TEMA 1:

ARQUITECTURAS PARALELAS: CLASIFICACIÓN Y PRESTACIONES

Lección 1: CLASIFICACIÓN DEL PARALELISMO IMPLÍCITO EN UNA APLICACIÓN.

Criterios de clasificaciones del paralelismo implícito en una aplicación.

Clasificación del paralelismo implícito:

- Niveles de paralelismo implícito en un código que resuelve la aplicación.
- Paralelismo de tareas – Paralelismo de datos.
- Granularidad.

Niveles de paralelismo implícito en una aplicación.

En una aplicación se pueden distinguir distintos niveles de paralelismo, que se aprovechan en diferentes niveles del computador. Se pueden clasificar los niveles de paralelismo de una aplicación en función del nivel de abstracción dentro del código secuencial de un programa en el que se puede encontrar implícito el paralelismo. Se considera que un programa está compuesto por funciones (nivel de funciones), éstas a su vez están compuestas de bucles (nivel de bucles), y éstos se basan en operaciones (nivel de operaciones).

Dentro del código secuencial se puede encontrar paralelismo implícito en los siguientes niveles de abstracción:

- **Nivel de programas:** Los diferentes programas que intervienen en una aplicación o en diferentes aplicaciones se pueden ejecutar en paralelo. Es poco probable que exista dependencia entre ellos.
- **Nivel de funciones:** En un nivel de abstracción más bajo, un programa puede considerarse constituido por funciones. Las funciones llamadas en un programa se pueden ejecutar en paralelo, siempre que no haya dependencias inevitables, como las dependencias verdaderas.
- **Nivel de bucle (bloques):** Una función puede estar basada en la ejecución de uno o varios bucles. El código dentro de un bucle se ejecuta múltiples veces, en cada iteración se completa una tarea. Se pueden ejecutar en paralelo las iteraciones de un bucle, siempre que se eliminen los problemas derivados de dependencias verdaderas. Para detectar las dependencias habrá que analizar las entradas y salidas de las iteraciones del bucle.
- **Nivel de operaciones:** Las operaciones independientes se pueden ejecutar en paralelo. En los procesadores de propósito general y especifico, podemos encontrar instrucciones compuestas de varias operaciones que se aplican en secuencia al mismo flujo de datos de entrada. En este nivel se puede detectar la posibilidad de usar instrucciones compuestas, que van a evitar las penalizaciones por dependencias verdaderas.

A este paralelismo que se puede detectar en distintos niveles de un código secuencial se le ha denominado paralelismo funcional.

Paralelismo de tareas – Paralelismo de datos.

También se ha clasificado el paralelismo en paralelismo de tareas y paralelismo de datos.

El **paralelismo de tareas** se encuentra extrayendo, generalmente de la definición de la aplicación, la estructura lógica de funciones de la aplicación. En esta estructura, los bloques son funciones, y las conexiones entre ellos reflejan el flujo de datos entre funciones. Equivaldría al paralelismo a nivel de función dentro del código de alto nivel.

El **paralelismo de datos**, se encuentra implícito en las operaciones con estructuras de datos (vectores y matrices). Las operaciones vectoriales y matriciales engloban múltiples operaciones con datos escalares, que se puede realizar en paralelo. Las operaciones con vectores y matrices se implementan mediante bucles. Por tanto, el paralelismo de datos está relacionado con el paralelismo a nivel de bucle. El paralelismo de datos se puede extraer de los bucles analizando las operaciones realizadas con la misma estructura de datos en las diferentes iteraciones de un bucle.

Granularidad.

El paralelismo puede también clasificarse en función de la granularidad o magnitud de la tarea (número de operaciones) candidata a la paralelización. El grano más pequeño (grano fino) se asocia al paralelismo entre operaciones o instrucciones, y el grano grueso al paralelismo entre programas. Entre ambos extremos está el grano medio, asociado a los bloques funcionales lógicos de la aplicación.

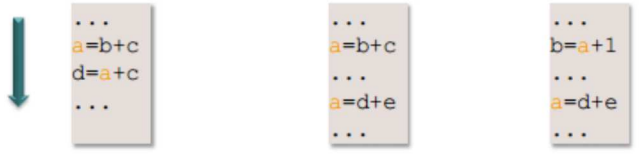
Dependencias de datos:

Condiciones que se deben cumplir para que bloque de código B2 presente dependencia de datos con respecto a B1.

- Deben hacer referencia a una misma posición de memoria M (variable).
- B1 aparece en la secuencia de código antes que B2.

Tipos de dependencias de datos (de B2 respecto a B1):

- **RAW** (Read After Write) o dependencia verdadera (primera imagen).
- **WAW** (Write After Write) o dependencia de salida (segunda imagen).
- **WAR** (Write After Read) o antidependencia (tercera imagen).



Paralelismo implícito (nivel de detección), explícito y arquitecturas paralelas.

El paralelismo ente programas se utiliza a nivel de procesos. En el momento en el que se ejecuta un programa, se crea el proceso asociado al programa. El paralelismo disponible entre funciones se puede extraer para utilizarlo a nivel de procesos o hebras. El paralelismo dentro de un bucle, se puede extraer también a nivel de procesos o de hebras.

Se puede aumentar la granularidad asociando un mayor número de iteraciones del ciclo a cada unidad a ejecutar en paralelo. El paralelismo dentro de un bucle también se puede hacer explícito dentro de una instrucción vectorial para que sea aprovechado por arquitecturas SIMD o vectoriales. El paralelismo entre operaciones se puede aprovechar en arquitecturas con paralelismo a nivel de instrucción (ILP) ejecutando en paralelo las instrucciones asociadas a estas operaciones independientes.

Nivel de paralelismo explícito. Unidades en ejecución en un computador.

El hardware se encarga de gestionar la ejecución de instrucciones. A nivel superior el SO se encarga de gestionar la ejecución de unidades de mayor granularidad, procesos y hebras.

Una hebra tiene su propia pila y contenido de registros, entre ellos un puntero de instrucciones, pero comparte el código, las variables globales y otros recursos con las hebras del mismo proceso. Estas características hacen que las hebras se puedan crear y destruir en menor tiempo que los procesos, y que la comunicación, sincronización y conmutación entre hebras de un proceso sea más rápida que entre procesos. Esto permite que las hebras puedan tener una granularidad menor que los procesos.

El paralelismo implícito en el código se puede hacer explícito a nivel de instrucciones, a nivel de hebras o a nivel de procesos.

Instrucciones: La unidad de control de un core o procesador gestiona la ejecución de instrucciones por la unidad de procesamiento.

Thread o light process:

- Menor unidad de ejecución que gestiona el SO.
- Menor secuencia de instrucciones que se pueden ejecutar en paralelo o concurrentemente.

Proceso o process:

- Mayor unidad de ejecución que gestiona el SO.
- Un proceso consta de uno o varios thread.

Thread vs procesos.

- Proceso: comprende el código del programa y todo lo que hace falta para su ejecución:
    - Datos en pila, segmentos (Variables globales y estáticas) y en heap.
    - Contenido de los registros.
    - Tabla de páginas.
    - Tabla de ficheros abiertos.

- Para comunicar procesos hay que usar llamadas al SO.
- Un proceso puede constar de múltiples flujos de control, llamados threads o procesos ligeros. Cada thread tiene:
    - Su propia pila.
    - Contenido de los registros, en particular el contador de programa o puntero de pila.
- Para comunicar threads de un proceso se usa la memoria que comparten.
- Menor granularidad para threads:
    - Creación de threads en menor tiempo.
    - Destrucción de threads en menor tiempo.
    - Conmutación de threads en menor tiempo.
    - Comunicación en menor tiempo.

Detección, utilización, implementación y extracción del paralelismo.

En los procesadores ILP superescalares o segmentados la arquitectura extrae el paralelismo. Para ello eliminan dependencias de datos falsas entre instrucciones y evitan problemas debidos a dependencias de datos, de control y de recursos. En estos procesadores, la arquitectura extrae paralelismo implícito en las entradas en tiempo de ejecución.

El grado de paralelismo de las instrucciones aprovechado se puede incrementar con ayuda del compilador y del programador. Podemos definir el grado de paralelismo de un conjunto de entradas a un sistema, como el máximo número de entradas del conjunto que se puede ejecutar en paralelo. En las arquitecturas ILP VLIW el paralelismo está ya explícito en las entradas. Las instrucciones que se van a ejecutar en paralelo se captan juntas de memoria. El compilador es el principal responsable de extraer paralelismo para arquitecturas VLIW.

Hay compiladores que extraen el paralelismo de datos implícito a nivel de bucle. Algunos compiladores lo hacen explícito a nivel de hebra, y otros dentro de una instrucción para que se pueda aprovechar en arquitecturas SIMD o vectoriales. Aún es difícil para un compilador extraer paralelismo a nivel de función sin ayuda del programador.

La distribución de tareas independientes entre hebras o entre procesos dependerán: de la granularidad de las unidades de código independientes, de la posibilidad que ofrezca la herramienta para programación paralela, de la arquitectura disponible y del SO disponible.