DR. GABRIEL LEON PAREDES

gleon@ups.edu.ec

www.linkedin.com/in/gabrielleonp

Cloud Computing, Smart Cities & High-Performance Computing

Cuenca, Ecuador

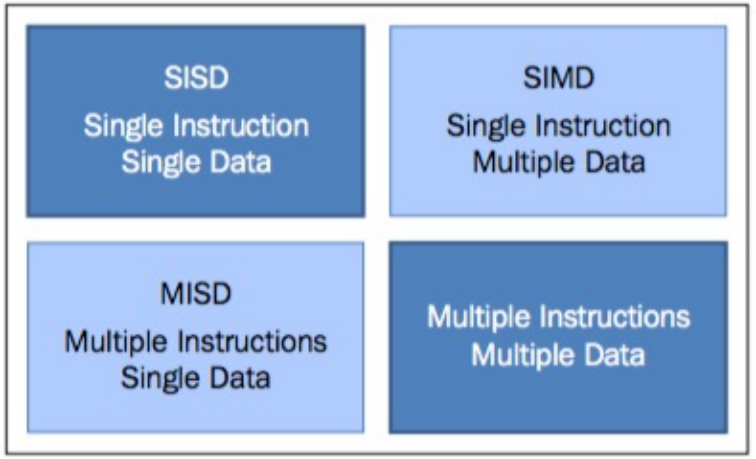
COMPUTO PARALELO

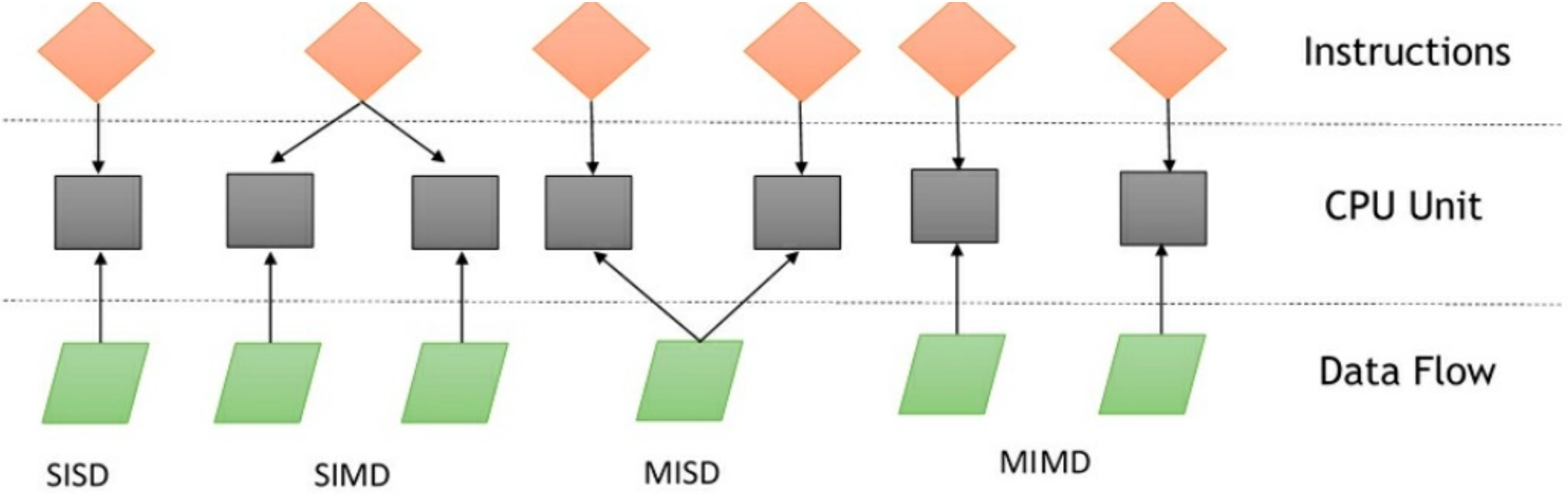
Introducción

- La arquitectura (hardware) para el calculo en paralelo esta estrechamente relacionado con los algoritmos en paralelo. Por lo que, no podemos pensar en la arquitectura sin primero mencionar sobre el software en paralelo que se ejecutara en dicha arquitectura:
  - **Paralelismo a nivel de datos**: donde simultáneamente opera en múltiples datos. Ej. Multiplicación y división de números binarios, operaciones sobre matrices y vectores, etc.
  - **Paralelismo a nivel de instrucción (ILP**): donde simultáneamente un procesador ejecuta mas de una instrucción. Ej. Pipelining
  - **Paralelismo a nivel de hilos (TLP):** Un hilo es una parte de un programa que comparte los recursos del procesador con otros hilos. En TLP, varios hilos son ejecutados simultáneamente sobre uno/varios procesadores.
  - **Paralelismo a nivel de procesos:** Un proceso es un programa que esta ejecutándose en una computadora. Cada proceso tiene sus propios recursos computacionales. Se conoce como multi-tareas donde varios programas son ejecutados simultáneamente en una/varias computadoras.

Taxonomía de Flynn

- La taxonomía de Flynn es un sistema para clasificar arquitecturas informáticas. Se basa en dos conceptos principales:
    - **Flujo de instrucciones**: un sistema con *n* CPUs tiene *n* contadores de programa y, por lo tanto, *n* flujos de instrucciones. Esto corresponde a un contador de programa.
    - **Flujo de datos**: un programa que calcula una función con base a una lista de datos, tiene un flujo de datos. El programa que calcula la misma función en varias listas diferentes de datos tiene más flujos de datos. Esto se compone de un conjunto de operandos.
- Como las instrucciones y los flujos de datos son independientes, existen cuatro categorías de máquinas paralelas:



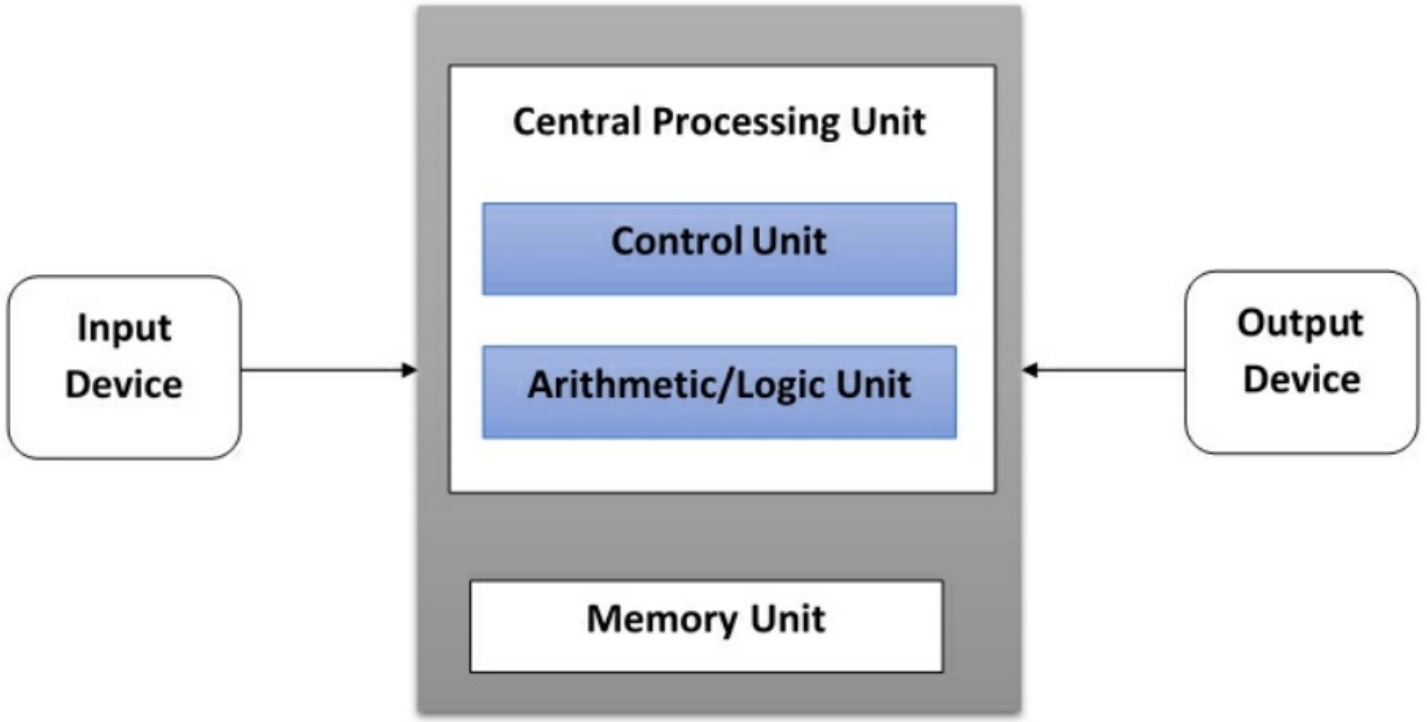


Taxonomía de Flynn

Single Instruction Single Data (SISD)

- El sistema computacional SISD es como la máquina von Neumann, una maquina de un solo procesador.
  - Ejecuta una instrucción que opera sobre un flujo de datos
  - Las instrucciones son procesadas secuencialmente
- En un ciclo de reloj la CPU ejecuta las siguientes operaciones:
  - **Recuperar**: La CPU recupera los datos e instrucciones desde la memoria, que se conoce como registro.
  - **Decodificar**: La CPU decodifica las instrucciones.
  - **Ejecutar**: La instrucción es ejecutada sobre los datos, y el resultado de la operación es almacenado en nuevo registro.
- Una vez que se completa la etapa de ejecución, la CPU se configura para comenzar otro ciclo de CPU:

Etapas de SISD en la CPU



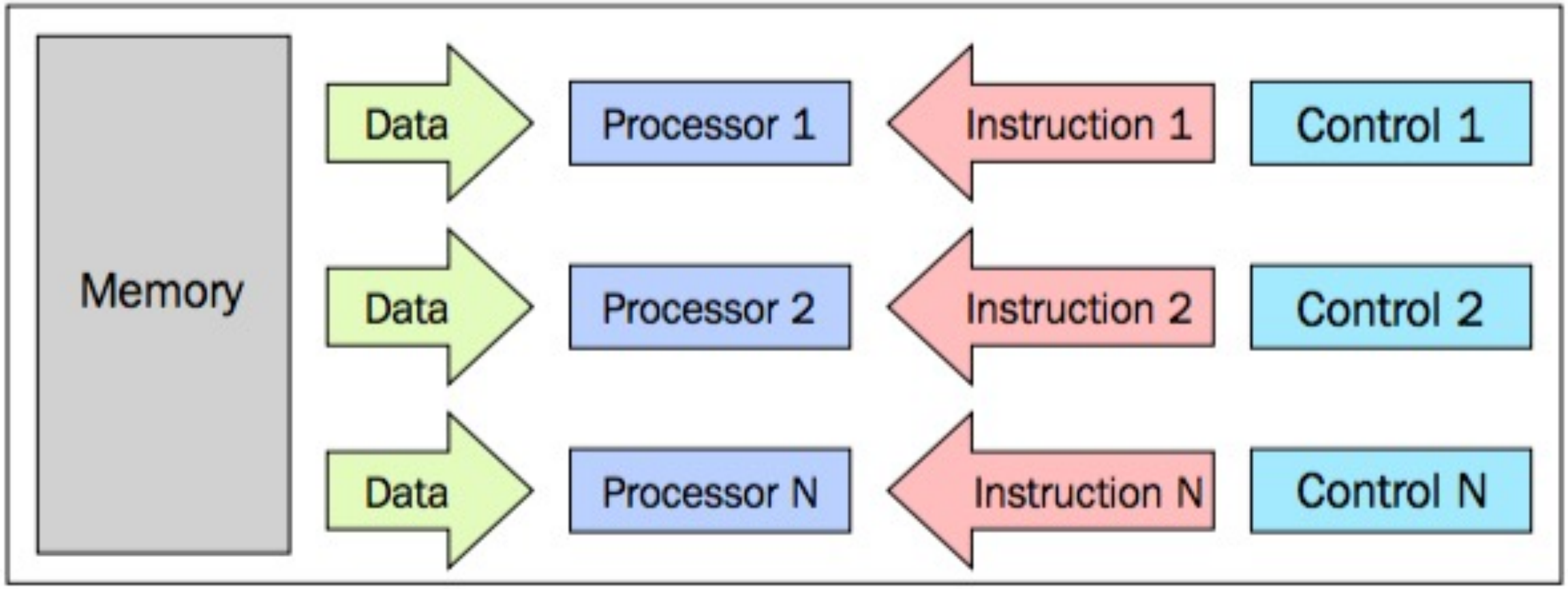
The SISD architecture schema

- Los algoritmos que se ejecutan en este tipo de computadora son secuenciales (o seriales) ya que no contienen ningún paralelismo.
  - Un ejemplo de una computadora SISD es un sistema de hardware con una sola CPU.
- Los elementos principales de estas arquitecturas son los siguientes:
  - **Unidad de memoria central**: se utiliza para almacenar tanto las instrucciones como los datos del programa.
  - **CPU**: se utiliza para obtener las instrucciones y/o datos de la unidad de memoria, que decodifica las instrucciones y las implementa secuencialmente
  - **Sistema de E/S**: se refiere a los datos de entrada y salida del programa.

Multiple Instruction Single Data (MISD)

- El sistema computacional MISD es una maquina de *N* procesadores.
  - Cada procesador tiene su propia unidad de control y comparten una sola unidad de memoria.
  - En cada ciclo de reloj, los datos recibidos desde la memoria son procesados por todos los procesadores simultáneamente, según las instrucciones recibidas por su unidad de control.
  - Se obtiene el paralelismo (a nivel de instrucciones) al ejecutar varias operaciones sobre los mismos datos.
  - Los problemas que puede resolver esta arquitectura son bastante especiales. Por ejemplo, en la encriptación de datos.
- La computadora MISD no ha encontrado espacio en el sector comercial. Por lo que, el uso de las computadoras MISD son más un ejercicio intelectual que práctico.

Arquitectura MISD



Single Instruction Multiple Data (SIMD)

- El sistema computacional SIMD es una maquina de *N* procesadores idénticos.
  - Cada procesador tiene su propia memoria local, donde es posible almacenar datos.
  - Todos los procesadores trabajan bajo el control de un flujo de instrucciones. Además, tienen *n* flujos de datos, uno para cada procesador.
  - Los procesadores trabajan simultáneamente y ejecutan la misma instrucción, pero con diferente datos (valores). Paralelismo a nivel de datos
- La arquitectura SIMD es más versátiles que la arquitectura MISD.
  - Puede resolver varios problemas de diferentes áreas al paralelizar algoritmos en SIMD.
  - Los algoritmos son fáciles de diseñar, analizar e implementar.
- Su limitación es que sólo problemas que puedan dividirse en varios sub-problemas idénticos pueden ser ejecutados bajo la arquitectura SIMD.

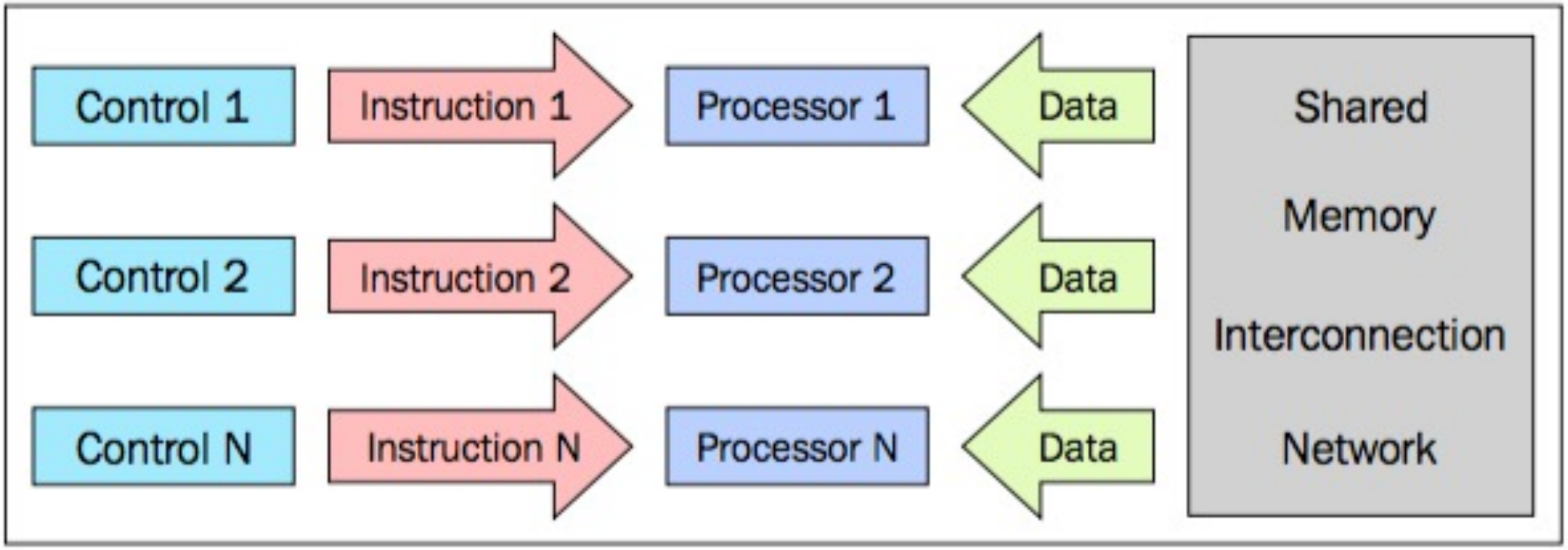
- Las tarjetas gráficas (GPU) utilizan la arquitectura SIMD y han extendido el uso de este paradigma

https://en.wikipedia.org/wiki/Connection_Machine
https://arc.aiaa.org/doi/abs/10.2514/3.19770?journalCode=jgcd

Multiple Instruction Multiple Data (MIMD)

- El sistema computacional MIMD es una maquina de *N* procesadores, *N* flujos de instrucciones y *N* flujos de datos.
  - Es la arquitectura más general y más poderosa dentro del calculo en paralelo de acuerdo a la clasificación de Flynn.
  - Cada procesador tiene su propia memoria local y unidad de control, lo que lo hace más poderosa computacionalmente que la arquitectura SIMD.
  - Cada procesador funciona bajo el control de un flujo de instrucciones emitidas por su propia unidad de control. Por lo que, los procesadores pueden ejecutar diferentes programas/instrucciones con diferentes datos.
  - La arquitectura MIMD se logra con la ayuda del nivel de paralelismo de hilos y/o procesos.
  - Esta arquitectura es utilizada para resolver problemas que no tienen una estructura regular, ya que los procesadores trabajan de manera asíncrona.
- Hoy en día, esta arquitectura se aplica a muchas PC, supercomputadoras y redes de computadoras. Sin embargo, ee debe considerar que los algoritmos asíncronos son difíciles de diseñar, analizar y/o implementar.

Arquitectura MIMD



Otras arquitecturas

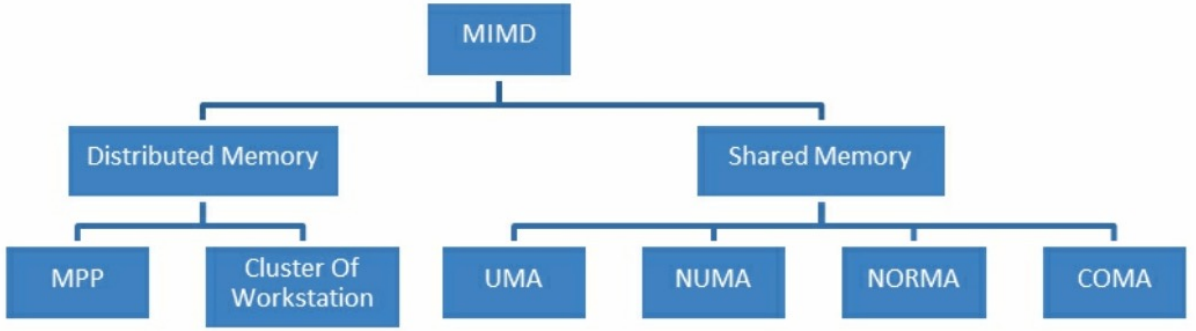
- La taxonomía de Flynn se puede ampliar considerando que las máquinas SIMD se pueden dividir en dos subgrupos:
  - supercomputadoras numéricas
  - máquinas vectoriales
- Por otro lado, MIMD se puede dividir en máquinas que tienen
  - una memoria compartida y
  - una memoria distribuida.

Organización de la memoria

- Otro aspecto que debemos tener en cuenta para evaluar las arquitecturas paralelas, es la organización de la memoria, o más bien, la forma en que se accede a los datos.
- No importa qué tan rápido sea la unidad de procesamiento, si la memoria no puede mantener y proporcionar instrucciones y datos a una velocidad suficiente, no habrá mejora en el rendimiento.
- El principal problema que debemos superar para que el tiempo de respuesta de la memoria sea compatible con la velocidad del procesador es el tiempo del ciclo de memoria, que se define como el tiempo transcurrido entre dos operaciones sucesivas.
- El tiempo de ciclo del procesador suele ser mucho más corto que el tiempo de ciclo de la memoria.

Organización de la memoria en MIMD

- Cuando un procesador inicia una transferencia hacia o desde la memoria, los recursos del procesador permanecerán ocupados durante todo el ciclo de memoria; además, durante este período, ningún otro dispositivo (controlador de E/S, procesador o el procesador que realizó la solicitud) podrá usar la memoria debido a la transferencia en curso.
- Las soluciones al problema del acceso a la memoria han resultado en una dicotomía de arquitecturas MIMD.
  - El primer tipo de sistema, conocido como sistema de memoria compartida, tiene una memoria virtual alta y todos los procesadores tienen el mismo acceso a los datos e instrucciones en esta memoria.
  - El otro tipo de sistema es el modelo de memoria distribuida, en el que cada procesador tiene memoria local que no es accesible para otros procesadores.

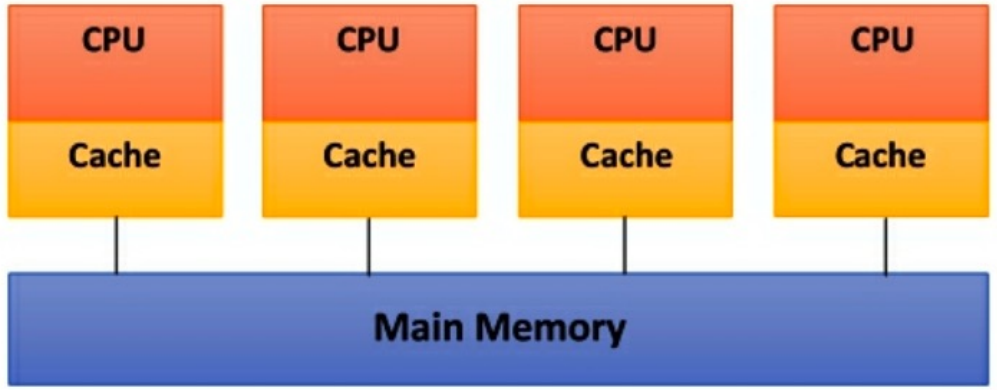


Memory organization in the MIMD architecture

Memoria compartida

- Las conexiones físicas aquí son bastante simples:
- La estructura del bus permite un número arbitrario de dispositivos (CPU + Caché) que comparten el mismo canal (Memoria principal).
- Los protocolos de bus se diseñaron originalmente para permitir que un único procesador y uno o más discos o controladores de cinta se comuniquen a través de la memoria compartida.

Cada procesador se ha asociado con la memoria caché, ya que se supone que la probabilidad de que un procesador necesite tener datos o instrucciones presentes en la memoria local es muy alta.



Shared memory architecture schema

Coherencia de la memoria caché

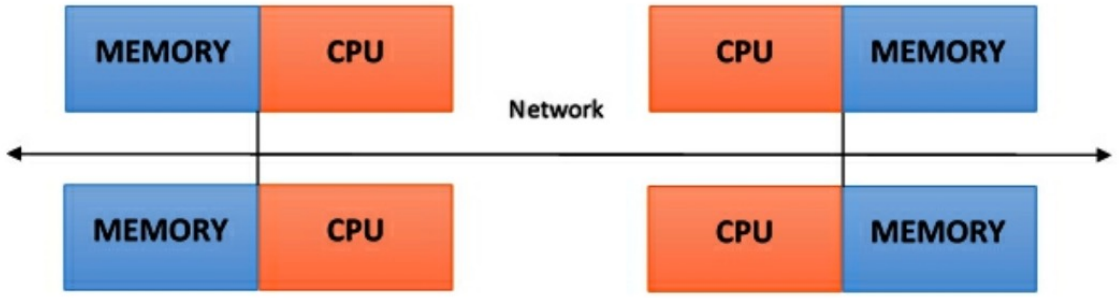
- El problema ocurre cuando un procesador modifica los datos almacenados en el sistema de memoria que otros procesadores usan simultáneamente.
- El nuevo valor pasará de la memoria caché del procesador que se ha cambiado a la memoria compartida. Más tarde, sin embargo, también debe pasarse a todos los demás procesadores, para que no funcionen con un valor obsoleto.
- Este problema se conoce como el problema de la coherencia de la memoria caché, un caso especial del problema de la consistencia de la memoria, que requiere implementaciones de hardware que pueden manejar los problemas de concurrencia y sincronización, similar a la programación de subprocesos.

Características de la memoria compartida

- Las características principales de los sistemas de memoria compartida son:
  - **La memoria es la misma para todos los procesadores**. Por ejemplo, todos los procesadores asociados con la misma estructura de datos funcionarán con las mismas direcciones de memoria lógica, accediendo así a las mismas ubicaciones de memoria.
  - **La sincronización se obtiene leyendo las tareas de varios procesadores y permitiendo la memoria compartida**. De hecho, los procesadores solo pueden acceder a una memoria a la vez.
  - **Una ubicación de memoria compartida no debe cambiarse desde una tarea mientras otra tarea accede a ella.**
  - **Compartir datos entre tareas es rápido**. El tiempo requerido para comunicarse, es el tiempo que uno de ellos tarda en leer una sola ubicación (dependiendo de la velocidad de acceso a la memoria).

Memoria distribuida

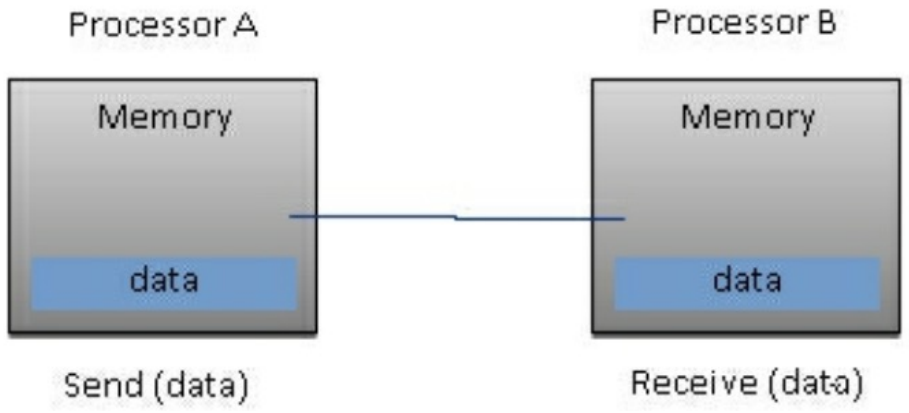
- En un sistema con memoria distribuida, la memoria está asociada con cada procesador y un procesador solo puede direccionar su propia memoria. Algunos autores se refieren a este tipo de sistema como un multicomputador, lo que refleja el hecho de que los elementos del sistema son, en sí mismos, sistemas pequeños y completos de un procesador y memoria.



The distributed memory architecture schema

Ventajas y desventajas

- Este tipo de organización tiene varias ventajas:
  - No hay conflictos de comunicación a nivel del bus o conmutador (swicth). Cada procesador puede usar el ancho de banda completo de su propia memoria local sin ninguna interferencia de otros procesadores.
  - La falta de un bus común significa que no hay límite intrínseco para el número de procesadores. El tamaño del sistema sólo está limitado por la red utilizada para conectar los procesadores.
  - No hay problemas con la coherencia de caché. Cada procesador es responsable de sus propios datos y no tiene que preocuparse por actualizar las copias.
- La principal desventaja es que la comunicación entre procesadores es más difícil de implementar. Si un procesador requiere datos en la memoria de otro procesador, entonces los dos procesadores no necesariamente deben intercambiar mensajes a través del protocolo de paso de mensajes.
  - Esto introduce dos fuentes de desaceleración: crear y enviar un mensaje de un procesador a otro lleva tiempo, y además, cualquier procesador debe detenerse para administrar los mensajes recibidos de otros procesadores. Un programa diseñado para trabajar en una máquina de memoria distribuida debe organizarse como un conjunto de tareas independientes que se comunican a través de mensajes:



Basic message passing

Características de la memoria distibuida

- Las características principales de los sistemas de memoria distribuida son:
  - **La memoria se distribuye físicamente entre los procesadores**; cada memoria local es directamente accesible solo por su procesador.
  - **La sincronización se logra moviendo datos** (incluso si es solo el mensaje en sí) entre procesadores (comunicación).
  - **La subdivisión de datos en las memorias locales afecta el rendimiento de la máquina**; es esencial hacer que las subdivisiones sean precisas, para minimizar la comunicación entre las CPU. Además de esto, el procesador que coordina estas operaciones de descomposición y composición debe comunicarse efectivamente con los procesadores que operan en las partes individuales de las estructuras de datos.
  - **El protocolo de paso de mensajes se utiliza para que las CPU puedan comunicarse entre sí a través del intercambio de paquetes de datos**. Los mensajes son unidades discretas de información, en el sentido de que tienen una identidad bien definida, por lo que siempre es posible distinguirlos entre sí.

Procesamiento Masivamente Paralelo (MPP)

- Las máquinas MPP están compuestas por cientos de procesadores (que pueden ser tan grandes como cientos de miles de procesadores en algunas máquinas) que están conectados por una red de comunicación.
- Las computadoras más rápidas del mundo se basan en estas arquitecturas; Algunos ejemplos de estos sistemas de arquitectura son Earth Simulator, Blue Gene, ASCI White, ASCI Red y ASCI Purple and Red Storm.

Clústeres de computadoras (workstations)

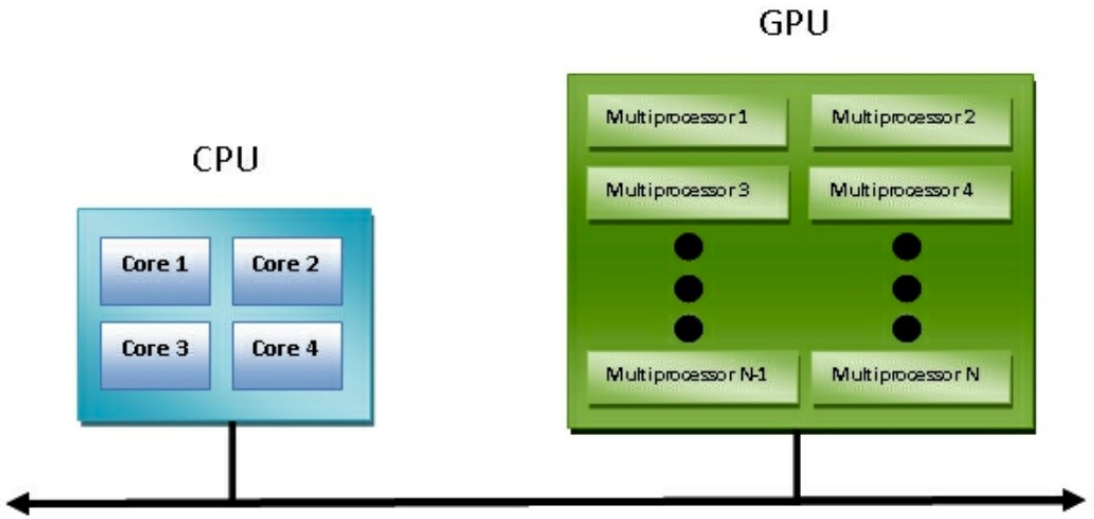
- Estos sistemas de procesamiento se basan en computadoras clásicas que están conectadas por redes de comunicación.
- En una arquitectura de clúster, definimos un nodo como una única unidad informática que participa en el clúster. Para el usuario, el clúster es totalmente transparente: toda la complejidad del hardware y el software está enmascarada y los datos y las aplicaciones se hacen accesibles como si fueran todos de un solo nodo.
- Se ha identificado tres tipos de clústeres:
  - **Clúster de alta disponibilidad (tolerancia a fallos)**: en este caso, la actividad del nodo se supervisa continuamente y, cuando uno deja de funcionar, otra máquina se hace cargo de esas actividades. El objetivo es garantizar un servicio continuo debido a la redundancia de la arquitectura.
  - **Clúster de balanceo de carga**: en este sistema, se envía una solicitud de trabajo al nodo que tiene menos actividad. Esto asegura que se tome menos tiempo para procesar el trabajo.
  - **Clúster de computo de alto rendimiento**: en este, cada nodo está configurado para proporcionar un rendimiento extremadamente alto. El proceso también se divide en múltiples trabajos sobre múltiples nodos. Los trabajos están paralelizados y se distribuyen a diferentes máquinas.

Arquitecturas heterogéneas

- La introducción de la GPU como acelerador en el mundo homogéneo de la supercomputación ha cambiado la naturaleza de cómo las supercomputadoras se utilizan y se programan.
- A pesar del alto rendimiento que ofrecen las GPUs, no pueden considerarse como una unidad de procesamiento autónoma, ya que siempre deben ir acompañadas de una combinación de CPUs.
- El paradigma de programación, por lo tanto, es muy simple: la CPU toma el control y computa en serie, asignando tareas al acelerador de gráficos que, computacionalmente, son muy costosas y tienen un alto grado de paralelismo.

Arquitecturas heterogéneas

- La comunicación entre una CPU y una GPU puede tener lugar, no solo mediante el uso de un bus de alta velocidad, sino también mediante el intercambio de una sola área de memoria para la memoria física o virtual. De hecho, en el caso de que ambos dispositivos no estén equipados con sus propias áreas de memoria, es posible hacer referencia a un área de memoria común utilizando las bibliotecas de software proporcionadas por los distintos modelos de programación, como CUDA y OpenCL.
- Las aplicaciones sobre arquitecturas heterogéneas pueden crear estructuras de datos en un solo espacio de direcciones y enviar un trabajo al hardware del dispositivo, lo cual es apropiado para la resolución de la tarea. Varias tareas de procesamiento pueden operar de manera segura en las mismas regiones para evitar problemas de consistencia de datos, gracias a las operaciones atómicas.
- Entonces, a pesar del hecho de que la CPU y la GPU no parecen funcionar de manera eficiente juntas, con el uso de esta nueva arquitectura, podemos optimizar su interacción y el rendimiento de las aplicaciones paralelas:



The heterogeneous architecture schema

Referencias

- Zaccone, Giancarlo. Python Parallel Programming Cookbook: Over 70 recipes to solve challenges in multithreading and distributed system with Python 3, 2nd Edition . Packt Publishing.