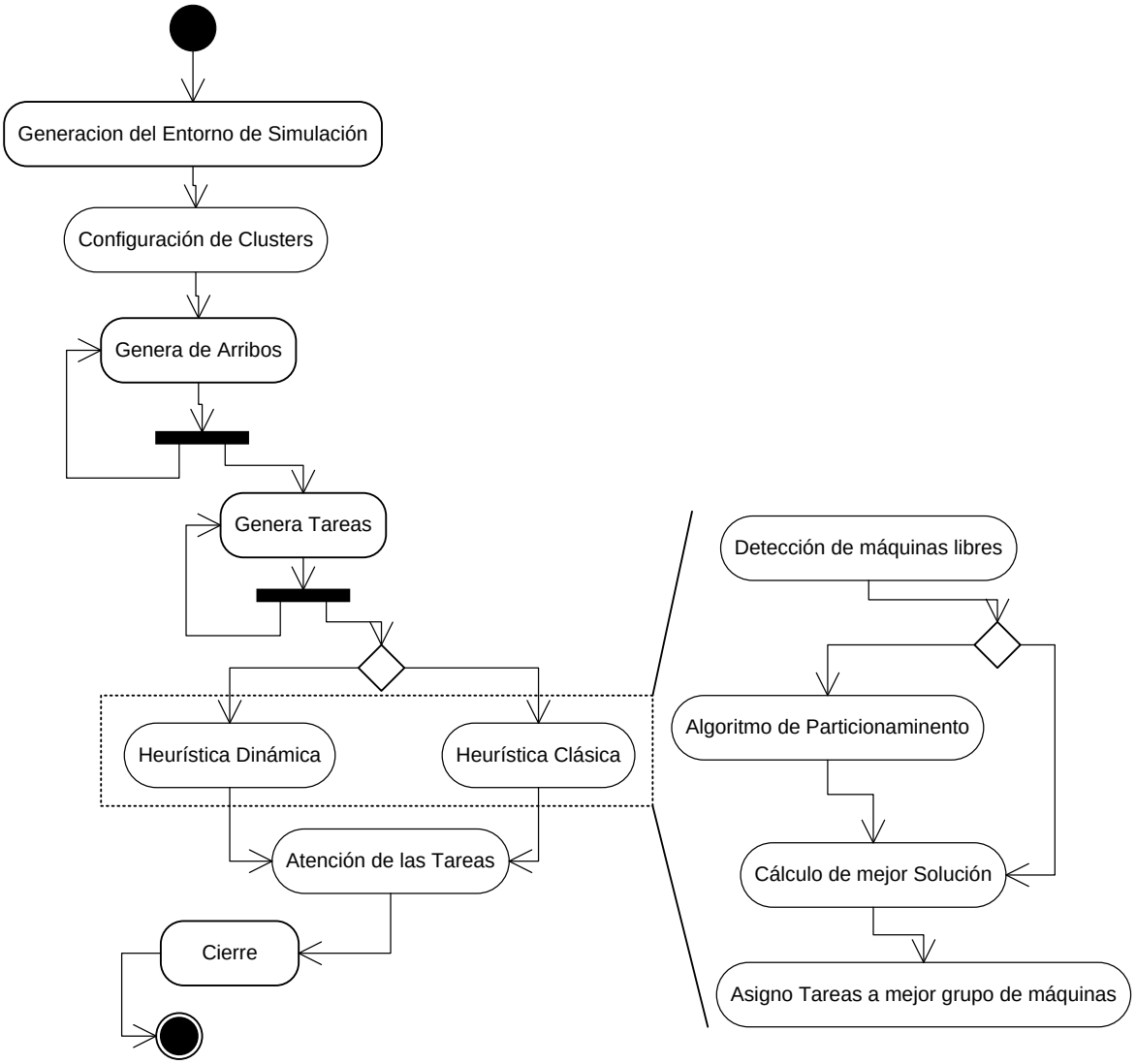


Figura 44. Diagrama de flujo del Simulador Grid

11.2. Herramientas

Para el desarrollo del simulador se usaron distintas herramientas. Dos de las más importantes son:

- SimPy (Simulación en Python) [76] es un lenguaje de simulación orientado a objeto de evento discreto basado en Python. Se puede obtener bajo la licencia GNU Lesser GPL (LGPL). Provee los componentes del modelo de simulación incluyendo

los procesos, y componentes activos como consumidores, mensajes, recursos, por ejemplo componentes pasivos como puntos de congestión de capacidad limitada y contadores. También provee variables del tipo monitor para la obtención de estadísticas. Dispone de variables aleatorias provistas por el módulo aleatorio de Python, así como también paquetes de impresión, planillas de cálculo y bases de datos. En el simulador este lenguaje se usó para la definición de la distribución en la tasa de arribo de las tareas, los servidores pertenecientes a cada cluster, y las colas de atención de los mismos.

- Visualización de grafos es la manera de representar información estructurada como diagramas de redes. El dibujo automático de grafos tiene aplicaciones importantes y es sumamente complejo organizar visualmente estas relaciones. Graphviz [77] es un software de visualización *open source* y dispone de múltiples programas para la organización de nodos y aristas visualmente. Este programa toma la descripción de un lenguaje y tiene la posibilidad de generar múltiples formatos, incluso navegadores interactivos.

11.3. Validación del Simulador

Para la validación del modelo se consideró un caso real descripto en [29], con tres clusters compuestos de ocho máquinas cada uno. Los primeros dos clusters se encontraban ubicados físicamente en América del Sur (Argentina y Brasil), tenían poco ancho de banda, aproximadamente 23KBits en promedio, y poder de cómputo limitado. El tercero con mayor capacidad de procesamiento, se encontraba en España. Si se fuerza en el simulador el mismo cluster cabecera que en el experimento real (es decir, la aplicación se inicia en uno de los cluster sudamericanos y los nodos de trabajo son los clusters en Sudamérica y Europa), los resultados finales en tiempo de cómputo y consumo de red son similares. Este experimento sirvió para validar el simulador ajustándolo a entornos reales con mejor exactitud. Si no se forzara la configuración los algoritmos seleccionarían como cluster de inicio en Europa y el tiempo total de ejecución disminuiría un 50%. Esto es porque si se inicia la aplicación en Sudamérica, habrá máquinas ociosas en el cluster de España saturando el enlace rápidamente y bloqueando máquinas con poder de cómputo. Este hecho hace la diferencia en el cálculo, mientras que si inicia en Europa los tres clusters tendrán mejor rendimiento.

12. Resultados Experimentales

Los resultados experimentales se dividen en dos fases. En la primera se analiza el comportamiento del simulador de los distintos algoritmos, tasa de arribo y patrones de comportamiento según la transferencia de datos de cada uno de los clusters y luego se muestra la mejora usando el algoritmo de selección de máquinas. La estrategia seleccionada para mostrar la mejora, es comparar con algoritmos clásicos Grid para conjunto de actividades independientes de las tareas. Los algoritmos considerados son:

- **FIFO No Restringido**: En este método, los recursos se encuentran en una lista circular y a medida que llegan nuevos requerimientos, se selecciona el próximo cluster. La mayor ventaja de estos algoritmos es la simplicidad y el balance de carga pero se encuentra lejos de lo óptimo.
- **Tiempo Mínimo de Ejecución (MET)**: La estrategia detrás de MET es darle la mejor máquina para cada tarea. Esto puede causar serios desbalances entre los

clusters y además esta heurística no es aplicable en entornos heterogéneos salvo si las tareas pueden ser caracterizadas. Esto significa que las tareas de referencia o aplicaciones que se ejecuten en forma repetitiva tengan un historial de tiempo de ejecución para mejorar la toma de decisiones.

- **Tiempo Mínimo de Finalización (MCT):** MCT asigna a cada tarea en orden arbitrario al recurso con el tiempo de finalización mínimo esperado. Esto genera que algunas tareas sean asignadas a máquinas que no tienen tiempo mínimo de ejecución. La estrategia de MCT es combinar los beneficios de OLB y MET, mientras que trata de evitar sus pobres rendimientos y las colas de espera.

En la Figura 45 se puede observar una traza parcial de las ejecuciones con los algoritmos clásicos implementados. Los distintos colores en los grafos simbolizan los diferentes clusters que se encuentran disponibles para la ejecución. En las tres celdas que contiene a la serie de grafos, es importante notar que la primera figura es similar, ya que todos comienzan desde la misma situación inicial. En la secuencia de Balance de carga optimista (OBL) en primer término se asigna el cluster rojo, cuando arriba el requerimiento 02 se asigna el azul y en el tercer requerimiento se asigna el cluster que se muestra de color verde. Para la llegada del cuarto requerimiento se han liberado los tres primeros clusters y todo comienza nuevamente. Para la secuencia de menor tiempo de ejecución, siempre se ha seleccionado el cluster rojo que es el más eficiente, y cuando se libera es automáticamente ocupado, aún cuando los demás clusters se encuentren libres. Finalmente en la secuencia de mínimo tiempo de finalización se calcula el poder del cluster y el tiempo de espera y los clusters van rotando en función a este cálculo. En la figura se puede observar que se toma primero el verde, luego el rojo, y así sucesivamente.

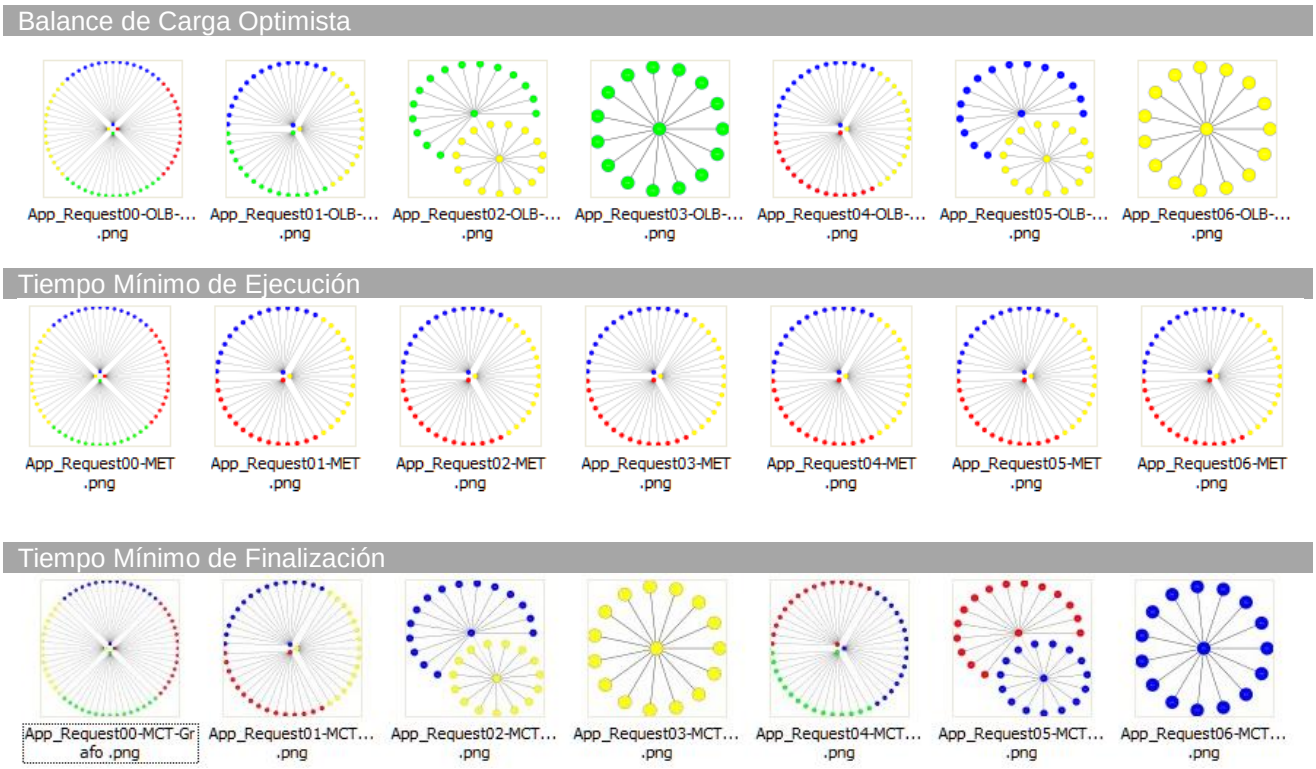


Figura 45. Asignación de Clusters según el algoritmo de planificación

No fueron seleccionados para este análisis algoritmos clásicos de Grid como Min-Min y Max-Min donde el análisis se inicia con el conjunto completo de tareas disponibles, ya que disponer de todas las tareas para poder realizar la planificación es un caso demasiado restrictivo.

Con el fin de estudiar el comportamiento de los algoritmos, la siguiente serie de pruebas se realizó modificando la tasa de arribos. En la Figura 46 se muestran distintas trazas sobre la ejecución del algoritmo propuesto. Para dar un ejemplo de la ejecución del algoritmo se utilizaron tres tasas de arribo. La primera con distribución Exponencial (donde el objetivo fue forzar que muchos requerimientos quedaran encolados), en la siguiente se usó una distribución Gaussiana y por último una tasa Uniforme, para observar al sistema en un estado estable. Todas las trazas comienzan desde el mismo estado inicial. En el segundo gráfico o los que poseen una sigla *Kr* en el nombre son las particiones de Kruscal (a lo largo de la tesis se puede observar que se usaron tres particiones, este número deriva de la cantidad de clusters). En el caso de la distribución Gaussiana el patrón siempre es el mismo ya que no alcanza a llegar el nuevo requerimiento que el anterior ya ha finalizado y siempre elige el mismo cluster. Lo mismo ocurre en la distribución Uniforme hasta la llegada del segundo requerimiento, éste ya comienza a trabajar con las máquinas libres que no usó el primer requerimiento y se ve aún más acentuado en la llegada del tercer requerimiento.

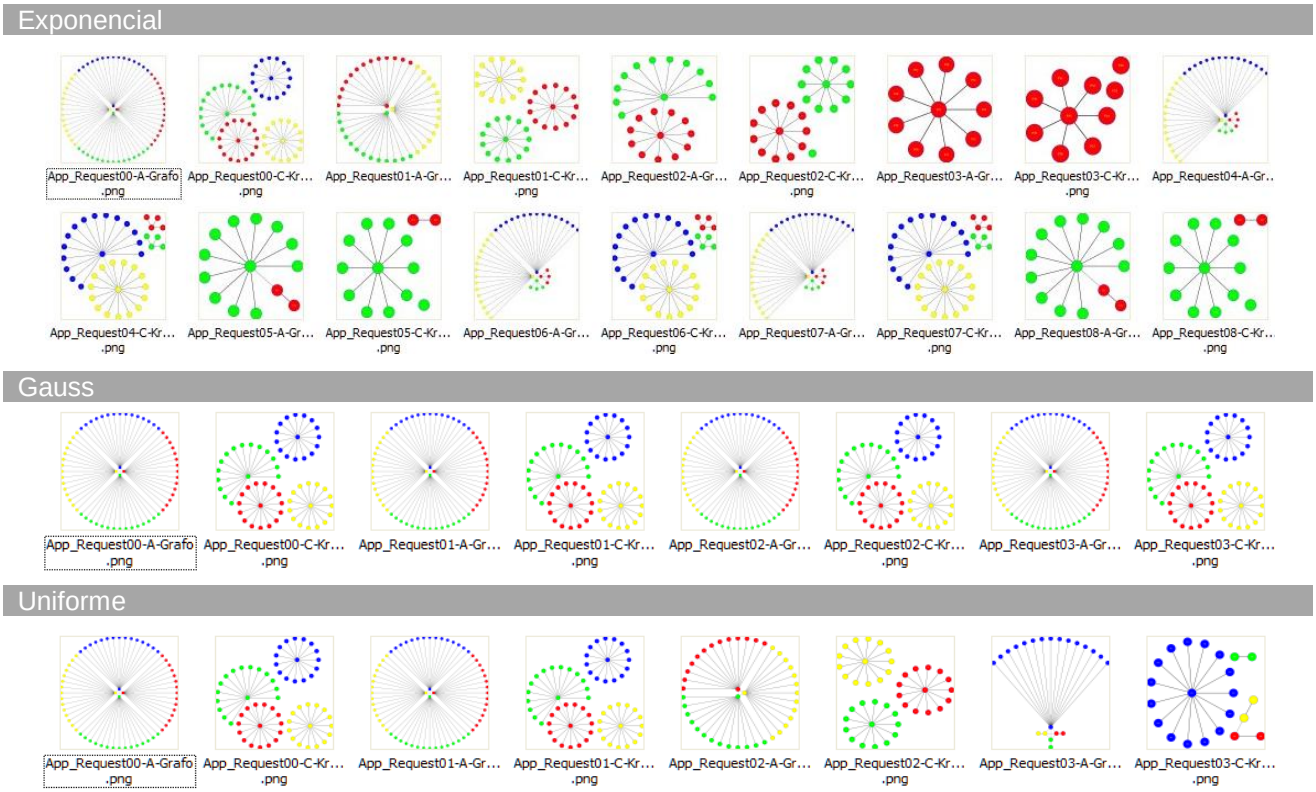


Figura 46. Comportamiento del simulador con distintas tasas de arribo.

En el caso de tasa de arribo Exponencial, el impacto de arribos de tareas es inmediato: con la llegada del primer requerimiento se puede observar que no se dispone de la

totalidad de equipos como el caso inicial y a medida que las ejecuciones van terminando la disponibilidad de equipamiento libre varía notablemente.

En la Figura 47 se puede observar cómo afectan a la ejecución los cambios en la cantidad de información de entrada y salida por los distintos requerimientos. En la línea de tendencia en la transferencia de 100MB los tiempos de espera de las tareas son menores porque el tiempo total de ejecución es menor y a medida que se incrementa, aumentan los tiempos de espera. Es necesario aclarar que la pendiente positiva de las líneas de tendencia se debe a que la tasa de arribo de los requerimientos es exponencial. Este tipo de pruebas permitió determinar umbrales para las configuraciones de máquinas y conexiones entre clusters, con el fin de probar condiciones atípicas en el algoritmo propuesto. Por lo que se puede determinar que los parámetros que tienen mayor incidencia en los algoritmos son: la capacidad de cómputo del equipamiento, la tasa de entrada y salida de las tareas y la tasa de arribo de los requerimientos.

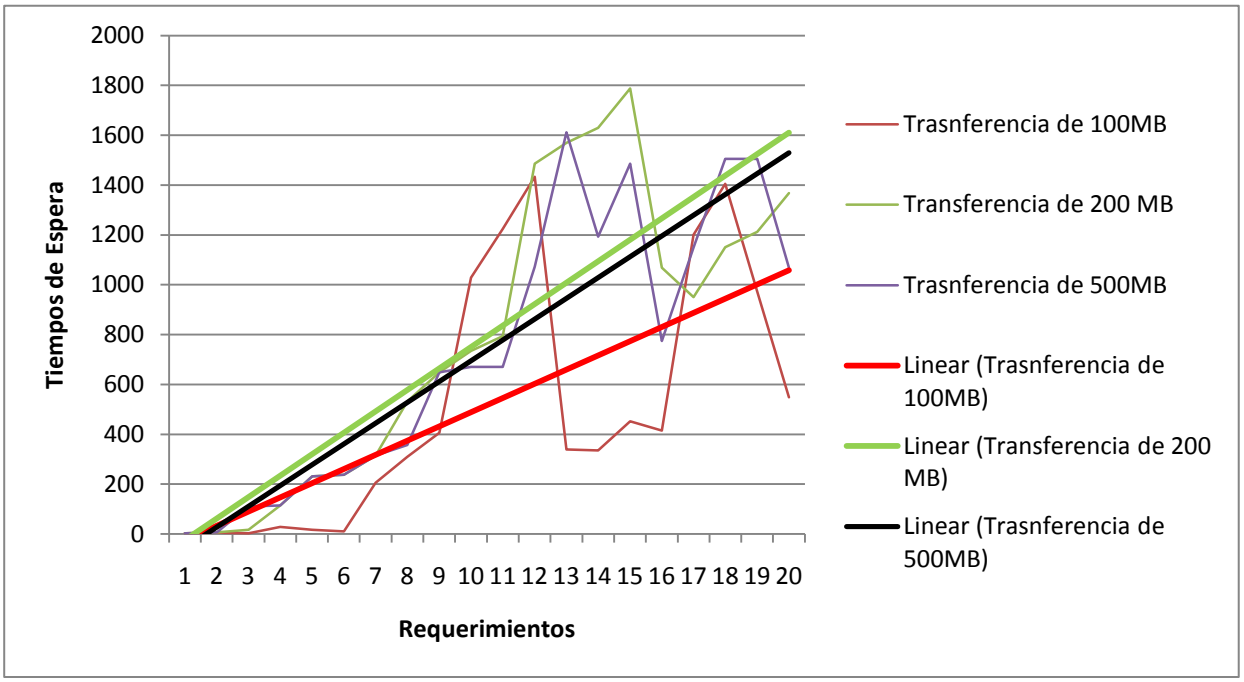


Figura 47. Variación en los tiempos de espera de las tareas según transferencia de datos

12.1. Simulación de la Heurística

En todos los casos MCT obtuvo mejor rendimiento que MET y OLB, por lo que en los gráficos finales se compara siempre la heurística propuesta con las ejecuciones del algoritmo de tiempo mínimo de ejecución (MCT). En el gráfico de la Figura 48 se observa la diferencia de tiempos entre la heurística y el algoritmo. El algoritmo *Hill-Climbing* usa todas las máquinas de los clusters, seleccionando el mejor cluster cabecera y MCT busca el cluster con mejor tiempo de finalización. Eligiendo correr la aplicación en todas las máquinas la heurística mejora un 70% sobre MCT (por supuesto es el mejor caso).

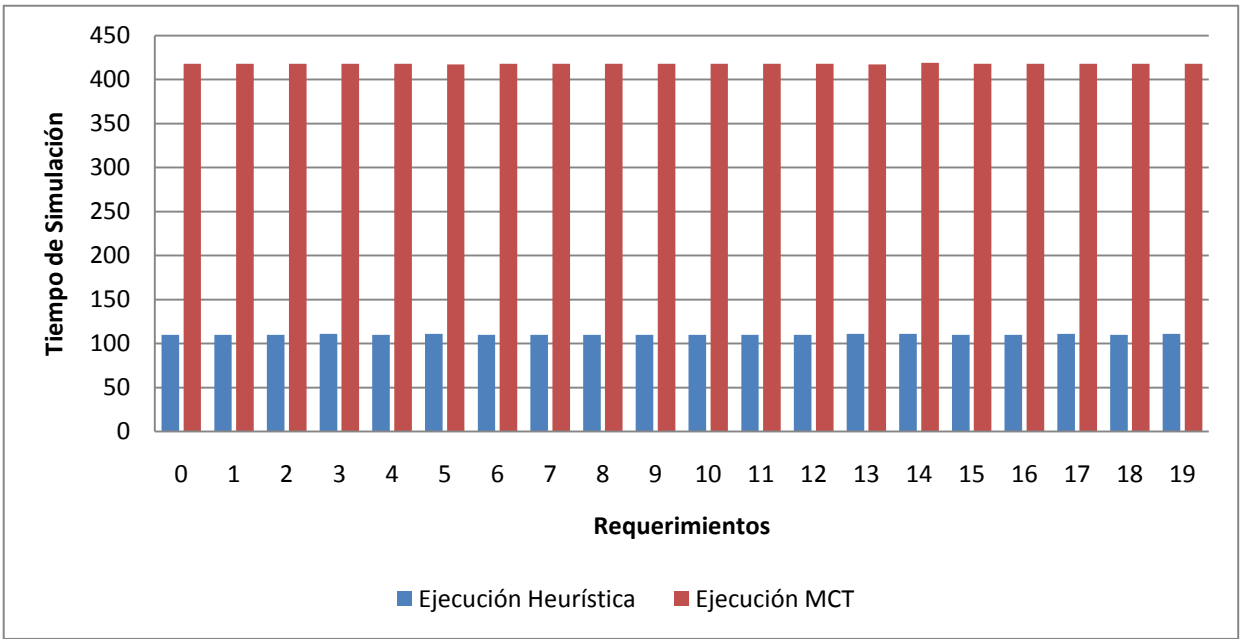


Figura 48. Tiempos de Simulación Heurística vs MCT

Si cada tarea paralela envía más datos mientras procesa, entonces la aplicación comienza a ser limitada por la comunicación, limitando el número de máquinas que cada cluster puede usar para realizar el cómputo, y bajando el rendimiento del algoritmo Heurística. Esto puede verse reflejado en la Figura 49: en la primera columna se encuentra la asignación de clusters según el algoritmo MCT y en la segunda según la heurística propuesta con una tasa de entrada/salida de datos media en relación al ancho de banda disponible. Para el primer arribo el algoritmo MCT asigna un cluster completo; la heurística asigna un cluster completo, el C3, pero como el intercambio de datos lo permite, también utiliza máquinas de los otros clusters (cuatro sobre dieciséis disponibles, es decir un 25%). Con el arribo del segundo requerimiento el cluster MCT realiza sus cálculos y asigna el cluster C2. La heurística asigna el cluster C4 y algunas máquinas más de los clusters C2 y C1.

Es interesante analizar por qué no se asignaron estas máquinas en el primer requerimiento. La respuesta se encuentra en la configuración del entorno de ejecución, ya que se asignaron clusters de igual capacidad de cómputo y ancho de banda. Los clusters en que se realiza el trabajo tienen capacidad ociosa de conexión, porque solo con el 50% de cada enlace el cluster cabecera satura su conexión. Finalmente se puede observar que se llega al mismo estado en ambos algoritmos, aunque es importante destacar que si bien el final es el mismo, durante la ejecución los recursos se han usado de manera más eficiente en la heurística utilizando recursos que de otra manera quedarían ociosos, como se ven en el algoritmo MCT.

Una vez que el simulador se adaptó correctamente al cambio de condición de las tareas y entorno de red, y el algoritmo probó ser eficiente con baja tasa de arribos, el siguiente paso fue aumentar esta tasa en forma exponencial. La cantidad de equipos es similar a la de los experimentos anteriores, se realizaron variaciones entre cuatro y ocho clusters con equipos que variaban entre ocho y dieciséis máquinas por cluster. También se probaron distintas capacidades y tipos de comunicación.

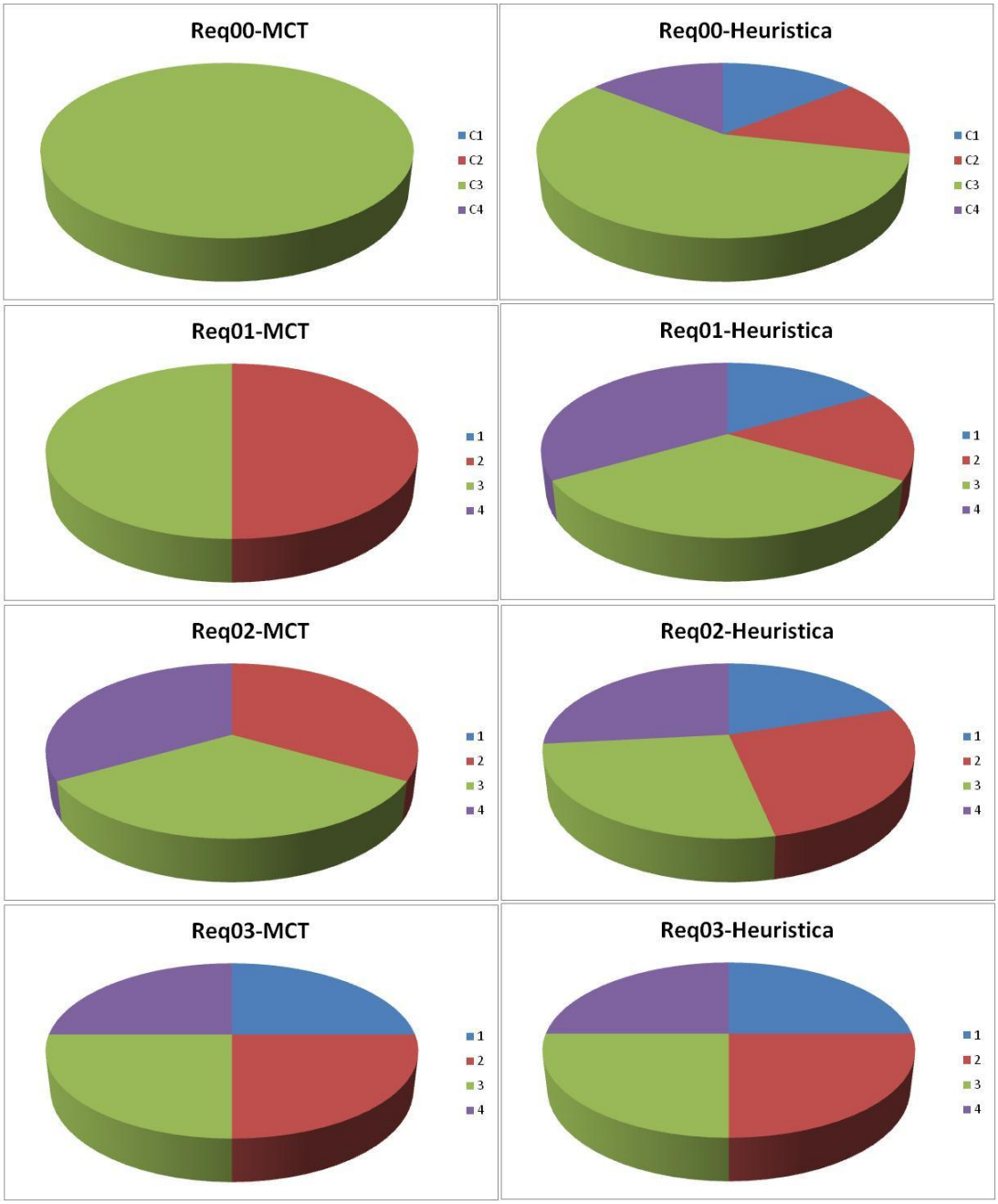


Figura 49. Asignación de clusters. Primera columna algoritmo MCT, segunda Heurística HC.

En todos los casos el resultado de la simulación fue similar al de la Figura 50, con una ejecución de MCT constante y muchas variaciones en la heurística, sobre todo con la variación de capacidad de cómputo y la transferencia de los datos. Si se aumenta la capacidad de cómputo o si se tiene mucha transferencia de información, el algoritmo heurística tiende a ser similar a MCT ya que satura los enlaces y termina solo por ser el cluster cabecera el que queda funcionando. Si se disminuye la capacidad de cómputo de los clusters, con el parámetro de tareas por segundo, o si la transferencia de datos es mínima, la figura se parece cada vez más a la número 48 donde se logra una diferencia de un 70%. Hasta la llegada del segundo requerimiento el tiempo de ejecución es menor en MCT. En la Figura 50 (una ejecución promedio de computo y transferencia), se observan mejores tiempos de ejecución en los primeros requerimientos y luego un

aumento considerable. Esto se debe a un fenómeno similar al que ocurre en el sistema operativo con la utilización de las páginas de memoria, que es el de la fragmentación. Los clusters al ser usados parcialmente incurren en muchos costos de comunicación con pocas máquinas asignadas. Al analizar este problema, ya que si bien se tenían mayores tiempos de ejecución, en los tiempos finales se lograba una modesta mejora, se incorporó la variable de tiempos de espera. De esta manera por requerimiento se tomaba el tiempo total de ejecución o *turnaround*. En el gráfico se ve que el algoritmo de MCT tiene mayores tiempos totales de ejecución debido a que las tareas con una tasa de arribo exponencial deben esperar más para ser atendidas que la heurística.

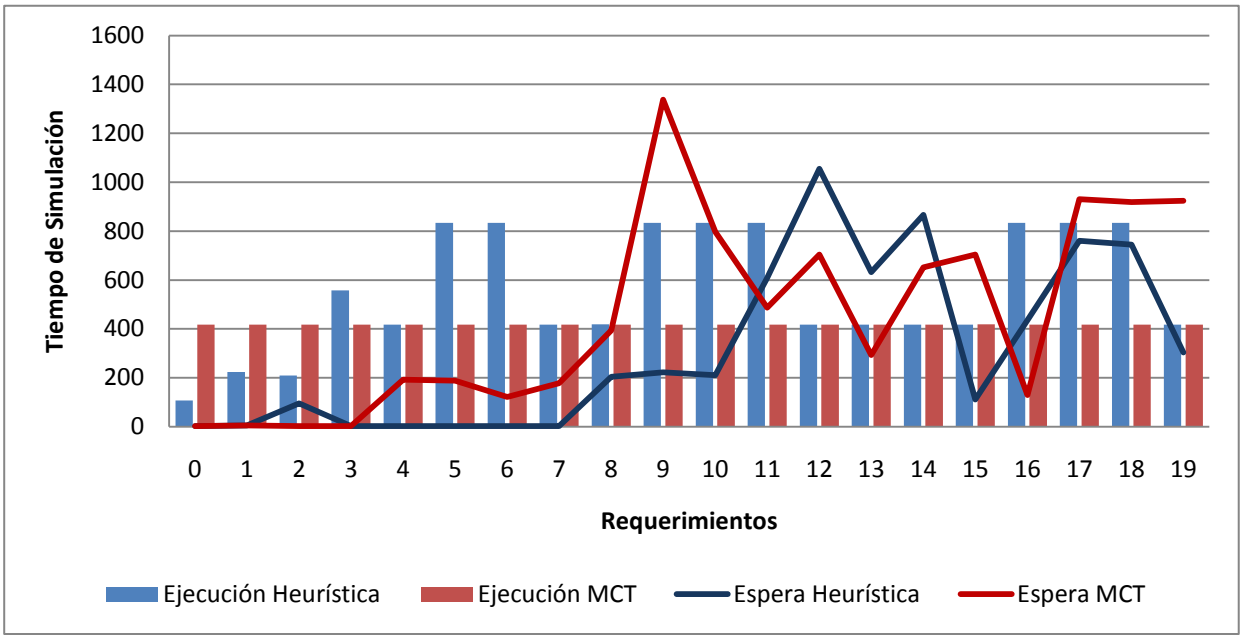


Figura 50. Ejecución Heurística y MCT con respectivos tiempos de espera

La heurística depende mucho de los factores del entorno como tipo de máquinas y conexiones; por lo que modificar el comportamiento de las aplicaciones no tiene mucho impacto en estos escenarios. Se puede afirmar que la estrategia de planificación, agentes locales y distribuidos del entorno Grid podrán lograr un máximo aprovechamiento en aplicaciones de tipo master-worker que corran de manera recurrente en un entorno Grid, como de hecho ocurre en este tipo de entornos siendo estas de las más comunes.

Luego de realizar un mayor número de corridas y hacer múltiples combinaciones de los parámetros disponibles se concluyó que en el peor de los casos el tiempo de *Hill-Climbing* con particionamiento tiende a ser similar al de MCT. El peor caso es que se transfieran tantos datos, que se sature el enlace de red y quede un solo cluster activo. En el escenario promedio (baja tasa de arribos con picos en intervalos regulares) y la taza de transferencia de información habilita la incorporación de máquinas de otros clusters, *Hill-Climbing* tiene una mejora de un 20% en los tiempos de espera de los requerimientos. En el anexo A se puede observar algunas de las ejecuciones para los distintos algoritmos.

13. Experiencias para adaptación de clusters

En las pruebas de adaptación de clusters, diferentes tipos de tareas fueron enviadas con distintos requerimientos de entrada/salida. También fueron mezcladas con tasas de arribos exponenciales y uniformes. Algunos requerimientos se superponían en el tiempo en espera pero estos tiempos no fueron largos. Luego de la ejecución de varias simulaciones se pudo notar que en algunos clusters el uso de los mismos no llegaba al máximo aprovechamiento, debido a que tareas con gran necesidad de entrada/salida tomaban poco porcentaje de CPU disponible y rápidamente saturaban el enlace de red a Internet. En el caso de arribo de tareas con requerimiento de poco ancho de banda, la utilización de los equipos se incrementaba minimizando la perdida de recursos con máquinas bloqueadas por comunicación.

Para solucionar el problema del uso limitado del cluster se propuso utilizar la característica de migración de máquinas virtuales en conjunto con la información disponible de los procesos de ejecución para acotar el consumo de ancho de banda. Si una máquina virtual en ejecución consumía un ancho de banda determinado y se migraba a otra máquina física que ya tenía otra máquina ejecutándose [78], el algoritmo de planificación del entorno virutal por defecto realiza una asignación “justa” del procesador repartiendo el poder de cómputo en partes iguales; este tipo de asignación limita la cantidad de trabajo por unidad de tiempo liberando ancho de banda consumido. De esta manera se liberó ancho de banda pudiendo ser utilizado para un nuevo proceso. Este proceso es ventajoso cuando el tiempo de ejecución ganado con la incorporación de mayor cantidad de máquinas a la tarea recién arribada es mayor a la penalización de quitarle equipos a la tarea que se encontraba ejecutando en ese momento.

En el gráfico 51 se puede observar conceptualmente cual es la solución propuesta. Se dispone de dos requerimientos (A y B); en un primer momento solo se encuentra ejecutando el requerimiento A con cuatro tareas, dos por equipo. Cada una de estas tareas se ejecuta sobre una máquina virtual. Cuando arriba el requerimiento B, si se migran dos de las tareas, teniendo en una sola máquina física cuatro de las tareas compartirán el procesador de manera equitativa, y esto se debe al planificador de virtualización. El requerimiento B encuentra procesadores libres donde ejecutar y teniendo la caracterización del requerimiento A se puede calcular, por el tiempo de ejecución transcurrido, cuál es el tiempo restante con la nueva configuración.

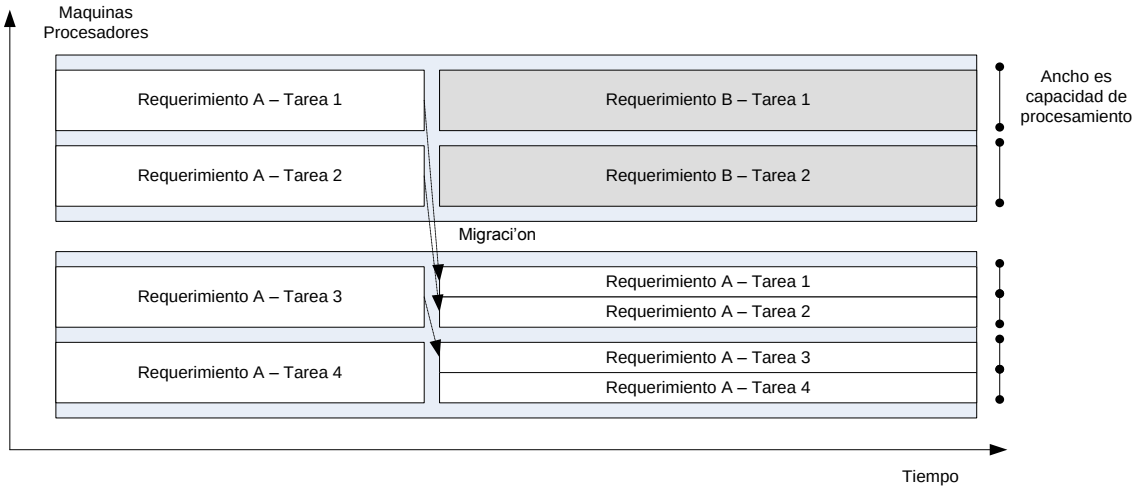


Figura 51. Migración Dinámica de Máquinas virtuales

Se ha descrito en este capítulo el proceso de simulación (sección 2.2) y el proceso para seleccionar máquinas libres. La metodología de adaptación tiene como fin ampliar la oferta de estas máquinas libres para procesos con necesidad de transferencias de entrada/salida mínimas. Por este motivo, se parametrizó el simulador y se incorporó esta funcionalidad al código fuente. A continuación se puede observar el diagrama de bloque (Figura 52) sobre los módulos del simulador al que se incorporaron los algoritmos de migración.

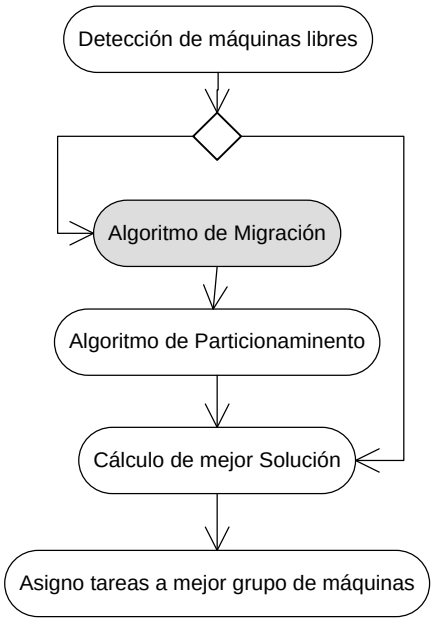


Figura 52. Diagrama de Flujo para la migración dinámica de MV

Primero se detecta si ha arribado un requerimiento de las características antes mencionadas, y el algoritmo de migración (griseado) verifica en la bitácora si los requerimientos en ejecución pueden ser migrados. Si los requerimientos disponen de las suficientes máquinas calcula el tiempo restante con una menor asignación de tiempos. Si el tiempo restante es menor al tiempo total de ejecución del nuevo requerimiento que llega entonces la migración se realiza; si no fuera el caso, no tiene sentido realizar la migración ya que la pérdida en performance es mayor que la ganancia de incorporar una nueva tarea.

En el gráfico de la Figura 53 se pueden observar tres procesos de migración sobre una ejecución en el simulador; la llegada del requerimiento 8, 13 y por último 18. En la primera columna se describen la conectividad de las máquinas libres previa al proceso de migración, en la segunda columna las máquinas libres luego de haber corrido el algoritmo de migración y por último se ve el proceso de particionamiento de Kruscal sobre el grupo de máquinas libres. En el arribo del requerimiento 8 se agregan cinco máquinas, en la primera columna se dispone de sólo dos máquinas para el cluster azul y luego cuenta con siete. Con el arribo del requerimiento 13, la cantidad de máquinas que se agregan es mucho mayor, el cluster amarillo pasa de dos a siete, el cluster rojo de cuatro a quince, y por último el azul de siete a once máquinas. Con el nuevo grupo de máquinas disponibles son evidentes los beneficios del proceso de migración, siempre y cuando el impacto

sobre las tareas en ejecución sea considerado porque de lo contrario solo será sobrecarga para el tiempo total de ejecución.

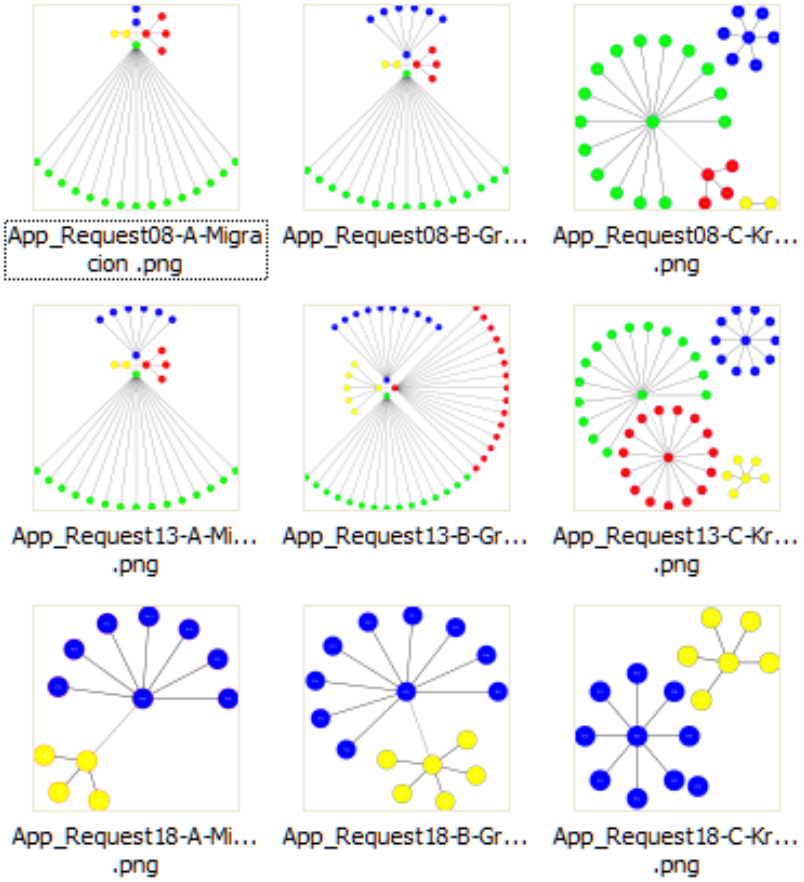


Figura 53. Reasignación de Clusters según el algoritmo de planificación con migración

Para calcular el impacto de esta migración, en el simulador se realiza una asignación ficticia. Se calculan los tiempos y una vez analizadas todas las combinaciones de asignación y comparaciones sobre el cálculo de tiempo total, en ese momento, se lleva a cabo el proceso de migración realmente. Se asignan los requerimientos y sus tareas a las nuevas máquinas, agregando las penalizaciones por la migración.

Una vez que se toma la decisión de realizar la migración se deben llevar a cabo movimientos sobre la asignación de requerimientos y sus tareas sobre los clusters. En la Figura 54 se ve la asignación de clusters con el arribo del requerimiento 18, en la primera columna antes de ejecutar la asignación y en la segunda luego de realizar la migración de las tareas. En la primera fila se puede observar que el requerimiento 17 se encuentra asignado un 100% al cluster número 1 y lo mismo ocurre después de la migración, con el cluster C2 y los requerimientos 2 y 19. La modificación ocurre sobre los clusters C3 y C4, el requerimiento 17 pierde un porcentaje de sus máquinas, el requerimiento 19 mantiene sus máquinas, dando como resultado que el requerimiento 18 que recién ha arribado es el requerimiento con mayor porcentaje de equipos ya que toma las máquinas liberadas por el requerimiento 10 y suma las del 17. Otro cambio importante se observa en el cluster C4, donde en un primer momento cuatro requerimientos tenían un cuarto cada uno de los clusters disponibles, el requerimiento 18 toma el lugar del 10 pero nuevamente

el requerimiento 17 es el afectado por la migración. Este requerimiento demorará más tiempo en ejecutarse pero al aumentar la cantidad de máquinas del requerimiento 18 este tiempo se verá compensado.

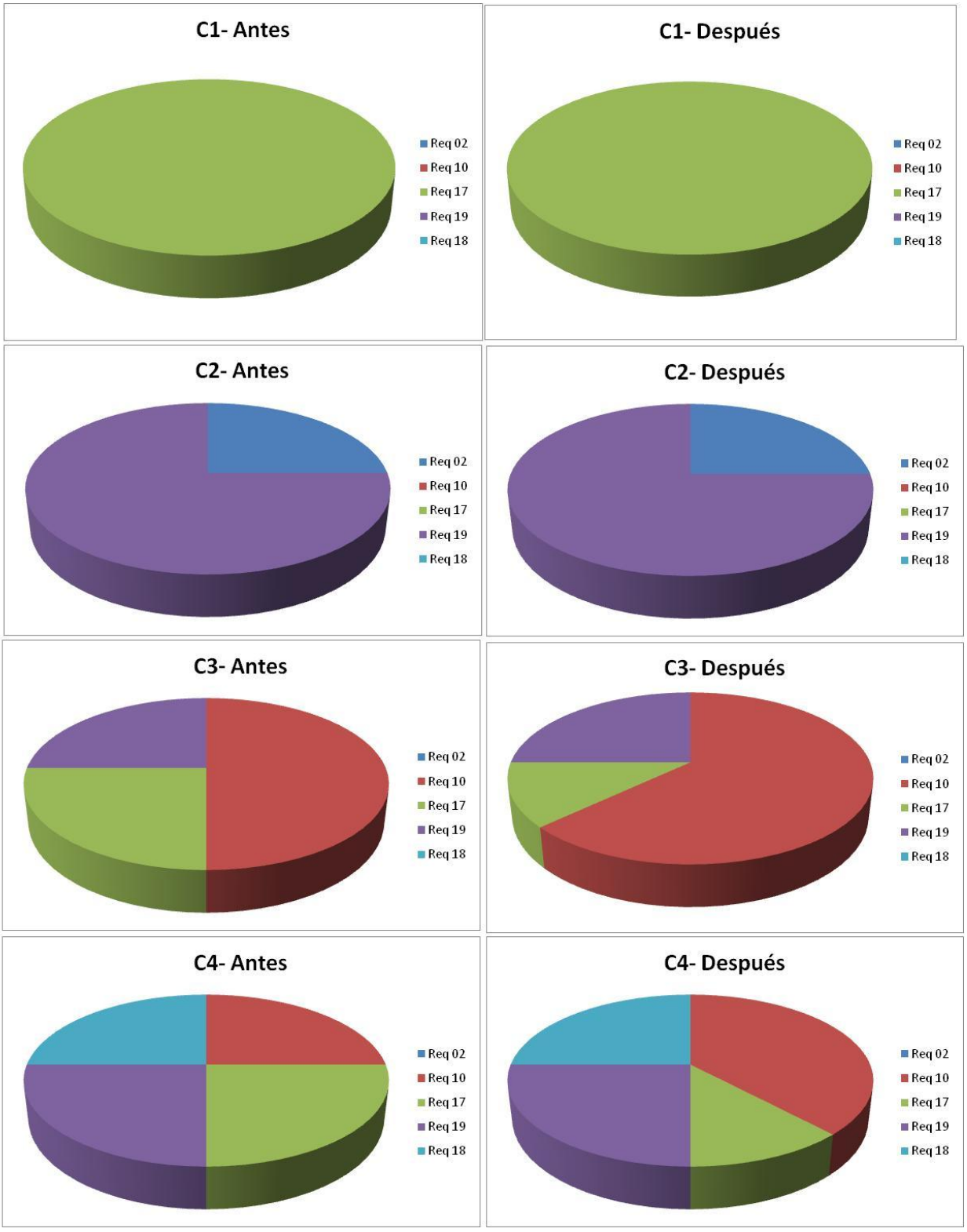


Figura 54. Uso de clusters según algoritmo de migración

En el ejemplo anterior, la totalidad de las máquinas habían sido asignadas previamente y el algoritmo hizo lugar para el ingreso de un nuevo requerimiento. También se puede encontrar otro tipo de ejemplo, habitual en las simulaciones, que de un cluster de dieciséis máquinas solo estaban ocupadas un 25% y estas cuatro máquinas saturaban el enlace de red quedando ociosas las doce restantes; dejando al requerimiento original con dos máquinas, se pudieron utilizar seis máquinas y llevar el rendimiento del cluster a un 50% con nuevos requerimientos de entrada y salida de datos limitada.

Luego de un exhaustivo análisis de las características del proceso de migración y del impacto que éste tenía sobre el algoritmo de planificación, se realizaron múltiples simulaciones comparando el comportamiento habitual de la heurística propuesta, la heurística con migración y finalmente con el algoritmo MCT. Una ejecución promedio puede observarse en la Figura 55. En la caja de la izquierda, el gráfico muestra cómo se asignan en un primer momento las tareas a las máquinas virtuales, en este caso, el requerimiento 3 y el requerimiento 5, y luego cómo se asignan estas máquinas después del proceso de migración. En el primer caso se asignaron ocho máquinas y luego dos de estas máquinas bloqueadas fueron asignadas. Un pequeño incremento puede notarse en el tiempo de ejecución en el requerimiento 1 debido a la disminución del poder de cómputo luego de la migración. Lo mismo pasa con el requerimiento 5. El tiempo que gana es más evidente en este caso. En el resto de la simulación se pueden encontrar otros casos de migración. Esto ocurre debido a los cambios de configuración luego de los procesos de migración.

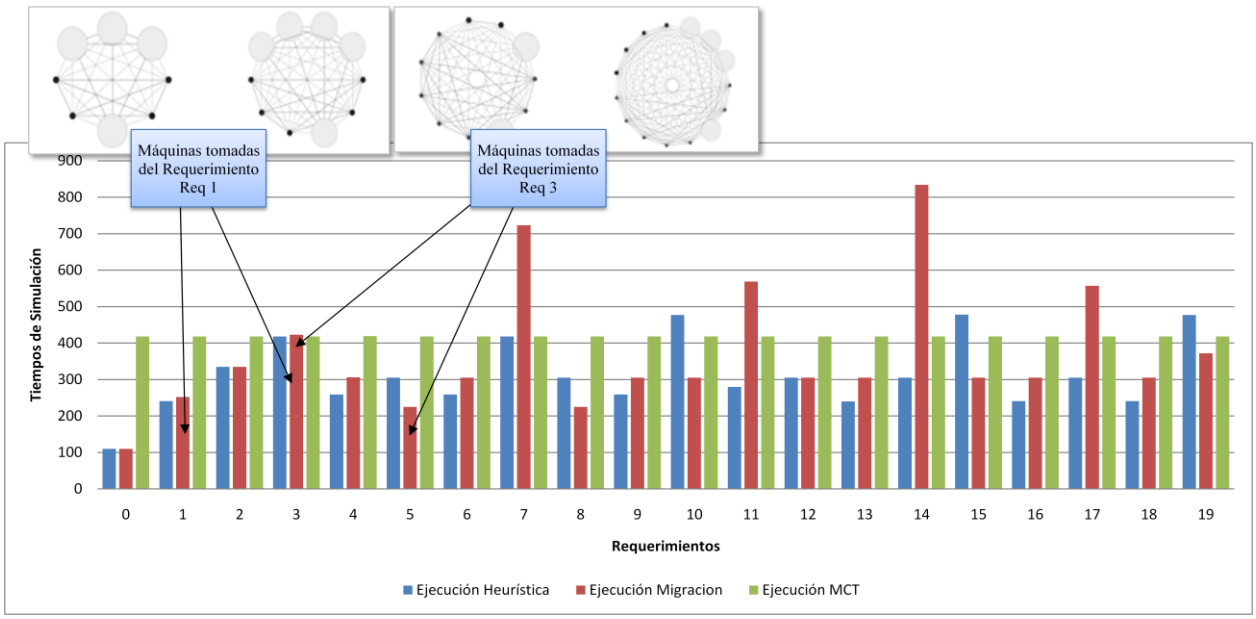


Figura 55. Ejemplo de procesos de migración.

En la siguiente Figura 56 se observan los tiempos de espera para la atención de las heurísticas, lo que nos permite concluir que el tiempo total de ejecución del algoritmo es mejor con migración que sin ella. En promedio de la mayoría de las simulaciones realizadas, la mejora de rendimiento es de un 5% sobre el algoritmo originalmente propuesto.

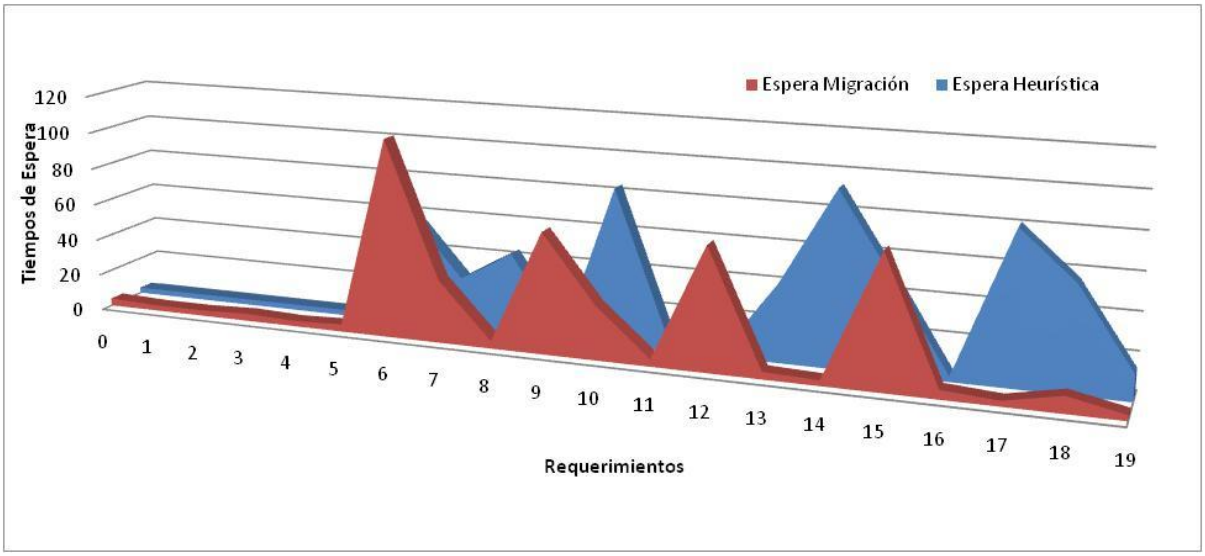


Figura 56. Tiempos de espera de los procesos según algoritmo de planificación con migración

En el porcentaje de mejora que se obtuvo, tiene mucha influencia el balance entre el tipo de tareas. Es decir, qué relación tienen las tareas entre computo-comunicación y qué relación tienen dependiendo su arribo al sistema. Si las tareas tienen similares características de comunicación pocas veces se ejecutará algún tipo de migración, ya que la proporción de tiempo ganado con la incorporación de equipos para cómputo será ínfima. Cuanto más diferencia haya en esta relación (y si a esto se le suma una mayor cantidad de arribos de tareas con mucha necesidad de computo), la heurística tendrá un impacto muy positivo en el tiempo de ejecución total de los requerimientos.

14. Resumen

En este capítulo se han presentado las pruebas de los algoritmos propuestos para la extensión e implementación de metaplanificador para la generación de laboratorios de máquinas virtuales. Esta arquitectura no solo toma en cuenta el tiempo para completar la tarea sino el consumo de ancho de banda, con el propósito de tomar ventaja de un mayor número de máquinas disponibles en el entorno Grid. El algoritmo de particionamiento geográfico a través del uso del algoritmo de Kruscal también soluciona el problema de escalabilidad, minimizando la complejidad en la búsqueda de la solución optima. Hill-Climbing con múltiples recomienzos no asegura encontrar la solución óptima, pero por las pruebas realizadas esta solución es encontrada la gran mayoría de las veces. Se realizaron pruebas cambiando las tasas de arribos, poder de cómputo de los clusters, número de tareas y requerimientos de entrada y salida. Los algoritmos de selección han sido implementados para encontrar grupos de máquinas con características especiales en un espacio de búsqueda limitado, con el compromiso de devolver una solución de manera rápida y óptima. El conjunto de estas implantaciones incrementaron el poder de cómputo, mejorando el rendimiento de todo el sistema aproximadamente en un 20%. En el anexo A se puede observar algunas de las corridas realizadas y los datos obtenidos.

Han sido diseñados e implementados algoritmos dinámicos para adaptar el uso del cluster en tiempo de ejecución con la migración de máquinas virtuales. Esta implementación resulta en un 10% de mejora en el tiempo total de cómputo, con

máquinas trabajando a un 100% de su tiempo asignado. Estos resultados permiten concluir que la factibilidad de implementar la solución en entornos Grid, donde un análisis multicluster puede guiar un balance entre el cómputo y la comunicación, determinando una manera efectiva de regular el ancho de banda disponible impactando en el poder de cómputo comprometido.

Capítulo 7 Conclusión

El objetivo de esta Tesis es el análisis para la gestión de entornos virtuales de manera eficiente. En este sentido, se realizó una optimización sobre el middleware de planificación en forma dinámica sobre entornos de computación Grid, siendo la meta a alcanzar la asignación y utilización óptima de recursos para la ejecución coordinada de tareas.

Se investigó en particular la interacción entre servicios Grid y la problemática de la distribución de tareas en meta-organizaciones con requerimientos de calidad de servicio no trivial, estableciendo una relación entre la distribución de tareas y las necesidades locales pertenecientes a organizaciones virtuales.

La idea tuvo origen en el estudio de laboratorios virtuales y remotos para la creación de espacios virtuales. En muchas organizaciones públicas y de investigación se dispone de gran cantidad de recursos, pero estos no siempre se encuentran accesibles, debido a la distancia geográfica, o no se dispone de la capacidad de interconectarlos para lograr un fin común. El concepto de espacio virtual introduce una capa de abstracción sobre estos recursos logrando independencia de ubicación y la interactividad entre dispositivos heterogéneos, logrando de esta manera hacer uso eficiente de los medios disponibles.

Durante el desarrollo se ha experimentado y logrado la implementación de un entorno para la generación de espacios virtuales. Se ha definido la infraestructura, se implementaron dos tipos de laboratorios y se ha propuesto una optimización para lograr el máximo aprovechamiento en un entorno para aplicaciones paralelas. Actualmente estos conceptos han evolucionando y algunas de las ideas publicadas se han implementado en prototipos funcionales para infraestructuras comerciales, si bien aún se encuentra en investigación la planificación sobre centros de cómputos para miles de equipos.

En la Figura 57 se puede observar la línea de tiempo del desarrollo, con los distintos temas que se fueron abordando. En cada uno de ellos se realizaron implementaciones así como publicaciones, y la maduración de los conceptos desarrollados en los últimos capítulos también se refleja en publicaciones de congresos a nivel internacional. A continuación se describen las conclusiones y aportes de los diferentes capítulos desarrollados.

Figura 57. Línea de tiempo desarrollo de la tesis

15. Aportes y Conclusiones

15.1. Multiclusters

En lo referente a multiclusters se tuvo como objetivo evaluar cómo afecta un entorno Grid a la interconexión de clusters dedicados a través de Internet. En primer lugar se seleccionaron aplicaciones paralelas, se realizaron experiencias de adaptación al entorno Grid y luego se efectuaron búsquedas de patrones de conexión y predicciones en el comportamiento de transferencia de datos sobre clusters y el conjunto interconectado mediante Internet. Por último, se realizó una optimización en la asignación del equipamiento disponible en los distintos clusters para lograr un rendimiento efectivo del poder de cómputo del conjunto.

Según las experiencias realizadas con distintas configuraciones de equipos y conexiones de red, la incorporación del middleware de Grid no altera en absoluto el desarrollo de aplicaciones paralelas de tipo master-worker. Pero si altera, la incorporación de middleware Grid el diseño de la aplicación paralela. En este trabajo se debió modificar la lógica de trabajo de multicluster por la de un modelo orientado a servicios, la cual es más acorde al uso eficiente del middleware Grid.

15.2. Meta planificación

Debido a la gran cantidad de recursos de hardware y software que componen los sistemas Grid, cada uno con diferentes características y complejidades, se torna imperioso simplificar y automatizar su administración. Para esto se presentó una arquitectura con dos niveles de planificación: el primero a nivel de meta-organización, con el objetivo de generar clusters a demanda, basado en requerimientos de aplicaciones, y el segundo a nivel de organización local, gestionando los recursos del cluster en forma dinámica para lograr un máximo aprovechamiento de los recursos ofrecidos. Como recursos para generar el entorno virtual se definieron máquinas virtuales que interconectadas forman un cluster homogéneo entre organizaciones.

El aporte de esta sección, luego de realizar un relevamiento del estado del arte, fue la generalización de la RFC 2753 a entornos Grid. Sumado a esto, la extensión del meta-planificador CSF para la generación de espacios virtuales. Para el administrador local de recursos, se han verificado a través de experimentación, los beneficios del uso de máquinas virtuales para lograr un máximo aprovechamiento de los clusters. De esta forma se pueden modificar de manera homogénea parámetros de ejecución como memoria y cantidad de procesadores en forma dinámica, y migrar máquinas virtuales reasignando el espacio de máquinas físicas sin tener que detener la aplicación que está corriendo en el cluster. Estas pruebas permitieron concluir que ésta tecnología es factible de aplicarse en espacios Grid, donde existe la necesidad de adaptación y control del entorno. En éste sentido, redunda en beneficios importantes para la calidad de administración y una mejor utilización de recursos.

15.3. Espacio Virtual

Luego de proponer la arquitectura y de realizar ciertas pruebas, se decidió profundizar aún más. Se implemento una solución sobre los aspectos de planificación y gestión desarrollando la arquitectura propuesta en el capítulo anterior. Esta utiliza dispositivos virtuales para lograr un marco de trabajo homogéneo, permitiendo al mismo tiempo una configuración dinámica y flexible, dentro de una organización virtual sobre un entorno Grid. Con foco en la interconexión de recursos de cómputo, asegura un espacio de trabajo con estándares y protocolos abiertos bajo estrictas normas de seguridad.

Con este capítulo se deduce que es posible interconectar y utilizar en forma coordinada recursos de cómputo geográficamente dispersos, en forma dinámica y flexible, gracias a un esquema de organización virtual provisto por la infraestructura Grid y a las distintas opciones que provee el uso de máquinas virtuales.

Como aporte del trabajo aparecen aspectos de:

- Configuración: En lo relativo a configuración se ha desarrollado un pseudo lenguaje orientado a objetos para el diseño y validación de la especificación lógica.
- Acceso: Para el acceso se ha desarrollado un modelo de puntos de acceso realizando la redirección de terminales virtuales hasta el usuario final a través de Internet.
- Gestión de los recursos: En cuanto a la gestión se ha utilizado la funcionalidad del middleware Grid para permitir la ejecución y transferencia de información entre dominios administrativos diferentes sin la intervención de administradores locales.

Las pruebas de rendimiento no están exentas de sobrecarga por la virtualización de cómputo de red. En el caso donde los requerimientos de performance no son específicos y lo importante es la creación de un entorno virtual de trabajo aislado del entorno físico que lo contiene, ésta es una solución aceptable que permite hacer un uso eficiente de los recursos disponibles o incorporar elementos de otro modo inaccesibles. Además se debe contemplar el hecho de que la aplicación paralela no sufriría ninguna modificación al utilizar esta solución, mientras que se debería incurrir en dicho costo si debiera ser "gridificada".

15.4. Algoritmos de Planificación

Los planificadores para entornos Grid deben contemplar aspectos tales como heterogeneidad de recursos, adaptación dinámica a las tareas y altos costos de comunicación, por lo que los planificadores tradicionales no suelen ser una opción válida. Durante el estudio de este tipo de planificadores se han presentado características especiales de los algoritmos de planificación Grid, como la capacidad de trabajar en entornos heterogéneos y dinámicos. Según una taxonomía de algoritmos distribuidos, el tipo de estructura de planificación es distribuida-cooperativa, ya que la información y toma de decisión es compartida por los componentes de la meta organización y la organización local. Los recursos son el factor clave en la función objetivo, observando que estos trabajen de manera eficiente durante la ejecución de la aplicación y el punto de referencia para adaptar la planificación.

Como aporte de este trabajo se plasmó una propuesta a través de métodos analíticos en lo referente al tipo de implementación, partiendo desde un requerimiento de cómputo, un espacio virtual de manera casi-óptima en base a la información del entorno computacional. Buscando el balance entre efectividad y rapidez se seleccionó el algoritmo *Hill-Climbing* con *K-recomienzos*. En este caso, con una sustancial modificación sobre el algoritmo clásico, donde las *K* opciones no son aleatorias, sino una deducción en base a las características particulares del problema utilizando información de enlaces de comunicación. Esta optimización logró un substancial aumento del rendimiento sobre el tiempo total de ejecución de las aplicaciones debido a la mejor elección del conjunto de máquinas.

15.5. Simulación y Optimización

Una aplicación paralela en un entorno multi-cluster está limitada por el rendimiento de las máquinas en el cluster (*compute-bound*) o por la capacidad de la red (*communication bound*). El rendimiento máximo que puede lograr la aplicación se alcanza por la aplicación en un cluster determinado si el limite esta dado por la capacidad de cálculo. En el desarrollo para la optimización del algoritmo de planificación se describen y se implementan las técnicas utilizadas para el uso de multi-clusters en Grid, incorporando la utilización de máquinas virtuales. Asimismo, se propone una optimización sobre la adaptación dinámica del espacio virtual, con el fin de mejorar el aprovechamiento del ancho de banda y los recursos ociosos disponibles. Se utilizan características de migración en línea y la planificación dinámica en la gestión de máquinas virtuales para lograr transformar entornos limitados por la red, a entornos limitados por el cómputo, alcanzando altos rendimientos en el uso del equipamiento disponible. Para lograr un espacio de pruebas configurable, repetible y rápido, las ejecuciones de los algoritmos y heurísticas se desarrollaron sobre un simulador, implementado como parte del trabajo doctoral.

Los algoritmos de selección han sido implementados para encontrar grupos de máquinas con características especiales en un espacio de búsqueda limitado, con el compromiso de devolver una solución de manera rápida y óptima. El conjunto de estas implantaciones incrementaron el poder de cómputo casi en un 20%. Se han diseñado e implementado algoritmos dinámicos para adaptar el uso del cluster en tiempo de ejecución con la migración de máquinas virtuales. Esta implementación resulta en un 10% de mejora en el tiempo total de cómputo, con máquinas trabajando a un 100% de su tiempo asignado.

Estos resultados permiten concluir que la factibilidad de implementar la solución en entornos Grid, donde un análisis multicluster puede guiar un balance entre el computo y la comunicación, determinando una manera efectiva de regular el ancho de banda disponible impactando en el poder de computo comprometido.

En la Figura 58 se puede observar un gráfico integrando todos los conceptos analizados a lo largo de esta Tesis. De arriba hacia abajo en primer término se encuentra el punto de toma de decisión en la meta organización; este es físico, gestiona los requerimientos del entorno Grid de la meta-organización y aplica las reglas de negocio teniendo en cuenta los planificadores de las organizaciones locales. En este primer nivel se encuentra el metaplanificador, base de datos de políticas, usuarios y tareas a nivel de meta organización.

En el segundo nivel se encuentran los entornos locales, en el caso de la figura son clusters con sus respectivos grupos de máquinas. Estas máquinas están coordinadas por un planificador local que envía la información al metaplanificador y aplica las decisiones que se toman en el nivel superior. En este nivel también se encuentran los agentes que gestionan las máquinas virtuales y adaptan sus características dependiendo del estado del sistema.

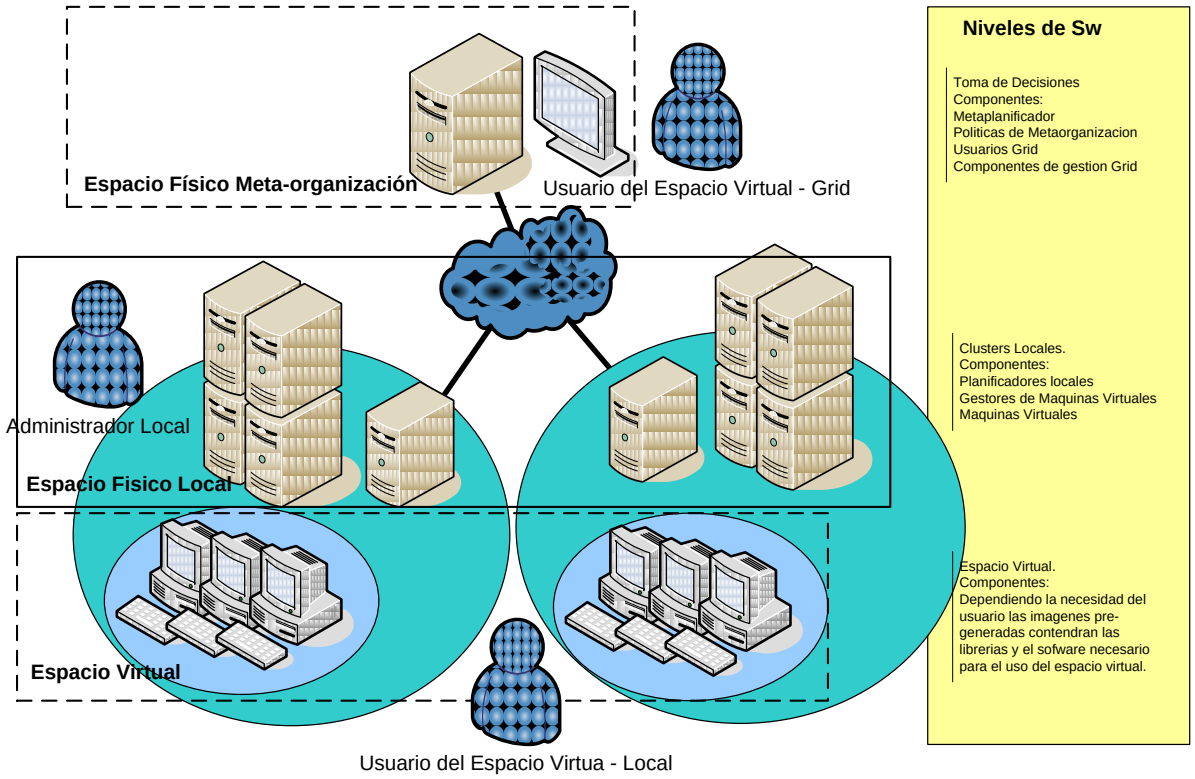


Figura 58. Línea de tiempo desarrollo de la tesis

En el nivel inferior se puede observar el espacio virtual, que es generado por el nivel de gestión superior, y desde el punto de vista del usuario final es un espacio auto contenido para ejecutar aplicaciones. Los recursos necesarios para la ejecución de aplicaciones son provistos a demanda en base a la información disponible en el sistema.

Durante el desarrollo de esta tesis se ha analizado en detalle cada uno de estos niveles y se han llevado a cabo distintas metodologías experimentales como pruebas de concepto, prototipos y la implementación de un simulador para probar la factibilidad de las soluciones propuestas teóricamente.

16. Trabajos Relacionados

Durante el desarrollo del doctorado se publicaron múltiples trabajos con el fin de compartir los conocimientos y los avances logrados en los conceptos explorados. A continuación se encuentran descriptos los resúmenes de los trabajos publicados como dirección de pasantías y dirección de tesis de grado. Los trabajos fueron derivados del trabajo del doctorado con el fin de profundizar características secundarias y de esta manera completar aspectos de usabilidad de la herramienta fundamentales para el uso práctico de las experiencias.

16.1. Caracterización de Máquinas Virtuales[2] (Virtualization Metering)

Las técnicas de virtualización proveen eficientes técnicas de aislamiento, por ejemplo fallas, seguridad y entorno para las aplicaciones. El aislamiento de fallas limita el impacto de las aplicaciones con problemas del almacenamiento y operación de otras máquinas virtuales. El aislamiento de seguridad limita el acceso a objetos, archivos, direcciones de memoria, puertos, etc. Por último, el aislamiento del entorno permite múltiples sistemas operativos operar con mínima interferencia en los recursos de cómputo asignados.

La posibilidad de compartir recursos entre diferentes máquinas virtuales implica una mayor utilización de los equipos físicos aprovechando al máximo estos recursos. Esta ventaja también puede traer inconvenientes, debido a que todos los recursos que comparten la ejecución de una máquina virtual pueden interferir en la ejecución de otra. La ejecución de una máquina virtual compartiendo el sistema con otras máquinas virtuales tienen la mayoría de las veces distintos tiempos de ejecución dependiendo de las actividades de las otras máquinas. En las investigaciones éste tema es referenciado como aislamiento de rendimiento. Depende de las distintas cargas de trabajo, configuración y tecnología de virtualización. El sistema operativo virtual y la aplicación que en ese momento está corriendo, no puede saber por la naturaleza del aislamiento el estado de trabajo de otra máquina virtual. Por esto, no se puede auto regular para lograr bajo impacto en las demás ejecuciones.

Los trabajos realizados consistieron en la caracterización del entorno virtual con el objetivo medir las interferencias de rendimiento entre las máquinas virtuales. La comparación se realizó entre OpenVZ, técnica de virtualización implementada en el sistema operativo. Xen para-virtualizado es decir realizado en máquinas sin soporte de virtualización por hardware y finalmente Xen con virtualización con soporte en hardware.

[2] Trabajo de Mentor con Alejandro Paredes. Intel Corp. 2008.

16.2. Evaluación de herramientas para el ciclo de desarrollo de aplicaciones Grid

La ejecución de tareas en un Grid requiere conocimiento de complejas sintaxis. Más aún, cuando se desea ejecutar tareas que lleven datos de un nodo a otro para su procesamiento, o cuando se desea utilizar dispositivos especializados. En el mejor de los casos, existirán pequeños programas formados por secuencias de comandos pre establecidos (scripts) y el usuario deberá utilizar una interfaz no gráfica para interactuar con ellos.

Existen diversas maneras de proveer interfaz a las aplicaciones. Esto abre las puertas a un nuevo problema: cada usuario debe lidiar con una interface distinta según la aplicación que esté utilizando. De este modo un mismo usuario interactúa con distintas interfaces para cada aplicación que utiliza en la realización de sus tareas. En este sentido se pierden los beneficios de Grid de proveer una visión integrada de los sistemas y organizaciones. Para resolver este último problema de diversas interfaces para un mismo usuario sería deseable desarrollar un entorno de trabajo integrado. Este entorno podría ser el punto de acceso a distintos recursos: aplicaciones, información y servicios. Todos estos recursos estarían disponibles bajo una misma interfaz. Si además se sumara la capacidad de personalización entonces el usuario podría ajustar el entorno de trabajo a sus necesidades y preferencias optimizando aun más su interacción con los sistemas. Los portales son la herramienta que brinda estos y otros servicios. Por lo tanto desarrollar aplicaciones Grid integradas a los entornos de los portales resolvería el problema de las interfaces.

Este trabajo desarrolló la factibilidad de desarrollar aplicaciones Grid en el contexto de portales a través de la evaluación de los componentes necesarios para la implementación de esta solución. La evaluación se realizó a través de la experimentación, implementando distintos portales Grid con la publicación de un servicio de *rendering* para gráficos en movimiento, logrando una comparación efectiva de las distintas metodologías de integración y publicación de dicho servicio.

16.3. Integración de datos

Un aspecto fundamental de la tecnología Grid es el procesamiento de grandes volúmenes de datos. Este trabajo estuvo enfocado en dar soporte para las consultas paralelas sobre múltiples bases de datos. La herramienta seleccionada para realizar estas tareas fue OGSA-DAI. El objetivo de esta herramienta es poder compartir los datos entre los recursos para permitir la colaboración. La funcionalidad que fue implementada durante la fase de experimentación fue:

- Acceso a datos: Acceso a datos estructurados en recursos de información heterogéneos y distribuidos.
- Transformación de datos: por ejemplo usar un esquema de datos origen y poder mostrarlo usando diferentes vistas en un esquema destino.
- Integración de datos: por ejemplo exponer al usuario a múltiple fuentes de datos como una sola base de datos virtual.
- Distribución de datos: seleccionar automáticamente la forma más apropiada de distribución de información (por ejemplo servicios web, e-mail, HTTP, FTP, GridFTP).

Para lograr ésto, OGSA-DAI es un marco de trabajo que ejecuta flujos de actividades. Cada actividad es una unidad funcional con datos, una tarea se realiza en esa actividad y los datos son procesados obteniendo una salida. Los flujos de actividades son enviados a través de clientes por servicios web, son adaptados por funcionalidad y finalmente los repositorios de información, como motores de bases de datos, pueden interactuar con estos servicios. Esta arquitectura fue diseñada para trabajar como un kit para construir aplicaciones de alto nivel.

En el trabajo de tesis se utilizo para poder realizar consultas en paralelo sobre información en distintas bases de datos como Oracle, IBM DB2, MySQL, Postgres donde se almacenaban ontologías desde distintas ubicaciones geográficas.

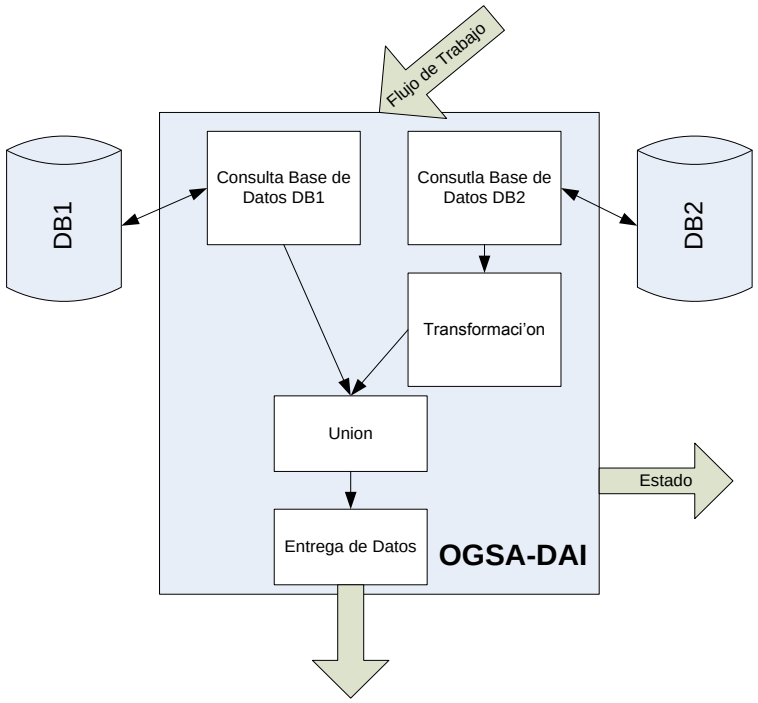


Figura 59. Integración de datos usando un solo flujo de datos

17. Publicaciones

En esta sección se enumeran todas las publicaciones que dieron origen y fundamentaron durante el desarrollo el trabajo doctoral. En un principio las publicaciones se encuentran relacionadas con conceptos generales sobre laboratorios remotos, y gradualmente se profundiza la publicación de aspectos particulares como planificación y algoritmos aspectos tratados durante el desarrollo propiamente dicho del doctorado. En cada caso se agruparon por congreso y año de publicación.

Wicc 2004
Título: **Prácticas Remotas sobre Laboratorios Físicos y Virtuales**
Resumen: Debido a la complejidad de acceso y diversidad en los distintos tipos de recursos involucrados, los entornos de capacitación a distancia tienen una deficiencia en cuanto a la realización de prácticas de laboratorio en forma remota. En este trabajo se plantea un marco de trabajo para la definición y utilización remota de laboratorios físicos y

virtuales. Esto abarca: la creación de plantillas de trabajo de los laboratorios basadas en sus correspondientes entornos, un sistema para el control, administración y seguimiento de las distintas prácticas y sus asistentes, así como también la posibilidad de incorporar métodos de evaluación para los distintos niveles de enseñanza.

MECOM 2005
Título: **Grid computing. Panorama y actividades en argentina**
Resumen: La tecnología denominada Grid Computing es un entorno persistente que permite la integración de recursos informáticos administrados por diversas organizaciones geográficamente alejadas, asumiendo la ausencia de una ubicación central de control. GRID reviste gran interés en otros lugares del mundo como Europa y USA e incluso en Brasil, aunque es poco conocida en nuestro país. La puesta en marcha de un entorno GRID es una tarea de gran envergadura para permitir que se compartan de manera segura y eficaz los recursos a disposición de la GRID. En este trabajo se presenta la tecnología, se introducen sus antecedentes y se muestran distintas aplicaciones y experiencias realizadas por los grupos de LSC, LAPIC y Universidad del Comahue.

CACIC 2005
Título: **Evaluación de herramientas para el desarrollo de servicios Grid**
Resumen: Grid posibilita la creación de sistemas que aprovechan esta capacidad dando lugar a sistemas distribuidos que antes eran demasiado complejos, tanto desde el punto de vista de sus objetivos como la calidad de servicio con la que pueden alcanzar estas metas. En este trabajo se explora y evalúan distintas herramientas para el desarrollo de los componentes básicos de esta tecnología, los servicios Grid.

Título: **Framework for GRID Metascheduling with SLAs**
Resumen: La integración de recursos heterogéneos en diferentes dominios administrativos hace que del control y gestión una tareas compleja, y esto puede ser aún peor si las organizaciones intentan usar esos recursos de manera coordinada. El objetivo de este trabajo es simplificar estas tareas con un esquema de gestión basado en políticas. Las políticas con alto nivel de abstracción se transforman automáticamente en reglas de negocio y se aplican en los recursos apropiados. Por lo que se define un marco de trabajo y aspectos de diseño dando un ejemplo de cómo esa gestión puede ser llevada a cabo. También se describe un meta planificador y como este esquema puede ser fácilmente integrado.

Título: **Arquitectura para Laboratorios Remotos Físicos y Virtuales**
Resumen: Los entornos de capacitación y operación a distancia presentan problemáticas complejas debido a la dificultad del acceso y la diversidad en los tipos de recursos involucrados. En este trabajo se describe un marco de trabajo para la definición y utilización remota de laboratorios físicos y virtuales, así como también detalles de implementación en aspectos como la interacción cliente-recurso, seguridad o disponibilidad. Se han definido y desarrollado prototipos sobre laboratorios con dispositivos de interconexión de redes de computadoras. Esto ha llevando al diseño e implementación de nuevos aspectos de la herramienta dando como resultado la extensión a nuevos conjuntos de laboratorios.

Wicc 2006
Título: **Entorno Colaborativo sobre Laboratorios Remotos.**

Resumen: Dentro del proyecto de investigación de procesos para software colaborativo existe una línea sobre el desarrollo de experiencias en laboratorios remotos, en esta línea se incluye distintas acciones destinada a profundizar sobre el complemento de las tareas de aprendizaje con un soporte tecnológico de sistemas. Esta relación se entiende como el conjunto de todas aquellas actividades de infraestructura que promuevan la disponibilidad y uso de recursos, además de escalabilidad de procesos en la tarea educativa apoyada en tecnologías de información. En este trabajo se describen cada uno de los grupos de trabajo que participan de este proyecto analizando las actividades que actualmente se desarrollan así como también perspectivas futuras.

Título: **Optimización de planificación dinámica sobre entornos Grid.**
La integración de recursos heterogéneos dispersos geográficamente sobre distintos dominios administrativos hace que la gestión de estos entornos sea una tarea difícil. Si agregamos el requerimiento de que estos recursos puedan trabajar de manera coordinada y eficiente, aumenta aún más la complejidad de la tarea. En este trabajo se presenta una línea de investigación sobre esta problemática caracterizada por los entornos Grid, teniendo como objetivo lograr una metodología para la administración automática basada en parámetros de eficiencia y calidad.

CACIC 2006
Título: **Evaluación de tecnologías para la publicación de servicios Grid.**
Resumen: Un importante problema que encuentran las aplicaciones Grid es la distancia que existe entre estas y los usuarios. Esta brecha puede ser acortada con el desarrollo de interfaces de usuario que escondan las complejidades del Grid. En este sentido los portales proveen una interfaz de usuario amigable y conocida que ayuda con esta tarea. El presente trabajo evalúa portales, portlets y frameworks de portales Grid que pueden ser usados en el desarrollo de aplicaciones Grid con el objetivo de acercar al usuario final la tecnología Grid.

Wicc 2007
Título: **Laboratorios Remotos sobre Espacios Virtuales**
Resumen: Las nuevas tecnologías de Internet permiten el uso de sistemas de software distribuido que proporcionan a los usuarios el acceso en forma remota a laboratorios físicos y virtuales, para llevar a cabo actividades que normalmente son realizadas localmente. Para que el acceso remoto a estos laboratorios sea posible, se debe diseñar y construir una arquitectura de software y hardware que provea las interfaces adecuadas a los usuarios que quieran acceder. En este trabajo se describen las actividades de investigación llevadas adelante en el departamento de Cs. de la Computación dentro del marco de laboratorios remotos, y puntualmente en lo que se refiere a los espacios virtuales generados para la implementación y el uso de los mismos.

Ibergrid 2007
Título: **Parallel Algorithms on multi-cluster architectures using GRID middleware. Experiences in Argentine Universities**
Resumen: Este trabajo presenta una línea de investigación y desarrollo, en el que tareas de cómputo son llevadas a cabo por algoritmos paralelos en arquitecturas multicluster y su evolución hacia Grid es llevada a cabo en el laboratorio III-LIDI (Universidad de La Plata, Argentina). Se presentan los temas fundamentales en relación al desarrollo y

evaluación de algoritmos paralelos en arquitecturas distribuidas débilmente, el empleo de *middleware* Grid como soporte de esquemas multicluster, y el desarrollo de predicción de modelos de rendimiento para estas arquitecturas. El desarrollo es una actividad en conjunto de tres universidades argentinas (UN Comahue, UN Sur, and UN San Luis). Asimismo, el trabajo es realizado con universidades Ibero-Americanas participando en el proyecto CyTED “Grid Technology as a Regional Development Engine”, principalmente con la Universidad Autónoma de Barcelona, Universidad Politécnica de Valencia, y Universidad Complutense de Madrid.

CACIC 2007
Título: **Planificación dinámica de clusters a demanda en entornos Grid**
Resumen: Debido a la gran cantidad de recursos de hardware y software que componen los sistemas Grid, cada uno con diferentes características y complejidades, se torna imperioso simplificar y automatizar su administración. En este trabajo como un primer paso se presentan dos niveles de planificación, el primero a nivel de meta-organización, con el objetivo de generar clusters a demanda basado en requerimientos de aplicaciones y el segundo a nivel de organización local, gestionando los recursos del cluster en forma dinámica para lograr un máximo aprovechamiento de los recursos ofrecidos. Como recursos se estudian máquinas virtuales que interconectadas forman un cluster homogéneo entre organizaciones.

VHPC 2007
Título: **Grid Virtual Laboratory Architecture**
Resumen. Este trabajo describe un método para gestionar recursos en red virtuales y físicos bajo una organización virtual basada en Grid, vistas desde componentes elementales de un laboratorio remoto virtual. Manteniendo la sobrecarga sobre las organizaciones locales al mínimo, buscando una configuración automática de los recursos disponibles, además de proveer acceso interactivo a estos recursos desde interfaces de Internet. Estos recursos pueden ser involucrados es tareas como computo paralelo, simulación de laboratorios, etc. Dos casos de test fueron implementados para probar la factibilidad del concepto.

WICC 2008
Título: **Algoritmos de planificación dinámica en entornos Grid**
Resumen: Debido a la gran cantidad de recursos de hardware y software que componen los sistemas Grid, cada uno con características y políticas propias, la gestión del sistema se torna compleja. Un aspecto clave que permite obtener la máxima potencia del sistema de manera eficiente se encuentra en el algoritmo de planificación de tareas. Debido al dinamismo de los recursos y las organizaciones que los contienen los planificadores tradicionales no son una opción válida. Los nuevos planificadores deben contemplar aspectos tales como heterogeneidad de recursos, adaptación dinámica al entorno y altos costos de comunicación. En este trabajo se presenta una línea de investigación sobre algoritmos de planificación Grid, se analiza una propuesta de planificación y se describen aspectos que se encuentran en desarrollo.

VHPC 2008
Título: **Dynamic on Demand Virtual Clusters in Grid**
Resumen: La contribución de este trabajo consiste en la extensión e implementación de un metaplanificador Grid que descubre dinámicamente, crea y administra clusters a

demanda de manera virtual. El primer módulo describe la selección de equipos usando heurísticas de grafos, el algoritmo trata de encontrar una solución buscando sobre los cluster mapeándolos a un grafo, obteniendo el mejor rendimiento para la tarea dada. El segundo módulo, uno por nodo Grid, monitorea y gestiona máquinas físicas y virtuales. Cuando una nueva tarea arriba, estos módulos modifican dinámicamente las máquinas virtuales o usan las características de migración en línea para adaptar la distribución de recursos en el cluster logrando máxima utilización. Los componentes del metaplanificador y módulos en el gestor local trabajan en conjunto para tomar decisiones en tiempo de ejecución para balancear y optimizar el trabajo generado por el sistema. Esta implementación resulta en una mejora de rendimiento de aproximadamente 20% sobre el tiempo total de computación con una utilización del 100% de los recursos. Estos resultados nos permiten concluir que la solución es factible para ser implementada en entornos Grid, donde la automatización y autocontrol del sistema son esenciales para un uso efectivo de los recursos.

TEYET Journal Nro 4.
Título: **Infraestructura para laboratorios de acceso remoto**
Resumen: Este trabajo define una infraestructura para la implementación de aplicaciones para el acceso remoto a laboratorios físicos y virtuales y para la gestión de los mismos. Se presentan conceptos que permiten extender la modalidad tradicional de conexión con un dispositivo, para llevar adelante actividades prácticas de laboratorio colaborativas y concurrentes.

18. Estado actual del arte

Al comienzo de esta tesis la tecnología Grid se encontraba en un proceso de pleno desarrollo con grupos de investigación alrededor del mundo desarrollando trabajos para el despliegue del entorno. Actualmente se siguen investigando aspectos puntuales como es el de la planificación para entornos de investigación y desarrollo de dominio específico. Pero el concepto ha derivado en nuevas ideas; esta vez no necesariamente llevadas a cabo por universidades o laboratorios, sino por grandes empresas que han desplazado el foco de los postulados Grid con un objetivo comercial. Si bien el objetivo es otro, la tecnología de soporte sigue siendo muy similar, los conceptos que en este trabajo se han propuesto y probado fácilmente pueden ser implementados en este tipo de entorno que se ha dado en llamar Cloud Computing [79, 81, 82].

18.1. Cloud Computing

Puede el termino Cloud Computing decirse que es el nuevo termino para Grid Computing comercial?
No hay una respuesta única. Sí la visión es la misma. Reducir el costo de la computación, incrementar la confiabilidad e incrementar la flexibilidad transformado computadoras que operamos nosotros mismos a ser operadas por terceros. Pero no, porque el objetivo ha cambiado. El beneficio de transformar mainframes a cluster es evidente, pero los cluster son difíciles de operar. El bajo costo de la virtualización y la incorporación de cientos de miles de máquinas por Amazon, Google y Microsoft en grandes centros de datos distribuidos por el mundo para crear sistemas a gran escala genera un tipo de problema totalmente diferente. Pero finalmente sí, el problema es básicamente el mismo en Grid y Clouds. Hay dificultades para gestionar grandes instalaciones, definir métodos para que los usuarios encuentren, soliciten y usen los recursos provistos y generalmente sobre

estos recursos se ejecutan usando computación paralela. Los detalles difieren, pero las dos comunidades sufren esencialmente de los mismos inconvenientes.

Por lo que para una definición de Cloud Computing se puede describir los siguientes puntos:

1. Es masivamente escalable.
2. Puede ser encapsulada en entidades abstractas que puede ofrecer diferentes niveles de servicio a los usuarios fuera del Cloud.
3. El objetivo es lograr una economía de escala.
4. Los servicios se configuran dinámicamente (generalmente vía virtualización) y entregados a demanda.

Algunas de las características de Cloud Computing se superponen con tecnologías existentes como Grid, Utility Computing y Service Computing o los sistemas Distribuidos en General. Las discusiones todavía no se han asentado pero el autor de esta tesis adhiere a la corriente que postula que Cloud Computing no se superpone a Grid, sino que es una evolución de Grid y esta soportada por Grid como infraestructura. La evolución del concepto ha sido debido al cambio de foco, solo una infraestructura que ofrece poder de cómputo y almacenamiento, como Grid, a una basada en la economía para entregar recursos y servicios pensando en la publicación de escala no de rendimiento. En la figura 60 se puede observar una relación entre Cloud y otros dominios. Web 2.0 cubre todo el espectro de aplicaciones orientadas a servicios.

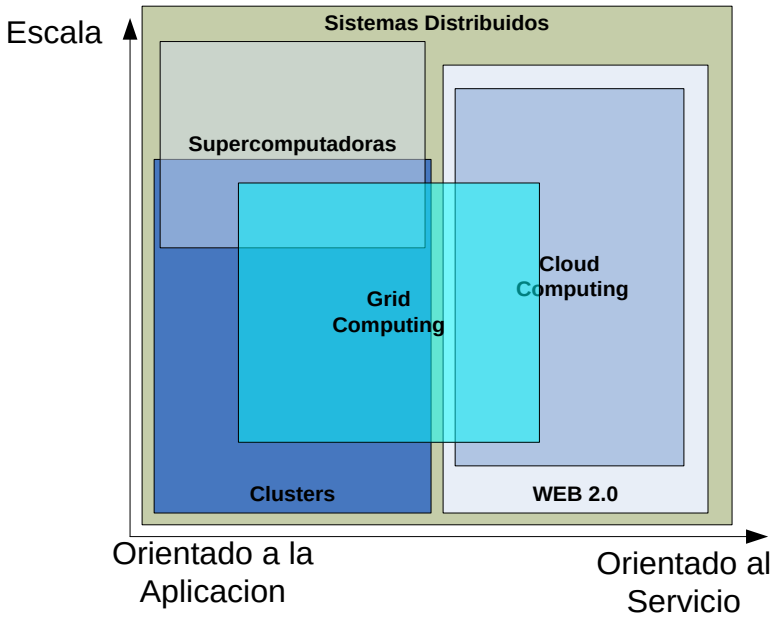


Figura 60. Grids and Clouds Overview

18.2. Computación orientada al servicio (SOC)

Otro de los ejemplos de proyectos actualmente en ejecución con una fuerte inversión en investigación es el proyecto RESERVOIR [83, 84]. Este proyecto intenta dar soporte a la computación orientada al servicio como un nuevo paradigma de la computación. En este paradigma los servicios son componentes de software accesibles por Internet siendo

independientes de la plataforma, el lenguaje y las interfaces. Además se pueden componer en aplicaciones distribuidas bajamente acopladas. Estos servicios pueden llegar a ser elementos de una economía en línea, componentes de negocio o actividades de gobierno como comercio electrónico a través de organizaciones, aplicaciones, sistemas de telecomunicación, etc.

Esta orientación permite bajar los costos y complejidad, llevando soluciones más rápidamente al mercado, aumentando la confiabilidad y mejorando el acceso a recursos que pueden estar restringidos para algunos segmentos como el gobierno. Pero uno de los desafíos es lograr una infraestructura capaz de soportar estos requerimientos, y el proyecto RESERVOIR intenta dar solución dentro de la comunidad europea con financiación estatal.

Para lograr este objetivo se combinan e integran tres tecnologías: virtualización, computación Grid y gestión para servicios de negocio. Uno de los aspectos de investigación es la tecnología de virtualización que ha sido vista como una solución factible para algunas barreras para la adopción comercial. Algunas de las opciones que se están investigando es la migración a través de distintos dominios administrativos, por ejemplo usar sistemas de seguimiento para la toma de decisiones. Como puede observarse todos los conceptos analizados en esta tesis aún siguen siendo tema de investigación pero con implementaciones específicas en funcionamiento.

18.3. Infraestructura como servicio.

En esta sección analizaremos la implementación libre más conocida para la generación de Cloud Computing y la compararemos con la solución propuesta en esta tesis con el fin de evidenciar la similitud de Eucalyptus [80] (una solución pensada para Cloud Computing) y el framework pensado para una solución de Grid Computing.

Eucalyptus fue diseñado con el fin de que sea sencillo de instalar y no intrusivo, pudiendo usar recursos no exclusivos. Es altamente modular, siguiendo estándares de la industria, con mecanismos de comunicación agnóstica al lenguaje. La interface externa está basada en la API EC2 de Amazon. Por último este conjunto de aplicaciones es único en el mundo open source ya que provee una red virtual aislada exponiendo al usuario un único espacio de direcciones virtual disponible para el servicio en ejecución.

La arquitectura de Eucalyptus es simple, flexible y modular reflejando un diseño jerárquico común en los entornos académicos. En esencia el sistema permite arrancar, controlar y acceder máquinas virtuales usando una emulación de Amazon EC2 SOAP. Actualmente soporta hypervisores Xen y KVM/QEMU. Las interfaces de alto nivel son publicadas por servicios web. Esta característica permite publicar APIs independientes del lenguaje en formato WSDL conteniendo operaciones y estructuras de datos. Finalmente, las políticas de seguridad son usadas por WS-Security estándar.

Eucalyptus consta de cuatro componentes principales cada uno con su propia interface web, a saber:

- Controlador de Nodos: Controla la ejecución, inspección y finalización de las instancias de VM en la máquina servidor durante la ejecución.

- Controlador de Cluster: Obtiene la información sobre los planificadores en los componentes controladores de nodos, y gestiona la configuración de red entre las máquinas virtuales.
- Controlador de Almacenamiento: Este es un servicio de almacenamiento que implementa la interfaz de Amazon S3, proveyendo mecanismos de almacenamiento, acceso a imágenes de máquinas virtuales y datos de usuario.
- Controlador de Cloud: Es el punto de entrada para los usuarios y administradores. Realiza peticiones a los nodos gestores para información por los recursos, hace decisiones de planificación de alto nivel que son implementadas por los componentes controladores de clusters.

En la Figura 61 se puede observar la distribución jerárquica de los componentes descriptos.

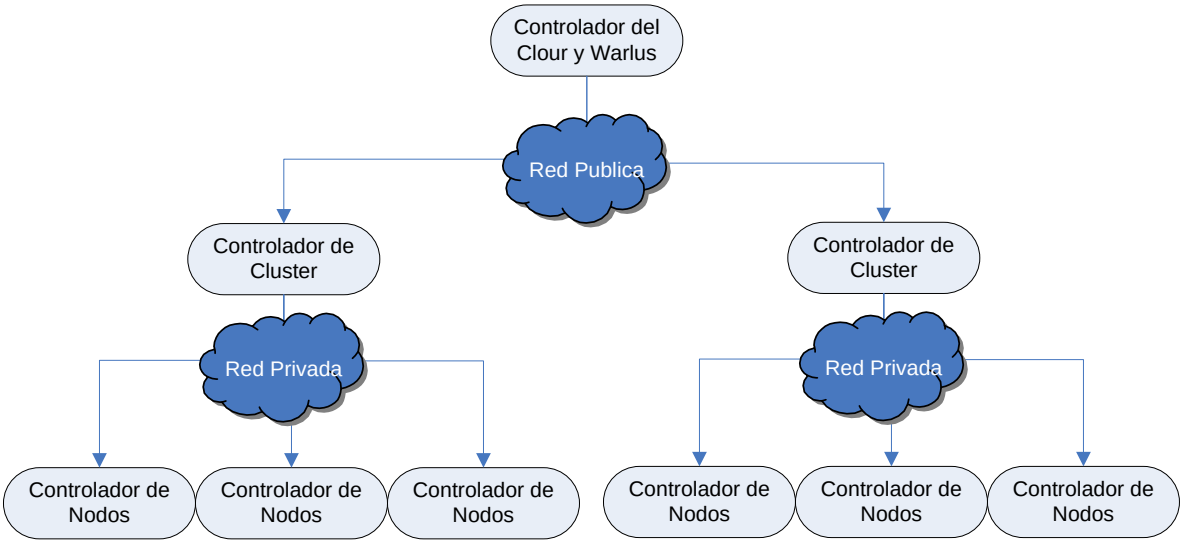


Figura 61. Estructura jerárquica de componentes de Eucalyptus

En la Tabla 9 se puede observar cómo cada una de las capas identificadas en Eucalyptus tiene su correspondencia en la propuesta Grid de esta tesis y la solución propuesta en cada caso es similar, concluyendo de esta manera que si bien el objetivo de la propuesta de Cloud Computing ha variado hacia un aspecto económico la infraestructura que la soporta es similar a las implementaciones realizadas en Grid Computing.

Componentes	Eucalyptus	Propuesta Grid
Capa de toma de Decisiones	*Controlador del Cloud, acceso a través de API de Amazon o portal web. Manejo de datos Warlus compatible con Amazon*	*Metaplanificador extendiendo CSF interfaz propia publicada por servicios web. Gestión de datos y control de usuarios utilizando servicios web y clientes provistos por el estándar de facto GT4.x*

Capa de Gestión del espacio físico	*Desarrollo propio con interfaz compatible con Amazon, soporte Xen, KVM. Acceso a planificadores locales*	*Desarrollo de agentes propios con interfaz interfaces usando librerías libvirt compatibles xon Xen, Qemu, VMWare, Vritual Box, etc. Acceso a planificadores locales en los clusters. Adaptación dinámica de máquinas virtuales según los requerimientos*
Capa del espacio Virtual	Espacio generado por las máquinas virtuales disponibles, dependiente de las imágenes generadas.	Espacio generado por las máquinas virtuales disponibles, dependiente de las imágenes generadas.

Tabla 9. Comparación Eucalyptus vs Propuesta Grid

Bibliografía

1. I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International Jounral of Supercomputer Applications*, vol. 15, no. 3, 2001.
2. I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," 2002.
3. Luis Ferreira, Fabiano Lucchese et at.. Grid Computing in research and Education. IBM Redbooks. 2005.
4. Austin, Jackson, et al, Predictive Maintenance: Distributed Aircraft Engine Diagnostics, in The Grid: 2nd Edition, edited by Ian Foster and Carl Kesselman. MKP/Elsevier , Oct 2003
5. Stevens, R. and Future Lab Group, Group-Oriented Collaboration: The Access Grid. Collaboration System, in The Grid: 2nd Edition, edited by Ian Foster and Carl Kesselman. MKP/Elsevier , Oct 2003
6. Alexander S. Szalay and Jim Gray A .Scientific Data Federation: The World-Wide Telescope, in The Grid: 2nd Edition, edited by Ian Foster and Carl Kesselman. MKP/Elsevier , Oct 2003
7. Luis Ferreira, Arun Thakore et at. Grid Services Programming and Application Enablement. IBM Redbooks. 2004.
8. Borja Sotomayor, Lisa Childers. Globus® Toolkit 4: Programming Java Services. December 2005, Morgan Kaufmann Publishers.
9. J. Nabrzyski, J. Schopf and J. Weglarz, eds. Grid Resource Management - State of the Art and Future Trends. Kluwer Academic Publishers, 2004
10. OpenPBS Project, A Batching Queuing System, Software Project, Altair Grid Technologies, LLC, www.openpbs.org [Online] http://dcwww.camd.dtu.dk/pbs.html
11. LSF Administrator’s Guide, Version 4.1, Platform Computing Corporation, February 2001. [Online] http://batch.web.cern.ch/batch/doc-lsfsets.html
12. Rajesh Raman, Miron Livny, and Marvin Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing", Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998
13. B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, "Xen and the art of virtualization," in Proceedings of the ACM Symposium on Operating Systems Principles, October 2003.
14. Keahey, K., T. Freeman . Contextualization: Providing One-Click Virtual Clusters,. To be presented: eScience 2008, Indianapolis, IN. December 2008.
15. Christina Hoffa, Gaurang Mehta, Tim Freeman, Ewa Deelman, Kate Keahey, Bruce Berriman, John Good, "On the Use of Cloud Computing for Scientific

Workflows," escience,pp.640-645, 2008 Fourth IEEE International Conference on eScience, 2008.

16. K. Keahey, I. Foster, T. Freeman, X. Zhang and D. Galron, Virtual Workspaces in the Grid, in Europar, Lisbon, Portugal, 2005
17. Keahey, K., Foster, I., Freeman, T., Zhang, X. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. CCGRID 2006, Singapore, May 2006.
18. I. Foster, "What is the grid? - a three point checklist," GRIDtoday, vol. 1, no. 6, July 2002.
19. OASIS. [online] http://www.oasis-open.org
20. Global Grid Forum [online]. http://www.ggf.org
21. E. Argollo, A. Gaudiani, D. Rexachs y E. Luque. ”Predicción del computo paralelo de una aplicación sobre una colección de clusters geográficamente distribuidos” In CACIC 2005. Concordia Argentina, 2005.
22. E. Argollo, D. Rexachs and E. Luque, “Efficient Execution of Scientific Computation on Long-distance Geographically Distributed Clusters”, PARA'04 State-of-the-Art in Scientific Computing June 20-23, 2004.
23. E. Argollo A. Furtado, J. Souza, A. Rebouças, D. Rexachs and E. Luque, “Application Execution Over a CoHNOWS”, In: International Conference on Computer Science, Software Engineering, Information Technology, e-business, and Application (CSITeA'03), 2003
24. A. Furtado, J. Souza, A. Rebouças, D. Rexachs y E. Luque, “Architectures for an Efficient Application Execution in a Collection of HNOWS”, In: D. Kranzlmüller et al. (Eds.): Euro PVM/MPI 2002, LNCS 2474, pp. 450-460, 2002.
25. A. Tirumala, M. Gates, F. Qin, J. Dugan and J. Ferguson. “Iperf - The TCP/UDP band-width measurement tool”. [Online]. Available:http://dast.nlanr.net/Projects/Iperf.
26. I. Foster. “A globus Premier, Describing Globus Toolkit Version 4” Draft, 2005.
27. M. Carusela, R. Perrazo, L. Romanelli. “Stochastic resonant memory storage device”, Phys. Rev. E 64 031101, 2001.
28. M. Carusela, R. Perazzo, L. Romanelli, “Information transmission and storage sustained by noise”. Elsevier Science B.V. Physica D. 177-183, August 2002.
29. Eduardo Argollo, Adriana Gaudiani, Dolores Rexachs, Emilio Luque.”Tuning Application in a Multi-cluster Environment”. EuroPar'06. 2006.
30. Log 4j. Servicio de monitoreo. [Online]. Available: http://logging.apache.org/log4j /docs/
31. Netperf. network performance project. [Online]. Available: http://www.netperf.org
32. M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Parallel Computing, vol. 30, no. 7, July 2004.

33. Douglas Thain and Miron Livny: Condor and the Grid. in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making the Global Infrastructure a Reality, John Wiley, 2003

34. Graham,S. et.al., Publish-Subscribe Notification for Web services, Version 1.0, 2004, [Online]. Available: http://www.oasis-open.org/committees/download.php/6661/WSNpubsub-1-0.pdf

35. Yavatkar, R., Pendarakis, D. and R. Guerin, "A Framework for Policy-based Admission Control", RFC 2753, January 2000.

36. Jeffrey O. Kephart, David M. Chess, ”The Vision of Autonomic Computing”, IEEE Computer 36(1): 41-50 2003.

37. I. Foster et al., “The Grid2003 Production Grid: Principles and Practice”, Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, 2004.

38. Moore, B., Ellesson, E., Strassner, J. and A. Westerinen, "Policy Core Information Model -- ", RFC 3060, February 2001.

39. Jean-Christophe Martin. Policy-Based Networks. Sun BluePrints. OnLine, October 1999.

40. Czajkowski, K., Foster, I., Kesselman, C.,Sander, V. and Tuecke, S., SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. in 8th Workshop on Job Scheduling Strategies for Parallel Processing, (2002).

41. Dumitrescu, C. and I. Foster. Usage Policy-based CPU Sharing in Virtual Organizations. in 5th International Workshop in Grid Computing. 2004.

42. Modeling Stateful Resources With Web Services. http://www-106.ibm.com /developerworks/library/ws-resource/ws-modelingresources.pdf

43. The Web Services Resource Framework.

44. http://www-106.ibm.com/developerworks /library/ws-resource/ws-wsrfpaper.html

45. IBM, WSLA Language Specification, Version 1.0. 2003.

46. K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke and M. Xu.Agreement-based Grid Service Management (OGSIAgreement) Version 0. https://forge.gridforum.org /projects/graap-wg/document/Draft_OGSI-agreement_Specification/en/1/Draft_ OGSI-Agreement_Specification.doc

47. Providing Data Transfer with QoS as Agreement-Based Service, H. Zhang, K. Keahey,W. Allcock. 2004.

48. MDS, http://www.globus.org

49. GLUE, http://www.cnaf.infn.it

50. Massie, M., B. Chun, and D. Culler. The Ganglia Distributed Monitoring: Design, Implementation, and Experience. in ParallelComputing. May 2004.

51. Hawkeye monitoring tool, http://www.cs.wisc.edu/condor/hawkeye/

52. Community Scheduler Framework. http://sourceforge.net/projects/gcsf/

53. Sun Grid Engine. http://gridengine.sunsource.net/

54. Sanjay Kamat, R. Rajan, D. Verma, E. Felstaine, and S. Herzog: A Policy Framework for Integrated and Differentiated Services in the Internet in Special Issue of IEEE Network Magazine on Integrated and Differentiated Services in the Internet, September 1999.

55. OASIS “An Introduction to WSDM”, February,2006, http://www.oasisopen.org/committees/download.php/16998/wsdm-1.0-intro-primer-cd-01.doc.

56. Paul T. Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy L. Harris, Alex Ho, Rolf Neugebauer “Xen and the Art of Virtualization”, SOSP 2003 , Pages 164-177.

57. KVM: “Kernel Based Virtual Machine”. http://kvm.qumranet.com/kvmwiki

58. Fabrice Bellard. "QEMU, a fast and portable dynamic translator", Proceedings of USENIX. April 2005.

59. D.C. Verma, "Simplifying network administration using policy-based management," Network, IEEE, vol. 16, no. 2, pp. 20-26, 2002.

60. Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen.”Live Migration of Virtual Machines”. Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation. 2005.

61. J.O. Kephart and D.M. Chess, The vision of autonomic computing, IEEE Computer 36(1) (2003) 41–503.

62. Chase, J., L. Grit, D. Irwin, J. Moore, and S. Sprenkle, “Dynamic Virtual Clusters in a Grid Site Manager”. accepted to the 12th International Symposium on High Performance Distributed Computing (HPDC-12), 2003.

63. Paul Ruth, Phil McGachey, Dongyan Xu, "VioCluster: Virtualization for Dynamic Computational Domains", Proceedings of the IEEE International Conference on Cluster Computing (Cluster'05), 2005.

64. Sumalatha Adabala, Vineet Chadha, Puneet Chawla, Renato Figueiredo, Jose A. B. Fortes, Ivan Krsul, Andrea Matsunaga, Mauricio Tsugawa, Jian Zhang, Ming Zhao,Liping Zhu, Xiaomin Zhu, “From Virtualized Resources to Virtual Computing Grids: The In-VIGO System”, In Future Generation Computing Systems, 2004.

65. Platform Computing Co. Open source metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF)[WP]. http://www.cs.virginia.edu/grimshaw/CS851-2004/Platform/CSF_architecture.pdf , 2004

66. Cisco_7200_Simulator. [Online] http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator

67. R. Davoli. "VDE: Virtual distributed ethernet". In Proceedings of Tridentcom, 2005, Trento. 2005.

68. GSI-Sshterm. [Online] https://sourceforge.net/projects/gsi-sshterm/

69. Net-Cat. [Online]. http://netcat.sourceforge.net/

70. F. Dong and S. G. Akl, “Scheduling Algorithms for Grid Computing: State of the Art and Open Problems,” Technical Report of the Open Issues in Grid Scheduling Workshop, School of Computing, University Kingston, Ontario, January 2006

71. A. Andrieux, D. Berry, J. Garibaldi, S. Jarvis, J. MacLaren, D. Ouelhadj and D. Snelling, “Open Issues in Grid Scheduling”, Official Technical Report of the Open Issues in Grid Scheduling Workshop, Edinburgh, UK, October 2003

72. T. Casavant, and J. Kuhl, A Taxonomy of Scheduling in General-purpose Distributed Computing Systems, in IEEE Trans. on Software Engineering Vol. 14, No.2, pp. 141--154, February 1988.

73. R. Buyya and D. Abramson and J. Giddy and H. Stockinger, Economic Models for Resource Management and Scheduling in Grid Computing, in J. of Concurrency and Computation: Practice and Experience, Volume 14, Issue.13-15, pp. 1507-1542, Wiley Press, December 2002.

74. A. YarKhan and J. J. Dongarra. Experiments with Scheduling Using Simulated Annealing in a Grid Environment. In Grid, November 2002.

75. S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 2nd ed. Prentice Hall, December 2002.

76. Klaus Müller. “Advanced systems simulation capabilities in SimPy”. Presented at EuroPython 2004 in Gothenburg, Sweden.

77. Ellson, John, Emden Gansner, Eleftherios Koutsofios, Stephen North, and Gordon Woodhull. “Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools. Graph Drawing Software” pp. 127-148, Springer-Verlag. 2003.

78. Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in Performance Analysis of Systems & Software, 2009. ISPASS 2009. IEEE International Symposium on, 2009, pp. 200-209.

79. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," Dec 2008. [Online]. Available: http://arxiv.org/abs/0901.0131

80. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), vol. 0. Washington, DC, USA: IEEE, May 2009, pp. 124-131. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.93

81. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep., February 2009.

82. L. M. Vaquero, L. R. Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," SIGCOMM Comput. Commun. Rev., vol. 39, no. 1, pp. 50-55, 2009. [Online]. Available: http://dx.doi.org/10.1145/1496091.1496100

83. B. Rochwerger, A. Galis, E. Levy, J. A. C‡ceres, D. Breitgand, Y. Wolfsthal, I. M. Llorente, M. Wusthoff, R. S. Montero, E. Elmroth "RESERVOIR: Management Technologies and Requirements for Next Generation Service Oriented

Infrastructures" - 11th IFIP/IEEE International Symposium on Integrated Management 1-5 June 09, New York, USA; http://www.ieee-im.org/2009/

84. RESERVOIR http://www.reservoir-fp7.eu/home/

Anexo A. Ejecuciones.

Ejecución Heurística y Simulación para tipo de tarea similar y I/O baja, arribo exponencial.

	Heurística Tiempos de Simulación					
Requerimient	Inicio	Finalizacion	Heuristica	Atendido	Espera Heuristica	Ejecucion Heuristica
0	0	140.32	140.32	3	3	137.32
1	0.24	245.89	245.65	6	5.76	239.89
2	13.92	850.85	836.93	16.92	3	833.93
3	111.75	518.32	406.57	140.32	28.57	378
4	228.92	663.96	435.04	245.89	16.97	418.07
5	235.13	1079.82	844.69	246	10.87	833.82
6	312.76	895.92	583.16	518	205.24	377.92
7	354.27	1071.16	716.89	664	309.73	407.16
8	444.42	1684.79	1240.37	850	405.58	834.79
9	449.13	1896	1446.87	1478	1028.87	418
10	459.69	2475	2015.31	1684	1224.31	791
11	463.29	2314	1850.71	1896	1432.71	418
12	556.47	1313.99	757.52	895	338.53	418.99
13	560.45	5896	5335.55	896	335.55	5000
14	619.39	1478.37	858.98	1071	451.61	407.37
15	664.16	1913	1248.84	1079	414.84	834
16	712.71	2747	2034.29	1913	1200.29	834
17	744.83	2768	2023.17	2150	1405.17	618
18	759.81	2150	1390.19	1732	972.19	418
19	764.6	1732	967.4	1313	548.4	419
					517.0595	15037.26

	Minimo en completarse					
Requerimient	Inicio	Finalizacion	MCT	Atendido	Espera MCT	En ejecucion MCT
0	0	3754	3754	3	3	3751
1	0.24	424.06	423.82	6	5.76	418.06
2	13.92	434.99	421.07	16.92	3	418.07
3	111.75	532.82	421.07	114	2.25	418.82
4	228.92	842.13	613.21	424	195.08	418.13
5	235.13	853	617.87	434	198.87	419
6	312.76	950	637.24	532	219.24	418
7	354.27	1260	905.73	842	487.73	418
8	444.42	1678	1233.58	1260	815.58	418
9	449.13	2107	1657.87	1689	1239.87	418
10	459.69	2205	1745.31	1787	1327.31	418
11	463.29	1368	904.71	950	486.71	418
12	556.47	2514	1957.53	2096	1539.53	418
13	560.45	2525	1964.55	2107	1546.55	418
14	619.39	2623	2003.61	2205	1585.61	418
15	664.16	1271	606.84	853	188.84	418
16	712.71	2932	2219.29	2514	1801.29	418
17	744.83	1787	1042.17	1368	623.17	419
18	759.81	1689	929.19	1271	511.19	418
19	764.6	2096	1331.4	1678	913.4	418
					684.699	11696.08

Ejecución Heurística y Simulación para tipo de tarea similar y I/O media, arribo exponencial.

	Heurística Tiempos de Simulación					
Requerimient	Inicio	Finalizacion	Heuristica	Atendido	Espera Heuris	Ejecucion Heuristica
0	0	109	109	3	3	106
1	0.24	166	165.76	6	5.76	160
2	13.92	296	282.08	16.92	3	279.08
3	111.75	532	420.25	114	2.25	418
4	228.92	649	420.08	231	2.08	418
5	235.13	794	558.87	238	2.87	556
6	312.76	733	420.24	315	2.24	418
7	354.27	950	595.73	532	177.73	418
8	444.42	1068	623.58	648	203.58	420
9	449.13	1151	701.87	734	284.87	417
10	459.69	1212	752.31	794	334.31	418
11	463.29	1914	1450.71	1486	1022.71	428
12	556.47	1988	1431.53	1569	1012.53	419
13	560.45	2048	1487.55	1630	1069.55	418
14	619.39	2205	1585.61	1787	1167.61	418
15	664.16	1486	821.84	1068	403.84	418
16	712.71	1368	655.29	950	237.29	418
17	744.83	1569	824.17	1151	406.17	418
18	759.81	1630	870.19	1212	452.19	418
19	764.6	1787	1022.4	1368	603.4	419
					369.849	7802.08

	Minimo en completarse					
Requerimiento	Inicio	Finalizacion	MCT	Atendido	Espera MCT	En ejecucion
0	0	421	421	3	3	418
1	0.24	424	423.76	6	5.76	418
2	13.92	435	421.08	16.92	3	418.08
3	111.75	532	420.25	114	2.25	418
4	228.92	839	610.08	421	192.08	418
5	235.13	842	606.87	424	188.87	418
6	312.76	853	540.24	435	122.24	418
7	354.27	950	595.73	532	177.73	418
8	444.42	1257	812.58	839	394.58	418
9	449.13	2205	1755.87	1787	1337.87	418
10	459.69	1675	1215.31	1257	797.31	418
11	463.29	1368	904.71	950	486.71	418
12	556.47	1678	1121.53	1260	703.53	418
13	560.45	1271	710.55	853	292.55	418
14	619.39	1689	1069.61	1271	651.61	418
15	664.16	1787	1122.84	1368	703.84	419
16	712.71	1260	547.29	842	129.29	418
17	744.83	2093	1348.17	1675	930.17	418
18	759.81	2096	1336.19	1678	918.19	418
19	764.6	2107	1342.4	1689	924.4	418
					448.249	8361.08

Ejecución Heurística y Simulación para tipo de tarea similar y I/O Alta, arribo exponencial.

	Heurística Tiempos de Simulación					
requerimiento	Inicio	Finalizacion	Heuristica	Atendido	Espera Heurística	Ejecución Heurística
0	0	110.3	110.3	3	3	107.3
1	0.24	230	229.76	6	5.76	224
2	13.92	320	306.08	110	96.08	210
3	111.75	671	559.25	114	2.25	557
4	228.92	649	420.08	231	2.08	418
5	235.13	1072	836.87	238	2.87	834
6	312.76	1149	836.24	315	2.24	834
7	354.27	775	420.73	357	2.73	418
8	444.42	1068	623.58	649	204.58	419
9	449.13	1505	1055.87	671	221.87	834
10	459.69	1505	1045.31	671	211.31	834
11	463.29	1906	1442.71	1072	608.71	834
12	556.47	2029	1472.53	1611	1054.53	418
13	560.45	1611	1050.55	1193	632.55	418
14	619.39	1904	1284.61	1486	866.61	418
15	664.16	1193	528.84	775	110.84	418
16	712.71	1983	1270.29	1149	436.29	834
17	744.83	2339	1594.17	1505	760.17	834
18	759.81	2339	1579.19	1505	745.19	834
19	764.6	1486	721.4	1068	303.4	418
			869.418		313.653	555.765

	Minimo en completarse					
Req	Inicio	Finalizacion	MCT	Atendido	Espera MCT	Ejecución MC
0	0	421	421	3	3	418
1	0.24	424	423.76	6	5.76	418
2	13.92	435	421.08	16.92	3	418.08
3	111.75	532	420.25	114	2.25	418
4	228.92	839	610.08	421	192.08	418
5	235.13	842	606.87	424	188.87	418
6	312.76	853	540.24	435	122.24	418
7	354.27	950	595.73	532	177.73	418
8	444.42	1257	812.58	839	394.58	418
9	449.13	2205	1755.87	1787	1337.87	418
10	459.69	1675	1215.31	1257	797.31	418
11	463.29	1368	904.71	950	486.71	418
12	556.47	1678	1121.53	1260	703.53	418
13	560.45	1271	710.55	853	292.55	418
14	619.39	1689	1069.61	1271	651.61	418
15	664.16	1787	1122.84	1368	703.84	419
16	712.71	1260	547.29	842	129.29	418
17	744.83	2093	1348.17	1675	930.17	418
18	759.81	2096	1336.19	1678	918.19	418
19	764.6	2107	1342.4	1689	924.4	418
			866.303		448.249	418

Ejecución Heurística y Simulación para tipo de tarea similar y I/O Alta, arribo Gauss.

	Heurística Tiempos de Simulación					
Requerimento	Inicio	Finalizacion	Heuristica	Atendido	Espera Heuris	Ejecución Heu
0	0	113	113	3	3	110
1	163.26	276	112.74	166	2.74	110
2	323	436	113	326	3	110
3	486	600	114	489	3	111
4	648	761	113	651	3	110
5	815	929	114	818	3	111
6	963	1076	113	966	3	110
7	1121	1234	113	1124	3	110
8	1287	1400	113	1290	3	110
9	1448	1561	113	1451	3	110
10	1607	1720	113	1610	3	110
11	1771	1884	113	1774	3	110
12	1928	2041	113	1931	3	110
13	2095	2209	114	2098	3	111
14	2249	2363	114	2252	3	111
15	2408	2521	113	2411	3	110
16	2576	2689	113	2579	3	110
17	2731	2845	114	2734	3	111
18	2895	3008	113	2898	3	110
19	3059	3173	114	3062	3	111
					2.987	2206

	Minimo en completarse					
Requerimient	Inicio	Finalizacion	MCT	Atendido	Espera MCT	Ejecución MC
0	0	421	421	3	3	418
1	163.26	584	420.74	166	2.74	418
2	323	744	421	326	3	418
3	486	907	421	489	3	418
4	648	1069	421	651	3	418
5	815	1236	421	819	4	417
6	963	1384	421	966	3	418
7	1121	1542	421	1124	3	418
8	1287	1708	421	1290	3	418
9	1448	1869	421	1451	3	418
10	1607	2028	421	1610	3	418
11	1771	2192	421	1774	3	418
12	1928	2349	421	1931	3	418
13	2095	2516	421	2099	4	417
14	2249	2671	422	2252	3	419
15	2408	2829	421	2411	3	418
16	2576	2997	421	2579	3	418
17	2731	3152	421	2734	3	418
18	2895	3316	421	2898	3	418
19	3059	3480	421	3062	3	418
					3.087	8359

Anexo B. Traza

Traza de ejecución del simulador

Arribo requerimiento número 00

```
2009-10-08 17:33:40,291 INFO ------------------------------------------------------------
2009-10-08 17:33:40,291 INFO Requerimiento App_Request00
2009-10-08 17:33:40,292 INFO ------------------------------------------------------------
```

Hace chequeo Kruscal con nodo 17

```
2009-10-08 17:33:46,198 INFO ------------------------------------------------------------
2009-10-08 17:33:46,198 INFO Solo prueba Test Cabeza P17
2009-10-08 17:33:46,198 INFO ------------------------------------------------------------
2009-10-08 17:33:46,198 DEBUG estos son los anchos de banda de cluster c2   0.20
2009-10-08 17:33:46,199 DEBUG estos son los anchos de banda de cluster c1   0.20
2009-10-08 17:33:46,199 DEBUG estos son los anchos de banda de cluster c4   0.20
2009-10-08 17:33:46,199 DEBUG Limita Master..........
2009-10-08 17:33:46,199 DEBUG La cuenta del cluster c2 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,199 DEBUG La cuenta del cluster c1 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,199 DEBUG La cuenta del cluster c4 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,199 DEBUG Anchos de banda resultantes {'c3': 0, 'c2': 0.066666666666666666, 'c1':
0.066666666666666666, 'c4': 0.066666666666666666}
2009-10-08 17:33:46,200 DEBUG Los clusters con maquinas libres ['c3', 'c2', 'c1', 'c4']
2009-10-08 17:33:46,200 INFO aquireMachines...Desde P17 al cluster c3
2009-10-08 17:33:46,200 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,200 INFO aquireMachines...Desde P17 al cluster c2
2009-10-08 17:33:46,201 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,201 DEBUG El costo total es de 200.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,201 INFO aquireMachines...Desde P17 al cluster c1
2009-10-08 17:33:46,201 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,201 DEBUG El costo total es de 400.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,201 INFO aquireMachines...Desde P17 al cluster c4
2009-10-08 17:33:46,202 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,202 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
```

Hace chequeo Kruscal con nodo 09

```
2009-10-08 17:33:46,202 INFO ------------------------------------------------------------
2009-10-08 17:33:46,202 INFO Solo prueba Test Cabeza P09
2009-10-08 17:33:46,202 INFO ------------------------------------------------------------
2009-10-08 17:33:46,203 DEBUG estos son los anchos de banda de cluster c3   0.20
2009-10-08 17:33:46,203 DEBUG estos son los anchos de banda de cluster c1   0.20
2009-10-08 17:33:46,203 DEBUG estos son los anchos de banda de cluster c4   0.20
2009-10-08 17:33:46,203 DEBUG Limita Master..........
2009-10-08 17:33:46,203 DEBUG La cuenta del cluster c3 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,203 DEBUG La cuenta del cluster c1 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,204 DEBUG La cuenta del cluster c4 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,204 DEBUG Anchos de banda resultantes {'c3': 0.066666666666666666, 'c2': 0, 'c1':
0.066666666666666666, 'c4': 0.066666666666666666}
2009-10-08 17:33:46,204 DEBUG Los clusters con maquinas libres ['c3', 'c2', 'c1', 'c4']
2009-10-08 17:33:46,204 INFO aquireMachines...Desde P09 al cluster c3
2009-10-08 17:33:46,204 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,204 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,205 INFO aquireMachines...Desde P09 al cluster c2
2009-10-08 17:33:46,212 DEBUG El costo total es de 200.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,212 INFO aquireMachines...Desde P09 al cluster c1
2009-10-08 17:33:46,212 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,212 DEBUG El costo total es de 400.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,212 INFO aquireMachines...Desde P09 al cluster c4
2009-10-08 17:33:46,213 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,213 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
```

Hace chequeo Kruscal con nodo 01

```
2009-10-08 17:33:46,213 INFO ------------------------------------------------------------
2009-10-08 17:33:46,213 INFO Solo prueba Test Cabeza P01
2009-10-08 17:33:46,213 INFO ------------------------------------------------------------
2009-10-08 17:33:46,213 DEBUG estos son los anchos de banda de cluster c3   0.20
2009-10-08 17:33:46,214 DEBUG estos son los anchos de banda de cluster c2   0.20
2009-10-08 17:33:46,214 DEBUG estos son los anchos de banda de cluster c4   0.20
2009-10-08 17:33:46,214 DEBUG Limita Master..........
2009-10-08 17:33:46,214 DEBUG La cuenta del cluster c3 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,214 DEBUG La cuenta del cluster c2 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,214 DEBUG La cuenta del cluster c4 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,214 DEBUG Anchos de banda resultantes {'c3': 0.066666666666666666, 'c2': 0.066666666666666666,
'c1': 0, 'c4': 0.066666666666666666}
2009-10-08 17:33:46,215 DEBUG Los clusters con maquinas libres ['c3', 'c2', 'c1', 'c4']
2009-10-08 17:33:46,215 INFO aquireMachines...Desde P01 al cluster c3
2009-10-08 17:33:46,215 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,215 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,215 INFO aquireMachines...Desde P01 al cluster c2
2009-10-08 17:33:46,216 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,216 DEBUG El costo total es de 200.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,216 INFO aquireMachines...Desde P01 al cluster c1
2009-10-08 17:33:46,216 DEBUG El costo total es de 400.0000 y la fuerza de trabajo del cluster es de  0.1600
```

```
2009-10-08 17:33:46,216 INFO aquireMachines...Desde P01 al cluster c4
2009-10-08 17:33:46,216 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,217 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
```

Elección y ejecución con nodo 17

```
2009-10-08 17:33:46,217 INFO ------------------------------------------------------------
2009-10-08 17:33:46,217 INFO Definitivo Test Cabeza P17
2009-10-08 17:33:46,217 INFO ------------------------------------------------------------
2009-10-08 17:33:46,218 DEBUG estos son los anchos de banda de cluster c2   0.20
2009-10-08 17:33:46,218 DEBUG estos son los anchos de banda de cluster c1   0.20
2009-10-08 17:33:46,218 DEBUG estos son los anchos de banda de cluster c4   0.20
```

Calcula los anchos de banda contra todos los nodos

```
2009-10-08 17:33:46,218 DEBUG Limita Master..........
2009-10-08 17:33:46,218 DEBUG La cuenta del cluster c2 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,218 DEBUG La cuenta del cluster c1 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,219 DEBUG La cuenta del cluster c4 ancho de banda   0.20, total   0.60,  cluster origen   0.20
2009-10-08 17:33:46,219 DEBUG Anchos de banda resultantes {'c3': 0, 'c2': 0.066666666666666666, 'c1':
0.066666666666666666, 'c4': 0.066666666666666666}
2009-10-08 17:33:46,219 DEBUG Los clusters con maquinas libres ['c3', 'c2', 'c1', 'c4']
2009-10-08 17:33:46,219 INFO aquireMachines...Desde P17 al cluster c3
2009-10-08 17:33:46,219 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,219 INFO aquireMachines...Desde P17 al cluster c2
2009-10-08 17:33:46,220 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,220 DEBUG Worker asignado P09 con fuerza  0.020
2009-10-08 17:33:46,220 DEBUG El procesador P09 consume  0.010, datos  0.500 fuerza  0.020, total  0.010
2009-10-08 17:33:46,220 DEBUG Worker asignado P10 con fuerza  0.020
2009-10-08 17:33:46,220 DEBUG El procesador P10 consume  0.010, datos  0.500 fuerza  0.020, total  0.020
2009-10-08 17:33:46,220 DEBUG Worker asignado P11 con fuerza  0.020
2009-10-08 17:33:46,221 DEBUG El procesador P11 consume  0.010, datos  0.500 fuerza  0.020, total  0.030
2009-10-08 17:33:46,221 DEBUG Worker asignado P12 con fuerza  0.020
2009-10-08 17:33:46,221 DEBUG El procesador P12 consume  0.010, datos  0.500 fuerza  0.020, total  0.040
2009-10-08 17:33:46,221 DEBUG Worker asignado P13 con fuerza  0.020
2009-10-08 17:33:46,221 DEBUG El procesador P13 consume  0.010, datos  0.500 fuerza  0.020, total  0.050
2009-10-08 17:33:46,222 DEBUG Worker asignado P14 con fuerza  0.020
2009-10-08 17:33:46,222 DEBUG El procesador P14 consume  0.010, datos  0.500 fuerza  0.020, total  0.060
2009-10-08 17:33:46,222 DEBUG Antes de sumar  0.060
2009-10-08 17:33:46,222 DEBUG El costo total es de 200.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,222 INFO aquireMachines...Desde P17 al cluster c1
2009-10-08 17:33:46,222 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,223 DEBUG Worker asignado P01 con fuerza  0.020
2009-10-08 17:33:46,223 DEBUG El procesador P01 consume  0.010, datos  0.500 fuerza  0.020, total  0.010
2009-10-08 17:33:46,223 DEBUG Worker asignado P02 con fuerza  0.020
2009-10-08 17:33:46,223 DEBUG El procesador P02 consume  0.010, datos  0.500 fuerza  0.020, total  0.020
2009-10-08 17:33:46,223 DEBUG Worker asignado P03 con fuerza  0.020
2009-10-08 17:33:46,223 DEBUG El procesador P03 consume  0.010, datos  0.500 fuerza  0.020, total  0.030
2009-10-08 17:33:46,224 DEBUG Worker asignado P04 con fuerza  0.020
2009-10-08 17:33:46,224 DEBUG El procesador P04 consume  0.010, datos  0.500 fuerza  0.020, total  0.040
2009-10-08 17:33:46,224 DEBUG Worker asignado P05 con fuerza  0.020
2009-10-08 17:33:46,224 DEBUG El procesador P05 consume  0.010, datos  0.500 fuerza  0.020, total  0.050
2009-10-08 17:33:46,224 DEBUG Worker asignado P06 con fuerza  0.020
2009-10-08 17:33:46,224 DEBUG El procesador P06 consume  0.010, datos  0.500 fuerza  0.020, total  0.060
2009-10-08 17:33:46,225 DEBUG Antes de sumar  0.060
2009-10-08 17:33:46,225 DEBUG El costo total es de 400.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,225 INFO aquireMachines...Desde P17 al cluster c4
2009-10-08 17:33:46,225 DEBUG maxWork  0.133 tarea/seg, ancho de banda  0.067, datos  0.500
2009-10-08 17:33:46,225 DEBUG Worker asignado P25 con fuerza  0.020
2009-10-08 17:33:46,225 DEBUG El procesador P25 consume  0.010, datos  0.500 fuerza  0.020, total  0.010
2009-10-08 17:33:46,226 DEBUG Worker asignado P26 con fuerza  0.020
2009-10-08 17:33:46,226 DEBUG El procesador P26 consume  0.010, datos  0.500 fuerza  0.020, total  0.020
2009-10-08 17:33:46,226 DEBUG Worker asignado P27 con fuerza  0.020
2009-10-08 17:33:46,226 DEBUG El procesador P27 consume  0.010, datos  0.500 fuerza  0.020, total  0.030
2009-10-08 17:33:46,226 DEBUG Worker asignado P28 con fuerza  0.020
2009-10-08 17:33:46,226 DEBUG El procesador P28 consume  0.010, datos  0.500 fuerza  0.020, total  0.040
2009-10-08 17:33:46,226 DEBUG Worker asignado P29 con fuerza  0.020
2009-10-08 17:33:46,227 DEBUG El procesador P29 consume  0.010, datos  0.500 fuerza  0.020, total  0.050
2009-10-08 17:33:46,227 DEBUG Worker asignado P30 con fuerza  0.020
2009-10-08 17:33:46,227 DEBUG El procesador P30 consume  0.010, datos  0.500 fuerza  0.020, total  0.060
2009-10-08 17:33:46,227 DEBUG Antes de sumar  0.060
2009-10-08 17:33:46,227 DEBUG El costo total es de 2000.0000 y la fuerza de trabajo del cluster es de  0.1600
2009-10-08 17:33:46,228 DEBUG Al cluster c2, antes  0.200 y se le saca  0.060 ancho de banda
2009-10-08 17:33:46,228 DEBUG Al cluster c1, antes  0.200 y se le saca  0.060 ancho de banda
2009-10-08 17:33:46,228 DEBUG Al cluster c4, antes  0.200 y se le saca  0.060 ancho de banda
2009-10-08 17:33:46,228 INFO ------------------------------------------------------------
2009-10-08 17:33:46,228 INFO Solucion Cabeza P17, costo 4600.0000, fuerza  0.6400
2009-10-08 17:33:46,228 INFO ------------------------------------------------------------
```

Asignación de máquinas por cluster

```
2009-10-08 17:33:46,229 INFO El cluster es el c3, con bw externo  0.020
2009-10-08 17:33:46,229 INFO Procesador P17 status App_Request00
2009-10-08 17:33:46,229 INFO Procesador P18 status App_Request00
2009-10-08 17:33:46,229 INFO Procesador P19 status App_Request00
2009-10-08 17:33:46,229 INFO Procesador P20 status App_Request00
2009-10-08 17:33:46,230 INFO Procesador P21 status App_Request00
2009-10-08 17:33:46,230 INFO Procesador P22 status App_Request00
2009-10-08 17:33:46,230 INFO Procesador P23 status App_Request00
2009-10-08 17:33:46,230 INFO Procesador P24 status App_Request00
2009-10-08 17:33:46,230 INFO Porcentage de uso  1.000
```

```
2009-10-08 17:33:46,230 INFO El cluster es el c2, con bw externo  0.140
2009-10-08 17:33:46,230 INFO Procesador P09 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P10 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P11 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P12 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P13 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P14 status App_Request00
2009-10-08 17:33:46,231 INFO Procesador P15 status
2009-10-08 17:33:46,232 INFO Procesador P16 status
2009-10-08 17:33:46,232 INFO Porcentage de uso  0.750
2009-10-08 17:33:46,232 INFO El cluster es el c1, con bw externo  0.140
2009-10-08 17:33:46,232 INFO Procesador P01 status App_Request00
2009-10-08 17:33:46,232 INFO Procesador P02 status App_Request00
2009-10-08 17:33:46,232 INFO Procesador P03 status App_Request00
2009-10-08 17:33:46,232 INFO Procesador P04 status App_Request00
2009-10-08 17:33:46,233 INFO Procesador P05 status App_Request00
2009-10-08 17:33:46,233 INFO Procesador P06 status App_Request00
2009-10-08 17:33:46,233 INFO Procesador P07 status
2009-10-08 17:33:46,233 INFO Procesador P08 status
2009-10-08 17:33:46,233 INFO Porcentage de uso  0.750
2009-10-08 17:33:46,233 INFO El cluster es el c4, con bw externo  0.140
2009-10-08 17:33:46,233 INFO Procesador P25 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P26 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P27 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P28 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P29 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P30 status App_Request00
2009-10-08 17:33:46,234 INFO Procesador P31 status
2009-10-08 17:33:46,234 INFO Procesador P32 status
2009-10-08 17:33:46,235 INFO Porcentage de uso  0.750
```

Una vez que finaliza la tarea restituye el ancho de banda

```
2009-10-08 17:33:46,910 INFO Antes local c2 bw externo  0.140 master c3 bw externo  0.020
2009-10-08 17:33:46,911 INFO Retorna maquina P14 con bw externo  0.010
2009-10-08 17:33:46,911 INFO Antes local c2 bw externo  0.150 master c3 bw externo  0.030
2009-10-08 17:33:46,911 INFO Retorna maquina P13 con bw externo  0.010
2009-10-08 17:33:46,912 INFO Antes local c2 bw externo  0.160 master c3 bw externo  0.040
2009-10-08 17:33:46,912 INFO Retorna maquina P12 con bw externo  0.010
2009-10-08 17:33:46,913 INFO Antes local c1 bw externo  0.140 master c3 bw externo  0.050
2009-10-08 17:33:46,913 INFO Retorna maquina P06 con bw externo  0.010
2009-10-08 17:33:46,914 INFO Antes local c1 bw externo  0.150 master c3 bw externo  0.060
2009-10-08 17:33:46,914 INFO Retorna maquina P05 con bw externo  0.010
2009-10-08 17:33:46,915 INFO Antes local c1 bw externo  0.160 master c3 bw externo  0.070
2009-10-08 17:33:46,915 INFO Retorna maquina P04 con bw externo  0.010
2009-10-08 17:33:46,916 INFO Antes local c4 bw externo  0.140 master c3 bw externo  0.080
2009-10-08 17:33:46,916 INFO Retorna maquina P30 con bw externo  0.010
2009-10-08 17:33:46,920 INFO Antes local c4 bw externo  0.150 master c3 bw externo  0.090
2009-10-08 17:33:46,920 INFO Retorna maquina P29 con bw externo  0.010
2009-10-08 17:33:46,920 INFO Antes local c4 bw externo  0.160 master c3 bw externo  0.100
2009-10-08 17:33:46,921 INFO Retorna maquina P28 con bw externo  0.010
```

Anexo C. Código de Heurística en el simulador

```
from opus7.graphAsMatrix import GraphAsMatrix
from opus7.graphAsLists import GraphAsLists
from opus7.digraphAsMatrix import DigraphAsMatrix
from opus7.digraphAsLists import DigraphAsLists
from opus7.binaryHeap import BinaryHeap
from opus7.association import Association
from opus7.partitionAsForest import PartitionAsForest


from Pygraph_4 import *
from Data_4 import *
import logging

class Coordinador:

    def costEval(self,nodoMaster,definitive,nameReq,taskName,numTask,clusters,tasks,App,solForce,procs):
        clustersKeys = clusters.cluster.keys()
        #procs = clusters.procs()

        logging.info("----------------------------------------------------------------")
        if definitive:
            logging.info( "Definitivo Test Cabeza %s"%(procs[nodoMaster].name))
        else:
            logging.info( "Solo prueba Test Cabeza %s"%(procs[nodoMaster].name))
        logging.info("----------------------------------------------------------------")

        clustersMaxBW = clusters.masterBW(nodoMaster)
        clusterFreeList=clusters.free_cluster()
        logging.debug("Los clusters con maquinas libres %s"%(clusterFreeList))
        resultTupleTot=[0,0]

        for ci in clusterFreeList:
            reqList={}
            reqListTime={}

            #mando a ver con la cabeza y los datos las maquinas que tomo
            #vienen marcadas por el requerimiento si es definitivo sino solo saco las cuentas

resultTuple=clusters.cluster[ci].aquireMachines(definitive,nodoMaster,nameReq,taskName,numTask,clusters,procs,task
s,clustersMaxBW,solForce)
            resultTupleTot[0]+=resultTuple[0]
            resultTupleTot[1]+=resultTuple[1]

        if definitive:
            #Actualizo loa anchos de banda consumidos
            clusters.updateMaxBW(nodoMaster,procs,clustersMaxBW)

        return [resultTupleTot,reqListTime]


    @staticmethod
    def KruskalsAlgorithm(g):
        """
        (Graph) -> GraphAsLists
        Kruskal's algorithm to find a minimum-cost spanning tree
        for the given edge-weighted, undirected graph.
        """
        n = g.numberOfVertices
        result = GraphAsLists(n)
        for v in xrange(n):
            vG = g.getVertex(v)
            result.addVertex(v, vG.weight)
        queue = BinaryHeap(g.numberOfEdges)
        for e in g.edges:
            weight = e.weight * -1
            queue.enqueue(Association(weight, e))
        partition = PartitionAsForest(n)
        enough = False
        while (not queue.isEmpty and partition.count > 1) and not enough:
            assoc = queue.dequeueMin()
            e = assoc.value
            n0 = e.v0.number
            n1 = e.v1.number
            s = partition.find(n0)
            t = partition.find(n1)
            if s != t:
```

```
                partition.join(s, t)
                result.addEdge(n0, n1, e.weight)
            # Con esto me aseguro por lo menos 3 reintentos para hill-climbing
            if (partition.count == 3):
                enough = True
        resultVertices = {}
        for v in result:
            if not(resultVertices.has_key(partition.find(v.number))):
                resultVertices[partition.find(v.number)]=[]
            resultVertices[partition.find(v.number)].append(v.number)
        return resultVertices,result


    def Heuristica(self,taskName,numTask,nameReq,clusters,tasks,App):
        #Serializo....
        App.lock.acquire()
        # veo si hay maquinas
        if len(clusters.free_machines())<3:
            logging.info( "No hay suficientes maquinas para el Requerimiento %s, hay
%02d"%(nameReq,len(clusters.free_machines())))
            App.lock.release()
            return

        graph = Heuristica_Graph()
        logging.info("----------------------------------------------------------------")
        logging.info( "Requerimiento %s"%(nameReq))
        logging.info("----------------------------------------------------------------")

        # Creo el grafo
        g = GraphAsMatrix(32)
        g = graph.setup(clusters)
        #graph.paint(nameReq + "-A-Grafo", graph.convertGraph(g),clusters,"blue")

        # Tomas las maquinas libres para ser coherente
        procs = clusters.free_machines()
        # Lo particiono
        print "Kruskal's Algorithm"
        graphPart = self.KruskalsAlgorithm(g)
        graphPartition = graphPart[0]
        #graph.paint(nameReq + "-C-Kruscal ", graph.convertGraph(graphPart[1]),clusters,"blue")

        print "Hill Climbing Algorithm with k-restarts"
        # Seleccion de nodo inicial
        partKeys = graphPartition.keys()
        solHead=0
        solCost = 10000
        solForce = 0
        # Recorro todas las soluciones  y selecciono la de menor tiempo con la lista de maquinas solucion
        for part in partKeys:
            # Calculo el tiempo de cada cluster, el ultimo parametro es el costo total para el calculo definitivo
            cost_list =
self.costEval(graphPartition[part][0],False,nameReq,taskName,numTask,clusters,tasks,App,0,procs)
            if solForce < cost_list[0][1]:
                solHead = graphPartition[part][0]
                solCost = cost_list[0][0]
                solForce = cost_list[0][1]
                migraList = cost_list[1]

        # Una vez que se elige la solucion se hace en forma definitiva
        cost_list = self.costEval(solHead,True,nameReq,taskName,numTask,clusters,tasks,App,solForce,procs)
        print "Finaliza heuristica"
        App.lock.release()
        logging.info( "------------------------------------------------------------")
        logging.info( "Solucion Cabeza %s, costo %7.4f, fuerza %7.4f"%(procs[solHead].name,solCost,solForce))
        logging.info( "------------------------------------------------------------")
        clusters.cluster_use()
```