



Aspectos avanzados en el uso del clúster UO

Dando poder a los investigadores

Computación paralela

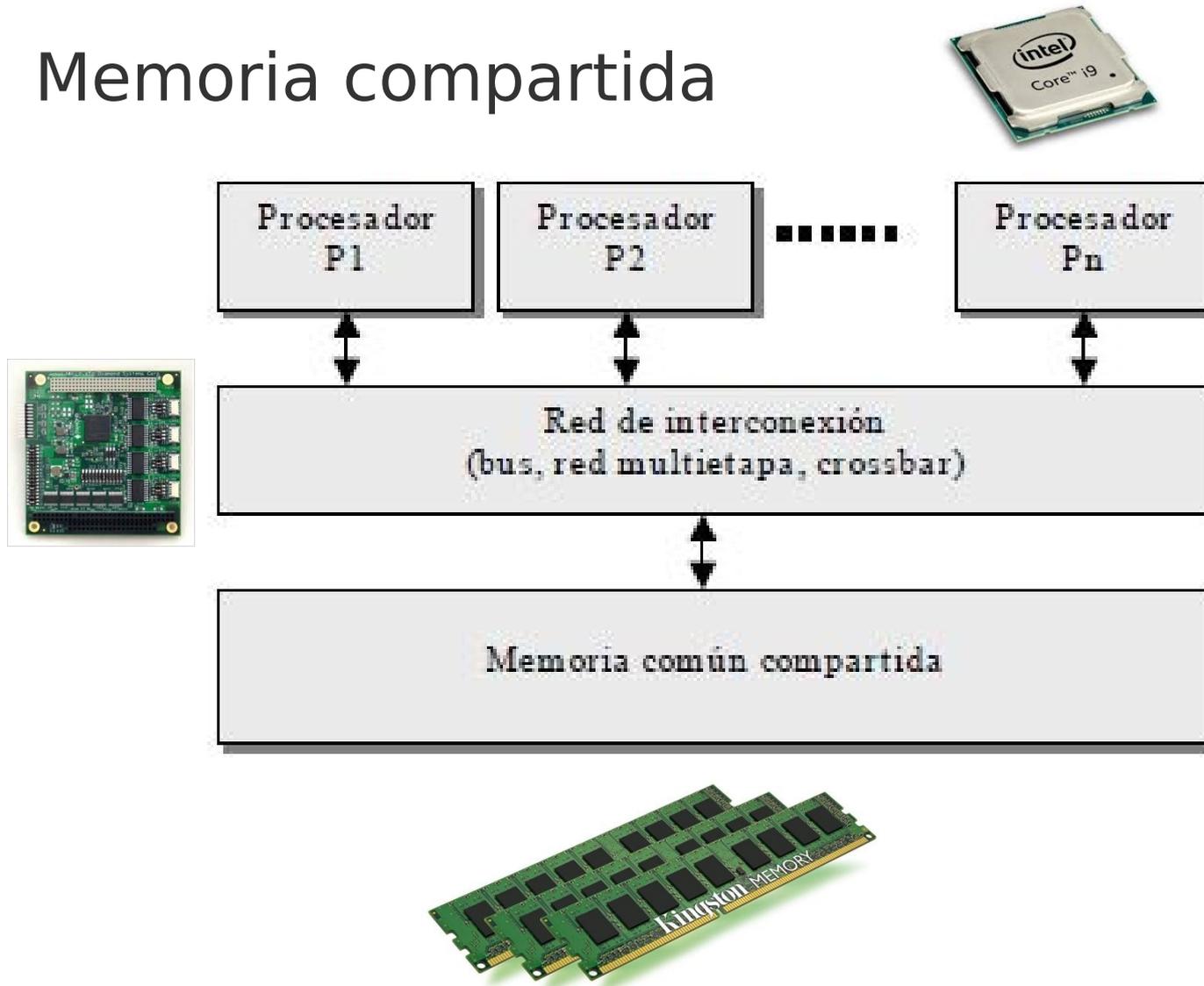
- **Antecedentes:** una computadora desktop con 1 procesador → **CONCURRENCIA**
- Una computadora desktop con más de un procesador (2, 4, 8 o 16 procesadores).
- Una computadora masivamente paralela (cientos o miles de procesadores).
- Un clúster de computadoras escalable (cientos o miles de procesadores conectados mediante una red).



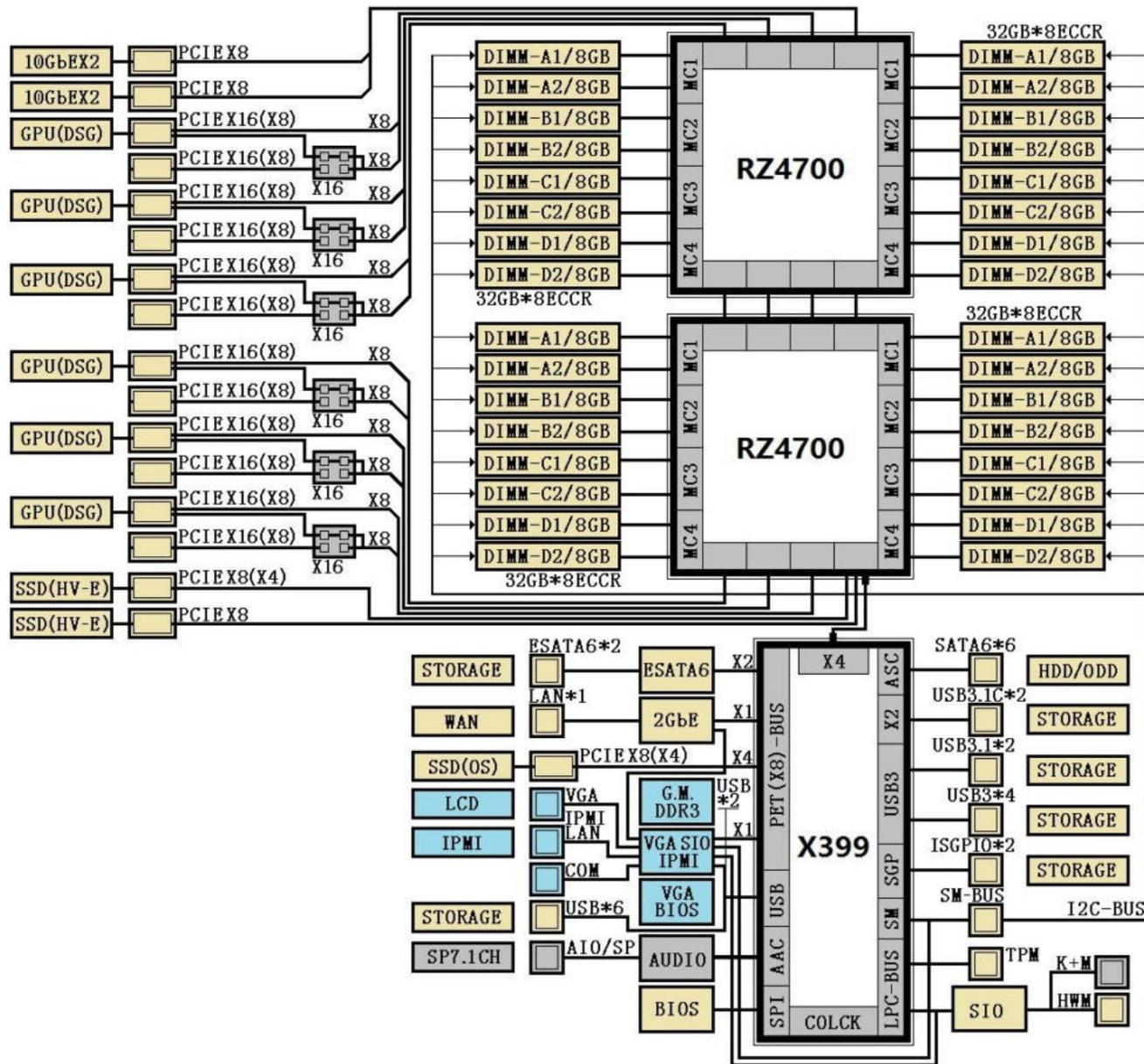
Esquemas de memoria para el trabajo en paralelo

Desktop con varios cores

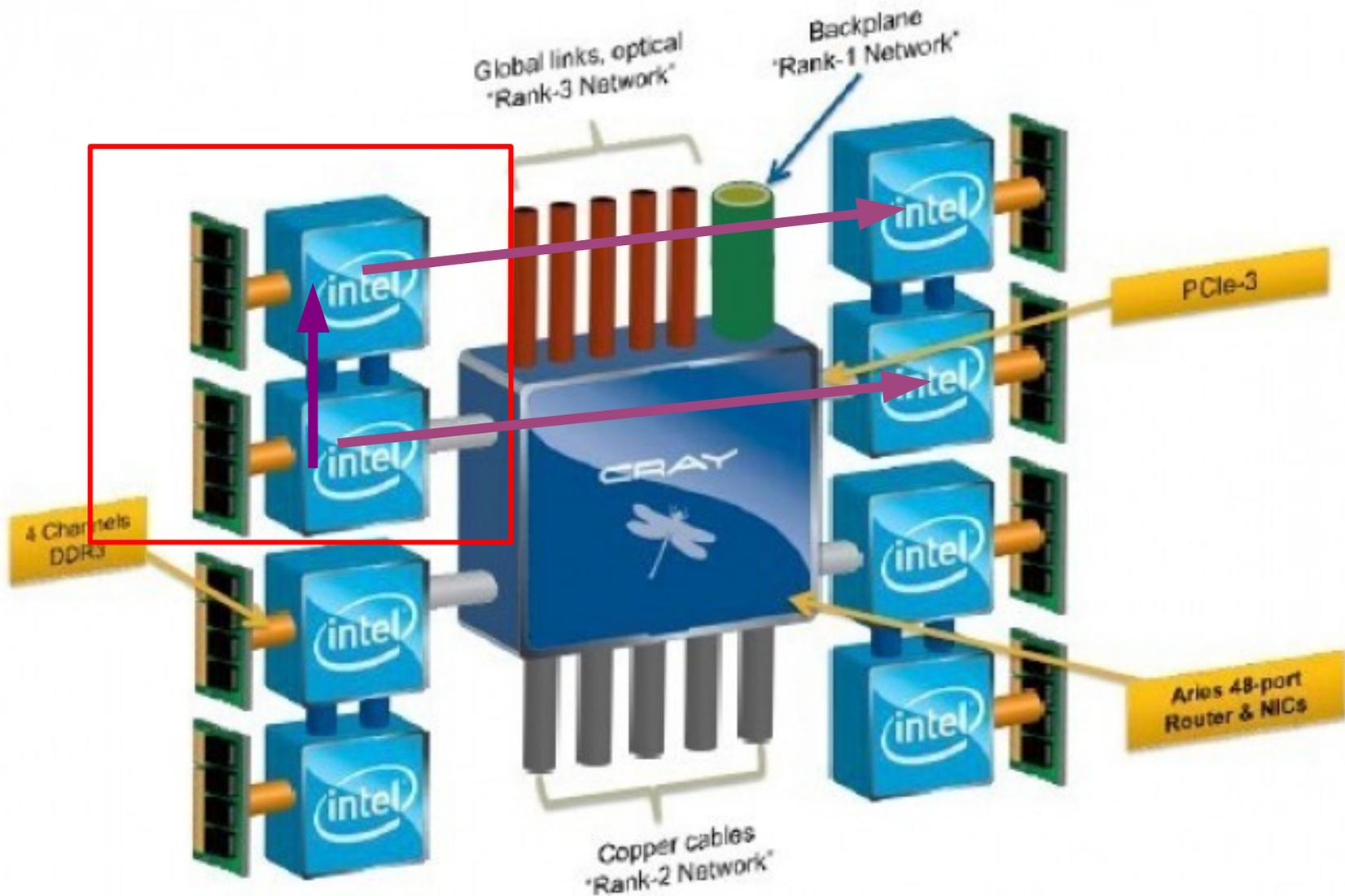
- Memoria compartida



AMD-X399-Chipset-1000...



Multicomputadora

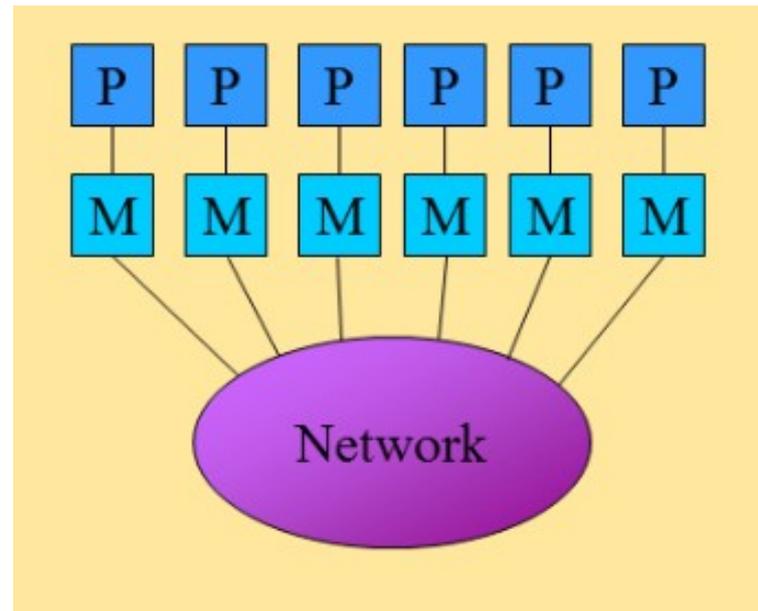


Clústers de computadoras

- Beowulf
- Especializado

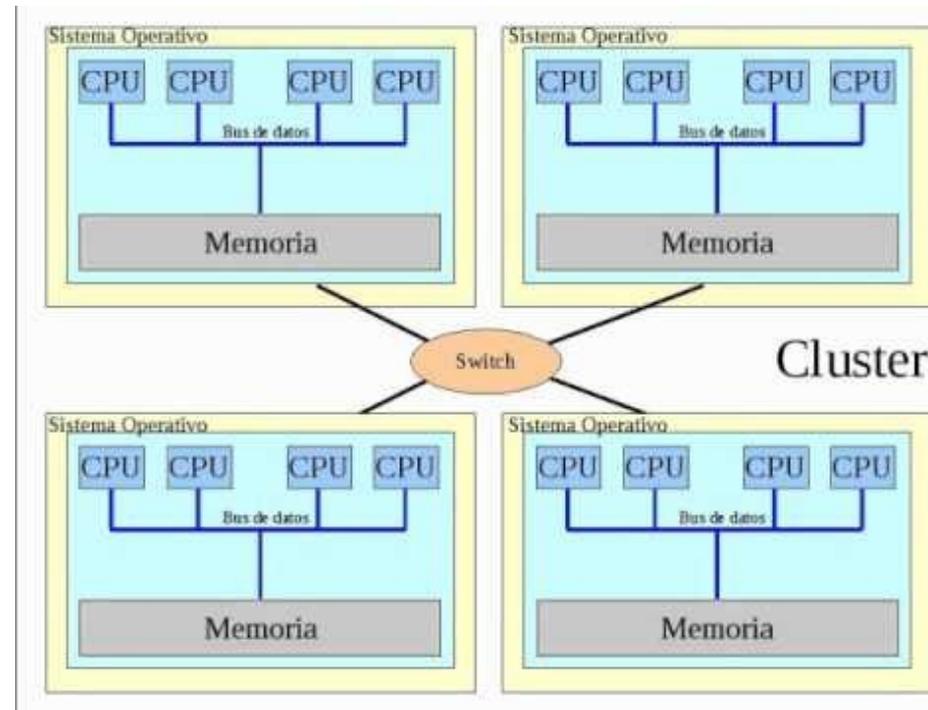


Memoria distribuida



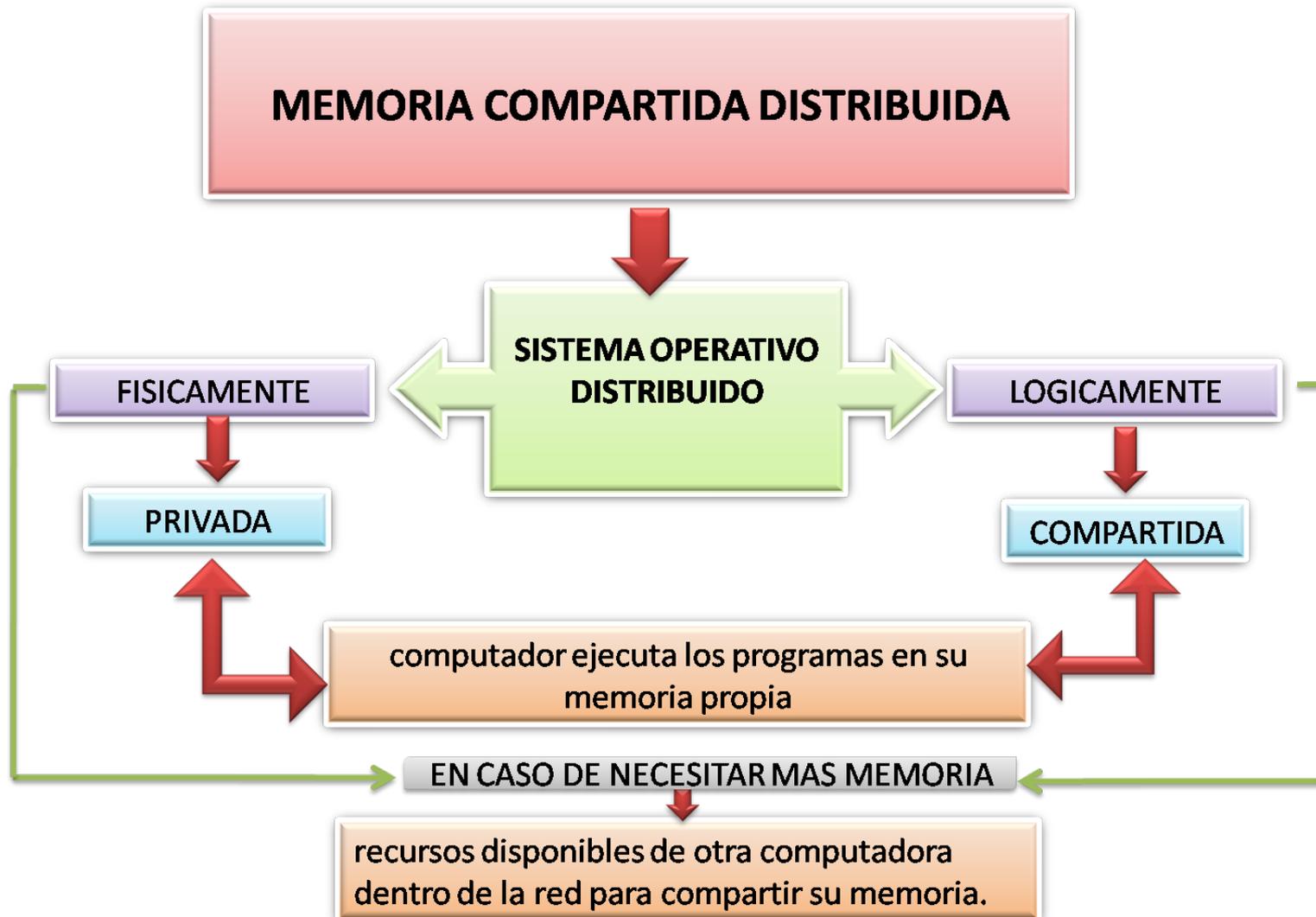
- Cada unidad del clúster tiene su propia memoria, controlada por su procesador.
- La comunicación entre procesadores se realiza mediante mensajes a través de la red.

Memoria compartida y distribuida



- A nivel de nodo está compartida la memoria entre los procesadores locales.
- A nivel de clúster está distribuida y se accede mediante mensajes.

Compartida/distribuida





Clúster de la Universidad de Oriente (HPC-UO)

<http://portal.uo.hpc.cu>
<ssh://login.uo.hpc.cu>

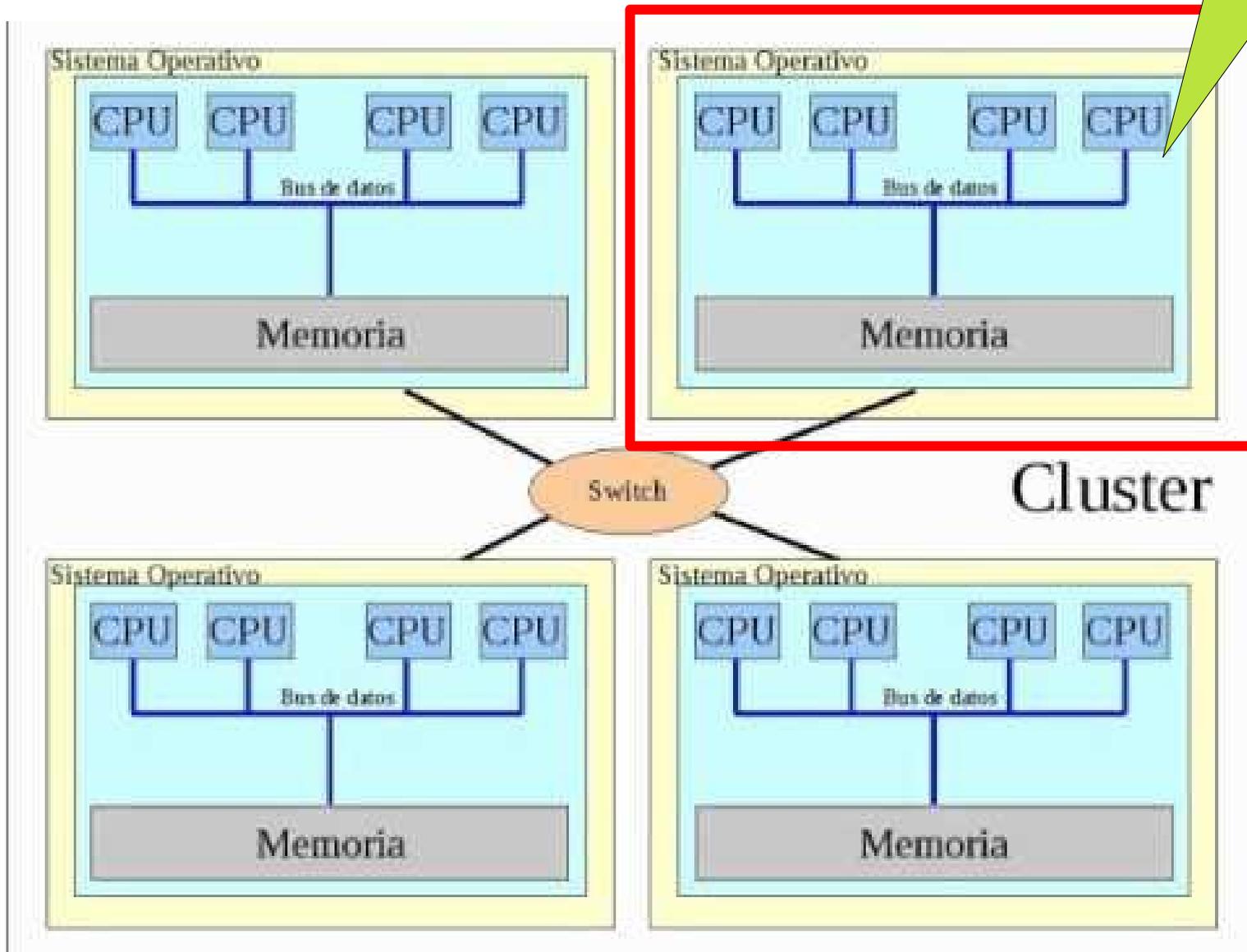
HPC-UO



Sistema de supercómputo de la Universidad de Oriente

Dando poder a los investigadores

Memoria compartida y distribuida



Capacidad de un nodo

- 32 cores x 2.4 GHz
- 64 GB RAM
- Red Ethernet (1 Gb)
- Red Infiniband (40 Gb)

HPC

Para el servicio de HPC se utilizan 13 nodos para el procesamiento CPU con una arquitectura de Memoria Distribuida y 4 procesamiento CPU con una arquitectura de Memoria Compartida.

Equipo	Modelo	Cantidad	Procesador	Cores	RAM	Conexion
Memoria Compartida	SUN-FIRE 4470 X	4	4 x Intel Xeon X7550 (Gainestown or Nehalem-EP)	64	64 GB	2 Ethernet GB
	DELL R720	1	1 x Intel Xeon E5-2609 (Sandy Bridge-EP)	4	16 GB	2 Ethernet GB
Memoria Distribuida	DELL Power Edge C6145	13	4 x AMD Opteron 6136	32	64 GB	1 Ethernet GB 40 InfiniBand GB

El rendimiento teórico del sistema es de 2,1 TFLOPs de operaciones en el cluster de Memoria Distribuida y 3,99 TFLOPs de operaciones en el cluster de Memoria Compartida

Big Data

Nodos	modelo	Cantidad	Procesador	Cores	RAM	Ethernet
DELL Power Edge C6145		1	4 x AMD Opteron 6136	32	64 GB	1 Ethernet GB
		1	2 x AMD Opteron 6136	16	64 GB	1 Ethernet GB

El rendimiento teórico del sistema es de 0,45 TFLOPs de operaciones en CPU

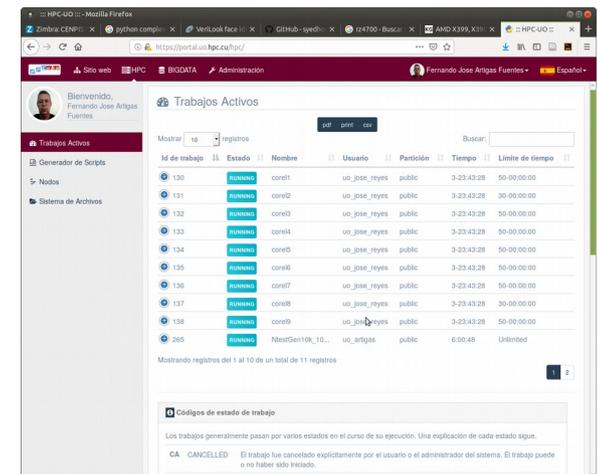


Capacidad total

- 13 nodos
- 416 cores
- 832 GB de RAM
- Almacenamiento disponible para usuarios: 3T

Acceso de los usuarios

- A través de una conexión ssh + scp (o winscp) para mover información.
- A través de la interfaz web
 - Todo en un único sitio.
- + Scripts, o guiones, que describen los trabajos.



Trabajos Activos

Id de trabajo	Estado	Nombre	Usuario	Partición	Tiempo	Limite de tiempo
130	Finalizado	core1	uo_jose_reyes	public	3-23-43:28	00:00:00:00
131	Finalizado	core2	uo_jose_reyes	public	3-23-43:28	00:00:00:00
132	Finalizado	core3	uo_jose_reyes	public	3-23-43:28	00:00:00:00
133	Finalizado	core4	uo_jose_reyes	public	3-23-43:28	00:00:00:00
134	Finalizado	core5	uo_jose_reyes	public	3-23-43:28	00:00:00:00
135	Finalizado	core6	uo_jose_reyes	public	3-23-43:28	00:00:00:00
136	Finalizado	core7	uo_jose_reyes	public	3-23-43:28	00:00:00:00
137	Finalizado	core8	uo_jose_reyes	public	3-23-43:28	00:00:00:00
138	Finalizado	core9	uo_jose_reyes	public	3-23-43:28	00:00:00:00
265	Finalizado	NtstGen10k_10...	uo_artgase	public	6:00:48	Unlimited

Mostrando registros del 1 al 10 de un total de 11 registros.

Códigos de estado de trabajo

Los trabajos generalmente pasan por varios estados en el curso de su ejecución. Una explicación de cada estado sigue.

CA CANCELLED El trabajo fue cancelado explícitamente por el usuario o el administrador del sistema. El trabajo puede o no haber sido iniciado.

Parámetros del script

- Cantidad de nodos
- Cantidad de memoria por nodo
- Cantidad de cores por nodo
- Tareas por nodo
- Programa a ejecutar, datos, módulos, etc.

Manejo de recursos

Lista de recursos	- Cantidad de nodos -	- Tareas totales (en todos los nodos) -
Lista de recursos	- Tareas por nodos -	- Núcleos de CPU por tarea -
Memoria	- RAM por nodo -	- RAM por núcleo de CPU -
Tiempo de ejecución	- Límite de tiempo (mm o [[dd-]hh:]mm:ss o dd-hh[:mm[:ss]]) -	



Relación entre el esquema de memoria usado por los programas de usuario y la reserva de recursos en el clúster

Procedencia de programas

- Programas de terceros
 - Programas secuenciales, en su mayoría.
 - Programas paralelos para **memoria compartida**, especializados.
 - ~~Programas paralelos para **GPUs**.~~
 - Programas paralelos para **memoria distribuida**, mediante MPI.

AVERIGUAR!!!!!!

Procedencia de programas

- Programas **proprios**, desarrollados por el usuario.
- Programas **encargados**, desarrollados por programadores del entorno.
- Fácil de averiguar!!!!!!!

Mirar la documentación

- ¿Memoria compartida o distribuida?
- **OpenMP** → Memoria **compartida** (**1** nodo)
- **OpenMPI** → Memoria **distribuida** (**1 o más** nodos)

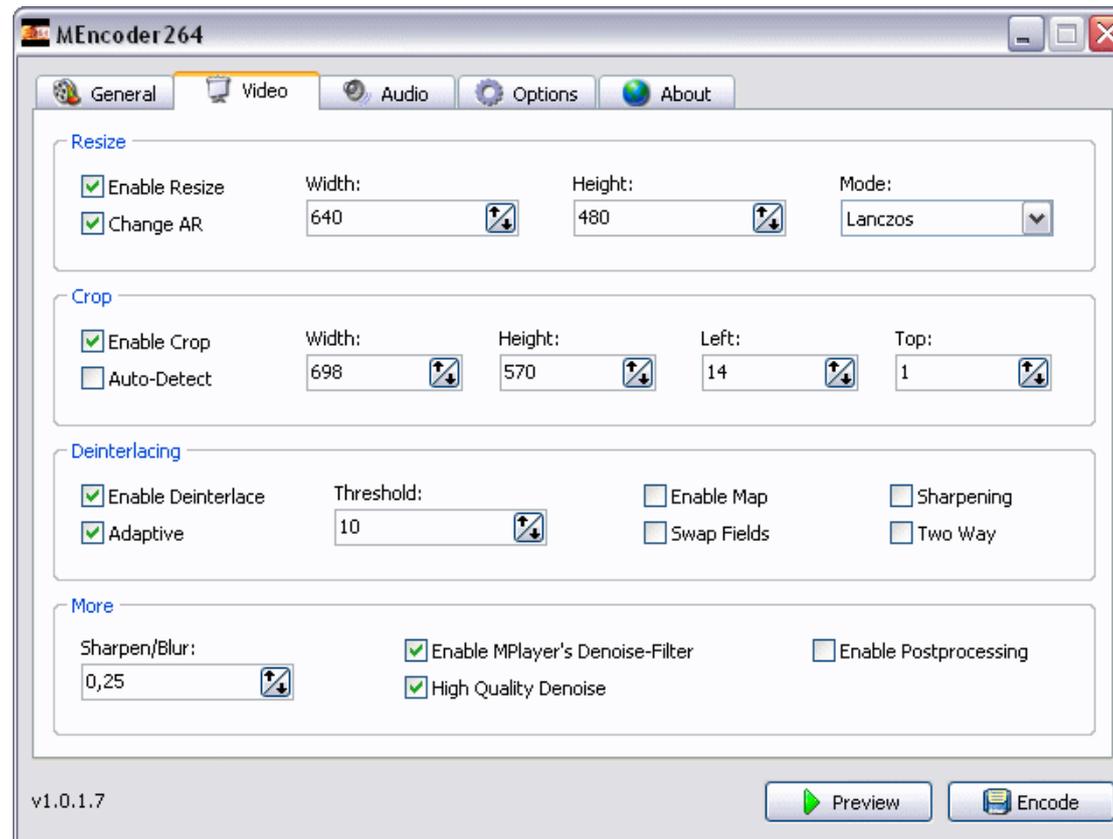
Mirar la documentación

- Requerimientos de memoria
 - 1 GB
 - 2 GB
 - 4 GB
- ¿No sabe cuál es? Obtener un **estimado...**

Caso de estudio I

Un programa secuencial

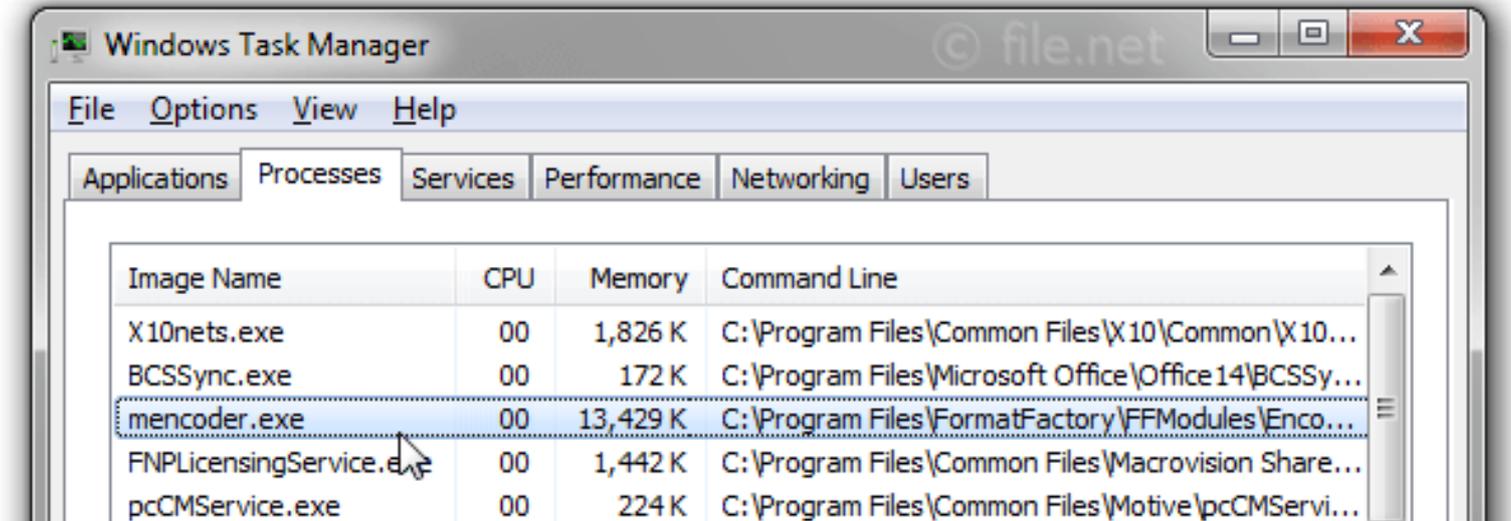
Mencoder



MENCODER

- Programa para convertir de formato ficheros de videos.
- `mencoder <fichero de entrada> -ovc xvid -oac mp3lame -oac mp3lame -lameopts abr:br=192 -xvidencopts pass=2:bitrate=-700000 -o <fichero de salida>`

Memoria...



Windows Task Manager

File Options View Help

Applications Processes Services Performance Networking Users

Image Name	CPU	Memory	Command Line
X10nets.exe	00	1,826 K	C:\Program Files\Common Files\X10\Common\X10...
BCSSync.exe	00	172 K	C:\Program Files\Microsoft Office\Office 14\BCSSy...
mencoder.exe	00	13,429 K	C:\Program Files\FormatFactory\FFModules\Enco...
FNPLicensingService.exe	00	1,442 K	C:\Program Files\Common Files\Macrovision Share...
pcCMService.exe	00	224 K	C:\Program Files\Common Files\Motive\pcCMServi...

- **13.5 MB** de memoria RAM
- **Es poco**, puede ejecutarse con otros programas sobre el mismo nodo → **No reservar el nodo en exclusivo.**

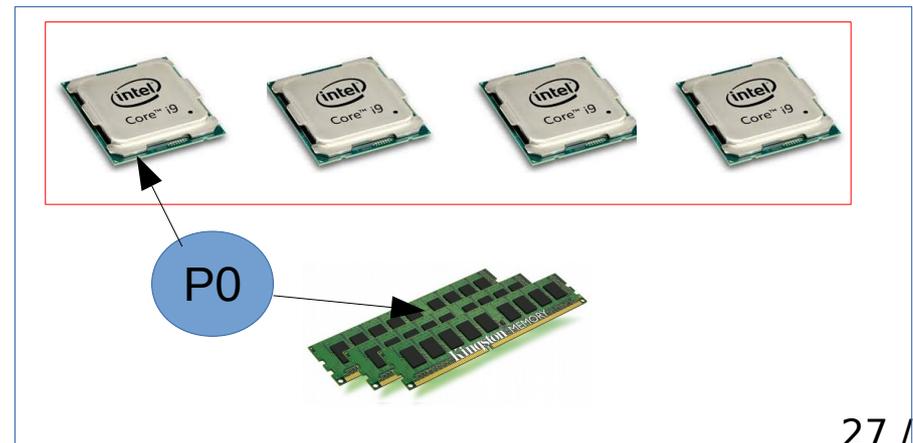
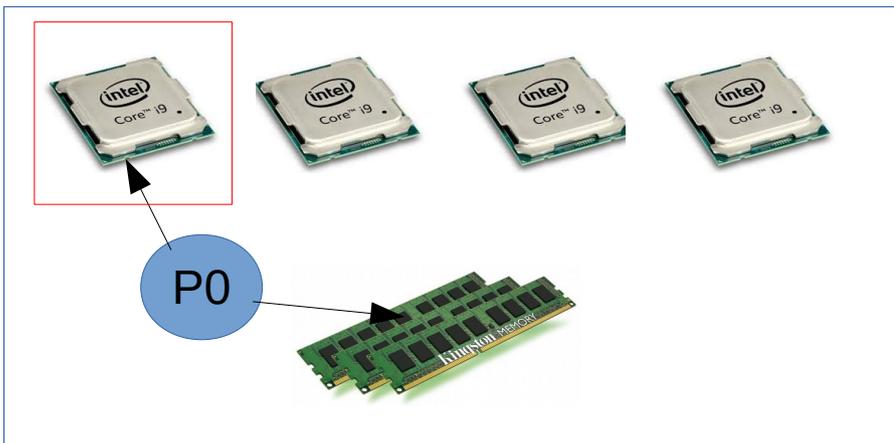
Memoria sobre Linux

```
artigas@GPU01: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
top - 12:11:13 up 41 min, 1 user, load average: 0,11, 0,19, 0,34  
Tareas: 225 total, 1 ejecutar, 224 hibernar, 0 detener, 0 zombie  
%Cpu(s): 4,4 usuario, 1,3 sist, 0,0 adecuado, 93,4 inact, 0,7 en espera, 0,0 hardw int, 0,2 softw int, 0,0 robar tiempo  
MiB Mem : 3882,8 total, 186,9 libre, 1689,2 usado, 2006,7 búfer/caché  
MiB Intercambio: 976,0 total, 975,2 libre, 0,8 usado. 1766,3 dispon Mem
```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
4205	artigas	20	0	2810308	453244	185624	S	7,3	11,4	2:38.54	firefox
1659	root	20	0	445860	129644	106504	S	3,3	3,3	1:17.91	Xorg
6489	artigas	20	0	1088224	48228	30624	S	2,3	1,2	0:01.50	mate-screenshot
1126	root	20	0	444388	11836	8916	S	1,7	0,3	0:00.55	udisksd
3142	artigas	20	0	743584	49940	28640	S	1,9	1,3	0:10.98	marco
4909	artigas	20	0	1863592	282000	115944	S	1,9	7,1	3:53.42	Web Content
3265	artigas	20	0	473328	33840	27884	S	0,7	0,9	0:00.54	mate-screensave
4290	artigas	20	0	1753696	273648	170772	S	0,7	6,9	0:33.12	Web Content
4408	artigas	20	0	1461484	128304	83908	S	0,7	3,2	0:07.58	WebExtensions
5918	artigas	20	0	1623116	190896	113044	S	0,7	4,8	0:27.86	Web Content
5983	artigas	20	0	1685492	197432	106600	S	0,7	5,0	0:18.36	Web Content
10	root	20	0	0	0	0	I	0,3	0,0	0:00.53	rcu_sched
222	root	20	0	0	0	0	S	0,3	0,0	0:01.17	usb-storage
1887	debian-+	20	0	41412	19804	8784	S	0,3	0,5	0:01.64	tor
3137	artigas	20	0	1240164	37204	27592	S	0,3	0,9	0:01.00	mate-settings-d
6872	artigas	20	0	29092	3956	3272	R	0,3	0,1	0:00.42	top
1	root	20	0	197264	9456	6728	S	0,0	0,2	0:01.74	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd

Requerimientos sobre HPC

- Programa secuencial → 1 nodo
- CORES → 1 core
- ¿Por qué no sirve de nada reservar más de un nodo o más de un core en este caso?



Script ejemplo (I)

```
#!/bin/bash
```

```
#SBATCH -p public # partition (queue)
```

```
#SBATCH --nodes=1
```

```
#SBATCH --ntasks-per-node=1
```

```
#SBATCH -o mencoder_%j.out # STDOUT
```

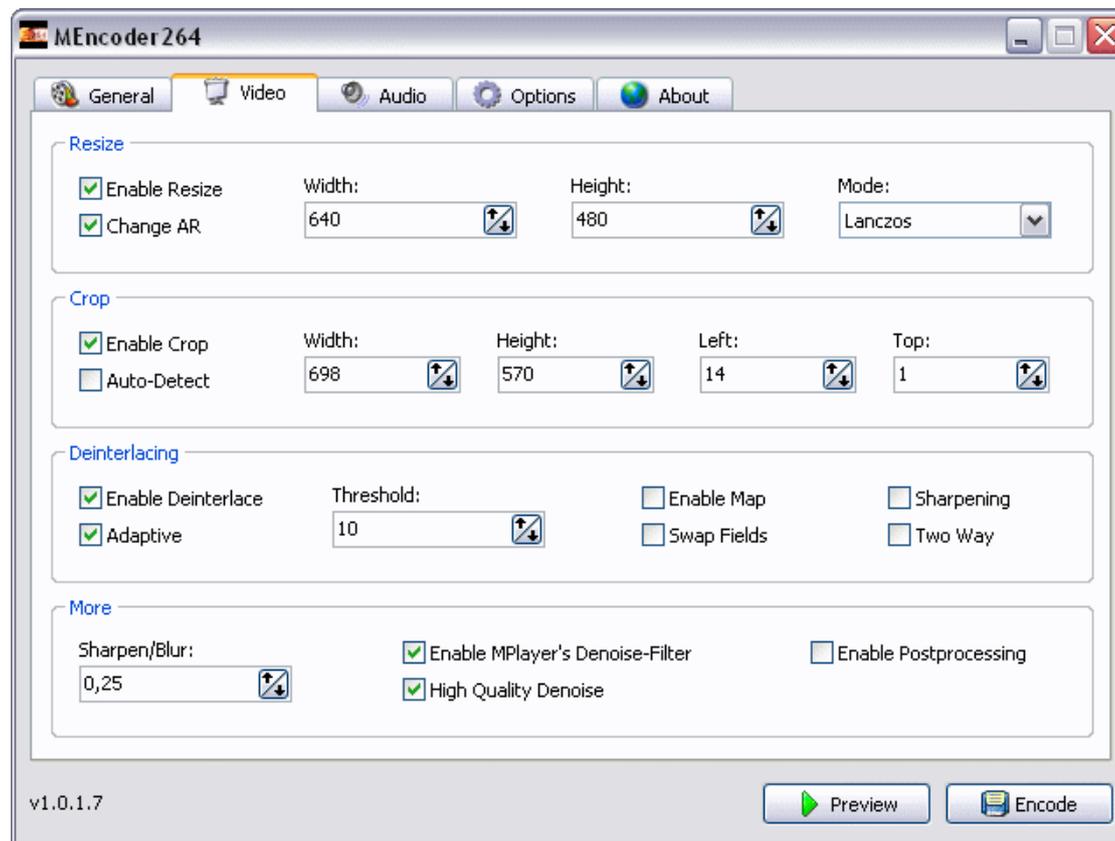
```
#SBATCH -e mencoder_%j.err # STDERR
```

```
mencoder entrada.mkv -ovc xvid -oac mp3lame -oac  
mp3lame -lameopts abr:br=192 -xvidencopts  
pass=2:bitrate=-700000 -o salida.avi
```

Caso de estudio II

Un programa para memoria compartida

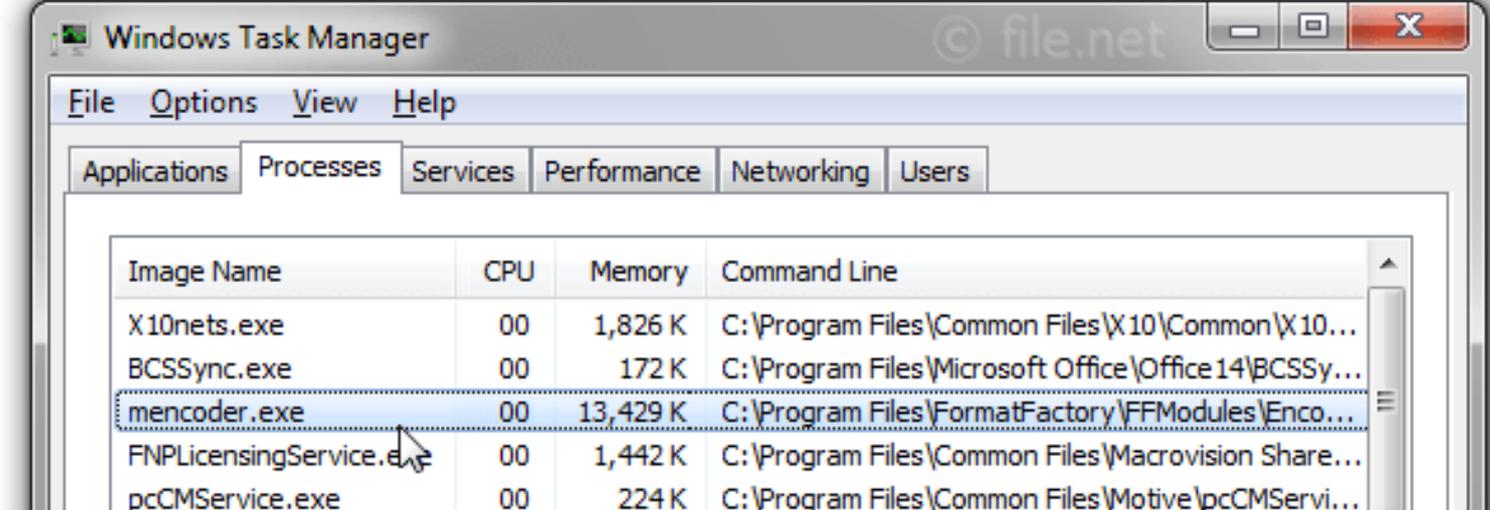
MEMCODER



MENCODER

- Programa para convertir de formato ficheros de videos.
- `mencoder <fichero de entrada> -ovc xvid -oac mp3lame -oac mp3lame -lameopts abr:br=192 -xvidencopts pass=2:bitrate=-700000:threads=4 -o <fichero de salida>`

Memoria...



Windows Task Manager

File Options View Help

Applications Processes Services Performance Networking Users

Image Name	CPU	Memory	Command Line
X10nets.exe	00	1,826 K	C:\Program Files\Common Files\X10\Common\X10...
BCSSync.exe	00	172 K	C:\Program Files\Microsoft Office\Office 14\BCSSy...
mencoder.exe	00	13,429 K	C:\Program Files\FormatFactory\FFModules\Enco...
FNPLicensingService.exe	00	1,442 K	C:\Program Files\Common Files\Macrovision Share...
pcCMService.exe	00	224 K	C:\Program Files\Common Files\Motive\pcCMServi...

- **13.5 MB** de memoria RAM
- **Es poco**, puede ejecutarse con otros programas sobre el mismo nodo → **No reservar el nodo en exclusivo.**

Requerimientos sobre HPC

- Memoria compartida → 1 nodo
- CORES (hasta 32 disponibles) → 1 hasta 32
 - Para procesar un fichero de entrada con 4 hilos va bien, entonces cores = 4, .
 - Para procesar MUCHOS ficheros de entrada, entonces cores = 32. ¿Reservar el nodo completo?

Script ejemplo (I)

```
#!/bin/bash
```

```
#SBATCH -p public # partition (queue)
```

```
#SBATCH --nodes=1
```

Siempre = 1

Entre 1 y 32

```
#SBATCH --ntasks-per-node=4
```

```
#SBATCH -o mencoder_%j.out # STDOUT
```

```
#SBATCH -e mencoder_%j.err # STDERR
```

```
mencoder entrada.mkv -ovc xvid -oac mp3lame -oac  
mp3lame -lameopts abr:br=192 -xvidencopts  
pass=2:bitrate=-700000:threads=4 -o salida.avi
```

Debe coincidir con
ntasks-per-node

Script ejemplo (II)

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --ntasks-per-node=32
#SBATCH -o mencoder_%j.out # STDOUT
#SBATCH -e mencoder_%j.err # STDERR

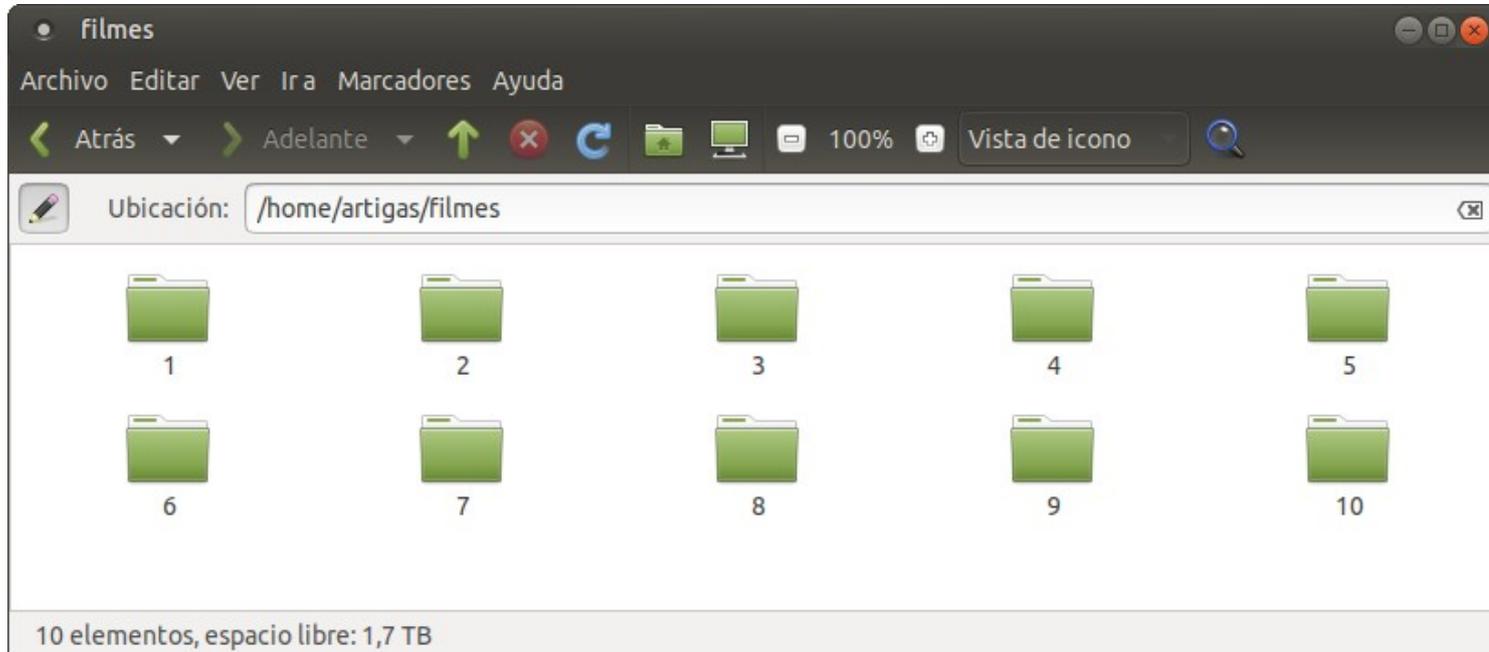
for infile in *.mkv #Sponga que son 100 ficheros a convertir
do
mencoder ${infile} -ovc xvid -oac mp3lame -oac mp3lame -
lameopts abr:br=192 -xvidencopts pass=2:bitrate=-
700000:threads=32 -o ${infile}.avi
done
```

Reservar de modo exclusivo,
aunque el sistema ya lo hace,
¿por qué?

¿Puedo ir más allá?

- Suponga que tiene que procesar 1000 ficheros de video de High Definition (7GB cada uno).
- Sobre un nodo demora una eternidad.
- **Solución: usar paralelismo de datos y ... un poco de bricolage.**

!Puedo ir más allá!

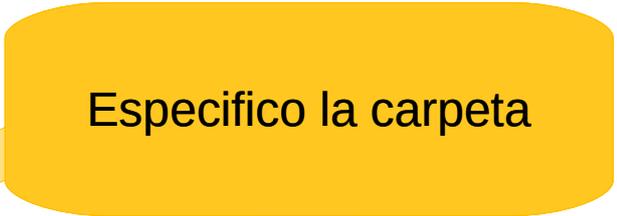


En cada carpeta almaceno, por ejemplo, **100 ficheros** de entrada, y ahora tengo **¡10 problemas distintos!**

Script ejemplo (III)

```
#!/bin/bash
#----- encoder_script -----
#SBATCH -p public # partition (queue)
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --ntasks-per-node=32
#SBATCH -o mencoder_%j.out # STDOUT
#SBATCH -e mencoder_%j.err # STDERR
for infile in $1/*.mkv
do
mencoder ${infile} ... bitrate=-700000:threads=32 -o ${infile}.avi
done
#-----

cd /home/artigas/filmes
for i in 1 2 3 4 5 6 7 8 9 10
do
    sbatch encoder_script $i
done
```

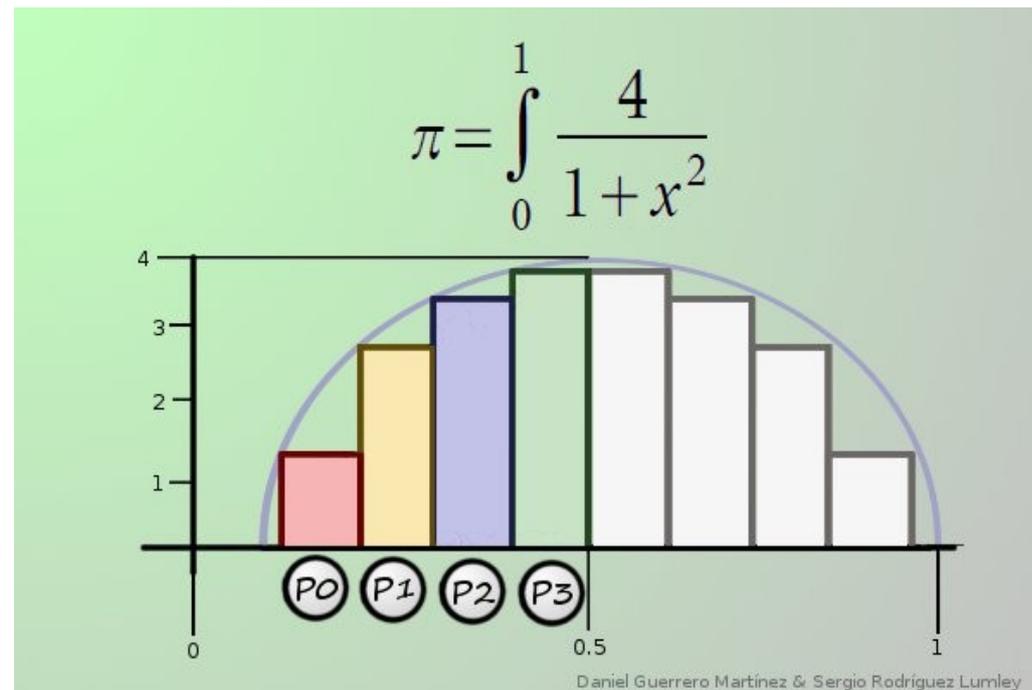


Especifico la carpeta

Caso de estudio III

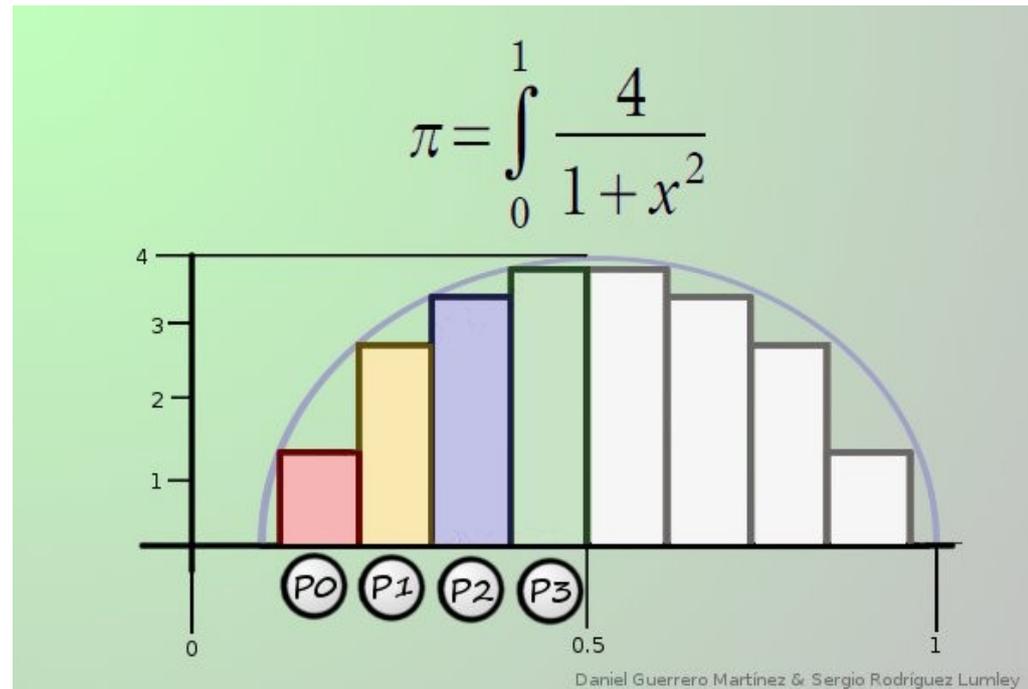
Un programa para memoria compartida

Paso de mensajes con MPI



Cálculo de PI en paralelo

- Esquema **puro** de memoria distribuida
 - Se usan **varios** nodos
 - En cada nodo se usa **un único** procesador



Ejemplo de script para MPI

```
#!/bin/bash
#----- encoder_script -----
#SBATCH -p public # partition (queue)
#SBATCH --nodes=10
#SBATCH --exclusive
#SBATCH --ntasks-per-node=1
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

```
#Cargar el módulo de MPI
module load OpenMPI
```

```
#ejecutar 10 copias del programa, una sobre cada nodo
mpirun -np 10 parallel_mpi
```

Totalmente ineficiente, a menos que pruebe escalabilidad, o cada copia use una cantidad enorme de memoria RAM

Ventajas de MPI

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=10
#SBATCH --exclusive
#SBATCH --ntasks-per-node=1
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --ntasks-per-node=10
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

- Funciona lo mismo para memoria compartida que para distribuida.
- La elección dependerá del **consumo de memoria y del costo de las comunicaciones.**

Ventajas de MPI

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=10
#SBATCH --ntasks-per-node=1
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=10
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

```
#!/bin/bash
#SBATCH -p public # partition (queue)
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=2
#SBATCH -o pi_%j.out # STDOUT
#SBATCH -e pi_%j.err # STDERR
```

- Funciona como un esquema híbrido.

Elección del modelo ...

Para una misma cantidad total de cores

Menor consumo de memoria

Mayor consumo de memoria



Menos nodos
Más cores x nodo

Equilibrio

Más nodos
Menos cores x nodo

Memoria
compartida

Esquema híbrido

Memoria
distribuida

Si necesito más cores



Reservo **más nodos** y
por lo tanto dispongo
de **más cores** y **más
memoria**

¿Número ideal de nodos?
¿Número ideal de cores?

Ver conferencia 2



Caso de estudio complejo
Autodock
Las 1001 formas de ejecutarlo

Autodock

The screenshot displays the Autodock Vina software interface. The main window shows a 3D visualization of a protein-ligand complex. The protein is represented as a ribbon structure, colored in shades of blue, green, and orange. A yellow wireframe box indicates the docking site. The ligand is shown as a stick model, colored in red and white, docked within the protein's binding pocket.

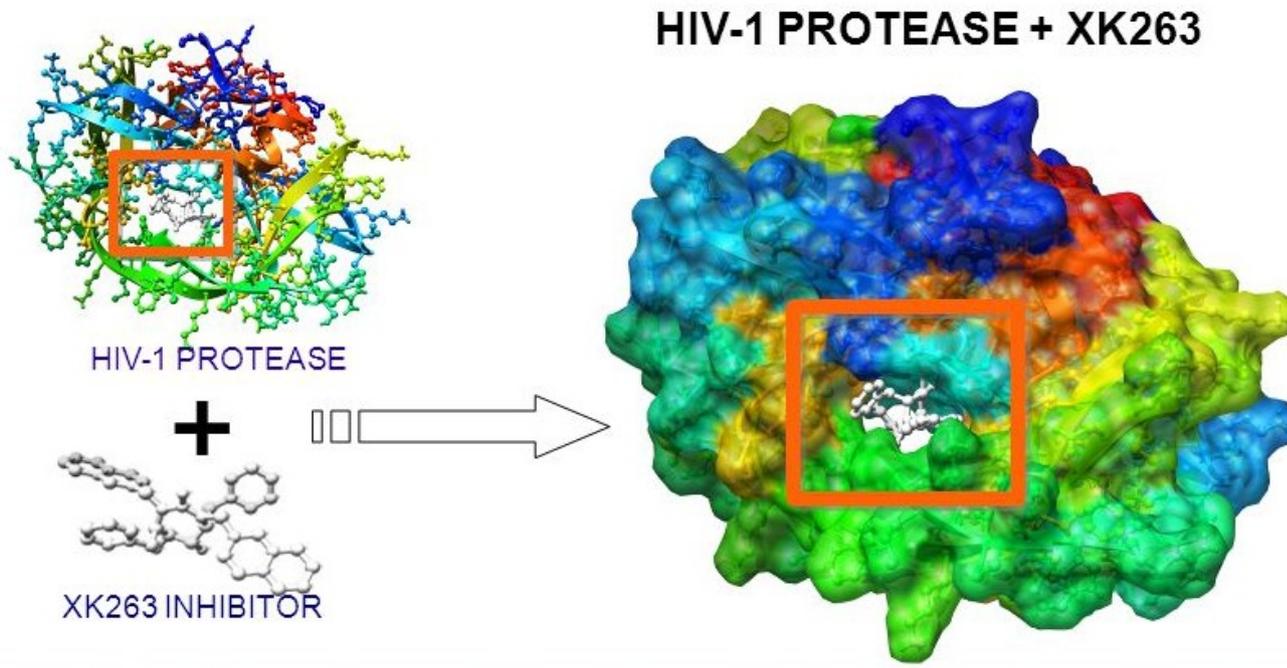
The interface includes a menu bar (File, Edit, Selection, Visualization, Simulation, App, Window) and a toolbar with various icons for file operations, selection, and visualization. On the left, a 'Document view' panel shows a tree structure of the current document, including 'VinaTest.sam', 'Camera 1', 'Layer 1', 'Receptor', 'Ligand', and 'Flexible residues'. On the right, a 'History' panel lists various actions like 'New document', 'Modify selection', and 'Add secondary structure'. A vertical toolbar on the far right contains icons for different visualization modes and tools.

The 'AutoDock Vina' configuration panel is open, showing the following settings:

- 1 - Setup receptor and ligand:** Buttons for 'Set receptor', 'Show receptor', 'Set flexible side-chains', 'Show flexible side-chains', 'Set ligand', and 'Show ligand'.
- 2 - Setup grid:** Center X: 42,00 A, Size X: 22,00 A, Center Y: 36,00 A, Size Y: 19,00 A, Center Z: 109,00 A, Size Z: 24,00 A.
- 3 - Dock:** Exhaustiveness: 100, Modes: 200. A progress bar shows 'Docking...' at 17%.
- 4 - Visualize results:** A dropdown menu showing 'No docking results yet'.

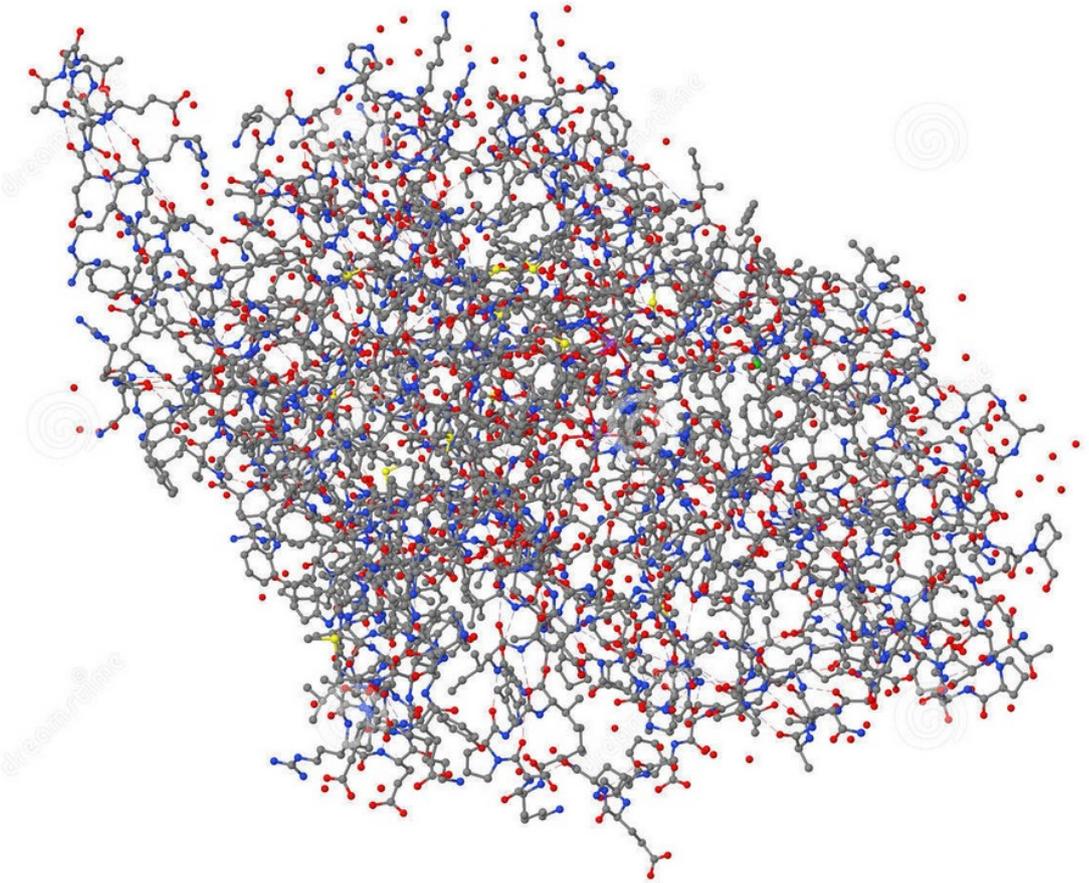
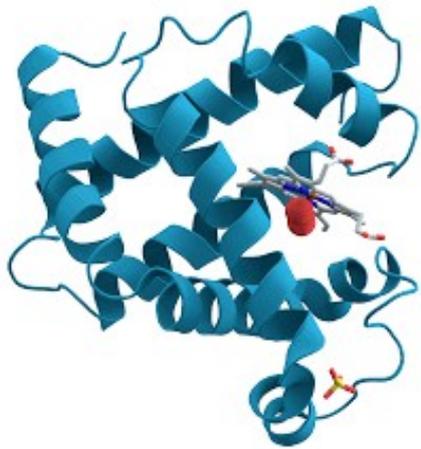
At the bottom left of the interface, a status bar indicates: '6 side-chains were set as flexible'.

Acoplamiento molecular



En el campo del Modelado molecular, es un método que predice la conformación preferida de una molécula, al estar unida a otra, con el fin de formar un complejo estable. (WIKIPEDIA)

Autodock



Necesidades de capacidad de cómputo y memoria variadas

AutoDock Paralelo

Article

Parallel implementation of AutoDock

June 2007 · Journal of Applied Crystallography 40(3):598-599

DOI: 10.1107/S0021889807011053

Prashant Khodade · R. Prabhu ·  Nagasuma R Chandra · [Show all 5 authors](#) · R. Govindarajan

Overview

Stats

Comments

Citations (29)

References (4)

Related rese

Abstract

Computational docking of ligands to protein structures is a key step in structure-based drug design. Currently, the time required for each docking run is high and thus limits the use of docking in a high-throughput manner, warranting parallelization of docking algorithms. AutoDock, a widely used tool, has been chosen for parallelization. Near-linear increases in speed were observed with 96 processors, reducing the time required for docking ligands to HIV-protease from 81 min, as an example, on a single IBM Power-5 processor (1.65 GHz), to about 1 min on an IBM cluster, with 96 such processors. This implementation would make it feasible to perform virtual ligand screening using AutoDock.

Autodock, 1001 vías de...

Journal List > Springer Open Choice > PMC4801993



JOURNAL OF COMPUTER-AIDED
MOLECULAR DESIGN

▶ springer.com

[J Comput Aided Mol Des.](#) 2016; 30: 237–249.

PMCID: PMC4801993

Published online 2016 Feb 20. doi: [10.1007/s10822-016-9900-9](https://doi.org/10.1007/s10822-016-9900-9)

PMID: [26897747](https://pubmed.ncbi.nlm.nih.gov/26897747/)

1001 Ways to run AutoDock Vina for virtual screening

[Mohammad Mahdi Jaghoori](#), [Boris Bleijlevens](#), and [Silvia D. Olabariaga](#)

▶ [Author information](#) ▶ [Article notes](#) ▶ [Copyright and License information](#) [Disclaimer](#)

Autodock, 1001 vías de ...

Table 2

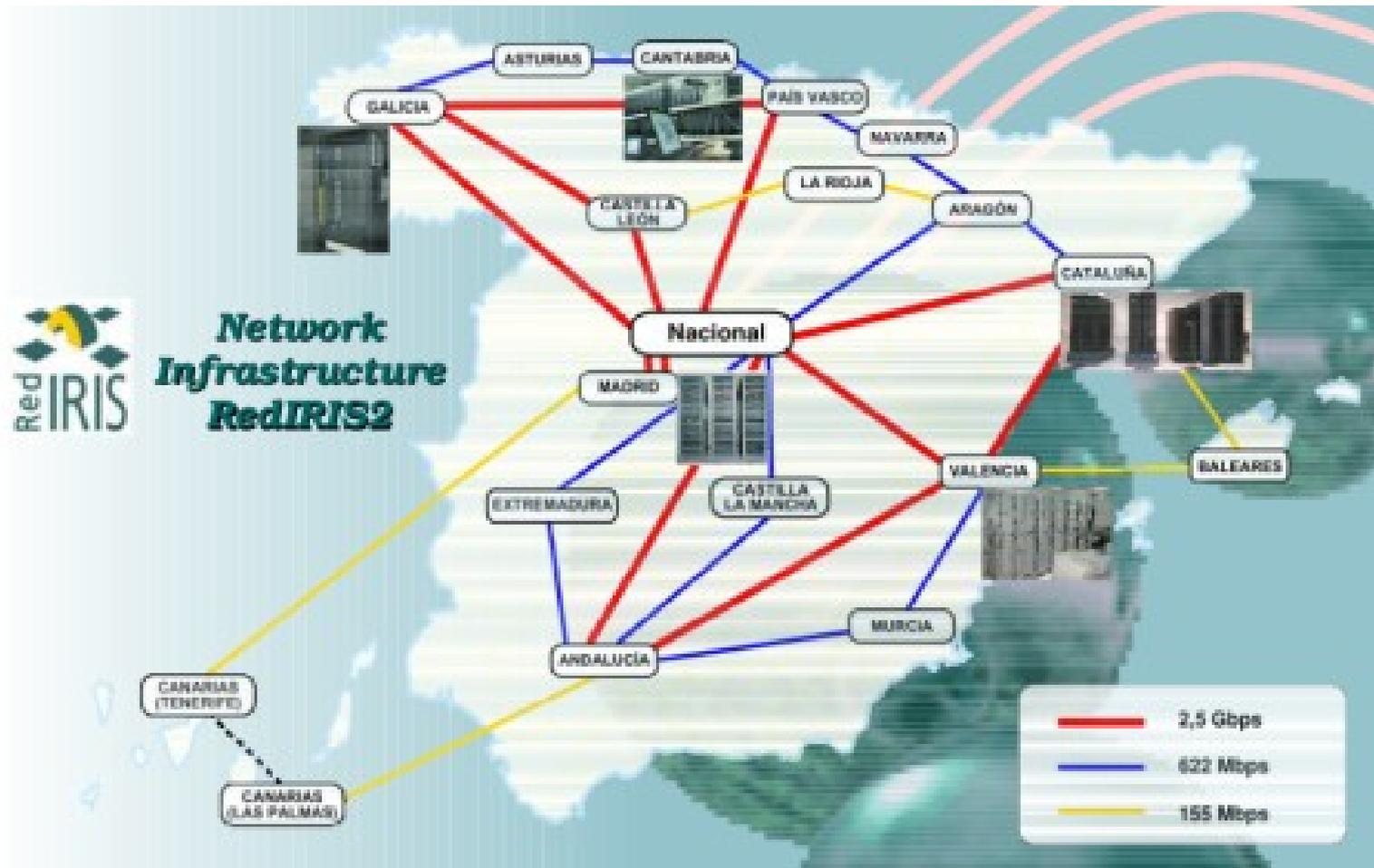
This table shows the four ligand libraries, four infrastructures, and the three docking boxes (FTO and the big and small boxes on NUR77)

	Multi-core	AMC cluster	Hadoop cluster	Grid
Nutra	Small	Small	Small	Small
	Big	Big	Big	Big
	FTO	FTO	FTO	FTO
HMDB	Small	Small	Small	Small
	Big	Big	Big	Big
	FTO	FTO	FTO	FTO
FDA	Small	Small	Small	Small
	Big	Big	Big	Big
	FTO	FTO	FTO	FTO
ZNP	Small	Small	Small	Small
	Big	Big	Big	Big
	FTO	FTO	FTO	FTO

When a name “Small”, “Big” or “FTO” is in black, it shows that we ran VS experiments with that docking box on the corresponding infrastructure and ligand libraries

- Sobre un nodo multicore.
- Sobre Clúster “HPC”
- Sobre un Clúster “Hadoop” (BigData)
- Sobre un GRID

Computación en GRID



Autodock ...

- ¿Qué cantidad de recursos debo reservar?
 - Depende del tamaño del problema a tratar.
 - Pudiera considerar tres tipos de problemas típicos:
 - Pequeño
 - Mediano
 - Grande
 - Ver la conferencia 2.



Resumen

TIPs para uso de nodos

- Programas secuenciales → **UN SOLO NODO**
- Programas para memoria compartida → **UN SOLO NODO**
- Programas para memoria distribuida → **Varios nodos**
- Programas híbridos → **Varios nodos**

Ver conferencia 2

TIPs para uso de cores

- Programas secuenciales → **UN SOLO CORE**
- Programas para memoria compartida → varios cores sobre un **solo nodo**
- Programas para memoria distribuida → **Un core sobre varios nodos**
- Programas híbridos → **varios cores sobre varios nodos**

Ver conferencia 2

TIPs para uso de memoria

- Si el programa no tiene requisitos especiales o no los conoce
 - Ejecutar normalmente (...ya explotará)
- Si requiere de mucha memoria
 - → especificar la memoria necesaria en el script (siempre es una buena práctica)
 - → valorar reservar el nodo de modo exclusivo, aunque no use todos los cores.

TIPs para reserva exclusiva

- ¿Cuándo necesito reservar todos los recursos de memoria y potencia de cálculo del nodo?
 - Si el programa tiene requerimientos altos.
 - Si el problema que abordo es complejo y costoso.
- Cuando las interferencias falsean los resultados.
 - Estoy midiendo el **tiempo de ejecución** y me afectan otros procesos. Ej. acceso al disco local.

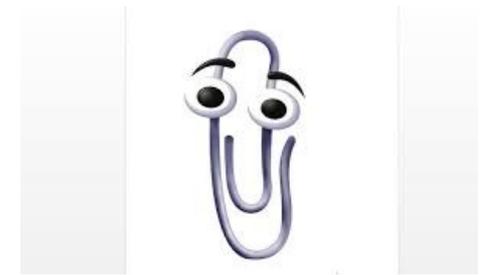
Para la conferencia 2

- ¿Cómo determinar el número ideal de nodos y cores a reservar para un caso particular?
- ¿Existen otras dudas en el uso del clúster?



Para el futuro cercano...

- Asistente virtual para el uso del clúster.





FIN
¿Preguntas?



Aspectos avanzados en el uso del clúster UO

¡GRACIAS!