Introducción al multiprocesamiento

Pedro O. Pérez M., Phd.

Diseño de sistemas embebidos avanzados Tecnológico de Monterrey

pperezm@tec.mx

08-2023

① Conceptos básicos

Multiprogramación y multiprocesamiento Multiprogramación Multiprocesamiento

Paralelismo vs. Concurrencia

2 Modelo de computación

Single Instruction stream, Single Data stream - SISD Single Instruction stream, Multiple Data stream - SIMD Multiple Instruction stream, Single Data stream - MISD Multiple Instruction stream, Multiple Data stream - MIMD

3 Problemas relacionados con el paralelismo

Condición de carrera Interbloqueo (Deadlock) Coherencia del cache

4 Tipos de paralelismo

Paralelismo a nivel de instrucción

Nivel de bits Pipelining Super-escalar

Paralelismo a nivel de procesador

Arreglo de computadoras Multiprocesadores Algoritmos paralelos

Cálculo de desempeño

- **5** Modelos de algoritmos paralelos
 - Paralelo de datos
 - Grafo de tareas
 - Work pool
 - Maestro-esclavo
 - Productor-consumidor

Multiprogramación

La multiprogramación (multiprogramming) se desarrolla como una solución para el uso eficiente del CPU

- Un solo usuario no puede mantener al CPU y a los dispositivos de E/S ocupados toso el tiempo.
- La multiprogramación organiza los trabajos para que el CPU siempre tenga algo que hacer.
- Existe una fila de trabajos (o tareas) en memoria. Desde esta fila, un trabajo es seleccionado a través de un algoritmo de calendarización.
- Cuando un trabajo está esperando a que termine una operación de E/S, el sistema operativo lo desaloja para poder ejecutar otro trabajo.

Multiprocesamiento

- Han crecido en uso e importancia.
- También conocidos como sistemas paralelos, sistemas estrechamente acoplados.
- Ventajas: mayor rendimiento, economía a escala, confiabilidad.
- Dos tipos: asimétricos (un procesador controla los demás), simétricos (cada procesador realiza la tarea que haya disponible).

Paralelismo vs. Concurrencia

- El cómputo paralelo está fuertemente relacionado con el cómputo concurrente. Es muy frecuente se empleen juntas, incluso que se combinen; sin embargo son muy diferente: podemos tener paralelismo sin concurrencia (como el paralelismo a nivel de instrucción) y podemos tener concurrencia sin paralelismo (como la multitarea).
- En el cómputo paralelo, una tarea se divide en varias sub-tareas muy similares que se pueden procesar de forma independiente y cuyos resultados se combinan para formar la solución final.
- En la computación concurrente, los diversos procesos no ejecutan tareas relacionadas; cuando lo hacen, las tareas pueden ser de una naturaleza variada y, muy a menudo, requieren alguna comunicación entre procesos durante la ejecución.

Paralelismo

Paralelismo es el proceso de ejecutar varias instrucciones simultáneamente. Su objetivo es reducir el tiempo de ejecución total. El paralelismo requiere de hardware con múltiples procesadores. Las computadoras paralelas requieren algoritmos paralelos, lenguajes de programación, compiladores y sistemas operativos que admitan la multitarea.



Fuente: https://upload.wikimedia.org/ wikipedia/commons/thumb/d/d3/IBM_Blue_ Gene_P_supercomputer.jpg/1280px-IBM_Blue_ Gene_P_supercomputer.jpg

Modelo de computación

Tanto las computadoras secuenciales como las paralelas trabajan con un conjunto (o flujo) de instrucciones y un cojunto de datos de entrada. Según el flujo de instrucciones y el flujo de datos, las computadoras se pueden clasificar en cuatro categorías:

- Flujo de instrucciones único, flujo de datos únicos (Single Instruction stream, Single Data stream - SISD).
- Flujo de instrucciones únicos, flujo de datos múltiples (Single Instruction stream, Multiple Data stream - SIMD).
- Flujo de instrucciones múltiples, flujo de datos único (Multiple Instruction stream, Single Data stream - MISD).
- Flujo de instrucciones múltiples, flujo de datos múltiples (Multiple Instruction stream, Multiple Data stream - MIMD).



Single Instruction stream, Single Data stream - SISD

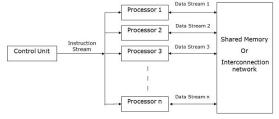
En este tipo de computadoras, el procesador recibe un único flujo de instrucciones y opera con un único flujo de datos procedente de la memoria.



Fuente: https://www.tutorialspoint.com/parallel_algorithm/images/sisd_computers.jpg

Single Instruction stream, Multiple Data stream - SIMD

- Una sola unidad de control envía instrucciones a todas las unidades de procesamiento.
- Cada una de las unidades de procesamiento tienen su propia unidad de memoria para almacenar tanto los datos como las instrucciones
- Algunos procesadores se encuentra ejecutando su conjunto de instrucciones, mientras otros esperan su siguiente conjunto.

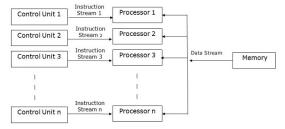


Fuente:

https://www.tutorialspoint.com/parallel_ algorithm/images/simd_computers.jpg

Multiple Instruction stream, Single Data stream - MISD

Cada procesador tiene su propia unidad de control y comparten una unidad de memoria común. Todos los procesadores reciben instrucciones individualmente de su propia unidad de control y operan en un solo flujo de datos según las instrucciones que han recibido de sus respectivas unidades de control.

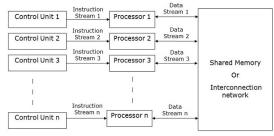


Fuente:

https://www.tutorialspoint.com/parallel_ algorithm/images/misd_computers.jpg

Multiple Instruction stream, Multiple Data stream - MIMD

Cada procesador tiene su propia unidad de control, unidad de memoria local y unidad aritmética y lógica. Reciben diferentes conjuntos de instrucciones de sus respectivas unidades de control y operan con diferentes conjuntos de datos.



Fuente:

https://www.tutorialspoint.com/parallel_ algorithm/images/mimd_computers.jpg

- Una computadora MIMD que comparte una memoria común se conoce como multiprocesadores, mientras que las que utilizan una red de interconexión se conocen como multicomputadoras.
- Según la distancia física de los procesadores, los multicomputadoras son de dos tipos:
 - Multicomputadora: cuando todos los procesadores están muy cerca unos de otros (por ejemplo, en la misma habitación).
 - Sistema distribuido: cuando todos los procesadores están lejos unos de otros (por ejemplo, en las diferentes ciudades).

En parejas,

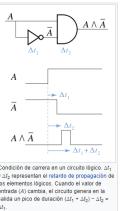
- Encuentra la respuesta a las siguientes preguntas,
 - ¿Es posible hacer una supercomputadora con dispositivos tipo Raspberry Pi? Muestra ejemplos de tu respuesta.
 - ¿Qué hardware y software requerimos para hacerlo?
 - ¿Qué tipo de modelo de computación (de los mencionados antes) sería y por qué?
- Coloca todos tus hallazgos en un documento y súbelo a Canvas. Recuerda colocar las referencias usadas.

Problemas relacionados con el paralelismo

- Condición de carrera.
- Interbloqueo.
- Coherencia del caché.

Condición de carrera

La condición de carrera se da cuando la salida o estado de un proceso es dependiente de una secuencia de eventos que se ejecutan en orden arbitrario y van a trabajar sobre un mismo recurso compartido, se puede producir un error cuando dichos eventos no se ejecutan en el orden que el programador esperaba.

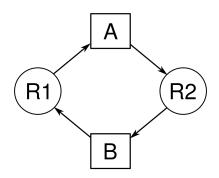


Condición de carrera en un circuito lógico. Δt₁ v \(\Delta t_0\) representan el retardo de propagación de los elementos lógicos. Cuando el valor de entrada (A) cambia, el circuito genera en la salida un pico de duración $(\Delta t_1 + \Delta t_2) - \Delta t_3 =$

Fuente: https: //en.wikipedia.org/wiki/Race_condition

Interbloqueo (Deadlock)

El interbloqueo (deadlock) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos.



Fuente:

https://upload.wikimedia.org/wikipedia/commons/thumb/9/99/DeadlockGraph.svg/336px-DeadlockGraph.svg.png

- La memoria cache es un componente usado por el CPU para reducir el costo promedio (en tiempo o energía) de acceso a los datos que se encuentran en memoria principal.
- La mayoría de los CPUs tienen varias memorias cache independiente, de datos o instrucciones, que se encuentran organizados en una jerarquía de varios niveles (L1, L2, etc.).
- La información que se cargan en la memoria cache depende de algoritmos sofisticados y ciertas suposiciones sobre el código del programa. El objetivo del sistema cache es garantizar que el CPU tenga el siguiente bloque de datos que necesitará ya cargado en memoria.

Coherencia del cache

La situación se complica en un entorno multiprocesador donde, además de mantener registros internos, cada una de los CPU también contiene una caché local. En tal entorno, una copia de A puede existir simultáneamente en varias cachés. Dado que los distintos CPU pueden ejecutarse todas en paralelo, debemos asegurarnos de que una actualización del valor de A en una caché se refleje inmediatamente en todas las demás cachés donde A reside. Esta situación se denomina coherencia de caché y suele ser un problema de hardware (que se maneja por debajo del nivel del sistema operativo).

https://bit.ly/2TPTmhy

Tipos de paralelismo

- Paralelismo a nivel de instrucción.
 - Nivel de bits.
 - Pipelining.
 - Super-escalar.
- Paralelismo a nivel de procesador.
 - Arreglo de computadoras.
 - Multiprocesadores.

- El paralelismo a nivel de bit es una forma de computación paralela basada en el aumento del tamaño de la palabra del procesador. Aumentar el tamaño de palabra reduce el número de instrucciones que el procesador debe ejecutar para realizar una operación en variables cuyos tamaños son mayores que la longitud de la palabra.
- Otra manera es el aumento del ancho del bus de datos externo. Por ejemplo, DDR1 transfiere a 128 bits por ciclo de reloj, mientras que DDR2 transfiere un mínimo de 256 bits por ráfaga.

Pipelining

- Un programa de computadora es, en esencia, una secuencia de instrucciones ejecutadas por un procesador. Si el paralelismo a nivel de instrucción, un procesador solo podría ejecutar menos de una instrucción por ciclo de reloj (Instruction Per Cycle, IPC < 1). Este tipo de procesadores se conocen como sub-escalares.
- Sin embargo, las instrucciones pueden reordenarse y combinarse en grupos que luego se ejecutan en paralelo sin cambiar el resultado del programa.

- Todos los procesadores modernos tiene tuberías de varias etapas. Cada etapa en la tubería corresponde a una acción diferente que el procesador realiza en una instrucción en esa etapa; un procesador con una tubería de N etapas puede tener hasta N instrucciones diferentes en diferentes etapas de ejecución (ciclo Fletch-Decode-Execute) y, por lo tanto, puede emitir una instrucción por ciclo de reloj (IPC = 1). Estos procesadores se conocen como escalares. Un ejemplo son los procesadores RISC que cuentan con cinco etapas: búsqueda de instrucción (IF), decodificación de instrucción (ID), ejecución (EX), acceso a memoria (MEM) y recuperación de registro (WB).
- Considera que un Intel Core i7-7700K tiene un IPC, usando un solo hilo, de 174.

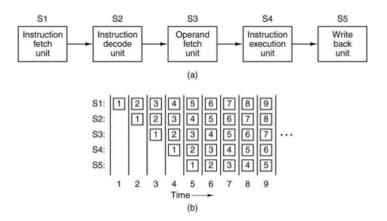


Figura: https://bit.ly/2MuP1PG

Super-escalar

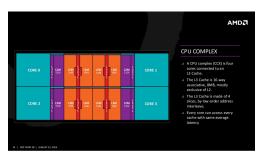
- Sin embargo, la mayoría de los procesadores actuales tiene múltiples unidades de ejecución. Por lo tanto, si combinamos el IPC de un solo hilo con los múltiples hilos ejecución tenemos un mayor IPC. Estos procesadores se conocen como super-escalares.
- De nuevo, tomemos el ejemplo del Intel Core i7-7700K, el cual cuenta con 8 hilos de ejecución. Este procesador super-escalar tiene un IPC de 872.

Arreglo de computadoras

- Computación en malla. La computación en malla (en inglés grid computing) es una tecnología que permite utilizar de forma coordinada recursos heterogéneos (entre ellos procesadores, almacenamiento y aplicaciones específicas) que no están sujetos a un control centralizado. Es una forma de computación distribuida, en la cual los nodos participantes pueden ser de iguales o distintas arquitecturas y cubrir toda la gama de potencia de cómputo, desde embebidos hasta supercomputadoras.
- Computación en Clúster. El término clúster (del inglés cluster, que significa 'grupo' o 'racimo') se aplica a los sistemas distribuidos de granjas de computadoras unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen un único servidor. A diferencia de la computación en malla, los clústeres de computadoras tienen a cada nodo realizando la misma tarea, controlada y planificada por software.

Multiprocesadores

- Un multiprocesador puede verse como un computador paralelo compuesto por varios procesadores interconectados que comparten un mismo sistema de memoria.
- Los sistemas multiprocesadores son arquitecturas MIMD con memoria compartida. Tienen un único espacio de direcciones para todos los procesadores y los mecanismos de comunicación se basan en el paso de mensajes desde el punto de vista del programador.



¿Dónde se están usando este tipo de algoritmos?

Revisa este página:

https://builtin.com/hardware/parallel-processing-example

En parejas,

- Investiga los siguiente términos,
 - Robots paralelos.
 - Sistemas multi-robots.
- ¿Qué significa cada concepto? ¿Cuáles ventajas y/o desventajas tienen con respecto a sus contraparte tradicionales? ¿En dónde se utilizan?
- Coloca todos tus hallazgos en un documento y súbelo a Canvas. Recuerda colocar las referencias usadas.

Algoritmos paralelos

- Como recordarás, un algoritmo es una secuencia finita de pasos que toma un conjunto de entradas para producir una tarea. Pues bien, un algoritmo paralelo es un algoritmo que puede ejecutar varias instrucciones simultáneamente en diferentes procesadores y luego combinar todas las salidas individuales para producir el resultado final.
- No es fácil dividir un problema en subproblemas. Nos enfrentamos a problemas tales como condición de carrera, interbloqueos o coherencia de cache.
- También hay que considerar el tiempo que necesitan los procesadores para comunicarse entre sí, ya que éste es mayor que el tiempo real de procesamiento. Por lo tanto, al diseñar un algoritmo paralelo, se debe considerar la utilización adecuada de la CPU para obtener un algoritmo eficiente.

Cálculo de desempeño

La velocidad de un programa es el tiempo que tarda el programa en ejecutarse. Esto podría medirse en cualquier incremento de tiempo. SpeedUp se define con el tiempo que le toma a un programa ejecutarse secuencialmente (con un procesador) dividido por el tiempo que lleva ejecutarse en paralelo (con muchos procesadores o hilos).

La fórmula para calcular el SpeedUp es: Donde:

- S_p La mejora obtenida al usar p procesadores.
- T_1 El tiempo que tarda el programa en ejecutarse secuencialmente.
- T_p El tiempo que tarda el programa en ejecutarse usando p procesadores

$$S_p = \frac{T_1}{T_p}$$

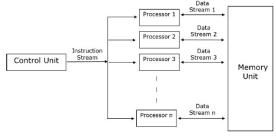
Modelos de algoritmos paralelos

El modelo de un algoritmo paralelo se desarrolla considerando una estrategia para dividir los datos y el método de procesamiento y aplicando una estrategia adecuada para reducir las interacciones. Entre los modelos existentes, encontraremos:

- Paralelo de datos.
- Grafo de tareas.
- Work pool.
- Maestro-esclavo.
- Producto-consumidor.
- Híbrido

Paralelo de datos

- Las tareas se asignan a los procesos y cada tarea realiza tipos similares de operaciones en diferentes datos.
- Se puede aplicar en espacios de direcciones compartidas y paradigmas de paso de mensajes.
- La interacción se pueden reducir seleccionando una localidad que preserva la descomposición, utilizando rutinas de interacción colectiva optimizadas o superponiendo el cálculo y la interacción.
- Ejemplos: Multiplicación de matrices densas

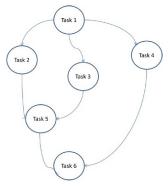


Fuente:

https://www.tutorialspoint.com/parallel_algorithm/images/data_parallel_model.jpg

Grafo de tareas

- El paralelismo se expresa mediante un grafo de tareas.
- La correlación entre las tareas se utiliza para promover la localidad o minimizar los costos de interacción.
- Este modelo se utiliza cuando la cantidad de datos es mayor que el cálculo asociado con las tareas.
- Las tareas se asignan para ayudar a mejorar el costo del movimiento de datos entre las tareas.
- Ejemplos:Quick sort paralelo, factorización de matrices escasas, algoritmos paralelos basados en la técnica Divide y conquista.

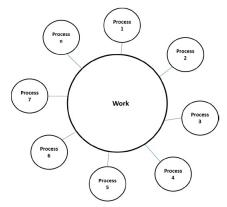


Fuente:

https://www.tutorialspoint.com/parallel_ algorithm/images/task_graph_model.jpg

Work pool

- Las tareas se asignan dinámicamente a los procesos para equilibrar la carga.
- Este modelo se utiliza cuando la cantidad de datos es menor que el cálculo asociado con las tareas
- La tarea puede estar disponible al principio o puede generarse dinámicamente.
- Ejemplos: árboles de búsqueda paralelos.



Fuente:

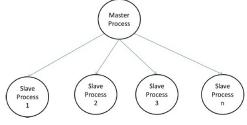
https://www.tutorialspoint.com/parallel_ algorithm/images/work_pool_model.jpg

Maestro-esclavo

En el modelo maestro-esclavo, uno o más procesos maestros generan tareas y las asignan a procesos esclavos. Las tareas pueden asignarse de antemano si:

- El maestro puede estimar el volumen de las tareas
- Una asignación aleatoria puede hacer un trabajo satisfactorio de equilibrio de carga
- A los esclavos se les asignan tareas más pequeñas en diferentes momentos.

Este modelo es generalmente igualmente adecuado para paradigmas de espacio de direcciones compartidas o de paso de mensajes, ya que la interacción es naturalmente de dos formas.

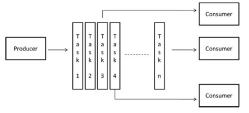


Fuente:

https://www.tutorialspoint.com/parallel_algorithm/images/master_slave_model.jpg

Productor-consumidor

También se conoce como modelo "pipeline". Un conjunto de datos se transmite a través de una serie de procesos, cada uno de los cuales realiza alguna tarea en él. La llegada de nuevos datos genera la ejecución de una nueva tarea por un proceso en la cola. Los procesos podrían formar una cola en forma de matrices lineales o multidimensionales, árboles o grafos generales con o sin ciclos.



Fuente:

https://www.tutorialspoint.com/parallel_ algorithm/images/pipeline_model.jpg