



Equipo 4

COMPUTACIÓN PARALELA

- Alejandro Javier Rivera Romo.
- Karla Cecilia Berumen Vazquez
- David Fernando Pavón Solís
- .joel misael leija de la rosa
- .jared



4.1 ASPECTOS BASICOS DE LA COMPUTACION PARALELA



La computación paralela es una técnica computacional, en la que varias instrucciones se ejecutan en simultáneo para ello se utilizan 2 o más elementos de procesamiento.

La computación paralela se basa en el principio de que los problemas grandes se pueden dividir en unos más pequeños que luego serán resueltos simultáneamente.



ENTENDAMOS MAS DE LA COMPUTACION PARALELA



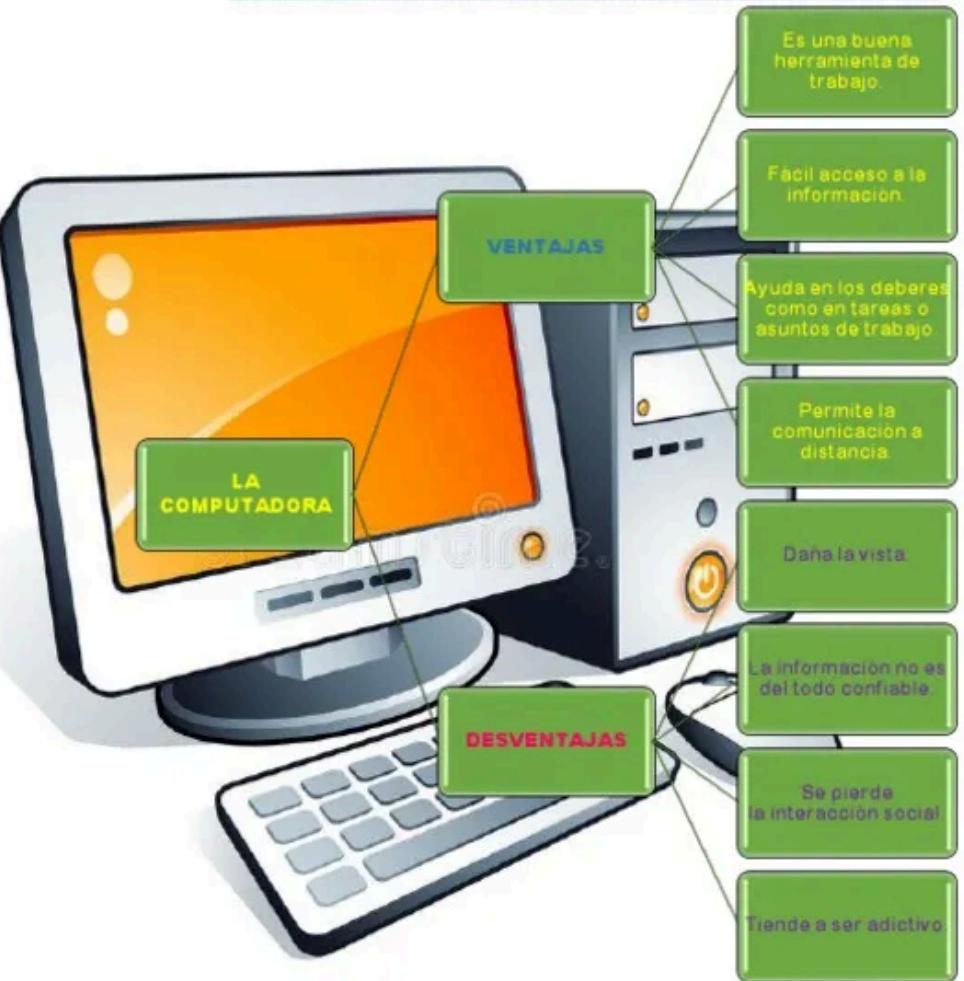
Imagina que tienes que recoger 100 canicas del piso. Si tú solo las recoges, te vas a tardar mucho tiempo. Pero si invitas a 3 amigos y cada uno recoge una parte, terminan más rápido, porque todos trabajan al mismo tiempo.

Eso es computación paralela: varias personas (o computadoras) haciendo partes de un trabajo al mismo tiempo para terminar más rápido.



VENTAJAS Y DESVENTAJAS DE LA COMPUTACION PARALELA

VENTAJAS Y DESVENTAJAS DE LA COMPUTADORA



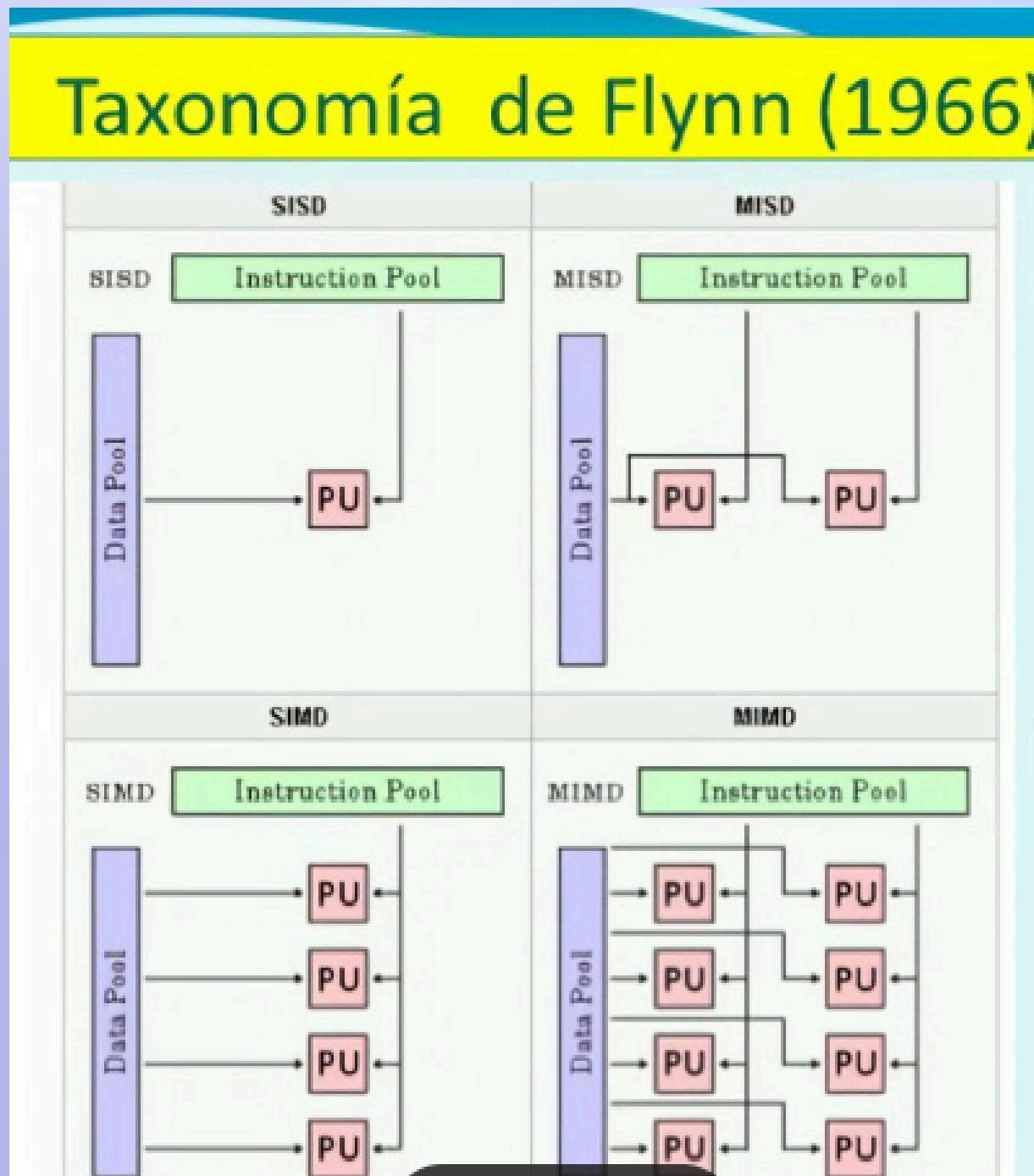
VENTAJAS:

- RESUELVE PROBLEMAS COMPLEJOS
- OBTENCIÓN DE RESULTADOS EN MENOS TIEMPO
- MEJOR BALANCE ENTRE RENDIMIENTO Y COSTO
- GRAN EXPANSIÓN Y ESCALABILIDAD

DESVENTAJAS

- MAYOR CONSUMO DE ENERGÍA
- MAYOR DIFICULTAD AL PROGRAMAR
- DIFICULTAD EN LA SINCRONIZACIÓN Y COMUNICACIÓN
- MAYOR POSIBILIDAD DE FALLOS

TAXONOMÍA DE FLYNN



SISD

Una instrucción, un dato.

💻 Computadoras tradicionales.

MISD

Varias instrucciones, un solo dato.

⚙️ Poco común, usado en sistemas de control.

SIMD

Una instrucción, muchos datos.

🎮 Útil en gráficos y procesamiento de imágenes.

MIMD

Varias instrucciones, muchos datos.

💻 Usado en computadoras modernas y servidores.

TIPOS DE PARALELISMO

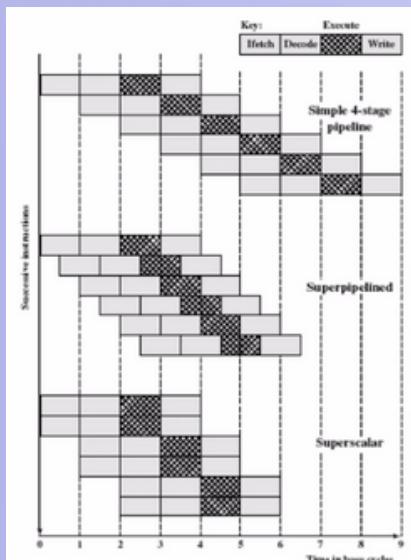
1. Paralelismo a nivel de bit

- Opera múltiples bits al mismo tiempo dentro de un procesador.
- Mejora la velocidad al procesar datos más grandes (por ejemplo, de 8 a 64 bits).



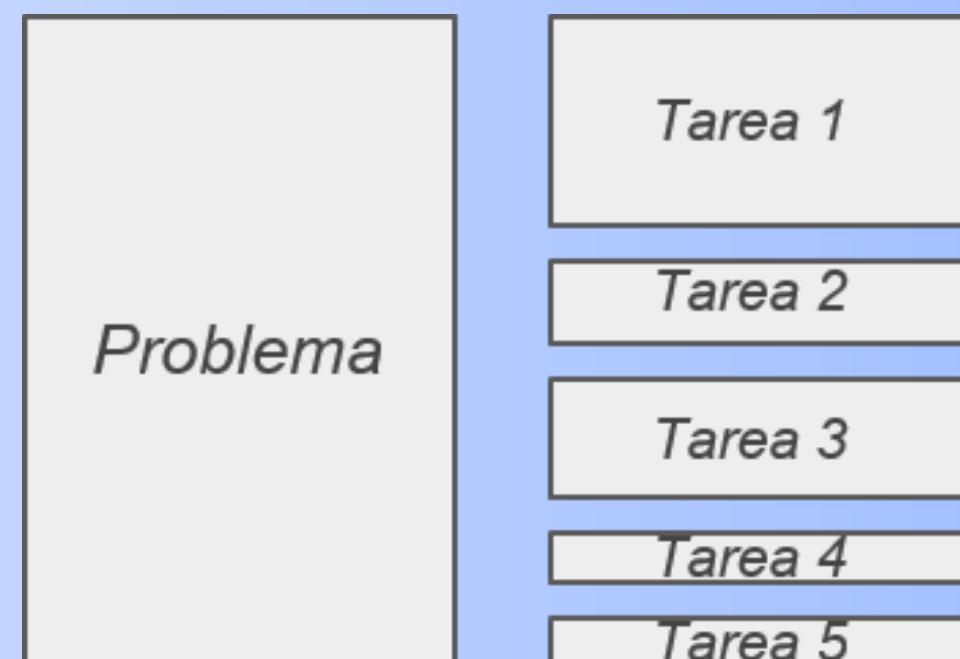
3. Paralelismo de datos

- La misma operación se aplica a conjuntos distintos de datos.
- Ejemplo: procesar miles de píxeles al mismo tiempo en una imagen (SIMD).



Paralelismo a nivel de instrucción (ILP)

- Ejecuta varias instrucciones al mismo tiempo dentro de un solo procesador.
- Usado en arquitecturas como pipeline o superescalar.



4.1-

ARQUITECTURA UMA

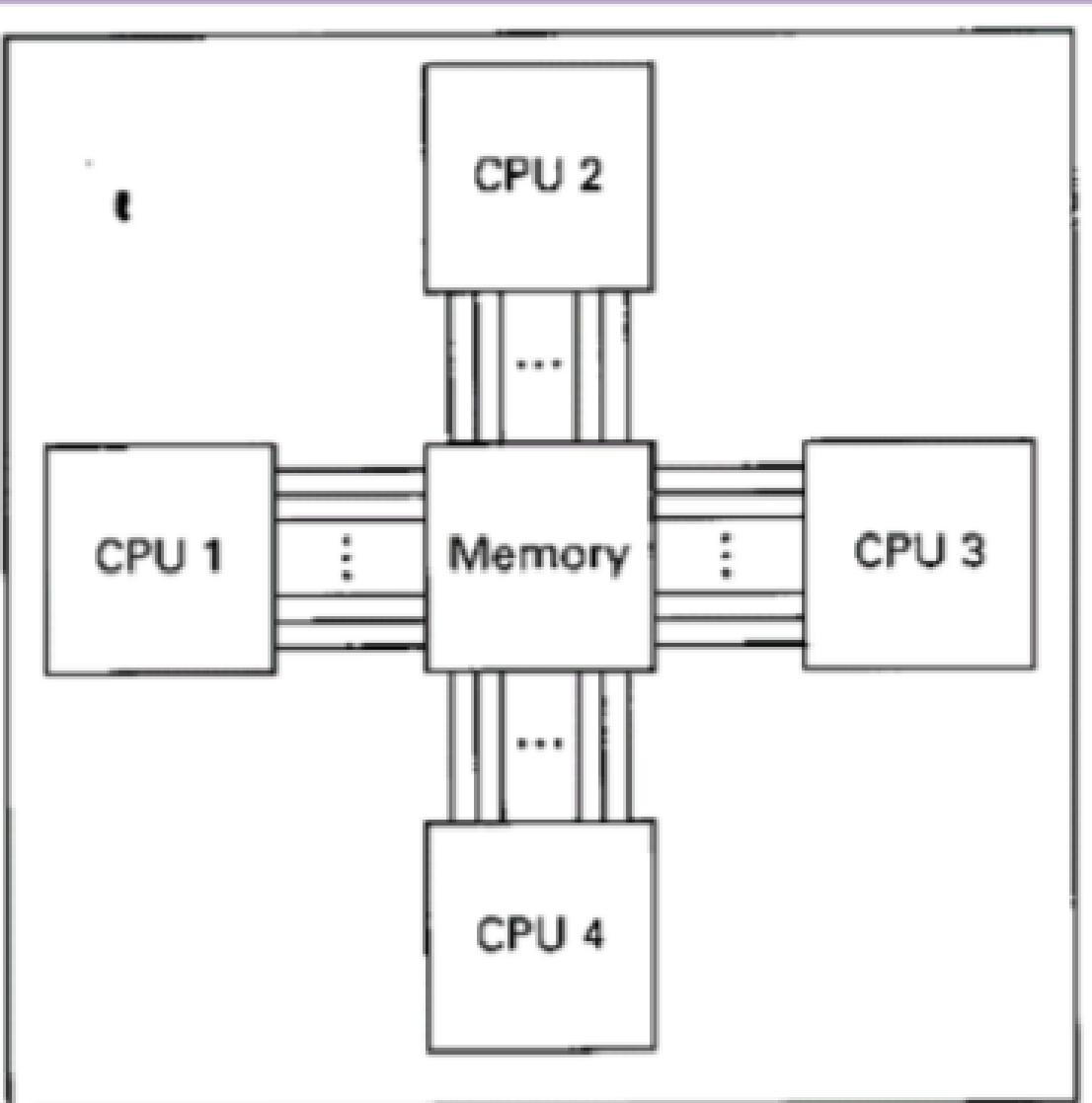
Memoria compartida (UMA)

UMA significa Acceso Uniforme a Memoria (por sus siglas en inglés: Uniform Memory Access).

Es un tipo de arquitectura donde todos los procesadores comparten la misma memoria y la acceden a la misma velocidad.

Características:

- Memoria central compartida.
- Todos los procesadores acceden a la memoria igual de rápido.
- Más fácil de programar.



4.1-

ARQUITECTURA NUMA

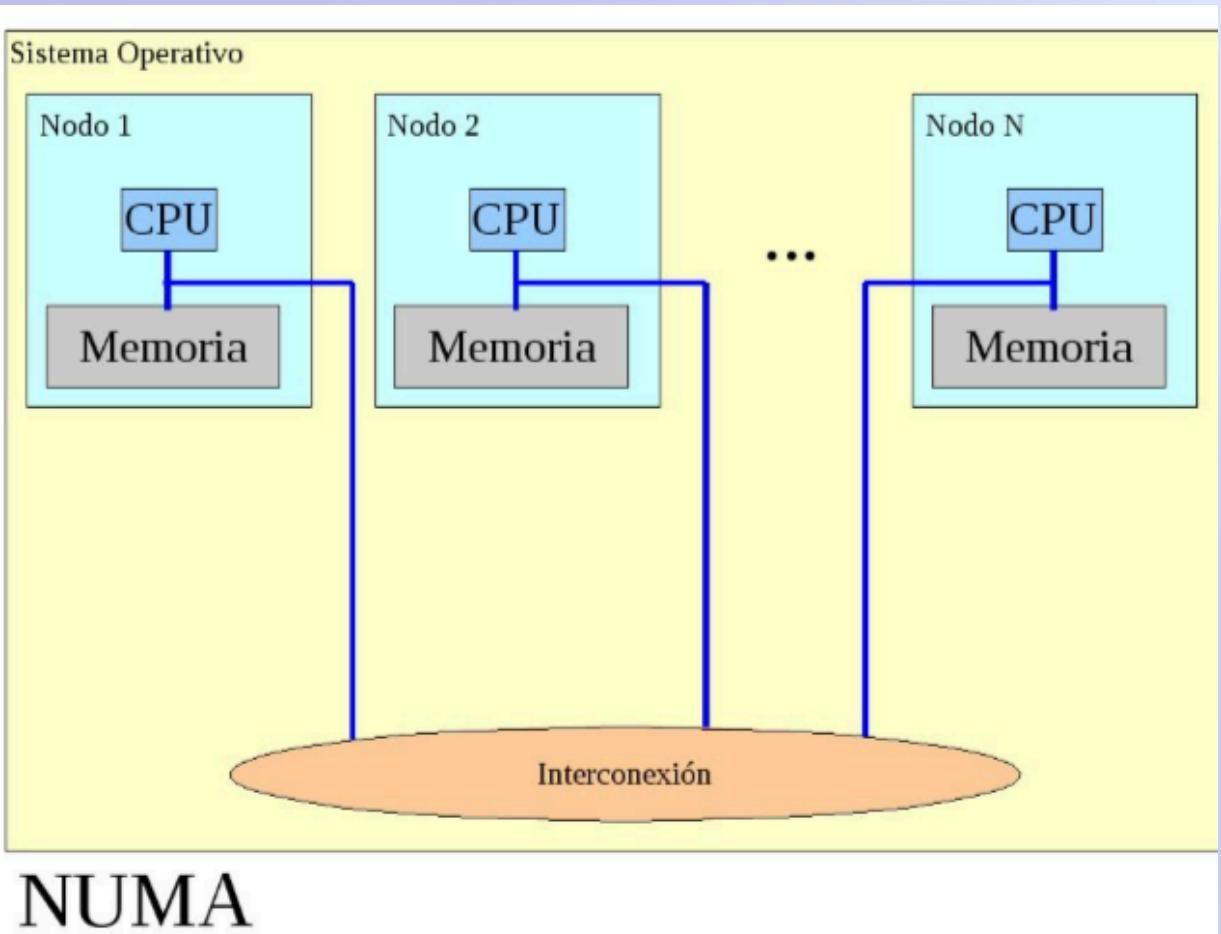
Memoria compartida (NUMA)

NUMA significa Acceso No Uniforme a Memoria (Non-Uniform Memory Access).

Es una arquitectura donde cada procesador tiene su propia memoria local, pero también puede acceder a la memoria de otros procesadores, aunque más lento.

Características:

- Cada procesador tiene memoria local.
- Acceso rápido a su propia memoria.
- Acceso más lento a la memoria de otros.
- Mejora el rendimiento en sistemas grandes.



4.2.1 TIPOS DE COMPUTACIÓN PARALELA

Los distintos tipos o arquitecturas de procesamiento en paralelo y cómo funcionan son:

- SISD (Single Instruction, Single Data)
- MISD (Multiple Instruction, Single Data)
- SIMD (Single Instruction, Multiple Data)
- MIMD (Multiple Instruction, Multiple Data)
- SPMD (Single Program, Multiple Data)
- MPP (Massively Parallel Processing)





SISD (SINGLE INSTRUCTION, SINGLE DATA)

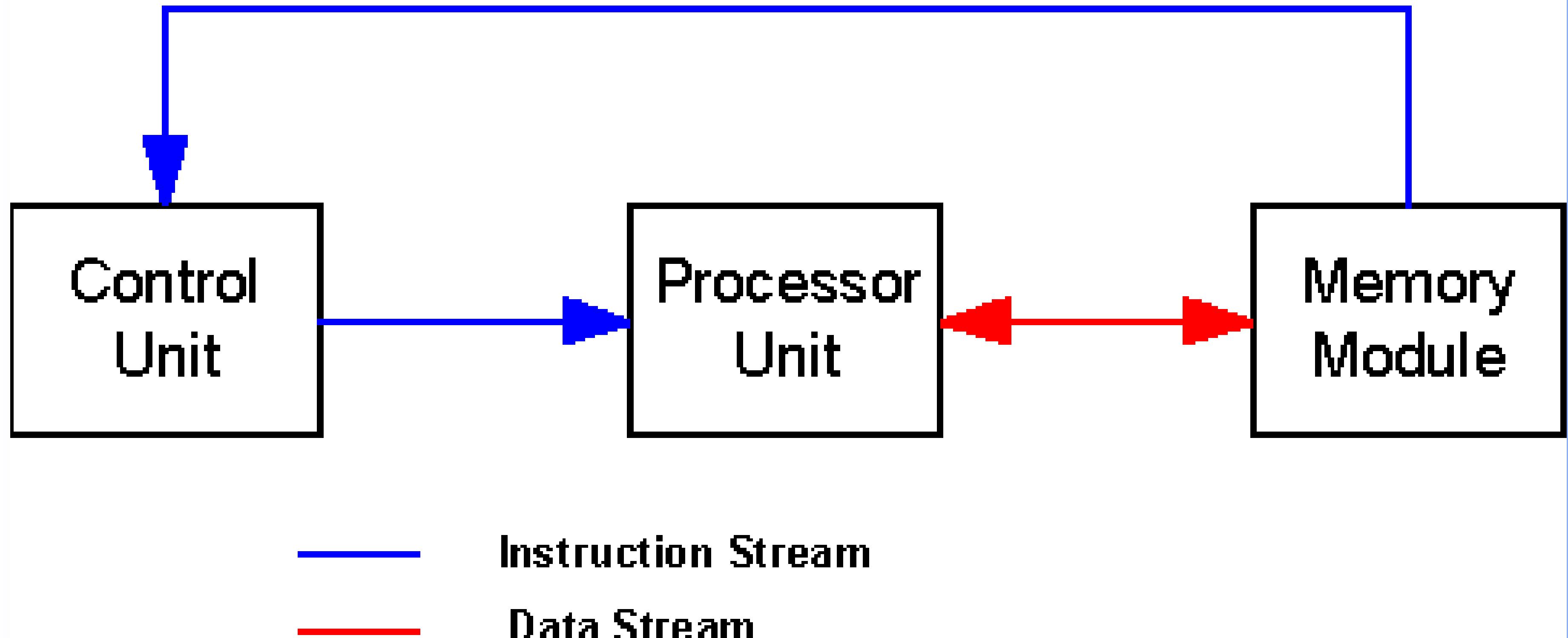


En el tipo de computación denominada Instrucción Única, Datos Únicos un único procesador se encarga de gestionar simultáneamente un algoritmo como una única fuente de datos.

SISD representa una organización informática que tiene una unidad de control, una de procesamiento y una de memoria similar a la computadora serie. SISD ejecuta las instrucciones secuencialmente y puede o no ser capaz de realizar procesamiento en paralelo, dependiendo de su configuración. Las instrucciones ejecutadas secuencialmente podrán cruzarse a lo largo de sus fases de ejecución. Es posible que haya más de una unidad funcional dentro de una computadora SISD. Sin embargo, una unidad de control está a cargo de todas las unidades funcionales.

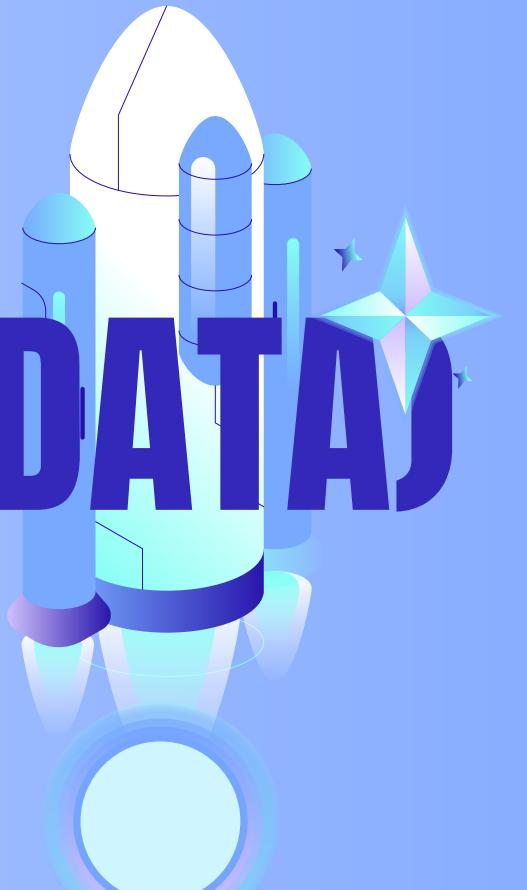
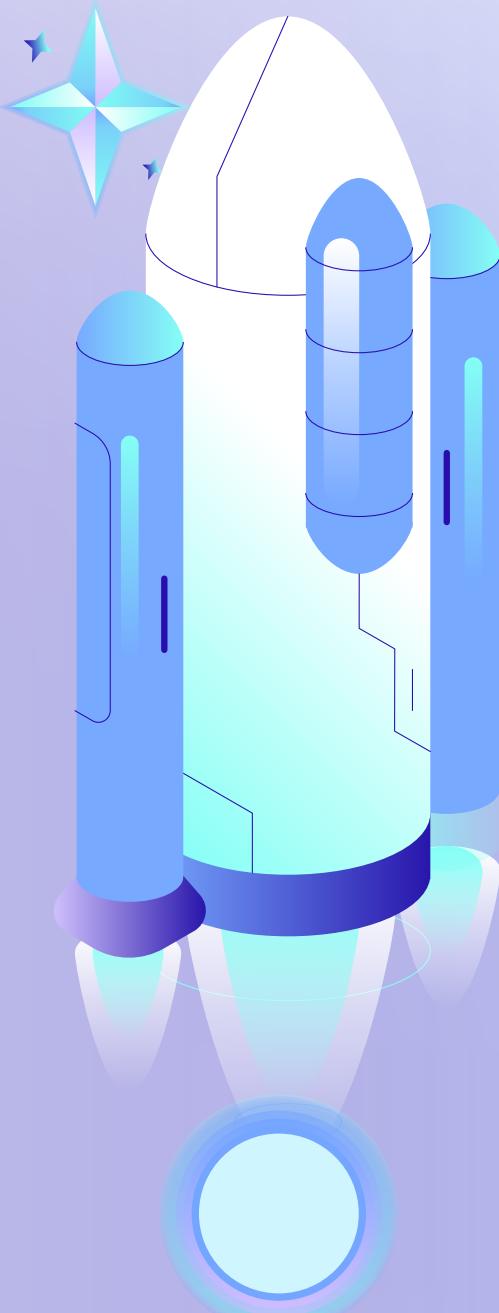
Dichos sistemas permiten el procesamiento de tuberías o el uso de numerosas unidades funcionales para lograr un procesamiento paralelo.

SISD Computer





MISD (MULTIPLE INSTRUCTION, SINGLE DATA)



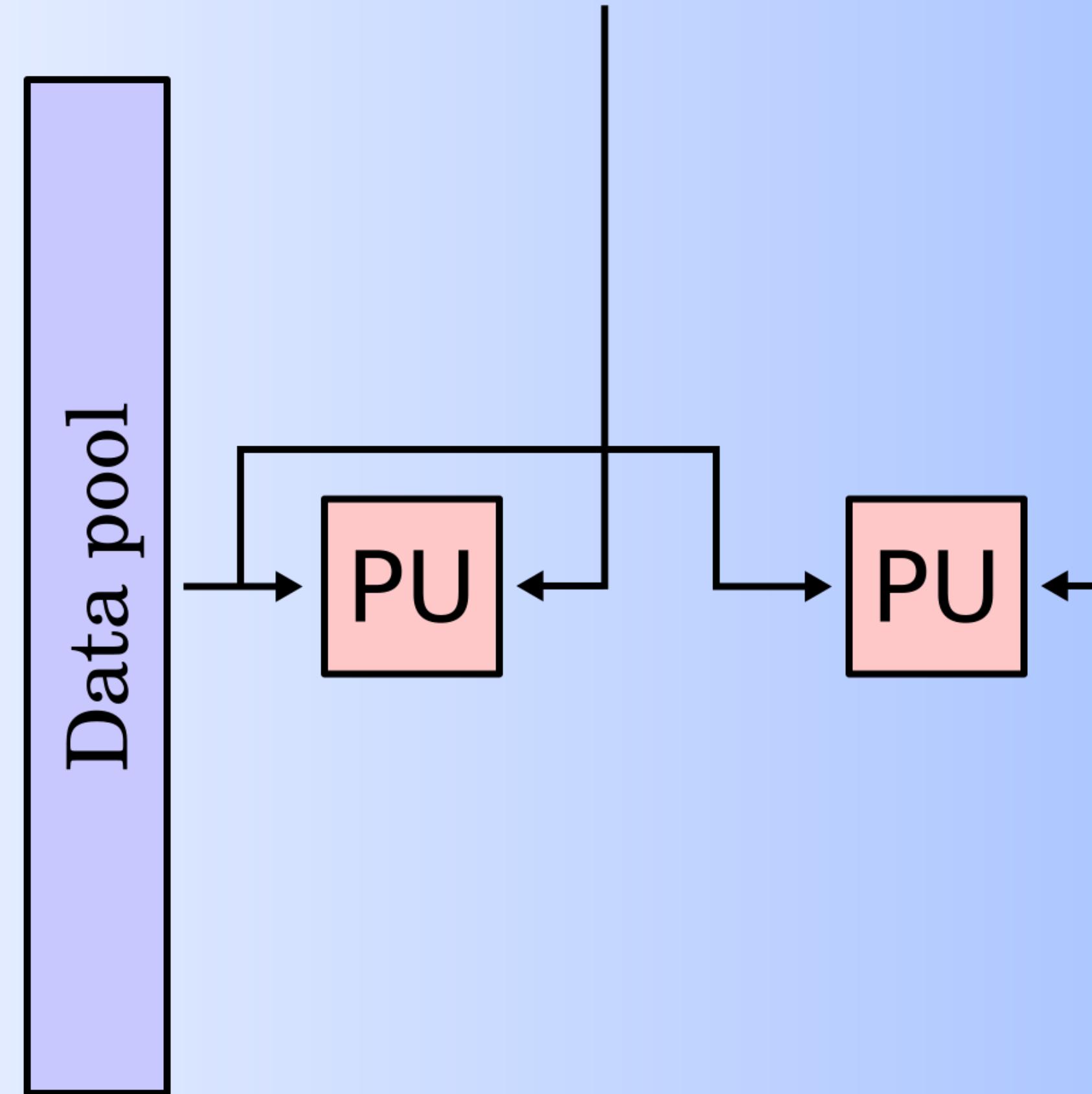
Los procesadores múltiples son estándar en las computadoras que utilizan Instrucción Múltiple, Datos Únicos (MISD). Al utilizar varios algoritmos, todos los procesadores comparten los mismos datos de entrada.

Las computadoras MISD pueden realizar simultáneamente muchas operaciones en el mismo lote de datos. Como era de esperar, la cantidad de operaciones se ve afectada por la cantidad de procesadores disponibles.

La estructura MISD consta de muchas unidades de procesamiento, cada una de las cuales opera según sus instrucciones y sobre un flujo de datos comparable. La salida de un procesador se convierte en la entrada del siguiente.

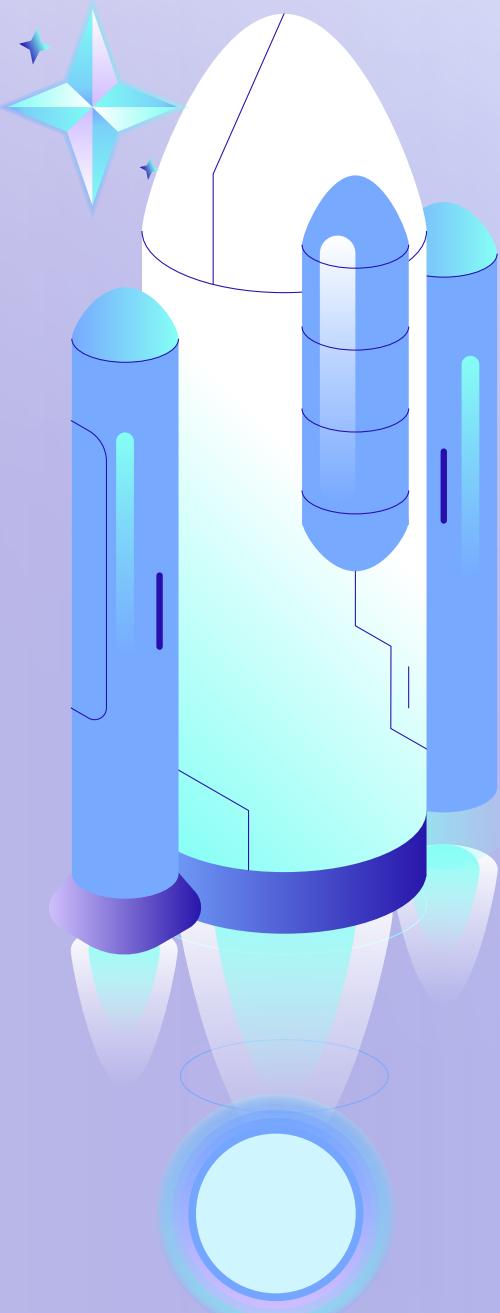
MISD

Instruction pool



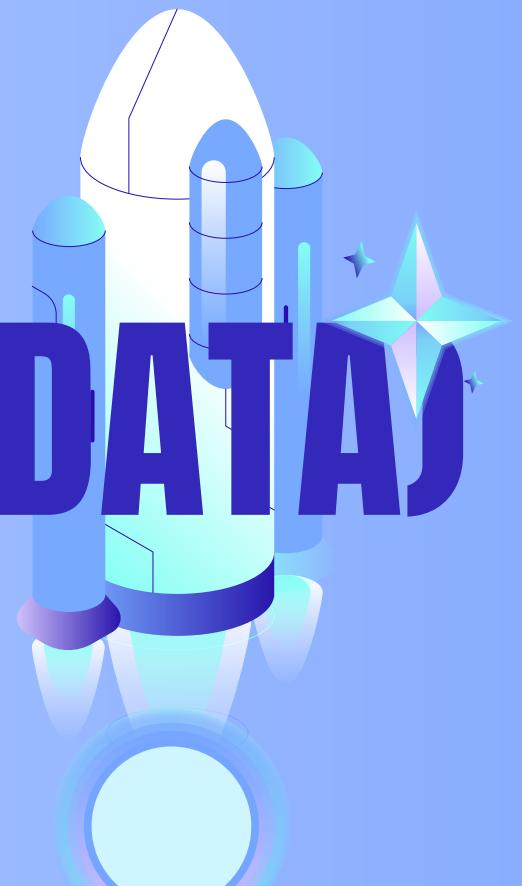


SIMD (SINGLE INSTRUCTION, MULTIPLE DATA)



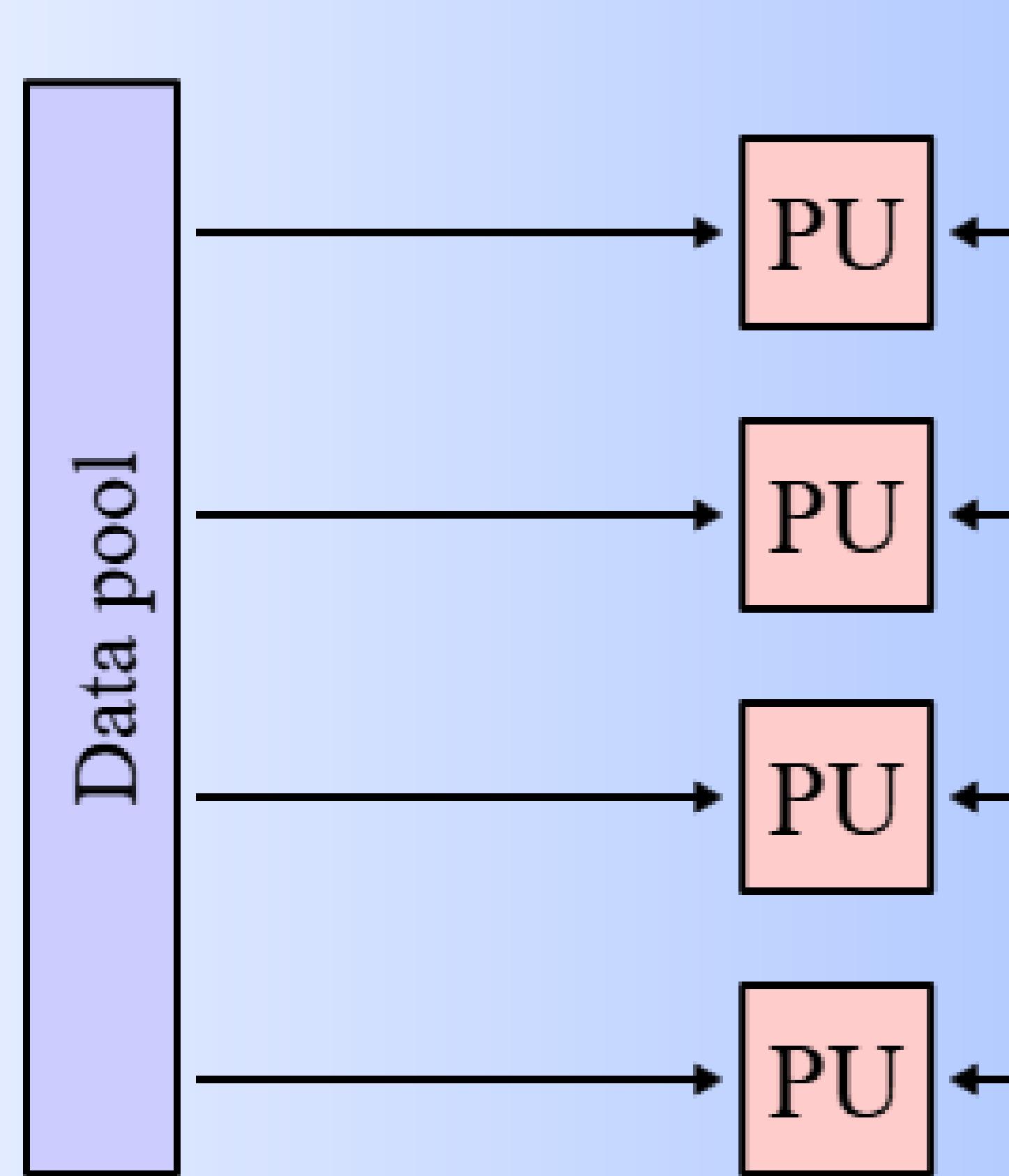
Las computadoras que utilizan la arquitectura SIMD (Instrucción Única, Datos Múltiples) tienen múltiples procesadores que ejecutan instrucciones idénticas. Sin embargo, cada procesador proporciona las instrucciones con su colección única de datos.

Las computadoras SIMD aplican el mismo algoritmo a varios conjuntos de datos. La arquitectura SIMD cuenta con varios componentes de procesamiento, los cuales están bajo la supervisión de una única unidad de control. Mientras procesa numerosos datos, cada uno recibe la misma instrucción de la unidad de control. Varios módulos incluidos en el subsistema compartido ayudan en la comunicación simultánea con cada CPU.



SIMD

Instruction pool

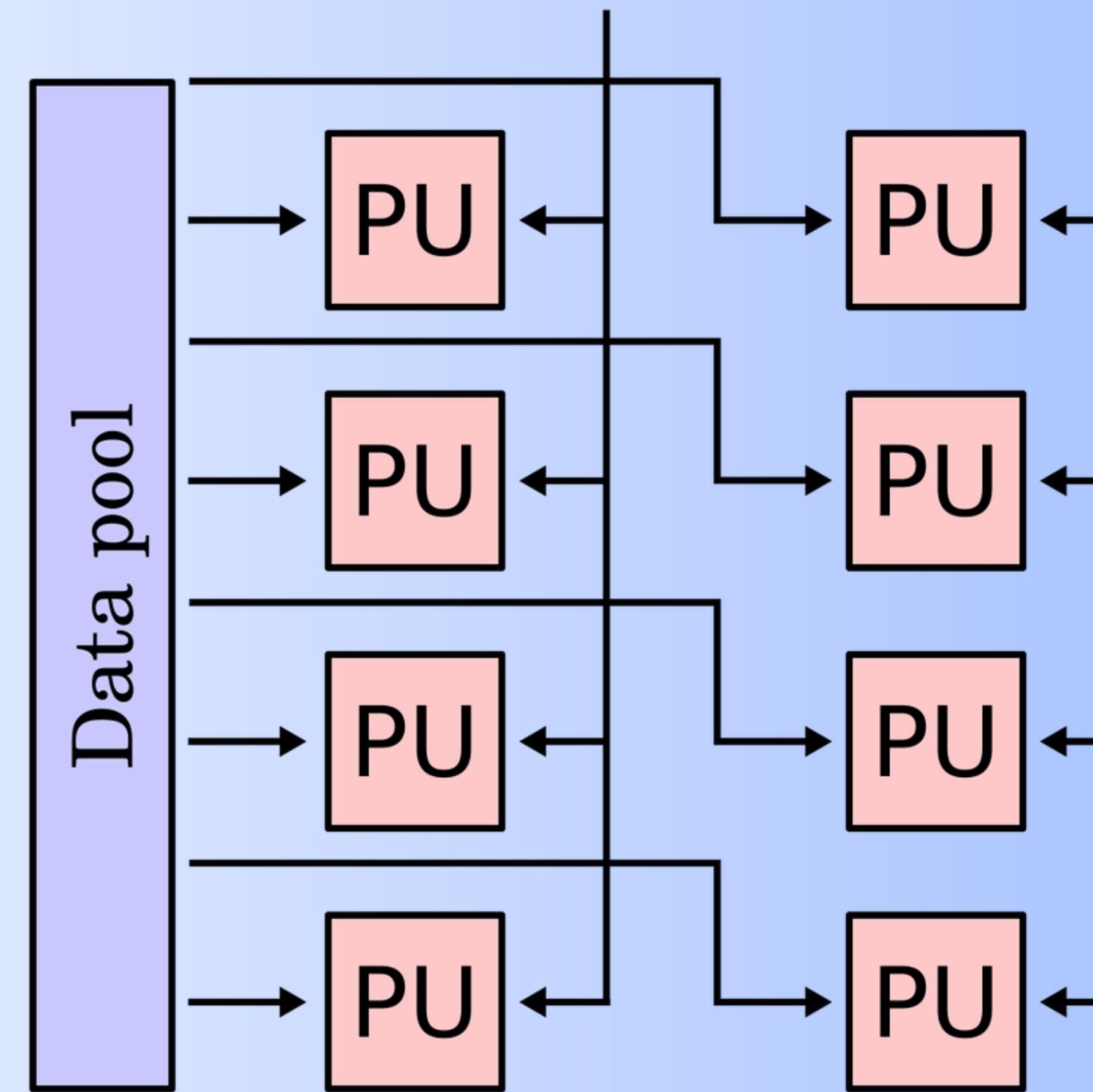


MIMD (MULTIPLE INSTRUCTION, MULTIPLE DATA)

- Definición: Es un modelo de computación paralela donde múltiples procesadores ejecutan instrucciones diferentes sobre diferentes datos de forma simultánea.
- Ejemplo: Cada procesador en un sistema MIMD puede estar resolviendo una tarea distinta, como uno procesando imágenes y otro haciendo cálculos matemáticos.
- Uso: Muy común en supercomputadoras y sistemas multiprocesador, como servidores y clusters.

MIMD

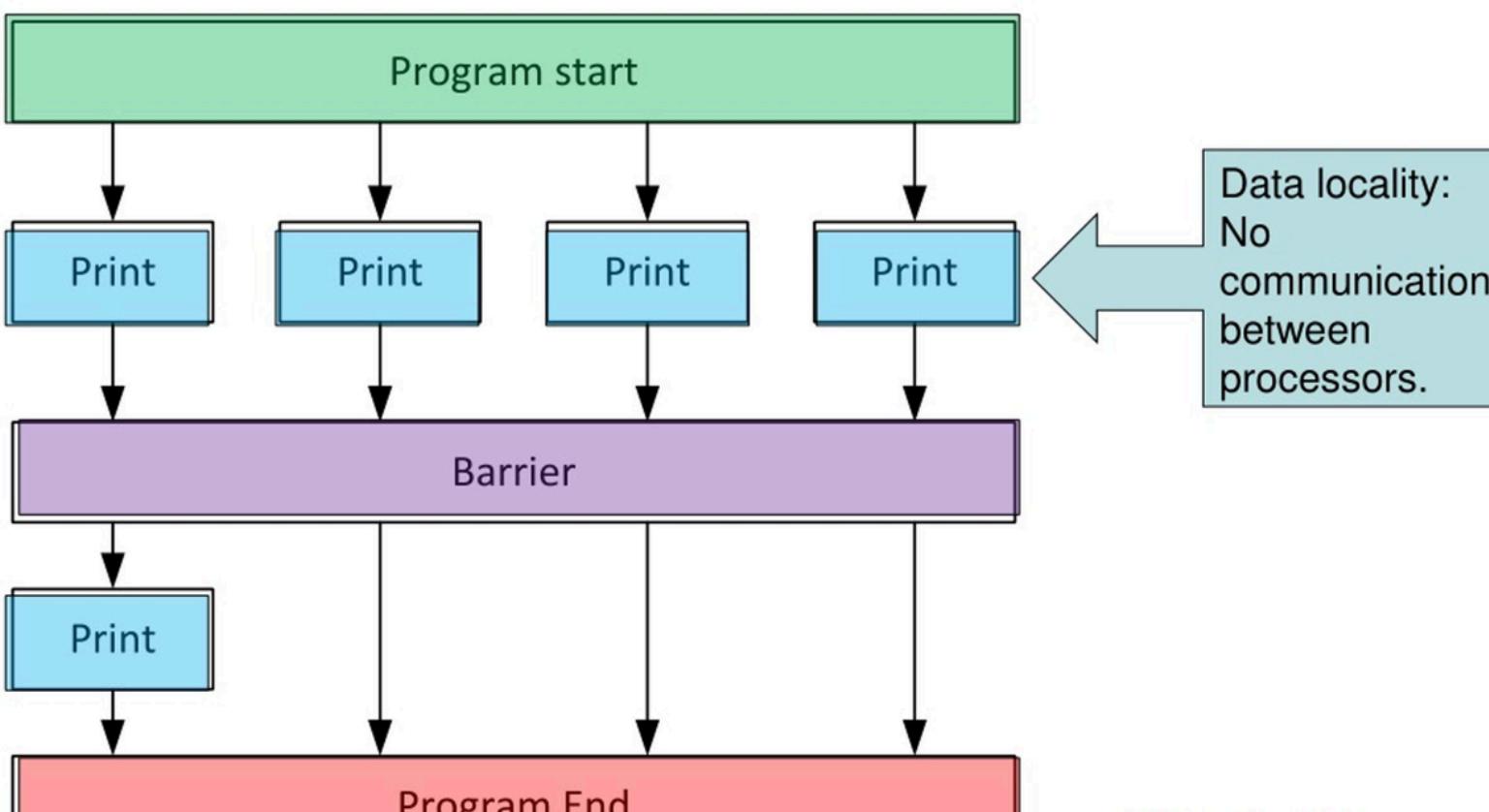
Instruction pool



SPMD (SINGLE PROGRAM, MULTIPLE DATA)

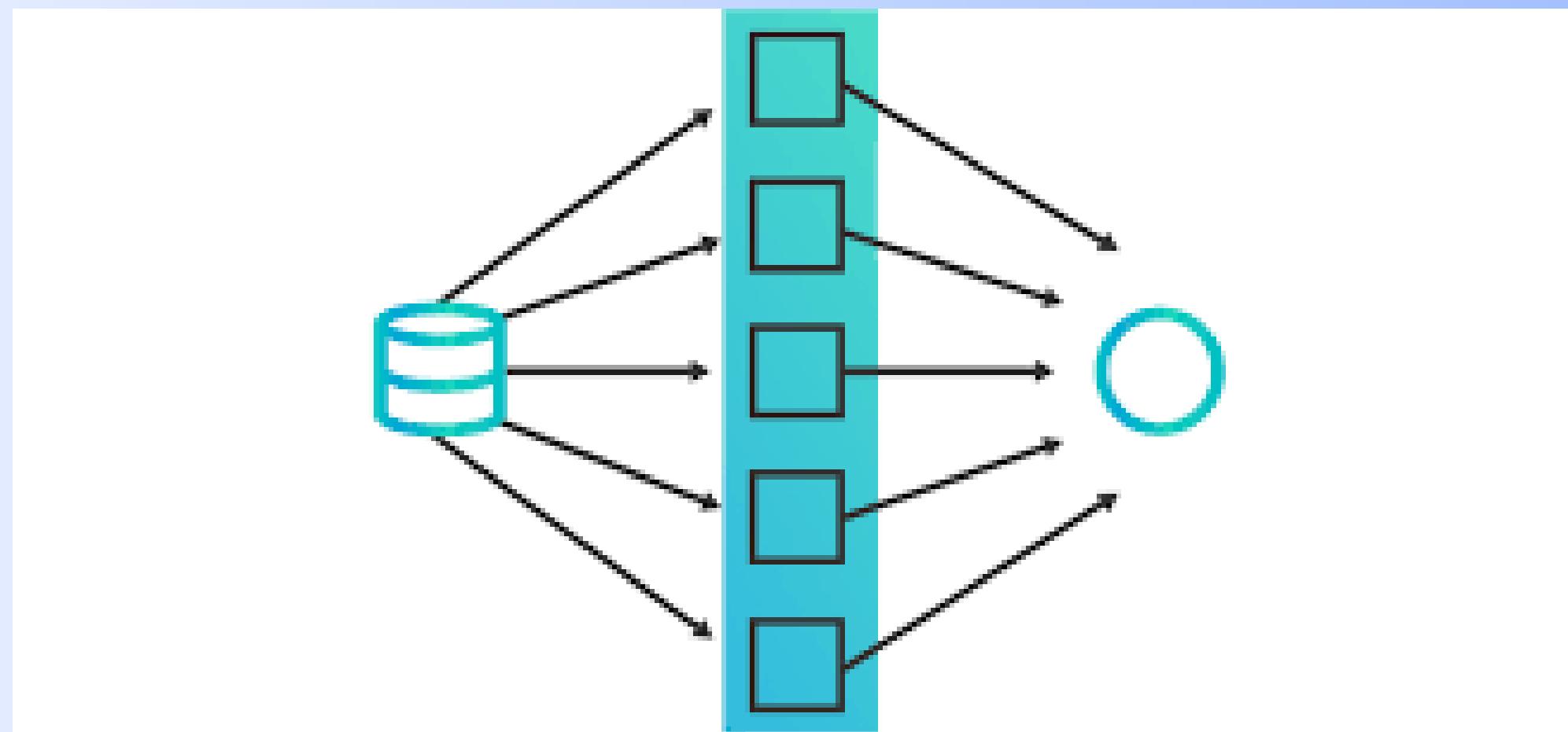
- Definición: Todos los procesadores ejecutan el mismo programa, pero operan sobre diferentes conjuntos de datos.
- Ejemplo: Varios núcleos ejecutando un mismo código para analizar distintas partes de un archivo grande.
- Uso: Muy usado en programación paralela moderna, como en CUDA (GPU) o MPI

The first example of SPMD: Titanium



MPP (MASSIVELY PARALLEL PROCESSING)

- Definición: Arquitectura de sistemas donde muchos procesadores trabajan en paralelo para resolver un problema grande dividiéndolo en partes.
- Ejemplo: Miles de procesadores resolviendo un problema científico complejo al mismo tiempo.
- Uso: Utilizado en supercomputación, análisis de big data, simulaciones científicas, etc.



4.2.2) ARQUITECTURA DE COMPUTADORAS SECUENCIALES



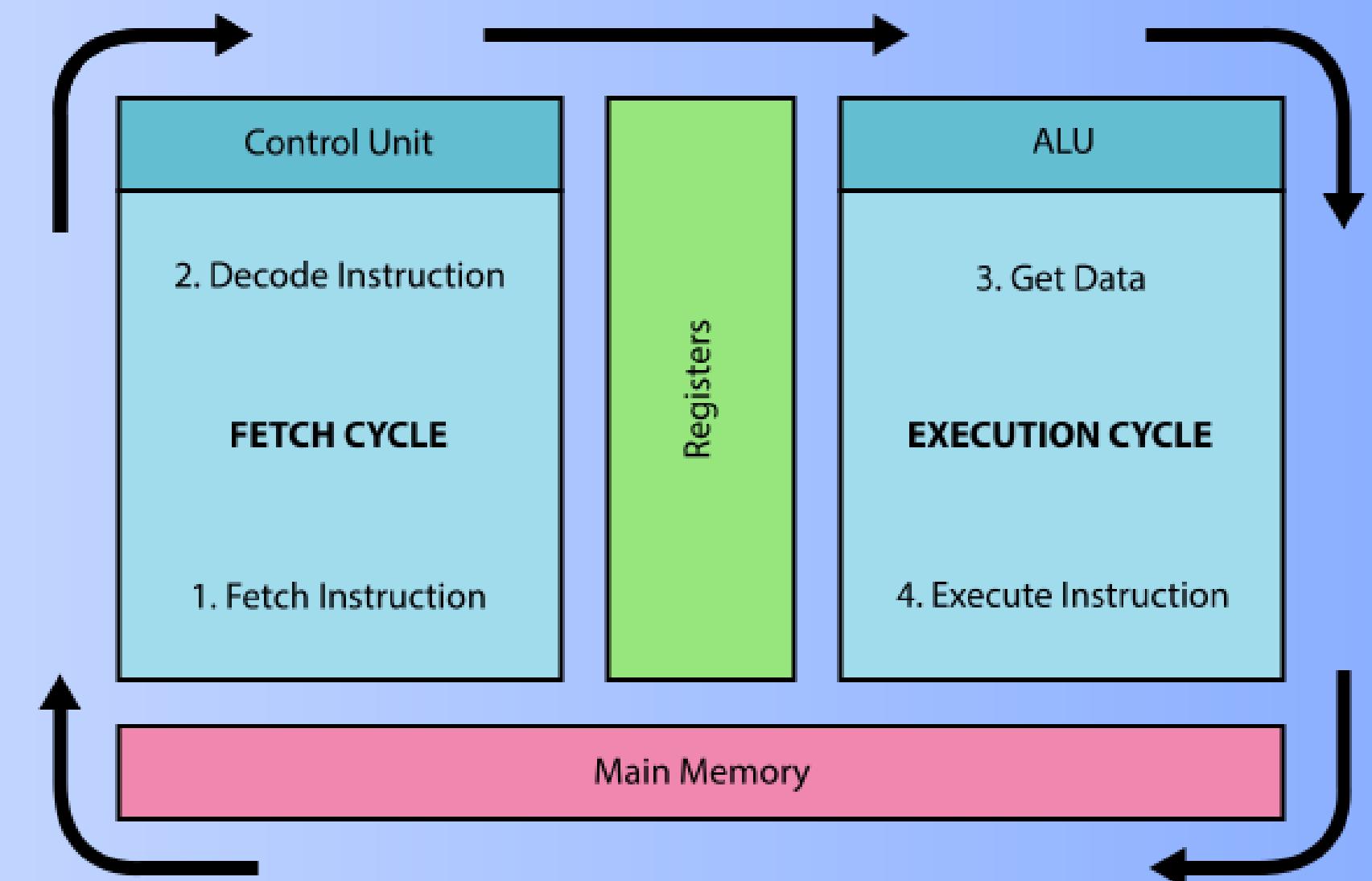
¿QUÉ ES UNA COMPUTADORA SECUENCIAL?

- Es aquella que ejecuta **instrucciones una por una**, de forma ordenada.
- Solo hay **un procesador** o un núcleo, que procesa todas las instrucciones del programa.
- Toda la lógica de control sigue una línea recta de ejecución (**flujo secuencial**).
- Ejemplo: Si un programa tiene 1000 instrucciones, se ejecutan una tras otra, sin dividirlas entre varios núcleos o hilos.

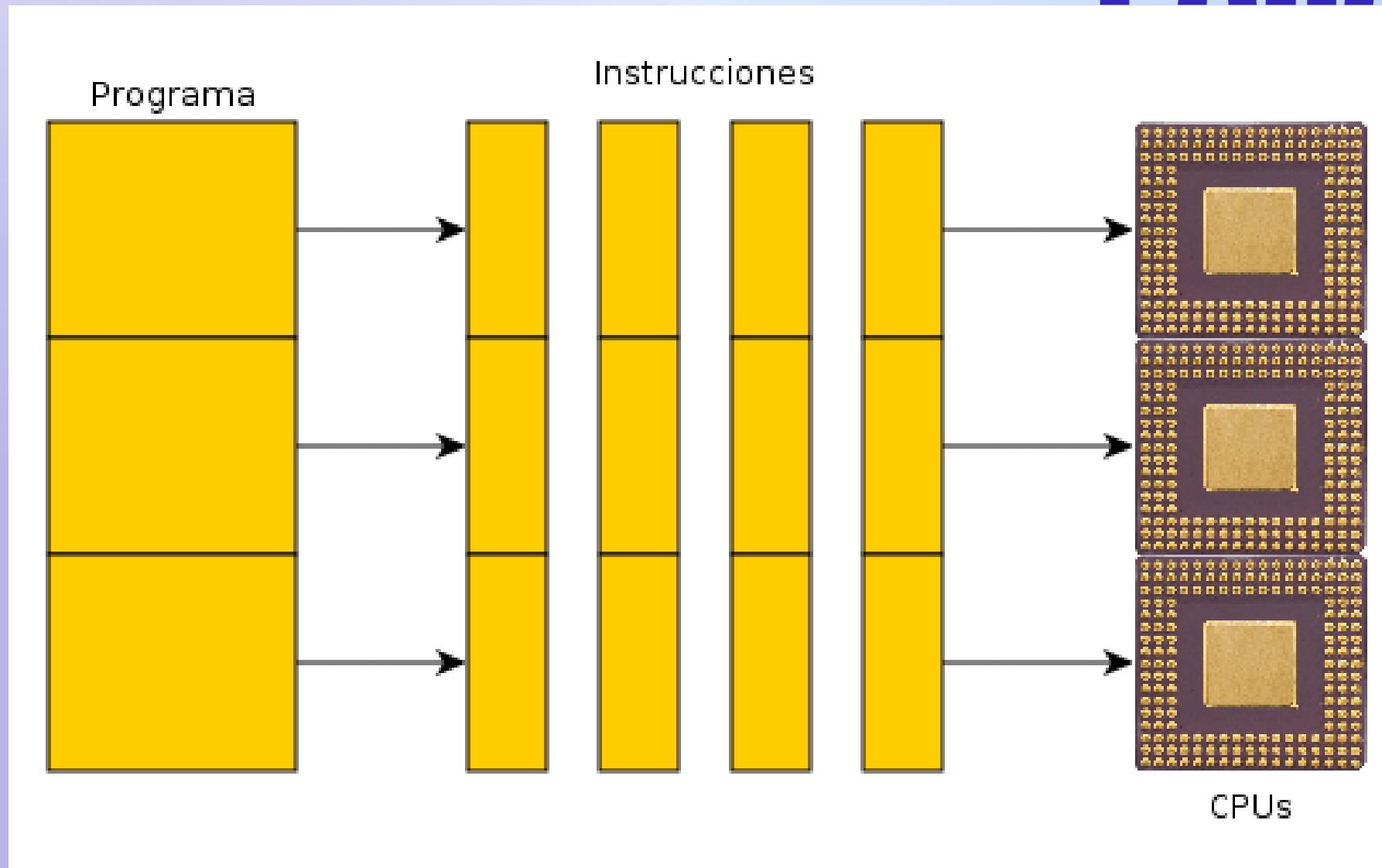


MODELO VON NEUMANN Y EL CICLO DE INSTRUCCIÓN (F-D-E)

- **Unidad de procesamiento (CPU)**: interpreta y ejecuta instrucciones.
 - **Memoria principal (RAM)**: almacena datos e instrucciones.
 - **Unidad de control**: dirige el flujo de instrucciones.
 - **Unidad aritmético-lógica (ALU)**: realiza operaciones matemáticas y lógicas.
 - **Bus de datos**: transporta datos entre la memoria y el procesador.
-
- **Fetch (Búsqueda)**: se obtiene la instrucción de la memoria.
 - **Decode (Decodificación)**: se interpreta qué debe hacer la CPU.
 - **Execute (Ejecución)**: la CPU realiza la acción (como una suma, acceso a memoria, etc.)



DIFERENCIAS CON LA ARQUITECTURA PARALELA



1. Múltiples Unidades de Procesamiento (CPUs o núcleos)

- Varios procesadores trabajan juntos.
- Cada procesador puede ejecutar una instrucción distinta al mismo tiempo.

2. Memoria compartida o distribuida

- Algunos sistemas comparten la memoria (como si todos miraran el mismo cuaderno).
- Otros tienen su propia memoria por separado (como si cada uno tuviera su propio cuaderno).

3. Interconexión

- Hay que comunicar los procesadores entre sí.
- Usan redes de interconexión para coordinar y compartir datos (esto lo verás más en los subtemas 4.3 y 4.4).

TIPOS DE PARALELISMO

a) Paralelismo a nivel de instrucción (ILP)

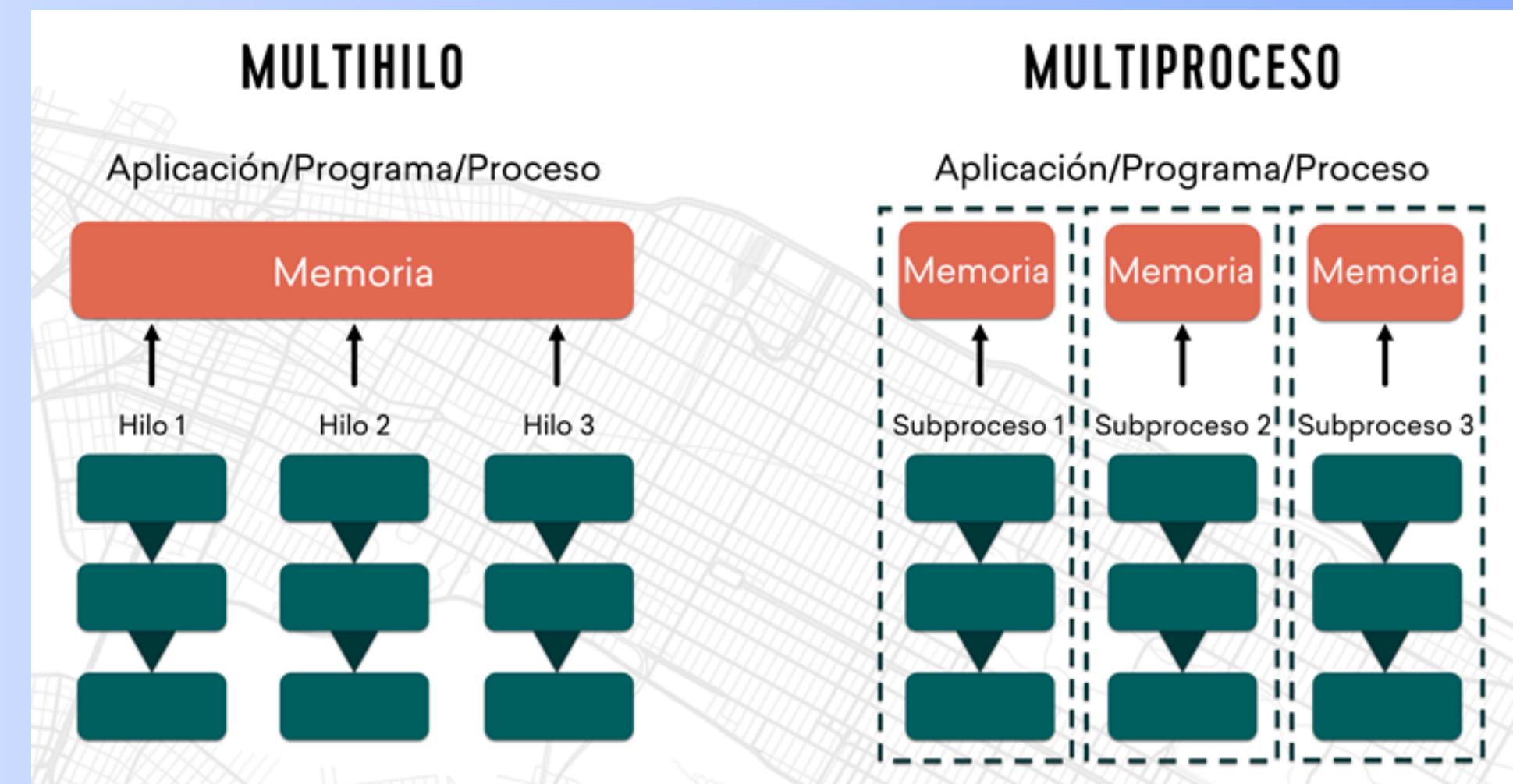
- Dentro de un mismo procesador, se intenta ejecutar varias instrucciones al mismo tiempo.
- Ejemplo: mientras una instrucción suma, otra puede estar cargando un dato.

b) Paralelismo a nivel de datos

- Se hacen las mismas operaciones sobre conjuntos diferentes de datos.
- Muy usado en procesamiento de imágenes, simulaciones científicas, etc.

c) Paralelismo a nivel de tareas

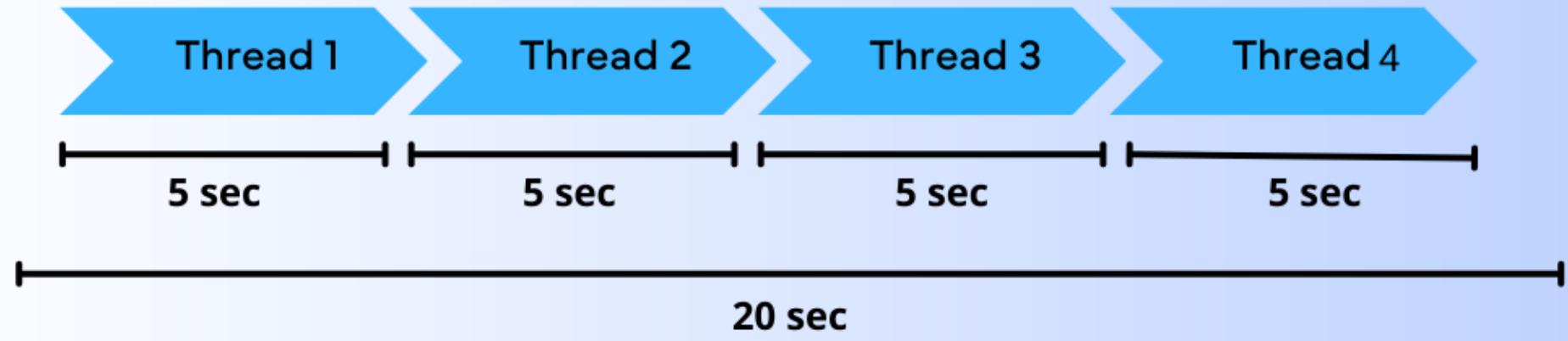
- Cada procesador ejecuta una tarea diferente del programa.
- Ejemplo: en un videojuego, un procesador maneja la física, otro la IA, otro los gráficos.



4.2.3

Organización de direcciones de memoria

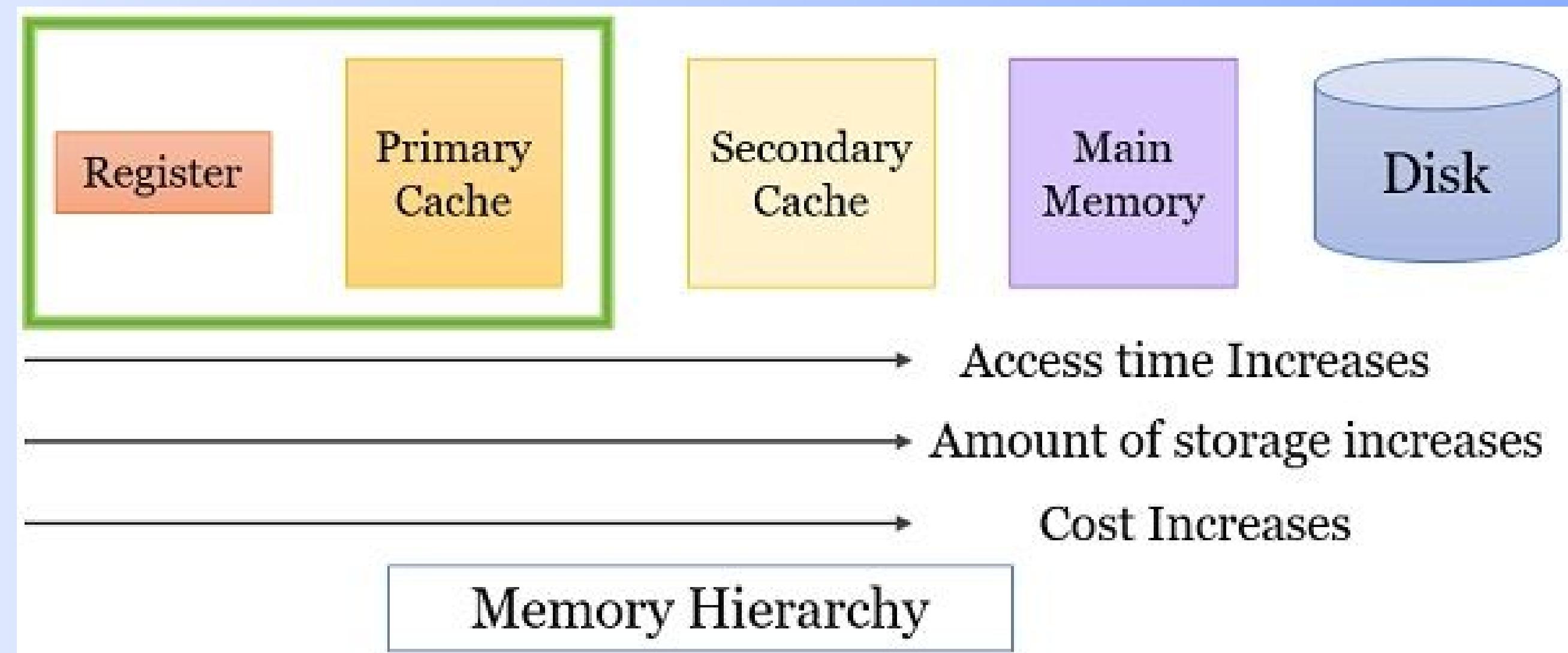
LA ORGANIZACIÓN DE DIRECCIONES DE MEMORIA EN COMPUTACIÓN PARALELA



La organización de memoria se refiere a cómo se distribuyen, acceden y gestionan los datos en memoria cuando múltiples unidades de procesamiento (procesadores, hilos, núcleos) ejecutan tareas simultáneamente.

Conceptos clave a tomar en cuenta cuando estudiamos la memoria en concurrencia:

- El rendimiento (latencia, ancho de banda).
- La coherencia de datos.
- La comunicación entre procesos/hilos.
- La dificultad de programación.

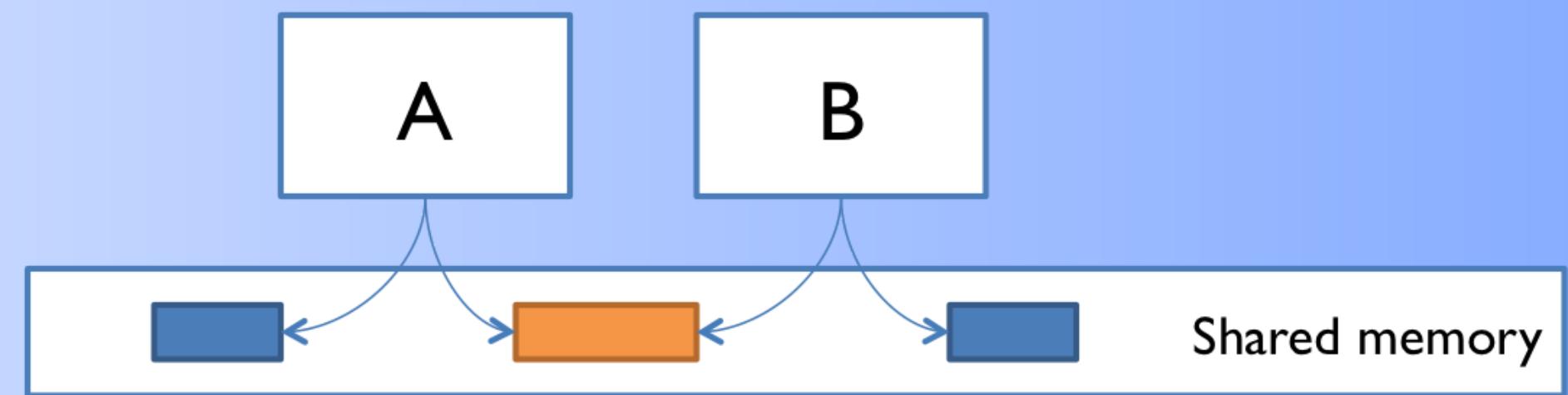


Hay dos modelos principales para organizar la memoria en sistemas paralelos:

MEMORIA COMPARTIDA (SHARED MEMORY)

Todos los hilos/procesadores comparten un mismo espacio de direcciones de memoria.

Este es el modelo que se usa típicamente en programas multihilo con Java usando Thread, ExecutorService, ForkJoinPool, etc.





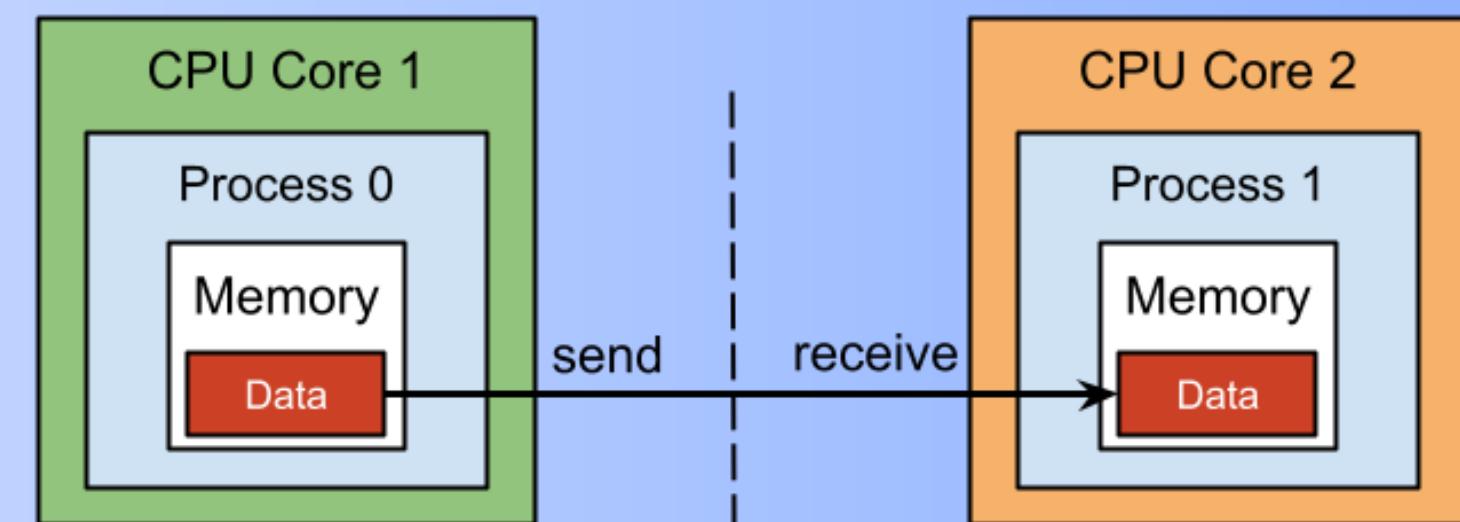
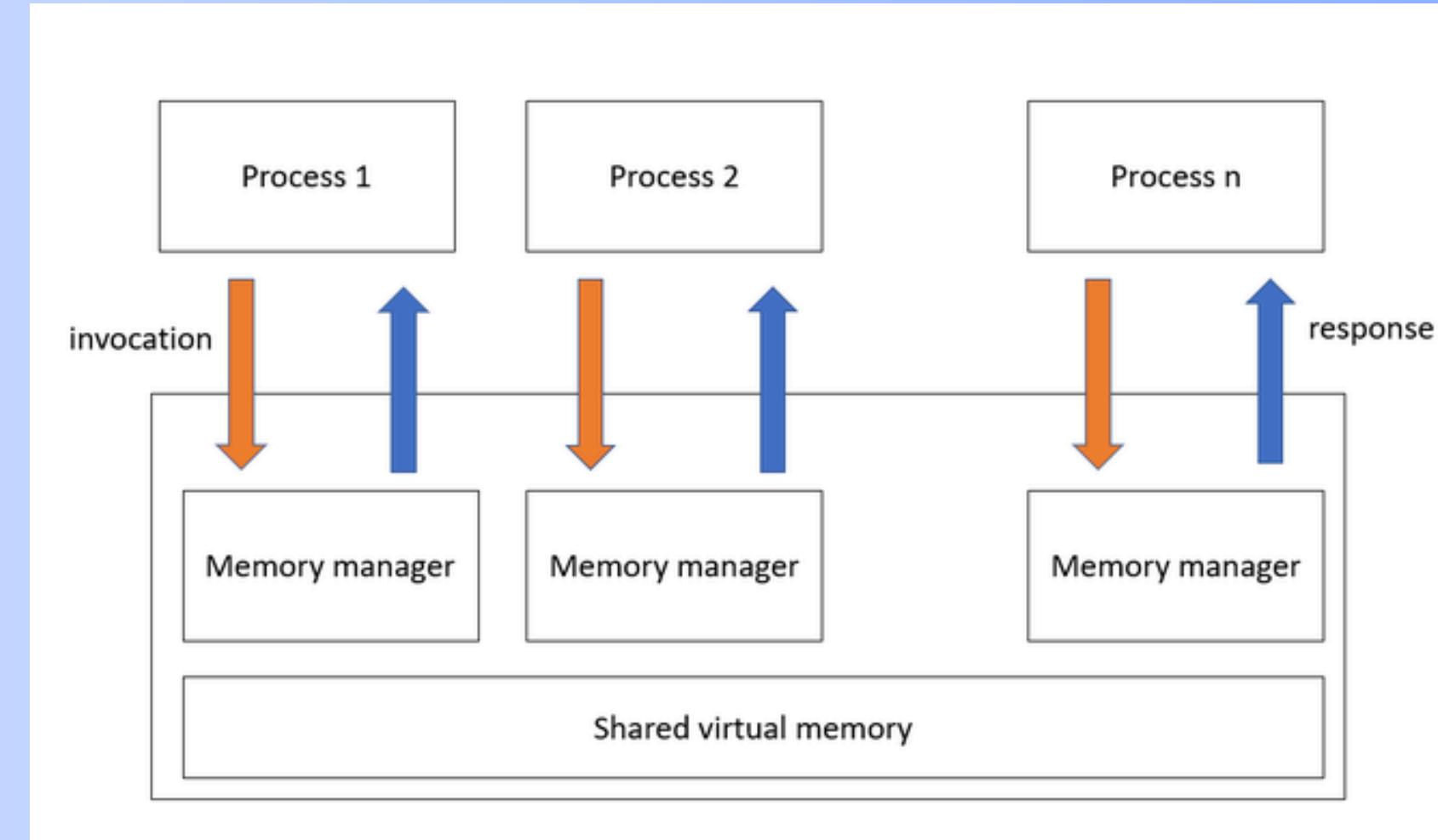
```
1 public class ContadorCompartido {  
2     private static int contador = 0;  
3  
4     public static void main(String[] args) throws InterruptedException {  
5         Runnable tarea = () -> {  
6             for (int i = 0; i < 1000; i++) {  
7                 contador++; // Acceso compartido (no seguro)  
8             }  
9         };  
10  
11     Thread hilo1 = new Thread(tarea);  
12     Thread hilo2 = new Thread(tarea);  
13  
14     hilo1.start();  
15     hilo2.start();  
16  
17     hilo1.join();  
18     hilo2.join();  
19  
20     System.out.println("Contador final: " + contador);  
21 }  
22 }  
23
```

MEMORIA DISTRIBUIDA (DISTRIBUTED MEMORY)

Cada procesador o nodo tiene su propia memoria local, y no puede acceder directamente a la memoria de los demás. La comunicación se realiza mediante paso de mensajes.

Este modelo se encuentra en:

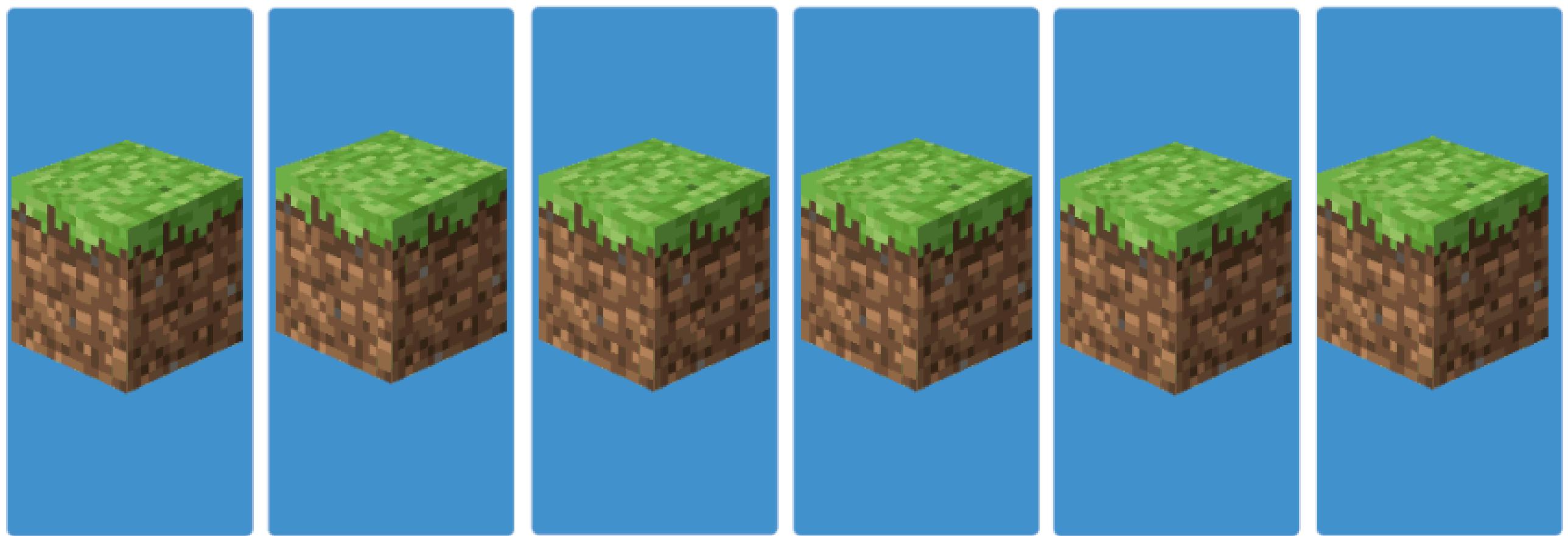
- Clústeres de computadoras.
- Sistemas con MPI.
- Algunas arquitecturas NUMA.



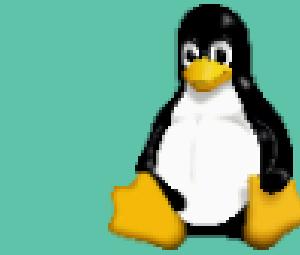
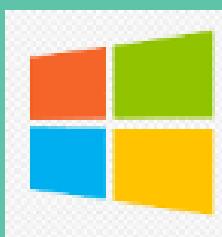


```
1 #include <mpi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main(int argc, char** argv) {
6     int size, rank;
7
8     // Inicializa el entorno MPI
9     MPI_Init(&argc, &argv);
10
11    // Obtiene el número total de procesos
12    MPI_Comm_size(MPI_COMM_WORLD, &size);
13
14    // Obtiene el identificador del proceso actual
15    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
16
17    int total_elementos = 8;
18    int datos[8] = {1, 2, 3, 4, 5, 6, 7, 8};
19
20    int elementos_por_proceso = total_elementos / size;
21    int* buffer_local = (int*)malloc(sizeof(int) * elementos_por_proceso);
22
23    // Distribuye partes iguales del arreglo a cada proceso
24    MPI_Scatter(
25        datos,                      // Datos de entrada (en proceso raíz)
26        elementos_por_proceso,      // Cuántos envía a cada proceso
27        MPI_INT,                   // Tipo de dato
28        buffer_local,              // Buffer donde cada proceso recibe
29        elementos_por_proceso,      // Cuántos recibe
30        MPI_INT,                   // Tipo
31        0,                         // Proceso raíz
32        MPI_COMM_WORLD             // Comunicador
33    );
34
```

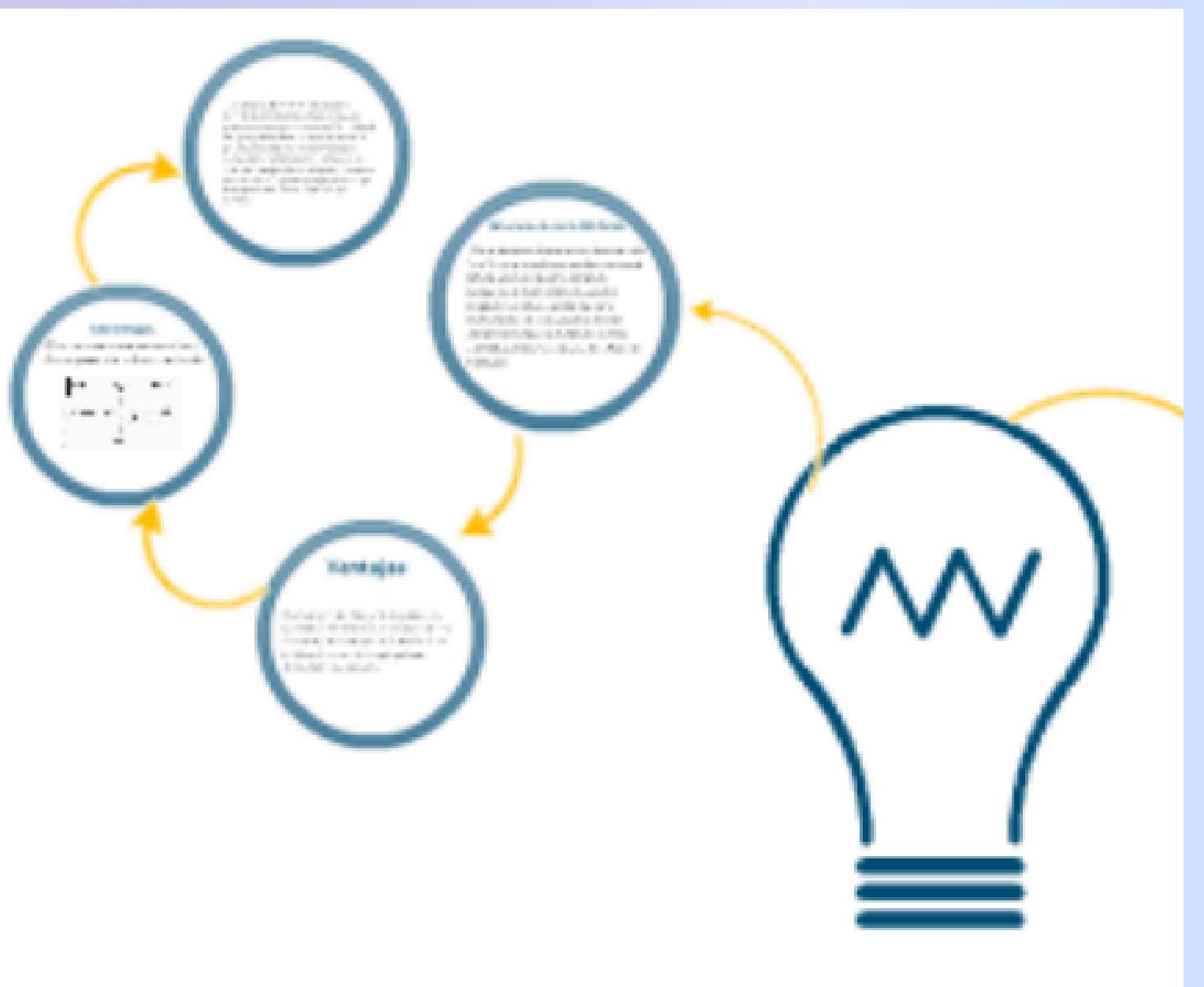
```
34
35    // Cada proceso calcula la suma parcial de su bloque
36    int suma_local = 0;
37    for (int i = 0; i < elementos_por_proceso; i++) {
38        suma_local += buffer_local[i];
39    }
40
41    // Proceso raíz recoge las sumas parciales
42    int suma_total = 0;
43    MPI_Reduce(
44        &suma_local, // Entrada local
45        &suma_total, // Resultado (solo válido en proceso raíz)
46        1,           // Número de elementos
47        MPI_INT,     // Tipo de dato
48        MPI_SUM,     // Operación de reducción
49        0,           // Proceso raíz
50        MPI_COMM_WORLD
51    );
52
53    // Proceso raíz muestra el resultado final
54    if (rank == 0) {
55        printf("Suma total: %d\n", suma_total);
56    }
57
58    // Limpieza
59    free(buffer_local);
60    MPI_Finalize();
61    return 0;
62}
63
```



Memoria compartida



4.3 SISTEMAS DE MEMORIA COMPARTIDA



Un multiprocesador puede verse como un computador paralelo compuesto por varios procesadores interconectados que comparten un mismo sistema de memoria.

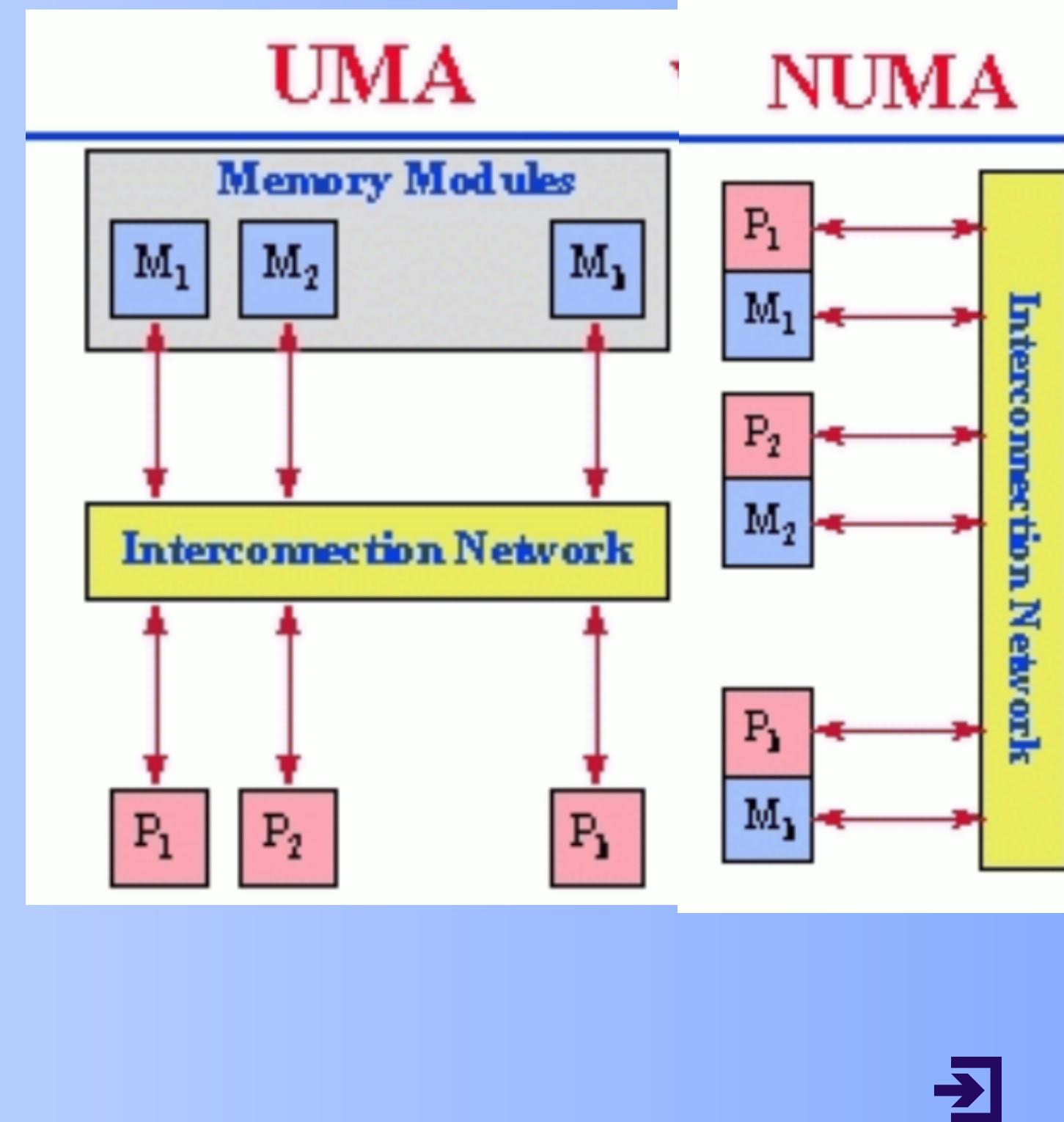
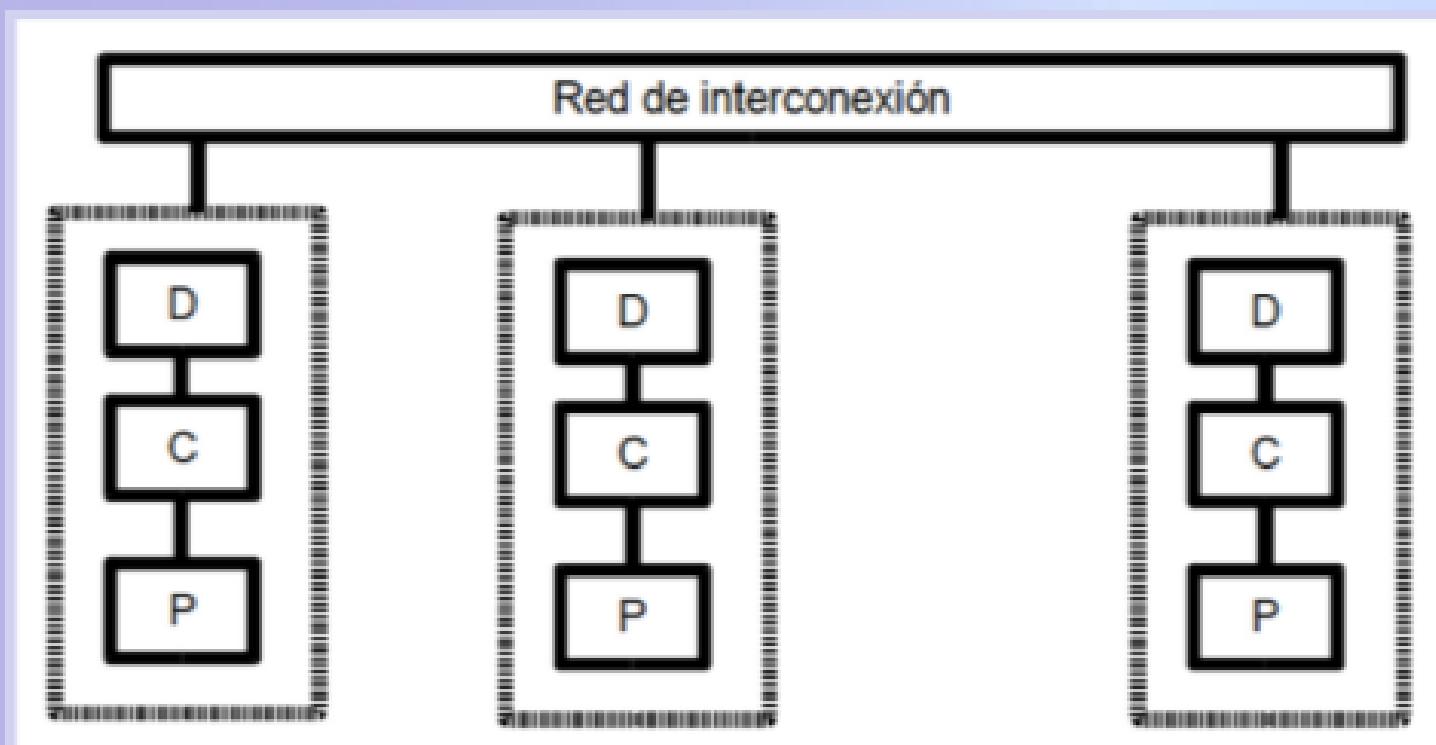
Los sistemas multiprocesadores son arquitecturas MIMD con memoria compartida.



SE CLASIFICAN EN:

dependiendo de la forma en la que los procesadores comparten la memoria se clasifican en sistemas multi-procesador “UMA” , “NUMA” y “COMA”.

el multi proceso para tareas generales, es a veces muy dificil de conseguir, debido a que pueden haber varios programas manejando datos internos, conocido como “estado” o “contexto”





4.3.1 REDES DE INTERCONEXIÓN DINÁMICAS Ó INDIRECTAS

Las redes de interconexión indirectas son aquellas en las que los procesadores no están conectados directamente entre sí, sino a través de una serie de conmutadores intermedios. Estas redes permiten establecer rutas entre múltiples nodos a través de diversos caminos posibles.

Redes dinámicas:

El término "dinámicas" se refiere a la capacidad de estas redes de cambiar las rutas de comunicación entre nodos durante la operación, lo cual permite adaptarse a fallos, congestión o necesidades específicas del sistema.

Características principales:

- Escalables para muchos procesadores.
- Soportan múltiples rutas entre origen y destino.
- Utilizan técnicas de conmutación como conmutación de paquetes o circuitos.

4.3.1 COMPARACION DE REDES DINÁMICAS Ó INDIRECTAS

| Característica | Redes Directas | Redes Indirectas |
|-------------------------|------------------------------|-----------------------------------|
| Tipo de conexión | Conexión directa entre nodos | Conexión a través de conmutadores |
| Topologías comunes | Malla, anillo, hipercubo | Árboles, MNs, Clos, Benes |
| Ruta de comunicación | Estática | Dinámica / Reconfigurable |
| Escalabilidad | Limitada | Alta |
| Costo | Menor (menos complejidad) | Mayor (más hardware) |
| Flexibilidad de tráfico | Limitada | Alta |

4.3.1 FUNCIONAMIENTO DE LAS REDES INDIRECTAS Y DINÁMICAS

Estructura:

- Constan de niveles de conmutadores.
- Los nodos finales (procesadores o memorias) se conectan en los extremos.
- El tráfico fluye por rutas seleccionadas dinámicamente.

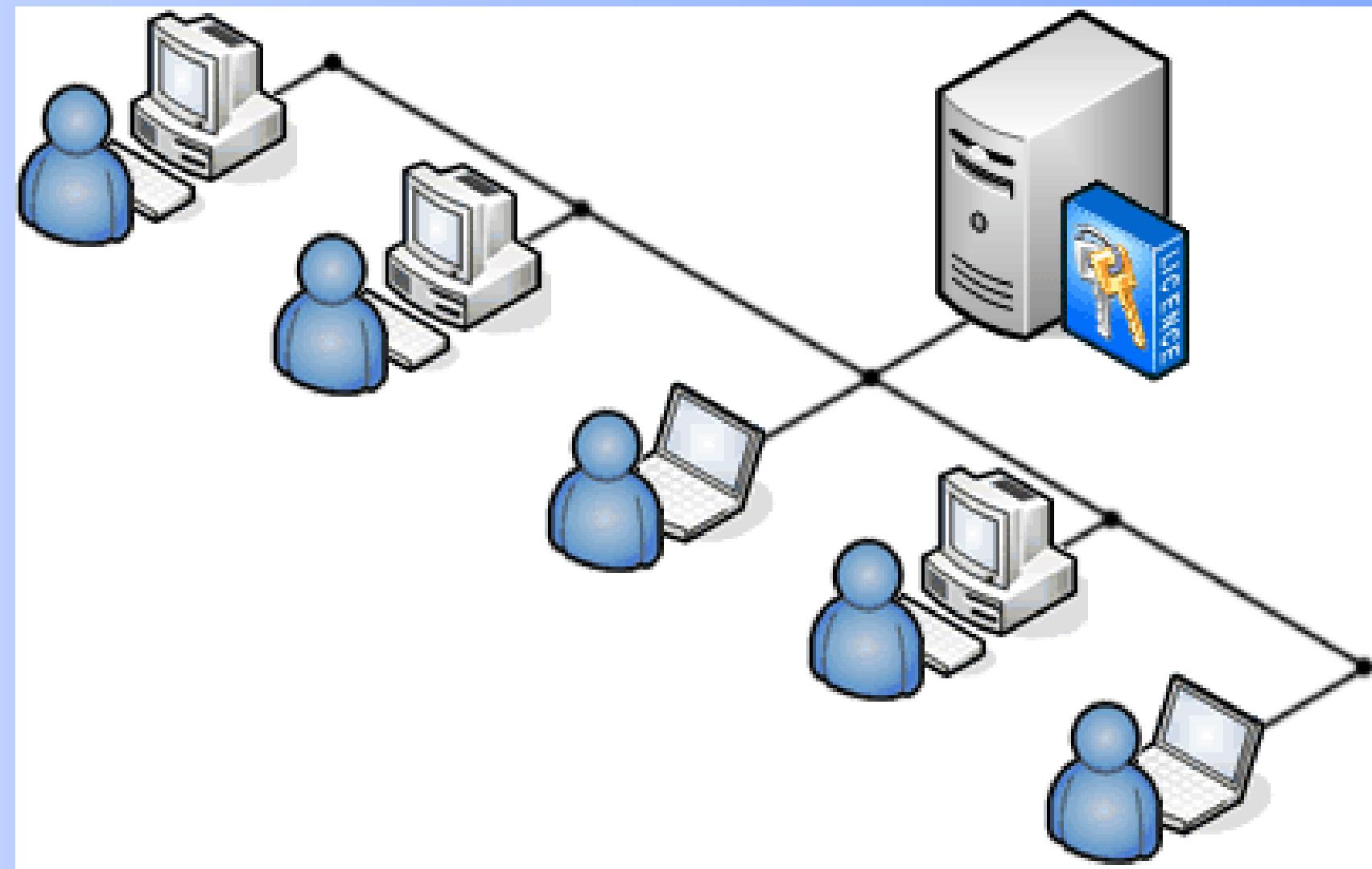
Proceso de comunicación:

1. El nodo origen envía un mensaje.
2. El mensaje se enruta por conmutadores intermedios.
3. Puede tomar diferentes caminos según disponibilidad o algoritmo de enrutamiento.
4. Llega al nodo destino.

4.3.1.1) REDES DE MEDIO COMPARTIDO

Es un mecanismo que permite que varios procesos (o hilos) accedan y usen una misma región de memoria.

Esto significa que todos los dispositivos "escuchan" todas las transmisiones y solo responden a aquellas dirigidas a ellos. Este tipo de red es relativamente simple de implementar, pero puede tener problemas de rendimiento cuando hay mucho tráfico.





EJEMPLOS

- Ethernet compartido: En este caso, todos los dispositivos se conectan a un concentrador o hub, y el tráfico se transmite a todos los dispositivos conectados.
- Wireless (Wi-Fi): En redes Wi-Fi, los dispositivos comparten el mismo canal de radio para comunicarse.



VENTAJAS Y DESVENTAJAS

VENTAJAS

- Simplicidad de implementación: Las redes de medio compartido son relativamente fáciles de configurar y mantener.
- Coste reducido: Requieren menos hardware y software que otros tipos de redes.

DESVENTAJAS

- Rendimiento limitado: El rendimiento de la red puede verse afectado por la cantidad de tráfico y la presencia de colisiones.
- Problemas de seguridad: Al compartir el mismo medio, las redes de medio compartido son más vulnerables a la escucha y a la manipulación de datos.

4.3.1.2) REDES COMMUTADAS



Son un tipo de red usada para interconectar varios procesadores o nodos en una arquitectura paralela. En la arquitectura paralela, los procesadores tienen que comunicarse entre sí para coordinarse y compartir datos. Entonces se necesita una estructura eficiente para:

- Enviar mensajes.
- Compartir datos.
- Coordinar tareas.

Esa estructura de comunicación es la red, y cuando es "comutada", significa que usa interruptores o caminos dinámicos para dirigir los mensajes hacia donde deben ir.

COMUTACIÓN

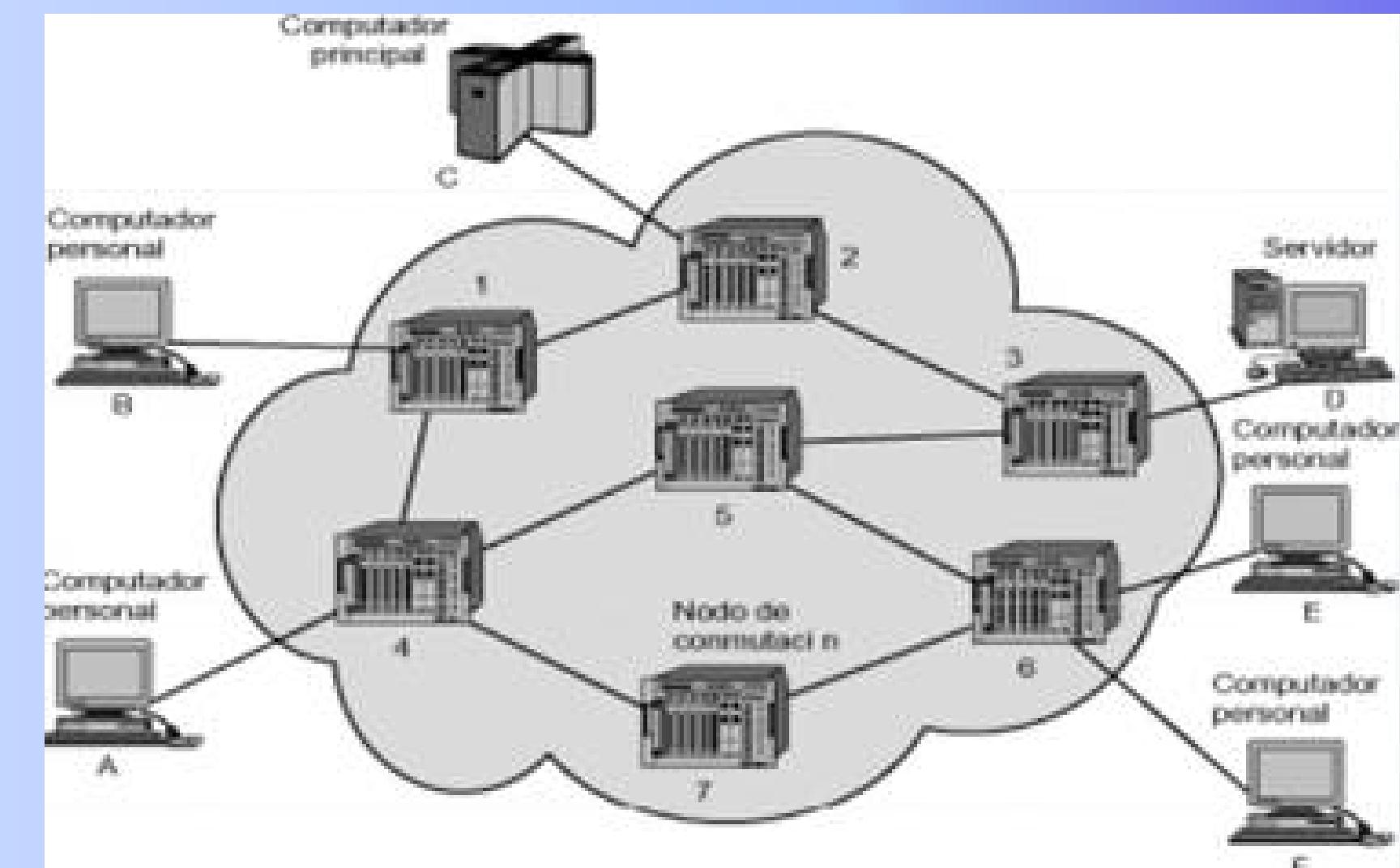
Comutar significa **cambiar de camino o ruta**.

Una red conmutada permite que los datos tomen diferentes rutas, dependiendo de:

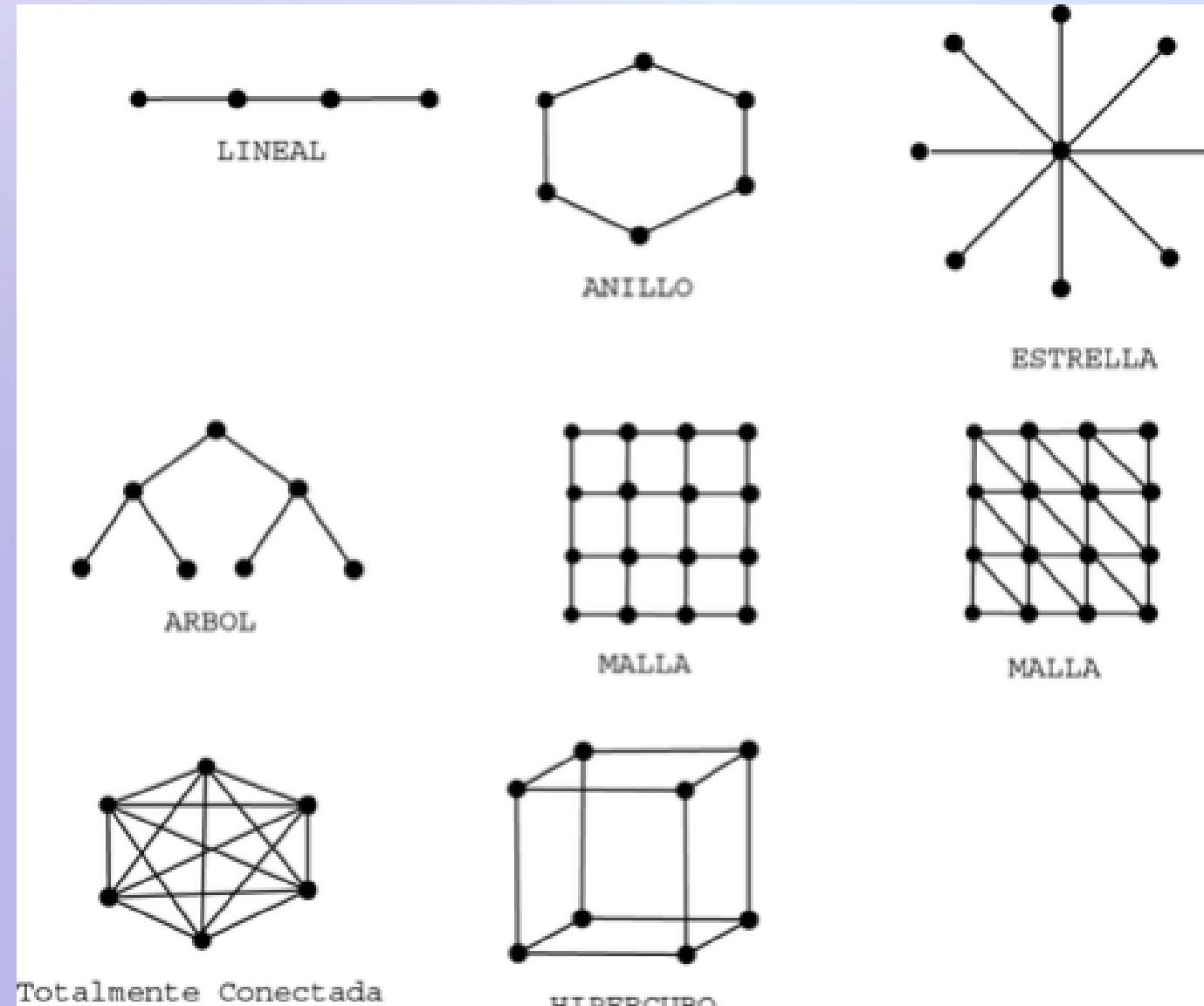
- El destino del mensaje.
- La congestión del camino.
- Las decisiones del conmutador (como si fuera un semáforo inteligente).

ELEMENTOS DE UNA RED CONMUTADA

- **Nodos:** los procesadores o unidades que se comunican.
- **Conmutadores:** dispositivos que dirigen los datos por el mejor camino.
- **Caminos posibles:** conexiones entre nodos que pueden activarse o desactivarse según sea necesario.



TIPOS DE REDES COMUTADAS



Red de conmutación en anillo

- Cada nodo está conectado con dos vecinos (como una rueda).
- Los datos viajan en una dirección hasta llegar a su destino.
- Simple.
- Puede tardar más si el destino está lejos.

Red de conmutación en malla (mesh)

- Cada nodo se conecta a varios vecinos (como una cuadrícula).
- Hay múltiples caminos posibles.
- Más rutas = más rapidez.
- Más compleja de gestionar.

Red tipo hipercubo

- Conexión en forma de cubo tridimensional (o más dimensiones).
- Muy eficiente para arquitecturas grandes.
- Ideal para muchos nodos.
- Difícil de implementar físicamente.

TIPOS DE CONMUTACIÓN

CONMUTACIÓN DE CIRCUITOS

Se establece un camino completo antes de enviar los datos. Es estable pero puede ser ineficiente.

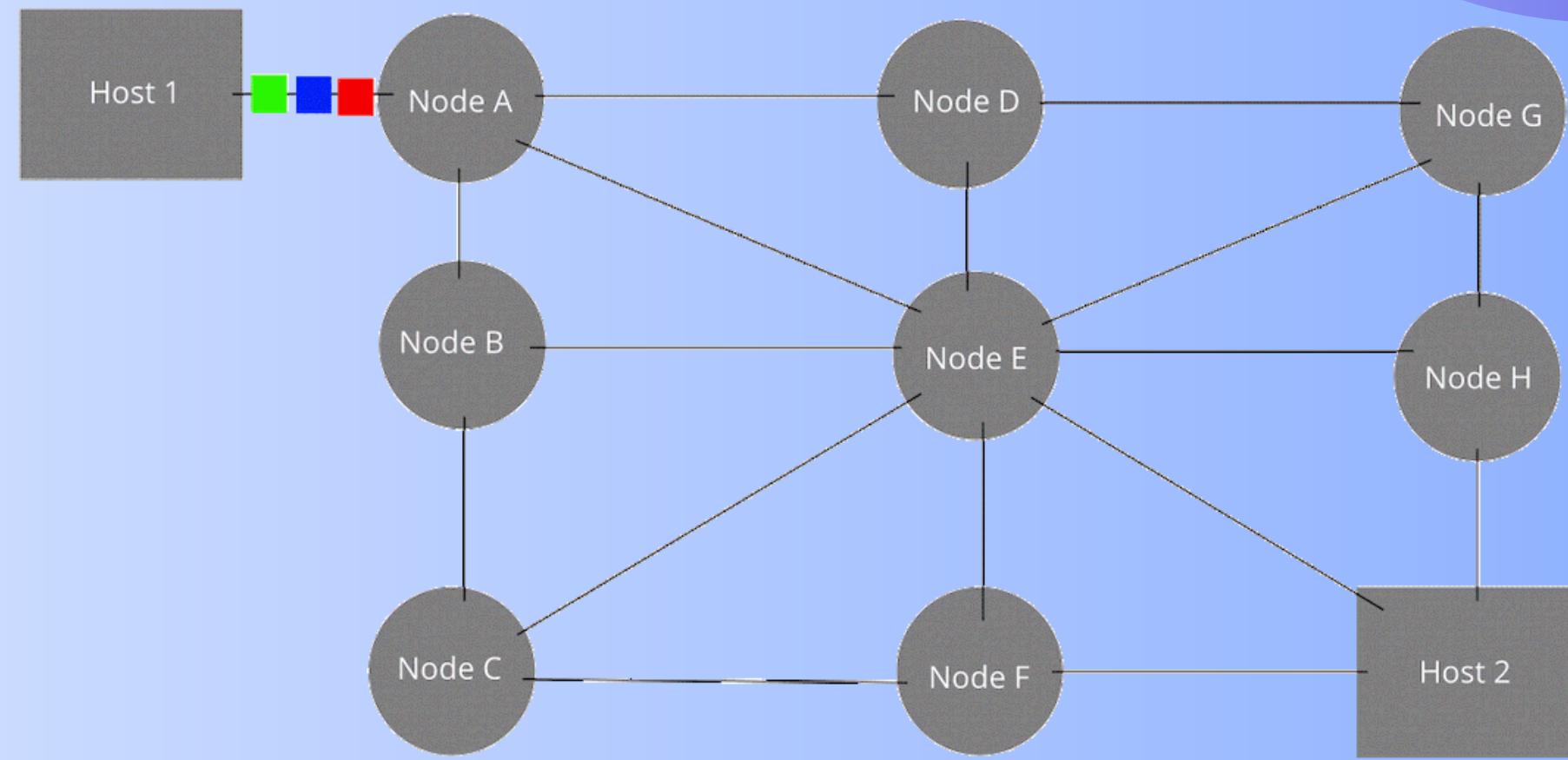
CONMUTACIÓN DE PAQUETES

Los datos se dividen en pequeños paquetes, cada uno puede tomar su propio camino. Es flexible y eficiente, pero puede causar retrasos y hay muchos datos viajando.

CONMUTACIÓN DE MENSAJES

El mensaje completo se envía de un nodo al siguiente, uno por uno. Es fácil de entender e implementar pero es lento si hay muchos mensajes grandes.

The original message is Green, Blue, Red.



VENTAJAS DE USAR REDES CONMUTADAS EN SISTEMAS PARALELOS

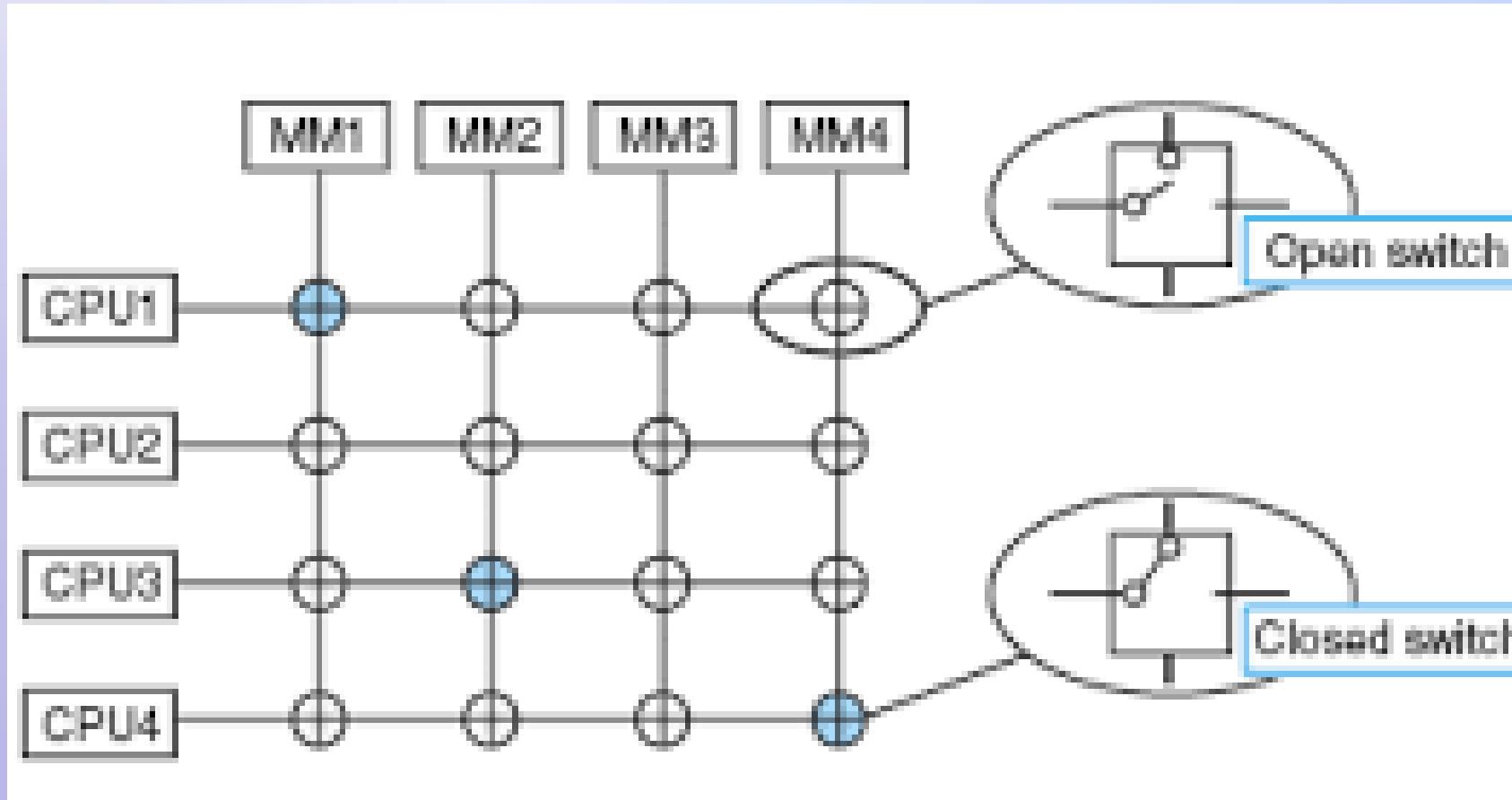


FIGURE 9.5 A Crossbar Network

ESCALABILIDAD

Se pueden agregar más nodos sin afectar tanto el sistema.

FLEXIBILIDAD

Los datos pueden tomar múltiples caminos.

EFICIENCIA

si se diseña bien, se pueden evitar cuellos de botella.

CONFIABILIDAD

Si un camino falla, se puede usar otro.

Las redes conmutadas no son WiFi o Internet comunes, sino **redes internas** que conectan los procesadores o nodos dentro de una arquitectura paralela. Su buen diseño es clave para el rendimiento de supercomputadoras y clústeres.

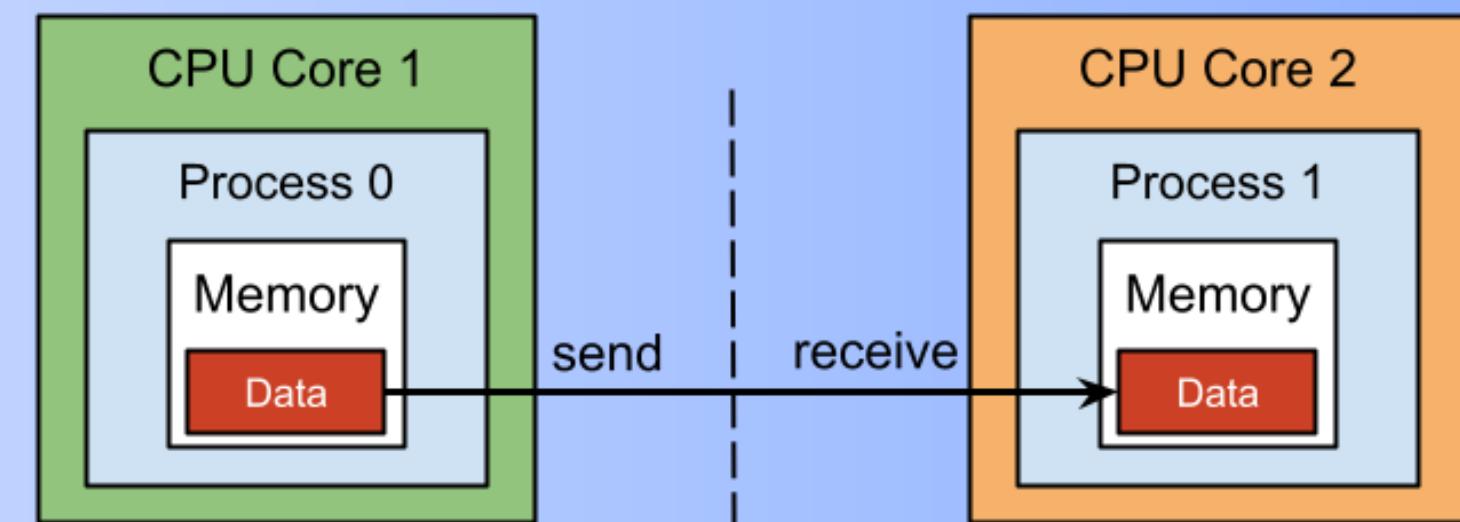
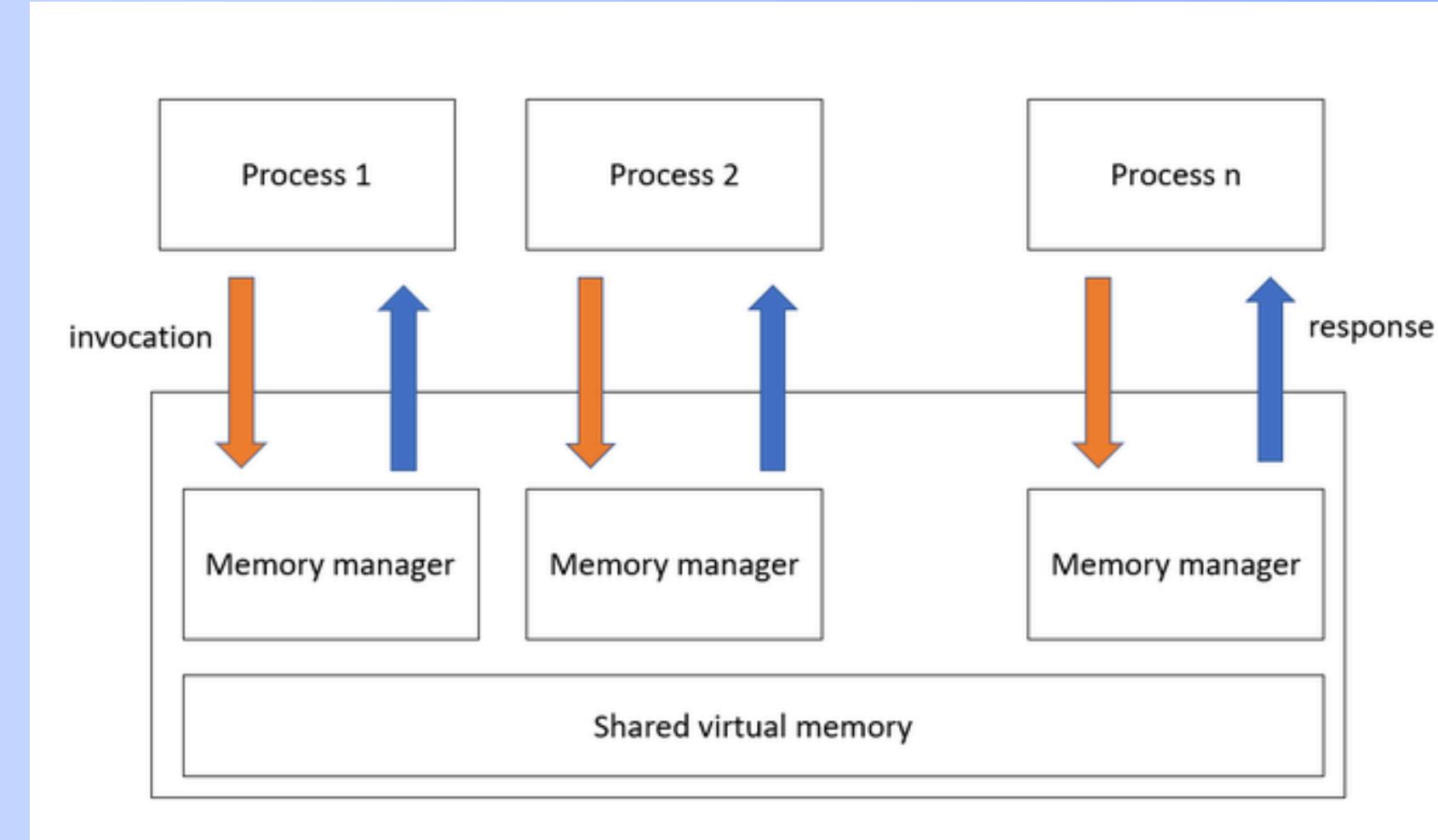
4.4

Sistemas de Memoria Distribuida: Multiprocesadores

MEMORIA DISTRIBUIDA (DISTRIBUTED MEMORY)

Un sistema de memoria distribuida es una arquitectura donde cada procesador (CPU o nodo) tiene su propia memoria local privada, y no hay una memoria compartida centralizada como en los sistemas UMA o NUMA.

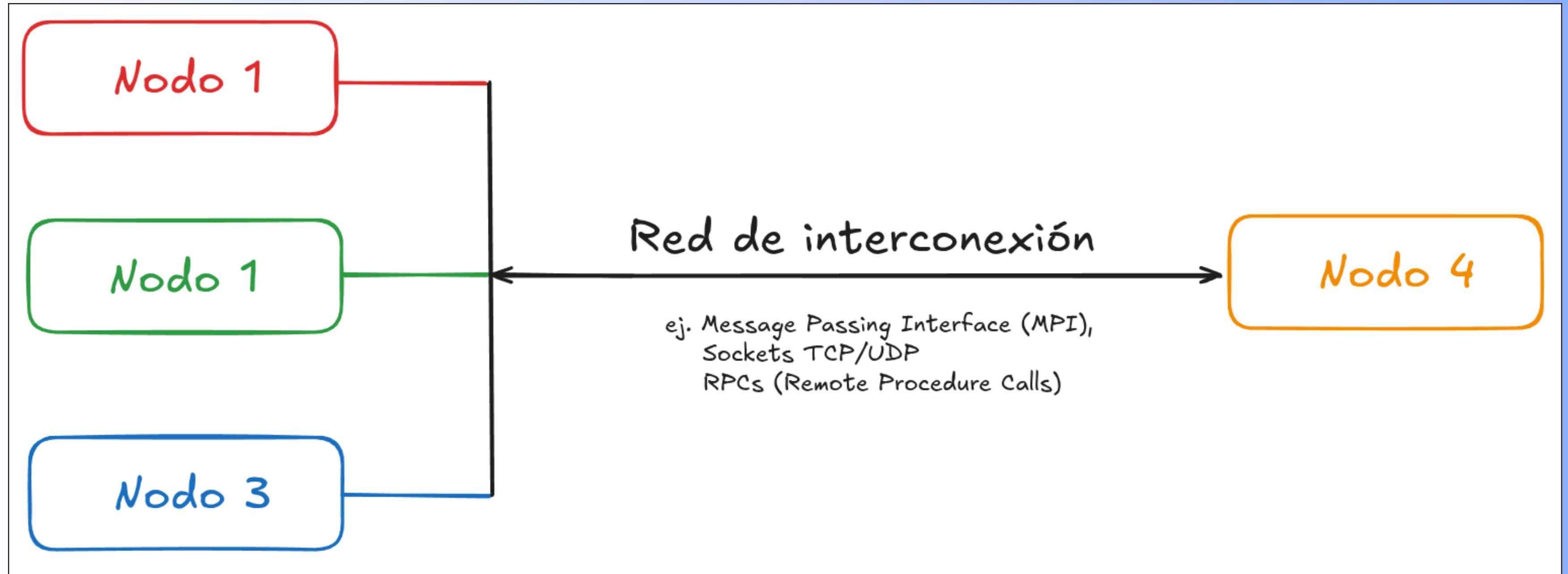
En lugar de acceder directamente a la memoria de otros procesadores, estos sistemas se comunican a través de mensajes. Por eso, también se les llama sistemas de paso de mensajes.



SISTEMAS DE MEMORIA DISTRIBUIDA

Cada nodo tiene:

- CPU(s)
 - Memoria RAM privada
 - Sistema operativo (puede ser o no compartido)
-
- No existe una memoria global
 - La única forma de comunicación es mediante el envío de mensajes (como sockets, MPI, RPC, etc.)
 - Escalabilidad muy alta (usado en supercomputadoras y clústeres grandes)
 - Latencia de comunicación alta en comparación con accesos a memoria local

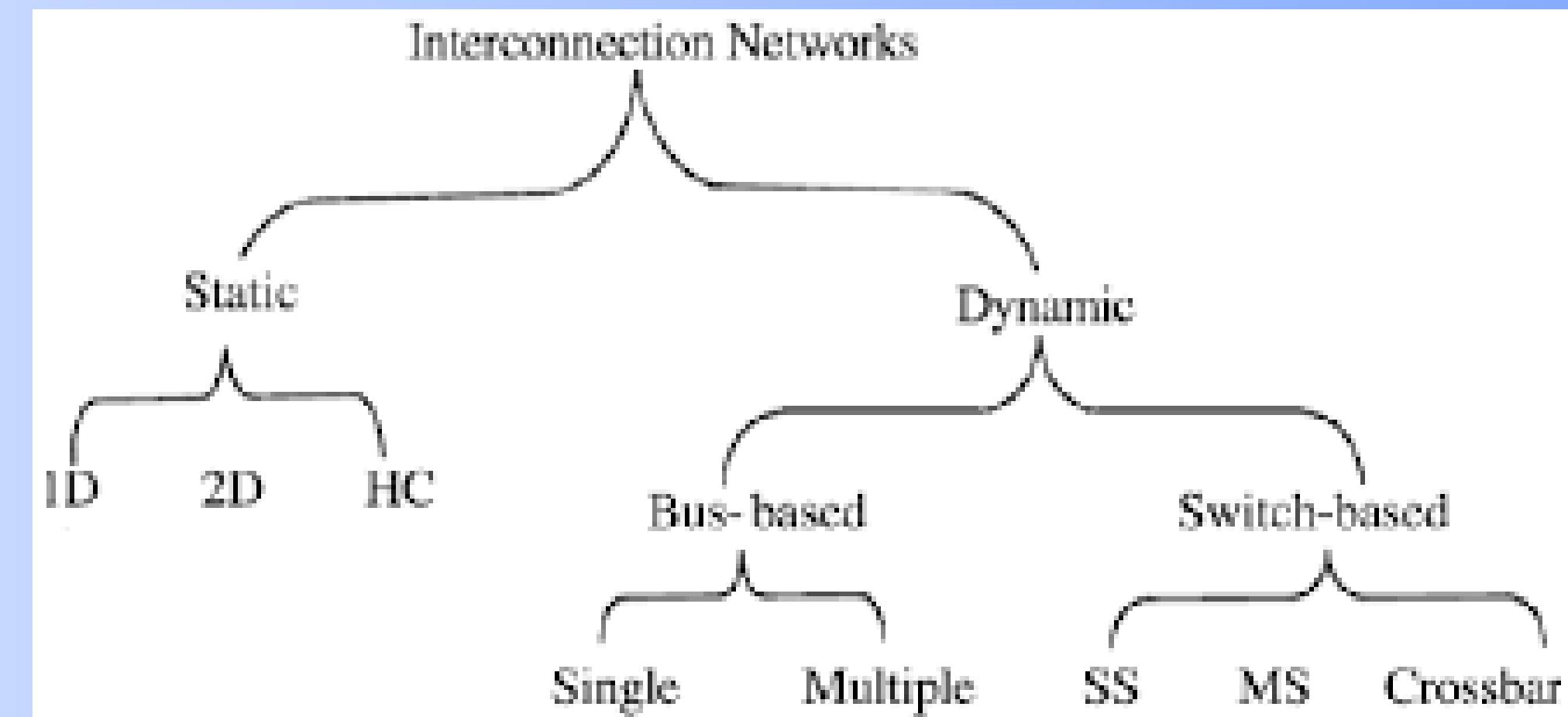


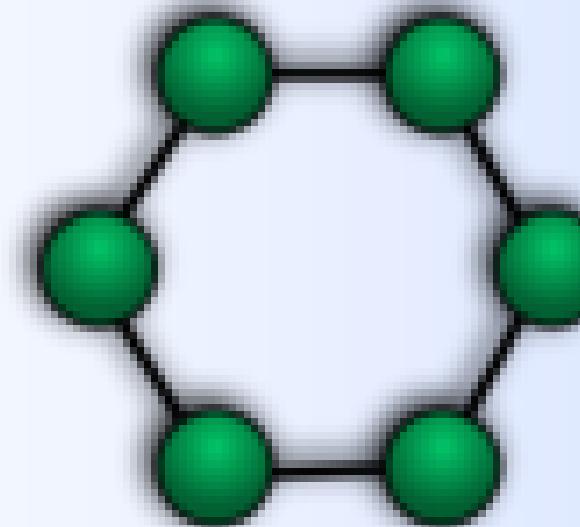
4.4.1

Redes de interconexión estáticas

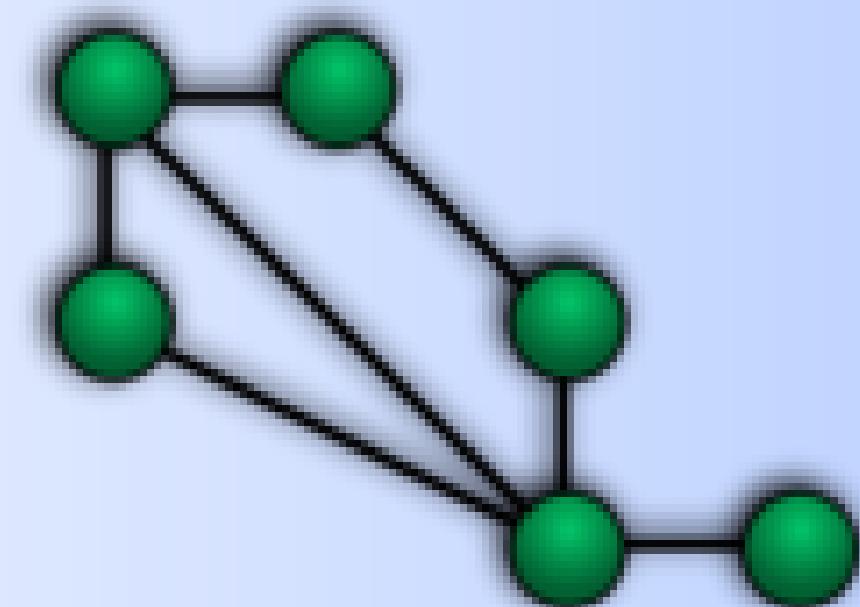
Una red de interconexión define cómo los procesadores o nodos están conectados entre sí para comunicarse en sistemas paralelos. En una red estática, la conectividad no cambia durante el tiempo de ejecución: está predefinida en la arquitectura del sistema.

En lugar de que todos estén conectados a todos (como en una red dinámica), aquí hay un patrón fijo.

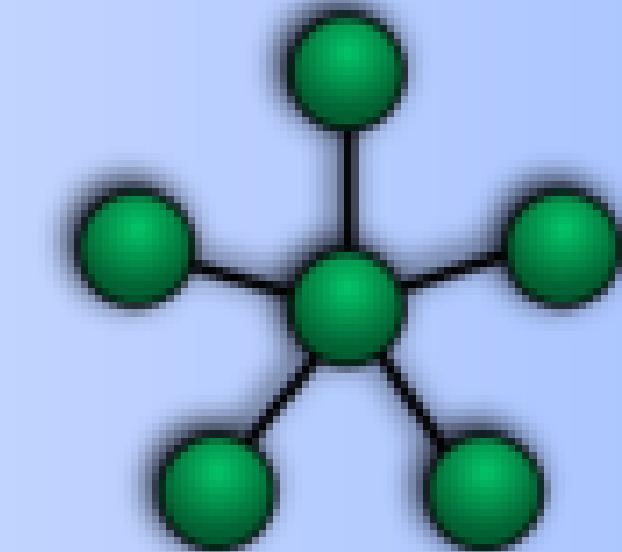




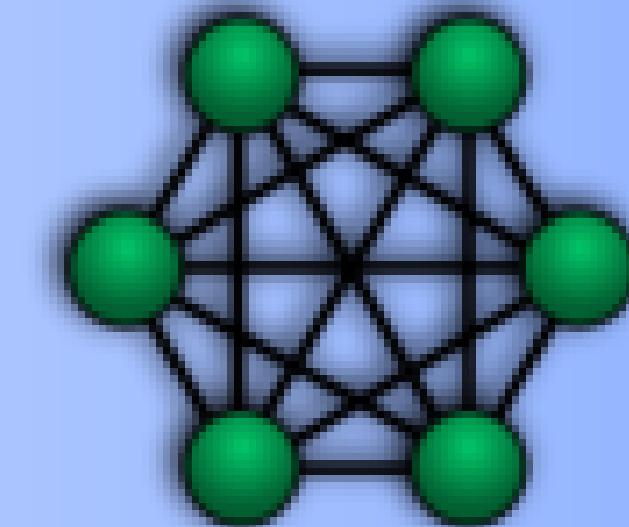
Ring



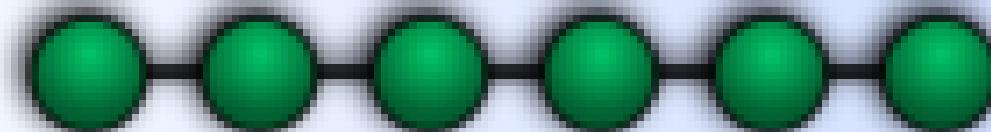
Mesh



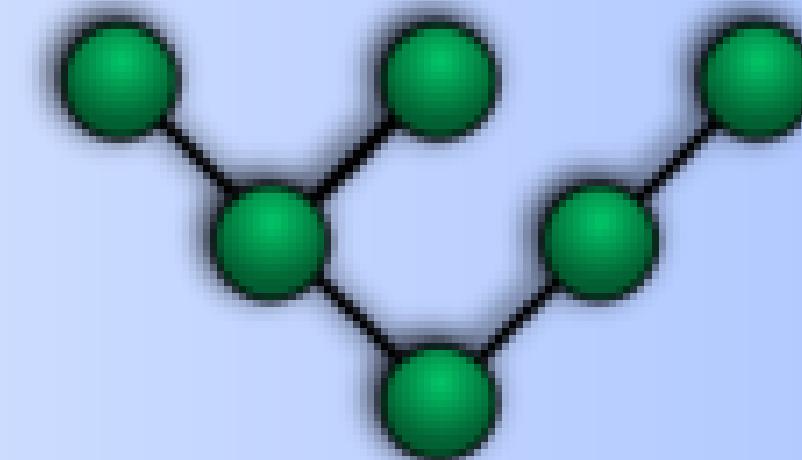
Star



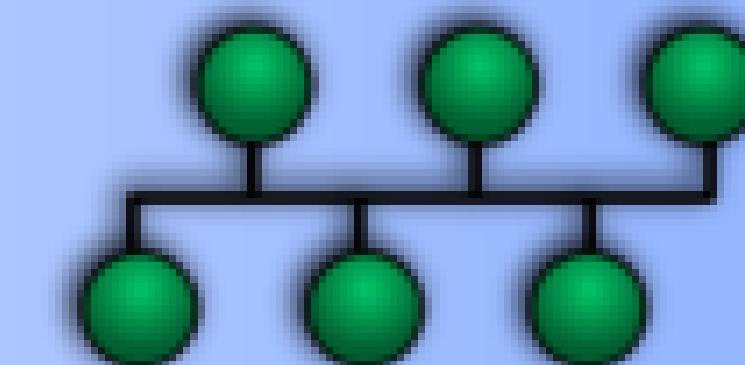
Fully Connected



Line



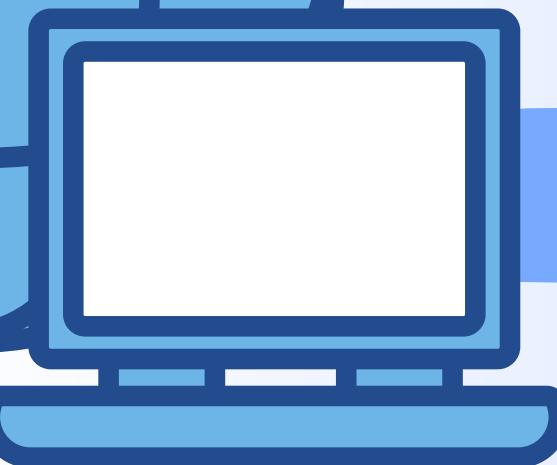
Tree



Bus

4.5

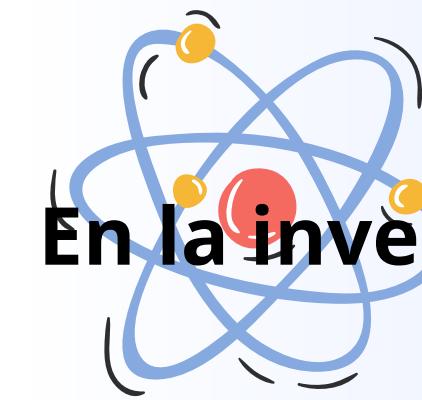
Casos de estudio



Casos de estudio

La computación paralela, que involucra dividir problemas en partes que se resuelven simultáneamente, se aplica en diversos campos. Un ejemplo es el procesamiento de grandes cantidades de datos en tiempo real, como en aplicaciones de aprendizaje automático o análisis de tráfico de red. También se utiliza para simular fenómenos complejos, como el clima o el flujo de fluidos, donde la capacidad de procesamiento paralelo es crucial para obtener resultados precisos y rápidos.

Ejemplos



Ciencia y tecnología médica:

En la investigación médica, la computación paralela se utiliza para analizar imágenes médicas, modelar enfermedades y desarrollar nuevos tratamientos.

Diseño y fabricación:

La computación paralela se utiliza en la industria para simular procesos de fabricación, optimizar diseños de productos y realizar pruebas de rendimiento, lo que permite ahorrar tiempo y recursos.



Inteligencia Artificial y aprendizaje automático:

Los algoritmos de aprendizaje automático, que requieren procesar grandes cantidades de datos, se benefician enormemente de la computación paralela para entrenar modelos y realizar predicciones en tiempo real.



Beneficios

Aumento de la velocidad de procesamiento: La computación paralela permite realizar cálculos más rápido, lo que es especialmente importante para tareas que requieren un procesamiento intensivo.


Ahorro de tiempo y recursos: Al acelerar el procesamiento, la computación paralela permite ahorrar tiempo y recursos, lo que es crucial para empresas y organizaciones que trabajan con grandes volúmenes de datos o que requieren realizar cálculos complejos.


Resolución de problemas complejos: La computación paralela permite resolver problemas complejos que serían imposibles o muy difíciles de abordar con computación secuencial.


Aprovechamiento de recursos: La computación paralela permite aprovechar al máximo los recursos informáticos, lo que es especialmente importante en entornos con gran cantidad de servidores o núcleos de CPU.

GRACIAS