Introducción a los Problemas Paralelos

Práctica 1

Vicente Martín Rueda
Pablo Requena González
Marcos González Verdu
21/09/2015

Introducción a los Problemas Paralelos

Práctica 1

Indice

INTRODUCCIÓN

Es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente operando sobre el principio de que problemas grandes se pueden dividir en más pequeños, lograr gracias a una mejora en que después son resueltos simultáneamente. Esto se ha conseguido gracias a los avances del hardware ya que se requieren medios físicos para hacer esta división y realizarla de forma eficaz y adecuada. Además hay que diferenciar paralelismo de concurrencia, y a la vez de un sistema distribuido, que podría decirse que es paralelismo computacional con varios ordenadores (suponiendo que cada ordenador solo tenga un core).

¿PARA QUÉ SIRVE EL PARALELISMO EN COMPUTACIÓN?

Es una función realizada por el procesador para ejecutar varias tareas al mismo tiempo, de forma paralela, se basa en un ejemplo claro de "divide y vencerás", el cual dice que un problema grande puede dividirse en varios subproblemas, solucionar los subproblemas y combinar las soluciones parciales para conseguir solucionar el problema completo.

Hay diferentes formas:

- **Paralelismo a nivel de bit.**
- **Paralelismo a nivel de instrucción.**
- **Paralelismo de datos.**
- **Paralelismo de tareas.**

La computación en paralelo se ha convertido en el paradigma dominante en los procesadores multinúcleo, debido múltiples limitaciones físicas de las máquinas, como pueden ser el aumento de la frecuencia o el consumo de energía que tanta importancia ha ganado en los últimos años.

Paralelismo a nivel de bit

Es la realización de una misma operación lógica a distintos bits.

Paralelismo a nivel de instrucción

Es una reordenación y combinación en grupos de instrucciones que después son ejecutadas en paralelo sin que el resultado de las instrucciones cambie.

Paralelismo de datos

Es el paralelismo inherente en programas con ciclos, que se centra en la distribución de los datos entre los diferentes nodos computacionales que deben tratarse en paralelo.

Paralelismo de tareas

Es la característica de un programa paralelo en la que "cálculos completamente diferentes se pueden realizar en cualquier conjunto igual o diferente de datos". Esto contrasta con el paralelismo de datos, el cual realiza el mismo cálculo en distintos o mismos grupos de datos. Por lo general, el paralelismo de tareas no escala con el tamaño del problema.

¿EL PARALELISMO PUEDE IMPLICAR ALGÚN PROBLEMA?

Como veremos un poco más adelante sí, puede implicar numerosos problemas aunque también tiene numerosas ventajas, como reducir el tiempo de las acciones a realizar, lo cual se traduce en tiempo y en menor coste energético.
Por la propia definición de paralelismo, éste requiere un gran número de ciclos de procesamiento o acceso a una gran cantidad de datos; es difícil encontrar un hardware y un software que aproveche estas utilidades sin generar inconvenientes de costos, seguridad y/o disponibilidad.

Problema de Planificación

Esto surge cuando hay que escoger el orden en el cual un número dado de actividades deben ser realizadas. Quizás, el problema que más ha influenciado esta área sea el clásico problema de secuencias de trabajos en una planta de producción, ya que los procesos de manufacturación envuelven varios tipos de operaciones para transformar el material que se va recibiendo en un nuevo producto. El problema es determinar la secuencia ideal de esas operaciones de acuerdo a cierto criterio, como puede ser el económico. Solución: Las muchas técnicas usadas para resolver este problema vienen desde la enumeración completa de las posibles soluciones, hasta la programación entera y las técnicas heurísticas.

Problema de Asignación de Tareas

El orden de precedencia de las tareas no es importante para el caso en sí, lo único que nos interesa de este problema es que la asignación tiene que hacer un uso eficiente de los recursos. En las arquitecturas MIMD la clave es el coste de las operaciones de comunicación, mientras que en las arquitecturas SIMD encontramos problemas de asignación de otros tipos, y se soluciona con un algoritmo de asignación de tareas que minimice el tiempo de ejecución de la aplicación.

Problema de Asignación de Datos

A los archivos se accede mediante operaciones de actualización o recuperación de usuarios o aplicaciones, por esto, el sistema debe resolver problemas tales como la asignación de los archivos, o sus niveles de replicación o de fragmentación para conseguir óptimos niveles de rendimiento.

Problema de Distribución de Carga de Trabajo

Es un problema que se crea al no saber bien la manera de distribuir las tareas sobre el sistema, dado un conjunto de tareas y un sistema computacional.

Problema de Partición de Datos/Programas

Este problema se da al dividir un programa o archivo en componentes que puedan ser ejecutados concurrentemente.

Problema de Tolerancia a Fallas

El sistema falla cuando no sigue sus especificaciones funcionales. Puede ser un fallo no muy importante aunque también puede llegar a ser catastrófico. Un fallo puede ser por un error de diseño, de programación, de fabricación, de operación, etc. Aun así, no todos los fallos de un componente implican un fallo total del sistema.

Ejemplo de paralelismo

El algoritmo que vamos a estudiar es el fork/join, que consiste en dividir un problema en varios subproblemas que se resuelven en paralelo, y la suma de las soluciones parciales, será la solución del problema inicial.

El ejemplo concreto de este algoritmo, será el MergeSort. Este algoritmo de ordenación, funciona dividiendo en dos el problema inicial, hasta que el sub-problema alcanza el tamaño mínimo, en ese momento, al contener sólo una unidad, el sub-problema está ordenado. Entonces utilizamos la función 'merge' para sumar subproblemas ordenados en orden, de manera que cuando la primera llamada resuelve su 'merge', la suma de todas las soluciones parciales ordenadas es la solución final al problema y el vector o array con el que estamos trabajando queda ordenado.

La paralización del MergeSort viene del hecho de que el algoritmo ya divide el problema en subproblemas más pequeños y podemos trabajar con ellos de forma paralelizada, y la lista de elementos queda ordenada por el orden en que se suman las soluciones de los subproblemas, por tanto, al importar el orden en que se resuelven, el primer problema que encontramos al paralelizarlo es la planificación.

Conclusiones

No todos los problemas se pueden resolver mediante paralelización.

Hay que buscar una buena relación rendimiento/precio.

Una de las mejores estrategias en problemas grandes, siempre que se pueda dividir en subproblemas.

BIBLIOGRAFÍA

- Paralelismo (Informática). EcuRed. Disponible en: http://www.ecured.cu/index.php/Paralelismo_%28_inform%C3%A1tica%29

- Computación paralela. Wikipedia. Disponible en: https://es.wikipedia.org/wiki/Computaci%C3%B3n_paralela

- Paralelismo (informática) Wikipedia. Disponible en: https://es.wikipedia.org/wiki/Paralelismo_%28inform%C3%A1tica%29

- Introducción a la Computación Paralela. J. Aguilar, E.Leiss. Disponible en: http://www.ing.ula.ve/~aguilar/publicaciones/objetos/libros/ICP.pdf

- Adictos al trabajo. Natalia Rosales Gonzákez. Disponible en: http://www.adictosaltrabajo.com/tutoriales/el-paralelismo-en-java-y-el-framework-forkjoin/