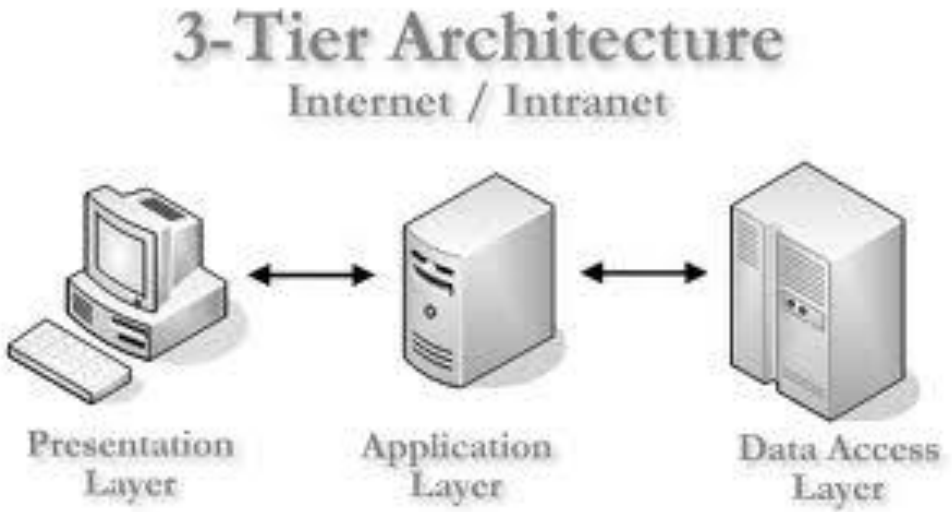
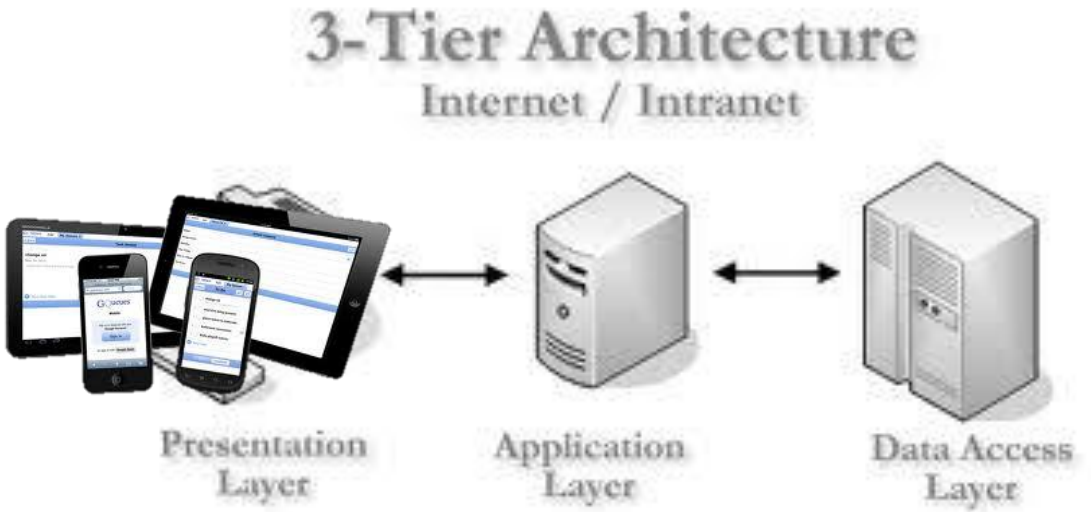
uc3m | Universidad Carlos III de Madrid

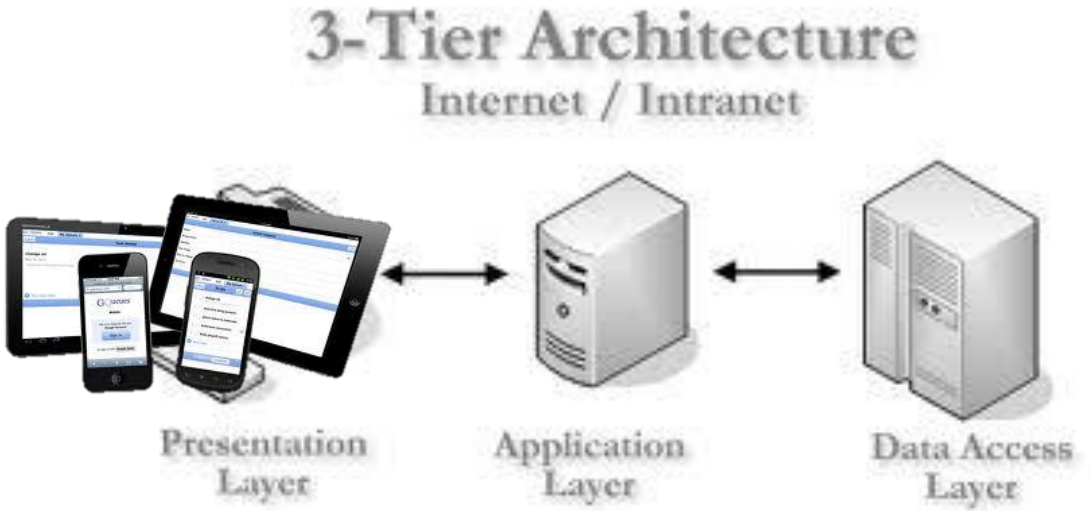
Sistemas Paralelos y Distribuidos

Félix García Carballeira
Alejandro Calderón Mateos

Sistemas de altas prestaciones en entornos distribuidos

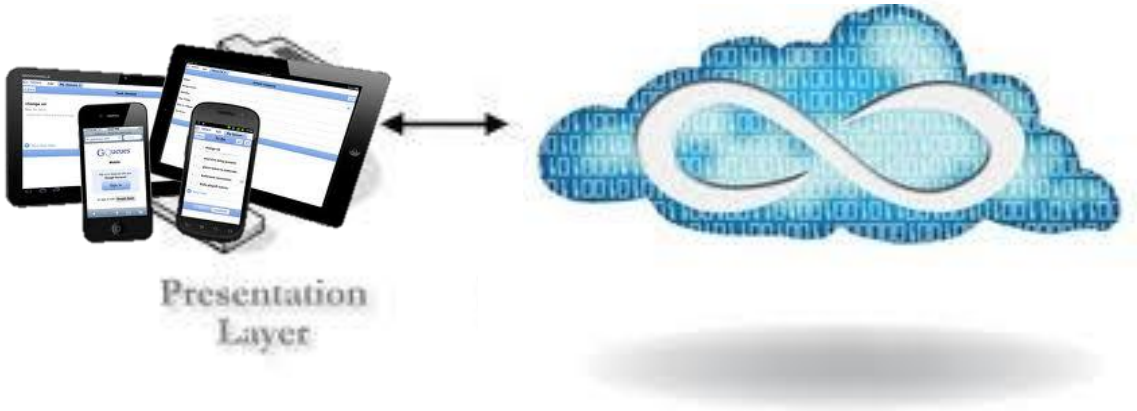


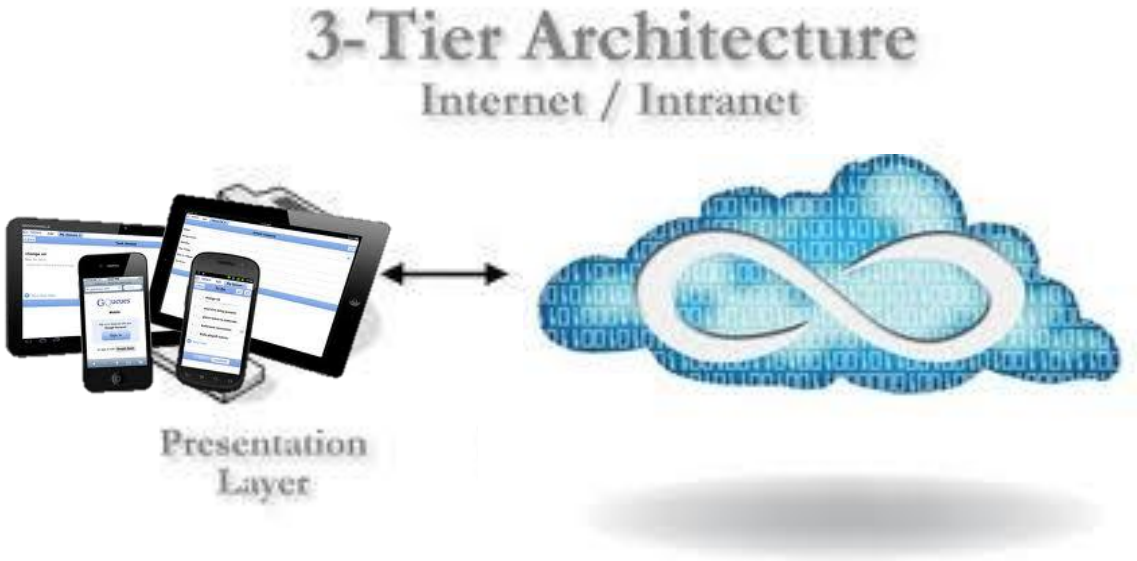


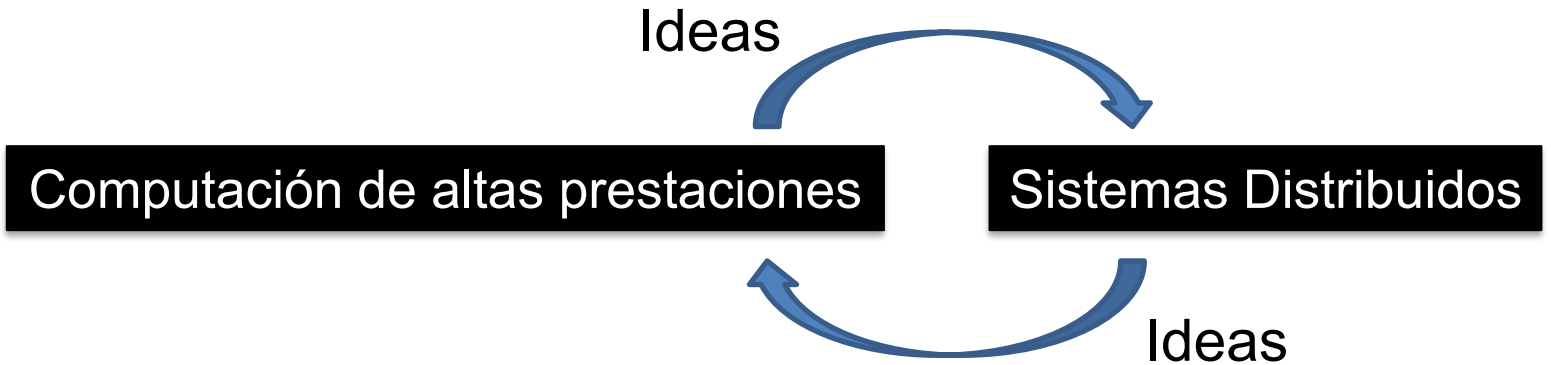


3-Tier Architecture

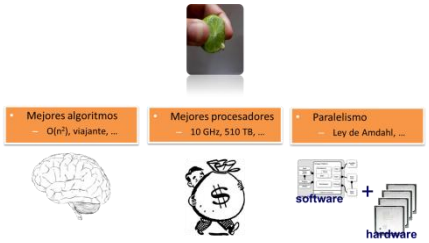
Internet / Intranet

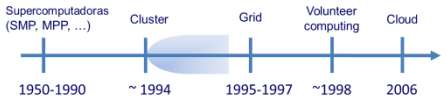



3-Tier Architecture
Internet / Intranet
Presentation
Layer


Ideas
Computación de altas prestaciones
Sistemas Distribuidos
Ideas

Agenda





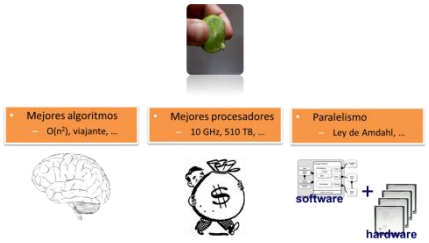
Introducción a la computación de altas prestaciones

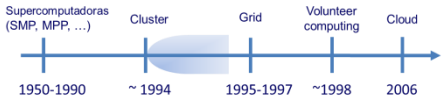
- Qué, dónde y cómo
- Hardware y software

Evolución de la computación de altas prestaciones

- Plataformas
- Tendencias

Agenda





Introducción a la computación de altas prestaciones

- Qué, dónde y cómo
- Hardware y software

Evolución de la computación de altas prestaciones

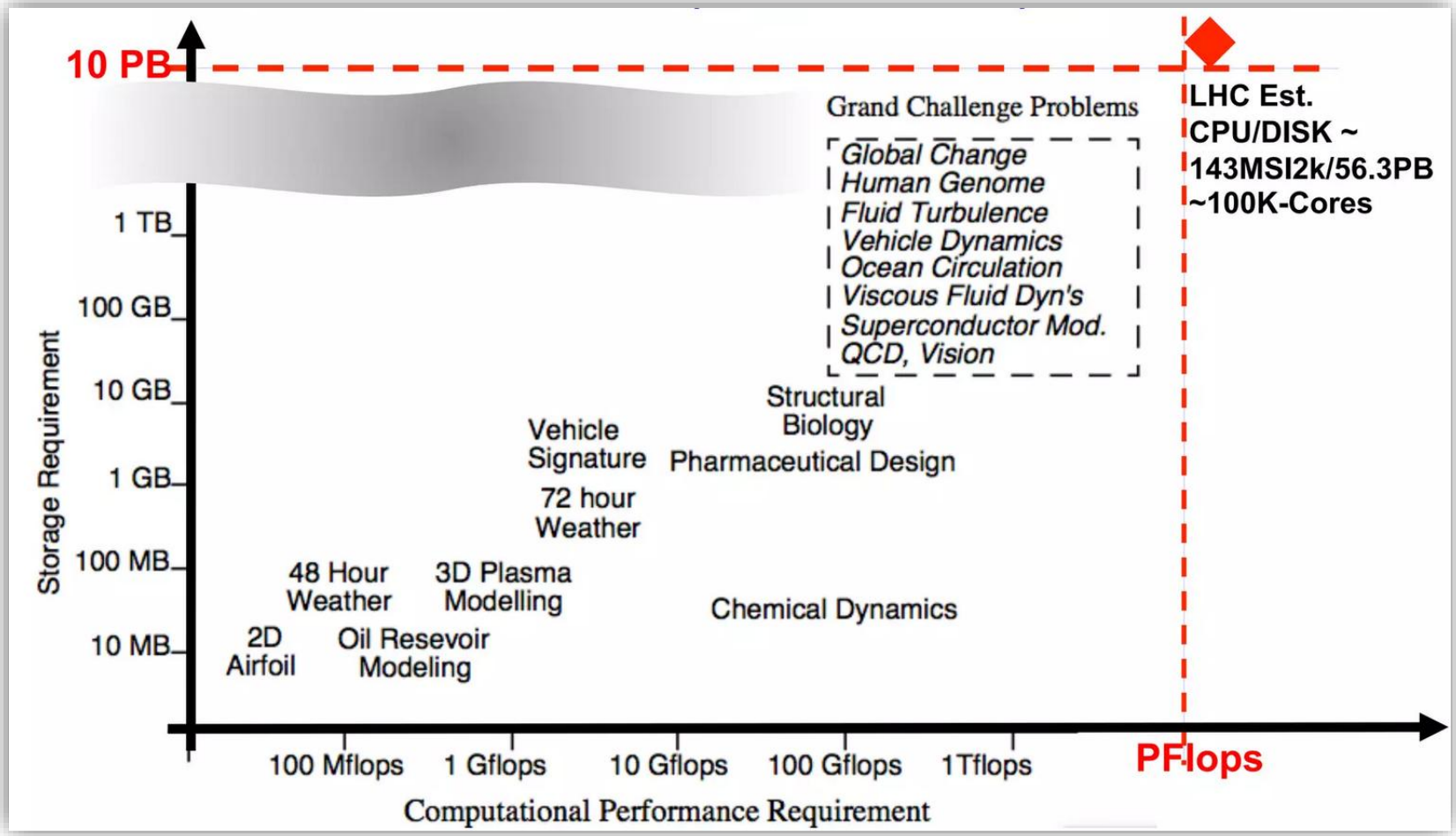
- Plataformas
- Tendencias

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.

¿Dónde se necesita?

[Culler99]

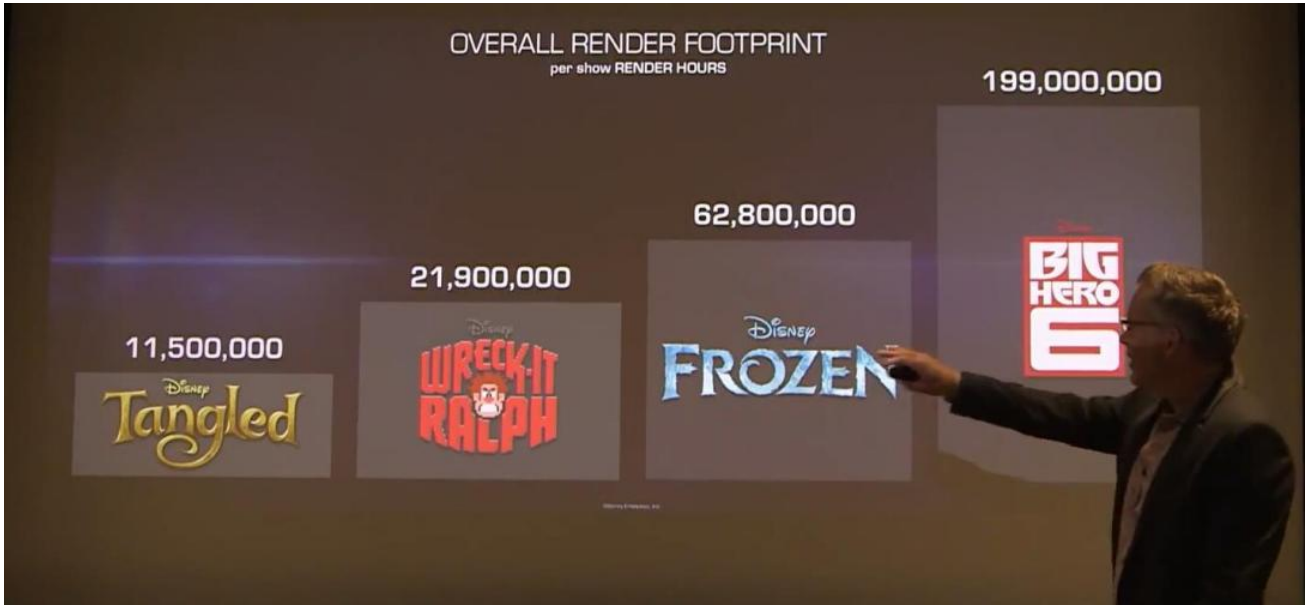


https://www.slideshare.net/hlshih/high-performance-computing-building-blocks-production-perspective

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."**

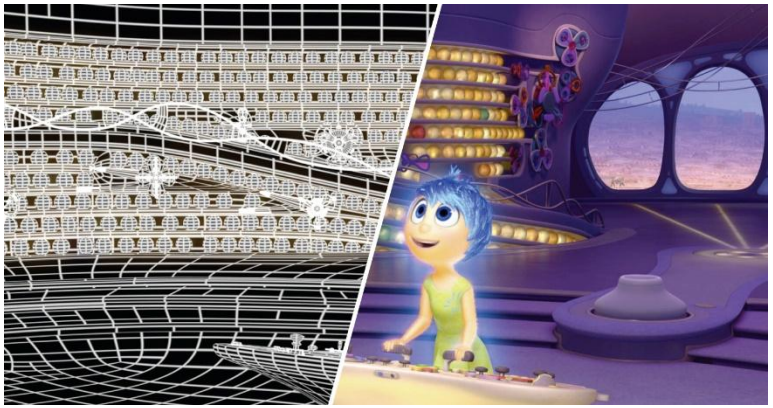


Ejemplo 2: Monster's University (2022)…

(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, …

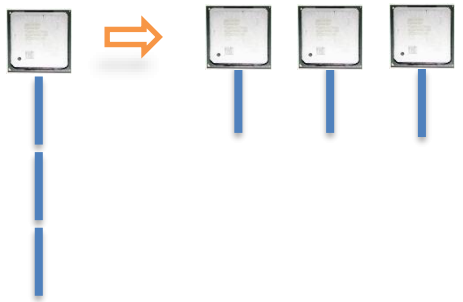
¿Cómo se consigue más velocidad?

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

- Mejores procesadores (mejoras en la tecnología)
  - CPU a 10 GHz, 510 TB de RAM, …

¿Cómo se consigue más velocidad?

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

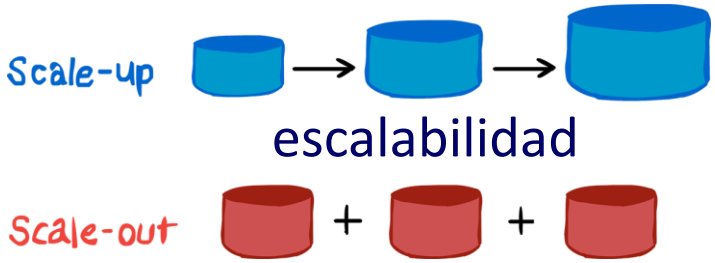


- Mejores procesadores (mejoras en la tecnología)
  - CPU a 10 GHz, 510 TB de RAM, …

- Paralelismo (mejoras en el uso de la tecnología actual)
  - Speedup, Ley de Amdahl, …

¿Eso del paralelismo qué implica?

- 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, …



https://cdn.turbonomic.com/wp-content/uploads/ScaleUpScaleOut.png

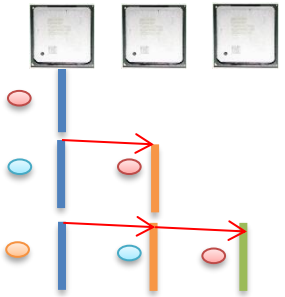
Tipos de paralelismo

- Tareas independientes:

Tipos de paralelismo

- Tareas independientes:

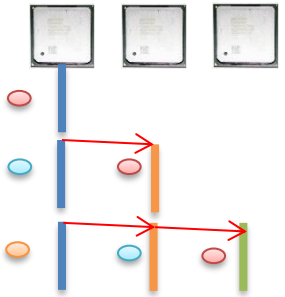
- Tareas cooperativas:
  - *Pipeline*
  - Coordinación (*mutex* y *conditions*)



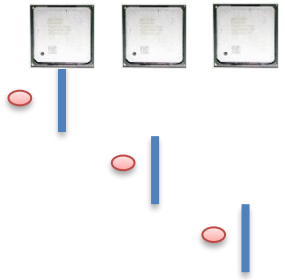
Tipos de paralelismo

- 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:

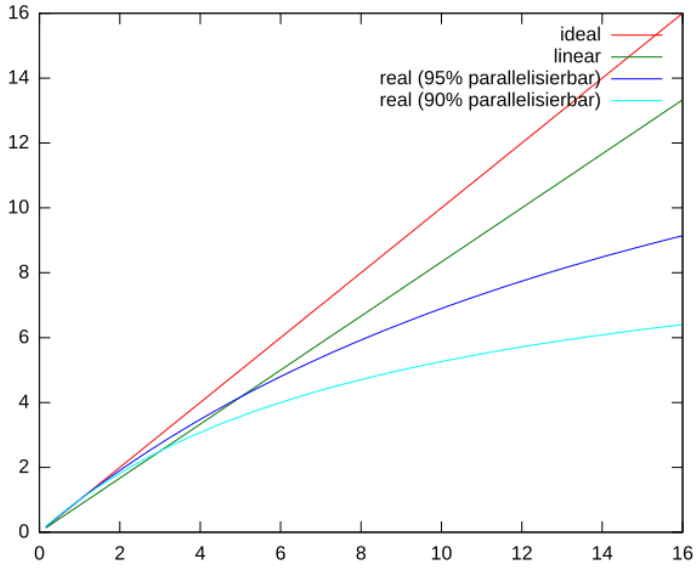
  speedup = tiempo_de_ejecución (1) / tiempo_de_ejecución (**n**)

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:



https://commons.wikimedia.org/wiki/File:Speedup.svg

Ley de Amdahl

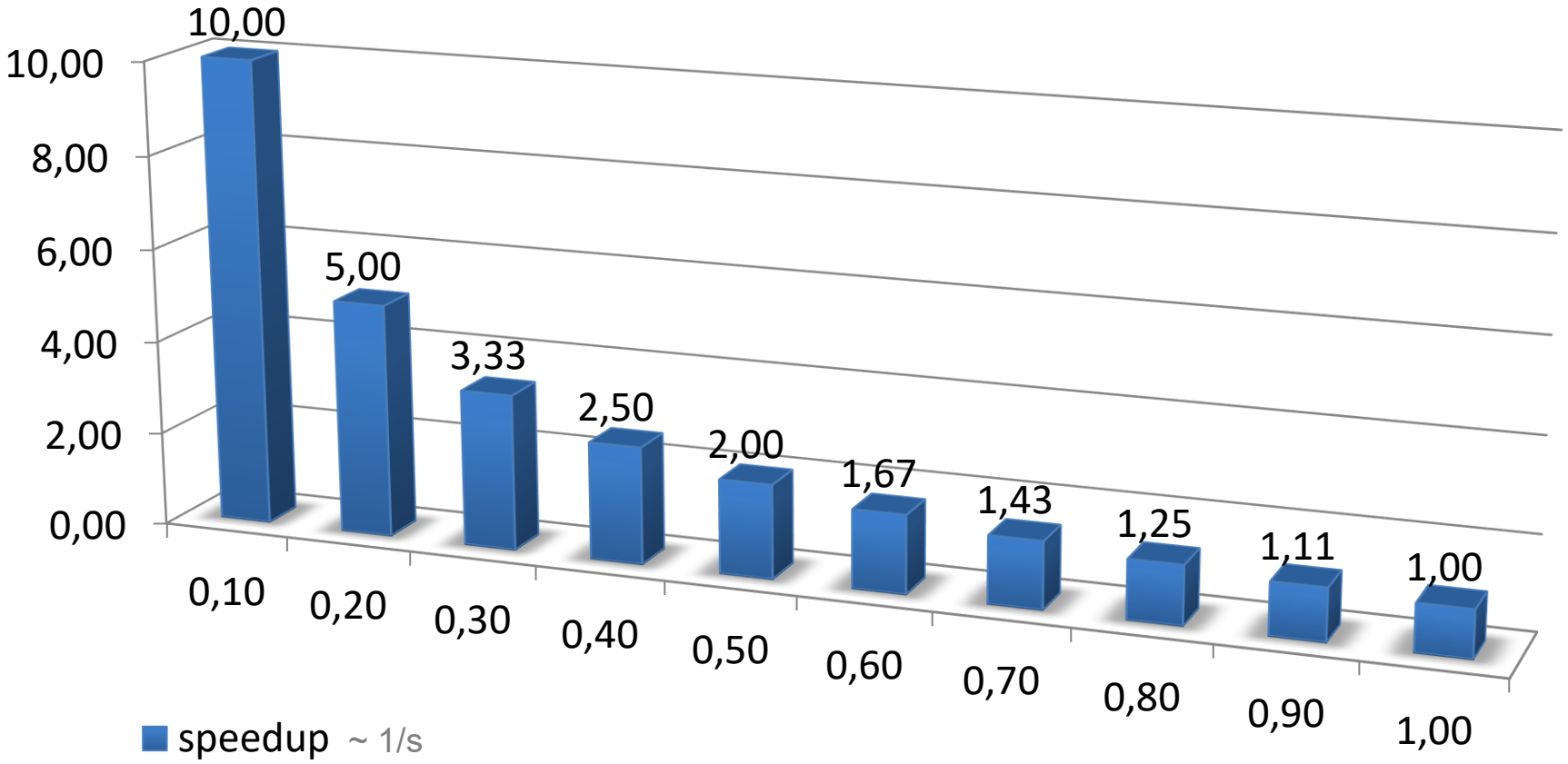
- Ley de Amdahl:

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

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

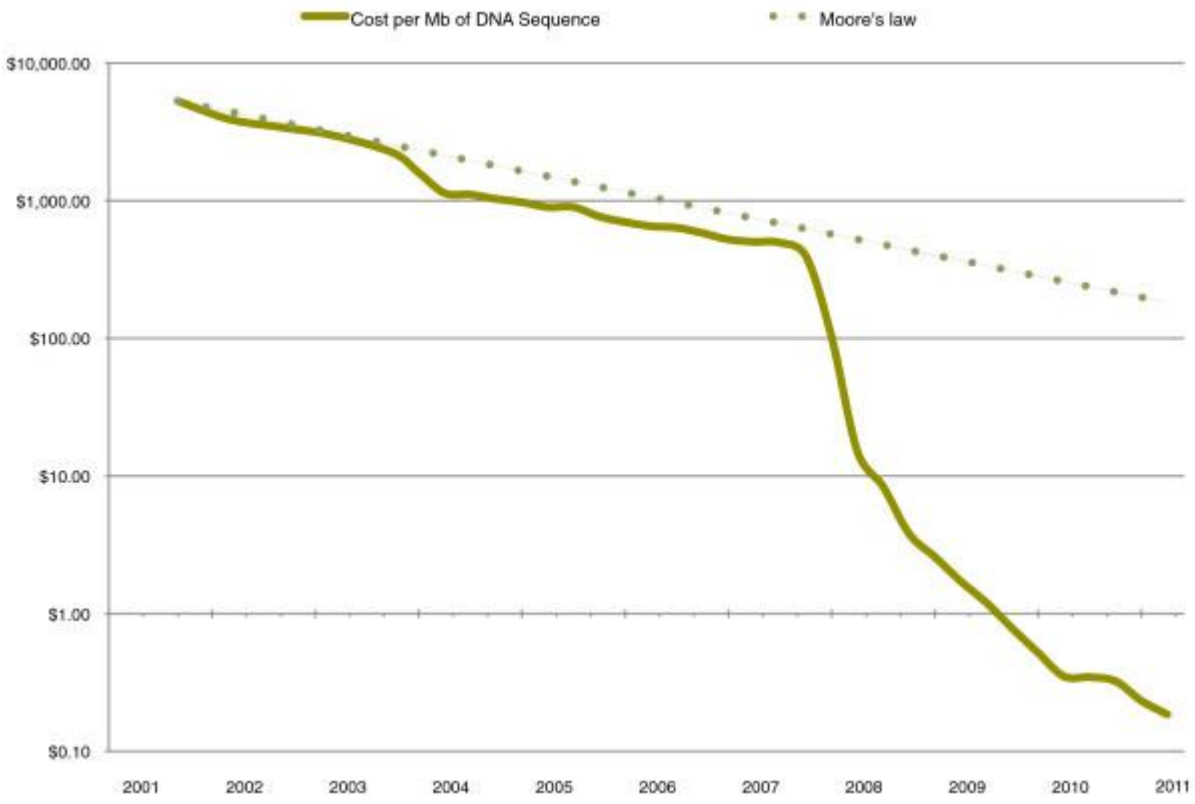
SI **n**↑ ENTONCES ***speedup* ~ 1 / s**

Ley de Amdahl



¿Eso del paralelismo ayuda?

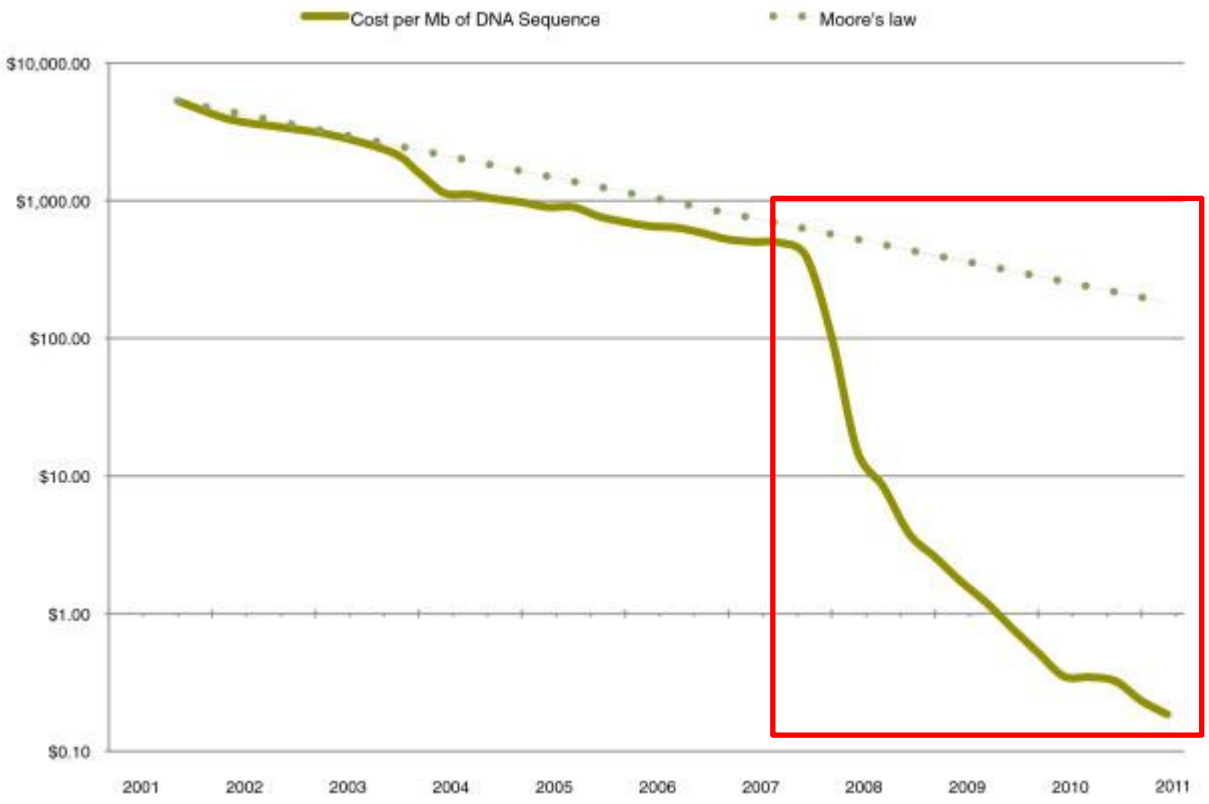
caso de estudio: genoma humano



http://genomebiology.com/2011/12/8/125
http://pressroom.nvidia.com/easyir/customrel.do?easyirid=A0D622CE9F579F09&prid=878712&releasejsp=release_157

¿Eso del paralelismo ayuda?

caso de estudio: genoma humano

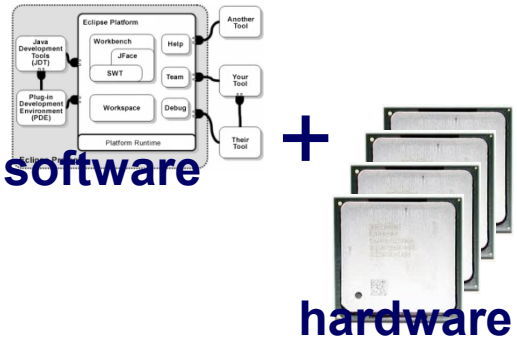


Yes!

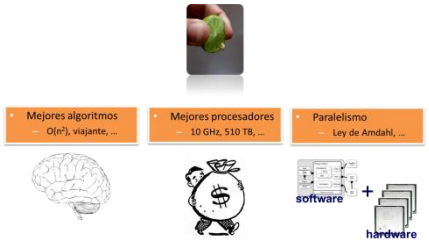
http://genomebiology.com/2011/12/8/125
http://pressroom.nvidia.com/easyir/customrel.do?easyirid=A0D622CE9F579F09&prid=878712&releasejsp=release_157

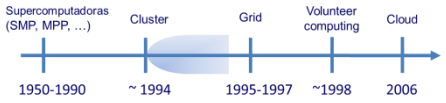
Computación de altas prestaciones

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



Agenda





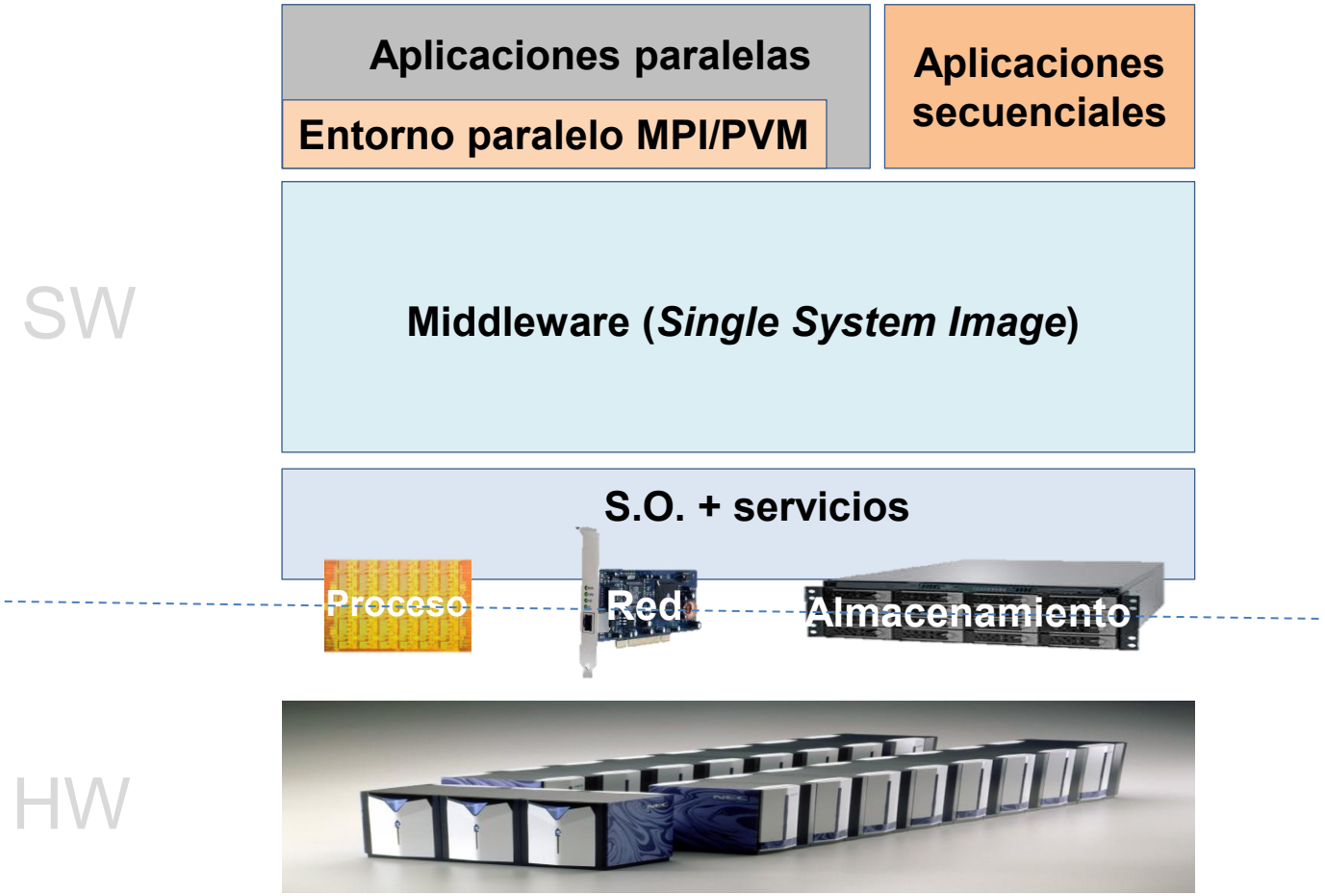
Introducción a la computación de altas prestaciones

- Qué, dónde y cómo
- Hardware y software

Evolución de la computación de altas prestaciones

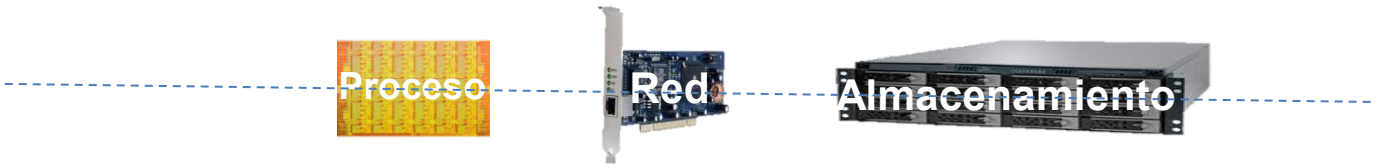
- Plataformas
- Tendencias

Plataforma hardware y software



Computador de altas prestaciones

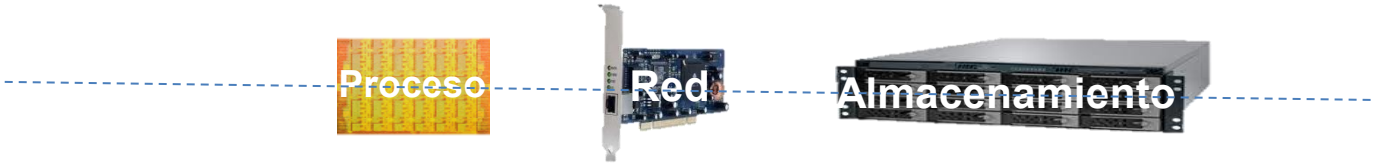
Plataforma hardware



HW

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

Plataforma hardware

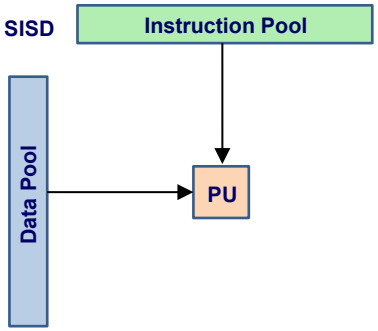
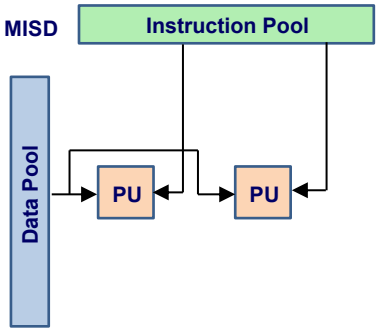
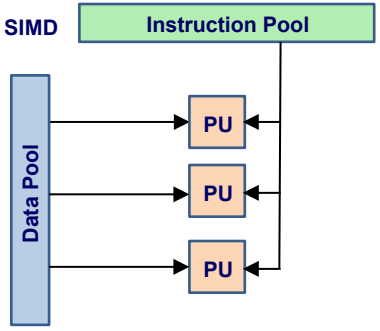
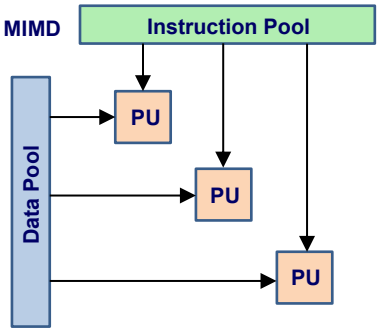


HW

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

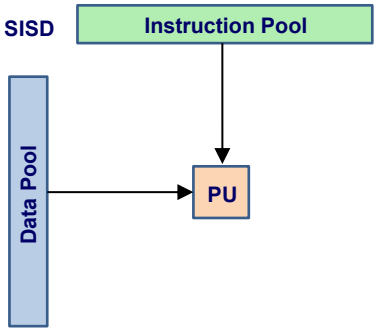
Taxonomía de Flynn

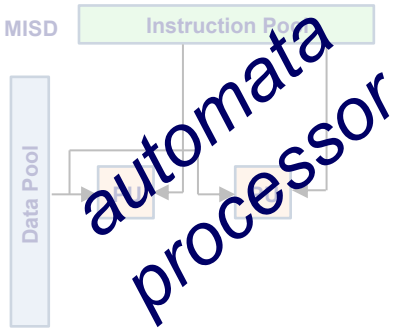
http://www.buyya.com/microkernel/chap1.pdf

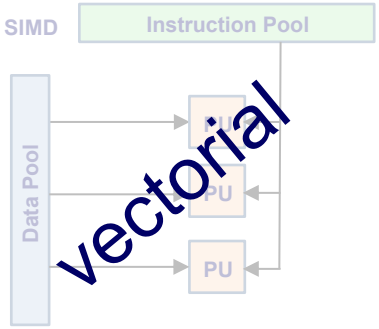
	Single Instruction	Multiple Instruction
Single Data		
Multiple Data		

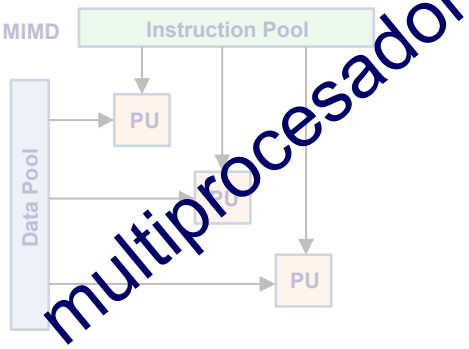
Taxonomía de Flynn

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

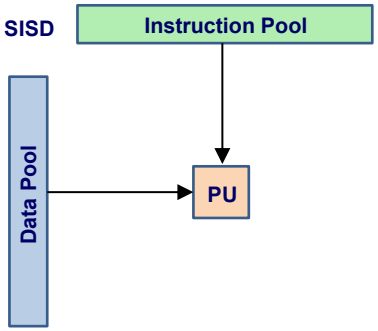
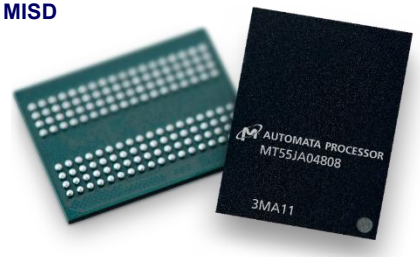
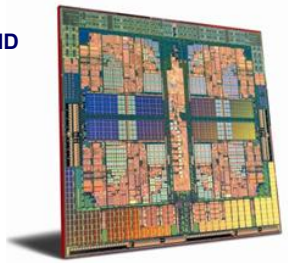




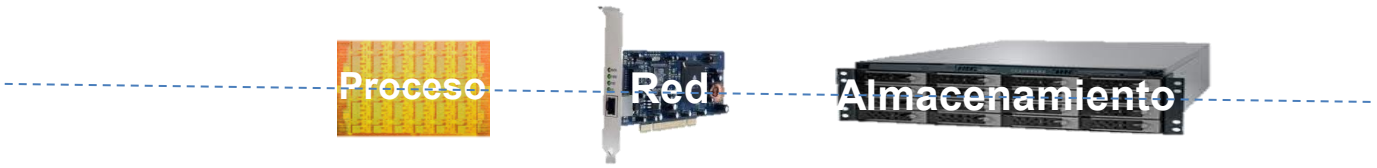




Taxonomía de Flynn

	Single Instruction	Multiple Instruction
Single Data	SISD 	MISD  *automata processor*
Multiple Data	SIMD  vectorial	MIMD  multiprocesador

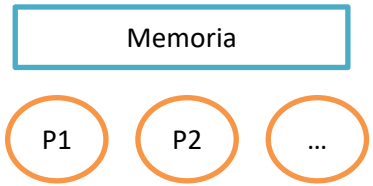
Plataforma hardware



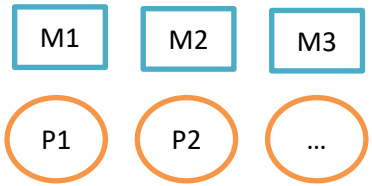
HW

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

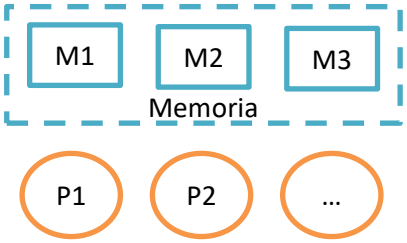
Acceso a memoria



- Memoria compartida (UMA)



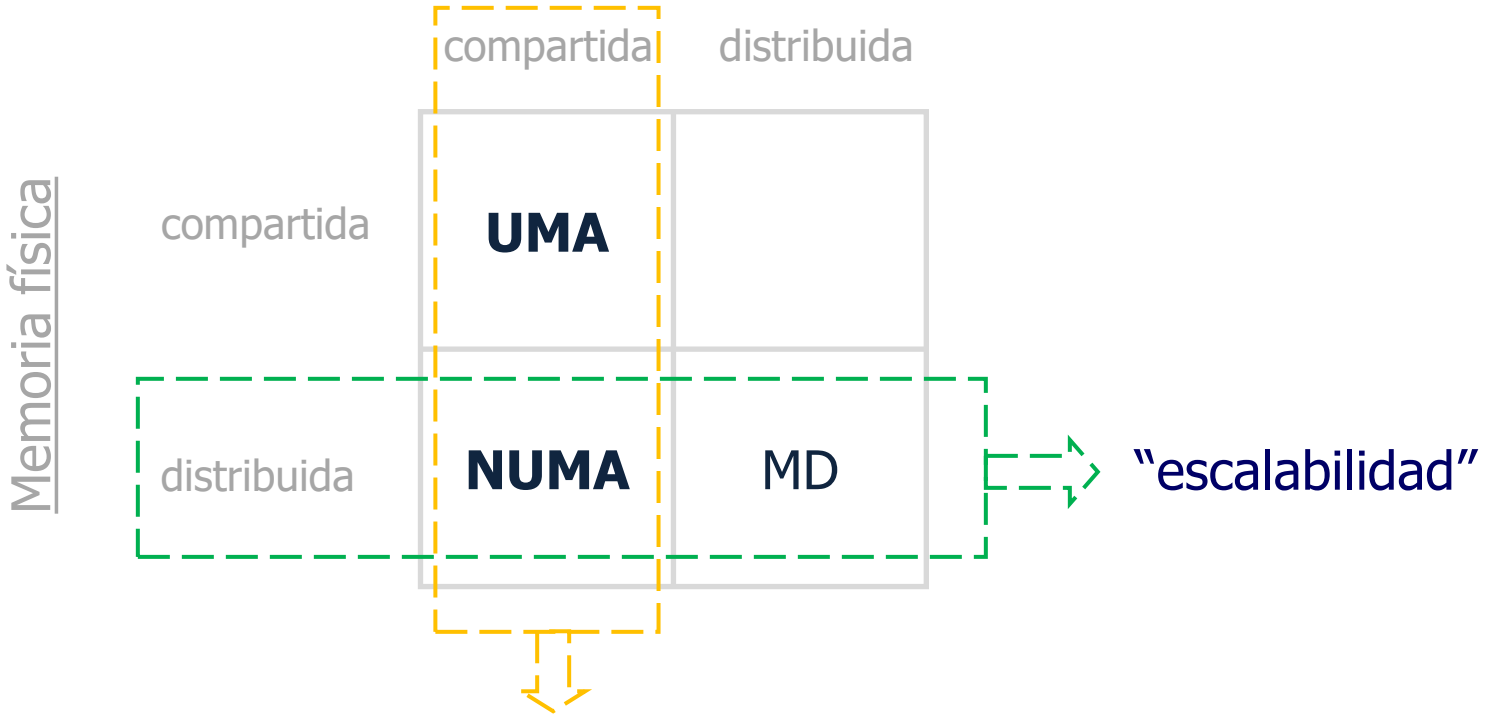
- Memoria distribuida (MD)



- Memoria lógicamente compartida (NUMA)

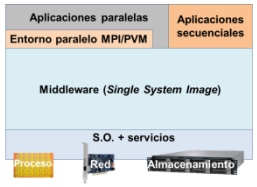
Acceso a memoria

Visión lógica de la memoria
(comunicación/sincronización)



Plataforma software

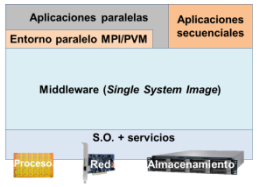
SW



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

Plataforma software

SW



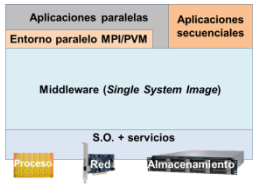
- Vectoriales
  - Uso de instrucciones especiales
- Multiprocesador
  - UMA, NUMA
    - OpenMP, …
  - M. Distribuida
    - MPI, …

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, …

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 5.0.8 (30/5/2025)

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

MPICH 4.3.1 (20/6/2025)

- 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

https://www.pinterest.es/pin/497577458813063755/

```
#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

https://www.pinterest.es/pin/497577458813063755/

```
#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 ;
}
```

Dato a enviar

Número de elementos

Tipo de datos

- MPI_CHAR
- MPY_BYTE
- MPI_INT
- MPI_FLOAT
- ....
- Tipos de datos derivados

Send-receive en MPI

https://www.pinterest.es/pin/497577458813063755/

```
#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 ;
}
```

Dato a enviar

Número de elementos

Tipo de datos

- MPI_CHAR
- MPY_BYTE
- MPI_INT
- MPI_FLOAT
- ….
- Tipos de datos derivados

Proceso destinatario

Etiqueta asociada al mensaje

Comunicador

Send-receive en MPI

https://www.pinterest.es/pin/497577458813063755/

```
#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 ;
}
```

Proceso origen del mensaje

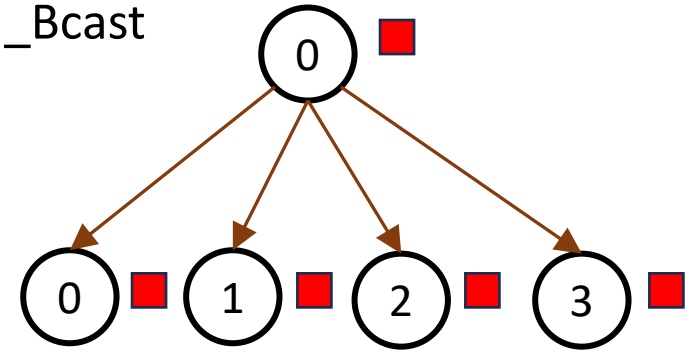
Etiqueta asociada al mensaje

Comunicador

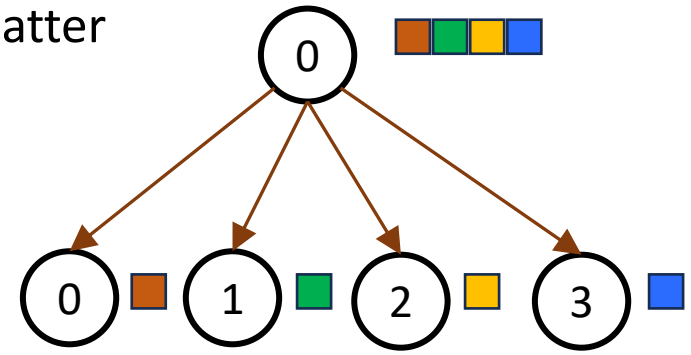
Operaciones colectivas

https://www.pinterest.es/pin/497577458813063755/

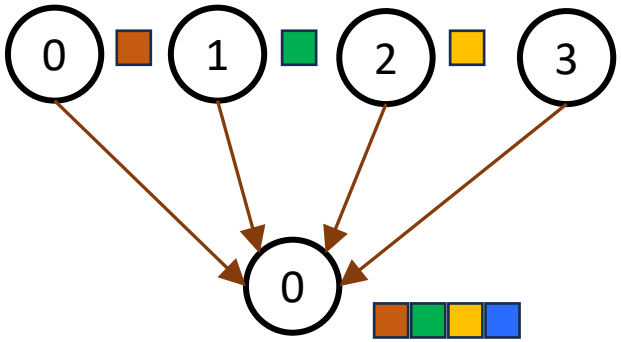
MPI_Bcast



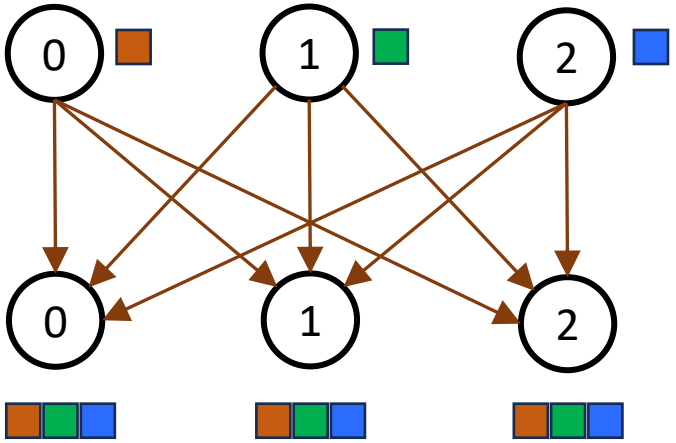
MPI_Scatter



MPI_Gather

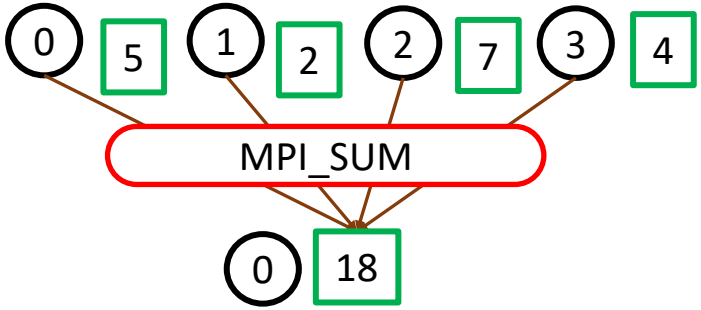


MPI_Allgather

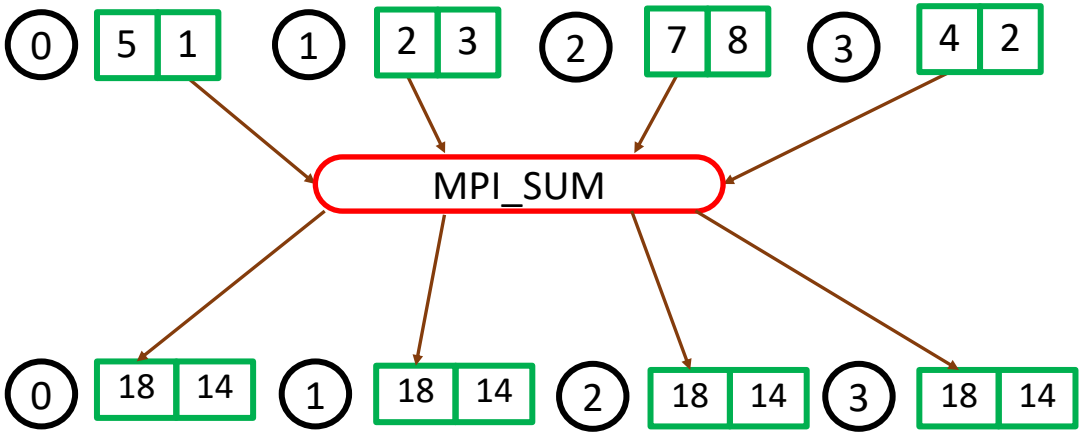


Operaciones colectivas de reducción

MPI_Reduce



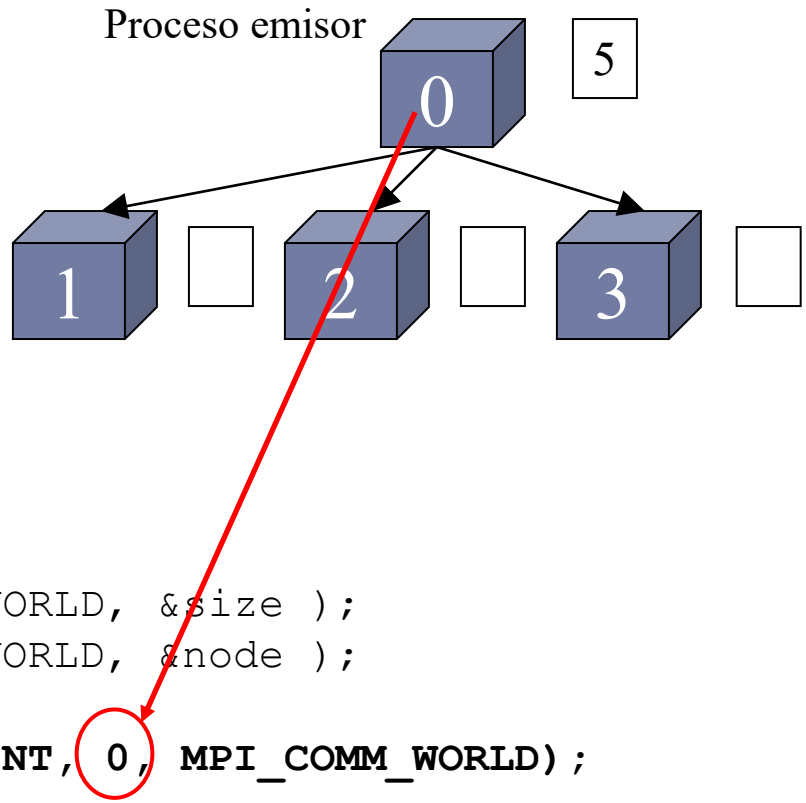
MPI_Allreduce



Operaciones colectivas en MPI

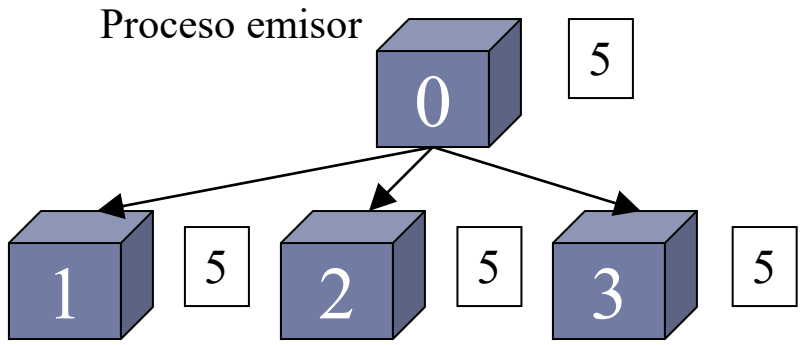
https://www.pinterest.es/pin/497577458813063755/

```
#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();
}
```

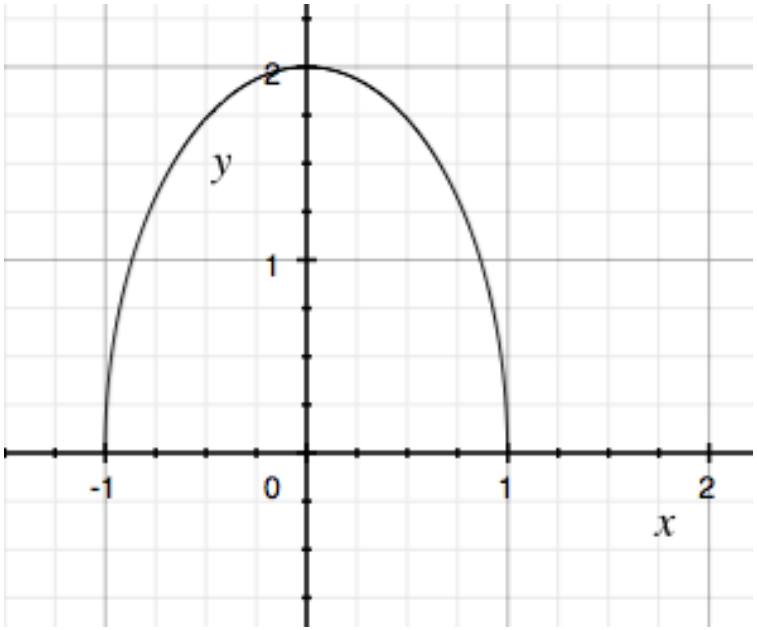


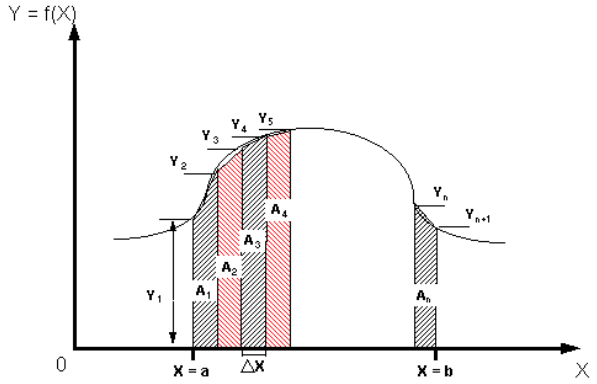
El resto de procesos reciben en num el mensaje enviado por el proceso 0

Ejemplo: Cálculo de π

https://www.pinterest.es/pin/497577458813063755/

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

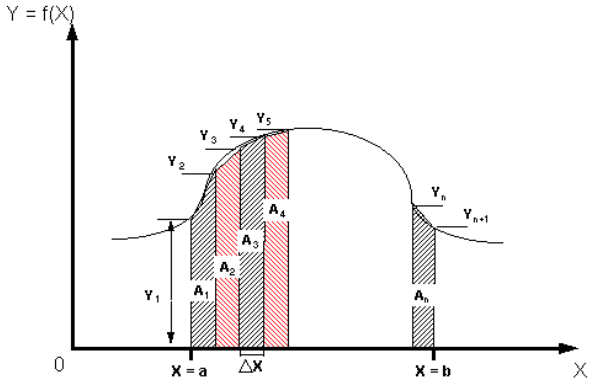




Cálculo secuencial

https://www.pinterest.es/pin/497577458813063755/

```
#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

https://www.pinterest.es/pin/497577458813063755/

```
#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

SW

Aplicaciones paralelas

Entorno paralelo MPI/PVM

Aplicaciones secuenciales

Software de desarrollo
(compiladores y bibliotecas)

Software de gestión de recursos

Software de gestión de sistema
(instalación, administración, monitorización)

S.O. + servicios

(cpu,gpu,…)

(ultrarápida)

(S.F. paralelo y compartido)

HW

Supercomputador

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+MAUI, SLURM, etc.

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

Top 500 Junio 2025

(http://www.top500.org)

Rank	System	Cores	R_{max} (PFLOP/s)	R_{peak} (PFLOP/s)	Power (kW)
1	**El Capitan** - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	**JUPITER Booster** - BullSequana XH3000, GH Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, RedHat Enterprise Linux, EVIDEN EuroHPC/FZJ Germany	4,801,344	793.40	930.00	13,088
5	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
6	**HPC6** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
7	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
8	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
9	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
10	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494

Top 500
(country=es)

- Junio ~~2020~~ ~~2021~~ ~~2022~~ ~~2023~~

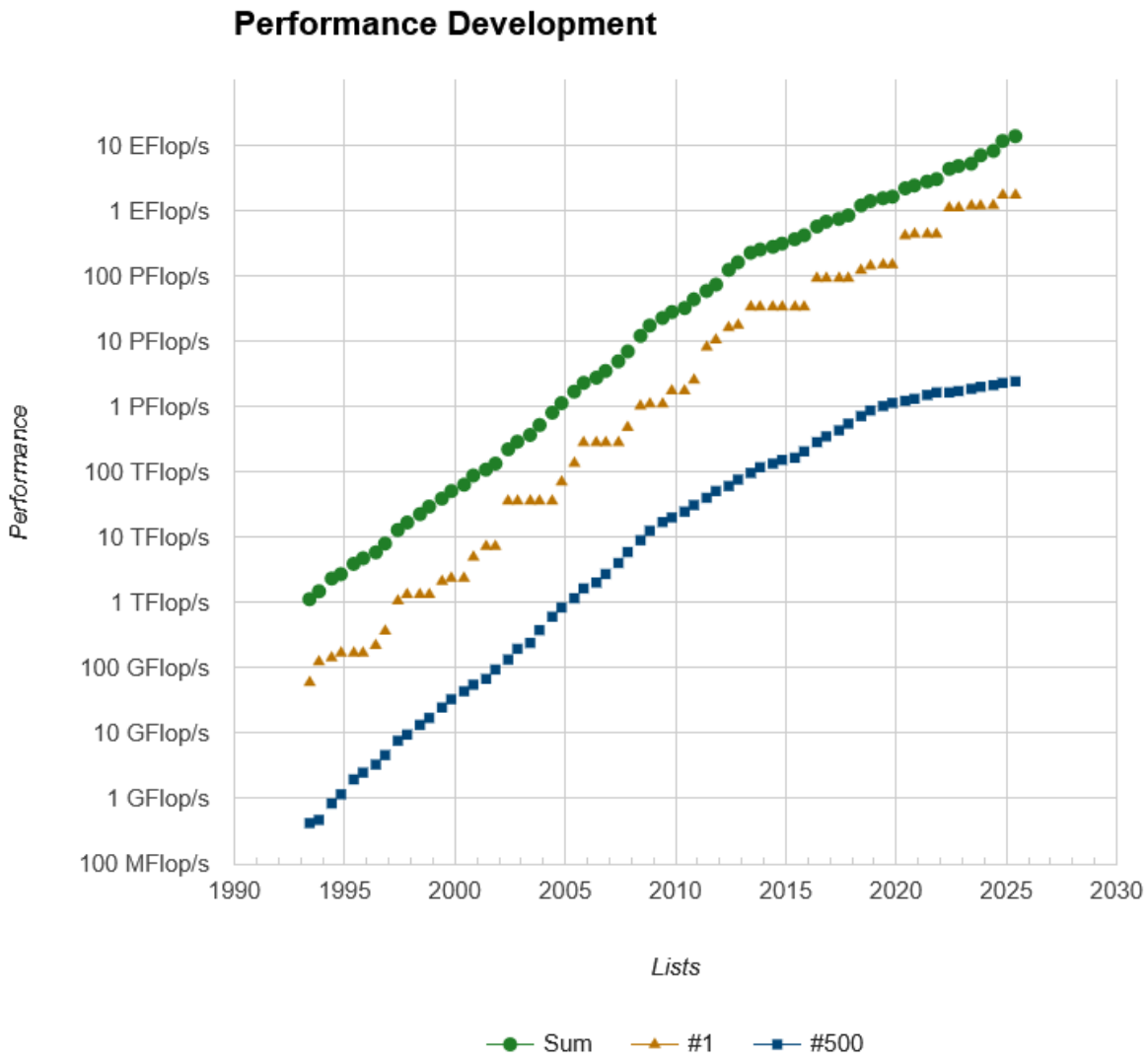
~~37~~ ~~63~~ ~~82~~ ~~98~~	Barcelona Supercomputing Center Spain	**MareNostrum** - Lenovo SD530, Xeon Platinum 8160 24C 2.1GHz, Intel Omni-Path , Lenovo	153,216	6,470.8	10,296.1	1,632

- Junio ~~2024~~ 2025

~~8~~ 14	**MareNostrum 5 ACC** - BullSequana XH3000, Xeon Platinum 8460Y+ 32C 2.3GHz, NVIDIA H100 64GB, Infiniband NDR, EVIDEN, EuroHPC/BSC Spain	663,040	175.30	249.44	4,159
~~22~~ 45	**MareNostrum 5 GPP** - ThinkSystem SD650 v3, Xeon Platinum 03H-LC 56C 1.7GHz, Infiniband NDR200, Lenovo, EuroHPC/BSC Spain	725,760	40.10	46.37	5,753
~~144~~ 203	**MareNostrum** - Lenovo SD530, Xeon Platinum 8160 24C 2.1GHz, Intel Omni-Path, Lenovo Barcelona Supercomputing Center Spain	153,216	6.47	10.30	1,632

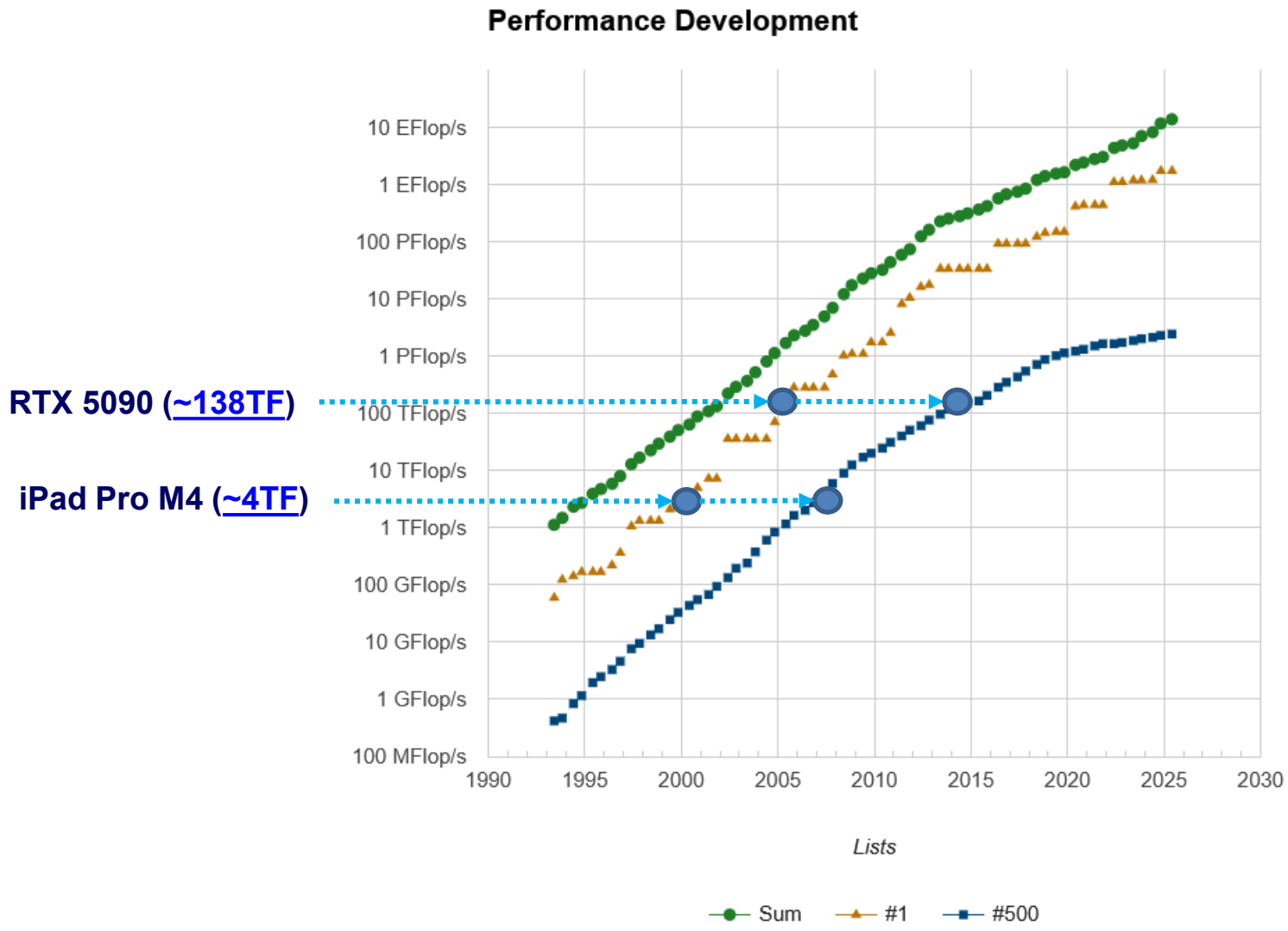
Top 500 Junio 2025

(http://top500.org/statistics/perfdevel/)

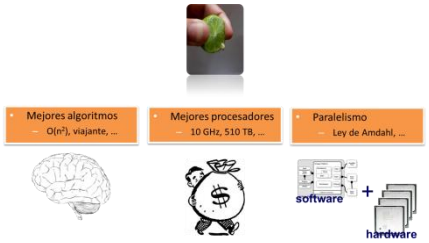


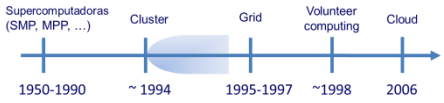
Top 500 Junio 2025

(http://top500.org/statistics/perfdevel/)



Agenda





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

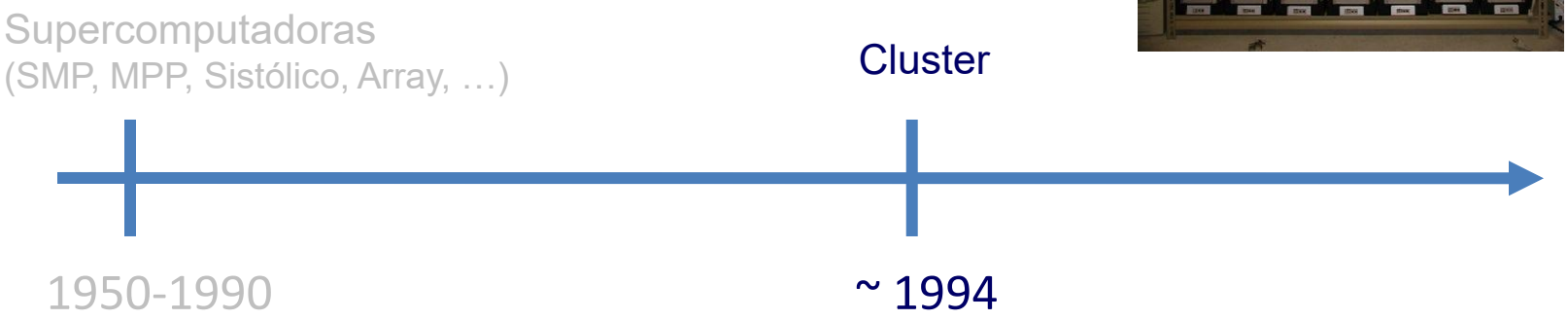
- Problemas con gran cantidad de datos tratados
- Más usado en administración
- Uso de paralelismo y alta frecuencia

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

1950-1990

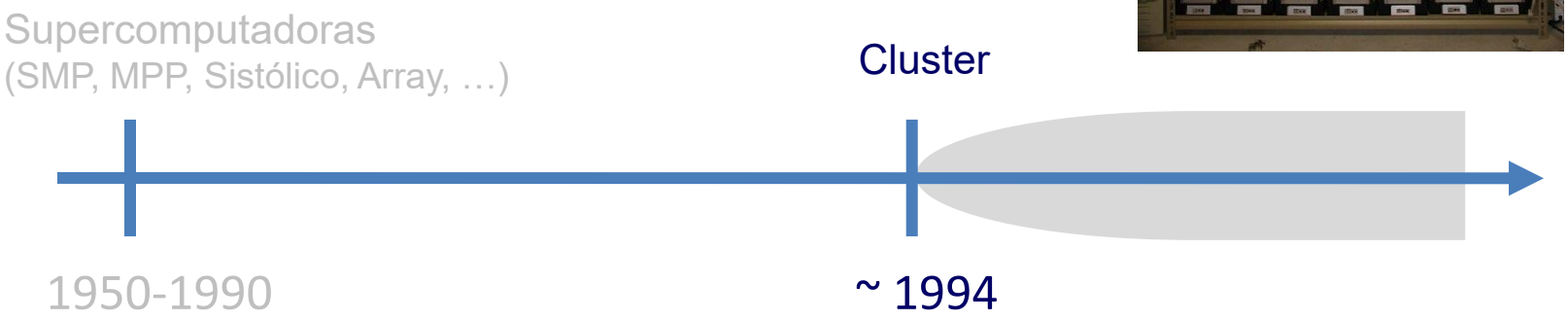
Evolución en las plataformas de computación de altas prestaciones

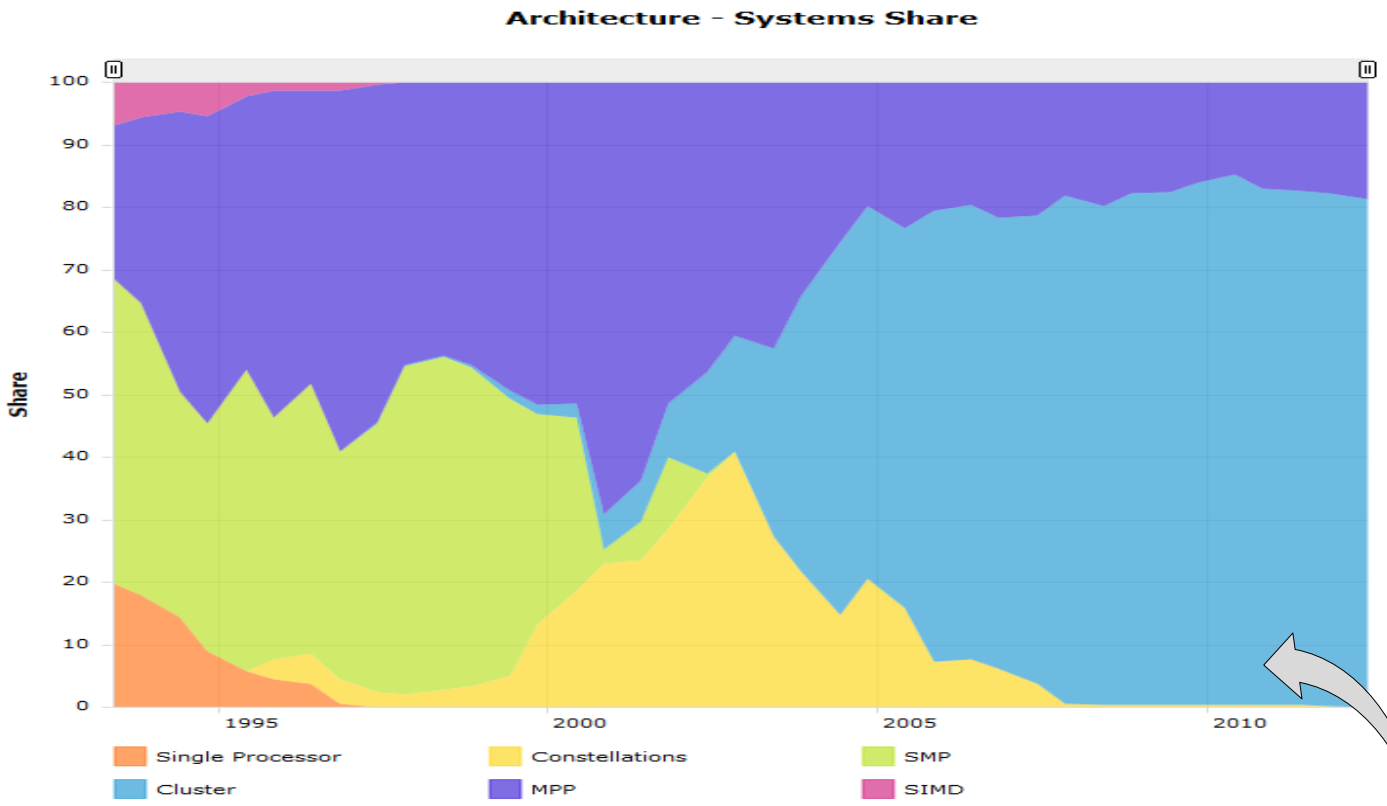
- Construido por Donald Becker y Thomas Sterling en 1994 (NASA)
- Formado por 16 computadores personales con procesador intel DX4 a 200 MHz interconectados por un switch Ethernet.
- Rendimiento teórico era de 3,2 Gflops
- Posibilidad de supercomputadoras ”baratas”



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”





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

Cluster

1950-1990

~ 1994

1997: THE FIRST INTEL® TERAFLOP COMPUTER
consisted of: 9,298 INTEL PROCESSORS
and occupied: 72 SERVER CABINETS

THE INTEL® XEON® PHI™ COPROCESSOR
will provide: 1 TERAFLOP OF PERFORMANCE
and occupy: 1 PCIe SLOT

intel inside
Xeon Phi

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

Cluster

1950-1990

~ 1994

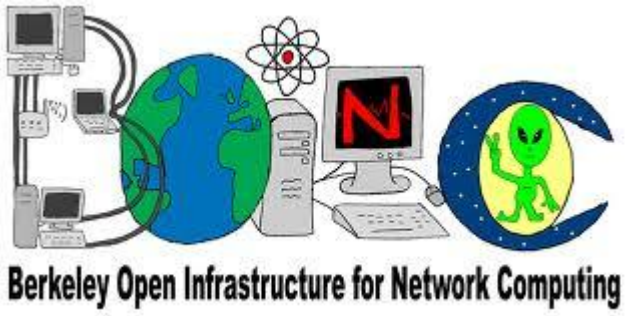
http://es.wikipedia.org/wiki/Intel_MIC

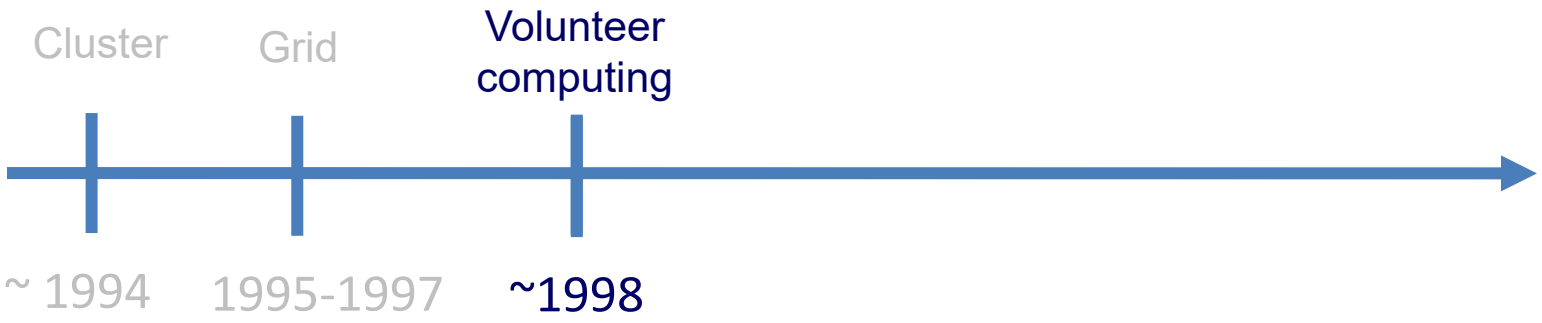
Evolución en las plataformas de computación de altas prestaciones

- 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



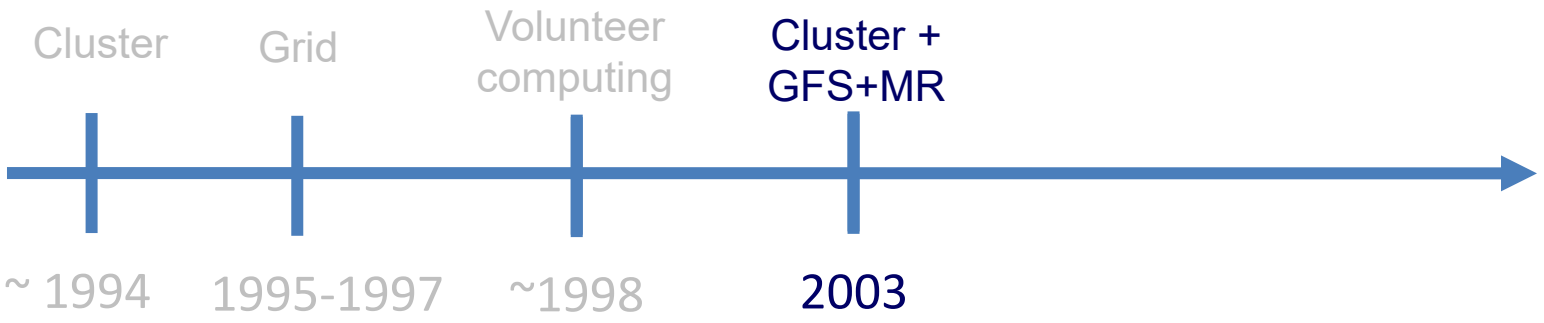


http://boincstats.com/es/stats/-1/project/detail

Evolución en las plataformas de computación de altas prestaciones

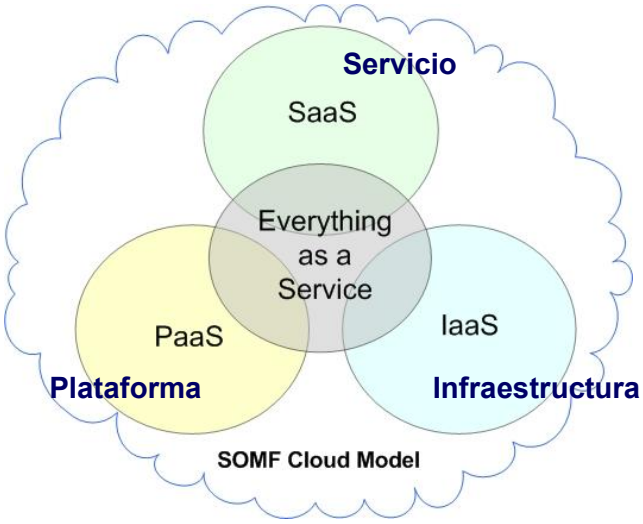
Doug Cutting y Hadoop

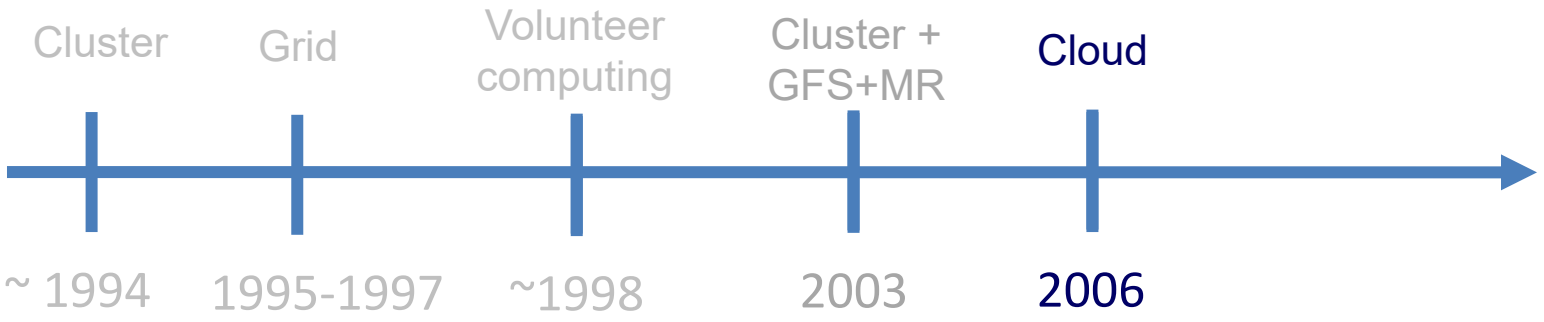
- Google presenta:
  - MapReduce como *framework* para trabajar con grandes conjuntos de datos: la misma función se aplica a diferentes particiones de datos (map) y después estos resultados se combinan (reduce)
  - GFS como forma de almacenar petabytes de datos (ordenadores normales, distribución escalable 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



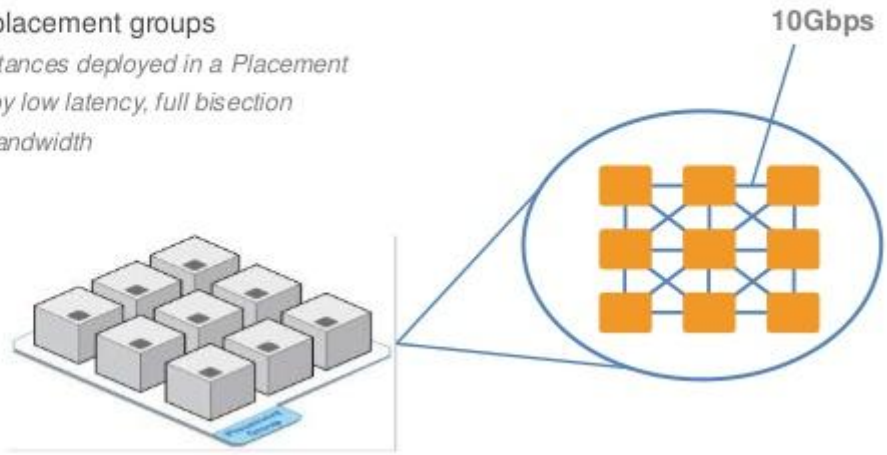


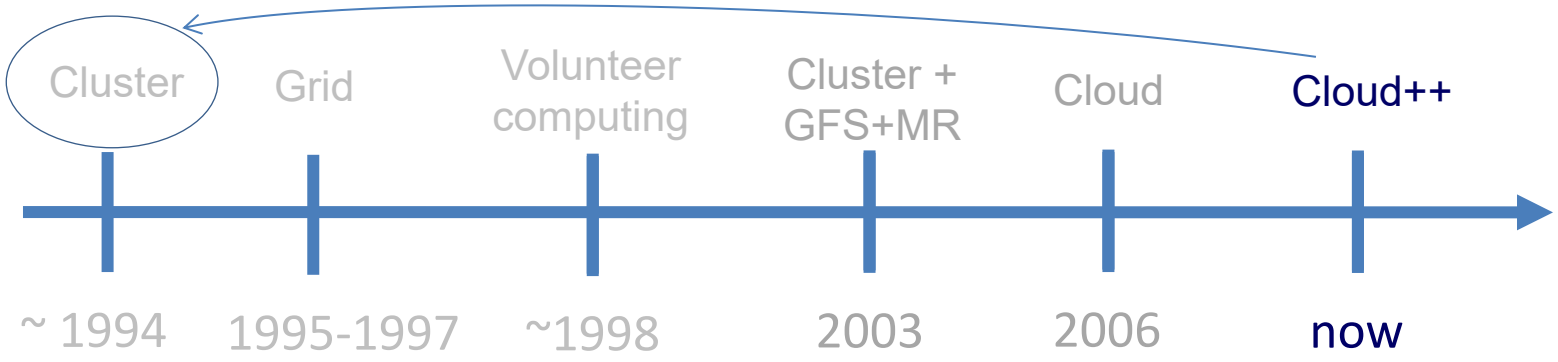
Amazon Cluster Compute Instance

Tightly coupled

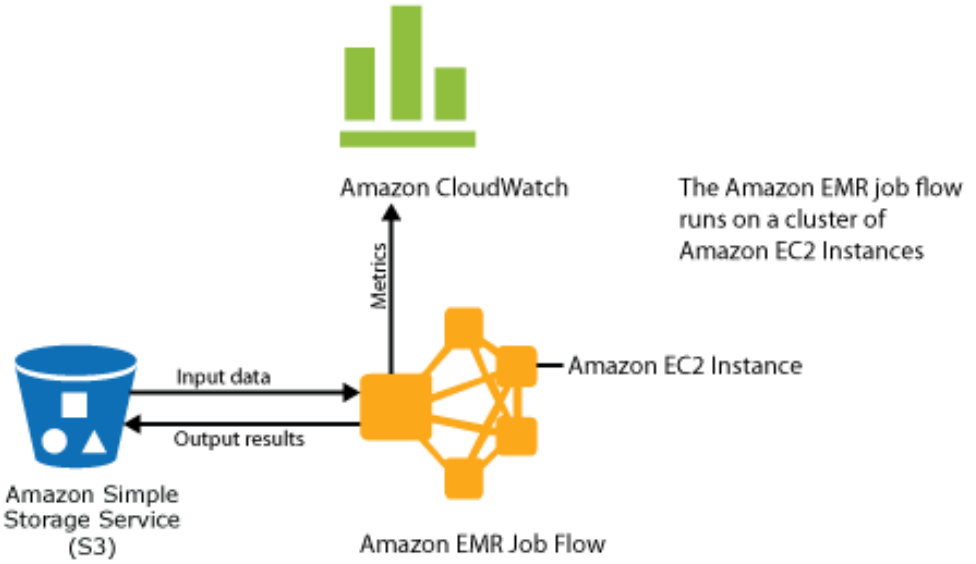
Network placement groups

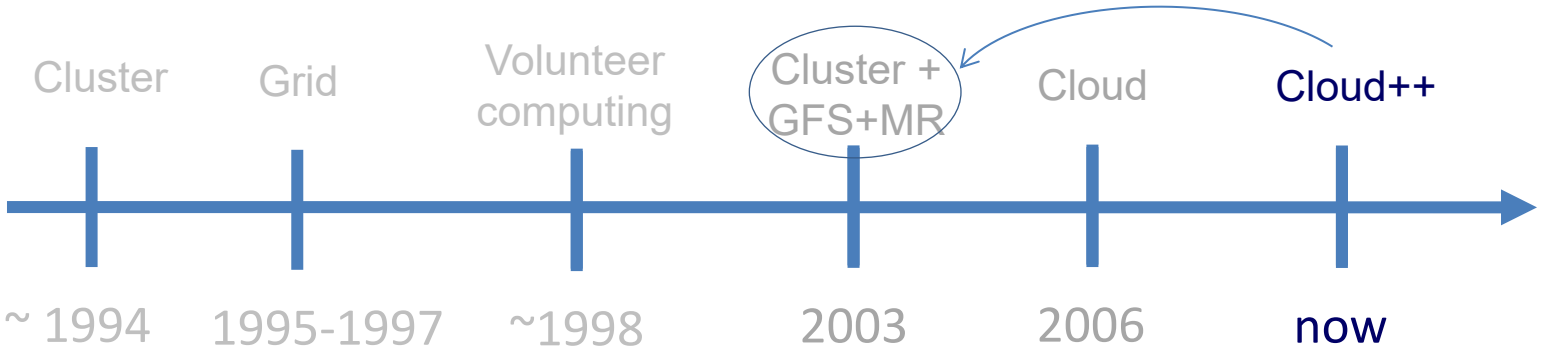
Cluster instances deployed in a Placement Group enjoy low latency, full bisection 10 Gbps bandwidth





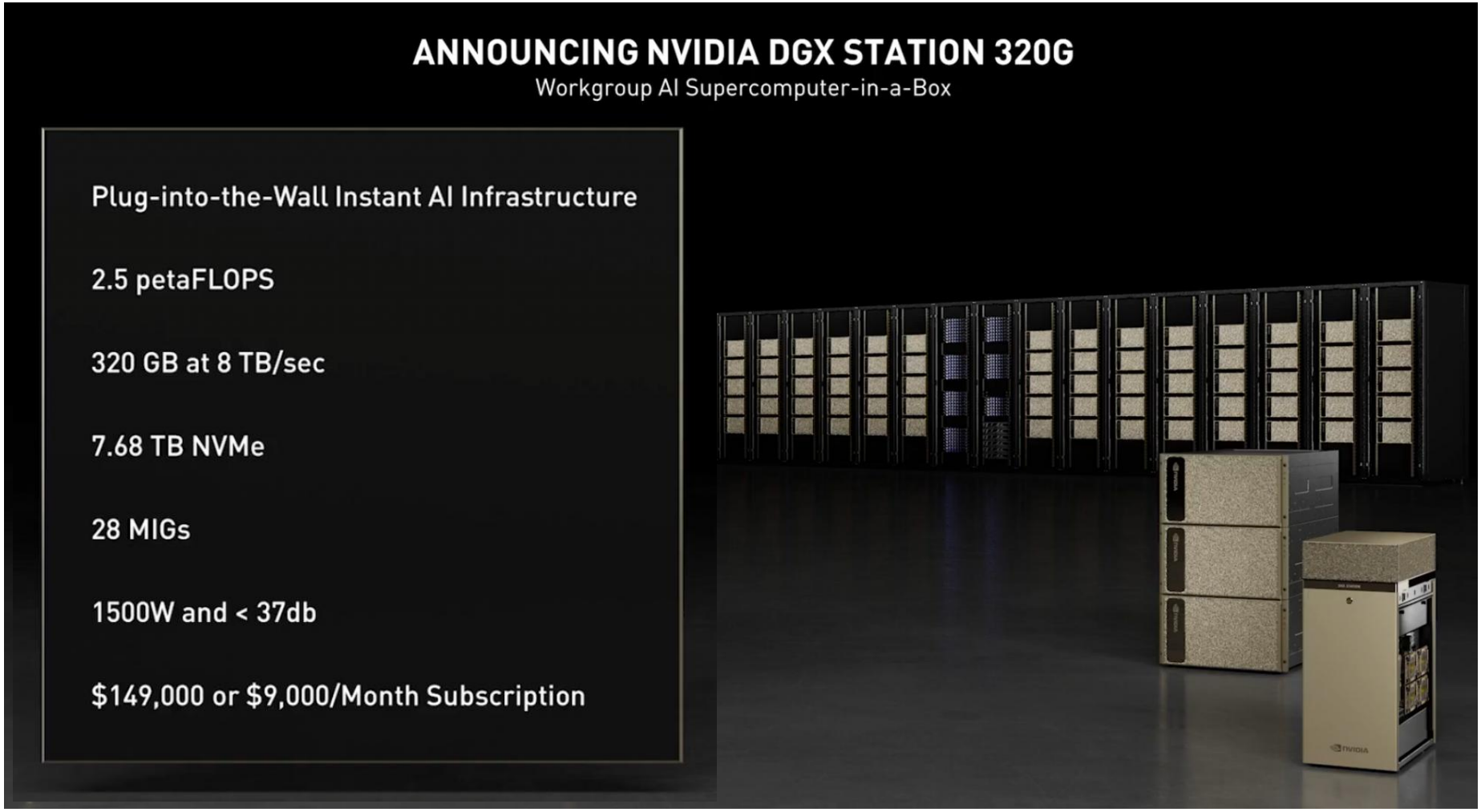
Amazon Elastic MapReduce



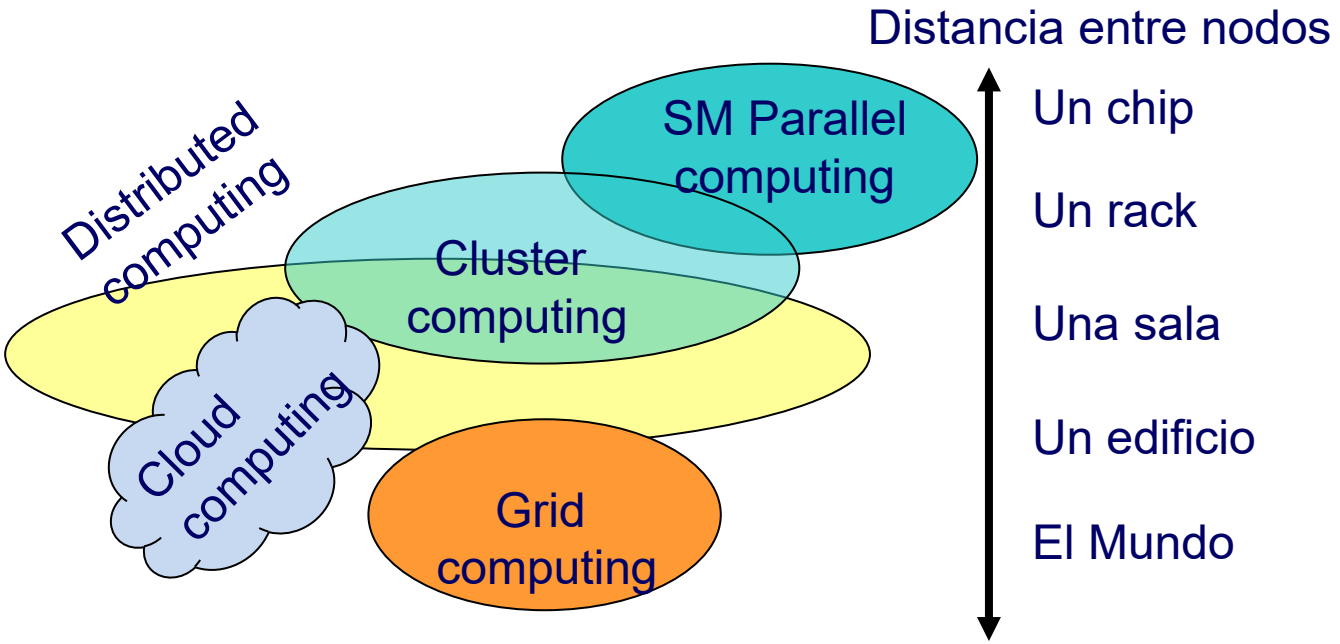


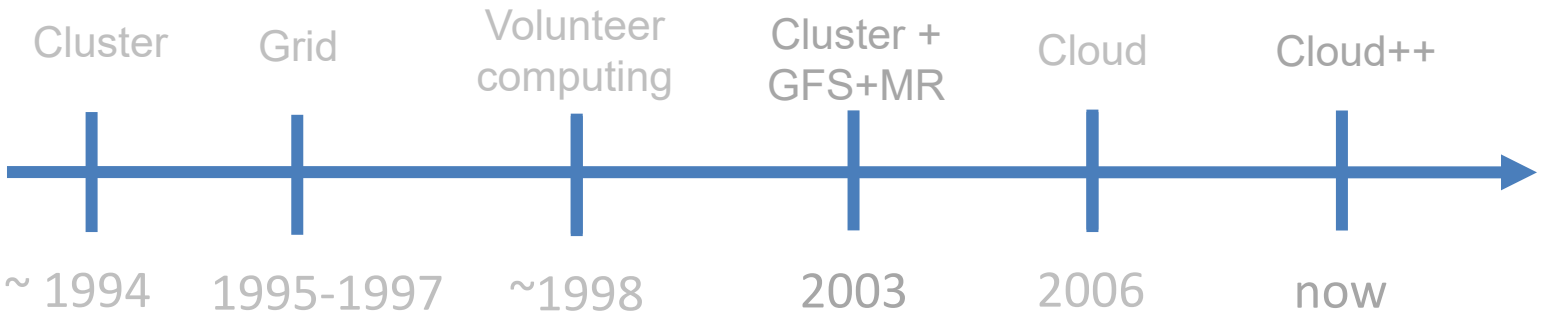
Nvidia DGX-320G

(https://www.nvidia.com/es-es/data-center/dgx-2/)

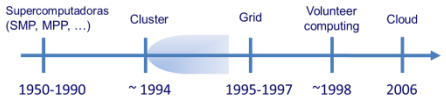


https://wccftech.com/nvidia-dgx-station-320g-quad-ampere-a100-gpus-with-320-gb-memory/


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


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

Agenda



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

Supercomputadoras (SMP, MPP, …)

Cluster

Grid

Volunteer computing

Cloud

Plataforma →

Aplicaciones paralelas

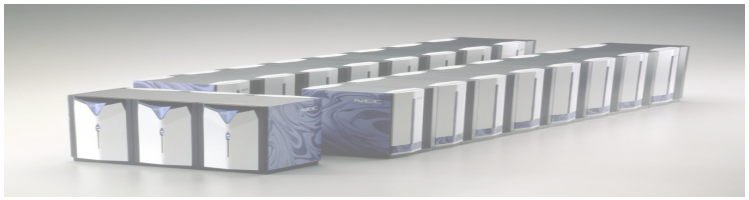
Entorno paralelo

Aplicaciones secuenciales

Middleware (*Single System Image*)

S.O. + servicios

Software →



Computador de altas prestaciones

Hardware →

Principales tendencias

Supercomputadoras (SMP, MPP, …)

Cluster

Grid

Volunteer computing

Cloud

Plataforma →

Aplicaciones paralelas

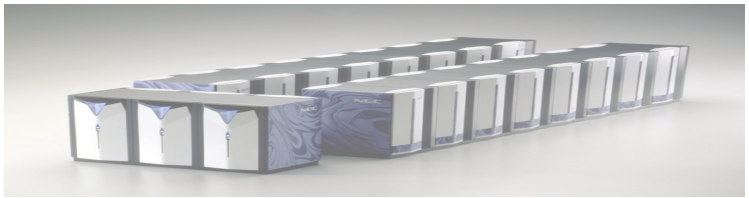
Entorno paralelo

Aplicaciones secuenciales

Middleware (*Single System Image*)

S.O. + servicios

Software →

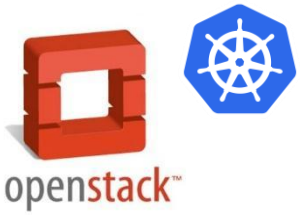


Computador de altas prestaciones

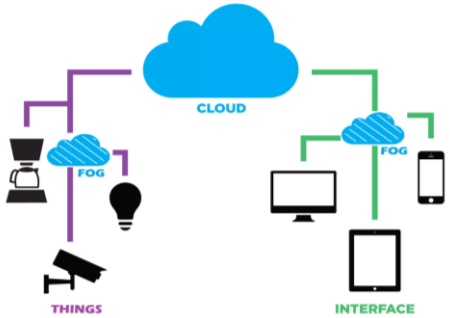
Hardware →

Plataforma:
uso de recursos distribuidos

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



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



https://iot.do/ngd-openfog-fog-computing-2016-10

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

Plataforma:

uso eficiente de recursos

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

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

Norton 360 Now Comes With a Cryptominer

https://krebsonsecurity.com/2022/01/norton-360-now-comes-with-a-cryptominer/

January 6, 2022 110 Comments

Norton 360, one of the most popular antivirus products on the market today, has installed a cryptocurrency mining program on its customers' computers. Norton's parent firm says the cloud-based service that activates the program and allows customers to profit from the scheme — in which the company keeps 15 percent of any currencies mined — is "opt-in," meaning users have to agree to enable it. But many Norton users complain the mining program is difficult to remove, and reactions from longtime customers have ranged from unease and disbelief to, "Dude, where's my crypto?"

Principales tendencias

Supercomputadoras (SMP, MPP, …)

Cluster

Grid

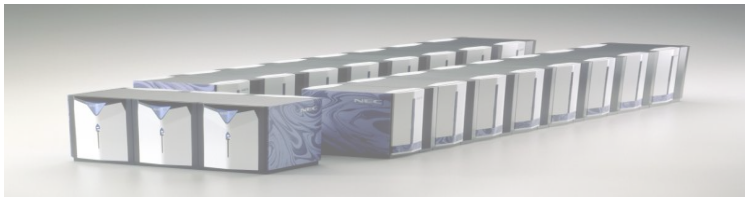
Volunteer computing

Cloud

Plataforma →

Aplicaciones paralelas	Aplicaciones secuenciales
Entorno paralelo	
Middleware (*Single System Image*)	
S.O. + servicios	

Software →

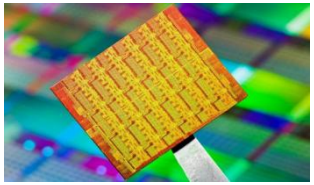


Computador de altas prestaciones

Hardware →

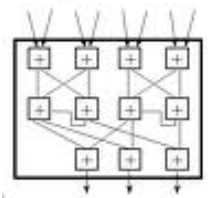
Hardware

A nivel de multicomputador



A nivel de procesador

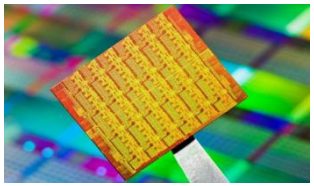
A nivel de instrucción



A nivel de bit

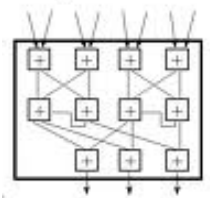
Hardware

A nivel de multicomputador



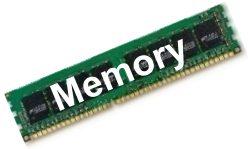
A nivel de procesador

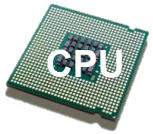
A nivel de instrucción



A nivel de bit

- **Hardware específico**
- Hardware multicore





Hardware:

”memoria” con capacidad de cómputo

- **Memoria “activa”**:
  computo simple en la propia memoria

Planted Motif Search Problem	Automata Processor	UCONN - BECAT Hornet Cluster
Processors	48 (PCIe Board)+CPU	48 CPU (Cluster/OpenMPI)
Power	245W-315W[1]	>2,000W[1]
Cost	TBD	~$20,000[1]
Performance (25,10)	12.26 minutes[2]	20.5 minutes
Performance (26,11)	13.96 minutes[2]	46.9 hours
Performance (36,16)	36.22 minutes[2]	Unsolved

- Planted Motif Search problem is a leading problem in bioinformatics and is NP Hard. Attempts to find common genomic sequences in noisy data.
- Solutions involving high match lengths and substitution counts are often presented to HPC clusters for processing.
- Independent research predicts the Micron Automata Processor significantly outperforms a multi-core HPC cluster in speed, power and estimated cost.

[1] Micron Technology Estimates, Not including Memory of 4GB DRAM /Core
[2] Research conducted by Georgia Tech (Roy/Aluru)

- http://www.hpcwire.com/2013/11/22/micron-exposes-memorys-double-life-automata-processor/

Hardware:

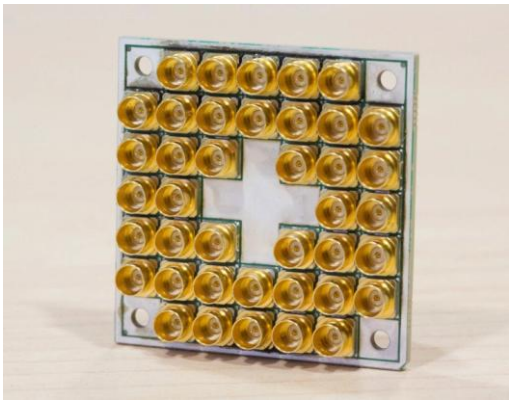
aceleradores específicos por USB

- Conector USB Type A.
- VPU (Vision Processing Unit) Myriad 2.
- 4 GB de memoria LPDDR3.
- Soporte del framework “Caffe”.
- Compatible con FP16 (precisión media).
- Consumo de 1 vatio.
- Precio: **79 dólares** (2017)

- https://www.muycomputer.com/2017/07/20/movidius-neural-compute-stick/
- https://www.movidius.com/MyriadX

Hardware:

qubit-chip

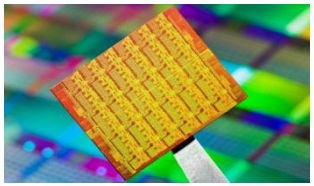


- "…While quantum computers promise greater efficiency and performance to handle certain problems, they won't replace the need for conventional computing or other emerging technologies like neuromorphic computing. We'll need the technical advances that Moore's law delivers in order to invent and scale these emerging technologies…"

- https://newsroom.intel.com/news/intel-delivers-17-qubit-superconducting-chip-advanced-packaging-qutech/

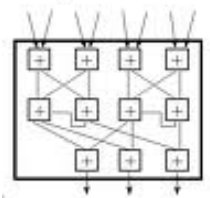
Hardware

A nivel de multicomputador



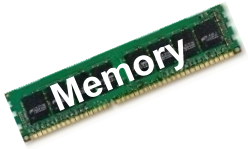
A nivel de procesador

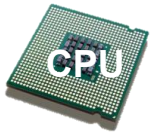
A nivel de instrucción



A nivel de bit

- Hardware específico
- **Hardware multicore**





Hardware:

más procesadores y cores heterogéneos

- **Tarjetas gráficas**: uso de la capacidad de procesamiento de las potentes tarjetas gráficas actuales

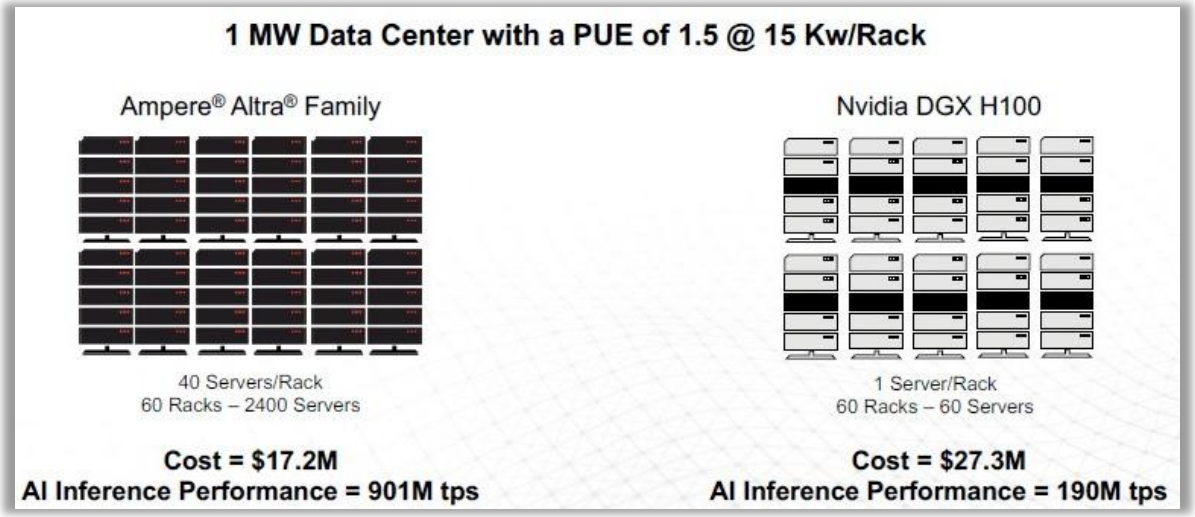
Hardware:

más procesadores y cores heterogéneos

- **Tarjetas gráficas**: uso de la capacidad de procesamiento de las potentes tarjetas gráficas actuales
  - CUDA:
    Entorno de programación para poder usar la potencia de las tarjetas gráficas de NVidia
  - OpenCL:
    lenguaje basado en C99 extendido para operaciones vectoriales y eliminando ciertas funcionalidades

Hardware:
más procesadores y cores heterogéneos

- **Procesadores heterogéneos**: gran cantidad de procesadores con coprocesadores especializados



- https://www.servethehome.com/this-is-ampere-ampereone-a192-32x-a-192-core-arm-server-cpu-arm/
- https://www.nextplatform.com/2024/04/16/ampere-readies-256-core-cpu-beast-awaits-the-ai-inference-wave/

Hardware:
más procesadores y cores heterogéneos

- **Procesadores heterogéneos**:
  gran cantidad de procesadores con coprocesadores especializados
  - <memoria compartida>:
    OpenMP, OpenACC, OpenCL
  - <paso de mensaje>:
    MPI

- https://www.servethehome.com/this-is-ampere-ampereone-a192-32x-a-192-core-arm-server-cpu-arm/
- https://www.nextplatform.com/2024/04/16/ampere-readies-256-core-cpu-beast-awaits-the-ai-inference-wave/

Principales tendencias

Supercomputadoras (SMP, MPP, …)

Cluster

Grid

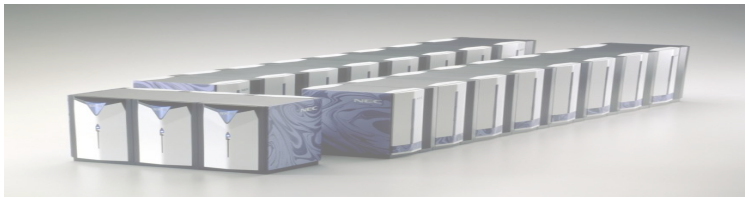
Volunteer computing

Cloud

Plataforma →

Aplicaciones paralelas	Aplicaciones secuenciales
Entorno paralelo	
Middleware (*Single System Image*)	
S.O. + servicios	

Software →



Computador de altas prestaciones

Hardware →

Software

Vectoriales

SSE, AVX, AVX2, …

Multiprocesador

UMA, NUMA

OpenMP,

iTBB, …

M. Distribuida

MPI,…

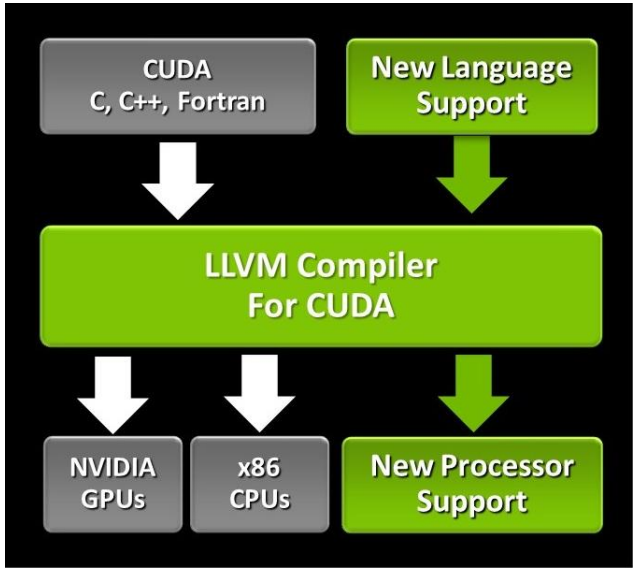
Map-reduce

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

Ejemplo:

CUDA/LLVM adaptado a nuevos entornos

- CUDA Compiler SDK
- Versión de Clang/LLVM con:
  - Generación de código para GPU
  - Compilación con CUDA
- Soporte para:
  - MacOS
  - Windows
  - Linux (algunos)



- http://developer.nvidia.com/cuda/cuda-llvm-compiler

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.**

Ejemplo:

MPI x: adaptación a requisitos actuales

- Programación híbrida
- Tolerancia a fallos
- Acceso remoto a memoria
- Comunicación colectiva y topología
- Soporte de herramientas
- Persistencia
- Compatibilidad hacia atrás

- http://meetings.mpi-forum.org/MPI_3.0_main_page.php

Software

Vectoriales

SSE, AVX, AVX2, …

Multiprocesador

UMA, NUMA

OpenMP,

iTBB, …

M. Distribuida

MPI,…

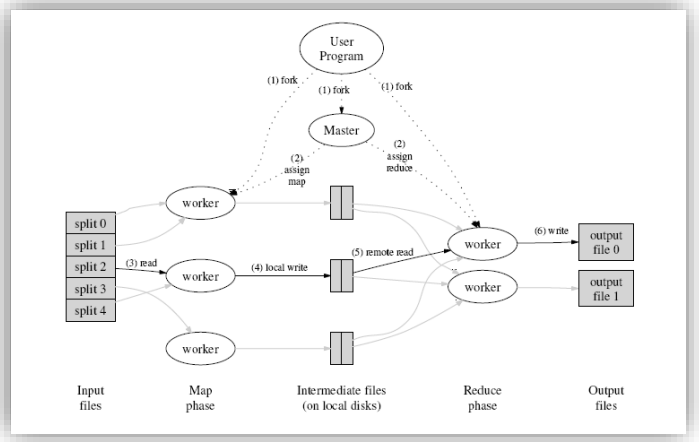
Map-reduce

- 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)

- http://code.google.com/edu/parallel/mapreduce-tutorial.html
- http://code.google.com/edu/submissions/mapreduce-minilecture/listing.html
- http://en.wikipedia.org/wiki/MapReduce

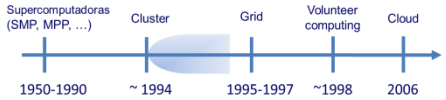
Aplicaciones:

Adaptación a computación de altas prestaciones

- Ejemplos:
  - Primal and dual-based algorithms for sensing range adjustment in WSNs
  - The unified accelerator architecture for RNA secondary structure prediction on FPGA
  - Protein simulation data in the relational model
  - Dynamic learning model update of hybrid-classifiers for intrusion detection

- http://www.springer.com/computer/swe/journal/11227

Agenda



Introducción a la computación de altas prestaciones

- Qué, dónde y cómo
- Hardware y software

Evolución de la computación de altas prestaciones

- Plataformas
- Tendencias

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**
  - http://www.hardwarezone.com.ph/articles/view.php?cid=3&id=2786
- **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

Sistemas Paralelos y Distribuidos

Félix García Carballeira
Alejandro Calderón Mateos

Sistemas de altas prestaciones en entornos distribuidos