S. Paralelos - Clase 1 - 2025

Fundamentos de procesamiento paralelo

Qué es el procesamiento paralelo? es el uso de múltiples unidades de procesamiento para resolver un problema computacional, dividiendolo en partes que se puedan resolver de forma concurrente, cada parte se divide en instrucciones que se ejecutan simultaneamente en diferentes procesadores.

Procesamiento concurrente, paralelo y distribuido

- Concurrente -> puede tener múltiples tareas avanzando en cualquier instante de tiempo.
- Paralelo -> múltiples tareas que se ejecutan simultáneamente cooperan para resolver un problema. Cooperan y comparten recursos. Busca reducir el tiempo de ejecución de un programa empleando múltiples procesadores al mismo tiempo.
- Distribuido -> múltiples tareas que se ejecutan físicamente en diferentes lugares cooperan para resolver uno o más problemas con diferentes recursos locales. Conjunto de computadoras autónomas interconectadas que cooperan compartiendo recursos.

Sobre HPC

El *Cómputo de Alto Desempeño* o *HPC (High Performance Computing)* utiliza sistemas de gran poder computacional y técnicas de procesamiento paralelo para resolver problemas complejos con alta demanda computacional.

Plataformas de procesamiento paralelo

Clasificación

1. Mecanismo de control (taxonomía de Flynn):

> De acuerdo a cómo se especifica el paralelismo entre instrucciones y datos

- SISD (Single Instruction, Single Data): Ejecución secuencial de instrucciones sobre un único flujo de datos. Instrucciones ejecutadas en forma secuencial, una por ciclo de reloj. *Ejemplo: mainframes, monoprocesadores*
- SIMD (Single Instrucion, Multiple Data): todas las unidades de procesamiento ejecutan la misma instrucción sobre diferentes datos de forma sincrónica y determinística. Adecuado para problemas con alto grado de regularidad como procesamiento de imágenes.
- MIMD (Multiple Instruction, Multiple Data): las unidades de procesamineot ejecutan diferetnes instrucciones sobre diferentes datos. Sincrónica, asincrónica, determinística o no. Es la clase más común de máquina paralela (*clústers, multiprocesadores multicore, etc*)
- MISD (Multiple Instruction, Single Data): las unidades de procesamiento ejecutan diferentes instrucciones el sobre el mismo dato. Modelo teórico.

2. Organización física:

> Se realiza de acuerdo al espacio de direcciones que tiene cada procesador.

- Memoria compartida: sus procesadores acceden a todad la memoria como un único espacio de direcciones global. Éstos operan en forma independiente pero que comparten los mismos recursos de memoria. Requiere mecanismos de coherencia de caché y tiene prolemas de escalabilidad. Subclasificación por modo de acceso a memoria: - UMA (*Acceso Uniforme a Memoria*) - NUMA (*Acceso No Uniforme a Memoria*)

- Memoria distribuida: cada procesador tiene su propia memoria y la comunicación requiere que el programador defina cómo y cuándo se comunican los datos entre procesos a través de una red interconexión. No requiere un mecanismo de coherencia de caché. Cada procesador accede forma más rápida a los datos. El acceso a datos remotos es mediante NUMA que es más lento.

- Memoria híbrida: Los sistemas más grandes combinan características de memoria compartida y distribuida, interconectando múltiples máquinas de ambos modelos para que sus procesadores puedan comunicarse.

Modelos de programación paralela

- Memoria compartida: múltiples tareas acceden a un espacio de memoria común para comunicación y sincronización y además cada una puede tener memoria local "exclusiva". Usado en plataformas de memoria compartida como multiprocesadores o multicores.

- Pasaje de mensajes: consiste en procesos con espacios de direcciones exclusivos (particionado), donde toda interacción requiere la cooperación de dos procesos. El intercambio de mensajes sirve para el intercambio explícito de datos y la sincronización de procesos. Usaado en plataformas de memoria distribuida como clusters.

Evolución del poder computacional

La mejora de los procesadores estuvo guiado por 2 factores: aumento en el número de transistores en el chip (Ley de Moore) y el aumento de la frecuencia del reloj. Esto permitió mejorar el rendimiento de los procesadores secuenciales como **ILP (Instruction Level Parallelism)** incluyendo el pipelining.

Pipelining

Consiste en solapar las diferentes etapas de la ejecución de instrucciones, reduciendo el tiempo de ejecución total. Dado que a mayor nivel de división del pipeline se incrementa la velocidad las compañías lo aumentaron pero la organización es más compleja y consumía mucha energía, aparte lo saltos condicionales retrasaban el rendimiento entonces se optó en usar *múltiples pipelines*.

Planificacion de instrucciones

El planificador analiza un conjunto de instrucciones de la cola de instrucciones a ejecutar y emite aquellas que pueden ser ejecutadas en forma concurrente, teniendo en cuenta las dependencias: • Si las instrucciones son ejecutadas en el orden en que aparecen en la cola, se dice que la emisión es *en orden* -> Simple pero limita significativamente la emisión de instrucciones • Si el procesador tiene la habilidad de reordenar las instrucciones en la cola, entonces se puede alcanzar el máximo rendimiento posible -> este modelo se

conoce como *fuera de orden* (o de emisión dinámica) y, aunque es más complejo, es el que se usa en la actualidad

La mejora de los procesadores de forma tradicional llega a tener limitaciones:

- **Memory Wall**: diferencia creciente entre velocidad de procesador y memoria
- **ILP Wall**: dificultad para exxtraer más paralelismo a los programas
- **Power Wall**: limitaciones de disipación de calor.

Cambio de paradigma

En lugar de seguir aumentando la compleja organización interna del chip, las empresas fabricantes optaron por integrar dos o más núcleos computacionales (también llamados cores) más simples en un sólo chip (multicores).

> Los procesadores multicore proveen el potencial de mejorar el rendimiento sin necesidad de aumentar la frecuencia de reloj, lo que los vuelve más eficientes energéticamente

Multi-hilado (SMT)

Es una técnica complementaria que permite mantener más de un hilo de ejecución al mismo tiempo en el procesador para aprovechar mejor las unidades de ejecución. Entonces, los recursos asociados al estado del procesador son replicados (contador de programa, registros, etc) de esta forma el procesador parece tener múltiples núcleos y por ende puede ejecutar múltiples flujos en paralelo, sin importar si pertenecen al mismo programa o a diferentes.

Los procesadores multicore tienen jerarquías de memoria caché de múltiples niveles. El nivel 1 siempre es privado, los siguientes niveles varían de acuerdo a la arquitectura.

Clusters

es una colección de computadoras individuales interconectadas por una red (e.g., Ethernet, Infiniband) que trabajan juntas como un único recurso integrado de cómputo. Cada nodo de procesamiento es un sistema de cómputo en sí mismo, con hardware y sistema operativo propio.
Dado que los clusters son arquitecturas distribuidas, el modelo de programación más utilizado suele ser pasaje de mensajes.

Anexo

UMA (Uniform Memory Access)

En esta arquitectura, todos los procesadores comparten un único espacio de memoria con tiempos de acceso uniformes.

- La memoria está conectada a través de un bus común o una interconexión central.
- Es utilizada en sistemas SMP (Symmetric Multiprocessing), donde todos los procesadores tienen acceso equitativo a la memoria. Ventaja: Es más fácil de programar y gestionar. Desventaja: A medida que aumenta el número de procesadores, el bus de memoria puede convertirse en un cuello de botella.

NUMA (Non-Uniform Memory Access)

Aquí, cada procesador tiene su propia memoria local, pero también puede acceder a la memoria de otros procesadores, aunque con mayor latencia.

- Se organiza en nodos donde cada uno tiene su propio conjunto de CPU y memoria.
- Es usada en sistemas multiprocesador de alto rendimiento, como servidores y supercomputadoras. Ventaja: Reduce la congestión del bus y mejora la escalabilidad. Desventaja: El acceso a memoria remota es más lento, lo que requiere optimización en el software para minimizarlo.

Dependencias

- Dependencia verdadera de datos (True data dependency): el resultado de una instrucción es la entrada para la siguiente
- Dependencia de recurso (Resource dependency): dos operaciones requieren el mismo recurso (por ejemplo, unidad de punto flotante)
- Dependencia de salto (Branch dependency): las instrucciones a ejecutar después de un salto condicional no pueden ser determinadas a priori sin tener margen de error.