Planificación dinámica de clusters a demanda en entornos Grid

Bertogna, Leandro
Departamento de Ciencias de la Computación
Universidad Nacional del Comahue
mlbertog@uncoma.edu.ar

Naiuf Marcelo, De Giusti Armando
Instituto de Investigación en Informática LIDI
Universidad Nacional de La Plata
{degiusti,mnaiouf }@lidi.info.unlp.edu.ar

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 metaorganización, 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.

Palabras claves: Grid, planificación, máquinas virtuales

1. Introducción

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. Tareas de gestión rutinarias que requieren recursos humanos especializados deberían disminuir en forma considerable para transformarse solo en definición de políticas, que sistemas autónomos de gestión de infraestructura cumplieran o hicieran cumplir.

Un punto clave para la gestión de recursos en los sistemas Grid se encuentra en la capa intermedia. Esta coordina recursos e intercambia información sin importar su ubicación física. Cuando alguna aplicación utiliza 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.

La componente que interviene de forma activa de 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 alternativa para maximizar la utilización de recursos, es la implementación de máquinas virtuales. Las máquinas virtuales ofrecen la posibilidad de instanciar entornos de trabajo preconfigurados e independientes, tiene 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.

La contribución de este trabajo se encuentra 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 en 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.

En la segunda sección se analizarán trabajos relacionados de otros grupos de investigación, en la tercera se describirá la arquitectura donde está incluido el metaplanificador, en las tercer y cuarta sección se describe la extensión del planificador y los algoritmos que lo implementan y por último, se analiza el administrador de recursos local y se realizan las conclusiones.

2. Trabajos Relacionados

Existen diferentes trabajos relacionados con esta problemática tomando como caso de uso entornos de altas prestaciones, cada uno enfatizando diferentes aspectos de la solución. El proyecto Virtual Workspaces[1] 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[2] 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[3] 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[4] donde el nivel de abstracción es mucho mayor, permite a las aplicaciones hacer uso de entornos virtuales a través de servicios Grid.

3. Arquitectura

La arquitectura donde se inscribe el presente trabajo tiene por objetivo la creación de redes de recursos virtuales dentro de entornos Grid[5]. 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.

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, y finalmente la capa de recursos que se relaciona con la instanciación de los recursos, en este caso máquinas virtuales sobre hosts físicos.

Los clientes accesan al servicio de planificación del sistema expresando sus requerimientos a través de un lenguaje basado en XML, que en forma paramétrica determinan cuantos nodos virtuales, que imágenes y cuales 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, a través del modelo que se presenta en este trabajo, basándose en información de configuración y estado de esos hosts 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, caso este específico para el presente trabajo, la asignación llevada a cabo por el planificador instanciará no más de una máquina virtual 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.

4 Metaplanificadores

La composición de módulos del metaplanificador se basa en el framework propuesto por la RFC 2753[6], este framework define protocolos y una jerarquía de módulos para el control, basado en políticas. Sus componentes arquitecturales principales son el punto donde las políticas son aplicadas (Policy Enforcement Point - PEP) y donde se toma la decisión de cuales aplicar (Policy Decision Point - PDP). Para el metaplanificador propuesto en este trabajo se ha invertido el flujo de control. En el caso de la RFC los eventos ocurren en los PEP y estos generan requerimientos al PDP para la toma de decisiones, en el caso propuesto en este trabajo, contrariamente la secuencia de eventos comienza con el ingreso de un requerimiento de espacio virtual en el PDP, este consulta el estado a los niveles inferiores y dependiendo de la información obtenida, genera una configuración para satisfacer el requerimiento.

Para permitir que estos puntos de decisión trabajen correctamente y la solución sea escalable se utiliza el principio de alcance en la toma de decisión, minimizando que requerimientos de las organizaciones locales y sus políticas suban a niveles superiores de la jerarquía en la Grid, para lo que se utiliza un punto de decisión local (Local policy desicion point LPDP), extensión prevista en la RFC. La RFC interpreta este LPDP como un representante local del PDP, en esta implementación se sigue la misma filosofía pero orientando su inteligencia a la solución de problemas locales y como representante de la organización local para negociar acuerdos de servicio con el PDP del Grid y hacerlos cumplir.

Para la implementación en primer término se realizó el estudio de distintos metaplanificadores disponibles en el mercado eligiéndose a CSF (Community Scheduler Framework -CSF)[7]. 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.

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. Un esquema de esta propuesta puede observarse en la figura 1.

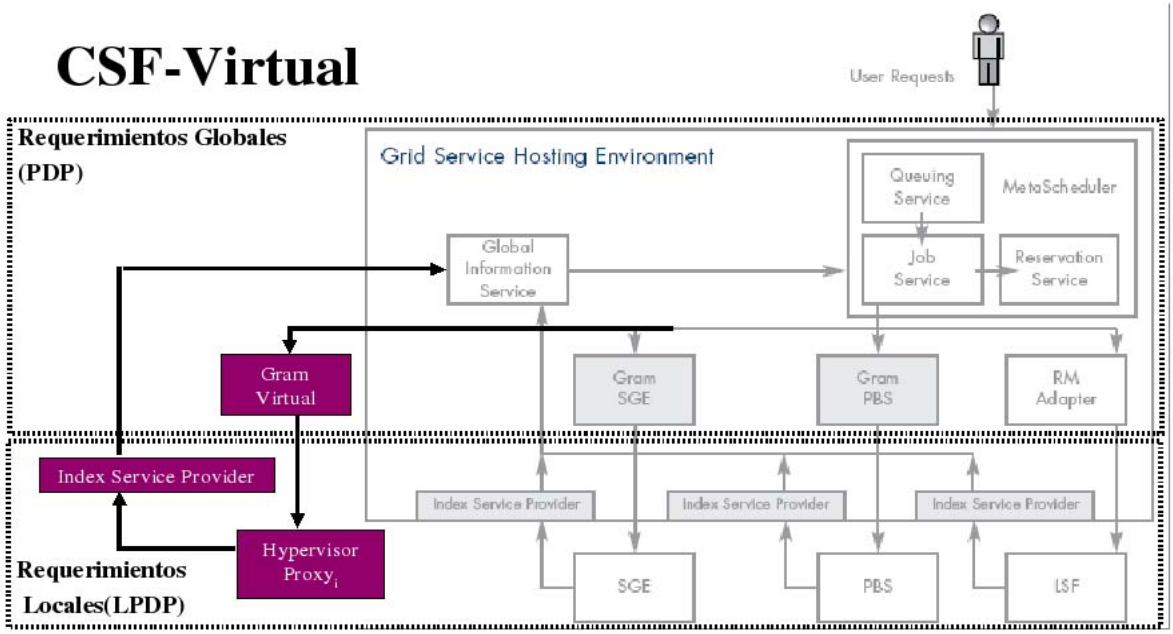


Figura 1. Arquitectura extendida de CSF

Los módulos grisados son parte del framework. Los módulos extendidos por este trabajo son las cajas rectangulares “GRAM Virtual”, este módulo 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 framework original, puede haber varios módulos “Hypervisor Proxy” coordinados trabajando en una ejecución bajo “GRAM Virtual”. Los rectangulos 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 a uno por vez. En el caso del adaptador de máquinas virtuales los recursos son dinámicos y con características heterogéneas, 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 framework 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, 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 1 , SGE o PBS generalmente son solo sistemas de colas, estas características se adaptan perfectamente en el esquema de puntos de toma de decisión local.

4.1 Algoritmos de selección de máquinas

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 podría ser costo o rendimiento y con todas las combinaciones de equipamiento posible no es un 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.

Se concluyó que una alternativa aceptable es mapear la red de comunicación y las máquinas disponibles de la Grid como un grafo no dirigido, para simplificar su tratamiento. Equipos disponibles serán representados por nodos, y 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. En la figura 2 se puede ver un ejemplo de un grupo de clusters. El metaplanificador realiza una consulta según los requerimientos de la aplicación y genera un grafo como el de la figura 3.

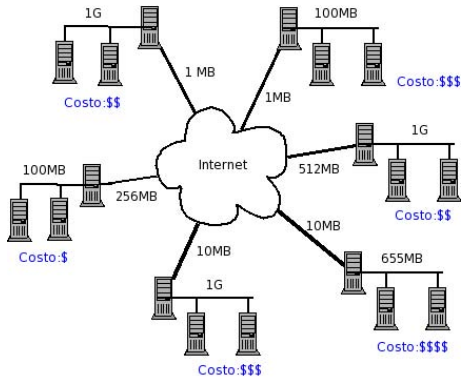


Figura 2. Ejemplo Multicluster en Grid

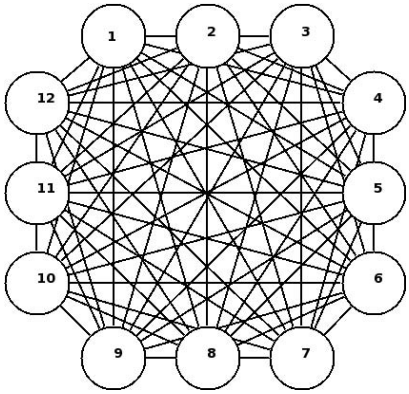


Figura 3. Mapeo de Multicluster a Grafo

Sobre el grafo se realizan búsquedas basadas en heurística como por ejemplo el algoritmo *hill-climbing*. Este algoritmo 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. Los problemas que tiene el algoritmo son encontrar la presencia de mínimos locales y mesetas, una vez que llegue a un mínimo local a pesar de no ser una buena solución se detendrá la búsqueda, de la misma manera si encontrara mesetas, no sabría por donde continuar y podría seguir un camino equivocado. Existen maneras 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 no moverse siempre en el sentido correcto sino perturbando el vector de sentido de acuerdo a una función aleatoria, como es el algoritmo *simulated annealing*.

Sin embargo para este problema se desestimaron los movimientos aleatorios debido a las características singulares del problema. Puede observarse que geográficamente, agrupando países o zonas se producen conjuntos con mejor calidad o ancho de banda de comunicación que otras, 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 y alguno de estos puede coincidir con el mínimo global. Debido a este análisis se modificó 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 nos asegura el mínimo global pero da una buena aproximación a la misma.

Se observó que el particionamiento del grafo para el problema de los recomienzos del *hill-climbing* mejora la solución, lo cual 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 árbol 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 árbol de expansión de dicho grafo.

Existen varios algoritmos para la encontrar el MST de un grafo, el que más se adapta a la solución del problema es el algoritmo de *Kruskal*. Este algoritmo ordena las aristas del grafo de menor a mayor peso y va agregando de a una y chequeando que no se formen ciclos en el grafo hasta encontrar la solución. La selección de este algoritmo fue por su metodología de construcción del grafo, el algoritmo 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, por lo que si 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*.

En las figuras 4 y 5 se puede observar el particionamiento según el algoritmo de *Kruskal* sobre el grafo de la figura 3. El requerimiento de la aplicación fue de cuatro procesadores por lo que se encontraron tres particiones que cumplían el requerimiento. La figura 4 muestra una solución de *hill-climbing* donde la función a optimizar fue el conjunto de máquinas más rápidas sin importar el costo. El primer paso fue buscar la máquina mas rápida por parición en cada partición; este paso está indicado en el grafo con flechas grisadas, cada uno de estos nodos sirve de inicio para el algoritmo de *hill-climbing*, de la misma manera en la figura 5 pero esta vez la función de a optimización fue el costo. En cada caso el algoritmo de *hill-climbing* realizo 3 reintentos y el máximo o mínimo local encontrado coincide con el global.

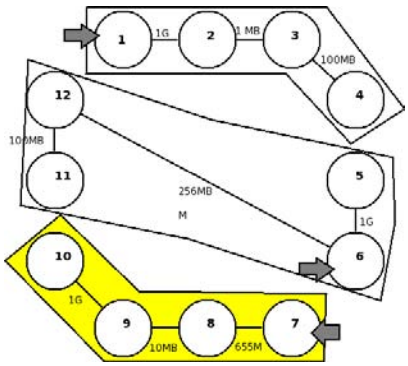


Figura 4. Solución por Heurística de Alta Prestación

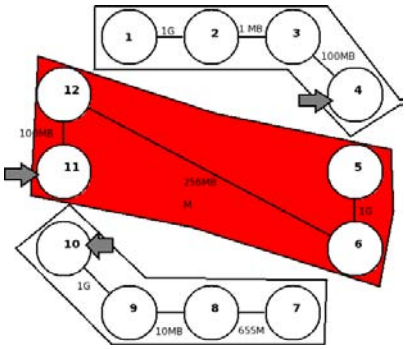


Figura 5. Solución por Heurística de Bajo Costo

Se realizó un análisis de complejidad de los algoritmos debido a la gran cantidad de aristas que se generan para un número pequeño de máquinas. Para el algoritmos MTS la complejidad es de $O(n \log(n))$ y para el algoritmo de *hill-climbing* usando lista de adyacencias es de $O(n \log(n))$. Una mejora sería detectar particiones a nivel de nodos Grid y no con todas las máquinas que integran los clusters del Grid.

5. Planificador Local

El módulo de planificación local o administrador de recursos es un servicio disponible en cada nodo Grid, 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, este protocolo se encuentra especificado en el proyecto OASIS WSDM (Web Services Distributed Management)[8] 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 permite 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[9], KVM[10] o Qemu[11]. 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.

Otra de las tareas, aún no implementada, del módulo de administración de recursos locales es la negociación de acuerdos de servicio con el planificador global. 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 ofrecen 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.

5.1 Resultados Experimentales

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[12], 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. Se experimentó en la modificación de parámetros como memoria y cantidad de procesadores asignados a las máquinas virtuales, también con la migración de las mismas, para analizar los

efectos de estos cambios dinámicos sobre la aplicación. 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.

En la modificación de parámetros se utilizaron dos variables, la memoria y cantidad de procesadores. La razón de esta selección se debió a que son dos de los parámetros que generalmente se conocen a priori y son condición necesaria para la ejecución de la aplicación. En esta experiencia se sometió a la ejecución de un worker a un stress de memoria, la ejecución de la aplicación comienza con un requerimiento de memoria de 256MB y sin que la ejecución se detenga se lo disminuye. En la figura 6 se puede observar la reacción de la máquina virtual a esa modificación durante dos corridas de la aplicación. La primer 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 requerimineto inferior para la ejecución.

Para las pruebas de asignación de procesadores se verificó el porcentaje de utilización fue variando proporcionalmente entre los procesos y procesadores asignados. 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.

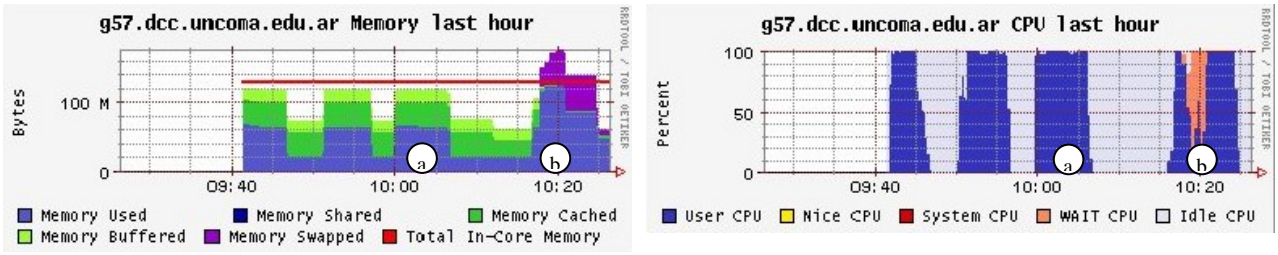


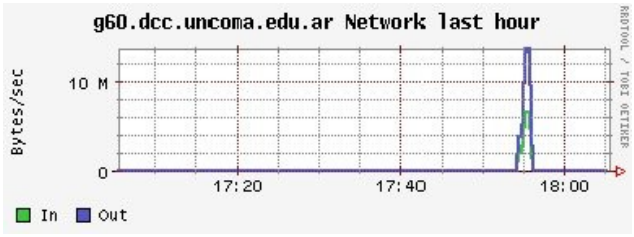
Figura 6. Modificación parámetro de memoria

Un caso donde la migración de máquinas virtuales es útil podría ser el siguiente: la configuración de un cluster es de dos equipos con doble procesador, uno dedicado a satisfacer requerimientos de la organización local y otro dedicado 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. Solo dejar dos procesadores disponibles para satisfacer los requerimientos de Grid y dejar otros ociosos sería costoso. Por lo que por política de la organización local, el administrador de recursos ofrece cuatro procesadores a la organización virtual, pero si llega algún

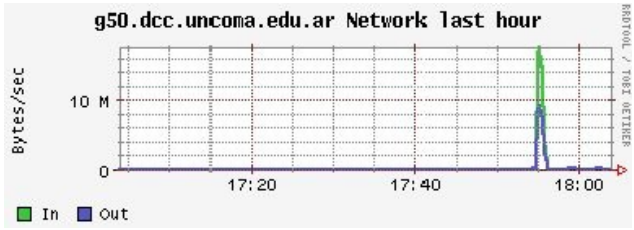
requerimiento local este tendrá prioridad sobre los procesos de la organización virtual. Si relacionamos estos conceptos con los de la figura 2, en donde cada máquina tiene un costo económico asociado, el costo será menor en el cluster en donde no necesariamente estarán todas las máquinas disponibles todo el tiempo por las políticas de la organización local.

Para el caso de uso mencionado un ejemplo sería que, 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 esta siendo ocupada por el requerimiento local, y se han asignado dos procesos a cada procesador y cada uno de ellos usa la CPU un 50%. Cuando la aplicación local termina, el administrador de recursos migra automaticamente 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 una CPU al 100%.

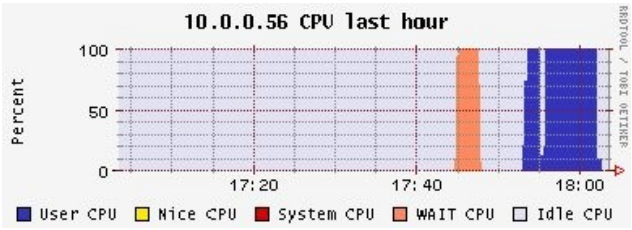
En el caso de que las políticas hubieran sido fijas una vez terminada la aplicación local, el equipamiento asignado a esas tareas hubiera quedado ocioso. De esta manera se obtiene una mejora en el aprovechamiento del equipamiento solo con adaptar las políticas.



(a) Máquina Física Origen



(b) Máquina Física Destino



(c) Máquina Virtual Migrada en Ejecución

Figura 7. Proceso de migración máquina virtual

En la figura 7 se ve un ejemplo de proceso de migración de máquinas virtuales [15]. En la subfigura (a) se encuentra el tráfico de red de entrada y salida de la máquina física origen, en la subfigura (b) el trafico de red de entrada y salida en la máquina física receptora. En la tercera subfigura 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.

7 Trabajo futuro

Durante el desarrollo de este trabajo se han encontrado distintas características que pueden ser explotadas para un mejor aprovechamiento del modelo propuesto. En cuanto a la selección de equipamiento, se está trabajando en incorporar múltiples criterios de selección, algunas experiencias se están orientando en poder determinar cuantas máquinas de un cluster son necesarias para lograr un rendimiento óptimo del conjunto. Hasta el momento el algoritmo busca optimizar el grafo dependiendo de una función heurística de una sola variable, pero agregando distintas variables la solución tiende a mejorar sensiblemente [13][14].

Otro tema que se encuentra en estudio es la migración interdominio, el ejemplo propuesto en este trabajo migra máquinas virtuales solo en un cluster y esto reduce de manera considerable el uso de la red. Las imágenes de las maquinas virtuales se encuentran compartidas a través de un repositorio común con NSF, una de las alternativas que se barajan es usar la característica *copy-on-write* para minimizar la transferencia.

El último punto en el que se está trabajando es la parada de la aplicación distribuida en el entorno Grid. Esto tiene relación con conceptos de sistemas distribuidos de estado global del sistema, se está analizando colocar monitores en las colas de mensajes de los hypervisores y detectar cuando hay un estado estable del sistema para un dominio restringido de aplicaciones.

8 Conclusión

En este trabajo se han presentado la extensión de un metaplanificador para la generación de laboratorios remotos de máquinas virtuales y administradores de recursos locales y su posterior administración; dentro de un esquema de administración basado en políticas.

Se han implementado algoritmos para la búsqueda de un conjunto de máquinas que satisfaga el requerimiento del uso de un espacio reducido de búsqueda, tratando de minimizar el tiempo de procesamiento y la calidad de la solución.

En el administrador local de recursos se han verificado 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.

Estos resultados permitieron concluir que ésta tecnología es factible de aplicase en espacios Grid donde existe la necesidad de adaptación y control del entorno para requerimientos de diversa índole. En éste sentido redunda en beneficios importantes para la calidad de administración y una mejor utilización de recursos.

9 Referencias

[1] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. “Virtual Clusters for Grid Communities”. CCGrid 2006.

[2] 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.

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

[4] 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.

[5] Eduardo Grosclaude, Francisco López Luro, Mario Leandro Bertogna,”Grid Virtual Laboratory Architecture”. VHPC Euro-Par'07. 2007.

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

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

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

[9] 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.

[10] KVM: “Kernel Based Virtual Machine”. http://kvm.qumranet.com/kvmwiki

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

[12] I. Foster, C. Kesselman, S. Tuecke.”The Anatomy of the Grid: Enabling Scalable Virtual Organizations”. International J. Supercomputer Applications, 15(3), 2001.

[13] Viktor Yarmolenko, Rizos Sakellariou. “An Eevaluation of Heuristics for SLABased Parallell Job Scheduling”**.**Proceedings to the Third High-Performance Grid Computing Workshop(HPGC), IEEE International Parallel & Distributed Processing Symposium (IPDPS'06) in Rhodes Island, Greece. 2006

[14] Eduardo Argollo, Adriana Gaudiani, Dolores Rexachs, Emilio Luque.”Tuning Application in a Multi-cluster Environment”. Euro-Par 2006 . 78-88, 2006.

[15] 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.