Arquitectura de Computadores. Tema 1

Lección 1. Clasificación del paralelismo implícito en una aplicación.

Clasificación del paralelismo

Según su naturaleza, podemos clasificar el paralelismo implícito en tres categorías.

Nivel de paralelismo implícito

El paralelismo puede darse a diferentes niveles: a nivel de programa, a nivel de funciones, a nivel de bloques, a nivel de operaciones, etc.

Granularidad

Es el volumen de trabajo, y será más fina o más gruesa si el volumen de trabajo es menor o mayor. Podemos decir que es el tamaño del conjunto de operaciones que pueden realizarse en paralelo.

Paralelismo de tareas vs paralelismo de datos

No es lo mismo trabajar en paralelo únicamente con datos, únicamente con tareas, o mezclando ambos.

El paralelismo de tareas se encuentra extrayendo la estructura lógica de funciones. Es por esto que está relacionado con el paralelismo *a nivel de función*. El paralelismo de datos se encuentra implícito en las operaciones con estructuras de datos, y relacionado con el paralelismo a nivel del *bucle*.

Condiciones de paralelismo

Para poder realizar ejecuciones en paralelo entre dos bloques de código, deben presentar una independencia de datos.

Dependencia de datos

Decimos que un bloque de código B_2 presenta dependencia de datos con respecto a otro bloque B_1 si:

- Hacen referencia a una misma posición de memoria `M` (variable).
- B_1 aparece en la secuencia de código antes que B_2.

Podemos clasificar la dependencia de datos en tres tipos, según la secuencia de operaciones de lectura/escritura que ocurran en los dos bloques con respecto a `M`.

RAW (Read After Write): dependencia verdadera. No se puede evitar.
WAW (Write After Write): dependencia de salida. Podría evitarse guardando la variable inicial en una variable temporal.
WAR (Write After Read): antidependencia. Podría evitarse realizando la lectura en una variable temporal que contuviese la variable inicial.

El caso *RAR (Read After Read)* no presenta dependencia, ya que simplemente se consulta el dato, sin modificarlo en ningún caso.

Paralelismo explícito

Un computador puede aprovechar el paralelismo entre diferentes entidades de forma explícita, a saber:

- Instrucciones.
- Hebras (*threads*).
- Procesos.

Hebras vs procesos

Un **proceso** comprende el código del programa, y todo lo necesario para su ejecución (datos en pila, segmentos, registros, tabla de páginas, ...). Para comunicar procesos hay que usar llamadas al SO.

Un proceso puede constar de múltibles **hebras** que controlan el flujo de control. Cada hebra tiene su propia pila, y también el contenido de los registros. Para comunicarse entre ellos, utilizan la memoria que comparten.

En general, las operaciones de creación, destrucción, conmutación y comunicación en las hebras consumen menos tiempo.

Lección 2. Clasificación de estructuras paralelas

Computación paralela y computación distribuida

La **computación paralela** estudia el desarollo y ejecución de aplicaciones en un sistema compuesto por múltiples cores/procesadores que es visto externamente como una unidad autónoma.

Mientras que la **computación distribuida** estudia los aspectos del desarollo y ejecución de aplicaciones en un sistema distribuido, es decir, en una colección de recursos autónomos situados en distintas localizaciones físicas.

Esta última puede ser a **baja escala**, si se encuentran situados en diversas localizaciones unidas por una misma infraestructura de red local. O bien **computación grid**, un conjunto de dominios administrativos distribuidos en distintas localidades geográficas y que están unidas con una infraestructura de telecomunicaciones.

Ejemplos de esto son la *computación cloud*, que se desarrollan en un *sistema cloud*, un conjunto de recursos virtuales (pay-per-use y con interfaz de auto-servicio) que abstraen los recursos físicos utilizados, con recursos que son captados/liberados de manera inmediata, con acceso múltiple de clientes y con métodos de conexión estándar independientes de la plataforma de acceso.

Criterios de clasificación de computadores

Por lo general la clasificación de computadores se basa en el número de núcleos que posean y operaciones que sean capaces de hacer, lo cual influye directamente en el precio.

Clasificación de Flynn

Se basa en el flujo de instrucciones, si son capaces de trabajar con una única instrucción o con varias y si son capaces de trabajar con un único flujo de datos o con varios.

SISD

Single Instruction, Single Data. Existe un único flujo de datos y una única instrucción a ejecutar en cada instante **<!- (algo de único flujo de datos y una única instrucción, completar) –>**

SIMD

Single Instruction, Multiple Data. La instrucción que se codifica va a cada uno de los procesadores (es la misma para todos), donde se le indica de dónde tiene que captar los datos, hace tantas operaciones como unidades de procesamiento se tengan.

MISD

Multiple Instruction, Single Data. No es un sistema que se implemente en la realidad, cada unidad de control estaría conectada con una única unidad de procesamiento con cada flujo de datos.

MIMD

Multiple Instruction, Multiple Data. Múltiples flujos de datos que permiten realizar múltiples **<!- (algo de múltiples flujos de datos y múltiples instrucciones, completar) –>**

Sistema de memoria

Existen dos tipos de máquinas según esta clasificación: multiprocesadores y multicomputadores. En los multiprocesadores todos los procesadores comparten el mismo espacio de direcciones. El programador no necesita conocer dónde están almacenados los datos. En los multicomputadores cada procesador tiene su propio espacio de direcciones. Están compuestos por computadores completos conectados entre sí por una interfaz de red.

<!- (Se comienza la leccion 3 pero falta acabar la 2) –>

Lección 3. Evaluación de prestaciones de una arquitectura

Tiempos de CPU

Tiempo de CPU

Tcpu = Ciclos*Tciclo = Ciclos/Frecuencia de reloj

Ciclos por Instrucción

CPI = Ciclos/NI

Donde NI es el nº de instrucciones y donde los ciclos son la suma de los CPI de cada instruccion. Así vemos que:

Tcpu = NI * CPI * Tciclo

Tcpu = NI * (CPE / IPE) * Tciclo

CPE: Número mínimo de ciclos transcurridos entre los instantes instantes en que el procesador procesador puede emitir instrucciones instrucciones

IPE: Instrucciones que pueden emitirse (para empezar su ejecución) cada vez que se produce dicha emisión