Procesamiento Paralelo en EAFIT

Christian ▪ Trefftz ▪ G.

El procesamiento paralelo se está convirtiendo en una tecnología necesaria para satisfacer las necesidades de cómputo de las entidades medianas y grandes en todo el mundo. Los microprocesadores se han convertido en bienes de consumo masivo y las empresas que los fabrican cuentan con presupuestos muy grandes para la investigación y el desarrollo de nuevos modelos. Los microprocesadores actuales tienen capacidades que rivalizan con las de "mainframes" de hace unos pocos años y la tendencia reciente indica que la velocidad de los microprocesadores se duplica cada 18 meses. Esta tremenda capacidad de cómputo de un microprocesador, paradójicamente, ha hecho

Christian Trefftz G. Profesor del Departamento de Informática y Sistemas, Universidad EAFIT.

que el procesamiento paralelo se vuelva mas atractivo. Para conseguir capacidades de cómputo superiores a las ofrecidas por un solo microprocesador, resulta económicamente justificable fabricar máquinas con varios procesadores.

En este artículo se hace una breve descripción de los conceptos teóricos básicos del área, de los tipos de computadoras paralelas existentes, del software necesario para utilizarlos y de las experiencias en la Universidad EAFIT.

1. CONCEPTOS BÁSICOS

Los sistemas paralelos se justifican porque reducen el tiempo de ejecución o el tiempo de respuesta de una aplicación. Para medir la ganancia en tiempo de ejecución, se usa la medida de Aceleración (Speed Up en inglés) la cual se calcula como:

El procesamiento paralelo se está convirtiendo en una tecnología necesaria para satisfacer las necesidades de cómputo de las entidades medianas y grandes en todo el mundo.

Aceleración = Tiempo de Ejecución en un equipo paralelo / Tiempo de Ejecución secuencial.

El tiempo de ejecución secuencial es el tiempo empleado por el programa en una computadora con un solo procesador. Al citar un valor de aceleración, es necesario citar el número de procesadores con el cual se obtuvo esa aceleración. En los artículos en los que se reportan los resultados de paralelizar una aplicación, usualmente se incluye una gráfica en la que en el eje X está el número de procesadores y en el eje Y está la aceleración lograda. La aceleración ideal es la aceleración lineal en la que la aceleración es igual al número de procesadores utilizados.

Un concepto relacionado es el de Eficiencia. La eficiencia se calcula como:

Eficiencia = Aceleración / Número de Procesadores utilizados

Lo usual es que la eficiencia vaya disminuyendo a medida que crece el número de procesadores. Al crecer el número de procesadores empiezan a aparecer fenómenos de congestión en diferentes componentes del equipo. Cuando un sistema (software y hardware) mantiene una eficiencia aceptable aun para números altos de procesadores, se considera "escalable". Un sistema escalable permite al usuario ir incrementando el número de procesadores en su equipo a medida que van creciendo sus necesidades.

Las aplicaciones paralelas requieren operaciones de sincronización o de intercambio de mensajes. Si estas operaciones ocurren con una alta frecuencia, la aplicación se clasifica como de "granularidad fina". En estas aplicaciones se realizan unas cuantas instrucciones de procesamiento de datos y se realiza una operación necesaria para su funcionamiento paralelo. Las aplicaciones de granularidad gruesa tienen muy pocas operaciones de sincronización o de compartir datos.

Amdahl, en 1968, observó que un programa tiene 2 partes: Una parte que es paralelizable y otra parte que es inherentemente

secuencial, que es imposible de paralelizar. Si llamamos x al porcentaje de un programa que es secuencial, la Ley de Amdahl afirma que la aceleración máxima que se puede obtener para un programa dado es 1/x. Por ejemplo, si un programa tiene una porción secuencial del 10% (0.1), la aceleración máxima es 1/0.1 = 10. Esta aceleración máxima es independiente del número de procesadores utilizados. Continuando con el ejemplo, supongamos que se lograra reducir el tiempo de ejecución de la parte paralelizable a un valor muy pequeño con respecto a la parte secuencial. En este caso, aumentar el número de procesadores no ayuda porque de todas formas no se puede reducir el tiempo de ejecución de la parte secuencial que domina el tiempo de ejecución total.

Los sistemas paralelos se justifican porque reducen el tiempo de ejecución o el tiempo de respuesta de una aplicación.

2. TIPOS DE COMPUTADORAS PARALELAS

Flynn definió una taxonomía de las computadoras paralelas que se utiliza ampliamente. La división de Flynn utilizó dos criterios: El número de instrucciones que se están ejecutando simultáneamente en un momento dado y el número de datos sobre los que se están ejecutando esas instrucciones. Surgen así cuatro categorías: SISD (una Sola Instrucción, un Solo Dato), SIMD (una Sola Instrucción, Múltiples Datos), MISD (Múltiples Instrucciones, un Solo Dato) y MIMD (Múltiples Instrucciones, Múltiples Datos).

En la categoría SISD están la gran mayoría de las computadoras existentes. Son equipos con un solo procesador que trabaja sobre un solo dato a la vez. A estos equipos se les llama también computadoras secuenciales.

Las computadoras SIMD tienen una sola unidad de control y múltiples unidades funcionales. La unidad de control se encarga de enviar la misma instrucción a todas las unidades funcionales. Cada unidad funcional trabaja sobre datos diferentes. Estos equipos son de propósito específico, es decir, son apropiados para ciertas aplicaciones particulares, como por ejemplo el procesamiento de imágenes.

Existe controversia acerca de si realmente existen equipos de tipo MISD. Hay quienes argumentan que estos equipos no existen. Otras personas consideran que un grupo de equipos que trabaja sobre un solo dato se puede considerar como un sistema de tipo MISD. Un ejemplo sería un conjunto de equipos que trata de factorizar un número primo muy grande utilizando diferentes algoritmos.

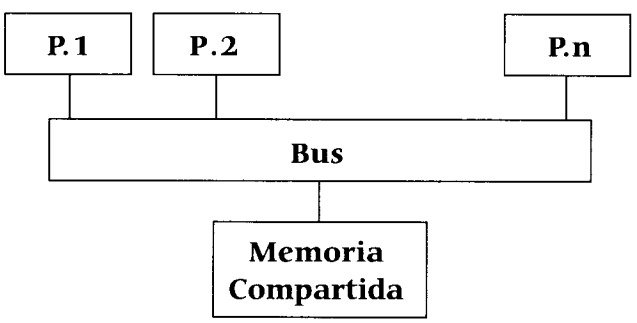
En la categoría MIMD están los equipos con varios procesadores completos. Cada procesador tiene su propia unidad de control y su propia unidad funcional. Esta categoría incluye varios subgrupos: Equipos de memoria compartida, equipos de memoria distribuida y redes de computadores. Los equipos MIMD son de propósito general. Dedicamos las siguientes subsecciones a cada una de estas categorías.

2.1 EQUIPOS MIMD DE MEMORIA COMPARTIDA

En los equipos de memoria compartida existen varios módulos cada uno con su propio procesador y su propia memoria caché (memoria de alta velocidad). Estos módulos están conectados a un bus a través del cual acceden a la memoria principal y a los periféricos. El bus y los módulos tienen circuitería adicional para resolver el problema de la coherencia del caché: En un momento dado puede haber varias copias de una variable de memoria principal en diferentes cachés y cuando un procesador actualice su copia local es necesario que los otros procesadores anulen sus copias respectivas. Estos equipos son de costo muy accesible y relativamente fáciles de programar. Estas máquinas no son muy escalables porque el bus se convierte en un cuello de botella cuando se le instalan al equipo muchos procesadores.

En el siguiente diagrama se ilustra la arquitectura básica de un equipo de este tipo.

FIGURA 1
UNA COMPUTADORA PARALELA CON MEMORIA COMPARTIDA



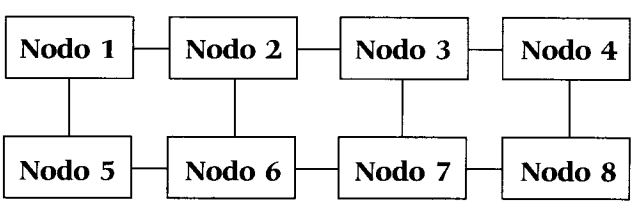
En los equipos de memoria compartida existen varios módulos cada uno con su propio procesador y su propia memoria caché (memoria de alta velocidad). Estos módulos están conectados a un bus a través del cual acceden a la memoria principal y a los periféricos.

2.2 EQUIPOS MIMD DE MEMORIA DISTRIBUIDA

En los equipos de memoria distribuida cada módulo tiene su propia memoria local, su propio procesador y su propia memoria caché. Los módulos se interconectan entre sí a través de algún tipo de red. Hay dos tipos principales de red: Redes de conexión directa, en las que existen enlaces físicos que conectan directamente pares de módulos y redes de múltiples etapas, en los que existen varias capas de suiches enrutadores.

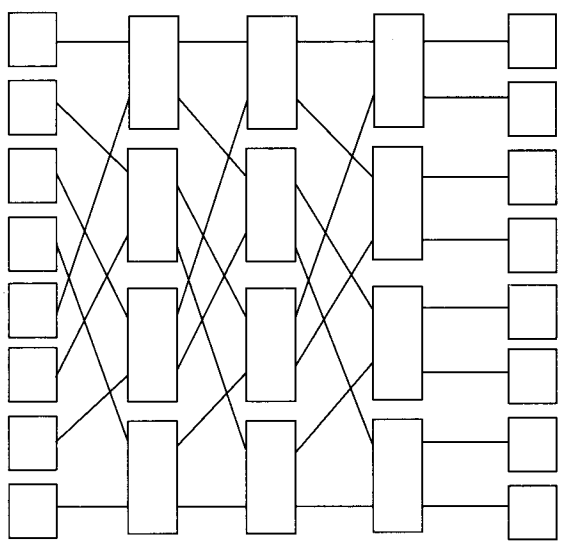
Hay diferentes formas de conectar las máquinas de conexión directa (topologías). Actualmente la topología mas usada es la de red bidimensional, por lo general de formato rectangular. En la siguiente figura se ilustra una red bidimensional de 8 nodos.

FIGURA 2
UNA MÁQUINA PARALELA DE MEMORIA DISTRIBUIDA DE CONEXIÓN DIRECTA (BIDIMENSIONAL)



Las redes de interconexión múltiple tienen como elemento básico un suiche enrutador de n entradas y n salidas. Estos suiches pueden dirigir una entrada a cualquiera de las salidas. Hay ciertas configuraciones bloqueantes cuando dos entradas quieren conectarse a la misma salida. En estos casos uno de los mensajes en tránsito tiene que esperar. Los suiches se agrupan en "etapas" y las etapas permiten que un mensaje viaje desde un procesador a cualquiera de las memorias de los otros procesadores. En la siguiente figura se ilustra una red construida con suiches de 2 entradas y 2 salidas que interconecta 8 procesadores con 8 memorias.

FIGURA 3
UNA MÁQUINA PARALELA DE MEMORIA DISTRIBUIDA CON UNA RED DE MÚLTIPLES ETAPAS



Procesadores **Suiches** **Memorias**

Las máquinas con memoria distribuida se programan, generalmente, mediante paso de mensajes. Existen librerías de funciones que se pueden llamar desde lenguajes como Fortran, C y C++ para poder crear instancias de programas en otros procesadores y para enviar y recibir mensajes para compartir los datos y sincronizar los procesos. En forma experimental se trabaja en sistemas de "Memoria Distribuida Compartida" (Distributed Shared Memory en inglés) en los que se le da al programador la ilusión de estar trabajando en una máquina con memoria compartida a pesar de que el hardware en realidad utiliza memoria distribuida. La programación mediante paso de mensajes es mas difícil que la de memoria compartida. Estas máquinas tienen una capacidad de crecimiento mucho mayor.

2.3 REDES DE ESTACIONES DE TRABAJO

Un caso especial de los equipos de memoria distribuida lo constituyen las redes de estaciones de trabajo (Networks of Workstations en inglés). Es muy común encontrar actualmente en organizaciones medianas y grandes redes de computadoras. Si se ve a la red que interconecta estos equipos como el medio que se puede utilizar para enviar los mensajes y a las computadoras como los módulos se puede considerar a este sistema como una máquina paralela virtual. La programación también se hace mediante paso de mensajes. Dado que usualmente existe una sola red que interconecta todos los equipos, esta red se puede convertir en un cuello de botella y no es posible utilizar eficientemente muchas máquinas. Adicionalmente, el software del sistema operativo y de comunicaciones no ha sido optimizado para enviar y recibir mensajes rápidamente.

3. SOFTWARE

Una computadora sin software es inútil. ¿Qué software hay disponible para equipos paralelos? Hasta ahora no se ha llegado a un modelo único para software para máquinas paralelas y actualmente el software depende del tipo específico de hardware que se esté utilizando. No es posible utilizar el mismo software en máquinas SIMD, de memoria compartida y de memoria distribuida.

3.1 SISTEMAS OPERATIVOS

Cada proveedor de computadoras paralelas ofrece un sistema operativo para los equipos que comercializa. Para ciertos modelos de memoria compartida hasta es posible encontrar varios sistemas operativos disponibles. Las versiones más recientes de estos sistemas operativos son "robustas" y confiables.

Una computadora sin software es inútil. ¿Qué software hay disponible para equipos paralelos? Hasta ahora no se ha llegado a un modelo único para software para máquinas paralelas y actualmente el software depende del tipo específico de hardware que se esté utilizando.

Si una entidad cambia su equipo de un solo procesador por un modelo similar de varios procesadores, usualmente podrá seguir utilizando las mismas aplicaciones sin necesidad de recompilar. La nueva versión del sistema operativo aprovechará la presencia de varios procesadores permitiendo que varias aplicaciones utilicen simultáneamente los diferentes procesadores.

3.2 SISTEMAS ADMINISTRADORES DE BASES DE DATOS

Los principales proveedores de sistemas administradores de bases de datos ofrecen versiones de sus productos que son capaces de explotar la presencia de varios procesadores. Las versiones paralelas utilizan máquinas de tipo MIMD, tanto de memoria compartida como de memoria distribuida. Existe un conjunto de comparaciones (benchmarks) llamados TPC-C que ofrecen una medida del rendimiento máximo que se puede obtener con un determinado paquete. Los mejores rendimientos reportados por los proveedores se obtienen en máquinas paralelas. Las aplicaciones existentes tampoco necesitan ser modificadas para aprovechar la mayor velocidad proporcionada por las versiones paralelas de los manejadores de bases de datos.

3.3 LENGUAJES DE PROGRAMACIÓN

La gran mayoría de las instalaciones no requerirán programar o reprogramar específicamente para máquinas paralelas. Utilizando los sistemas operativos y los paquetes administradores de bases de datos se obtienen mejoras de rendimiento razonables. Pero, ocasionalmente, puede ser conveniente o necesario programar para aprovechar la presencia de varios procesadores. Idealmente, quisiéramos tener compiladores paralelizantes que tomen un programa secuencial y generen código para el tipo de máquina paralela que estemos utilizando. Desafortunadamente los resultados obtenidos han sido decepcionantes. Se obtienen resultados razonables si el programador puede insertar instrucciones especiales para el compilador indicándole que una cierta porción del programa se puede paralelizar.

Para obtener rendimientos mejores es necesario programar teniendo en cuenta el tipo de máquina paralela específica de que se dispone. Hasta el momento no existe un modelo único de programación. Cada tipo de máquina paralela tiene su propio estilo de programación.

Las máquinas SIMD se programan con lenguajes de programación llamados "de datos paralelos" (data parallel en inglés). Estos son lenguajes tradicionales como C y Fortran a los que se les añaden extensiones para especificar como se distribuyen los datos entre las unidades funcionales y para mover los datos entre procesadores cuando sea necesario. Por el diseño del hardware no es necesario preocuparse por sincronización ya que todos los procesadores ejecutan simultáneamente la misma instrucción.

En las máquinas con memoria compartida se utilizan "hilos" (threads en inglés). Un hilo es un proceso independiente con su propio contador de programa y sus propias variables locales. Desde un programa principal es posible activar varios hilos simultáneamente. Los hilos activos comparten el mismo código y las mismas variables globales. Los programadores tienen a su disposición también variables de tipo "mutex" que permiten garantizar que solo un hilo acceda a una sección crítica a la vez. Cada hilo puede ejecutar en un procesador diferente.

El paso de mensajes se utiliza en equipos con memoria distribuida y en grupos de estaciones de trabajo. Los procesos que están colaborando para resolver un problema dado residen en diferentes nodos y se comunican mediante paso de mensajes con los otros procesos tanto para compartir datos como para sincronizarse. Programar mediante paso de mensajes es considerado mas difícil que utilizar hilos.

4. PROCESAMIENTO PARALELO EN EAFIT

En la Universidad EAFIT se han desarrollado varias actividades para difundir la tecnología de procesamiento paralelo: La Universidad ha patrocinado proyectos de investigación, se dictan cursos de procesamiento paralelo y varios grupos de estudiantes, tanto de EAFIT como de la Universidad de Antioquia, han desarrollado proyectos de grado en el área.

En 1995 se realizó un proyecto de investigación en el que se utilizó un grupo de tres microcomputadores con microprocesadores 486 interconectados a través de una red local para procesamiento paralelo. Los microcomputadores utilizados tenían instalado el sistema operativo Linux, de dominio público. En estos microcomputadores se instaló PVM (Parallel Virtual Machine) un paquete, también de dominio público, que permite utilizar un grupo de estaciones de trabajo como si fuera una máquina paralela virtual. Se realizaron pruebas con un programa paralelo que halla los valores propios de matrices tridiagonales simétricas. Se lograron aceleraciones razonables (sublineales dado que las máquinas tenían velocidades diferentes). Es de destacar que con tres equipos 486 se lograron tiempos de

ejecución muy cercanos a los que se obtienen con una estación de trabajo SPARC 10. La Universidad también adquirió una estación de trabajo SPARC 20 con dos procesadores en las que se ha programado con hilos.

En la Universidad EAFIT se han desarrollado varias actividades para difundir la tecnología de procesamiento paralelo: La Universidad ha patrocinado proyectos de investigación, se dictan cursos de procesamiento paralelo y varios grupos de estudiantes, tanto de EAFIT como de la Universidad de Antioquia, han desarrollado proyectos de grado en el área.

Se han dictado cursos de procesamiento paralelo tanto en pregrado en la materia de tópicos especiales de ingeniería de sistemas como en la maestría en ingeniería informática. En estos cursos se les ha dado a los estudiantes la oportunidad de programar en tres ambientes diferentes: En Parallaxis, un simulador de máquinas de tipo SIMD, que le permite a los estudiantes utilizar un lenguaje de datos paralelos, en hilos, en el ambiente Solaris y en PVM. Estos dos últimos ambientes están disponibles en los servidores sigma y delta de la Universidad. Algunos estudiantes por su propia iniciativa han utilizado otros ambientes: Hilos en Windows NT y Java.

Se han adelantado varios proyectos de grado: Omar Gómez y Octavio Quintero paralelizaron POV (un generador de imágenes sintéticas de dominio público) utilizando PVM. Con cinco microcomputadores 486 lograron reducir el tiempo requerido para generar una imagen bastante compleja de una hora a quince minutos. Neil Idárraga y Liliana Jaramillo implementaron un dispositivo de hardware diseñado en la Universidad de Purdue que permite sincronizar varios procesos que están corriendo en equipos diferentes en un tiempo mucho menor que el requerido cuando se utiliza paso de mensajes. Carlos Mario Echeverry y Juan Evangelista Gómez también utilizaron PVM para paralelizar una aplicación de física. En la actualidad se adelantan varios proyectos que tienen que ver con hilos de programación, programación utilizando paso de mensajes orientada por objetos, la paralelización de una aplicación de programación académica (dirigida por el profesor Juan Guillermo Lalinde) y el uso de una versión gráfica de un depurador que trabaja en conjunto con Parallaxis.

5. CONCLUSIONES

El procesamiento paralelo es una tecnología que está adquiriendo una importancia cada vez mayor para las entidades medianas y grandes. La mayoría de las entidades pueden lucrarse de las bondades ofrecidas por esta tecnología sin necesidad de cambiar sus aplicaciones.

Existe una gran variedad de equipos y de software y no se vislumbra que a corto plazo un estilo se imponga definitivamente sobre los demás. Cada tipo de equipo tiene sus fortalezas y desventajas. En la Universidad EAFIT se vienen adelantando actividades para familiarizar a los estudiantes con los principios básicos de esta tecnología.

BIBLIOGRAFÍA

Anderson T., D. Culler, D. Patterson y otros. (1995). A Case for NOW (Networks of Workstations). En: Revista IEEE Micro, pp. 54-64. Este proyecto tiene una página en WWW en http://now.cs.berkeley.edu

Braeunl T. (1993). Parallel Programming - An Introduction. Hertfordshire, G.B., Prentice Hall International, 1993. El proyecto de Parallaxis, desarrollado por el autor de este libro, tiene una página WWW en: http://www.informatik.uni-stuttgart.de/ipvr/bv/p3/p3.html

Geist A., A. Beguelin, J. Dongarra, W. Jiang, R. Machek y V. Sunderam. (1994). PVM: Parallel Virtual Machine - A Users´ Guide and Tutorial for Networked Parallel Computing, Cambridge. The MIT Press. El proyecto de PVM, implantado por los autores de este libro, tiene una página WWW en: http://www.epm.ornl.gov/pvm/

Hwang K. (1993). Advanced Computer Architecture -Parallelism, Scalability, Programmability", New York, E.U., McGraw-Hill, 1993.

Kumar, V. y otros. (1994). Introduction to Parallel Computing - Design and Analysis of Parallel Algorithms, Redwood, E.U., Benjamin Cummins.

Lewis B., and D. Berg. (1996). A Guide to Multithreaded Programming - Threads Primer, E.U., SunSoft Press, Mountain View. Hay información complementaria en el siguiente URL: http://www.sun.com/workshop/sig/threads/