INSTITUTO TECNOLÓGICO DE CIUDAD MADERO.



UNIDAD 1

PRACTICA 10

ALUMNA: CAVAZOS ARGOT ANA VICTORIA

N° CONTROL: 15071292

PROFESOR: DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN

MATERIA: PROGRAMACIÓN PARALELA

FECHA DE ENTREGA: 16 DE SEPTIEMBRE 2018

Índice:

[Introducción: 4](#_Toc524908171)

[Computación en serie: 4](#_Toc524908172)

[Computación paralela: 4](#_Toc524908173)

[Computadora paralela: 4](#_Toc524908174)

[Clasificación de computadoras paralelas: 4](#_Toc524908175)

[Taxonomía clásica de Flynn: 4](#_Toc524908176)

[Instrucción única, datos únicos (SISD): 5](#_Toc524908177)

[Instrucción única, datos múltiples (SIMD): 5](#_Toc524908178)

[Instrucción múltiple, datos únicos (MISD): 5](#_Toc524908179)

[Instrucción múltiple, datos múltiples (MIMD): 5](#_Toc524908180)

[Supercomputación / Computación de alto rendimiento (HPC): 6](#_Toc524908181)

[Nodo: 6](#_Toc524908182)

[Tarea (Task): 6](#_Toc524908183)

[Pipelining: 6](#_Toc524908184)

[Memoria compartida: 6](#_Toc524908185)

[Multiprocesador simétrico (SMP): 6](#_Toc524908186)

[Memoria distribuida: 6](#_Toc524908187)

[Comunicaciones: 6](#_Toc524908188)

[Sincronización: 7](#_Toc524908189)

[Granularidad: 7](#_Toc524908190)

[Observed Speedup: 7](#_Toc524908191)

[Parallel Overhead: 7](#_Toc524908192)

[Massively Parallel: 7](#_Toc524908193)

[Embarrassingly Parallel: 7](#_Toc524908194)

[Escalabilidad: 8](#_Toc524908195)

[Límites y costos de programación en paralelo: 8](#_Toc524908196)

[Ley de Amdahl: 8](#_Toc524908197)

[Complejidad: 8](#_Toc524908198)

[Portabilidad: 8](#_Toc524908199)

[Requerimientos de recursos: 8](#_Toc524908200)

[Memoria compartida: 9](#_Toc524908201)

[Memoria distribuida: 9](#_Toc524908202)

[Memoria Híbrida: 10](#_Toc524908203)

Introducción:

Ejercicio 2. Construya un glosario de términos con las secciones: Conceptos y Terminología y arquitecturas de memoria de las computadoras compartidas. Se hará una participación escrita.

Computación en serie:

Tradicionalmente, el software se ha escrito para el cálculo en serie:

* Un problema se divide en una serie discreta de instrucciones.
* Las instrucciones se ejecutan secuencialmente una tras otra.
* Ejecutado en un solo procesador.

Computación paralela:

En el sentido más simple, la computación paralela es el uso simultáneo de múltiples recursos informáticos para resolver un problema de cómputo:

* Un problema se divide en partes discretas que se pueden resolver al mismo tiempo
* Cada parte se divide en una serie de instrucciones
* Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
* Se emplea un mecanismo general de control / coordinación

Computadora paralela:

Prácticamente todas las computadoras autónomas de hoy en día son paralelas desde una perspectiva de hardware.

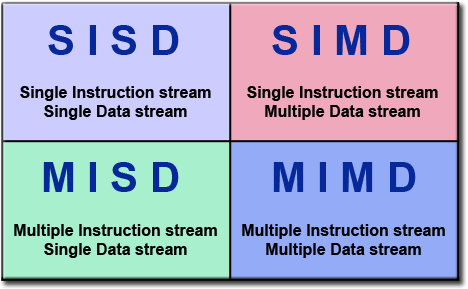
Clasificación de computadoras paralelas:

1) Clasificación basada en la instrucción y las secuencias de datos

2) Clasificación basada en la estructura de las computadoras

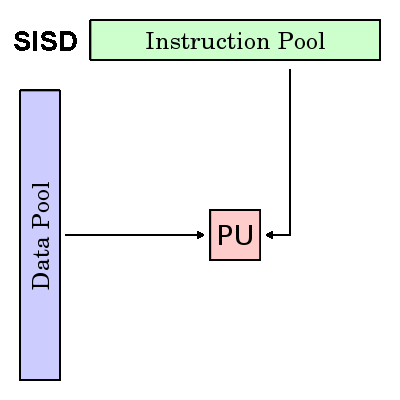
3) Clasificación basada en cómo se accede a la memoria

4) Clasificación basada en el tamaño de grano



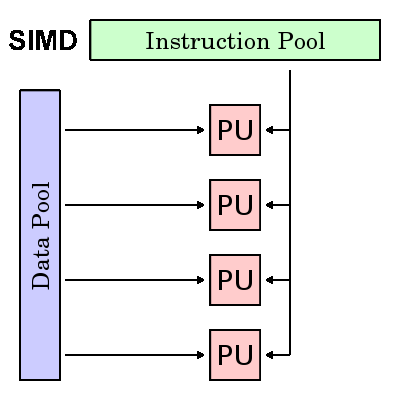
Taxonomía clásica de Flynn:

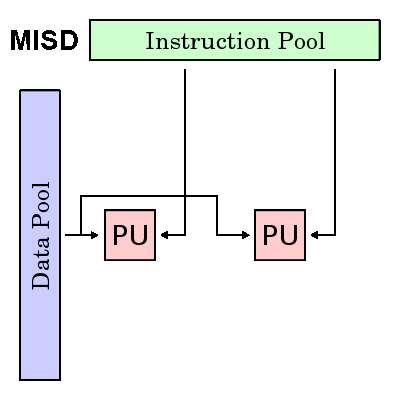
La taxonomía de Flynn distingue las arquitecturas informáticas de varios procesadores de acuerdo con la forma en que pueden clasificarse según las dos dimensiones independientes de **Instruction Stream** y **Data Stream**. Cada una de estas dimensiones puede tener solo uno de dos estados posibles: simple o múltiple.

Instrucción única, datos únicos (SISD):

* Una computadora serial (no paralela)
* Instrucción única: la CPU solo actúa sobre un flujo de instrucciones durante cualquier ciclo de reloj
* Datos únicos: solo se usa una transmisión de datos como entrada durante cualquier ciclo de reloj
* Ejecución determinista

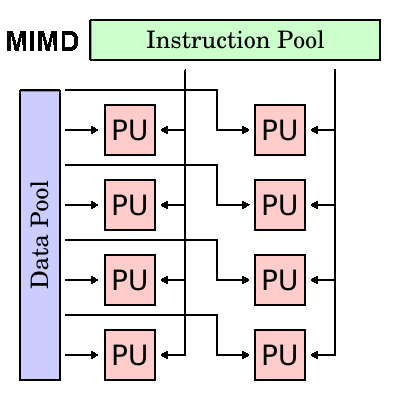
Instrucción única, datos múltiples (SIMD):

* Un tipo de computadora paralela
* Instrucción única: todas las unidades de procesamiento ejecutan la misma instrucción en cualquier ciclo de reloj dado
* Múltiples datos: cada unidad de procesamiento puede operar en un elemento de datos diferente
* Ideal para problemas especializados caracterizados por un alto grado de regularidad, como el procesamiento de gráficos / imágenes.
* Sincrónico (bloqueo) y ejecución determinista
* Dos variedades: Matrices de procesador y Tuberías de vector

Instrucción múltiple, datos únicos (MISD):

* Un tipo de computadora paralela
* Instrucción múltiple: cada unidad de procesamiento opera en los datos de manera independiente a través de secuencias de instrucciones separadas.
* Datos únicos: una secuencia de datos única se alimenta a múltiples unidades de procesamiento.

Instrucción múltiple, datos múltiples (MIMD):

* Un tipo de computadora paralela
* Instrucción múltiple: cada procesador puede estar ejecutando una secuencia de instrucciones diferente
* Múltiples datos: todos los procesadores pueden estar trabajando con una secuencia de datos diferente
* La ejecución puede ser sincrónica o asincrónica, determinista o no determinista
* Actualmente, el tipo más común de computadora paralela: la mayoría de los supercomputadores modernos entran en esta categoría.

Supercomputación / Computación de alto rendimiento (HPC):

Usar las computadoras más rápidas y grandes del mundo para resolver problemas grandes.

Nodo:

Una "computadora en una caja" independiente. Por lo general, se compone de múltiples CPU / procesadores / núcleos, memoria, interfaces de red, etc. Los nodos se conectan en red para formar una supercomputadora.

Tarea (Task):

Una sección lógicamente discreta de trabajo computacional. Una tarea suele ser un programa o un conjunto de instrucciones similar a un programa que ejecuta un procesador. Un programa paralelo consiste en múltiples tareas que se ejecutan en múltiples procesadores.

Pipelining:

Romper una tarea en pasos realizados por diferentes unidades de procesador, con entradas que fluyen a través, al igual que una línea de montaje; un tipo de computación paralela.

Memoria compartida:

Desde un punto de vista estrictamente de hardware, describe una arquitectura de computadora donde todos los procesadores tienen acceso directo (generalmente basado en el bus) a la memoria física común.

En un sentido de programación, describe un modelo donde todas las tareas paralelas tienen la misma "imagen" de memoria y pueden direccionar y acceder directamente a las mismas ubicaciones de memoria lógica, independientemente de dónde realmente exista la memoria física.

Multiprocesador simétrico (SMP):

Arquitectura de hardware de memoria compartida donde múltiples procesadores comparten un solo espacio de direcciones y tienen acceso igual a todos los recursos.

Memoria distribuida:

En hardware, se refiere al acceso a memoria basado en red para memoria física que no es común. Como modelo de programación, las tareas solo pueden "ver" lógicamente la memoria de la máquina local y deben usar las comunicaciones para acceder a la memoria en otras máquinas donde se ejecutan otras tareas.

Comunicaciones:

Las tareas paralelas generalmente necesitan intercambiar datos. Hay varias maneras de lograr esto, como a través de un bus de memoria compartida o en una red, sin embargo, el evento real de intercambio de datos se conoce comúnmente como comunicaciones, independientemente del método empleado.

Sincronización:

La coordinación de tareas paralelas en tiempo real, muy a menudo asociadas con las comunicaciones. A menudo se implementa estableciendo un punto de sincronización dentro de una aplicación donde una tarea puede no avanzar más hasta que otra o más tareas lleguen al mismo punto o al punto lógicamente equivalente.

La sincronización generalmente implica esperar al menos una tarea y, por lo tanto, puede aumentar el tiempo de ejecución de un reloj de pared de una aplicación paralela.

Granularidad:

En la computación paralela, la granularidad es una medida cualitativa de la relación entre cálculo y comunicación.

* Grueso: se realizan cantidades relativamente grandes de trabajo computacional entre eventos de comunicación
* Fino: cantidades relativamente pequeñas de trabajo computacional se realizan entre eventos de comunicación

Observed Speedup:

Aceleración observada de un código que se ha paralelizado, definido como:

|  |
| --- |
| tiempo de reloj de pared de la ejecución en serie  -----------------------------------  tiempo de reloj de pared de ejecución paralela |

Uno de los indicadores más simples y ampliamente utilizados para el rendimiento de un programa paralelo.

Parallel Overhead:

La cantidad de tiempo requerida para coordinar tareas paralelas, en lugar de hacer un trabajo útil. La sobrecarga en paralelo puede incluir factores tales como:

* Tiempo de puesta en marcha de la tarea
* Sincronizaciones
* Transmisión de datos
* Sobrecarga de software impuesta por lenguajes paralelos, bibliotecas, sistema operativo, etc.
* Hora de finalización de la tarea

Massively Parallel:

Se refiere al hardware que comprende un sistema paralelo dado, que tiene muchos elementos de procesamiento. El significado de "muchos" sigue aumentando, pero actualmente, las computadoras paralelas más grandes se componen de elementos de procesamiento que se encuentran en cientos o miles o en millones.

Embarrassingly Parallel:

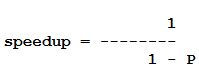
Resolviendo muchas tareas similares, pero independientes simultáneamente; poca o ninguna necesidad de coordinación entre las tareas.

Escalabilidad:

Se refiere a la capacidad de un sistema paralelo (hardware y / o software) para demostrar un aumento proporcional en la aceleración paralela con la adición de más recursos. Los factores que contribuyen a la escalabilidad incluyen:

* Hardware, en particular anchos de banda de memoria-cpu y propiedades de comunicación de red
* Algoritmo de aplicación
* Encabezado paralelo relacionado
* Características de su aplicación específica

Límites y costos de programación en paralelo:

Ley de Amdahl:

La Ley de Amdahl establece que la aceleración potencial del programa se define por la fracción de código (P) que se puede paralelizar:

* Si ninguno de los códigos se puede paralelizar, P = 0 y la aceleración = 1 (sin aceleración).
* Si todo el código está paralelizado, P = 1 y la aceleración es infinita (en teoría).
* Si el 50% del código se puede paralelizar, la velocidad máxima = 2, lo que significa que el código se ejecutará dos veces más rápido.

Complejidad:

En general, las aplicaciones paralelas son mucho más complejas que las aplicaciones serie correspondientes, tal vez un orden de magnitud. No solo tiene varias secuencias de instrucciones ejecutándose al mismo tiempo, sino que también tiene datos fluyendo entre ellas.

Portabilidad:

Gracias a la estandarización en varias API, como MPI, hilos POSIX y OpenMP, los problemas de portabilidad con programas paralelos no son tan graves como en años anteriores.

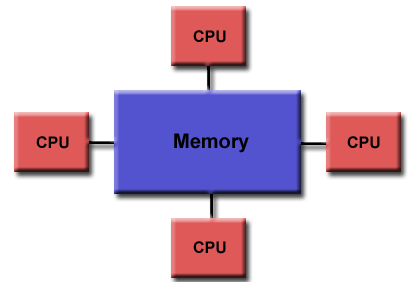
Los sistemas operativos pueden desempeñar un papel clave en los problemas de portabilidad de códigos.

Requerimientos de recursos:

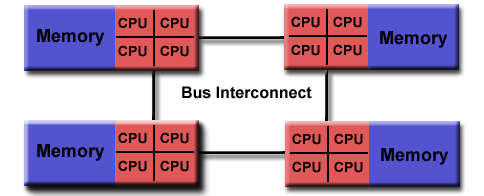
La intención principal de la programación paralela es disminuir el tiempo de ejecución del reloj de pared, sin embargo para lograr esto, se requiere más tiempo de CPU. Por ejemplo, un código paralelo que se ejecuta en 1 hora en 8 procesadores en realidad usa 8 horas de tiempo de CPU.

La cantidad de memoria requerida puede ser mayor para los códigos paralelos que para los códigos serie, debido a la necesidad de replicar datos y para los gastos generales asociados con las bibliotecas y subsistemas de soporte paralelo.

Memoria compartida:

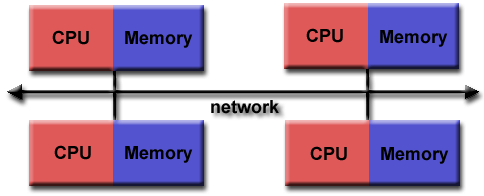
Acceso uniforme a la memoria (UMA):

* Más comúnmente representado hoy en día por las máquinas Symmetric Multiprocessor (SMP)
* Procesadores idénticos
* Igualdad de acceso y acceso a la memoria.
* A veces se llama CC-UMA - UMA coherente de memoria caché. Coherente en caché significa que si un procesador actualiza una ubicación en la memoria compartida, todos los demás procesadores conocen la actualización. La coherencia de la caché se logra a nivel de hardware.

Acceso a memoria no uniforme (NUMA):

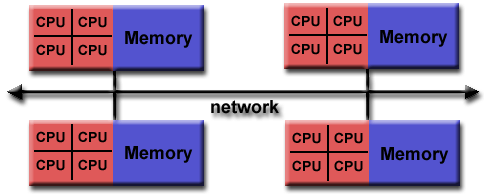
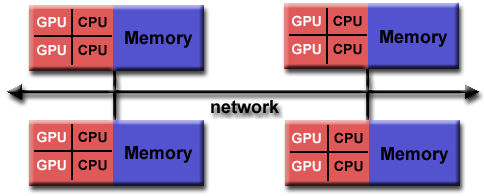
* A menudo se hace vinculando físicamente dos o más SMP
* Un SMP puede acceder directamente a la memoria de otro SMP
* No todos los procesadores tienen el mismo tiempo de acceso a todas las memorias
* El acceso a la memoria a través del enlace es más lento
* Si se mantiene la coherencia de la memoria caché, también se la puede llamar CC-NUMA - Cochuración coherente NUMA.

Memoria distribuida:



* Los sistemas de memoria distribuida requieren una red de comunicación para conectar la memoria entre procesadores.
* Los procesadores tienen su propia memoria local. Las direcciones de memoria en un procesador no se asignan a otro procesador, por lo que no existe un concepto de espacio de direcciones global en todos los procesadores.
* Como cada procesador tiene su propia memoria local, funciona de manera independiente. Los cambios que hace en su memoria local no tienen ningún efecto en la memoria de otros procesadores. Por lo tanto, el concepto de coherencia de caché no se aplica.
* Cuando un procesador necesita acceso a datos en otro procesador, generalmente es tarea del programador definir explícitamente cómo y cuándo se comunican los datos. La sincronización entre tareas también es responsabilidad del programador.
* El "tejido" de red utilizado para la transferencia de datos varía ampliamente, aunque puede ser tan simple como Ethernet.

Memoria Híbrida:

* Las computadoras más grandes y rápidas en el mundo de hoy emplean arquitecturas de memoria compartida y distribuida.
* El componente de memoria compartida puede ser una máquina de memoria compartida y / o unidades de procesamiento de gráficos (GPU).
* El componente de memoria distribuida es la conexión en red de varias máquinas de memoria compartida / GPU, que solo conocen su propia memoria, no la memoria de otra máquina. Por lo tanto, se requieren comunicaciones de red para mover datos de una máquina a otra.

Modelos de programación en paralelo:

Modelos de programación paralela de uso común:

* Memoria compartida (sin hilos)
* Trapos
* Memoria Distribuida / Paso de Mensaje
* Datos paralelos
* Híbrido
* Datos múltiples de programa único (SPMD)
* Programa múltiple de datos múltiples (MPMD)

Bibliografia:

<https://computing.llnl.gov/tutorials/parallel_comp/>

Autor: Blaise Barney, Lawrence Livermore National Laboratory

Titulo: Introduction to Parallel Computing