Computación y Sistemas Vol. 1 No. 4 pp. 255-257
© 1998, CIC - IPN. *ISSN 1405-5546* Impreso en México

Parallel Programming with MPI

PACHECO, S. PETER
Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997, 418 pp., ISBN 1-55860-339-5

Por: Sergio Sandoval Reyes
Centro de Investigación en Computación
Instituto Politécnico Nacional, México

Este libro constituye un curso introductorio sobre programación paralela basado en la biblioteca de paso de mensajes MPI (Message Passing Interface). El libro está diseñado para ser utilizado como un tutorial, o como texto en cursos de programación paralela.

Diseño General del libro

El autor ofrece en este magnifico texto, un gran número de ejemplos completamente desarrollados al introducir cada concepto; además describe de manera amplia, aspectos sobre diseño, codificación, depuración y desempeño de programas. Presenta además, diversas técnicas de solución al problema de Entradas/ Salidas en máquinas paralelas. Al final de cada capítulo se incluye un resumen de lo discutido en el mismo y un conjunto de ejercicios así como proyectos de programación.

Antecedentes

El autor presupone que el lector tiene acceso a una máquina paralela (por ejemplo una nCUBE 2, o IBM SP2, o bien a una red de estaciones de trabajo con sistema operativo UNIX), y conocimientos básicos de C. Si bien Fortran también es un lenguaje muy utilizado en la programación paralela, por razones de conveniencia, todos los ejemplos están escritos en C. Sin embargo tanto el código en C, como la versión en Fortran 77 de éstos, están disponibles en Internet en la dirección http: // www.usfca.edu/mpi.

Estructura del libro

El libro está organizado en 16 capítulos y 2 anexos. Excepto por el material de los capítulos 3 a 8, el resto puede leerse en cualquier orden. Los capítulos 3 a 7 forman un tutorial de introducción a MPI y el autor recomienda leerlos en dicho orden y antes de los restantes capítulos. Una breve descripción de los capítulos y anexos, se presenta a continuación.

Capítulo 1. *Introducción*: En este capítulo se discute el porqué se requiere cada vez más poder de cómputo, y el porqué la computación en paralelo puede ser empleada para satisfacer esta necesidad. Se discute además, las dificultades asociadas con la computación paralela y el porqué es necesario aprender a escribir software paralelo portable, así como la importancia del estándar MPI para la programación de computadoras paralelas.

Capítulo 2. *Una revisión de la Computación Paralela*: Aquí se hace una breve revisión de las ideas básicas sobre: 1) la taxonomía de Flynn en términos de flujos de datos e instrucciones (sistemas SISD, SIMD y MIMD: de memoria compartida o multiprocesadora y de memoria distribuida o multicomputadora); 2) redes de interconexión para conectar procesadores con procesadores, procesadores con memoria y procesadores con periféricos (redes dinámicas: bus, multietapa y barras de cruce; y redes estáticas: arreglo lineal, anillo, tutorial y consiste del programa "hello world" implementado con las primitivas *MPI_Send* y *MPI_Recv*. Se muestra además la forma general de programas MPI, y el cómo compilar y correr el programa con uno, dos o varios procesadores.

Se introduce también el concepto de comunicador y la función *MPI_Comm-rank*, para el envío de mensajes entre procesadores.

Capítulo 4. *Una Aplicación: Integración Numérica*: Una vez que se conoce cómo enviar y recibir mensajes, en este capítulo se aplican dichas primitivas de paso de mensajes, para calcular una integral definida, aplicando la regla del trapezoide. Se explica esta aplicación con un algoritmo secuencial y luego con su versión paralela. Por último, se discute los aspectos de lectura de datos y escritura de resultados del programa.

Capítulo 5. *Comunicación Colectiva*: En este capítulo se hacen varias mejoras de la aplicación del capítulo anterior. El propósito es mejorar el desempeño a través de la comunicación colectiva entre los procesos. Para ello se explican los diversos modos de comunicación del comunicador: difusión (broadcast), reducción (reduction), recolección (gather) y dispersión (scatter). Se muestra que el desempeño de los modos difusión y reducción puede mejorarse, si los procesos son vistos como un árbol y la comunicación (de datos y resultados) procede a lo largo de las ramas del árbol. Se explica por último, la importancia de la comunicación asíncrona con registros temporales de almacenamiento (buffered) para evitar situaciones de interbloqueaje (deadlock).

Capítulo 6. *Agrupamiento de Datos para la Comunicación:* En este capítulo se hace notar que actualmente en los sistemas paralelos, el enviar un mensaje es una operación costosa. Es decir, que mientras menos mensajes se manden, mejor será el desempeño global del programa. Esto implica que será necesario agrupar los datos y enviarlos con el menor número de mensajes. Se muestra así, como MPI provee tres mecanismos para agrupar datos en un solo mensaje: 1) conteo de parámetros, 2) tipos derivados mediante las directiva *MPI_Type_contiguous* y *MPI_Type_vector* y 3) el empaquetamiento y desempaquetamiento mediante la directiva *MPI_Pack/MPI_Unpack*. Se discute por último, el cómo decidir qué método usar en términos de la complejidad del mensaje.

Capítulo 7. *Comunicadores y Topologías:* Este es el último capítulo del tutorial. Aquí se estudian dos implementaciones del algoritmo de Fox para la multiplicación de matrices en paralelo, haciendo uso de comunicadores y topologías. Un comunicador es una colección de procesos que pueden enviarse mensajes entre sí. Una topología es una estructura impuesta sobre los procesos de un comunicador, que permite que los procesos sean direccionados en formas diferentes. Es decir, los procesos se direccionan en base a una estructura lógica como podría ser una malla de dos o tres dimensiones, o un anillo, etc. El énfasis del algoritmo es mapear filas y columnas de matrices, a procesos y luego ver dichos procesos como un arreglo virtual en forma de malla o submallas, en la que cada submalla constituye un universo de comunicación.

Capítulo 8. *Manejo de Entradas y Salidas:* Este capítulo discute diversas soluciones al problema de llevar a cabo operaciones de Entrada/Salida en computadoras paralelas. Aunque no se discute ninguno de los esfuerzos actualmente en desarrollo para estandarizar o acelerar dichas operaciones, si en cambio se discuten algunos de los problemas asociados con las operaciones de Entrada/Salida, y los intentos para desarrollar algunas funciones que sean razonablemente portables en computadoras paralelas.

Capítulo 9. *Depuración de Programas:* Aquí se analiza el espinoso problema de la depuración de programas paralelos. Si los programas fueran diseñados con perfección matemática, naturalmente no se requeriría "espulgarlos" (debugging). El hecho es que es más difícil espulgar programas paralelos que secuenciales, porque los primeros podrían exhibir indeterminismo debido a "condiciones de carrera", es decir, procesos que compiten entre sí para completar tareas; o bien un programa presumiblemente libre de errores en un sistema, podría no funcionar en otro, etc. El énfasis pues, se centra en reducir el tiempo de espulgado, a través de un cuidadoso diseño, desarrollo y prueba de los programas.

Capítulo 10. *Diseño y Codificación de Programas Paralelos:* En este capítulo se discuten dos métodos para el diseño de programas paralelos: Paralelismo de Datos, y Paralelismo de Control. El primero particiona los datos entre los procesos, y cada proceso ejecuta más o menos el mismo código sobre sus datos (se ilustra esto con un programa que resuelve un sistema de ecuaciones lineales usando el método de Jacobi). El segundo particiona las tareas entre los procesos, y el código que ejecuta cada proceso es esencialmente diferente. Se muestra además, el diseño y codificación de programas paralelos con un ejemplo que ordena una lista distribuida.

Capítulo 11. *Desempeño*: Este capítulo se centra en los métodos para estimar el desempeño de programas paralelos. La idea es estimar el tiempo que tarda un programa en correr en un procesador, y compararlo con n procesadores. La relación $S = T(1)/T(n)$ produce la ganancia en velocidad (*Speedup gain*). Se introducen además, conceptos como tiempo de comunicación, tiempo de cálculo, eficiencia, utilización, y latencia (o su recíproco ancho de banda), relacionados todos ellos con el desempeño. Por último, se muestra cómo medir el tiempo de ejecución de los procesos mediante las directivas *MPI_Wtime, MPI_Wtick* y *MPI_Barrier*.

Capítulo 12. *Más sobre el Desempeño*: Aquí se continúa la discusión sobre el desempeño introduciéndose su dependencia con respecto a la Ley de Amdahl (cuando el número de procesadores tiende a infinito, la ganancia en velocidad S está limitada por la porción de código secuencial que haya que ejecutarse); y el concepto de Escalabilidad (mantener una eficiencia dada, a medida que el número de procesos y procesadores se incrementa, aumentando simultáneamente el tamaño del problema). Se discute por último, el problema de estimar el desempeño de un programa paralelo, y el cómo analizar dicho desempeño con MPI (*MPI profile* y *MPI Upshot*).

Capítulo 13. *Comunicación Avanzada Punto a Punto*: En este capítulo se discuten funciones de comunicación punto a punto más versátiles como *MPI_Sendrecv* y *MPI_Sendrecv_replace*, las cuales proveen un medio simple de organizar comunicaciones en pares para evitar interbloqueaje (deadlock). Se continúa con una discusión de las funciones no bloqueantes *MPI_Isend* y *MPI_Irecv*, como un medio para sobrelapar las actividades de cálculo y de comunicación. Se termina este capítulo con una revisión de los modos de comunicación, para indicarle al sistema si se desea que los mensajes se almacenen temporalmente ("bufereen"), y si es así, en dónde deben serlo.

Capítulo 14. *Algoritmos Paralelos*: En este capítulo se revisan brevemente los aspectos involucrados en el diseño de algoritmos paralelos y se señala lo amplio del campo. Los algoritmos paralelos se pueden dividir en forma muy general, en algoritmos que son más o menos paralelizaciones directas de algoritmos secuenciales, y algoritmos que no lo son. El primer tipo se ilustra con un ejemplo de ordenamiento "bitónico"; el segundo con un ejemplo de búsqueda en un árbol paralelo. En este último se enfatiza un problema muy conocido en la computación paralela y distribuida: la detección de terminación de procesos distribuidos. Para dicho problema se implementa una solución.

Capítulo 15. *Bibliotecas de Funciones Paralelas*: Una de las mejores características de MPI es su soporte para el desarrollo de bibliotecas de funciones paralelas portables. Este capítulo proporciona una breve introducción al uso de dos bibliotecas que MPI emplea, para la solución de problemas en computación científica: ScaLAPACK y PETSc. La primera se utiliza en álgebra lineal especialmente en matrices densas; y la segunda en el diseño y desarrollo de programas para resolver ecuaciones lineales, ecuaciones no-lineales y ecuaciones diferenciales. El propósito de ambas, es hacer disponible software paralelo.

Capítulo 16. *Adónde continuar desde aquí*: Este último capítulo proporciona diversas fuentes de información adicional sobre MPI, y se presenta una breve discusión sobre las futuras direcciones en el desarrollo de MPI.

Apéndice A. *Resumen de Funciones de MPI*: Aquí se sumariza alfabéticamente, la sintaxis de cada función de MPI y se listan sus constantes y definiciones de tipos.

Apéndice B. *MPI en Internet*: Este apéndice contiene un reducido número de fuentes de MPI disponibles en Internet. Las fuentes incluyen: 1) direcciones de donde se pueden bajar implementaciones de MPI; 2) preguntas sobre MPI; y 3) páginas Web.

Bibliografía. Incluye 37 referencias actualizadas.

En sintesis, este libro es indispensable para aprender programación paralela con MPI.