

Sesión teórica

Debe incluir, como mínimo, lo siguiente:

- Filosofía del paradigma
- Conceptos claves
- Ventajas y desventajas
- Lenguajes de programación
- Ejemplos en distintos lenguajes. En lo posible usar IPython NoteBooks para publicar los ejemplos online (ver <https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages>).
- Aplicaciones de este paradigma: dominios de aplicación donde es más común encontrar aplicaciones desarrolladas bajo este paradigma, ejemplos de programas conocidos, etc.
- Referencias/Bibliografía
- **Además, cada grupo debe proponer un taller con mínimo 10 preguntas conceptuales de selección múltiple con única/múltiple respuesta (nivel de dificultad: medio-alto). Las soluciones deben ser entregadas al equipo docente.**

	0.0	1.0	2.0	3.0	3.5	4.0	4.5	5.0
Manejo del tema	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Claridad exposición	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Trabajo en equipo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Compleitud	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Materiales didácticos	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Manejo tiempo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Ejemplos adecuados	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Calificación general	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Lenguajes de programación

Programación Paralela

Santiago Peña

Camilo Mosquera

CONTENIDO

- Introducción/Filosofía
- Historia
- Ventajas y desventajas
- Conceptos clave
- Taxonomía de Flynn
- Arquitectura
- Sincronización
- Balanceador de carga
- Tipos de paralelismo
- Lenguajes de programación



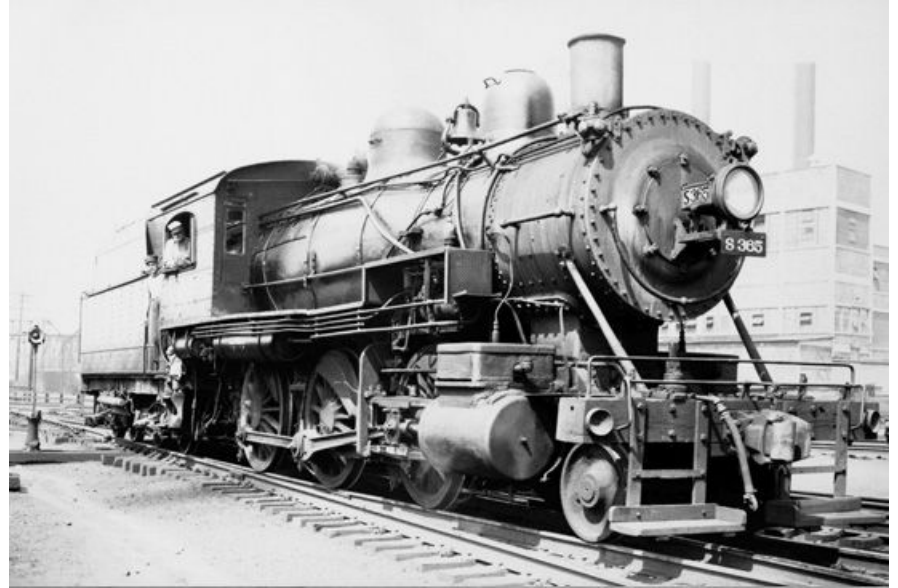
Introducción

La necesidad de resolver problemas de manera eficiente y en el menor tiempo posible ha hecho que se tengan que realizar mejoras en el poder computacional de los computadores.

Historia

Se tomaron términos propios de:
ferrocarriles
telegrafía

Fueron importantes los estudios de Dijkstra quien se le acredita el primer trabajo en este campo, identificando y resolviendo la exclusión mutua



Supercomputadoras

Interés en las supercomputadoras en los años 50, y años 60 y 70 se se tienen computadores tenían múltiples procesadores de memoria compartida.



Década de los 80

El proyecto “Concurrent Computation” del Instituto de Tecnología de California logró un rendimiento extremo usando microprocesadores regulares

Los clusters surgieron para competir y los MPP



Década de los 90

Estándar MPI: define la sintaxis y la semántica de las funciones contenidas en una [biblioteca](#) de paso de mensajes

Surgimiento de pthreads y OpenMP

Actualidad

Aparición de procesadores con varios núcleos



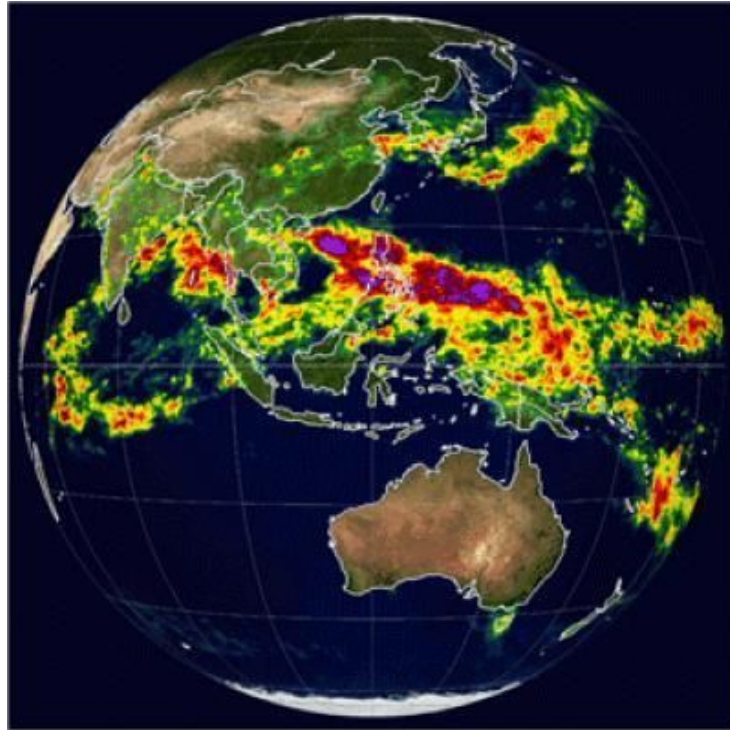


¿Qué problemas se
necesitaban resolver?


Biocomputación - Encontrar secuencias de ADN de varios organismos

ATCTCTTGGCTCCAGCATCGATGAAGAACGCA
TCATTTAGAGGAAGTAAAAGTCGTAACAAGGT
GAACGTCAAAACTTTTAACAACGGATCTCTT
TGTTGCTTCGGCGGGCGCCCGCAAGGGTGCCCG
GGCCTGCCGTGGCAGATCCCCAACGCCGGGCC
TCTCTTGGCTCCAGCATCGATGAAGAACGCAG
CAGCATCGATGAAGAACGCAGCGAAACGCGAT
CGATACTTCTGAGTGTTCTTAGCGAACTGTCA
CGGATCTCTTGGCTCCAGCATCGATGAAGAAC
ACAACGGATCTCTTGGCTCCAGCATCGATGAA
CGGATCTCTTGGCTCCAGCATCGATGAAGAAC
GATGAAGAACGCAGCGAAACGCGATATGTAAT


Predicción meteorológica



3



Entonces... ¿Qué es la
programación paralela y cómo
puede solucionar estos
problemas?



La computación paralela es forma de computación en la cual muchos cálculos son llevados de forma simultánea.



Con el uso de varios procesadores trabajando juntos en una tarea común.

Estos procesadores pueden trabajar de dos maneras



Cada procesador trabaja en un pedazo del problema



Los procesos pueden intercambiar datos entre ellos.

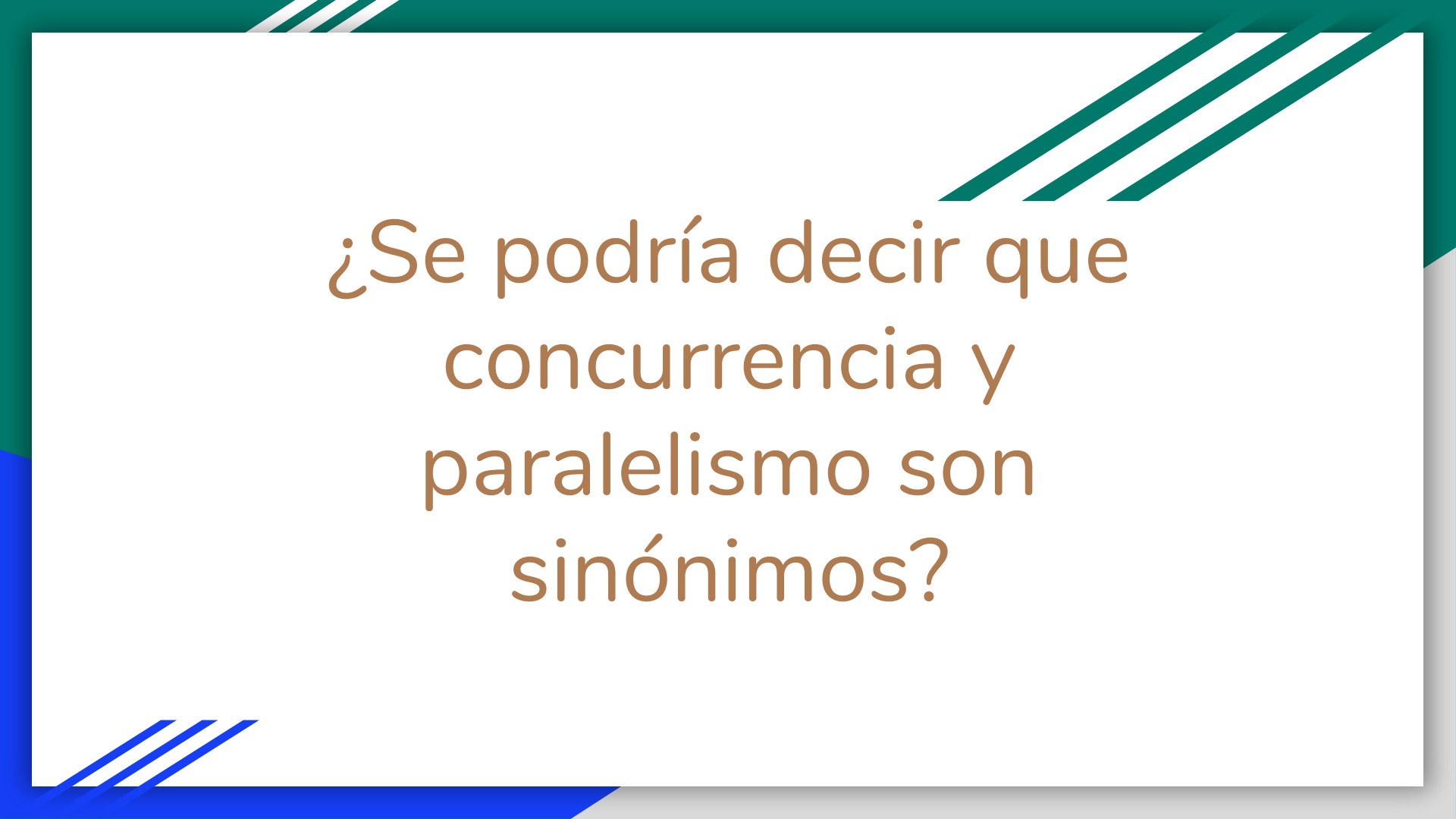
Concurrencia vs Paralelismo

¿Qué es concurrencia?

Capacidad de operar actividades al mismo tiempo.

¿Qué es Paralelismo?

Son muchas actividades teniendo lugar al mismo tiempo, *“la cualidad o el estado de ser paralelo”*

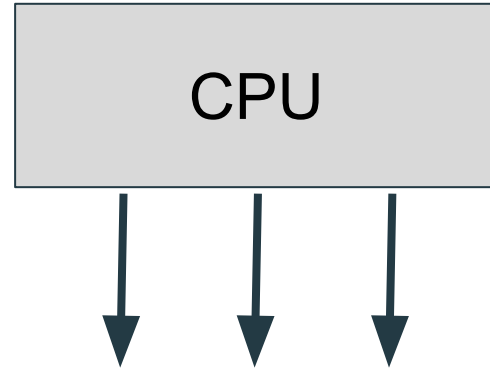
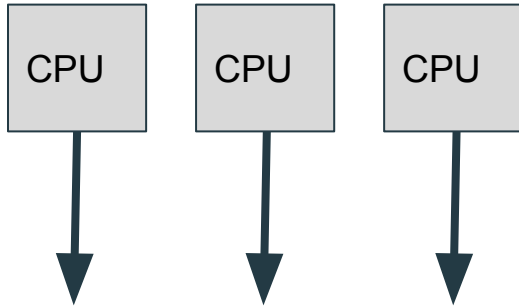


¿Se podría decir que
conurrencia y
paralelismo son
sinónimos?

Si... Pero no.

Concurrencia

Paralelismo





Filosofía del paradigma

Filosofía del paradigma

- Se realiza una ejecución en múltiples procesadores
- Cada parte de la ejecución se descompone en una serie de instrucciones que se pueden ejecutar simultáneamente
- Se emplea un mecanismo global de coordinación



Ventajas y desventajas

VENTAJAS

- Ahorro en tiempo y/o dinero
- Se pueden solucionar problemas muy grandes que no es posible resolver mediante programación secuencial
- Se mejora la eficiencia al dividir el problema en tareas más pequeñas.

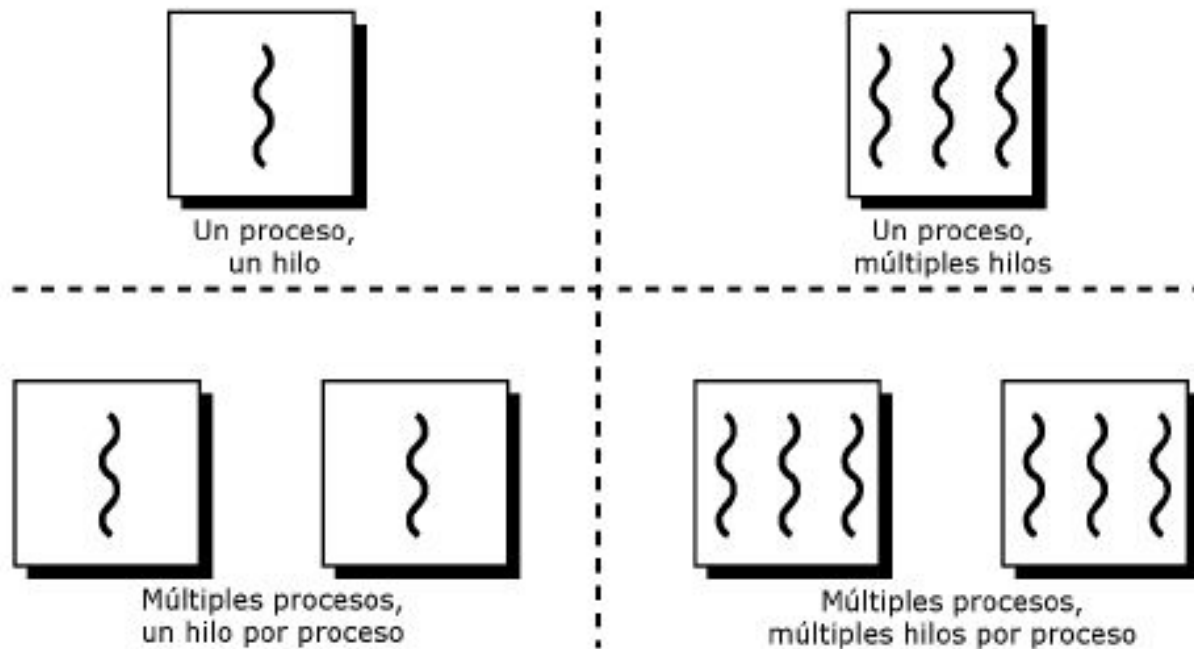
DESVENTAJAS

- Encontrar la solución a un problema tiene una complejidad mayor
- Se utilizan más recursos de la máquina
- Pueden existir problemas de sincronización entre los diferentes procesos que se estén llevando a cabo.



Conceptos clave

Hilo:



Tareas y Granularidad

El tamaño de cada tarea, en término del número de instrucciones. Cada tarea puede tener un tamaño diferente.



SCHEDULING El proceso mediante el cual las tareas son asignadas a los procesos o hilos, y se les da un orden de ejecución.



MAPPING Es la asignación de procesos e hilos a unidades de procesamiento, procesadores o núcleos.

COOPERACIÓN Los procesos están diseñados para trabajar conjuntamente en alguna actividad

SINCRONIZACIÓN procesos o hilos intercambian información



Pipelining Ruptura de una tarea en pasos realizados por diferentes unidades de procesador

Speedup Es un proceso para aumentar el rendimiento entre dos sistemas procesando el mismo problema.



Planificación:

Se refiere al proceso de repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución.

Tipos de paralelismo

A nivel de bit

Se basa en el tamaño de la palabra que es capaz de manejar el procesador

A nivel de tareas

Se le asigna una tarea a cada procesador

A nivel de instrucción

Las instrucciones de un programa pueden reordenarse para luego ser ejecutadas sin que esto afecte el resultado

A nivel de datos

Los datos se asignan a los diferentes procesadores, y estos realizan la misma acción sobre los datos

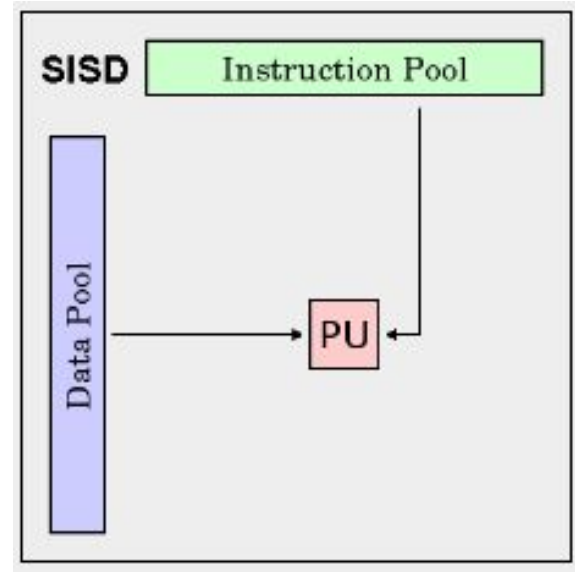
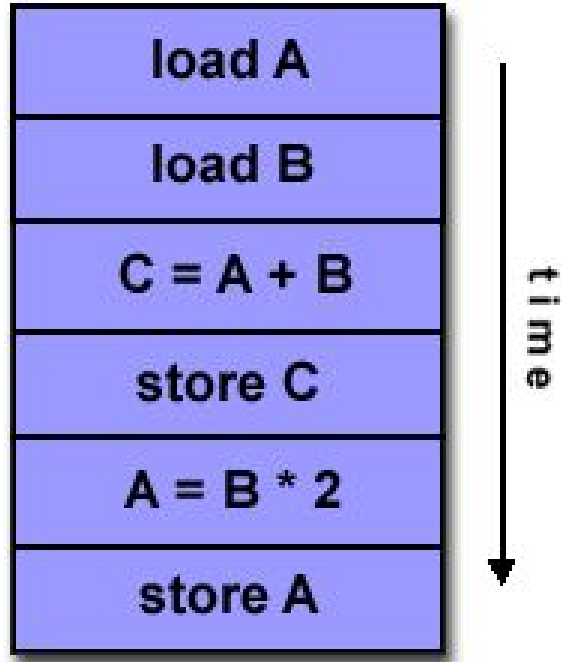


Taxonomía de Flynn

Es una clasificación de las arquitecturas paralelas, en las que se tiene en cuenta el número de instrucciones y el flujo de datos de la arquitectura. Existen 4 clasificaciones:

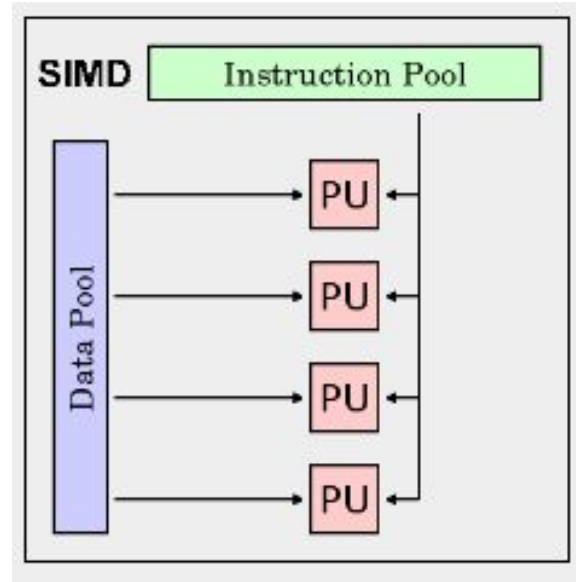
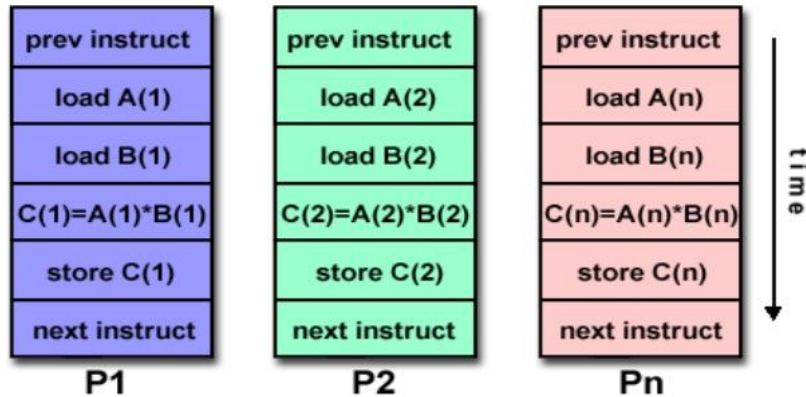
- Single Instruction Single Data (SISD)
- Single Instruction Multiple Data (SIMD)
- Multiple Instructions Single Data (MISD)
- Multiple Instructions Multiple Data (MIMD)

Single Instruction Single Data (SISD)

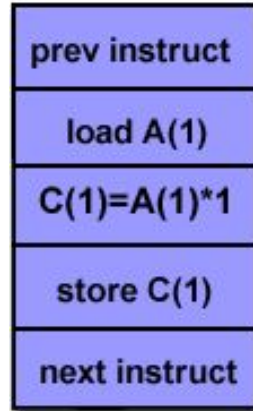
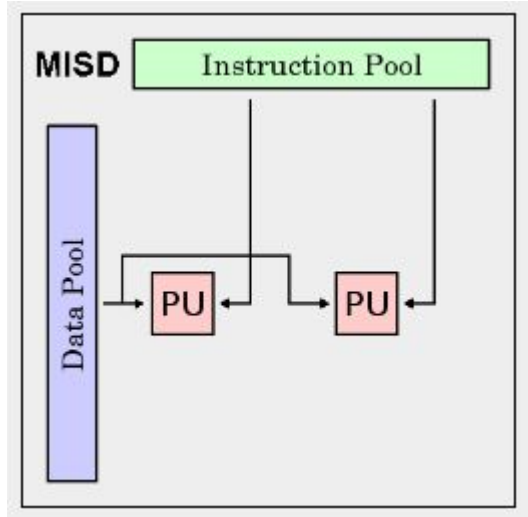


Single Instruction Multiple Data (SIMD)

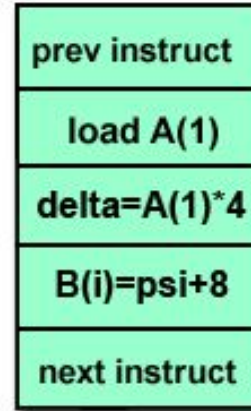
Single Instruction, Multiple Data (SIMD):



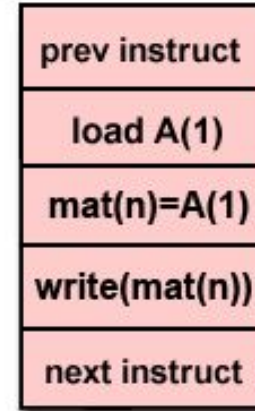
Multiple Instructions Single Data (MISD)



P1



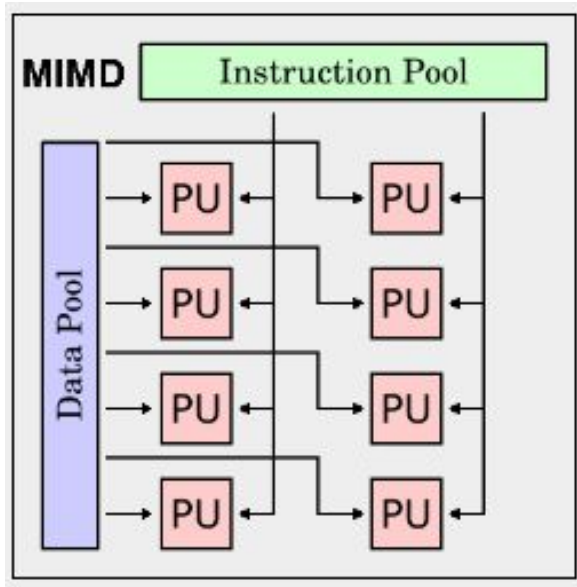
P2



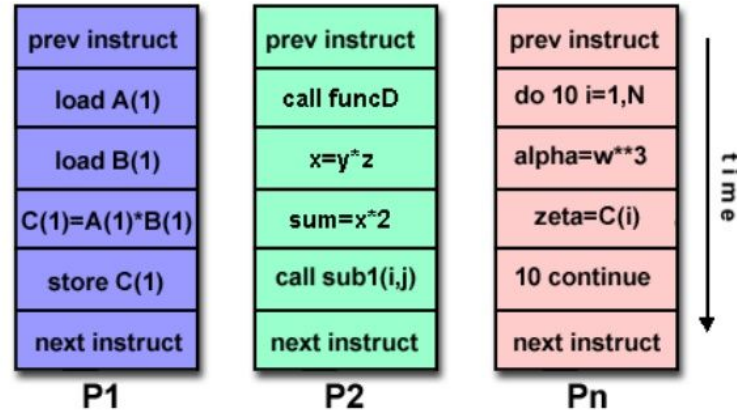
Pn

time

Multiple Instructions Multiple Data (MIMD)



Multiple Instruction, Multiple Data (MIMD):

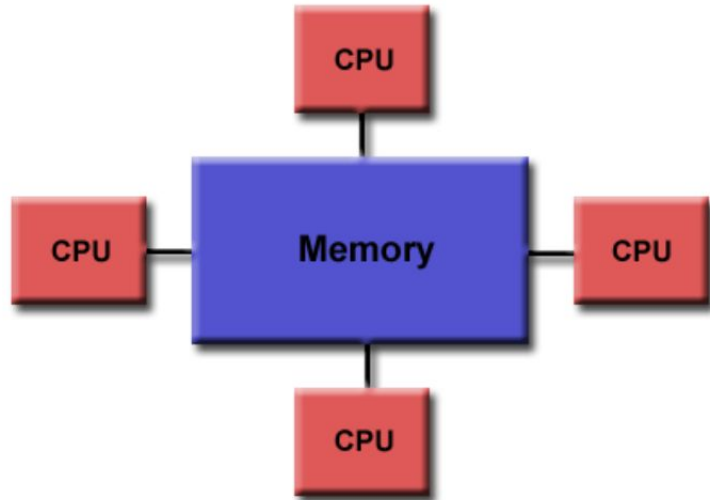




Tipos de arquitectura

Memoria compartida

Uniform Memory Access (UMA)



Llamado coherencia del caché

Si un procesador actualiza una ubicación en memoria compartida, todos los demás procesadores saben sobre la actualización

Procesadores idénticos con igual acceso y tiempos de acceso a la memoria

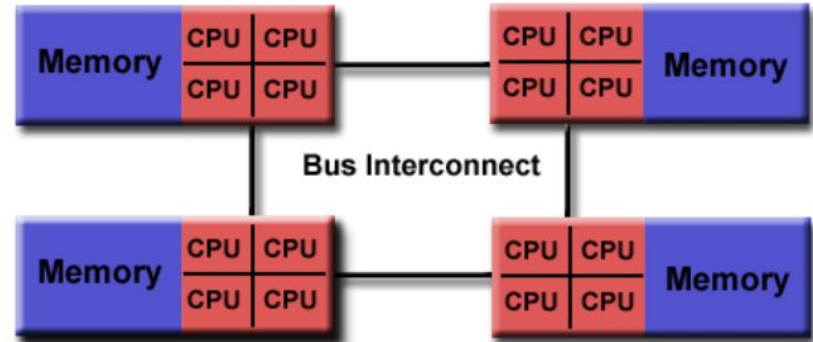
Memoria compartida

Hecho mediante la vinculación física de dos o más SMP

El acceso a la memoria es más lento

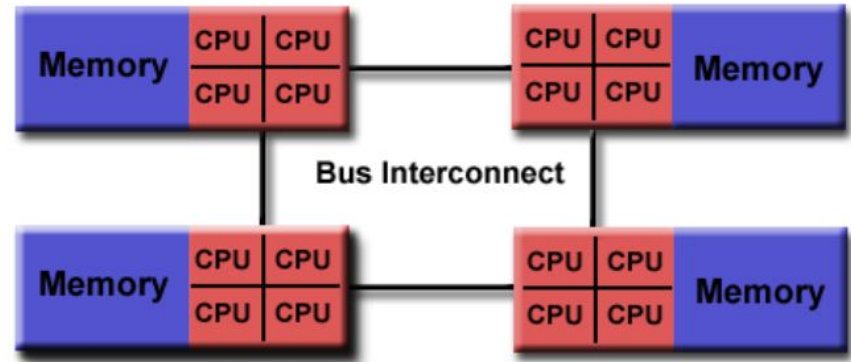
se mantiene la coherencia del caché

Non-Uniform Memory Access (NUMA)



Memoria distribuida

Las tareas intercambian datos por medio del paso y recepción de mensajes.

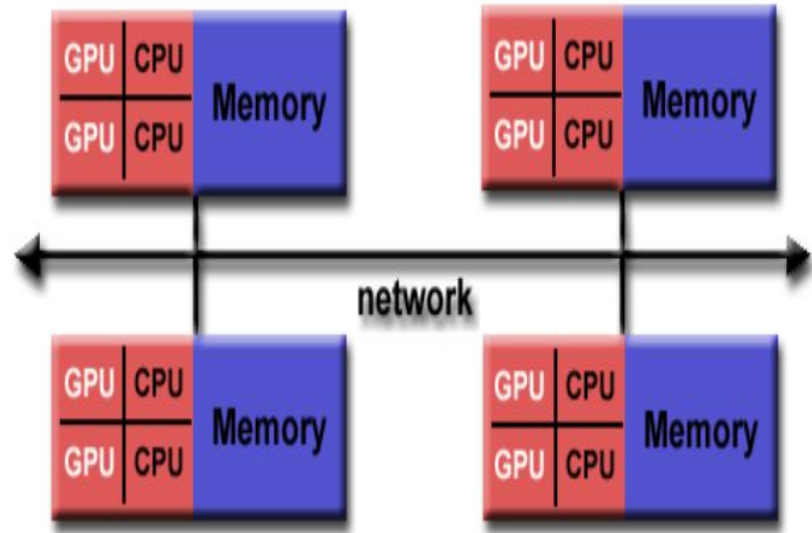


Debido a que cada procesador tiene su propia memoria local, funciona independientemente.

Híbrido memoria distribuida-compartida

Su principal ventaja es su escalabilidad.

Su principal desventaja es que la complejidad de programación aumenta.

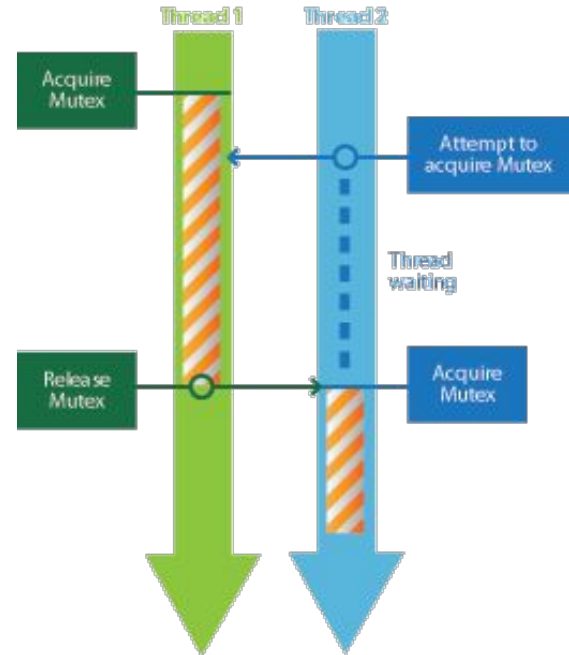




Tipos de sincronización

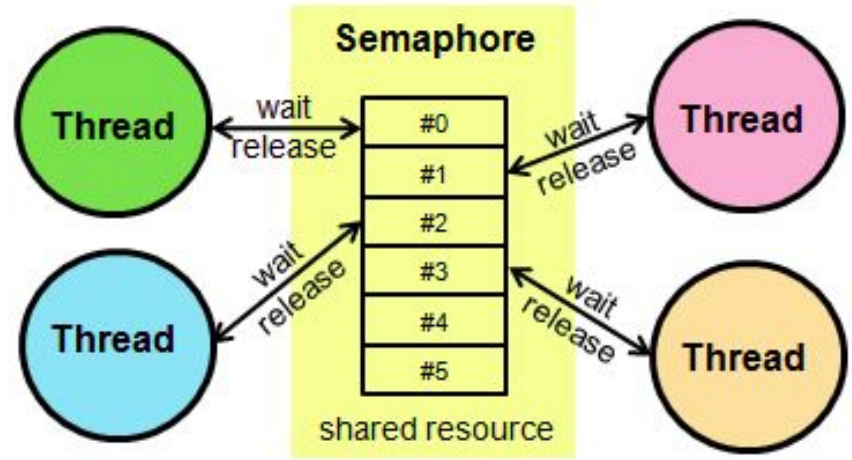
Barrier

- Todas las tareas están involucradas
- Cada tarea realiza su trabajo hasta que alcanza la barrera, ahí acaba o se suspende temporalmente
- Cuando la última tarea alcanza la barrera, todas las tareas están sincronizadas.



Semaphore

- Puede involucrar a cualquier número de tareas.
- Se utiliza comúnmente para proteger el acceso a una parte de la información o una sección de código
- Otras tareas diferentes a la que está accediendo al código no tienen permitido acceder a el hasta que la primera haya terminado



Operaciones de comunicación sincrónica

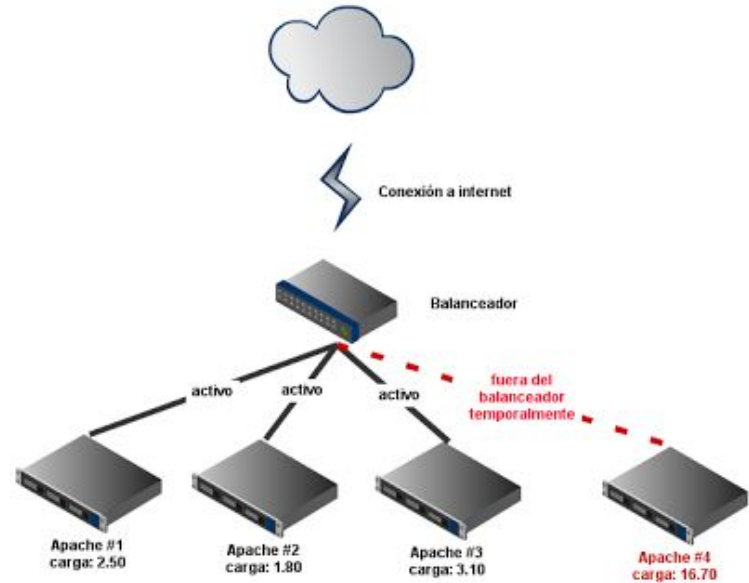
- Se utiliza para coordinar tareas de comunicación
- Tiene que haber una coordinación entre el emisor y el receptor para poder enviar un dato de un lado al otro

Balanceador de carga

Asignar el trabajo que recibe cada tarea equitativamente

Puede ser un factor significativo en el desempeño del programa

A menudo requiere "serialización" de segmentos del programa.



Asignación de trabajo dinámico

Cuando cada tarea termina su trabajo, espera en una cola para obtener una nueva pieza de trabajo.

Puede ser necesario diseñar un algoritmo que detecte y maneje desequilibrios de carga como ocurren dinámicamente dentro del código



Ejemplos

Bibliotecas

Posix
threads

MPI

PVM

Lenguajes

JULIA

LINDA

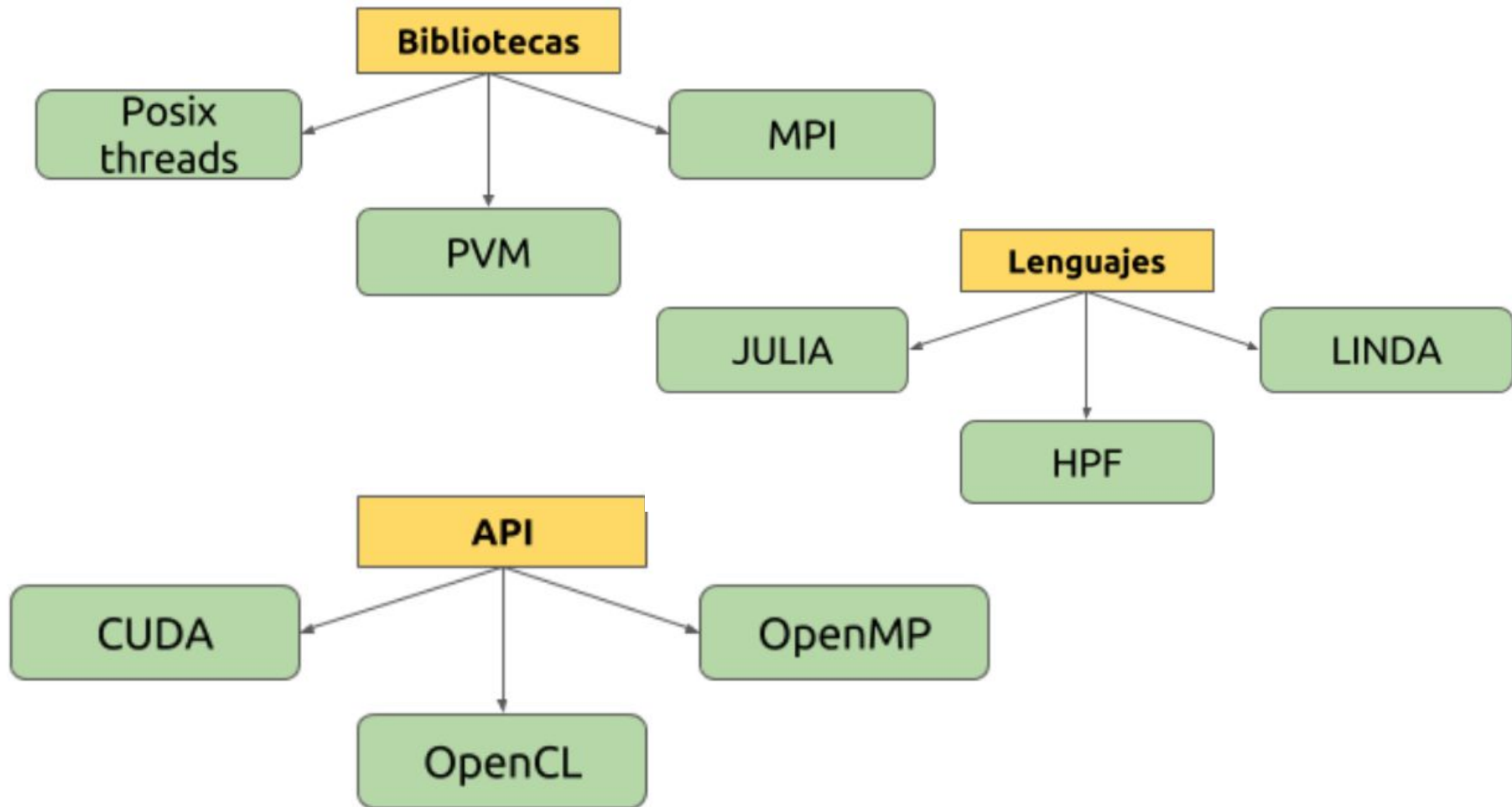
HPF

API

CUDA

OpenMP

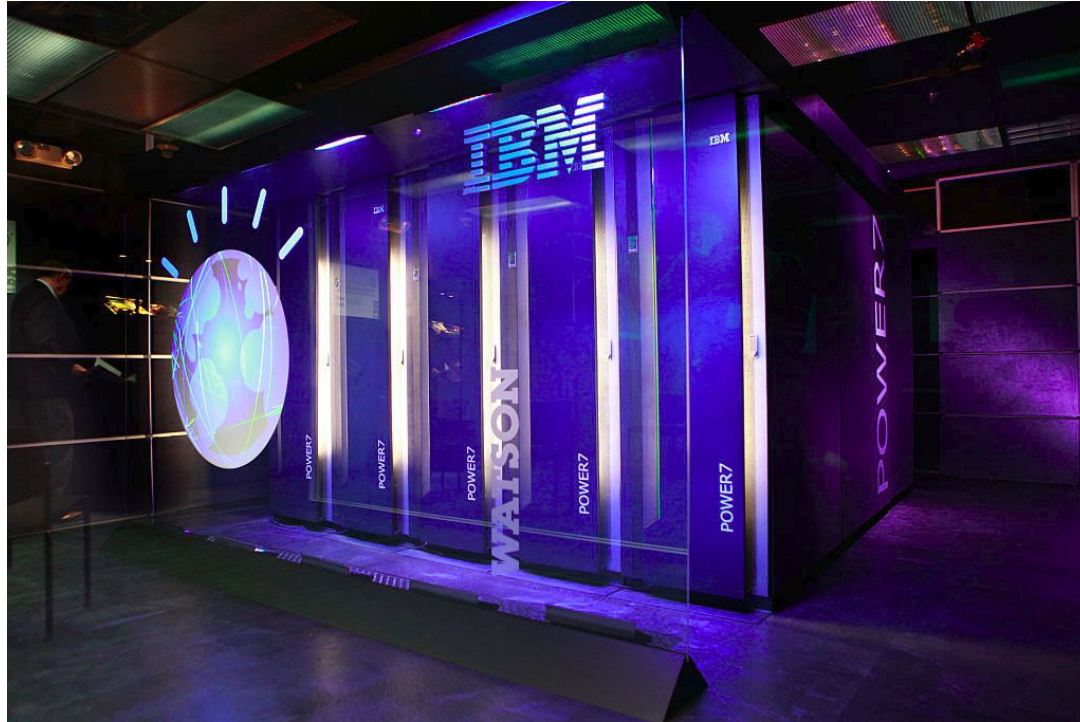
OpenCL





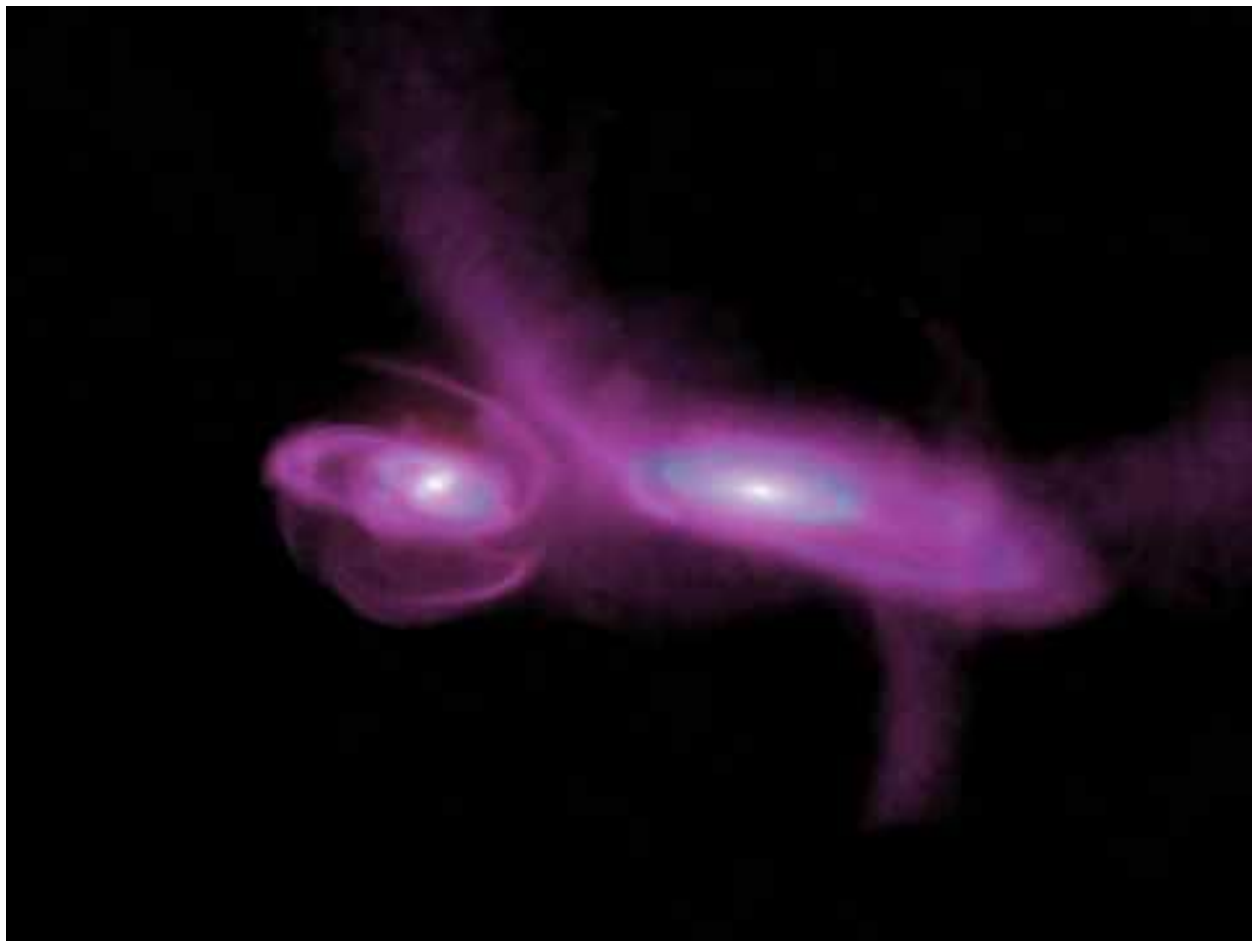
Aplicaciones

Reconocimiento de lenguaje natural

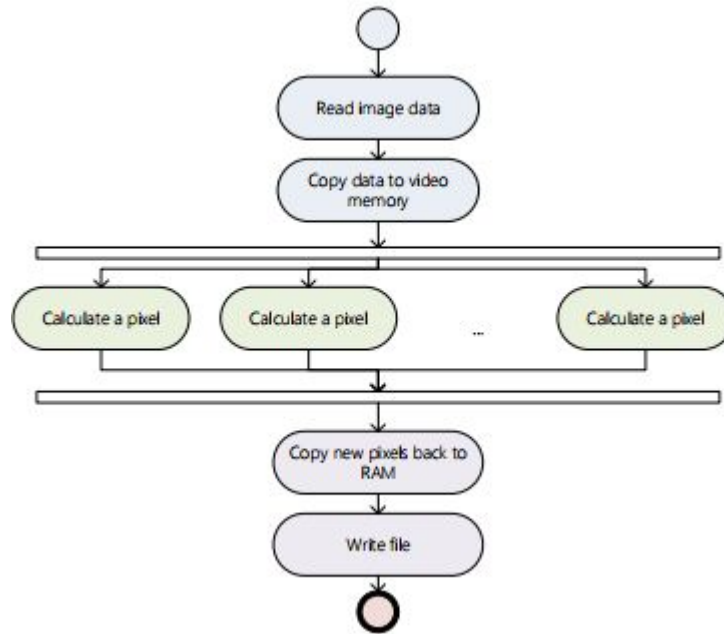


Astrofísica





Procesamiento de imágenes







Ejemplos



Gracias

Referencias

1. <http://www.bioblogia.com/wp-content/uploads/2011/05/secuencia-DNA.jpg>
2. By Kelvinsong - Own work, CC BY-SA 3.0 <https://commons.wikimedia.org/w/index.php?curid=23371669>
3. http://farm4.static.flickr.com/3281/2746545443_8e8ea69507.jpg
4. http://www-03.ibm.com/ibm/history/ibm100/images/icp/F892068E89296L65/us_en_us_ibm100_risc_architecture_john_cocke_750x990.jpg
5. <http://www.extremetech.com/wp-content/uploads/2012/04/cdc-6600-supercomputer.jpg>
6. <http://www.new-npac.org/projects/cdroms/cewes-1999-06-vol1/foils/cps615arch98/seporgimagedir/066.jpg>
7. https://computing.llnl.gov/tutorials/parallel_comp/
8. <https://www.slideshare.net/VinayGupta6/parallel-computing-12222922>
9. [https://en.wikipedia.org/wiki/Watson_\(computer\)](https://en.wikipedia.org/wiki/Watson_(computer))
10. <https://commons.wikimedia.org/wiki/File:NGC4676.jpg#/media/File:NGC4676.jpg>