Sistemas Paralelos - Clase 2 - 2025

Memory Wall: Limitaciones en el rendimiento del sistema de memoria

El término "Memory Wall" describe la disparidad cada vez mayor entre la velocidad a la que operan los procesadores y la velocidad a la que pueden acceder a los datos de la memoria principal. La limitación del rendimiento por la velocidad del sistema de memoria (**latencia** y **ancho de banda**) en comparación con la velocidad del procesador es un factor crítico en el diseño de plataformas paralelas eficientes.

Latencia -> Tiempo que transcurre desde que el procesador solicita el dato hasta que el mismo está disponible. Ancho de banda -> Velocidad con la cual el sistema puede transferir datos desde la memoria al procesador.

> Es fundamental para reconocer que aumentar el número de procesadores por sí solo no garantiza una mejora lineal del rendimiento. El diseño de algoritmos paralelos debe minimizar las dependencias de la memoria y buscar estrategias para ocultar o reducir la latencia.

Reducción de la latencia usando cachés

Las caché son memorias de alta velocidad y baja capacidad, se encuentran integradas al chip. Actúan como memoria intermedia entre los registros de la CPU y la memoria principal.

Funcionamiento básico

Cuando el procesador necesita un dato, primero verifica si está en la caché (**cache hit**). Si lo está, el acceso es muy rápido. Si no está (**cache miss**), debe buscarse en la memoria principal, lo que es mucho más lento. El dato se copia entonces a la caché para futuras accesos.

En caso de: - Caché hit -> pedido se satisface rápido -> baja latencia - Caché miss -> el dato es copiado en la caché antes de llegar a los registros

La efectividad de la caché se basa en el principio de **localidad temporal**, que establece que los datos que se han utilizado recientemente tienen una alta probabilidad de volver a utilizarse en un futuro cercano (suposición de que habrá una repetición de referencias a determinados datos en una ventana de tiempo pequeña).

> Al diseñar algoritmos paralelos, es crucial considerar cómo se acceden a los datos. Las estructuras de bucle, la reutilización de datos y la organización de los cálculos deben optimizarse para maximizar los aciertos de caché y reducir los costosos accesos a la memoria principal.

Impacto del ancho de banda

> El ancho de banda es la velocidad con la que los datos pueden ser transferidos desde la memoria al procesador y está determinado por el ancho de banda del bus de memoria como de las unidades de memoria

Técnicas como la transferencia de bloques de memoria más grandes en cada ciclo de reloj ayudan a mejorar el ancho de banda efectivo.

> El sistema de memoria requiere l unidades de tiempo (latencia) para obtener b unidades de datos (b es el tamaño del bloque medido en bits, bytes o words)

El principio de localidad espacial es fundamental. Si se accede a una ubicación de memoria, es probable que se acceda a las ubicaciones cercanas en un futuro cercano. Al organizar los datos de manera contigua y accederlos secuencialmente, se aprovecha mejor el ancho de banda, ya que se traen varios datos relacionados en un solo acceso a memoria.

Consecuencias de patrones de acceso no lineales

Cuando los programas paralelos acceden a la memoria de manera no contigua (por ejemplo, saltando a través de grandes bloques de memoria o accediendo por columnas en arreglos almacenados por filas), se reduce drásticamente la efectividad de la localidad espacial y se desaprovecha el ancho de banda. Esto puede llevar a un rendimiento muy pobre. Al dividir los datos entre los procesadores o hilos en un programa paralelo, es esencial considerar cómo cada unidad de procesamiento accederá a su porción de datos. Un diseño deficiente puede resultar en patrones de acceso no lineales y un rendimiento subóptimo.

Arreglos multidimensionales y su organización en memoria

Los arreglos multidimiensionales son estructuras de datos fundamentales en muchas aplicaciones de HPC. Existen dos maneras en que los datos de un arreglo son almacenados en memoria:

- Por filas (*Row-Major*): los elementos de una misma fila se almacenan de forma contigua en la memoria.
- Por columnas (*Column-Major*): los elementos de una misma columna se almacenan de forma contigua.

Definición de arreglos en C

Arreglo dinámico como vector de elementos

Ventajas: • Favorece al aprovechamiento de la localidad de datos • Hace posible el uso de instrucciones SIMD • Facilita el intercambio de arreglos entre programas escritos en diferentes lenguajes

Coherencia de caché en arquitecturas multiprocesador

En sistemas de memoria compartida donde múltiples procesadores tienen sus propias cachés, es esencial garantizar la coherencia de caché. Esto significa que si un procesador modifica un dato en su caché, todos los demás procesadores que tengan una copia de ese dato en sus cachés deben ser notificados y sus copias deben ser actualizadas o invalidadas para mantener la consistencia de los datos. El mecanismo de coherencia debe asegurar que todas las operaciones realizadas sobre las múltiples copias son *serializables* -> tiene que existir algún orden de ejecución secuencial que se corresponde con la planificación paralela.

Protocolos de coherencia de caché

- **Invalidación**: Cuando un procesador escribe en un bloque de memoria compartida, las copias de ese bloque en las cachés de otros procesadores se marcan como inválidas. La próxima vez que un procesador necesite ese dato, deberá obtener una copia actualizada de la memoria (o de otra caché que la tenga modificada). Cada copia se asocia con uno de 3 estados:
    - compartida (shared): múltiples copias válidas del datos en diferentes memorias. Ante una escritura, pasa a estado sucia donde se produjo mientras que el resto se marca como inválida,

    - inválida (invalid): la copia no es válida. Ante una lectura, se actualiza a partir de la copia válida (la que está en estado sucia),
    - sucia (dirty): la copia es válida y se trabaja con esa. **Consideraciones**:
    - Reduce el tráfico de red en situaciones donde un dato compartido se escribe con poca frecuencia por múltiples procesadores. Las actualizaciones solo se realizan cuando otro procesador realmente necesita el dato modificado.
    - Si un procesador lee un dato una vez y no lo vuelve a usar, las actualizaciones posteriores por otros procesadores no generan overhead innecesario en su caché.
    - Si los procesadores comparten una variable y la modifican alternativamente, el dato puede ser constantemente invalidado y tener que ser re-obtenido, lo que genera latencia adicional.
    - Puede llevar a un "ocio" o "stalling" mientras un procesador espera a que se actualice un dato invalidado.

- **Actualización**: Cuando un procesador escribe en un bloque compartido, las copias en otras cachés se actualizan con el nuevo valor, se propaga la nueva versión del datos a todas las otras cachés que lo contienen. Entonces, los procesadores que tienen el dato en su caché tendrán la versión actualizada inmediatamente sin necesidad de acceder a la memoria principal. **Consideraciones**:

    - Puede ser beneficioso cuando los datos compartidos se modifican con frecuencia y son accedidos repetidamente por otros procesadores, ya que evita la necesidad de re-obtener el dato tras cada invalidación.
    - Puede ser ligeramente mejor en situaciones de false sharing ya que todas las lecturas pueden ser locales y solo las escrituras deben ser actualizadas, ahorrando una operación de invalidación.
    - Genera más tráfico de red ya que cada escritura a un dato compartido resulta en actualizaciones a todas las cachés que lo contienen, incluso si esos procesadores no necesitan inmediatamente el valor actualizado.
    - Si un procesador simplemente lee un dato una vez y no lo vuelve a usar, las actualizaciones posteriores de ese dato por otros procesadores causan un overhead excesivo en términos de latencia en el origen y ancho de banda en la red.

> Actualmente, la mayoría de las máquinas con coherencia de caché se basan en protocolos de invalidación. Esto se debe a que, en muchas situaciones, el costo del tráfico de red generado por las actualizaciones constantes puede ser mayor que el costo de las invalidaciones ocasionales y las posteriores re-obtenciones de datos.

Implementación de protocolos de coherencia de caché

- **Sistemas de Caché Snoopy**: En sistemas conectados por un bus compartido (o una red de broadcast), cada caché "espía" (snoops) las transacciones del bus. Si un procesador escribe en una dirección que otro procesador tiene en su caché, el protocolo de snooping asegura la invalidación o actualización de la copia. Estos sistemas son relativamente simples pero pueden tener problemas de escalabilidad a medida que aumenta el número de procesadores debido al tráfico en el bus.
- **Sistemas Basados en Directorios**: Se utiliza un directorio en la memoria principal para mantener información sobre qué procesadores tienen en caché cada bloque de memoria compartida. Cuando un procesador necesita acceder a un dato compartido, consulta el directorio para determinar qué acciones de coherencia son necesarias. Esto permite operaciones de coherencia más selectivas y mejora la escalabilidad en comparación con los sistemas snoopy, aunque introduce overhead por la gestión del directorio. También existen sistemas de directorios distribuidos para mejorar aún más la escalabilidad.

Costos de comunicación