Conciencia Tecnológica No. 31, Enero-Junio 2006

Algoritmo Genético Distribuido Sobre PVM (Parallel Virtual Machine)

Reporte de Proyecto

Ing. Marco Antonio Castro Liera, M. C. Jesús Antonio Castro, Lic. Manuel Álvaro Pacheco Hoyo
Instituto Tecnológico de La Paz, Departamento de Sistemas y Computación, Boulevard Forjadores de Sudcalifornia #4720, La Paz, B.C.S. C.P. 23080, Tel: 01(612)1210424 ext. 119, Fax: 01(612)1211295,
email: mcastroliera@acm.org, jcastro@marinos.itlp.edu.mx, mpacheco@ipn.mx

Resumen

En el presente trabajo se describe la creación de la máquina paralela virtual del Departamento de Sistemas y Computación, del Instituto Tecnológico de La Paz la cual ha entrado en operación a partir la segunda mitad de septiembre del 2005.

Aquí se incluyen algunas de las motivaciones que han dado origen a dicho proyecto y conceptos generales sobre cómputo paralelo y máquinas paralelas virtuales. Asimismo se describe el proceso de instalación de PVM (Parallel Virtual Machine) bajo Scientific Linux.

Se muestran los primeros resultados obtenidos de la ejecución de una de las aplicaciones paralelas que se están desarrollando actualmente en el Instituto Tecnológico de La Paz, que es un algoritmo genético distribuido.

Palabras Clave

Cómputo Paralelo, PVM, Algoritmos Genéticos, Linux, Clusters.

Procesamiento Paralelo

En la actualidad el procesamiento paralelo es una de las herramientas más importantes que existen para resolver problemas computacionalmente intensivos. Algunos problemas que pueden requerir días o meses de procesamiento en una computadora con un solo procesador pueden realizarse en fracciones de horas en una computadora paralela.

Una máquina paralela es básicamente aquella que cuenta con dos o más procesadores y por lo tanto puede ejecutar varias instrucciones de manera simultanea, el poder de computación de estas máquinas puede ser incrementado haciendo crecer el número de procesadores con los que cuentan, en vez de reemplazar un sólo procesador con otros cada vez más complejos como ocurre en las computadoras tradicionales.

Desafortunadamente, aunque las computadoras paralelas con un par de procesadores empiezan a ser accesibles para todas las instituciones, el contar con una máquina con una cantidad importante de procesadores en paralelo todavía es demasiado costoso para la mayoría de las instituciones de educación pública, haciendo difícil que los alumnos cuenten con las herramientas necesarias para poder desarrollar aplicaciones en este tipo de ambientes.[1]

Máquinas Paralelas Virtuales (Clusters)

Una solución al problema anterior que está siendo aplicada en varias instituciones de educación superior [2], es la creación de una Máquina Paralela Virtual, en la que un conjunto de computadoras de un solo procesador puedan ser interconectadas para trabajar de manera cooperativa a través del paso de mensajes por medio de una red. Este enfoque presenta diversas ventajas, como:

- Cada uno de los nodos es una máquina que se puede utilizar como una estación de trabajo normal y, en sus tiempos muertos, como un componente de la máquina paralela virtual.
- La construcción de una supercomputadora con una cantidad elevada de microprocesadores es compleja y cara; sin embargo, crear una red con una cantidad elevada de nodos es mucho más sencillo y barato.
- La máquina paralela virtual puede ser altamente tolerante a fallos ya que, con el diseño adecuado de aplicaciones, la desconexión de uno de los nodos no impide que el resto de la máquina siga procesando.
- Escalar una arquitectura de este tipo es sumamente sencillo, ya que sólo se deberán incluir más estaciones de trabajo a la máquina paralela virtual.
- Hoy en día existen diversas herramientas de desarrollo libres y gratuitas que permiten la implementación de una máquina paralela virtual y el desarrollo de aplicaciones que hagan uso de la misma.[3]

Aunque el ancho de banda para la interconexión de las computadoras que conforman una máquina paralela virtual sigue siendo reducido en comparación con el que puede lograrse a través de un bus interno de una computadora paralela, las tecnologías de redes están avanzando de tal manera que esta situación bien podría verse subsanada en un futuro mediato. Sin embargo, siempre es posible considerar esta limitante en el

momento del diseño de las aplicaciones para, así, minimizar la cantidad de información a transmitir entre los diferentes procesadores. Con esto se evita saturar el medio de comunicación.

Cluster del Departamento de Sistemas y Computación

El cluster del Departamento de Sistemas y Computación está compuesto por las siguientes máquinas:

Cuatro computadoras con procesador AMD Athlon XP 3000+ (64 Bits, a 2 GHz, L1 Data caché 64KB, L1 Instruction caché 64 KB, L2 Caché 512 KB) 512 MB de memoria RAM DDR2, accedida por un bus de 400 Mhz. 1 Servidor DELL con procesador Pentium Xeon (32 bits, a 3 GHz, tecnología HT, L1 caché 8 KB, L2 caché 512 KB) 1 GB de memoria RAM accedida por un BUS de 533 Mhz.

Las computadoras se encuentran interconectadas a través de tarjetas Fast-Ethernet y un Switch de 8 puertos.

Para facilitar la configuración de los nodos esclavos, se utiliza un switch KVM que permite controlar 4 nodos con un solo monitor, teclado y ratón; mientras que el servidor cuenta con una consola independiente.

Las computadoras ejecutan la distribución de GNU/Linux Scientific Linux 4.1, PVM 3.4.4 y GCC 4.0.

Scientific Linux

Scientific Linux es una distribución GNU/Linux mantenida por el Fermilab, el CERN (los dos laboratorios de física de altas energías más importantes del mundo) y algunos otros laboratorios y universidades alrededor del mundo.

La versión 4.1 que se utilizó para la creación de la máquina paralela virtual está basada en las fuentes de Red Hat Enterprise Linux 4.0, incluyendo los cambios de la primera actualización de dicha versión.

Algunas de las ventajas que esta versión de Linux ofrece son: la facilidad de instalación, administración y actualización de paquetes gracias a la combinación de los sistemas rpm, yum y apt-get; así como la de incluir herramientas, como LAM/MPI y PVM 3.4.4, para la creación de clusters y programación paralela. El cluster del Departamento de Sistemas está basado en esta última.

PVM: Parallel Virtual Machine

PVM es un paquete de software libre desarrollado por el Oak Ridge National Laboratory. Con PVM es posible hacer que una colección heterogénea de computadoras conectadas en red funcione como una sola computadora paralela, lo cual permite la ejecución y programación de aplicaciones paralelas en red, mediante el modelo de paso de mensajes [4]. PVM está escrito en C y cuenta con librerías que soportan la generación de aplicaciones en C, C++ y Fortran.

Algunas de las características principales de PVM incluyen las siguientes:

- Conjunto de máquinas definido por el usuario: El usuario puede elegir a través de una consola, de entre las máquinas conectadas al cluster, en cuales de ellas se va a ejecutar una tarea. Incluso las máquinas que forman parte de ese conjunto pueden ser agregadas o eliminadas dinámicamente, lo cual es una característica importante para la implementación de tolerancia a fallos.
- Acceso traslúcido al hardware: Los usuarios pueden utilizar un acceso transparente al hardware, en el que se permite que PVM decida la asignación de tareas o puede asignarse de manera explícita cuál máquina ejecutará cada tarea, para explotar mejor las capacidades del hardware.
- Computación basada en procesos: La unidad de paralelismo en PVM es una tarea, que es un hilo independiente de control que se alterna entre los estados de comunicación y cálculos. No existe un mapeo directo entre tareas y procesadores. De hecho es posible ejecutar varias tareas en una sola máquina.
- Modelo de paso de mensajes explícito: Un conjunto de tareas cooperan en la solución de un problema enviándose mensajes entre sí. El tamaño de los mensajes sólo está limitado por la cantidad de memoria de cada computadora.
- Soporte de heterogeneidad: PVM soporta la heterogeneidad en términos de computadoras, redes y aplicaciones. Con respecto al paso de mensajes, PVM permite que mensajes que contienen más de un tipo de datos puedan ser intercambiados entre computadoras con diferentes formas de representación de datos.
- Soporte de multiprocesadores: PVM utiliza las facilidades nativas de paso de mensaje en computadoras paralelas, para hacer uso de las ventajas de este tipo de hardware subyacente. Algunos distribuidores proveen versiones de PVM optimizadas para sus arquitecturas particulares, que pueden comunicarse con las versiones públicas de PVM.

El sistema PVM está compuesto por dos partes: la primera parte es un proceso llamado pvmd3 y a veces abreviado como pvmd que se ejecuta en segundo plano en cada computadora conectada a la máquina virtual. Este proceso es el responsable de mantener la comunicación que permite el paso de mensajes y la creación y eliminación de tareas en el cluster. La

segunda parte es un conjunto de librerías que permiten a las aplicaciones hacer uso de las características que ofrece PVM.

Instalación de una PVM

Un cluster PVM está compuesto por un nodo maestro y varios nodos esclavos.

Para instalar PVM, es necesario que los nodos que van a ser incluidos en el cluster puedan accederse entre ellos mediante sus hostnames. Esto puede lograrse configurando un DNS (Domain Name Service) o agregando dichos nombres en los archivos de hosts de cada uno de los nodos.

Para la ejecución de los procesos de forma remota, PVM utiliza por omisión RSH (Remote Shell). Sin embargo, debido a que este es un protocolo que ya ha caído en desuso, se recomienda que se configure en cada nodo el servicio SSH (Secure Shell). Posteriormente se puede recompilar PVM con la opción de utilizar SSH o simplemente crear una variable de entorno que le indica a PVM que utilice SSH y no RSH como su método de ejecución remota.

Para instalar PVM en Scientific Linux, basta con seleccionar el paquete al momento de instalar el sistema operativo, o instalarlo posteriormente desde los discos de la distribución. También es posible descargar y compilar el código fuente de PVM desde:

http://www.netlib.org/pvm3/index.html

Antes de utilizar el paquete, deben crearse las siguientes variables de entorno en cada nodo del cluster:

- `PVM_ROOT`: que debe contener la ruta donde esta instalado PVM. Para Scientific Linux es: `/usr/share/pvm3`
- `PVM_ARCH`: que debe contener el nombre de la arquitectura para la cual se ha compilado (o se va a compilar) PVM. Por ejemplo `LINUXI386` para computadoras que ejecutan una versión de Linux con un procesador compatible con el 386 de Intel.
- `PVM_RSH`: Que permite seleccionar el programa que se utilizará para lanzar procesos en cada nodo del cluster. Se recomienda utilizar `/usr/bin/ssh`.

Para que estas variables se carguen de manera automática, al iniciar la sesión de cada usuario que vaya a utilizar PVM, se pueden agregar las siguientes líneas a su archivo .bashrc:

```
PVM_ROOT=/usr/share/pvm3
PVM_ARCH=LINUXI386
PVM_RSH=/usr/bin/ssh
export PVM_ROOT PVM_ARCH PVM_RSH
```

De forma opcional, es recomendable agregar al PATH el directorio donde se encuentran las librerías de pvm, lo cual permite la ejecución de la consola y aimk (una herramienta para la compilación automática que funciona sobre make). Esto se logra agregando la cadena $PVM_ROOT/lib a la variable PATH. Por ejemplo:

```
PATH=$PATH:$HOME/bin:$PVM_ROOT/lib
```

Acto seguido, pueden compilarse en cada nodo las aplicaciones de ejemplo de PVM que vienen en el directorio `$PVM_ROOT/examples`. Esto puede hacerse cambiándose a dicho directorio y ejecutando el comando aimk all, el cual compila todos los ejemplos incluidos con PVM.

Los ejecutables son depositados automáticamente por aimk en el directorio:

```
$PVM_ROOT/bin/$PVM_ARCH.
```

Una vez hecho esto en cada nodo del cluster es posible dar de alta una máquina paralela virtual ejecutando el comando PVM en el nodo maestro y agregando, desde esta consola cada uno de los nodos esclavos mediante el comando add <nombre del nodo>, SSH pedirá el password para poder acceder a cada uno de ellos. Para esto, es recomendable que en cada computadora se cree una cuenta con el mismo nombre de usuario, que permita acceder al cluster. El comando quit, permite salir de la consola de pvm sin dar de baja el cluster y ejecutar los ejemplos (como master1 o hello) desde la línea de comandos.

La página del proyecto PVM permite acceder a un manual de referencia muy completo, escrito por sus desarrolladores, que cubre la instalación y el desarrollo de aplicaciones en esta plataforma. Dicho manual está disponible en formato postscript y html en:

http://www.csm.ornl.gov/pvm/pvm_home.html

Aplicaciones

Como parte del trabajo de tesis doctoral del Ing. Marco Antonio Castro Liera ("Optimización Paramétrica de Sistemas Difusos de Identificación Mediante Algoritmos Genéticos Distribuidos"), en el cluster del Departamento de Sistemas y Computación se están llevando a cabo pruebas de desempeño de un algoritmo genético distribuido con migración, escrito en C con las librerías estándar de GCC 4.0.

Para proporcionar una idea aproximada del poder de cómputo del cluster, se presentan los tiempos de ejecución de mil corridas del algoritmo antes mencionado, con 8 poblaciones de 10,000 individuos cada una, un período de migración de 50 generaciones,

una tasa de migración de 10 individuos y un número máximo de 200 generaciones. Este algoritmo se programó para optimizar la función de Goldsten-Price cuya minimización se describe en [5] y está dada por:

$$f_G(x_1,x_2)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\cdot \left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right] \quad (1)$$

Cabe mencionar que esta función tiene un mínimo global en X1=0 y X2 = -1 con un valor de 3

A continuación se muestran los resultados obtenidos.

8 Poblaciones y 1000 Corridas		
Procesadores	Tiempo total	Tiempo promedio
1	1h 6m 3s	3.963s
2	37m 5s	2.225s
3	24m 26s	1.466s
4	17m 56s	1.076s

Tabla 1: Tiempos de ejecución con cuatro nodos (sólo procesadores AMD)

8 Poblaciones y 1000 Corridas		
Procesadores	Tiempo total	Tiempo promedio
1 (Intel)	1h 13m 14s	4.394s
2	39m 12s	2.352s
3	26m 8s	1.568s
4	19m 20s	1.160s
5	19m 26s	1.166s

Tabla 2: Tiempos de ejecución con cinco nodos (Procesadores Intel y AMD)

Con estos parámetros de ejecución, el error promedio, calculado como la diferencia entre el valor de la función para el individuo con mejor aptitud y el mínimo global, es de 0.000056, siendo el error máximo de 0.002338.

Como se puede observar en las tablas anteriores el tiempo de ejecución con 5 procesadores es mayor que el tiempo con 4 procesadores. Esta aparente paradoja tiene la siguiente explicación: La unidad mínima de paralelismo en PVM es una tarea. El algoritmo implementado para esta prueba trata a cada población independiente como una única tarea. Esto implica que cuando se ejecuta la aplicación con 4 computadoras, cada una deberá procesar dos poblaciones; para el caso de 2 computadoras, cada una deberá procesar cuatro poblaciones. En los casos de 3 y 5 computadoras, cada máquina tendrá una cantidad diferente de poblaciones a atender (dos máquinas con tres poblaciones y una con dos poblaciones en el caso de 3 nodos; tres máquinas con dos poblaciones y dos con una población en el caso de 5 nodos). Debido a lo anterior, los tiempos de ejecución no disminuyen de forma inversamente proporcional al número de nodos cuando el número de procesadores no es divisor del total de poblaciones utilizadas en la prueba.

Es por esto que, el tiempo de ejecución con 5 computadoras es mayor que el de 4, ya que la quinta computadora agregada no sólo no disminuye la cantidad máxima de poblaciones (tareas) que el procesador mas ocupado debe ejecutar, sino que, además, hace que PVM ahora deba administrar un nodo extra en el cluster.

El incremento del poder computacional al pasar de 4 a 5 nodos es más evidente usando una cantidad de poblaciones múltiplo de estas cantidades, como se muestra en la siguiente tabla.

20 poblaciones y 1000 corridas			
Arquitectura	Hosts	Tiempo total	Tiempo promedio
AMD	4	48m 47s	2.927s
Intel/AMD	4	49m 49s	2.989s
Intel/AMD	5	39m 42s	2.382s

Tabla 3: Tiempos de ejecución para veinte poblaciones

En este caso se obtuvo un error promedio de 0.000013 y un error máximo de 0.000211.

De momento, la aplicación que se utilizó para obtener los datos anteriores considera a los nodos conectados al cluster como homogéneos, esto es, considera que todos tienen la misma arquitectura y poder de cálculo y, por lo tanto, asigna partes equitativas del trabajo a cada uno. Esto, de momento, genera tiempos muertos en los procesadores más veloces que tienen que esperar a que el más lento termine su ejecución.

Actualmente se esta modificando la aplicación para permitir una asignación de cargas de trabajo proporcional al poder de cada nodo en el cluster, mediante la variación automática de parámetros (el tamaño de la población, el número de generaciones), lo cual permitirá reducir el problema de los tiempos muertos cuando se ejecuten aplicaciones en clusters heterogéneos.

Conclusiones

Las tendencias tecnológicas actuales nos hacen suponer que el desarrollo de aplicaciones paralelas será una de las principales actividades que los profesionales de la computación deberán dominar en un futuro cercano.

Puesto que la mayoría de las instituciones de educación superior en nuestro país no cuentan con los recursos económicos suficientes para la adquisición de supercomputadoras paralelas, la creación de máquinas paralelas virtuales se presenta como una opción viable para que maestros y alumnos se adentren en este campo.

El trabajo con este tipo de tecnología dará a nuestros egresados una importante ventaja competitiva en el futuro mercado laboral.

Referencias

[1] Torres, J. and E. Rodríguez, (2000) *Conceptos de cómputo paralelo*. 1a Ed. ITESM Universidad Virtual. México, D.F. Trillas. 157 p.

[2] Hoffman, F. and W. Hargrove, (1999) "Computación Paralela con Linux". ACM *Crossroads Students Magazine*,. **6** (1), p. 9.

[3] UNAM, *Conceptos Básicos de los Clusters http://clusters.unam.mx/Conceptos/Index.php*. 2003, Departamento de Super cómputo.

[4] Geist, A., et al., (2000) *PVM, Parallel Virtual Machine, A User's Guide and Tutorial for Networked Parallel Computing*. V ed., Cambridge, Massachusetts: MIT Press. 279 p.

[5] Jamshidi, M., et al., (2002), *Robust Control Systems With Genetic Algorithms*. Control Series, Ed. R.H. Bishop., USA: CRC Press. 210 p.