Soporte del Lenguaje para Mejorar el Rendimiento - Descomposición del Problema*

José Enrique Alvarez Estrada

Translated By:

José Enrique Alvarez Estrada

Based on *Language Support for Performance - Problem Decomposition*† by

Charles Severance

Kevin Dowd

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License ‡

Existen tres enfoques principales sobre cómo dividir o descomponer un trabajo, de modo que pueda distribuirse entre múltiples CPUs:

Descomposición de los cálculos: Ya hemos discutido esta técnica. Cuando la descomposición se hace basándose en los cálculos, llegamos a algún mecanismo para dividir equitativamente los cómputos (tales como las iteraciones o un bucle) entre nuestros procesadores. Generalmente se ignora la ubicación de los datos, y las preocupaciones principales son la duración de cada iteración y la uniformidad. Esta es la técnica predilecta para los sistemas de memoria uniformemente compartida, porque cualquier procesador puede acceder a los datos de modo igualitario.

Decomposición de los datos: Cuando el acceso a la memoria no es uniforme, la tendencia es enfocarse en la distribución de los datos en vez de los cómputos. Se asume que la recuperación de los datos "remotos" es costosa y por tanto debe minimizarse. Los datos se distribuyen entre las memorias. El procesador que contiene el dato realiza los cálculos sobre dicho dato tras recuperar cualquier otro dato necesario para realizarlos.

Decomposición de las tareas: Cuando las operaciones que deben realizarse son muy independientes entre sí, y toman algún tiempo, puede llevarse a cabo una descomposición de tareas. En este enfoque un proceso o hilo de ejecución maestro mantiene una cola de las unidades de trabajo. Cuando un

*Version 1.1: May 14, 2011 12:39 pm -0500

†http://cnx.org/content/m33762/1.3/

‡http://creativecommons.org/licenses/by/3.0/

procesador dispone de recursos, recupera la siguiente "tarea" de la cola y comienza a procesarla. Se trata de un enfoque muy atractivo para cálculos embarazosamente paralelos.[1]

En cierto sentido, el resto del capítulo está dedicado primordialmente a la descomposición de datos. En un sistema de memoria distribuida, el costo de comunicación usualmente es el factor de rendimiento dominante. Si su problema es tan violentamente paralelo que puede distribuirse entre tareas, prácticamente cualquier técnica funcionará. Los problemas con datos paralelos ocurren en muchas disciplinas. Pueden variar desde aquellos que son extremadamente paralelos, a aquellos otros que sólo lo son un poco. Por ejemplo, los cálculos fractales son extremadamente paralelos; cada punto se deriva independientemente del resto. Es sencillo dividir los cálculos fractales entre los procesadores. Y dado que los cálculos son independientes, los procesadores no requieren coordinarse o compartir datos.

Nuestro problema de flujo calórico, cuando se expresa en su forma rojo-negro (o FORTRAN 90) es extremadamente paralelo, pero requiere de cierto grado de compartición de datos. Un modelo gravitatorio o una galaxia es otra clase de programa paralelo. Cada punto ejerce una influencia sobre cualesquiera otro. Así, y al contrario que los cálculos fractales, los procesadores deben compartir datos.

Sea cual sea el caso, usted quiere acomodar los cálculos de forma que los procesadores puedan decirse los unos a los otros, "empieza a partir de aquí y trabaja en esto, mientras que yo trabajo en esto otro, y lo reuniremos nuevamente cuando hayamos terminado."

Aun los problemas que ofrecen menos independencia entre regiones son todavía muy buenos candidatos para la descomposición del dominio. Los problemas de diferencias finitas, la simulación de la interacción entre partículas próximas, y las columnas de matrices pueden tratarse de forma similar. Si puede usted dividir el dominio igualmente entre los procesadores, cada uno realizará aproximadamente la misma cantidad de trabajo en su camino hacia la solución.

Otros sistemas físicos no son tan regulares, o involucran interacciones de rango aomplio. Los nodos de una rejilla no estructurada pueden no estar acomodados en correspondencia directa con sus ubicaciones físicas, por ejemplo. O tal vez el modelo involucre fuerzas de rango amplio, tales como la atracción entre partículas. Estos problemas también pueden estructurarse para máquinas paralelas, aunque resulte algo más difícil. A veces se requiere de varias simplificaciones, o de "agrupamiento" de efectos intermedios. Por ejemplo, la influencia de un grupo de partículas distantes sobre otro puede tratarse como si se tratar de partículas compuestas actuando a distancia. Con ello se ahorran las comunicaciones que se requieren que cada procesador haga para hablar con cada uno de los otros, acerca de cada detalle. En otros casos, la arquitectura paralela ofrece oportunidades de expresar un sistema físico en formas distintas e ingeniosas, que tienen sentido en el contexto de la máquina. Por ejemplo, puede asignarse cada partícula a su propio procesador, de forma que puedan deslizarse unas sobre otras, agregando interacciones y actualizando un ciclo de tiempo.

Dependiendo de la arquitectura de la computadora paralela y del problema, elegir entre dividir o replicar (porciones de) el dominio puede agregar una sobrecarga de trabajo o costos inaceptables para todo el proyecto.

En problemas grandes, el costo económico de la memoria principal puede dejar fuera de toda discusión el mantener copias locales separadas del mismo dato. De hecho, a menudo es la necesidad de más memoria lo que conduce a las personas hacia las máquinas paralelas; el problema que requieren resolver no cabe en la memoria de una computadora convencional.

Invirtiendo algo de esfuerzo, puede usted permitir que el particionamiento del dominio evolucione conforme el programa se ejecuta, en respuesta a una distribución de carga dispareja. De esta forma, si hubiera muchas solicitudes para A, entonces varios procesadores podrán obtener dinámicamente una copia de la pieza A del dominio. O bien la pieza A puede estar dispersa entre varios procesadores, cada uno manejando un subconjunto distinto de definiciones de A. Incluso puede usted migrar copias únicas de datos de un lugar a otro, cambiando su residencia conforme se requiera.

[1] El esfuerzo distribuido para romper la clave RC5 fue coordinado de esta forma. Cada procesador recibió un bloque de claves y comenzó a probar dichas claves. En algún punto, si los procesadores no eran lo suficientemente rápidos o dejaban de funcionar, el sistema central reasignaba el bloque a otro procesador. Esto permite al sistema recuperarse de problemas en las computadoras individuales.

Cuando el dominio de datos es irregular, o cambia a lo largo del tiempo, el programa paralelo encuentra un problema de balance de cargas. Tal problema se vuelve esencialmente aparente cuando toma mucho más tiempo en completarse una porción de los cálculos paralelos que otros. Un ejemplo de la vida real puede ser un análisis de ingeniería de una retícula adaptativa. Conforme se ejecuta el problema, la retícula se vuelve más refinada en aquellas áreas que muestran mayor actividad. Si el trabajo no se reacomoda de vez en cuando, la sección de la computadora responsable de la sección de la retícula con mayor refinamiento sufrirá una caída progresiva del rendimiento respecto al resto de la máquina.