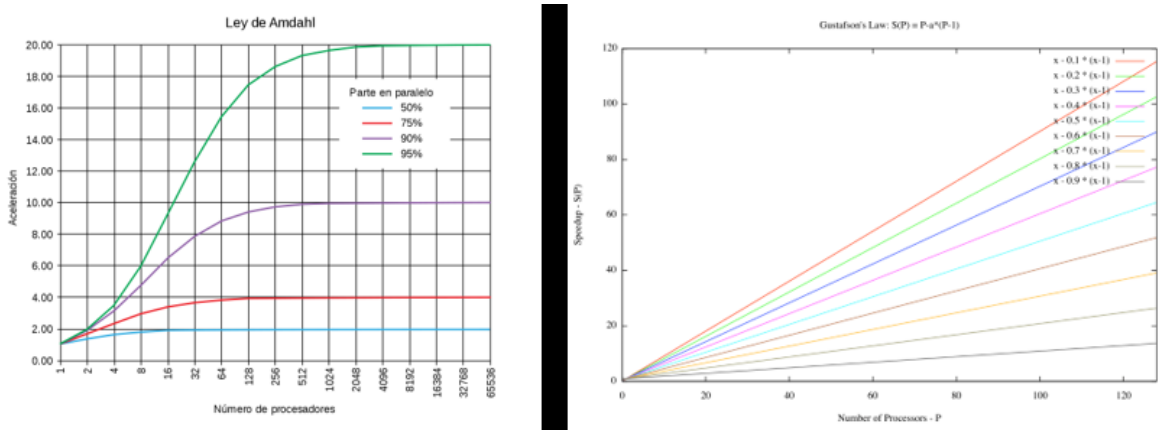
Unidad 4 Procesamiento Paralelo.

4.1 Aspectos Básicos de la computación paralela

La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo). Hay varias formas diferentes de computación paralela: paralelismo a nivel de bit, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tareas. El paralelismo se ha empleado durante muchos años, sobre todo en la computación de altas prestaciones, pero el interés en ella ha crecido últimamente debido a las limitaciones físicas que impiden el aumento de la frecuencia. Como el consumo de energía y por consiguiente la generación de calor de las computadoras constituye una preocupación en los últimos años, la computación en paralelo se ha convertido en el paradigma dominante en la arquitectura de computadores, principalmente en forma de procesadores multinúcleo.



Los programas informáticos paralelos son más difíciles de escribir que los secuenciales, porque la concurrencia introduce nuevos tipos de errores de software, siendo las condiciones de carrera los más comunes. La comunicación y sincronización entre diferentes subtareas son algunos de los mayores obstáculos para obtener un buen rendimiento del programa paralelo. La máxima aceleración posible de un programa como resultado de la paralelización se conoce como la ley de Amdahl.

Ley de Amdahl y ley de Gustafson

Idealmente, la aceleración a partir de la paralelización es lineal, doblar el número de elementos de procesamiento debe reducir a la mitad el tiempo de ejecución y doblarlo por segunda vez debe nuevamente reducir el tiempo a la mitad. Sin embargo, muy pocos algoritmos paralelos logran una aceleración óptima. La mayoría tienen una aceleración casi lineal para un pequeño número de elementos de procesamiento, y pasa a ser constante para un gran número de elementos de procesamiento.

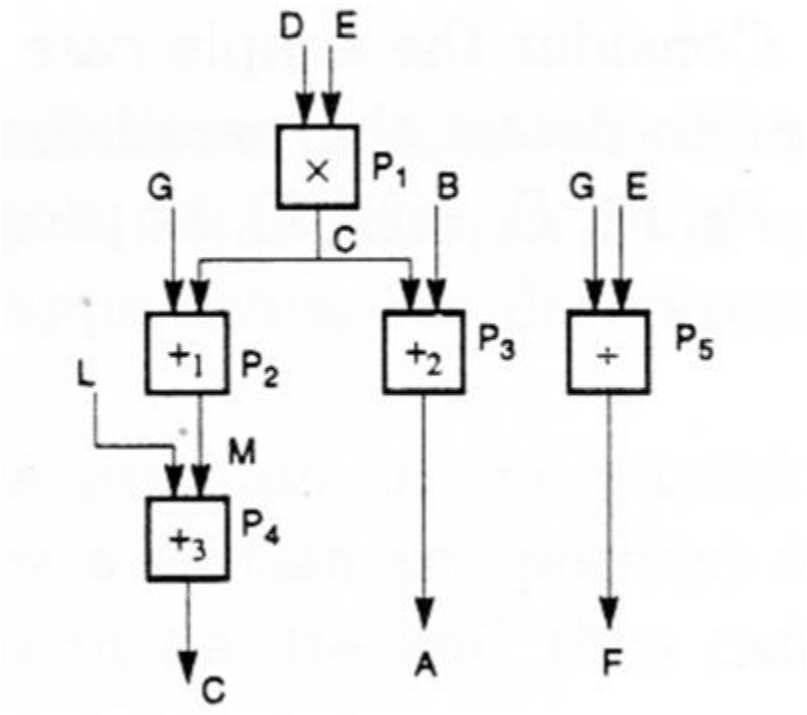
La aceleración potencial de un algoritmo en una plataforma de cómputo en paralelo está dada por la ley de Amdahl, formulada originalmente por Gene Amdahl en la década de 1960. Esta señala que una pequeña porción del programa que no pueda paralelizarse va a limitar la aceleración que se logra con la paralelización. Los programas que resuelven problemas matemáticos o ingenieriles típicamente consisten en varias partes paralelizables y varias no paralelizables (secuenciales).

La ley de Gustafson es otra ley en computación que está en estrecha relación con la ley de Amdahl. Ambas leyes asumen que el tiempo de funcionamiento de la parte secuencial del programa es independiente del número de procesadores. La ley de Amdahl supone que todo el problema es de tamaño fijo, por lo que la cantidad total de trabajo que se hará en paralelo también es independiente del número de procesadores, mientras que la ley de Gustafson supone que la cantidad total de trabajo que se hará en paralelo varía linealmente con el número de procesadores.

Dependencias.

Entender la dependencia de datos es fundamental en la implementación de algoritmos paralelos. Ningún programa puede ejecutar más rápidamente que la cadena más larga de cálculos dependientes (conocida como la ruta crítica), ya que los cálculos que dependen de cálculos previos en la cadena deben ejecutarse en orden. Sin embargo, la mayoría de los algoritmos no consisten sólo de una larga cadena de cálculos dependientes; generalmente hay oportunidades para ejecutar cálculos independientes en paralelo.

Sea P_i y P_j dos segmentos del programa. Las condiciones de Bernstein describen cuando los dos segmentos son independientes y pueden ejecutarse en paralelo. Para P_i, sean I_i todas las variables de entrada y O_i las variables de salida, y del mismo modo para P_j. P_i y P_j son independientes si satisfacen.



Ejecución paralela.

Una violación de la primera condición introduce una dependencia de flujo, correspondiente al primer segmento que produce un resultado utilizado por el segundo segmento. La segunda condición representa una anti-dependencia, cuando el segundo segmento (P_j) produce una variable que necesita el primer segmento (P_i). La tercera y última condición representa una dependencia de salida: Cuando dos segmentos escriben en el mismo lugar, el resultado viene del último segmento ejecutado.

Condiciones de carrera, exclusión mutua, sincronización, y desaceleración paralela.

Las subtareas en un programa paralelo a menudo son llamadas hilos. Algunas arquitecturas de computación paralela utilizan versiones más pequeñas y ligeras de hilos conocidas como hebras, mientras que otros utilizan versiones más grandes conocidos como procesos. Sin embargo, «hilos» es generalmente aceptado como un término genérico para las subtareas. Los hilos a menudo tendrán que actualizar algunas variables que se comparten entre ellos. Las instrucciones entre los dos programas pueden entrelazarse en cualquier orden.

Las aplicaciones a menudo se clasifican según la frecuencia con que sus subtareas se sincronizan o comunican entre sí. Una aplicación muestra un paralelismo de grano fino si sus subtareas deben comunicase muchas veces por segundo, se considera paralelismo de grano grueso si no se comunican muchas veces por segundo, y es vergonzosamente paralelo si nunca o casi nunca se tienen que comunicar.

Aplicaciones vergonzosamente paralelas son consideradas las más fáciles de paralelizar.

Grano de paralelismo.

Muy grueso: Programas.

Grueso: Subprogramas, tareas.

Fino: Instrucción.

Muy fino: Fases de instrucción.

Modelos de consistencia.

Los lenguajes de programación en paralelo y computadoras paralelas deben tener un modelo de consistencia de datos también conocido como un modelo de memoria.

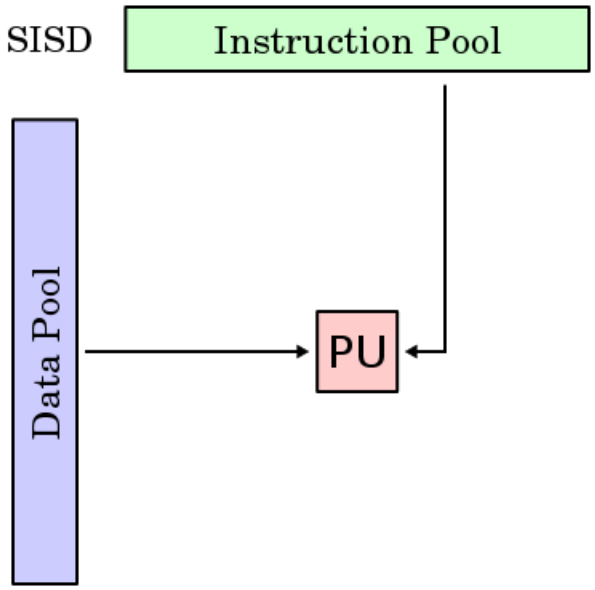
El modelo de consistencia define reglas para las operaciones en la memoria del ordenador y cómo se producen los resultados.

Uno de los primeros modelos de consistencia fue el modelo de consistencia secuencial de Leslie Lamport. La consistencia secuencial es la propiedad de un programa en la que su ejecución en paralelo produce los mismos resultados que un programa secuencial. Específicamente, es un programa secuencial consistente si "... los resultados de una ejecución son los mismos que se obtienen si las operaciones de todos los procesadores son ejecutadas en un orden secuencial, y las operaciones de cada procesador individual aparecen en esta secuencia en el orden especificado por el programa".

T**axonomía de Flynn.**

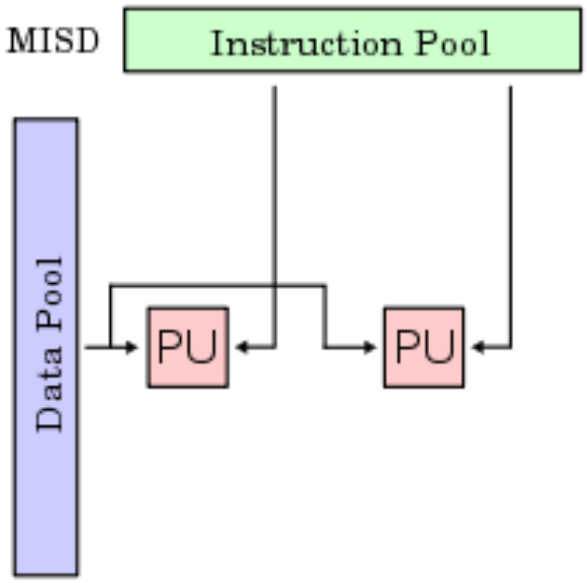
Single Instruction, Single Data (SISD).

Hay un elemento de procesamiento, que tiene acceso a un único programa y a un almacenamiento de datos. En cada paso, el elemento de procesamiento carga una instrucción y la información correspondiente y ejecuta esta instrucción. El resultado es guardado de vuelta en el almacenamiento de datos. Luego SISD es el computador secuencial convencional, de acuerdo al modelo de von Neumann.



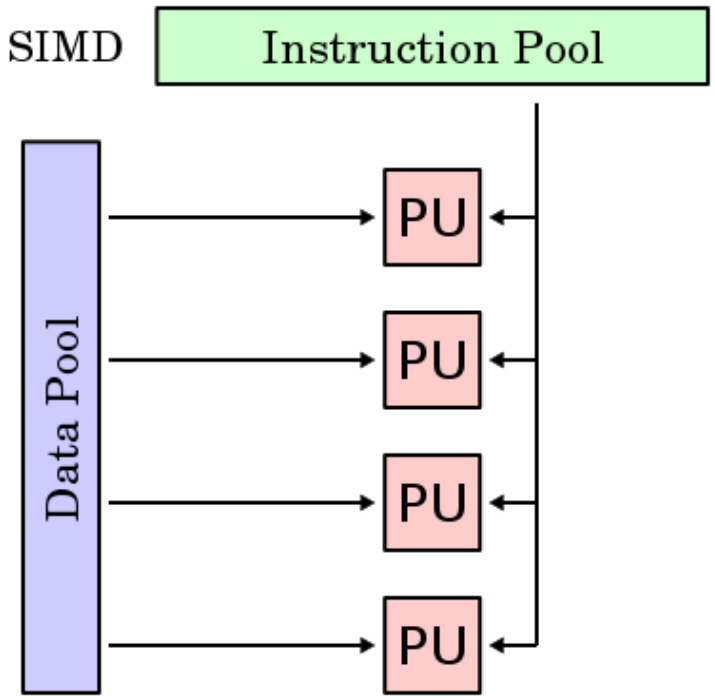
Multiple Instruction, Single Data (MISD).

Hay múltiples elementos de procesamiento, en el que cada cual tiene memoria privada del programa, pero se tiene acceso común a una memoria global de información. En cada paso, cada elemento de procesamiento de obtiene la misma información de la memoria y carga una instrucción de la memoria privada del programa. Luego, las instrucciones posiblemente diferentes de cada unidad, son ejecutadas en paralelo, usando la información (idéntica) recibida anteriormente. Este modelo es muy restrictivo y no se ha usado en ningún computador de tipo comercial.



Single Instruction, Multiple Data (SIMD).

Hay múltiples elementos de procesamiento, en el que cada cual tiene acceso privado a la memoria de información (compartida o distribuida). Sin embargo, hay una sola memoria de programa, desde la cual una unidad de procesamiento especial obtiene y despacha instrucciones. En cada paso, cada unidad de procesamiento obtiene la misma instrucción y carga desde su memoria privada un elemento de información y ejecuta esta instrucción en dicho elemento. Entonces, la instrucción es síncronamente aplicada en paralelo por todos los elementos de proceso a diferentes elementos de información. Para aplicaciones con un grado significante de paralelismo de información, este acercamiento puede ser muy eficiente. Ejemplos pueden ser aplicaciones multimedia y algoritmos de gráficos de computadora.



Multiple Instruction, Multiple Data (MIMD).

Hay múltiples unidades de procesamiento, en la cual cada una tiene tanto instrucciones como información separada. Cada elemento ejecuta una instrucción distinta en un elemento de información distinto. Los elementos de proceso trabajan asíncronamente. Los clusters son ejemplo son ejemplos del modelo MIMD.