uc3m | Universidad Carlos III de Madrid

OpenCourseWare

Sistemas Paralelos y Distribuidos

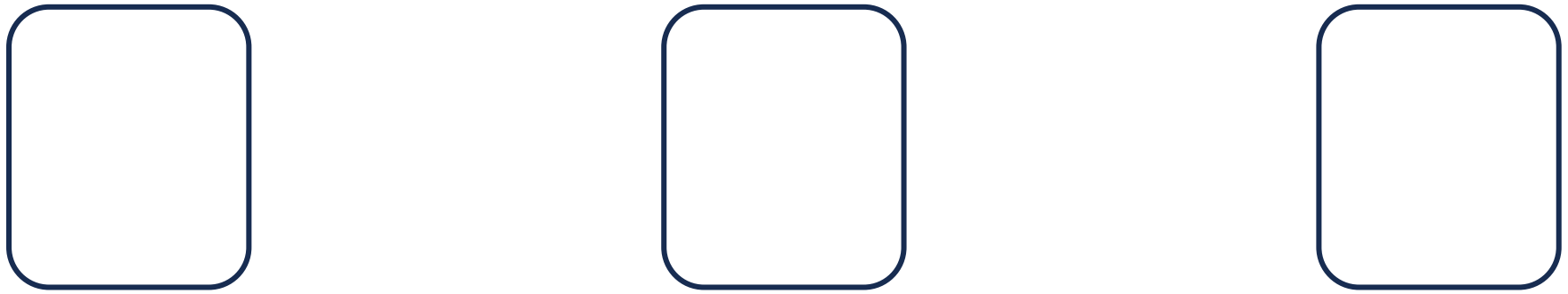
Félix García Carballeira

Alejandro Calderón Mateos

Sistemas de altas prestaciones en entornos distribuidos

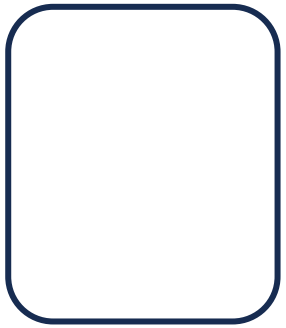
Motivación

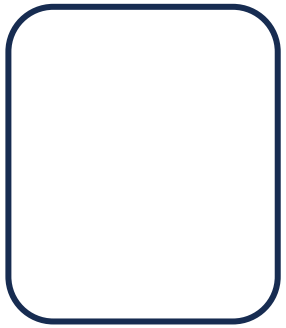
Arquitectura en 3 capas (*3-Tier architecture*)



Motivación

Arquitectura en 3 capas (*3-Tier architecture*)





Capa de presentación (*presentation layer*) ⟷ Capa de lógica de aplicación (*Aplication layer*) ⟷ Capa de acceso a datos (*Data access layer*)

Imagen generada con dall-e con la siguiente petición:
"generar una imagen con diferentes dispositivos móviles dispuestos encima de una mesa, incluyendo móviles, tablets, etc."

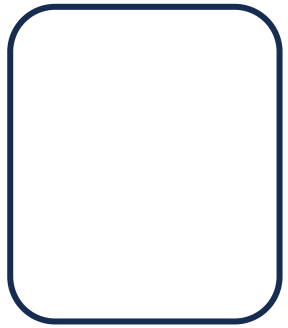
Motivación

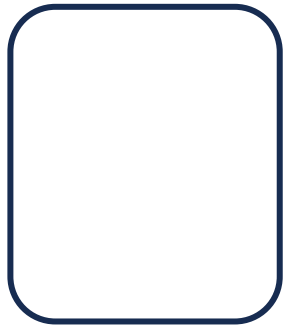
Arquitectura en 3 capas (*3-Tier architecture*)

Asistente IA

AR/VR

…





Capa de presentación (*presentation layer*) ⟷ Capa de lógica de aplicación (*Aplication layer*) ⟷ Capa de acceso a datos (*Data access layer*)

Motivación

Arquitectura en 3 capas (*3-Tier architecture*)

Asistente IA

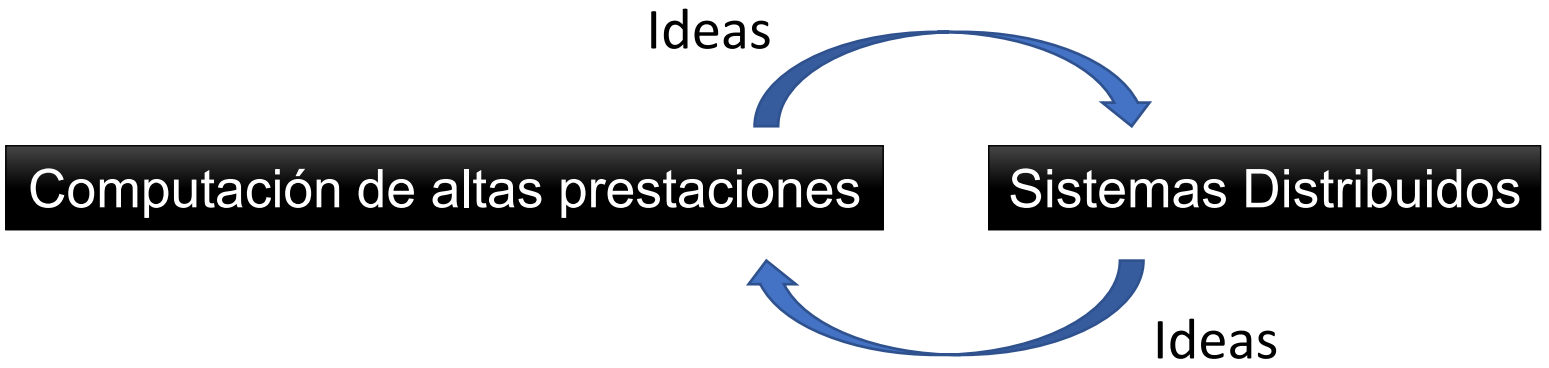
AR/VR

…

Capa de presentación
(*presentation layer*)

Imagen generada con dall-e con la siguiente petición:
“generar una imagen de una nube en cuyo interior haya un símbolo de infinito para indicar que la capacidad de la nube es infinita”

Motivación



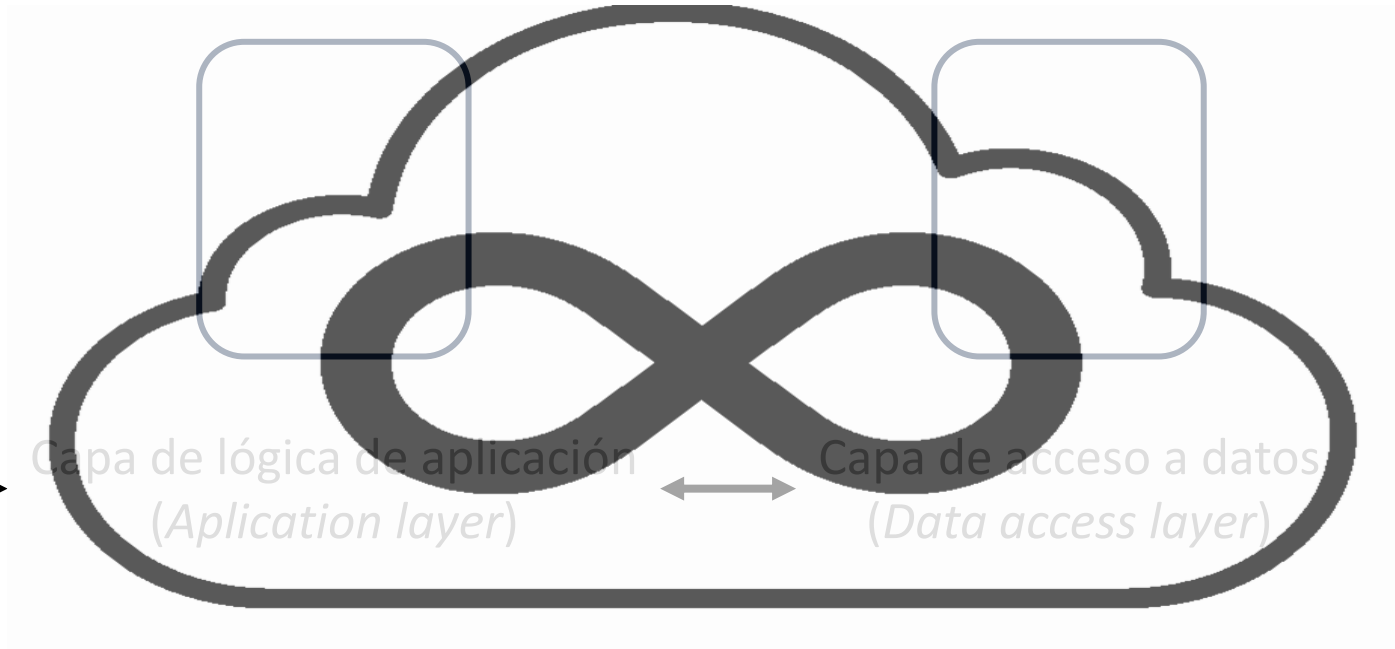
Arquitectura en 3 capas (*3-Tier architecture*)

Asistente IA

AR/VR

...

Capa de presentación
(*presentation layer*)



Capa de lógica de aplicación
(*Aplication layer*)

Capa de acceso a datos
(*Data access layer*)

Contenidos

- **Introducción** a la computación de altas prestaciones
  - Qué, dónde y cómo
  - Hardware y software
- **Evolución** de la computación de altas prestaciones
  - Plataformas
  - Tendencias

Contenidos

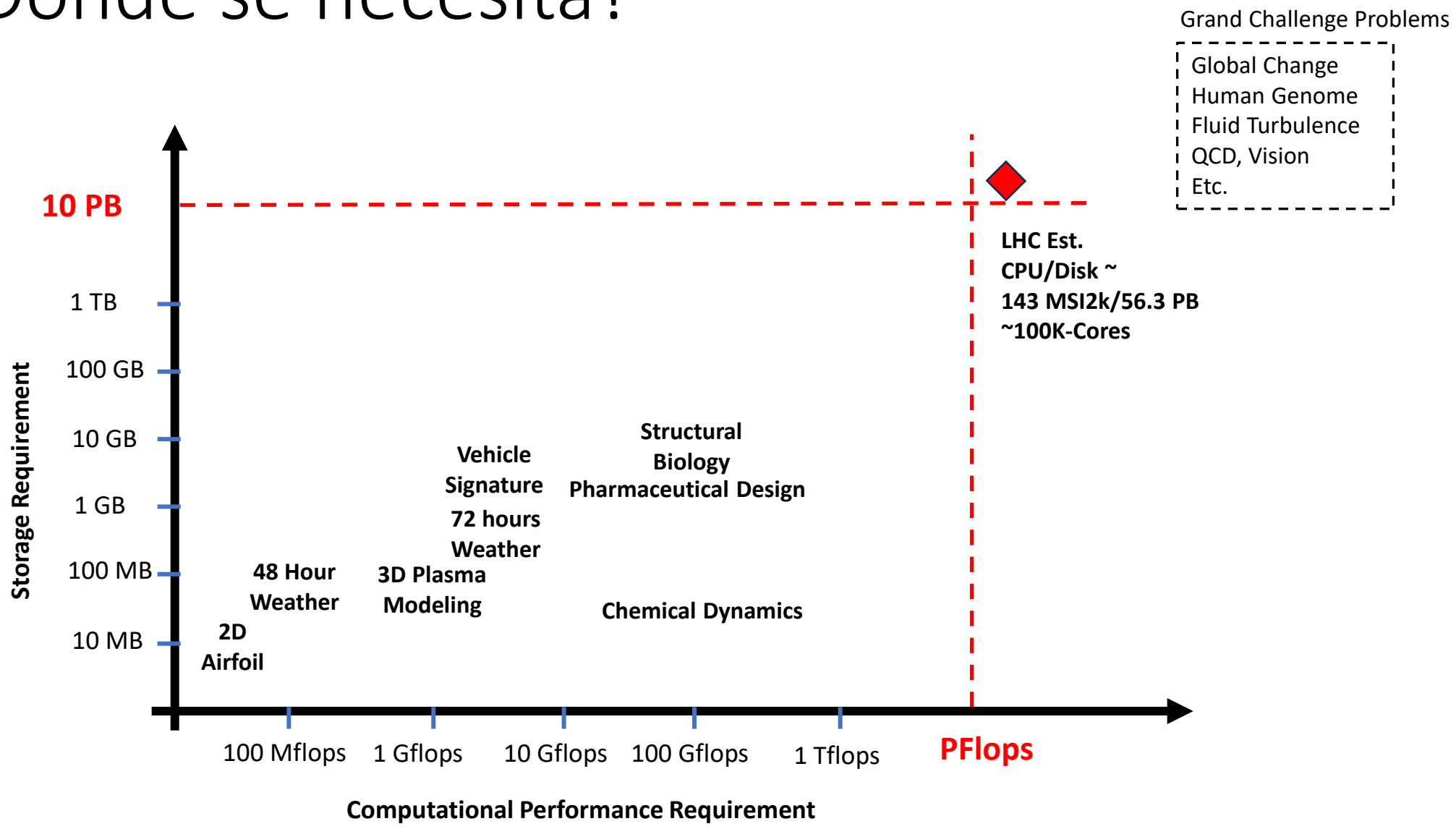
- **Introducción** a la computación de altas prestaciones
  - Qué, dónde y cómo
  - Hardware y software
- **Evolución** de la computación de altas prestaciones
  - Plataformas
  - Tendencias

Computación de altas prestaciones

- La computación de altas prestaciones o HPC (*High Performance Computing*) se centra principalmente en la velocidad.
- El objetivo es conseguir la máxima cantidad de cómputo posible en la mínima cantidad de tiempo.

Imagen generada con dall-e con la siguiente petición:
“generar la imagen de una mano estrujando un limón”

¿Dónde se necesita?



Basada en la figura de requerimientos de grandes aplicaciones del libro Parallel Computer Architectures: a Hardware/Software Approach.
D.E. Culler, J.P. Singh, with A. Gupta

Ejemplo 1: Big Hero 6 (2014)…
(http://www.engadget.com/2014/10/18/disney-big-hero-6/)

- **To manage that cluster and the 400,000-plus computations it processes per day (about 1.1 million computational hours/day), his team created software called Coda, which treats the four render farms like a single supercomputer**.
- **The film takes 199 million core-hours (181 days) of rendering**. To put the enormity of this computational effort into perspective, Hendrickson says that **Hyperion "could render *Tangled (2010)* from scratch every 10 days."**

Ejemplo 2: Monster's University (2022)… (https://sciencebehindpixar.org/pipeline/rendering)

- I'm a little confused on the whole rendering process. **They said it takes at least around 24 hours to render 1 frame**, and that there are **24 frames in a second**. If you take **a 100 minute movie**, then it would take around **400 years to render** that many frames.....
  — Jay.
- **Pixar has a huge "render farm,"** which is basically **a supercomputer composed of 2000 machines, and 24,000 cores.** This makes it **one of the 25 largest supercomputers in the world**. That said, with all that computing power, **it still took two years to render Monster's University**.
  — Peter Collingridge

¿Cómo se consigue más velocidad?

¿Cómo se consigue más velocidad?

- **Mejores algoritmos**
  - $O(n^2)$, viajante, …

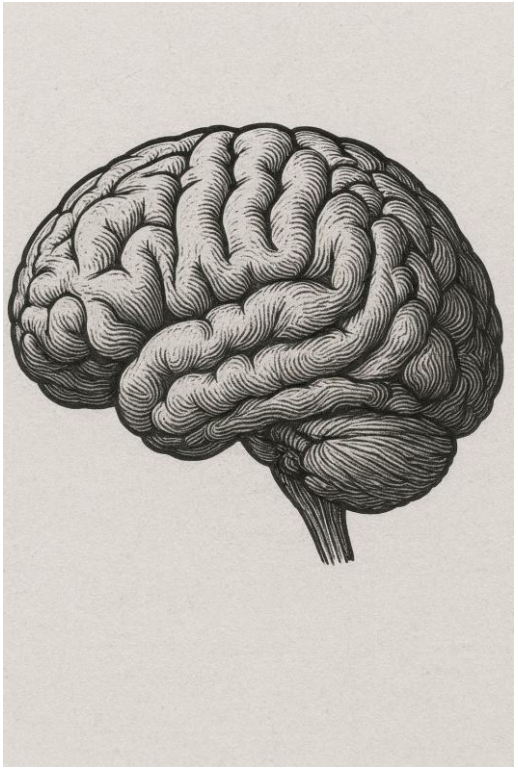


Imagen generada con dall-e con la siguiente petición:
"Generar una imagen de un encéfalo en blanco y negro donde se vean los surcos"

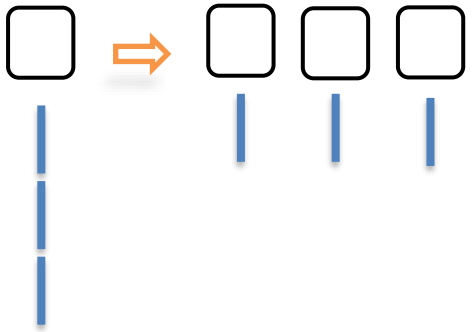
¿Cómo se consigue más velocidad?

- Mejores algoritmos
  - $O(n^2)$, viajante, …
- **Mejores procesadores (mejoras en la tecnología)**
  - CPU a 10 GHz, 510 TB de RAM, …

Imagen generada con dall-e con la siguiente petición:
"Generar una caricatura de una persona llevando una bolsa de dinero casi de su tamaño con la marca de dólar en la bolsa en blanco y negro"

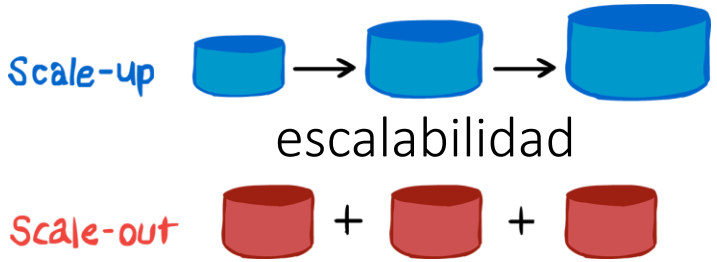
¿Cómo se consigue más velocidad?

- Mejores algoritmos
  - $O(n^2)$, viajante, …
- Mejores procesadores (mejoras en la tecnología)
  - CPU a 10 GHz, 510 TB de RAM, …
- **Paralelismo (mejoras en el uso de la tecnología actual)**
  - Speedup, Ley de Amdahl, …



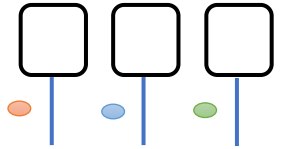
¿Cómo se consigue más velocidad?

- Mejores algoritmos
  - $O(n^2)$, viajante, …
- Mejores procesadores (mejoras en la tecnología)
  - CPU a 10 GHz, 510 TB de RAM, …
- Paralelismo (mejoras en el uso de la tecnología actual)
  - Speedup, Ley de Amdahl, …



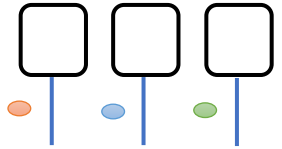
Tipos de paralelismo

- Tareas independientes:

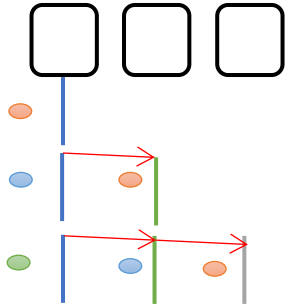


Tipos de paralelismo

- Tareas independientes:

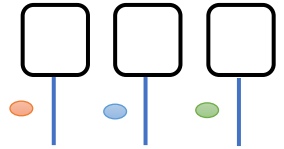


- Tareas cooperativas:
  - *Pipeline*
  - Coordinación (*mutex* y *conditions*)

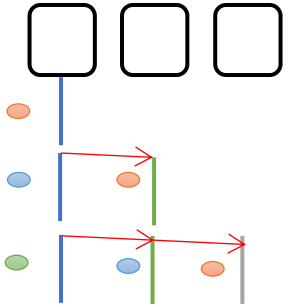


Tipos de paralelismo

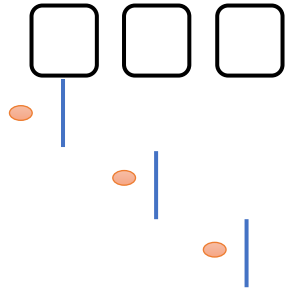
- Tareas independientes:



- Tareas cooperativas:
  - *Pipeline*
  - Coordinación (*mutex* y *conditions*)



- Tareas competitivas:
  - Código secuencial :-S



Speedup

- La mejora (o *speedup*) en la ejecución paralela con n elementos de cómputo será:

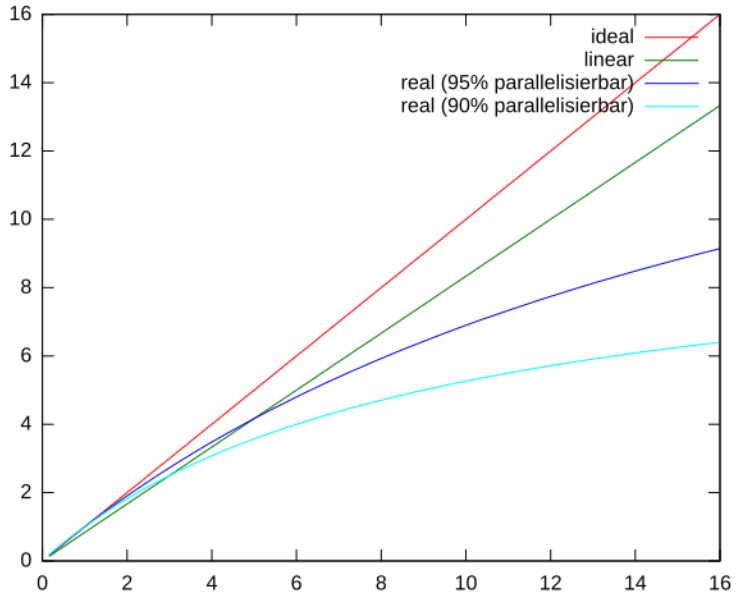
  speedup = tiempo_de_ejecución (1) / tiempo_de_ejecución (**n**)

Speedup

- La mejora (o *speedup*) en la ejecución paralela con **n** elementos de cómputo será:

  speedup = tiempo_de_ejecución (1) / tiempo_de_ejecución (**n**)

- No siempre se obtiene un *speedup* ideal:



Ley de Amdahl

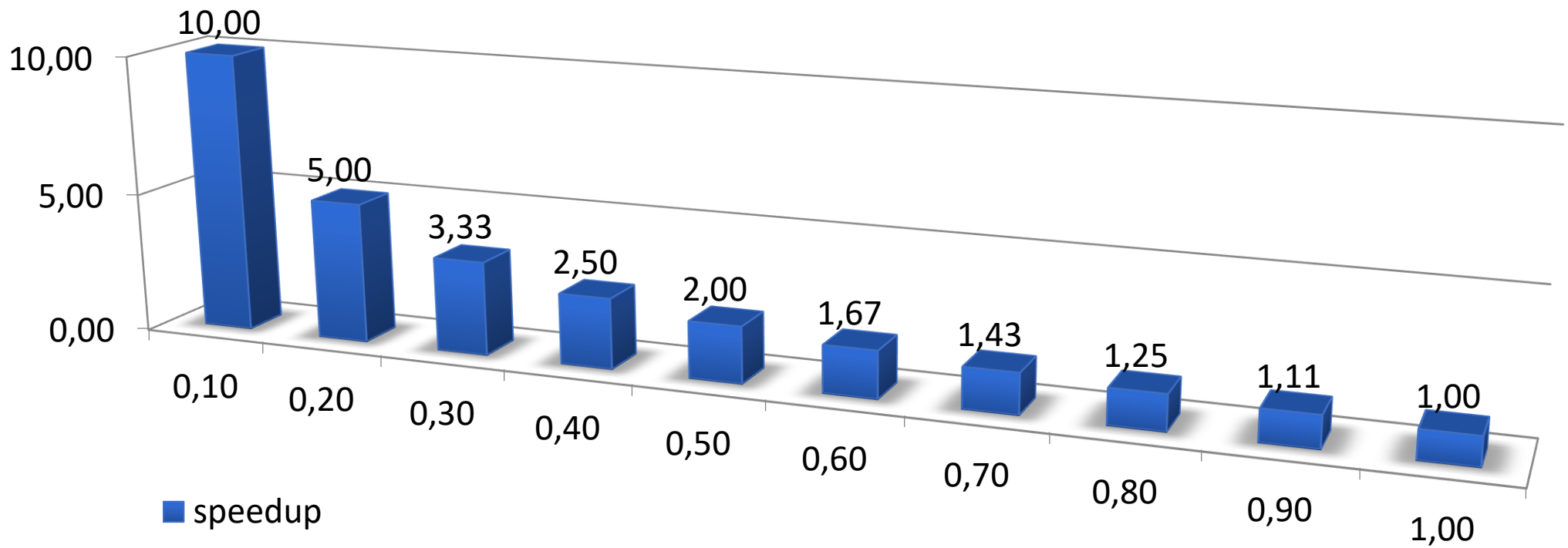
- Ley de Amdahl:

  "el *speedup* teórico está limitado por la fracción secuencial s del programa"

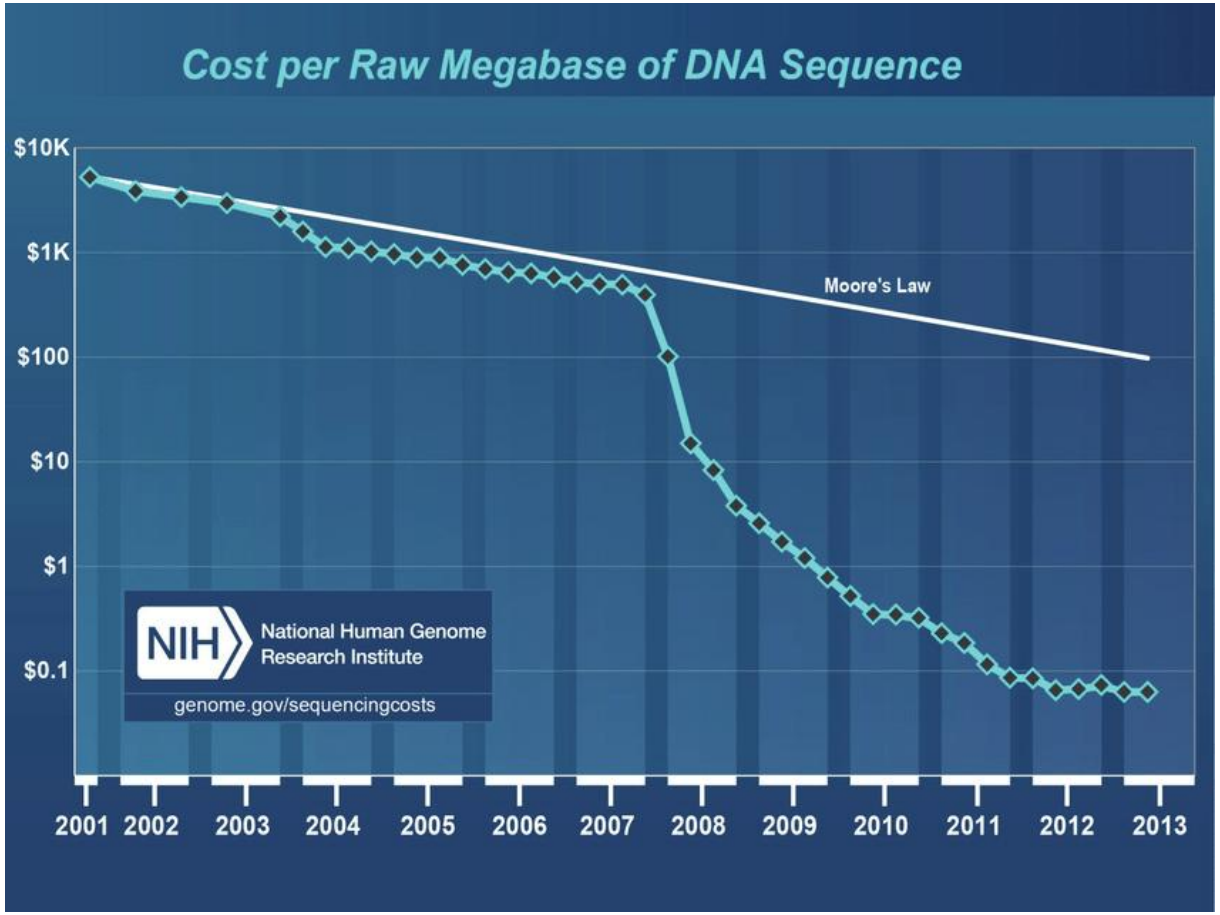
$$speedup <= \frac{1}{s + \frac{(1-s)}{\mathbf{n}}}$$

SI **n**↑ ENTONCES ***speedup* ~ 1 / s**

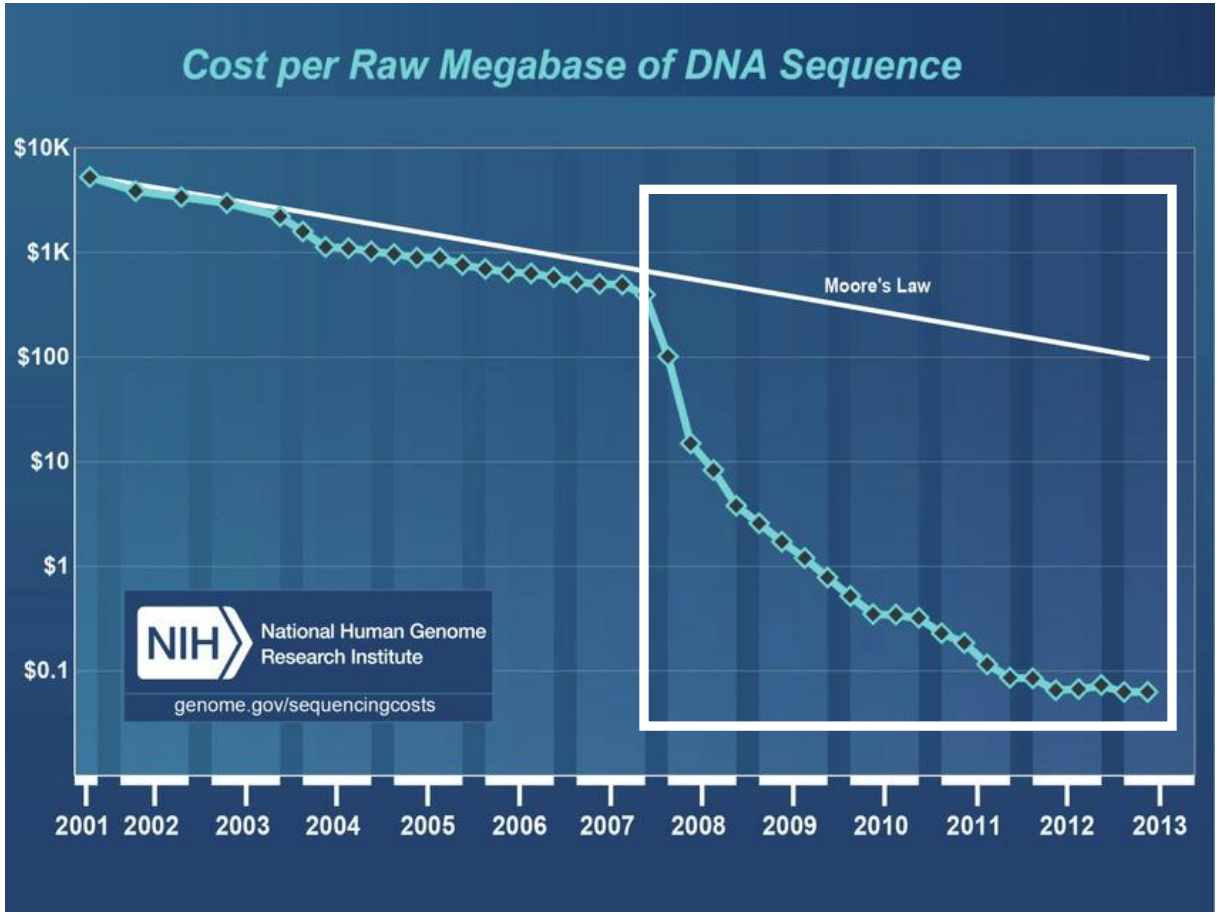
Ley de Amdahl



¿Eso del paralelismo ayuda?
caso de estudio: genoma humano



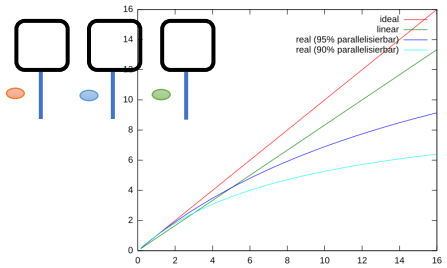
¿Eso del paralelismo ayuda?
caso de estudio: genoma humano



Yes!

¿Eso del paralelismo ayuda?
caso de estudio: genoma humano

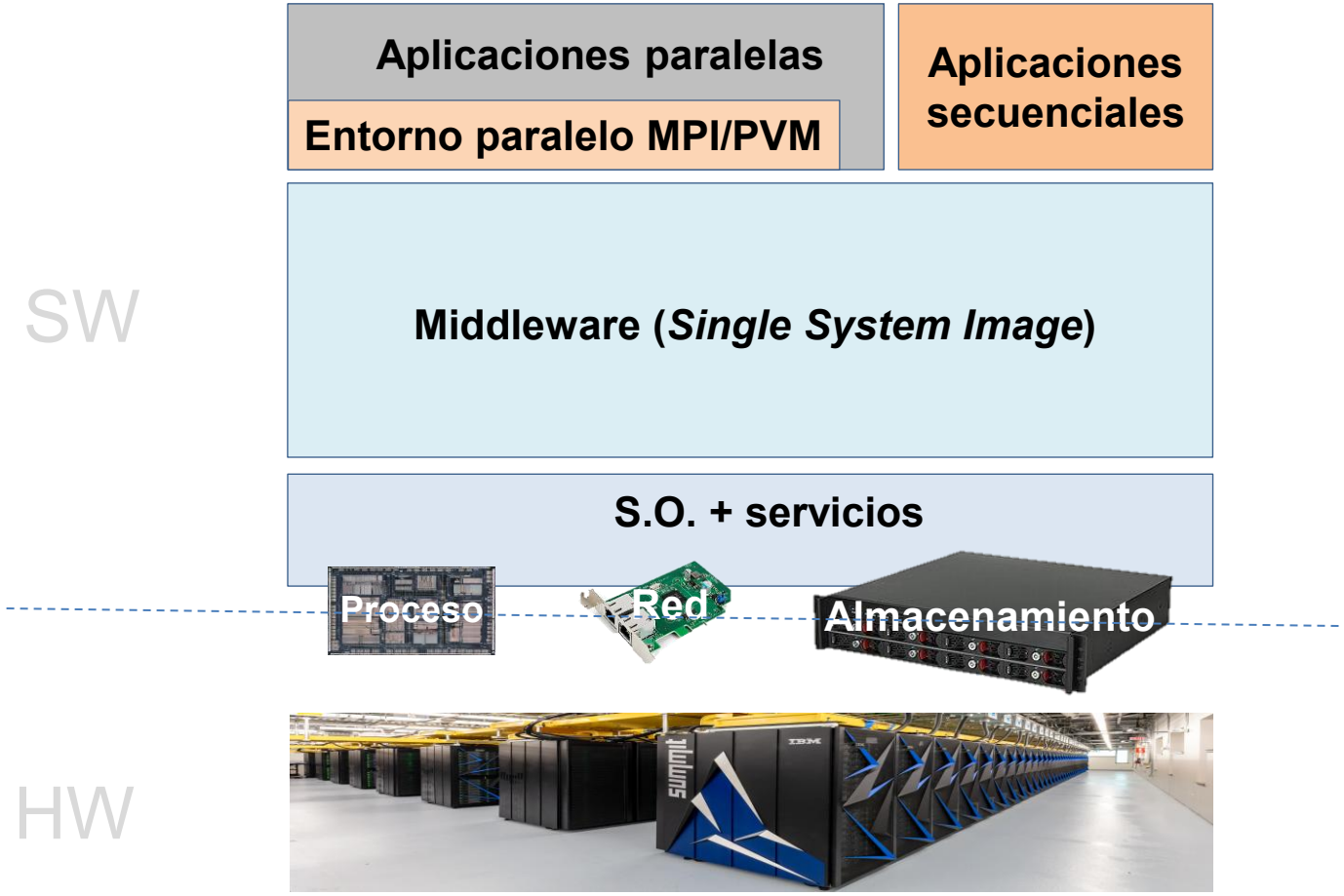
- Mejores algoritmos
  - $O(n^2)$, viajante, …
- Mejores procesadores
  - 10 GHz, 510 TB, …
- Paralelismo
  - Ley de Amdahl, …



Contenidos

- **Introducción** a la computación de altas prestaciones
  - Qué, dónde y cómo
  - Hardware y software
- **Evolución** de la computación de altas prestaciones
  - Plataformas
  - Tendencias

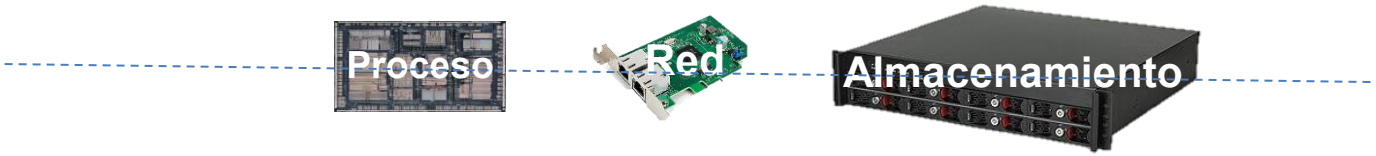
Plataforma hardware y software



Computador de altas prestaciones

Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

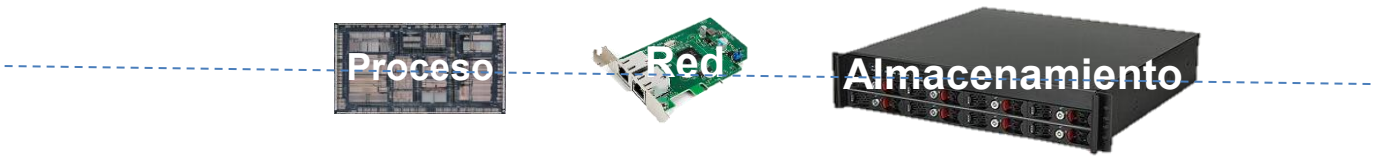
Plataforma hardware



HW

- **Procesamiento**
  (vectorial vs multiprocesador)
- **Memoria**
  (compartida vs distribuida)

Plataforma hardware



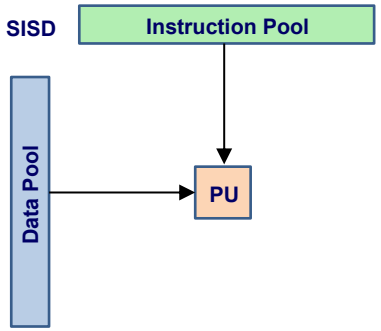
HW

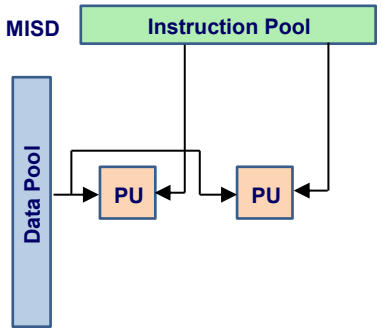
- **Procesamiento**
  (vectorial vs multiprocesador)
- **Memoria**
  (compartida vs distribuida)

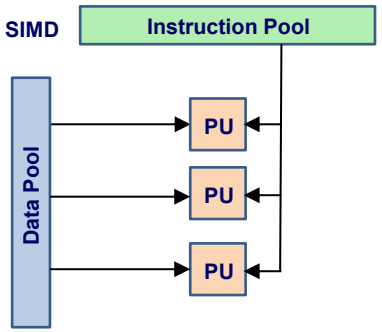
Taxonomía de Flynn

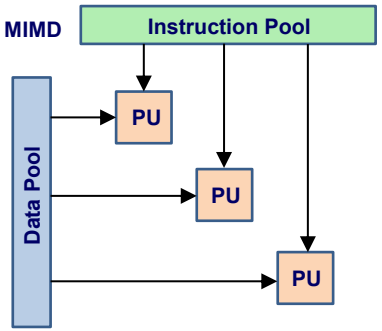
http://www.buyya.com/microkernel/chap1.pdf

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD





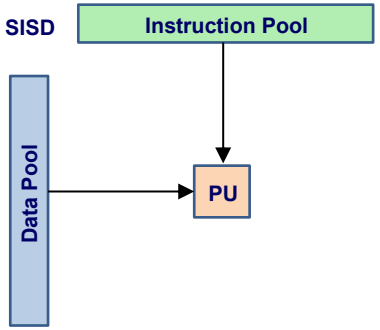


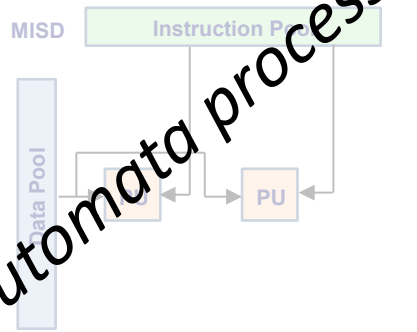


Taxonomía de Flynn

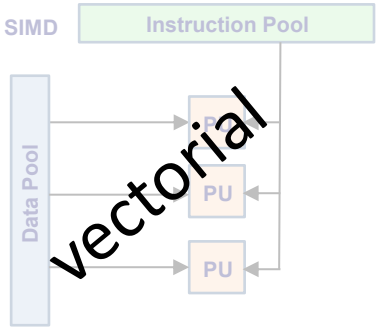
http://www.buyya.com/microkernel/chap1.pdf

	Single Instruction	Multiple Instruction
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

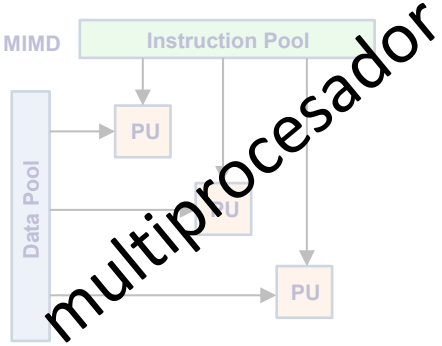




automata processor

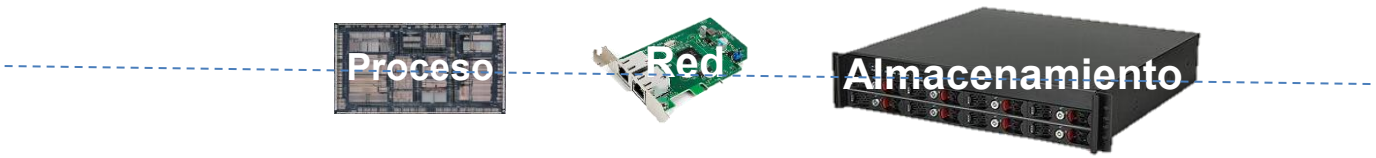


vectorial



multiprocesador

Plataforma hardware

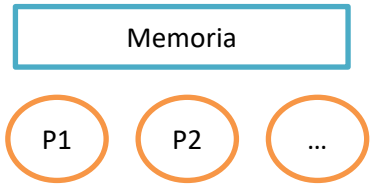


HW

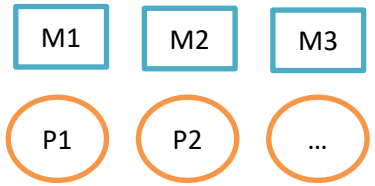
- **Procesamiento**
  (vectorial vs multiprocesador)
- **Memoria**
  (compartida vs distribuida)

Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

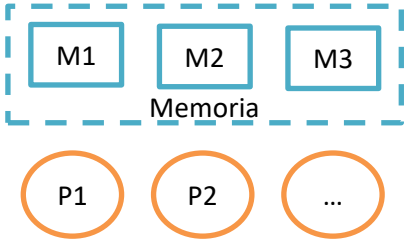
Acceso a memoria



- Memoria compartida (UMA)

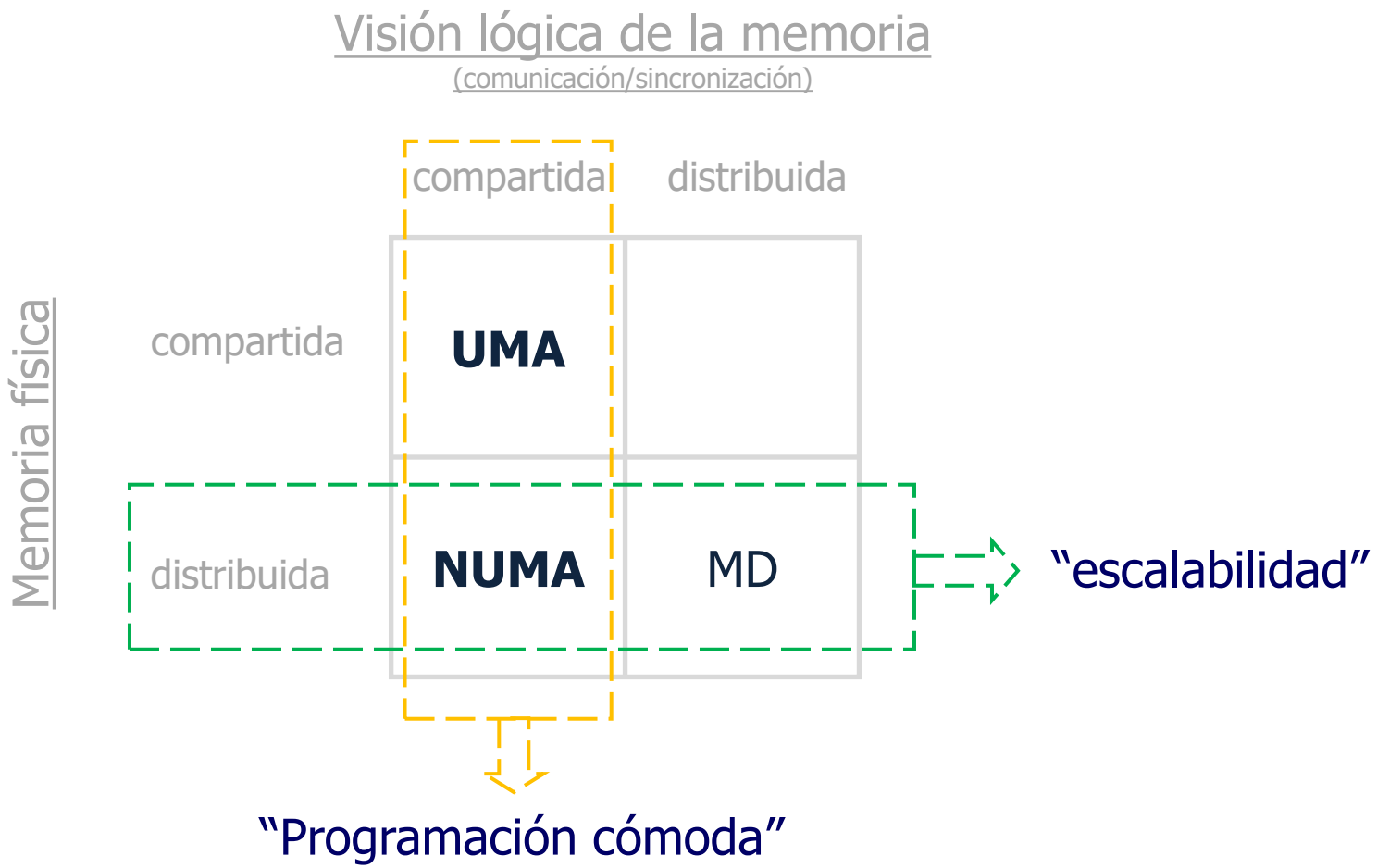


- Memoria distribuida (MD)



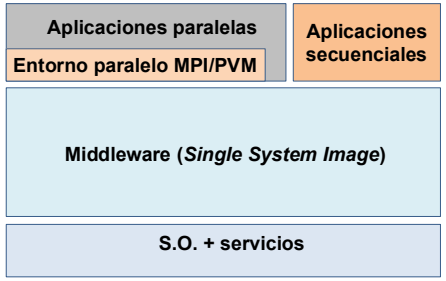
- Memoria lógicamente compartida (NUMA)

Acceso a memoria



Plataforma software

SW



- Vectoriales
  - Uso de instrucciones especiales
- Multiprocesador
  - UMA, NUMA
    - OpenMP, …
  - M. Distribuida
    - MPI, …

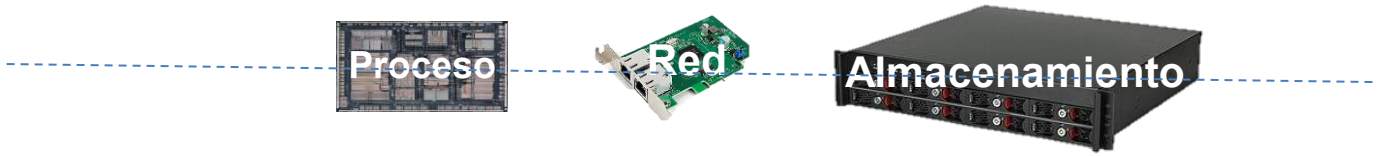
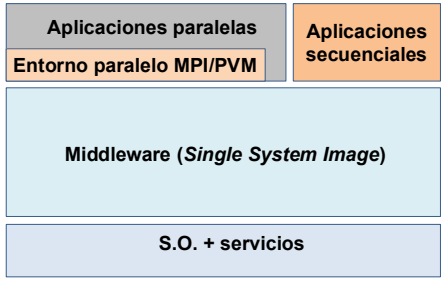


Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

Plataforma software

SW



- Vectoriales
  - Uso de instrucciones especiales
- Multiprocesador
  - UMA, NUMA
    - OpenMP, …
  - M. Distribuida
    - MPI, …

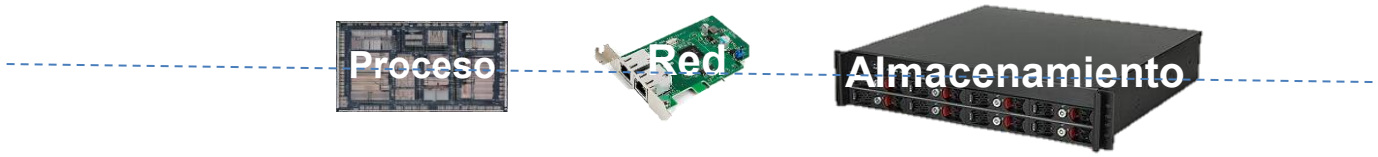


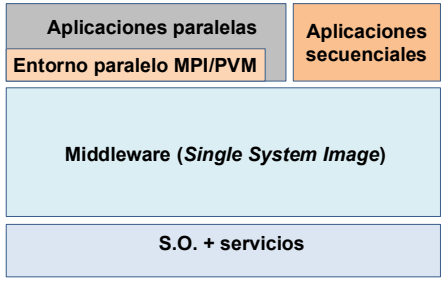
Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

Cómo es OpenMP

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
   #pragma omp parallel private(nthreads, tid)
   {
      int tid = omp_get_thread_num();
       printf("Hello World from thread = %d\n", tid);

      #pragma omp master
      {
         int nthreads = omp_get_num_threads();
          printf("Number of threads = %d\n", nthreads);
      }
   }
}
```

Plataforma software

SW



- Vectoriales
  - Uso de instrucciones especiales
- Multiprocesador
  - UMA, NUMA
    - OpenMP, …
  - M. Distribuida
    - MPI, …

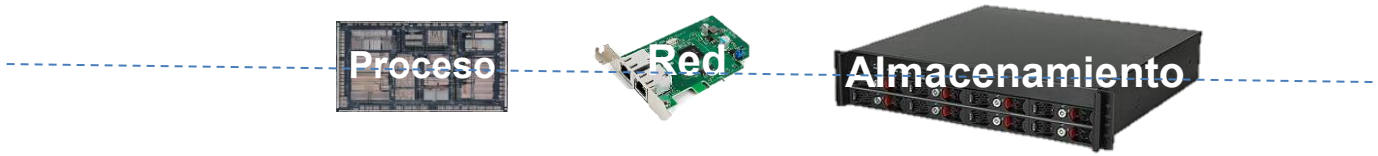


Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

Qué es MPI

- MPI es una interfaz de paso de mensaje que representa un esfuerzo prometedor de mejorar la disponibilidad de un software altamente eficiente y portable para satisfacer las necesidades actuales en la computación de alto rendimiento a través de la definición de un estándar de paso de mensajes universal.

William D. Gropp et al.

Principales pilares de MPI

- Portabilidad:
  - Definido independiente de plataforma paralela.
  - Útil en arquitecturas paralelas heterogéneas.
- Eficiencia:
  - Definido para aplicaciones multihilo (*multithread*)
  - Sobre una comunicación fiable y eficiente.
  - Busca el máximo de cada plataforma.
- Funcionalidad:
  - Fácil de usar por cualquier programador que ya haya usado cualquier biblioteca de paso de mensajes.

Implementaciones de MPI

Open MPI 5.0.5 (23/7/2024)

- http://www.open-mpi.org/
- FT-MPI + LA-MPI + LAM/MPI + PACX-MPI

MPICH 4.2.2 (3/7/2024)

- http://www.mpich.org/
- Argonne National Laboratory & University of Chicago

Cómo es MPI

```
#include <stdio.h>
#include "mpi.h"

main(int argc, char **argv)
{
        int node,size;
        int tam = 255;
        char name[255];

        MPI_Init(&argc, &argv);

       MPI_Comm_size (MPI_COMM_WORLD, &size);
       MPI_Comm_rank(MPI_COMM_WORLD, &node);
       MPI_Get_processor_name(name, &tam);
        printf("Hola Mundo2 del proceso %d de %d procesos (%s)\n", node, size, name);

        MPI_Finalize();
}
```

Cómo es MPI: uso interactivo

uc3m15672@login2:~/tmp> mpicc -g -o hello hello.c

uc3m15672@login2:~/tmp> cat > machines

```
login1
login2
login3
login4
^D
```

uc3m15672@login2:~/tmp> mpirun -np 4 -machinefile machines ~/tmp/hello

```
Hola Mundo2 del proceso 0 de 4 procesos (login1)
Hola Mundo2 del proceso 3 de 4 procesos (login4)
Hola Mundo2 del proceso 1 de 4 procesos (login2)
Hola Mundo2 del proceso 2 de 4 procesos (login3)
```

Cómo es MPI: uso de SLURM (1)

uc3m15672@login2:~/tmp> cat hello.cmd

```
#!/bin/bash
#SBATCH --job-name=mpi
#SBATCH --output=mpi_%j.out
#SBATCH --error=mpi_%j.err
#SBATCH --nodes=4
#SBATCH --exclusive
#SBATCH --time=00:05:00
#SBATCH --qos=debug

export HOME_PATH=/home/uc3m15/uc3m15672
scontrol show hostnames "${SLURM_JOB_NODELIST}"          > ${HOME_PATH}/tmp/machine_file
mpirun -np 2 -machinefile ${HOME_PATH}/tmp/machine_file  ${HOME_PATH}/tmp/hello
```

Cómo es MPI: uso de SLURM (2)

```
uc3m15672@login2:~/tmp> sbatch hello.cmd
Submitted batch job 25372807

uc3m15672@login2:~/tmp> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          25372807 sequentia hello.cm uc3m1567 PD       0:00      1 (None)


uc3m15672@login2:~/tmp> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          25372807 sequentia hello.cm uc3m1567 R        0:05      1 s07r2b72

uc3m15672@login2:~/tmp> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

Cómo es MPI: uso de SLURM (3)

uc3m15672@login2:~/tmp> cat slurm-25372807.out
Hola Mundo2 del proceso 1 de 2 procesos (s07r1b08)
Hola Mundo2 del proceso 0 de 2 procesos (s07r1b05)

Cómo es MPI: uso de SLURM (4)

```
uc3m15672@login2:~/tmp> bsc_queues
  queue         job          max          wall clock
  name          nodes        cores                time
------------    -----        -----       ----------------
debug             16          768           00:10:00
interactive        1            4           02:00:00
bsc               50         2400           48:00:00
RES Class A      200         9600           72:00:00
PRACE            400        19200           72:00:00
```

MPI 2.2 – 4.1

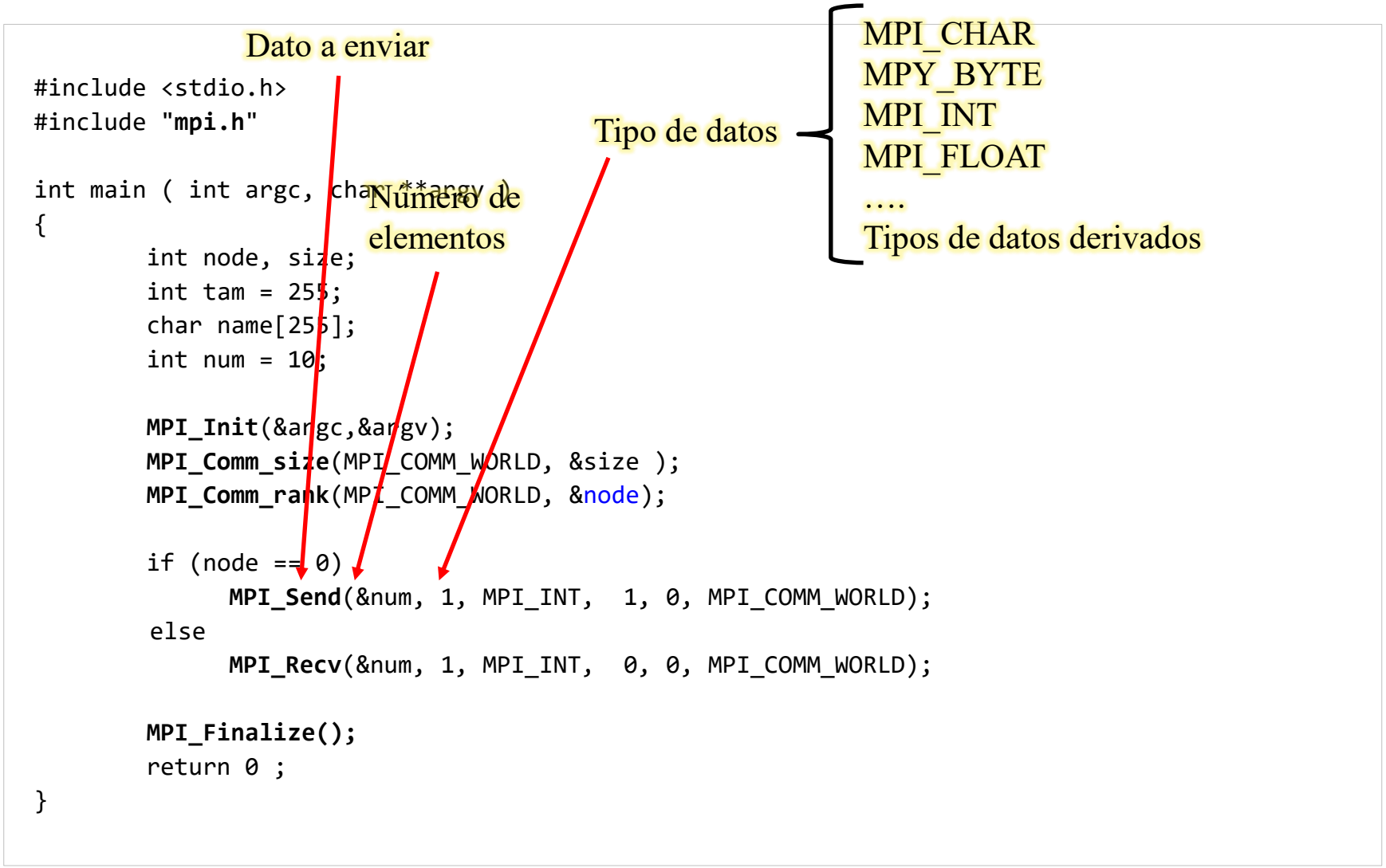
(http://mpi-forum.org/docs/)

- Estructuras de datos
  - Tipos de datos (básicos, vectores, compuestos, …)
  - Grupo de procesos (grupos, comunicadores, …)
- Paso de mensajes
  - Llamadas punto a punto (bloqueantes, …)
  - Llamadas colectivas (*bcast*, *scatter*, *gather*, …)
- Entrada y salida
  - Gestión de ficheros (apertura, cierre, …)
  - Gestión de contenidos (vistas, punteros, …)
- Procesos
  - Gestión de procesos (creación, …)
  - *Profiling*

Send-receive en MPI

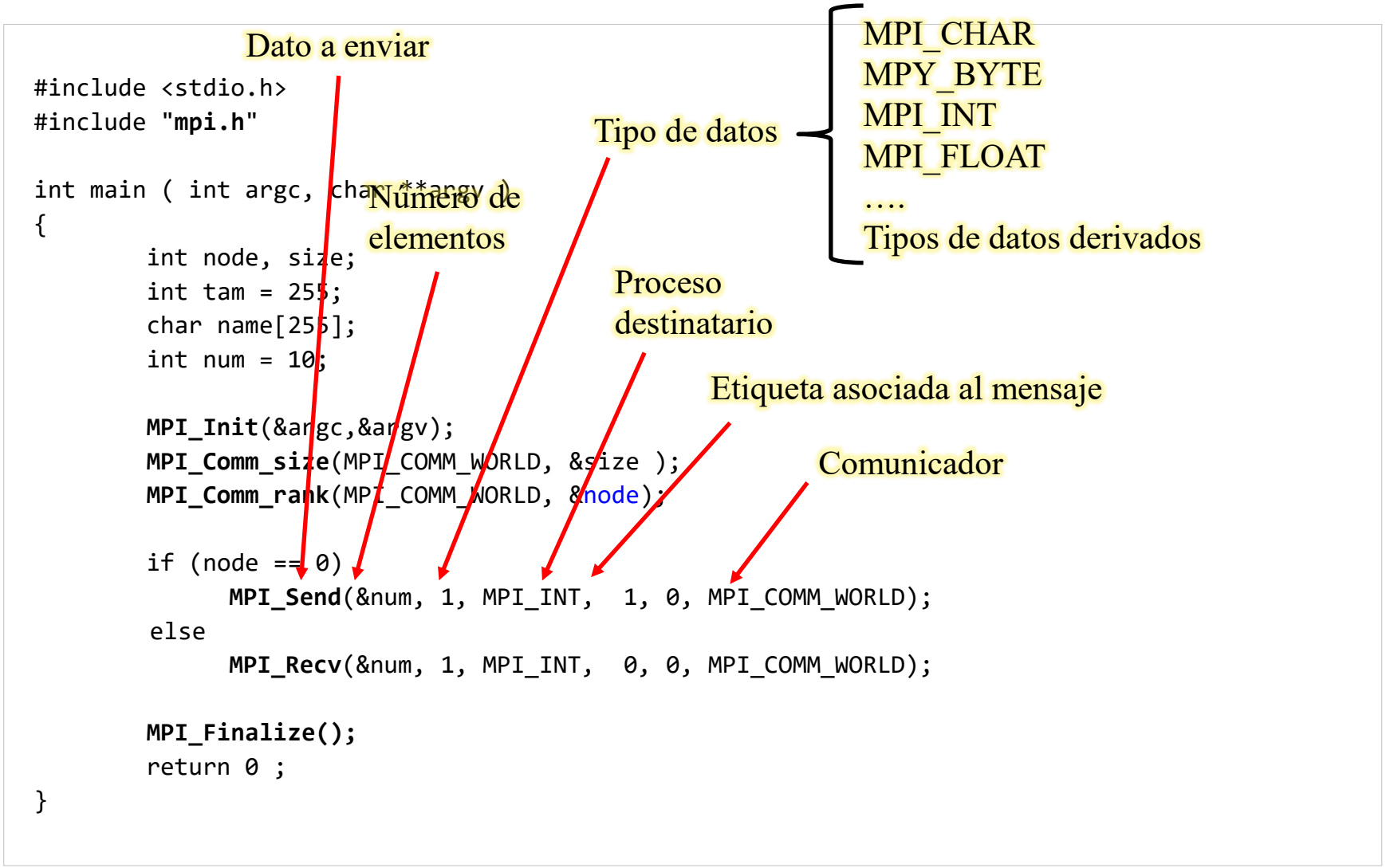
```
#include <stdio.h>
#include "mpi.h"

int main ( int argc, char **argv )
{
        int node, size;
        int tam = 255;
        char name[255];
        int num = 10;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node);

        if (node == 0)
              MPI_Send(&num, 1, MPI_INT,  1, 0, MPI_COMM_WORLD);
        else
              MPI_Recv(&num, 1, MPI_INT,  0, 0, MPI_COMM_WORLD);

        MPI_Finalize();
        return 0 ;
}
```

Send-receive en MPI




```c
#include <stdio.h>
#include "mpi.h"

int main ( int argc, char **argv )
{
        int node, size;
        int tam = 255;
        char name[255];
        int num = 10;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node);

        if (node == 0)
              MPI_Send(&num, 1, MPI_INT,  1, 0, MPI_COMM_WORLD);
        else
              MPI_Recv(&num, 1, MPI_INT,  0, 0, MPI_COMM_WORLD);

        MPI_Finalize();
        return 0 ;
}
```

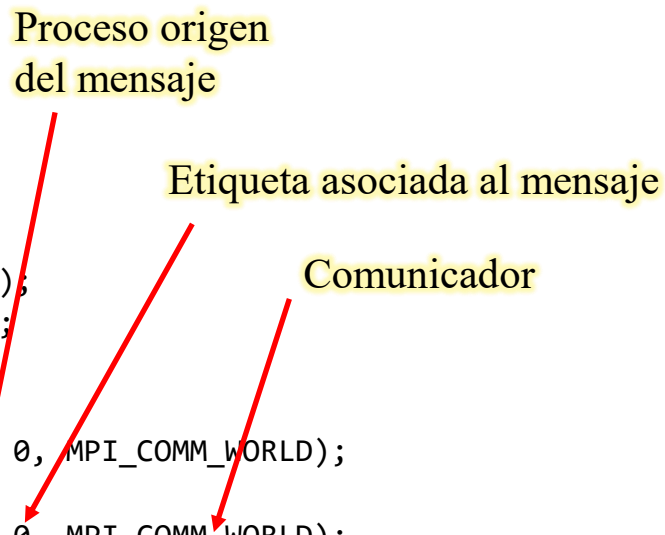
Send-receive en MPI




```c
#include <stdio.h>
#include "mpi.h"

int main ( int argc, char **argv )
{
        int node, size;
        int tam = 255;
        char name[255];
        int num = 10;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node);

        if (node == 0)
              MPI_Send(&num, 1, MPI_INT,  1, 0, MPI_COMM_WORLD);
        else
              MPI_Recv(&num, 1, MPI_INT,  0, 0, MPI_COMM_WORLD);

        MPI_Finalize();
        return 0 ;
}
```

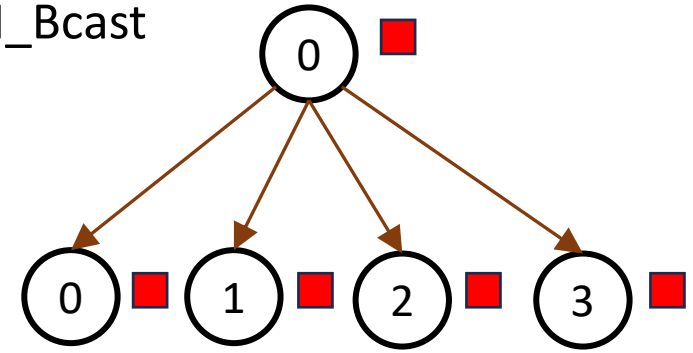
Send-receive en MPI

```
#include <stdio.h>
#include "mpi.h"

int main ( int argc, char **argv )
{
        int node, size;
        int tam = 255;
        char name[255];
        int num = 10;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node);

        if (node == 0)
              MPI_Send(&num, 1, MPI_INT,  1, 0, MPI_COMM_WORLD);
        else
              MPI_Recv(&num, 1, MPI_INT,  0, 0, MPI_COMM_WORLD);

        MPI_Finalize();
        return 0 ;
}
```

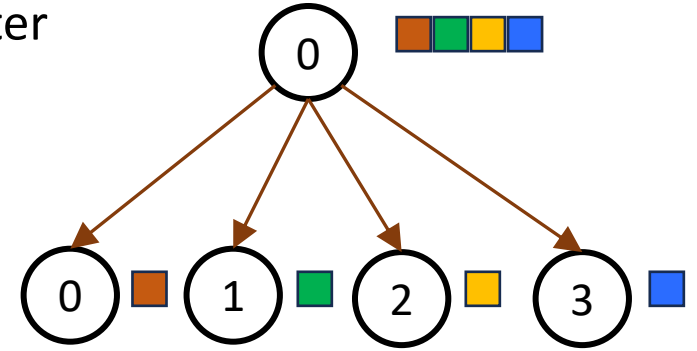


Operaciones colectivas

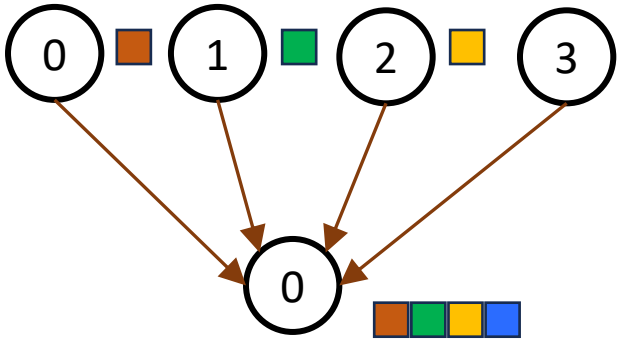
MPI_Bcast



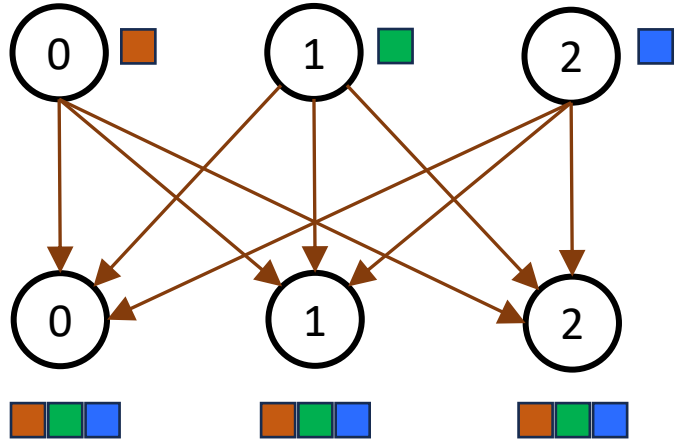
MPI_Scatter



MPI_Gather

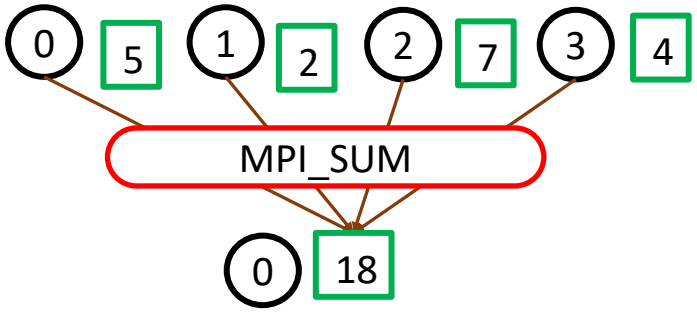


MPI_Allgather

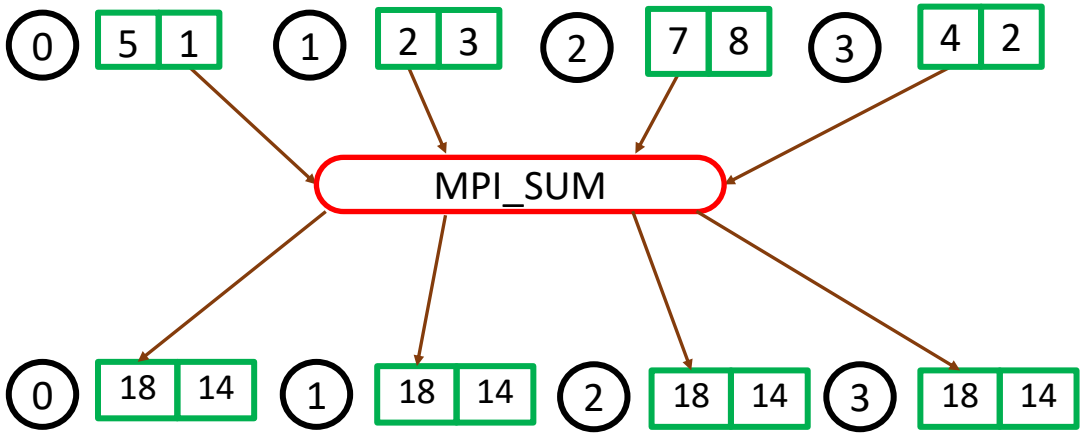


Operaciones colectivas de reducción

MPI_Reduce

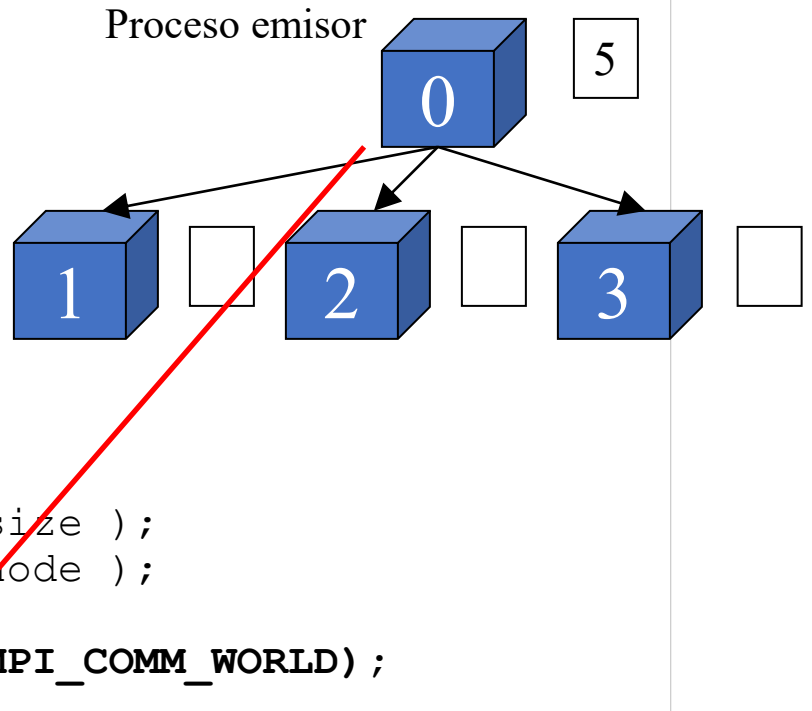


MPI_Allreduce



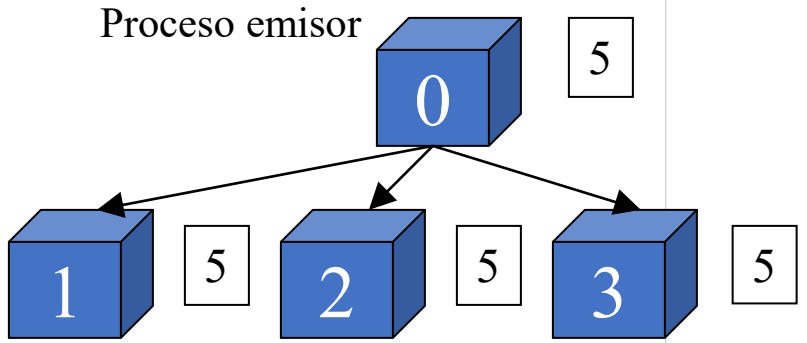
Operaciones colectivas en MPI

```
#include <stdio.h>
#include "mpi.h"

main (int argc, char **argv)
{
        int node,size;
        int tam = 255;
        char name[255];
        int num = 5;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node );

        MPI_Bcast(&num, 1, MPI_INT, 0, MPI_COMM_WORLD);

        MPI_Barrier(MPI_COMM_WORD);
        printf("El proceso %d recibe %d\n",node, num);
        MPI_Finalize();
}
```



Operaciones colectivas en MPI

```
#include <stdio.h>
#include "mpi.h"

main (int argc, char **argv)
{
        int node,size;
        int tam = 255;
        char name[255];
        int num = 5;

        MPI_Init(&argc,&argv);
        MPI_Comm_size(MPI_COMM_WORLD, &size );
        MPI_Comm_rank(MPI_COMM_WORLD, &node );

        MPI_Bcast(&num, 1, MPI_INT, 0, MPI_COMM_WORLD);

        MPI_Barrier(MPI_COMM_WORD);
        printf("El proceso %d recibe %d\n",node, num);
        MPI_Finalize();
}
```



El resto de los procesos reciben en num el mensaje enviado por el proceso 0

Ejemplo: Cálculo de π

$$\int_0^1 \sqrt{4(1-x^2)}dx = \frac{\pi}{2}$$

Cálculo secuencial $\int_0^1 \sqrt{4(1-x^2)}dx = \frac{\pi}{2}$

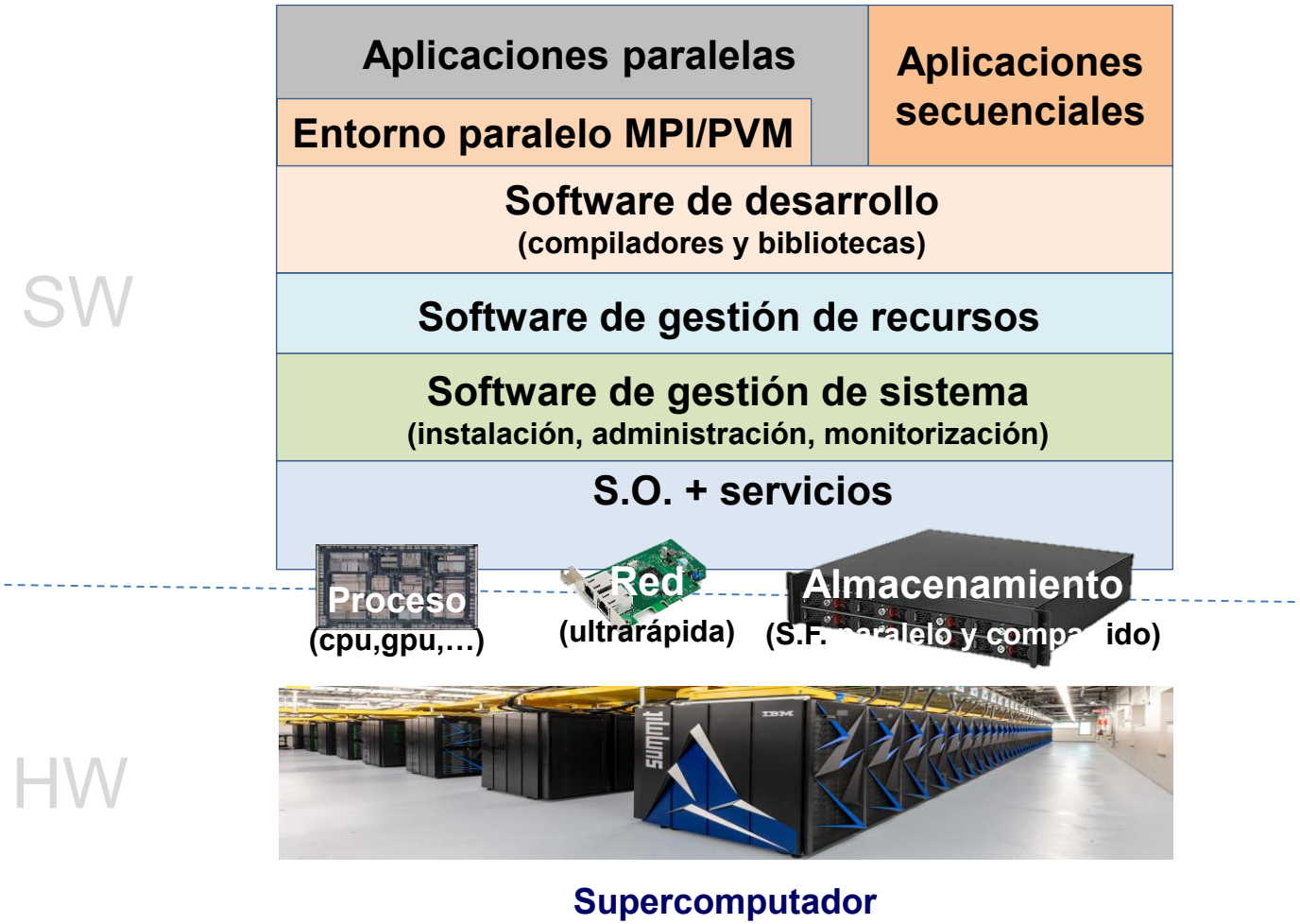
```
#include <stdio.h>
#include <math.h>

int main(int argc, char *argv[])
{
    int    n, i;
    double pi, h, sum, x;




    n   = 1000000 ; // number of intervals
    h   = 1.0 / (double) n;
    sum = 0.0;
    for (i = 0;        i <= n; i++) {
         x = h * ((double)i - 0.5);
         sum += 4.0 / (1.0 + x*x);
    }
    pi = h * sum;


        printf("pi is approximately %.16f\n", pi);

    return 0;
}
```

Cálculo paralelo

$$\int_0^1 \sqrt{4(1-x^2)}dx = \frac{\pi}{2}$$

```
#include "mpi.h"
#include <math.h>

int main(int argc, char *argv[])
{
    int    n, i, myid, numprocs;
    double pi, h, sum, x, mypi;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    n   = 1000000 ; // number of intervals
    h   = 1.0 / (double) n;
    sum = 0.0;
    for (i = myid + 1; i <= n; i += numprocs) {
         x = h * ((double)i - 0.5);
         sum += 4.0 / (1.0 + x*x);
    }
    mypi = h * sum;
    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    if (myid == 0)
        printf("pi is approximately %.16f\n", pi);
    MPI_Finalize();
    return 0;
}
```

Plataforma hardware y software



Supercomputador

Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

Plataforma hardware y software

SW

Código C/C++, Fortran

MPICH2/OpenMPI

Código C, C++, Fortran

Compiladores de GNU, Intel, PGI
BLAS, LAPACK, ACML, etc.

PBS/Torque (*batch*) + MAUI (planificador)

SSH, C3Tools, IPMI, SNMP,
Ganglia, Nagios, etc.

Linux (NTP, DNS, DHCP, TFTP, LDAP/NIS, etc.)

Nativo, virtualizado

Gigabit, Infiniband, Myrinet

NFS, LUSTRE, GPFS, GFS, SAN

HW

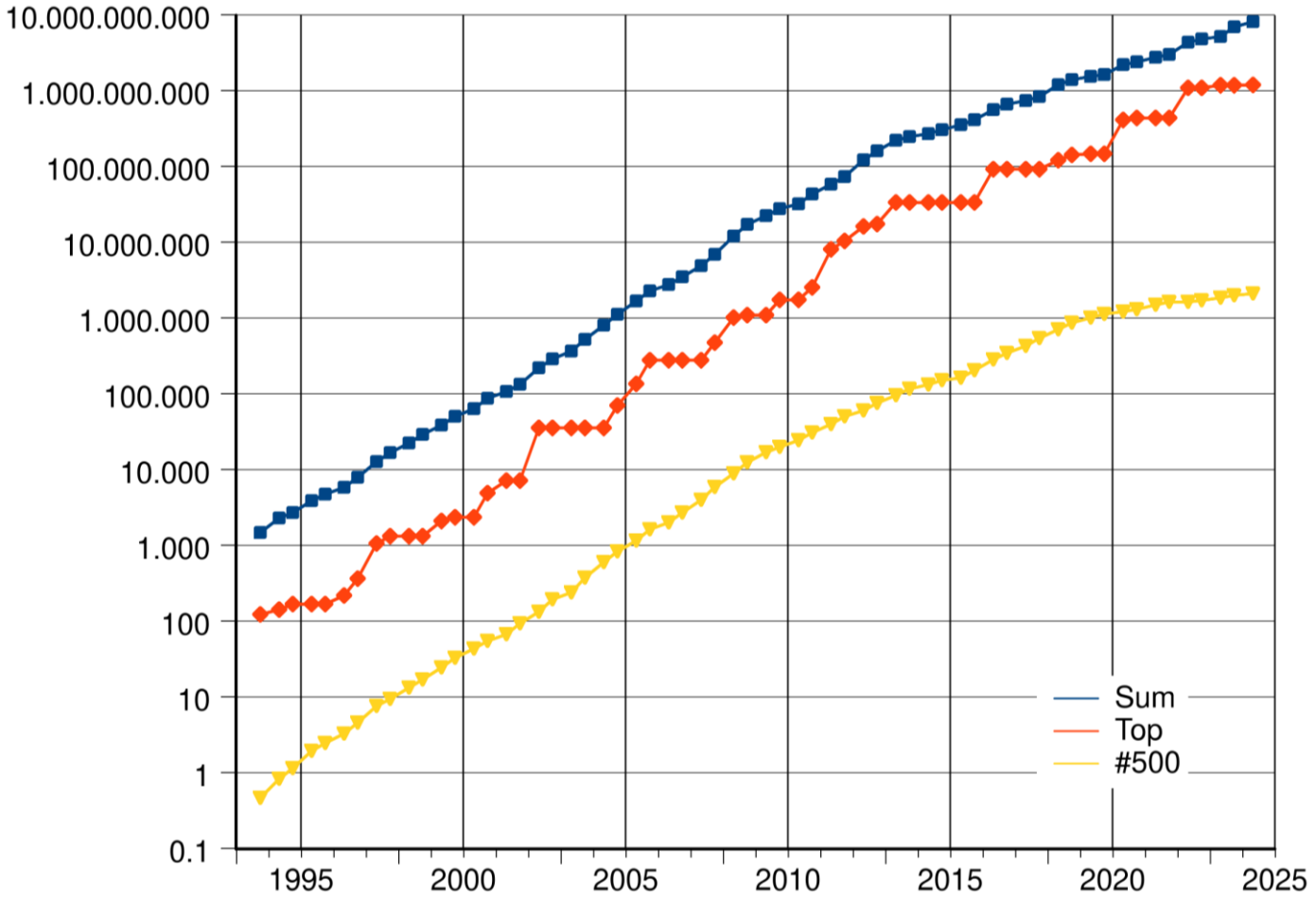
Supercomputador

Figura CC A 2.0 Generic de: https://commons.wikimedia.org/wiki/File:Summit_Supercomputer_2018.jpg
Figuras CC A-SA 4.0: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7100LC-CPU-Chip.jpg,
https://commons.wikimedia.org/wiki/File:Supermicro_AOC-SGP-I2_Gigabit_Ethernet_NIC,_PCI-Express_x4_card.jpg y
https://commons.wikimedia.org/wiki/File:VIA-NSR7800_(Front)_(3252146097).jpg

Top 500 (http://www.top500.org)

- Lista de los 500 supercomputadores más potentes del mundo (*)
  - (*) Resultados de ejecutar benchmarks (LINPACK) que miden los FLOPS y que se hacen públicos a través de la lista.
- Disponible desde: https://www.top500.org/
- Se inició en 1993 y se actualiza dos veces al año (junio y noviembre)

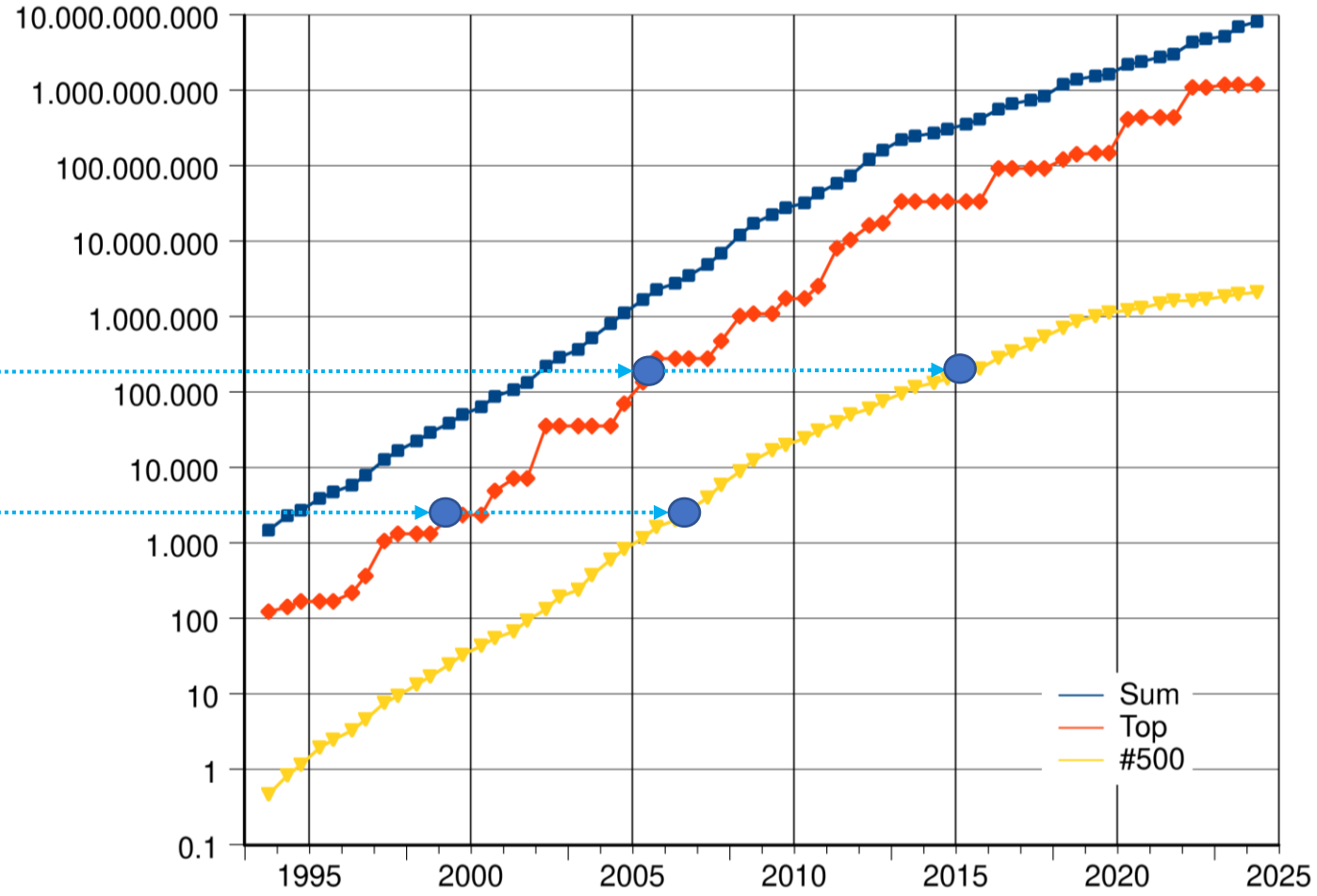
Top 500 Junio 2024 (http://top500.org/statistics/perfdevel)



Top 500 Junio 2024 (http://top500.org/statistics/perfdevel)

RTX 5090 (~138TF)

iPad Pro M4 (~4TF)



Contenidos

- **Introducción** a la computación de altas prestaciones
  - Qué, dónde y cómo
  - Hardware y software
- **Evolución** de la computación de altas prestaciones
  - Plataformas
  - Tendencias

Evolución en las plataformas de computación de altas prestaciones

- Problemas con gran cantidad de cómputo
- Más usado en ciencia y ejército
- Uso de paralelismo masivo

Supercomputadoras
(SMP, MPP, Sistólico, Array, …)

1950-1990

Evolución en las plataformas de computación de altas prestaciones

- Problemas con gran cantidad de datos tratados
- Más usado en administración
- Uso de paralelismo y alta frecuencia

Supercomputadoras & Mainframes
(SMP, MPP, Sistólico, Array, …)

1950-1990

Evolución en las plataformas de computación de altas prestaciones

- Construido por Donald Becker y Thomas Sterling en 1994 (NASA)
- Formado por 16 computadores personales con procesador intel DX4 a 200 MHz interconectados por un switch Ethernet.
- Rendimiento teórico era de 3,2 Gflops
- Posibilidad de supercomputadoras ”baratas”

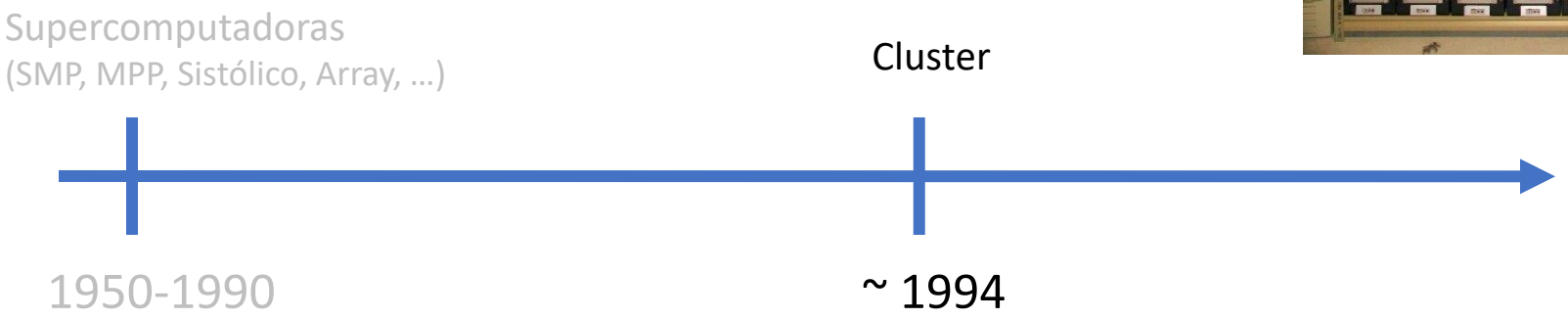


Figura CC0 1.0 de:
https://upload.wikimedia.org/wikipedia/commons/8/8c/Beowulf.jpg

Evolución en las plataformas de computación de altas prestaciones

- Construido por Donald Becker y Thomas Sterling en 1994 (NASA)
- Formado por 16 computadores personales con procesador intel DX4 a 200 MHz interconectados por un switch Ethernet.
- Rendimiento teórico era de 3,2 Gflops
- Posibilidad de supercomputadoras ”baratas”

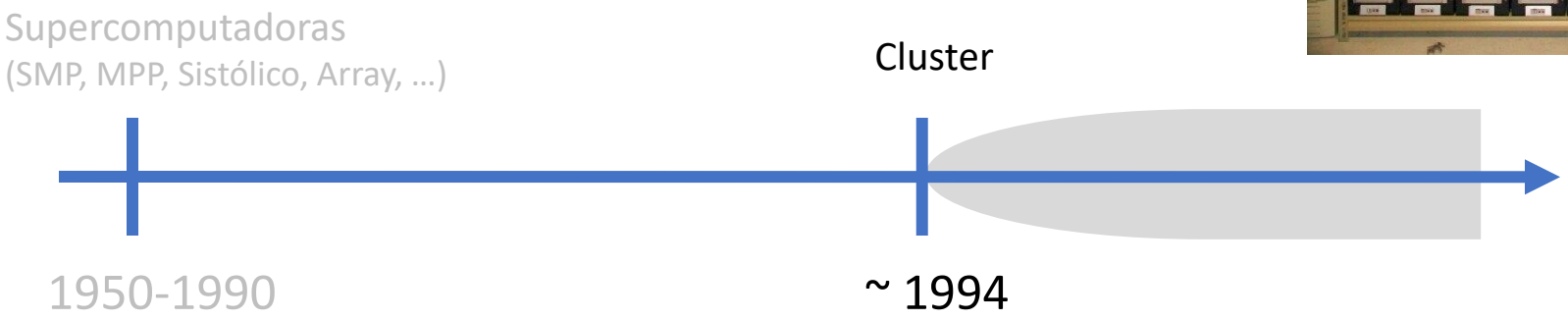
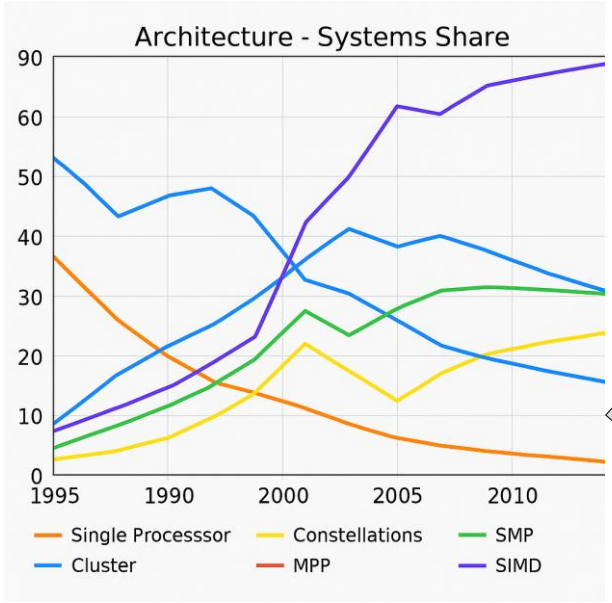


Figura CC0 1.0 de:
https://upload.wikimedia.org/wikipedia/commons/8/8c/Beowulf.jpg

Evolución en las plataformas de computación de altas prestaciones



Supercomputadoras
(SMP, MPP, Sistólico, Array, …)

Cluster

1950-1990

~ 1994

Imagen generada con chatgpt con la siguiente petición: “A partir de los datos mostrados en la página web https://www.top500.org/statistics/overtime/, genera un diagrama de dos dimensiones manteniendo el eje x con el tiempo y mostrando en el eje y líneas con el valor del porcentaje de uso (share).”

Evolución en las plataformas de computación de altas prestaciones

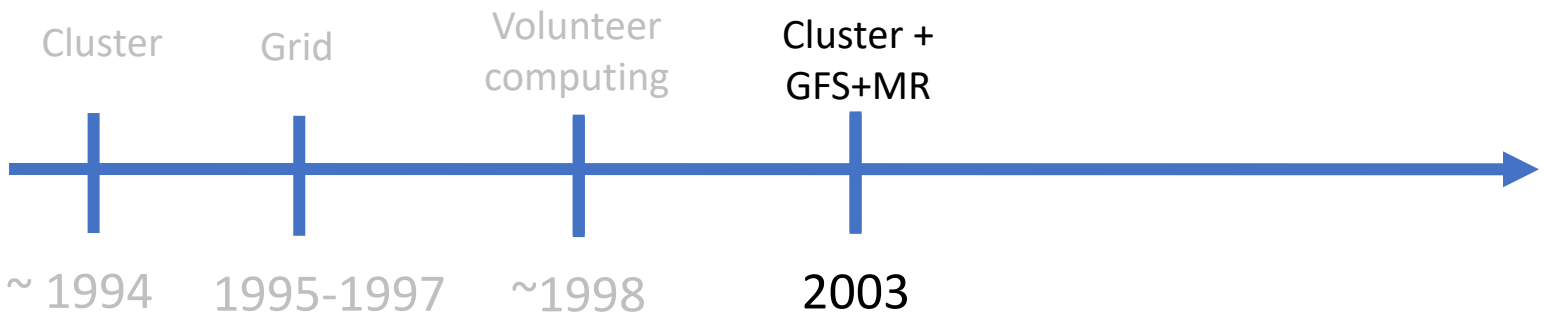
- Antecesor: metacomputing por Larry Smarr (NCSA) al inicio de los 80
  - Centros de supercomputación interconectados: más recursos disponibles
  - I-WAY demostrado en 1995
- Grid aparece en un seminario dado en 1997 en ANL por Ian Foster y Carl Kesselman

Evolución en las plataformas de computación de altas prestaciones

- Término acuñado por Luis F. G. Sarmenta (Bayanihan)
- En 1999 se lanza los proyectos SETI@home y Folding@home
- A día 4/10/2024 todos los proyectos BOINC suponen ~ 26898 TeraFLOPS

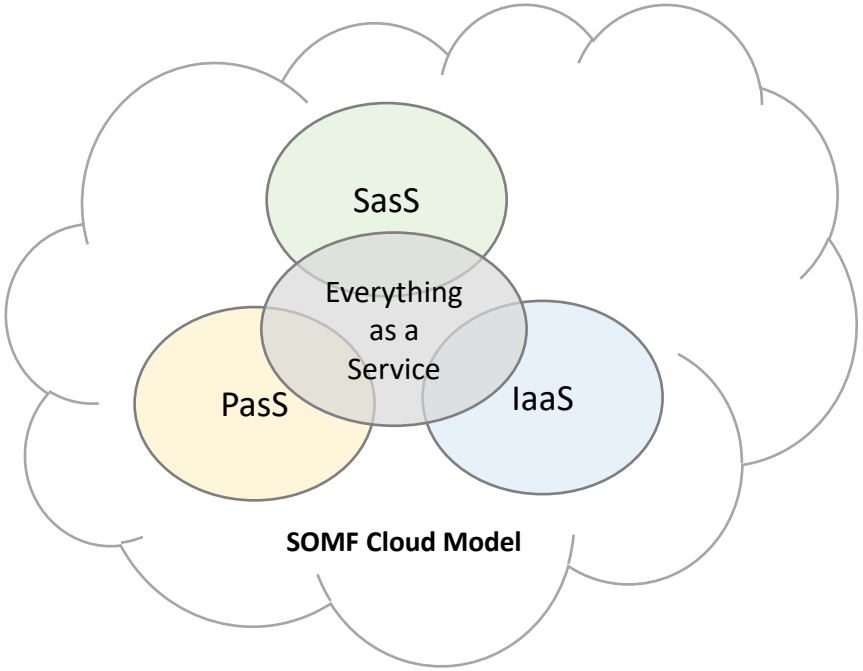
Evolución en las plataformas de computación de altas prestaciones

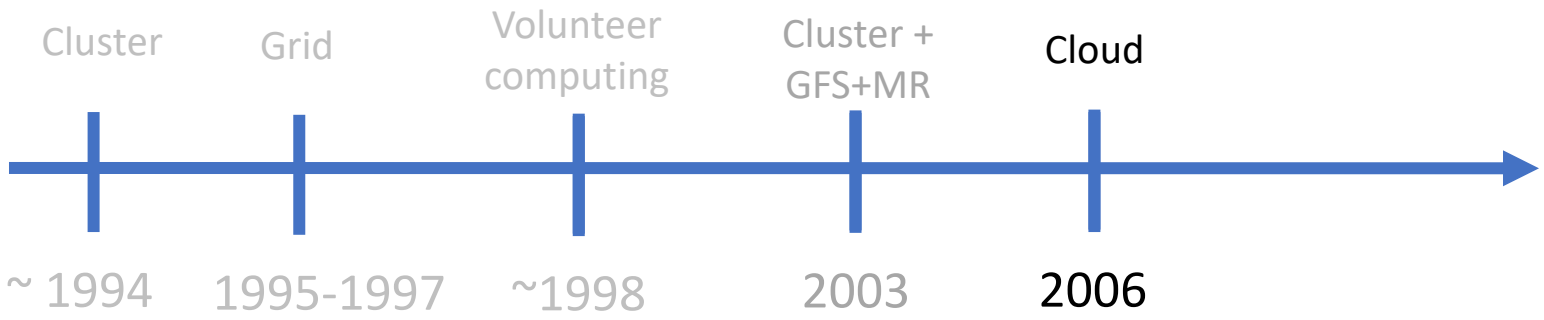
- Google presenta:
  - MapReduce como framework para trabajar con grandes conjuntos de datos: la misma función se aplica a diferentes particiones de datos (map) y después estos resultados se combinan (reduce)
  - GFS como forma de almacenar petabytes de datos (ordenadores normales, distribución escalable y tolerancia a fallos)
- GFS+MR permite a los usuarios construir “mainframes baratos” (GFS+MR vs mainframe “similar” a cluster vs supercomputador)



Evolución en las plataformas de computación de altas prestaciones

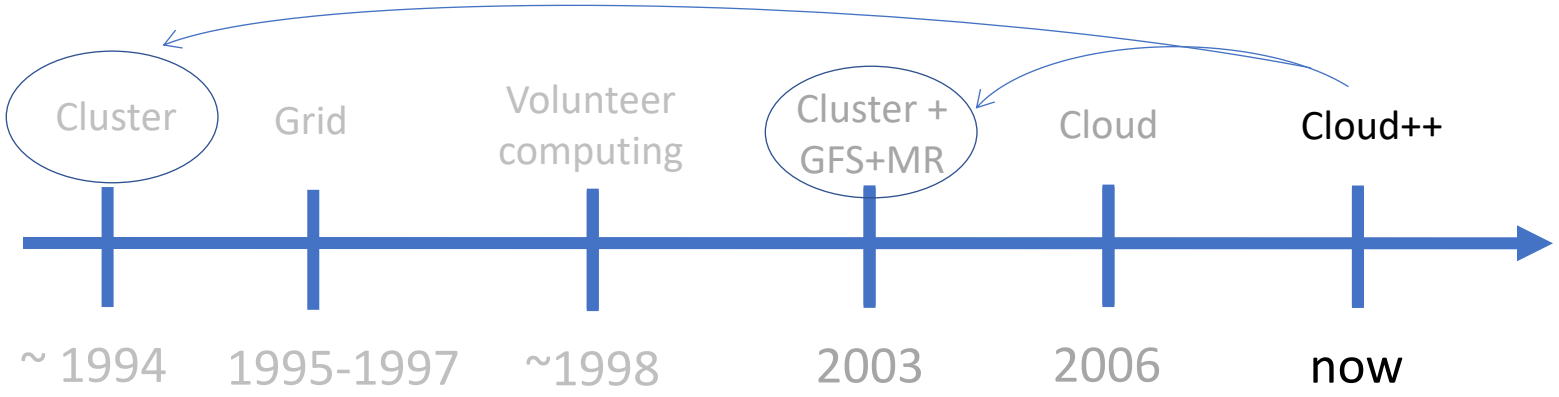
- Amazon inspira el Cloud computing actual:
  - *data centers* pensando en las compras de Navidad, el resto del tiempo se usaban “poco” -> alquilar
- Principales pilares fundamentales: computación bajo demanda y virtualización
- Principales mejoras: agilidad, coste, escalabilidad, mantenimiento, …
- Openstack: construir un cloud con un cluster

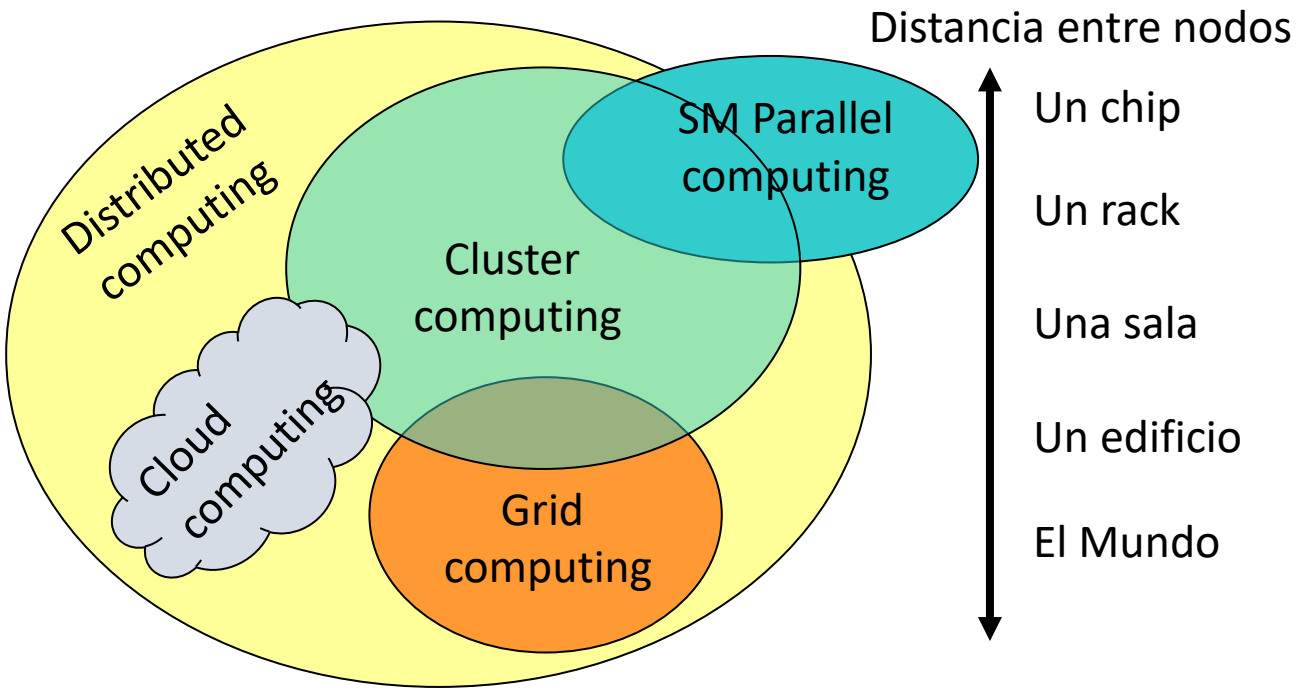


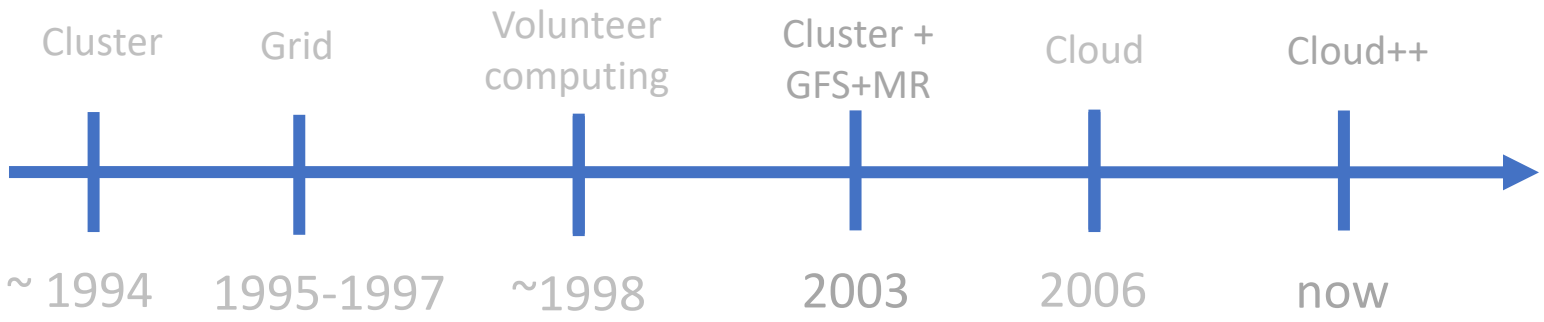


Evolución en las plataformas de computación de altas prestaciones

- Proveedor de Cloud que ofrece
  - Cluster como servicio en Cloud
  - GFS+MapReduce como servicio en Cloud



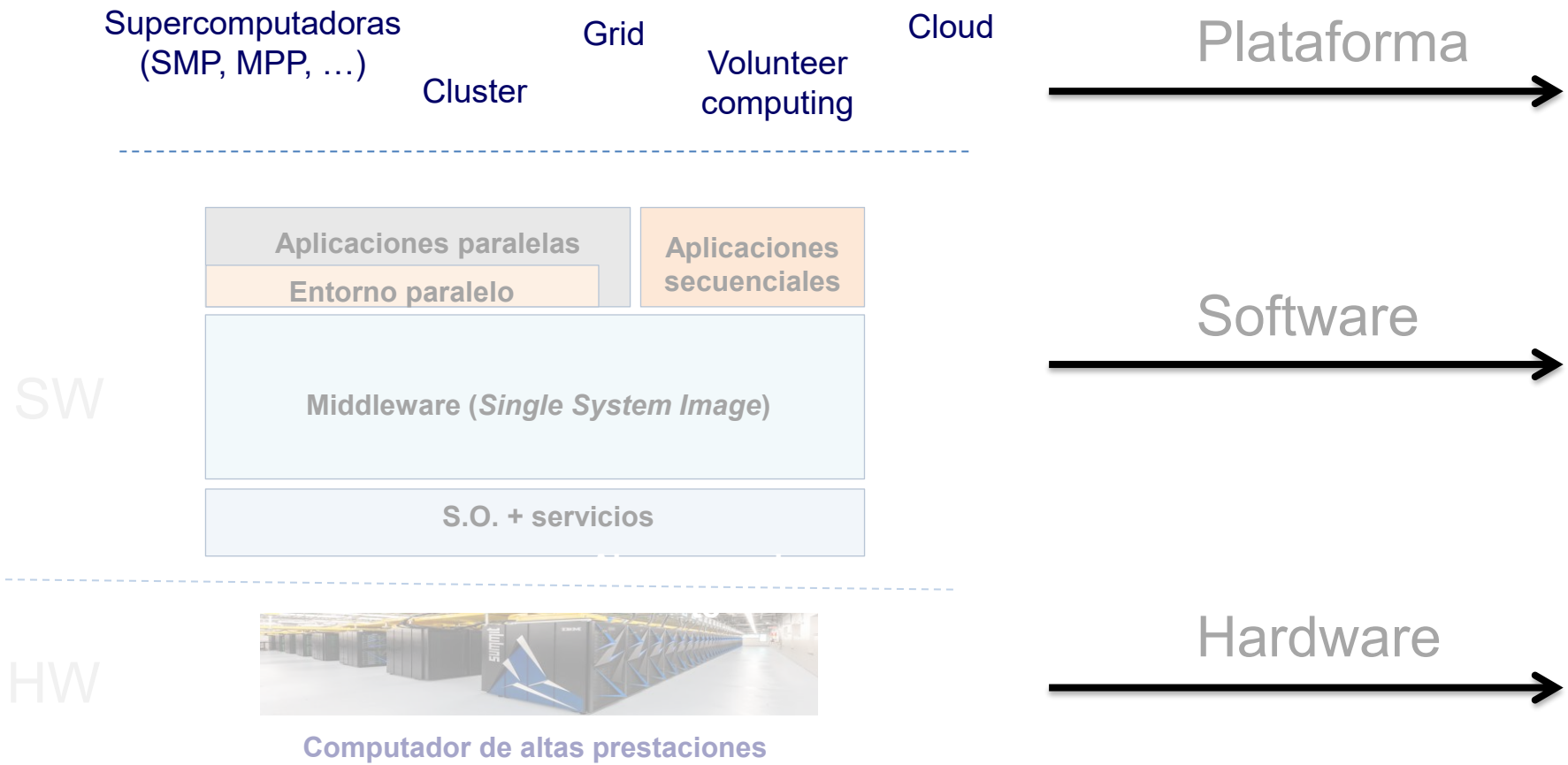

Distancia entre nodos
Distributed computing
SM Parallel computing
Cluster computing
Cloud computing
Grid computing
Un chip
Un rack
Una sala
Un edificio
El Mundo


Cluster
Grid
Volunteer computing
Cluster + GFS+MR
Cloud
Cloud++
~ 1994
1995-1997
~1998
2003
2006
now

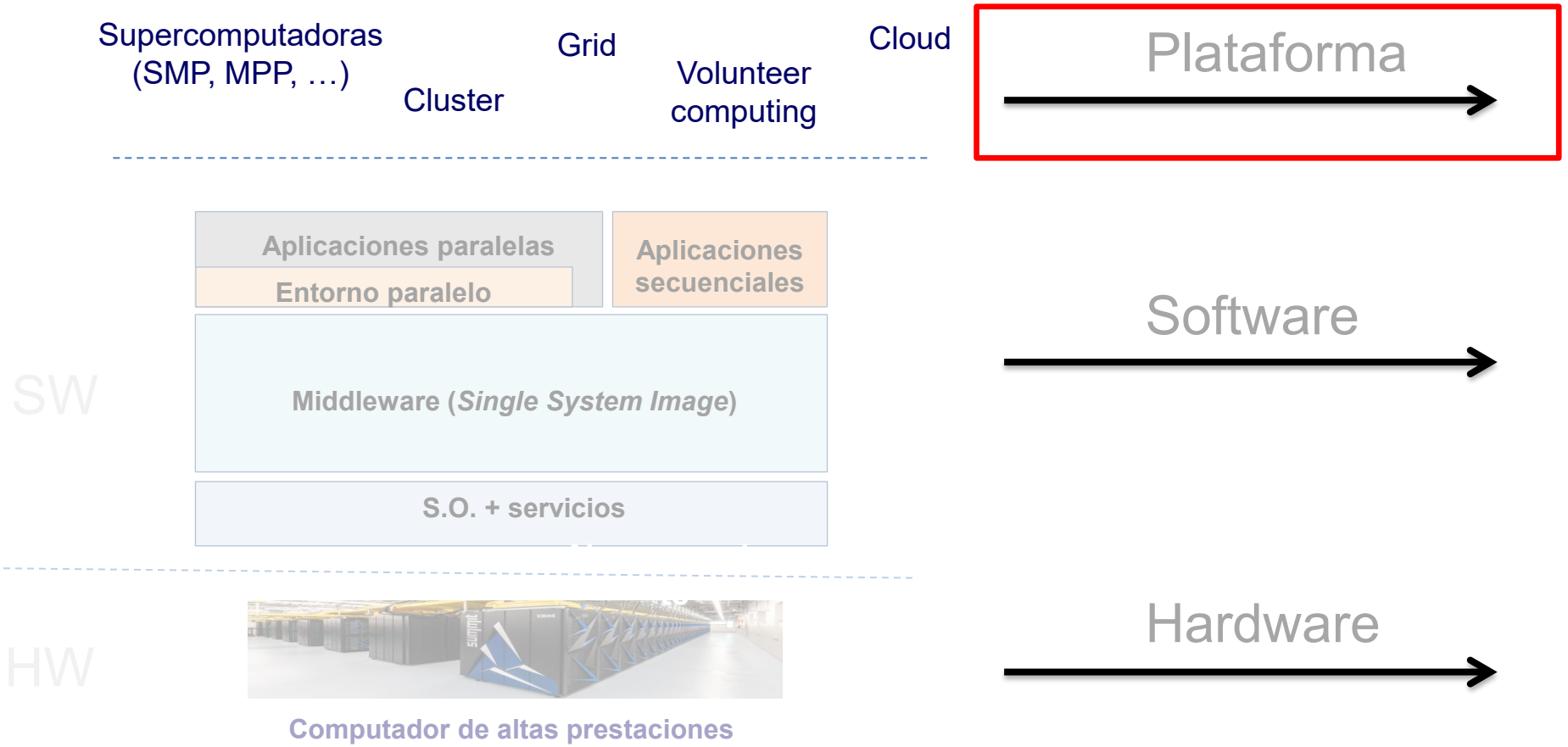
Contenidos

- **Introducción** a la computación de altas prestaciones
  - Qué, dónde y cómo
  - Hardware y software
- **Evolución** de la computación de altas prestaciones
  - Plataformas
  - Tendencias

Principales tendencias

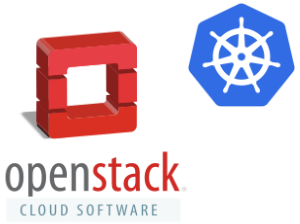


Principales tendencias



Plataforma:
uso de recursos distribuidos

- **Clouds**: empleo de recursos distribuidos alquilados bajo demanda



- **Clouds privados y públicos**:
  ajuste de infraestructura para minimizar gasto

- **Fog/Edge**:
  acercar el cloud a los dispositivos que lo usan…

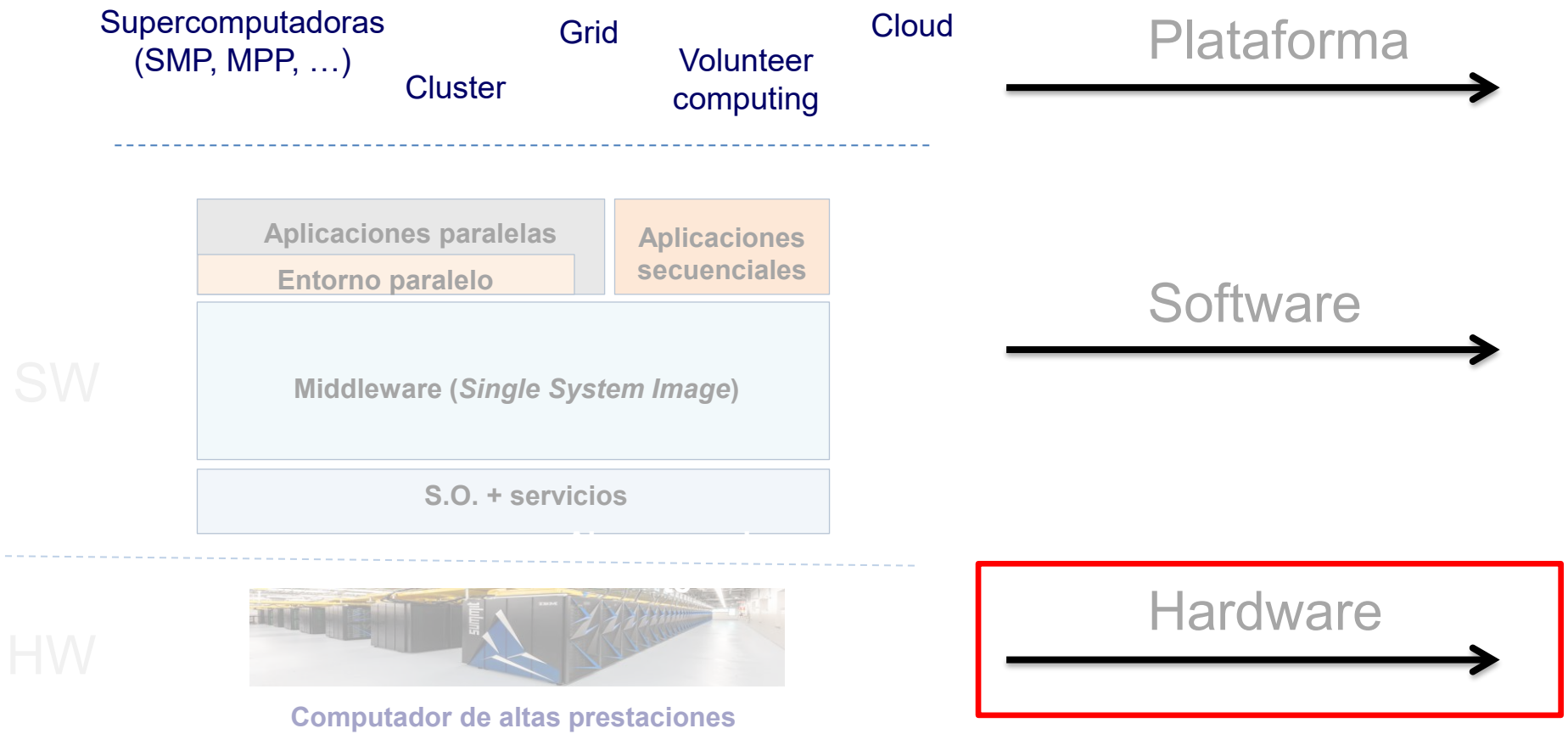
Figura OpenStack Public domain en: https://commons.wikimedia.org/wiki/File:The_OpenStack_logo.svg
Figura Kubernetes CC A 4.0 Internacional en: https://github.com/kubernetes/kubernetes/blob/master/logo/logo.svg
Figura OpenFog Public domain en: https://commons.wikimedia.org/wiki/File:OpenFog_Consortium.jpg

Plataforma:
uso eficiente de recursos

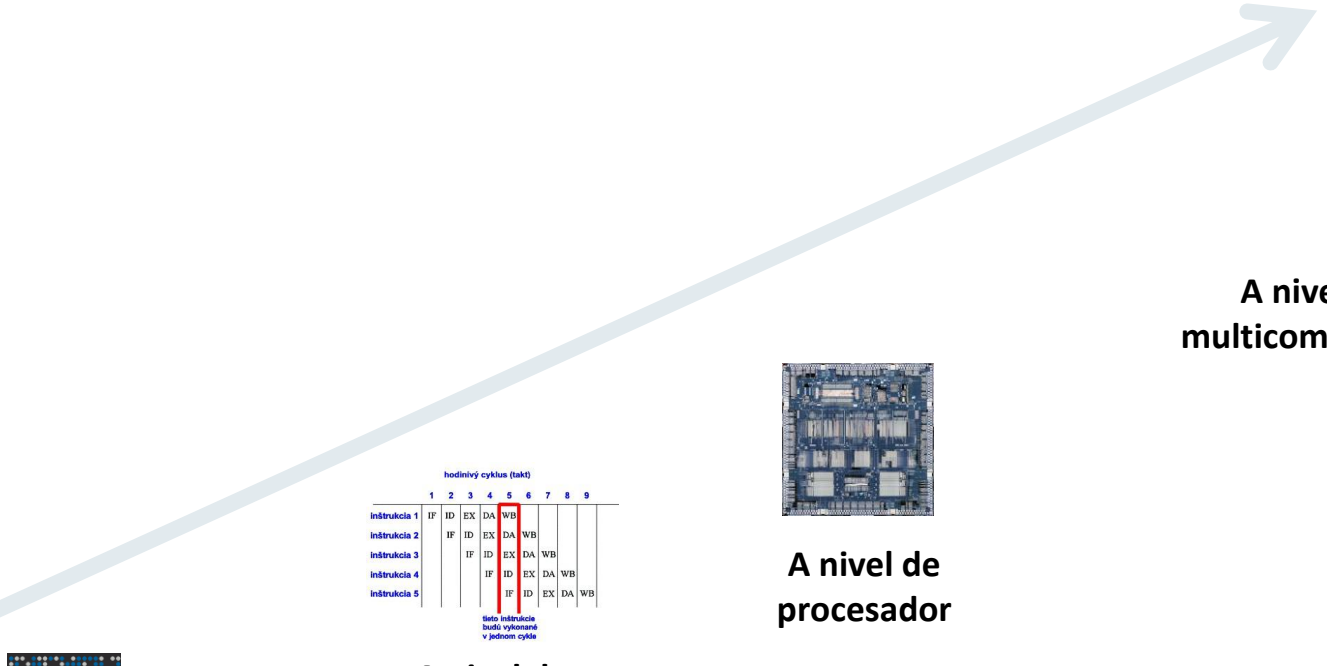
- **Green computing**: uso de recursos distribuidos de distintas organizaciones

- **Internet computing**: uso de ordenadores personales a escala global

Principales tendencias



Hardware



A nivel de bit

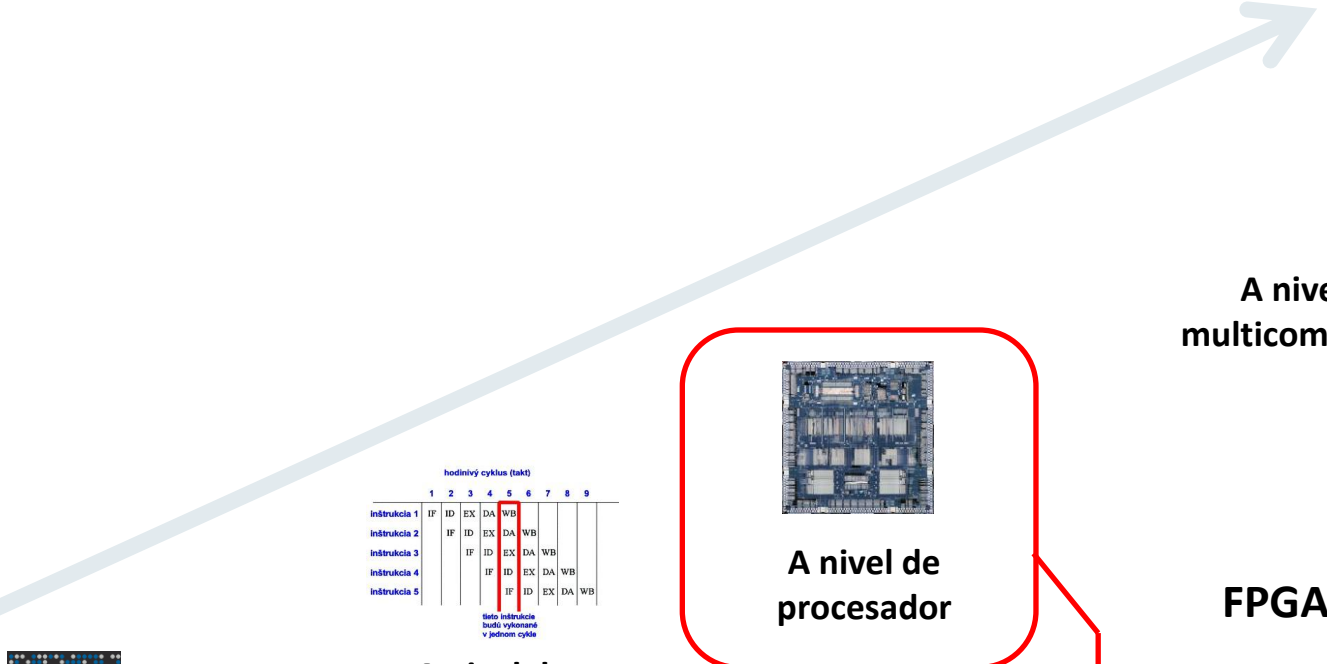
A nivel de instrucción

A nivel de procesador

A nivel de multicomputador

Figura CC BY-SA 4.0 en: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7000-CPU-Chip.jpg
Figura Public domain en: https://commons.wikimedia.org/wiki/File:Pipeline_tact.jpg
Figura CC BY-SA 3.0 en: https://commons.wikimedia.org/wiki/File:Ternary_wiki.gif

Hardware



A nivel de bit

A nivel de instrucción

A nivel de procesador

A nivel de multicomputador

FPGA, SoC

- Hardware específico
- **Hardware multicore**

CPU, GPU, etc.

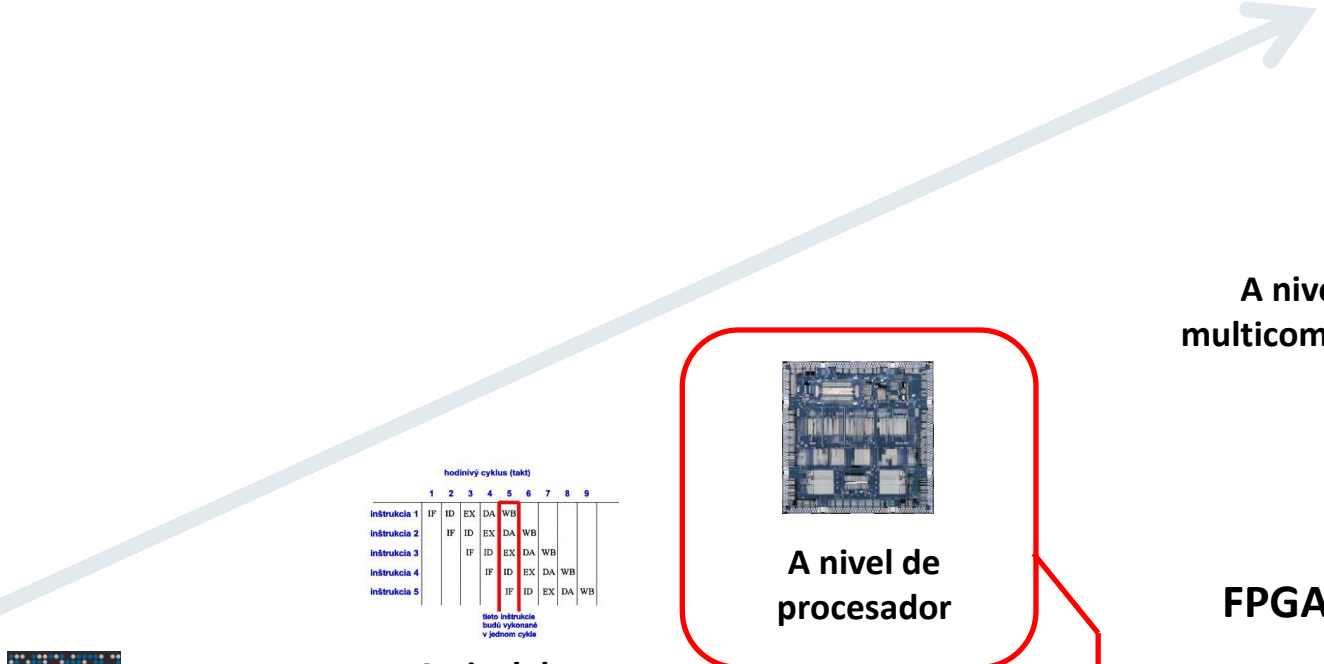
Figura CC BY-SA 4.0 en: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7000-CPU-Chip.jpg
Figura Public domain en: https://commons.wikimedia.org/wiki/File:Pipeline_tact.jpg
Figura CC BY-SA 3.0 en: https://commons.wikimedia.org/wiki/File:Ternary_wiki.gif

Hardware multicore

- **Tarjetas gráficas**: uso de la capacidad de procesamiento de las potentes tarjetas gráficas actuales
  - CUDA: entorno de programación orientado a las tarjetas gráficas de Nvidia
  - OpenCL: C99 extendido para operaciones vectoriales y eliminando ciertas funcionalidades

- **Procesadores heterogéneos**: gran cantidad de “cores” especializados (Ampere one, etc.)
  - <memoria compartida>: OpenMP, OpenACC, OpenCL
  - <paso de mensaje>: MPI

Hardware



A nivel de bit

A nivel de instrucción

A nivel de procesador

A nivel de multicomputador

FPGA, SoC

- **Hardware específico**
- Hardware multicore

CPU, GPU, etc.

Figura CC BY-SA 4.0 en: https://commons.wikimedia.org/wiki/File:HP-HP9000-PARISC-PA7000-CPU-Chip.jpg
Figura Public domain en: https://commons.wikimedia.org/wiki/File:Pipeline_tact.jpg
Figura CC BY-SA 3.0 en: https://commons.wikimedia.org/wiki/File:Ternary_wiki.gif

Hardware específico

- **Memoria “activa”**: computo simple en la propia memoria
  - Automata processor
- **Coprocesadores cuánticos:**
  - qubit-chip de D-Wave, IBM, IonQ, etc.
- **Computadores ópticos:**
  - Q.ANT
- **Etc.**

Principales tendencias

Supercomputadoras (SMP, MPP, …)

Cluster

Grid

Volunteer computing

Cloud

Plataforma →

SW

Aplicaciones paralelas

Entorno paralelo

Aplicaciones secuenciales

Middleware (*Single System Image*)

S.O. + servicios

Software →

HW

Computador de altas prestaciones

Hardware →

Software

- Vectoriales
  - SSE, AVX, AVX2, …
- Multiprocesador
  - UMA, NUMA
    - OpenMP,
    - iTBB, …
  - M. Distribuida
    - MPI,…
    - Map-reduce

- **Integrar soluciones vectoriales y multiprocesador (dentro de las herramientas de desarrollo)**

Software

- Vectoriales
  - SSE, AVX, AVX2, …
- Multiprocesador
  - UMA, NUMA
    - OpenMP,
    - iTBB, …
  - M. Distribuida
    - MPI,…
    - Map-reduce

- Integrar soluciones vectoriales y multiprocesador (dentro de las herramientas de desarrollo)
- **Integrar soluciones de memoria compartida y paso de mensaje con ayuda del sistema operativo.**

Software

- Vectoriales
  - SSE, AVX, AVX2, …
- Multiprocesador
  - UMA, NUMA
    - OpenMP,
    - iTBB, …
  - M. Distribuida
    - MPI,…
    - Map-reduce

- Integrar soluciones vectoriales y multiprocesador (dentro de las herramientas de desarrollo)
- Integrar soluciones de memoria compartida y paso de mensaje con ayuda del sistema operativo.
- **Buscar perfiles simplificados que permitan la mayor escalabilidad posible.**

Sistemas distribuidos:
Computación de altas prestaciones

- Google:
  - Modelo MapReduce
  - Sistemas de ficheros de Google
  - Algoritmos de clasificación (K-Means + Canopy)
  - Etc.

- Facebook/Meta

Bibliografía

- **Parallel Computer Architectures: a Hardware/Software Approach.**
  D.E. Culler, J.P. Singh, with A. Gupta
  - Capítulo 1

- **Organización y Arquitectura de Computadores (5ta. ed.)**
  William Stallings
  - Capítulo 16: Procesamiento Paralelo.

- **Organización de Computadoras (4ta. ed.)**
  Andrew S. Tanenbaum
  - Capítulo 8: Arquitecturas de computadoras paralelas.

Bibliografía

- **GPU + CPU**
  - https://medium.com/@prabhuss73/cpu-vs-gpu-architectural-differences-performance-metrics-and-use-cases-ccb44c018e2a
- **Cluster**
  - http://www.democritos.it/~baro/slides/LAT-HPC-GRID-2009/Part1.pdf
- **TOP500 Supercomputer Sites**
  - http://www.top500.org/
- **Beowulf**
  - http://www.beowulf.org/overview/index.html

uc3m | Universidad Carlos III de Madrid

OpenCourseWare

Sistemas Paralelos y Distribuidos

Félix García Carballeira

Alejandro Calderón Mateos

Sistemas de altas prestaciones en entornos distribuidos