Mosix2: La versión grid de Mosix para Linux-2.6

Juan P. Caballero Lionel Gutierrez Javier Echaiz Jorge R. Ardenghi

Laboratorio de Investigación de Sistemas Distribuidos (LISiDi)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Bahía Blanca (8000), Argentina
{cjp,gl,je,jra}@cs.uns.edu.ar

Resumen

De una manera informal e idealizada podemos pensar en un sistema grid como la posibilidad de crear la ilusión de una única aunque muy potente computadora virtual en base a una colección de clusters que comparten varias combinaciones de recursos, donde el usuario no tenga la necesidad de ser consciente de la ubicación física de los mismos.

Hoy en día la necesidad de administrar este ambiente computacional globalizado permite que surjan nuevas tecnologías y líneas de investigación. Mosix surge como una alternativa viable, incorporando herramientas que permiten llevar adelante esta tarea logrando un incremento en la performance.

1. Introducción

A medida que las computadoras incrementan su poder de procesamiento, la complejidad del software crece a un nivel superior, consumiendo todos los ciclos de CPU y requiriendo aun más. No sólo se incrementa el numero de ciclos de CPU requeridos, sino que también el tiempo de compilación y vinculación (*linking*) de los programas se ha incrementado.

La idea básica detrás del clustering es que un grupo de máquinas autónomas y heterogéneas se vean y funcionen como una única computadora más potente. Con la potencia y bajo costo de las computadoras personales de hoy en día, la relación precio/performance de las workstations actuales frente a los mainframes y las mejoras tecnológicas en el campo de las redes de computadoras (ATM, Gigabit Ethernet, Myrinet, etc) que brindan conexiones de alta velocidad, tiene sentido combinar este poder de cmputo individual para construir un entorno de computación paralela y de alta performance (*High Performance Computing* o HPC).

El termino cluster es utilizado, por lo tanto, para denotar un grupo independiente de computadoras (sistema débilmente acoplado) las cuales se combinan en un sistema unificado a través de software y conexiones de red. Los clusters se utilizan típicamente para computación de alta disponibilidad (*High Availability Computing* o HAV) o para computación de alta performance (*High Performance Computing*) para proveer de una potencia de cómputo mayor que la que puede obtenerse mediante una única computadora.

Los clusters ofrecen varios beneficios a los programadores de aplicaciones y a los usuarios. Un problema computacional muchas veces puede descomponerse en piezas más pequeñas y

ejecutarse en varios nodos en un cluster. A su vez, permite aprovechar al máximo los ciclos ociosos de CPU de las máquinas de escritorio de los usuarios.

Dentro de los objetivos más importantes de un cluster se encuentran la de proveer una imagen única de sistema[6] (*single system image*), permitir una comunicación rápida entre los nodos, ser altamente escalable, brindar balance de recursos (CPU, memoria, redes, almacenamiento), seguridad y privacidad y operación y mantenimiento fácil y simple.

El objetivo fundamental de un cluster de computadoras es la de brindar una mayor potencia de cómputo que la lograda por una sola computadora, incrementando el poder de cómputo y logrando menores tiempos de respuestas, al mismo tiempo que permiten la resolución de problemas más complejos y con mayor precisión como simulaciones y resolución de modelos matemáticos, físicos, químicos y económicos. Otro de los objetivos fundamentales es el de permitir el compartimiento de recursos. Cada computadora cuenta con un conjunto de recursos heterogéneos que son utilizados para la resolución de un problema particular. Al utilizar un cluster de computadoras, diferentes recursos heterogéneos pueden combinarse para la resolución de diferentes clases de problemas, y pueden ser combinados para brindar nuevas y mejores soluciones. Este compartimiento de recursos no se refiere únicamente al intercambio de información, sino al acceso directo a sistemas de computación, software, datos y otros recursos requeridos para la resolución de los nuevos problemas que han surgido en la industria, la ciencia y la ingeniería. Incluso las organizaciones pueden compartir sus recursos con otras organizaciones formando un grid (definido aquí como cluster de clusters), compartiendo la potencia computacional de sus propios clusters entre si.

Por lo tanto es de vital importancia un manejo adecuado de los recursos, a fin de lograr una alta disponibilidad y performance. Al mismo tiempo, es fundamental el descubrimiento dinámico de recursos, permitiendo que nuevas computadoras se integren al sistema en tiempo real al mismo tiempo que se descubren computadoras que han fallado o están caídas. Un cluster debe administrar e identificar dinámicamente los recursos disponibles de manera que el usuario se despreocupe de la disponibilidad, los métodos de acceso y cualquier otra política de seguridad y uso, y de manera tal de seleccionar y asignar los recursos más apropiados para cada *job* o proceso.

Uno de los principales conflictos que surge en este contexto de recursos compartidos es el de la seguridad. Los recursos son heterogéneos y propensos a errores. Este punto debe ser atacado, a fin de brindar una visión transparente de los recursos con los que se cuenta y permitiendo de esta manera acceder a ellos desde un único punto de entrada, ofreciendo así una "virtualización" del cluster. Los clusters de una organización pueden ser compartidos con otras organizaciones externas, es evidente entonces que no se puede tener un control sobre todos los posibles usuarios foráneos Las máquinas participantes se configuran para, no sólo compartir datos, sino también ejecutar programas provenientes de lugares remotos. Esto es lo que hace un cluster potencialmente fértil para virus y programas de Caballo de Troya provenientes de otros clusters. Además, es importante comprender los problemas involucrados al autenticar usuarios y asignar las responsabilidades de manera adecuada. Contar con mecanismos eficientes y flexibles en la administración de la seguridad se han hecho una exigencia esencial para los clusters.

A fin de lograr los objetivos de imagen única de sistema y de un manejo coherente y seguro de los recursos, un cluster debe balancear la carga del sistema. Esto se logra a través de la migración de procesos que permite mover procesos de nodos sobrecargados a nodos descargados. En lugar de sobrecargar un numero pequeño de computadoras, el cluster divide la carga de manera equitativa permitiendo una mejor utilización de los recursos, mejorando la performance, la disponibilidad y la seguridad del sistema. A su vez, permite reducir la comunicación en la red,

reduciendo el *overhead* en la misma.

2. Mosix2

Mosix es un conjunto de herramientas administrativas que permite que un cluster o un grid se vea como una única computadora con múltiples procesadores, al igual que en un sistema SMP.

Mosix es implementado como una capa de virtualización del sistema operativo que provee a los usuarios y sus aplicaciones un SSI con un entorno de tiempo de ejecución de Linux, permitiendo ejecutar aplicaciones sin la necesidad de modificarlas o enlazarlas con alguna librería especial. De esta forma, en un sistema Mosix los usuarios pueden crear sus aplicaciones basadas en la creación de múltiples procesos, los cuales a través de la búsqueda de recursos, en forma transparente, por parte de Mosix pueden automáticamente migrar, a los distintos nodos para mejorar la performance global sin cambiar el ambiente de tiempo de ejecución de los procesos migrados [1].

Mosix en su versión 1 fue originalmente desarrollado para manejar un único cluster. Mosix2 fue extendido con un conjunto de nuevas características de tal forma que puede manejar tanto un cluster como un grid con varios clusters, por ejemplo, de diferentes departamentos de una universidad, como una colección de servidores o workstations. El objetivo de Mosix2 es permitir a los propietarios de los nodos compartir sus recursos cuando lo desean, mientras preservan su autonomía, permitiendo que se desconecten sus nodos del grid en cualquier momento, sin sacrificar los procesos remotos de otros nodos.

Algunas de las principales características de Mosix2[5] se detallan a continuación:

- ▷ *Provee un Single-System Image (SSI).* El descubrimiento automático y transparente de recursos es llevado a cabo por un algoritmo de diseminación de información on-line, suministrando a cada nodo la última información sobre la disponibilidad y el estado de cada recurso en el grid. Mosix soporta dos tipos de procesos: procesos Linux y procesos Mosix. Los primeros no son afectados por Mosix y corren en el modo nativo de Linux y no pueden ser migrados. Los procesos Mosix son generalmente aplicaciones de usuario que pueden beneficiarse de la migración. Estos procesos comienzan su ejecución como los ejecutables estándar de Linux, pero corren en un ambiente virtual que les permite migrar de un nodo a otro, siempre conservando su nodo origen (*home-node*). La migración de procesos puede ser tanto automática como manual. La migración automática es supervisada por algoritmos on-line, que continuamente intentan mejorar la performance: balance de carga, memoria libre disponible, migración desde nodos lentos a nodos rápidos.

- ▷ *Soporte de organizaciones virtuales.* En un grid basado en Mosix2, los administradores autorizados de cada cluster físico pueden conectarse/desconectarse del grid en cualquier momento. Luego del pedido de desconexión todos los procesos remotos, si los hay, son migrados a otros nodos. Mosix2 soporta la ejecución de procesos de larga duración (*long-running processes*) a través del mecanismo de "congelamiento" y la reactivación gradual mediante el cual los procesos pueden ser congelados en el nodo origen cuando un cluster es desconectado y luego ser reactivados gradualmente cuando nuevos recursos están disponibles. Un método de prioridades asegura que los procesos con una prioridad más alta siempre pueden desplazar a los procesos con una prioridad inferior.

- ▷ *Un ambiente de ejecución seguro para los procesos remotos.* La capa de virtualización garantiza que un proceso migrado no pueda modificar u obtener acceso a algún recurso

que no sea la CPU o la memoria en el nodo remoto. El cuidado se realiza interceptando todas las llamadas a sistema para que no puedan tener acceso a los recursos del nodo remoto. La mayoría de ellas son enviadas al nodo origen. Como resultado tenemos un ambiente de ejecución seguro (*sandbox*), que protege al nodo remoto de procesos invitados sospechosos.

- ▷ *Live Queuing.* Mosix2 incorpora una cola dinámica que permite despachar un número de tareas, para correr cuando estén disponibles recursos suficientes.

- ▷ *Soporte de batch jobs, checkpoint y recuperación.* Mosix2 soporta batch jobs que pueden ser enviados a algún nodo del cluster local. Estos pueden ser tanto Linux batch jobs nativos como ser ejecutados bajo la disciplina Mosix. La mayoría de los procesos Mosix soportan Checkpoint y recuperación. Cuando se realiza un checkpoint la imagen del proceso es guardada en un archivo. En caso de ser necesario el proceso puede recuperar esta imagen desde el archivo y continuar con su ejecución en un nodo.

- ▷ *El monitor.* Mosix2 cuenta con un monitor que provee información sobre los recursos en el grid y en cada cluster, como ser: velocidad de CPU, carga del CPU, memoria libre vs. memoria usada, espacio de swap, entre otros.

3. Experiencias preliminares

Para el desarrollo del cluster, el Laboratorio de Investigación de Sistemas Distribuidos (LISiDi) cuenta con nueve computadoras Pentium IV de 3 GHz con 512Mb de memoria RAM cada una y placas de red de 1 Gbps, conectadas mediante un switch. Además una de ellas, que oficia de server, NAT y firewall, posee dos placas adicionales para conexión al exterior (Internet e Internet-2).

En cada máquina se instaló el sistema operativo GNU/Linux, distribución Gentoo 2006.1 x86 sobre el cual se modificó el kernel 2.6.19.2 para dar soporte al ambiente Mosix, en su versión 2-17.1.0, como así también se instalaron todas las herramientas adicionales (monitor, *daemous*, comandos adicionales) que permiten el funcionamiento de cada una de las características de Mosix mencionadas en la sección anterior.

Se verificó el correcto funcionamiento del sistema ejecutando las aplicaciones de test provistas por Mosix, como así también con aplicaciones propias desarrolladas para chequear y comprender el funcionamiento de Mosix2.

4. Trabajo futuro

A partir de las experiencias y pruebas realizadas se procederá a estudiar en detalle el funcionamiento interno de Mosix2, sus métodos de migración de procesos, balance de carga, descubrimiento y gestión de recursos y manejo de la seguridad y privacidad de los nodos del sistema.

Al mismo tiempo, se realizarán pruebas en ambientes grid, estudiando las características y limitaciones existentes en este área. Se implementarán soluciones en el área de seguridad y gestión, administración y descubrimiento de recursos.

Finalmente se configurará el cluster para el uso de otras tecnologías como PVM y MPI, lo que permitirá desarrollar nuevas aplicaciones y algoritmos distribuidos. Se procederá a comparar

estas tecnologías a fin de determinar limitaciones y beneficios de cada una de ellas en relación con la misgración de procesos provista por Mosix.

REFERENCIAS

[1] A. Barak y A. Shiloh. *The MOSIX2 Super Operating System for Cluster and Grids*. Enero 2007.

[2] Dejan S. Milojicic, Fred Douglis, Yves Paindaveine, Richard Wheeler, Songnian Zhou. *Process Migration*. Agosto 1999.

[3] M. Kacer, P. Tvrdik. *Load Balancing by Remote Execution of Short Processes on Linux Cluster*. Junio 2002.

[4] Russell W. Clarke, Benajamin T B Lee. *Cluster Operating Systems*. Junio 2002.

[5] http://www.mosix.org/

[6] Javier Echaiz, Jorge Ardenghi. *Single System Image: Pilar de los Sistemas de Clustering. V Workshop de Investigadores en Ciencias de la Computación (WICC 2003), pp. 210-214. Universidad Nacional del Centro de la Provincia de Buenos Aires*. 22 y 23 de mayo de 2003.

[7] Javier Echaiz, Jorge Ardenghi. *Extending an SSI Cluster for Resource Discovery in Grid Computing. The 5th International Conference on Grid and Cooperative Computing (GCC 2006), ISBN 0-7695-2694-2, pp. 287-293, published by IEEE Computer Society,ChangSha, China*. 21 al 23 octubre de 2006.