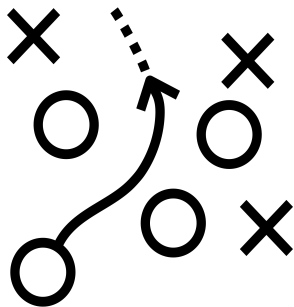


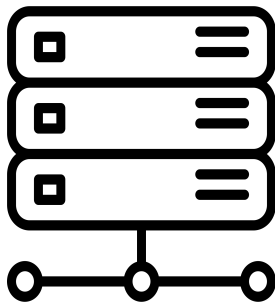
Servidores de datos y supercomputación

Danilo González, PhD.
Erica Bianco, PhD.

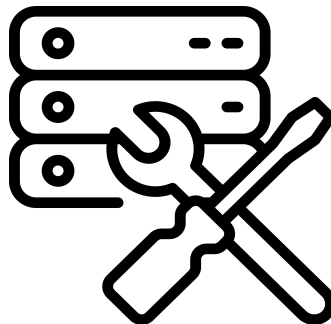
Servicios



CONSULTING



INSTALLATION

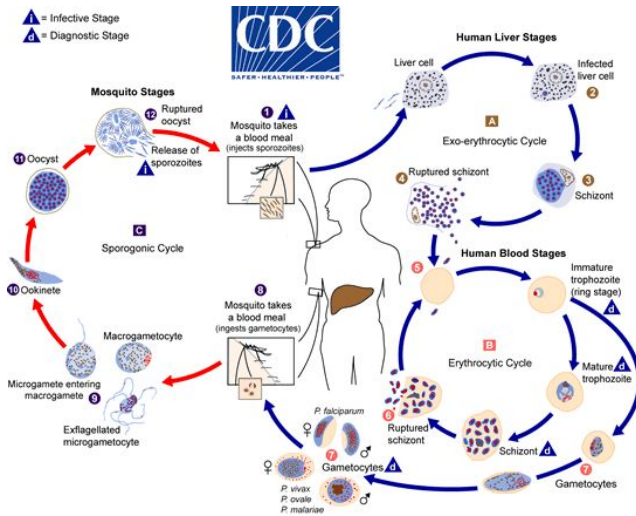
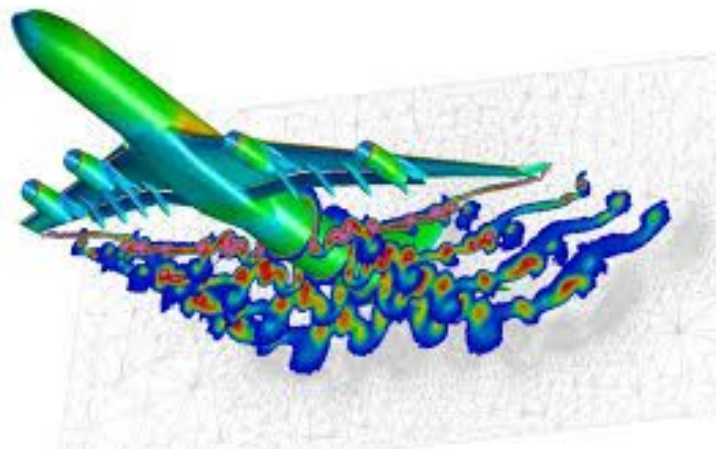
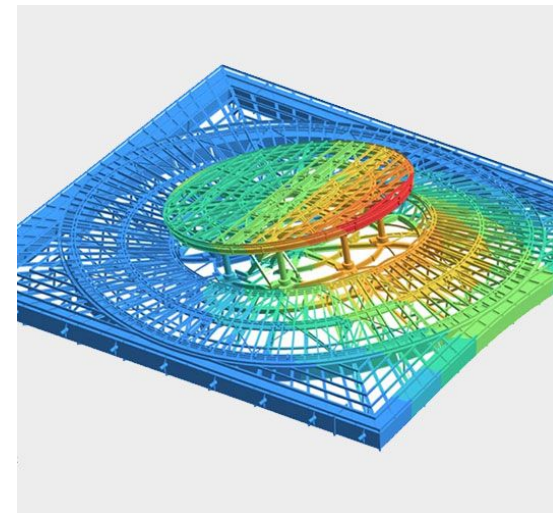
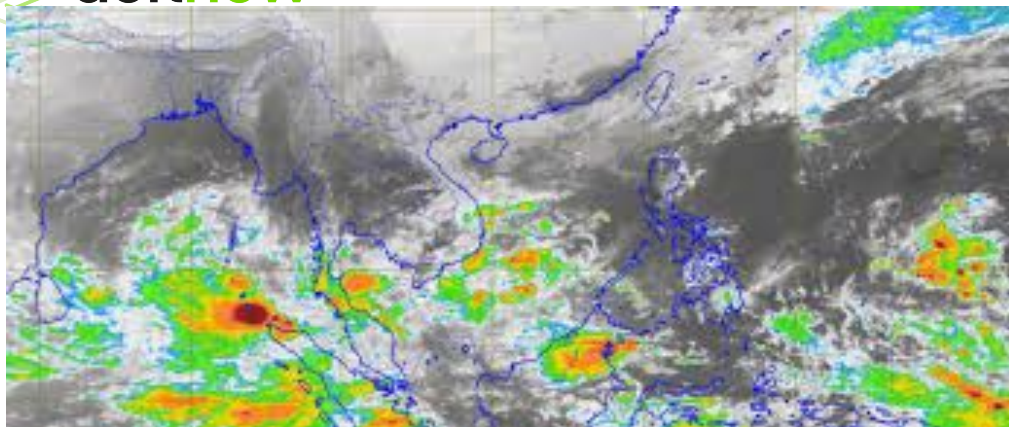


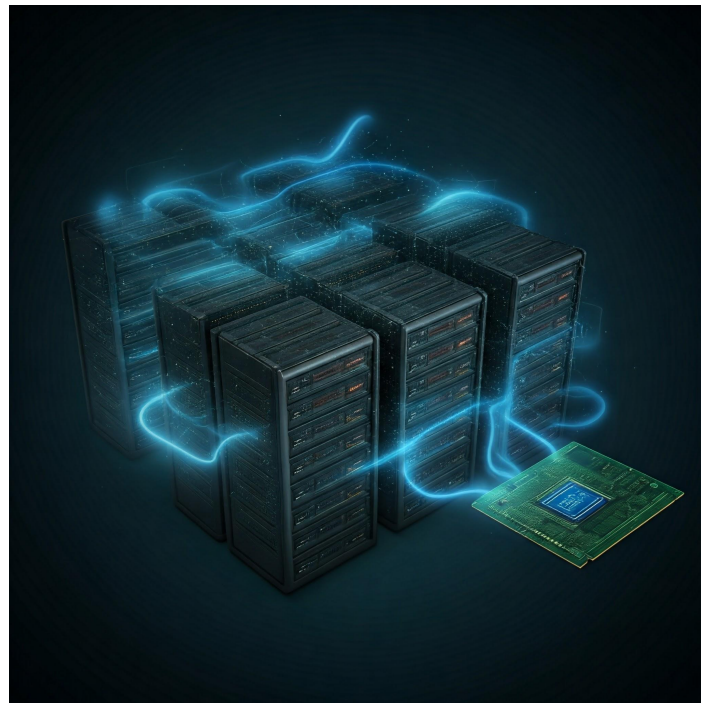
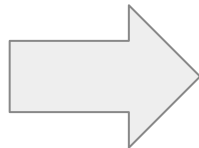
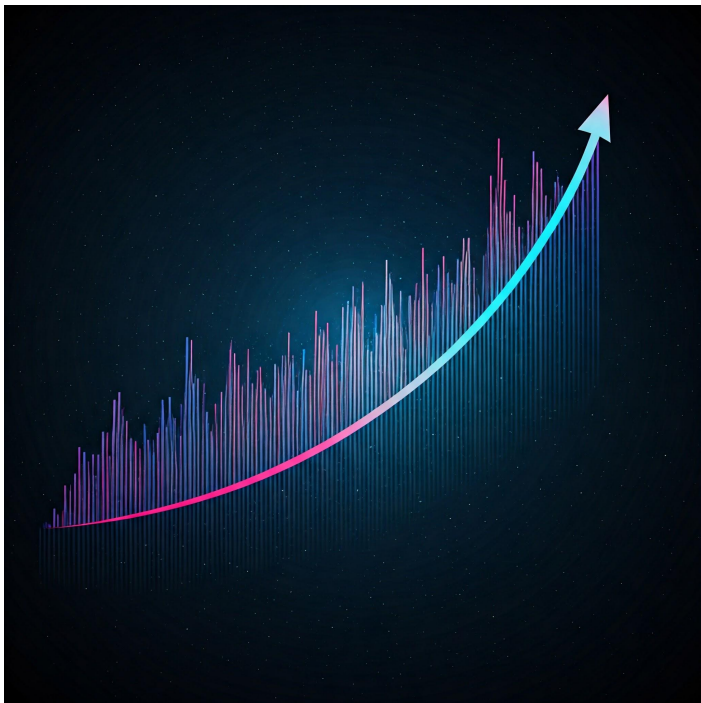
SUPPORT



R&D-as-a-Service

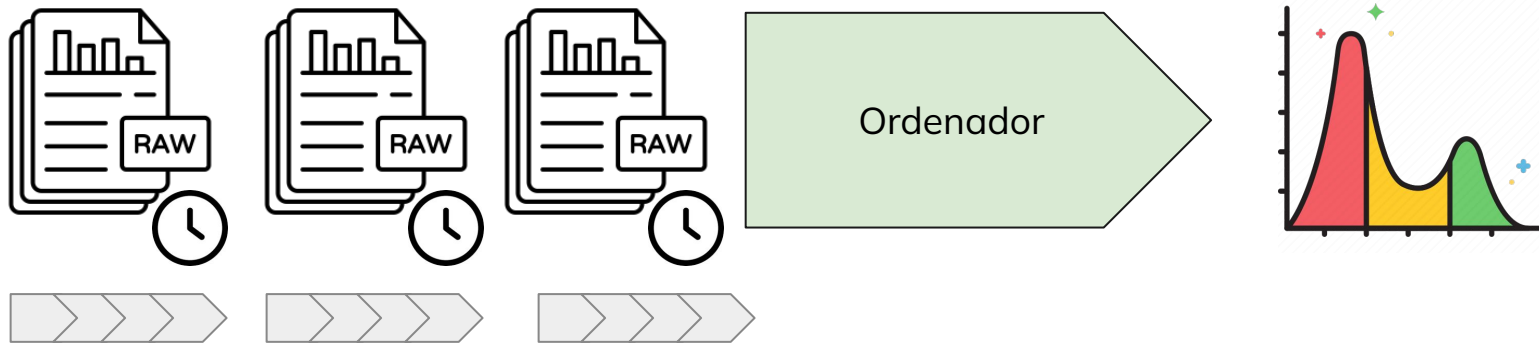
Motivación





Procesar rápidamente gran cantidad de datos requiere
músculo computacional

Super computación 101 (HPC 101)



Los ordenadores comunes hacen las tareas complejas dividiéndolas en más pequeñas. Una vez divididas, se ejecutan secuencialmente.

Super computación 101 (HPC 101)



X 100M

Tarea

Ordenador

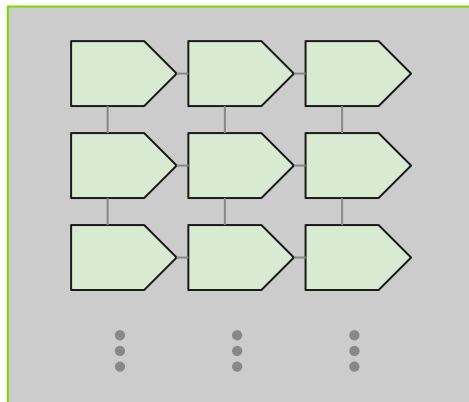


Super computación 101 (HPC 101)



X 100M

Tarea



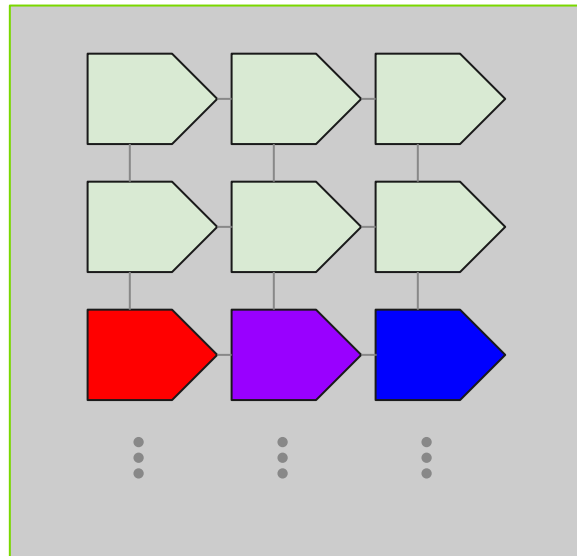
Super computación 101 (HPC 101)

Hardware:

- Servidores (Nodos)
- Red
- Aceleradores (GPU/QC/FPGA)
- Almacenamiento compartido

Software:

- Sistema Operativo
- Gestor de colas
- Software científico



Super computación 101 (HPC 101)

Hardware:

- Servidores (Nodos)
- Red
- Aceleradores (GPU/QC/FPGA)
- Almacenamiento compartido

Software:

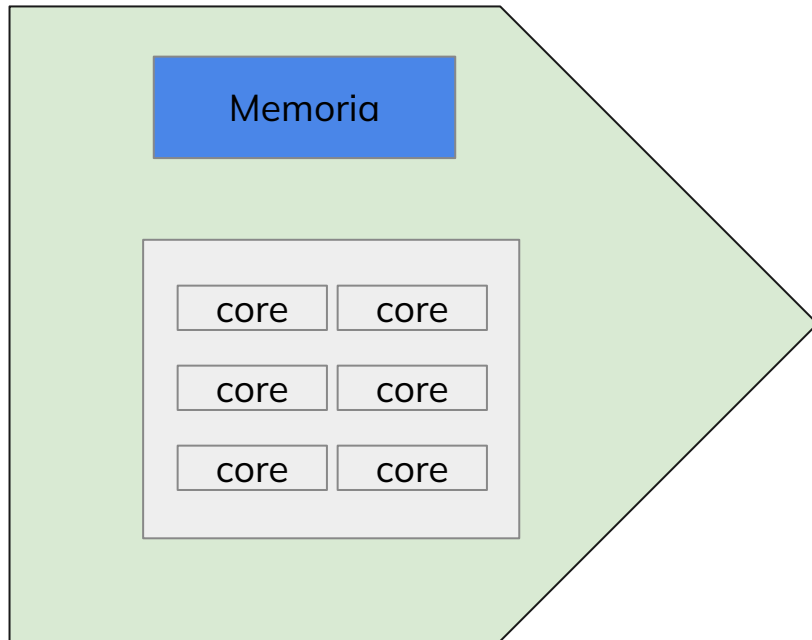
- Sistema Operativo
- Gestor de colas
- Software científico



Hardware Nodo

Componentes:

- CPU(s)
 - Core: unidad de ejecución
- Memoria
- Opcionales:
 - Almacenamiento
 - Aceleradores (GPU/FPGA)



Hardware Red

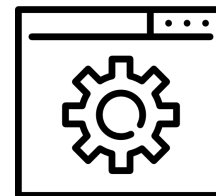
Conecta a los nodos para
compartir **recursos/datos**:

- Latencia: tiempo de respuesta entre nodos
- Ancho de banda
- Topologia: Cómo están conectados



Software

- Sistema operativo (linux-based 99%)
 - Ejecutar las tareas
 - Asegurar que se usen los recursos asignados
- **Gestor de colas**
 - Asignar recursos
 - Software de interacción con el usuario
- **Software científico**
 - Adaptado a HPC



Gestor de colas

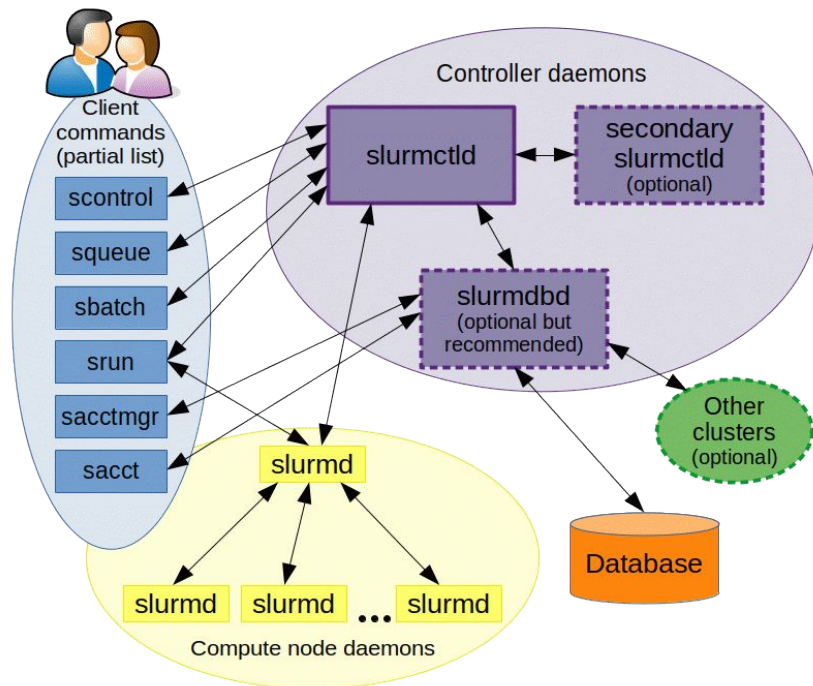
Slurm workload manager



Simple Linux Utility for Resource Management

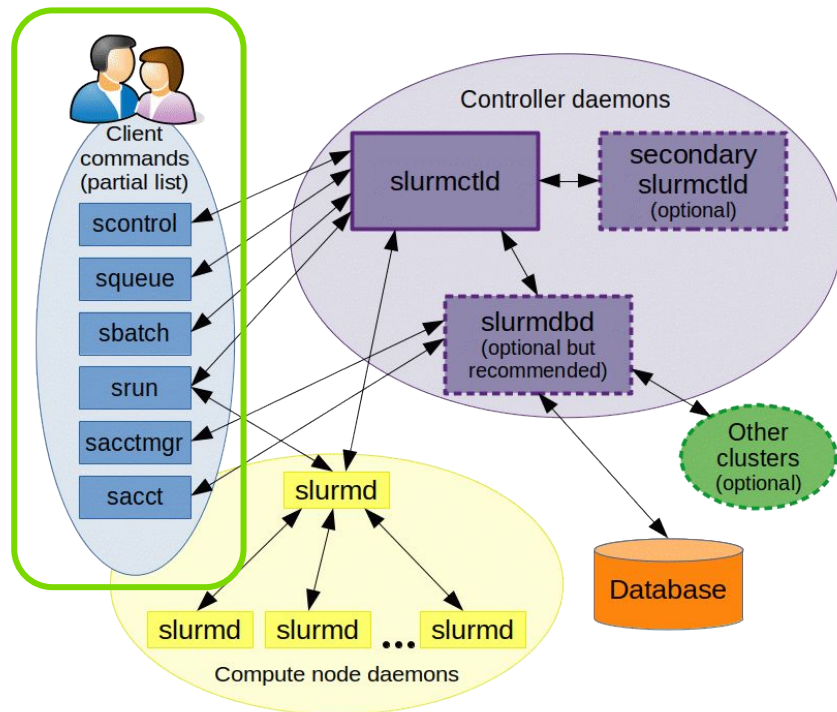
Arquitectura de Slurm

- Usuarios
 - Ejecutar comandos de Slurm
- Nodos de computo
 - Execute jobs
- Nodos de control
- Base de datos
 - Guardar información de los trabajos y usuarios.



Arquitectura de Slurm

- Usuarios
 - Ejecutar comandos de Slurm
- Nodos de computo
 - Execute jobs
- Nodos de control
- Base de datos
 - Guardar información de los trabajos y usuarios.



Comandos básicos: **enviar y cancelar** trabajos

Command	Description
sacct	Displays accounting data for all jobs.
salloc	Allocate resources for interactive use.
sbatch	Submit a job script to a queue
scancel	Signal jobs or job steps that are under the control of SLURM (cancel jobs or job steps)
scontrol	View SLURM configuration and state
sinfo	View information about SLURM nodes and partitions
sjstat	Display statistics of jobs (data from sinfo, squeue and scontrol).
smap	Graphically view information about SLURM jobs, partitions, and set config. param
squeue	View information about jobs located in the SLURM scheduling queue.
srn	Run a parallel task

Enviar un trabajo con sbatch

```
~ $ cat testjob.sl

#!/bin/bash #Interprete
#SBATCH --nodes=4 # Opciones de Slurm

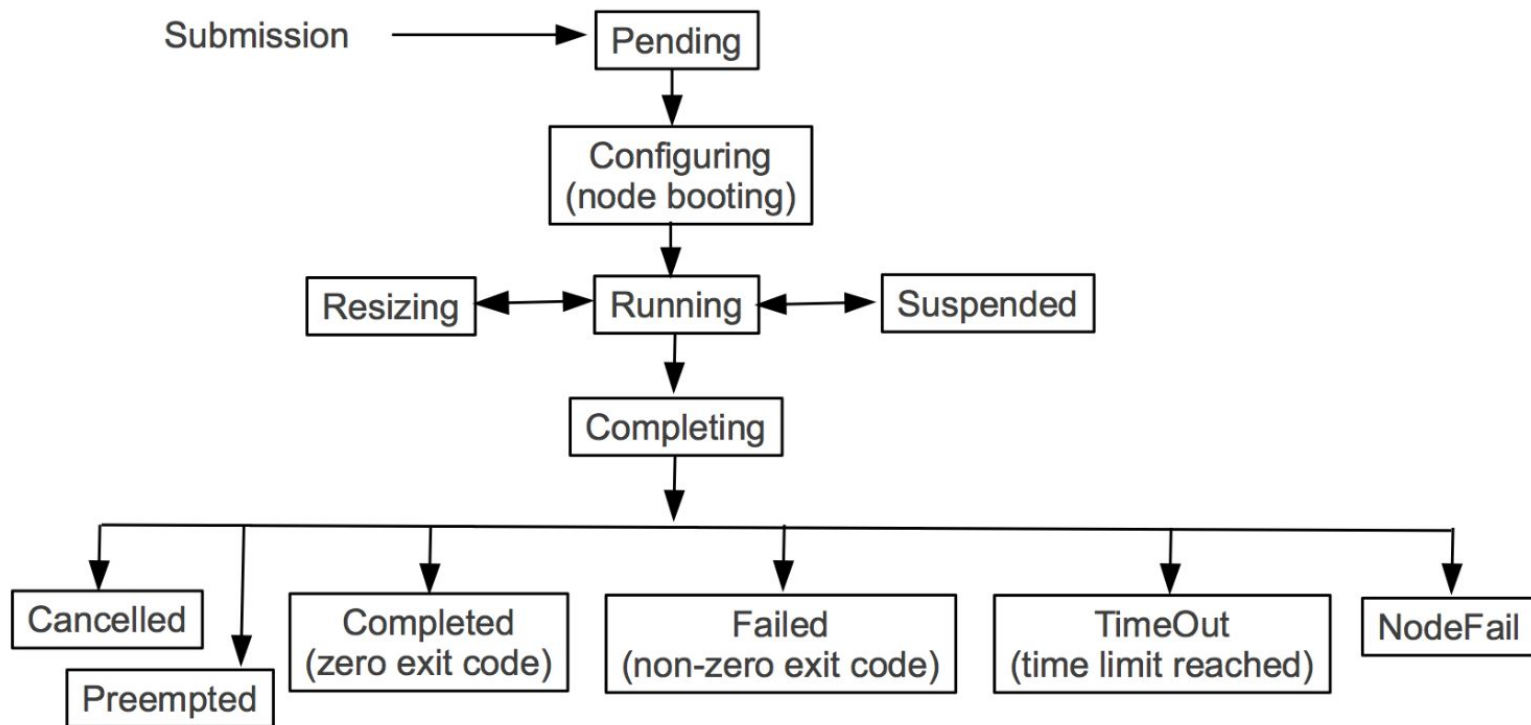
# Comandos a ejecutar
echo "running on : $(hostname)"
echo "allocation : $SLURM_NODELIST"

~ $ sbatch testjob.sl
Submitted batch job 11109

~ $ cat slurm-11109.out
running on : hsw001

allocation : hsw[001-010]
```

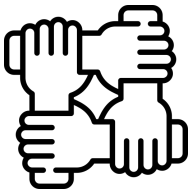
Estado del trabajo



Revisar estado del trabajo

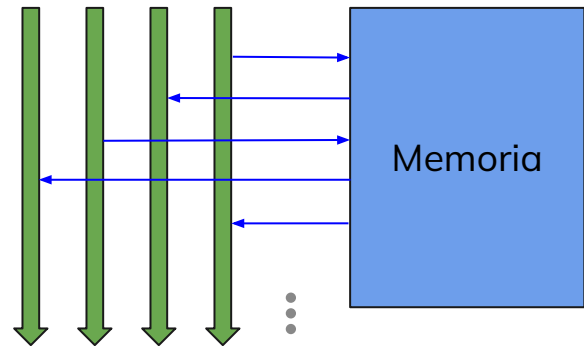
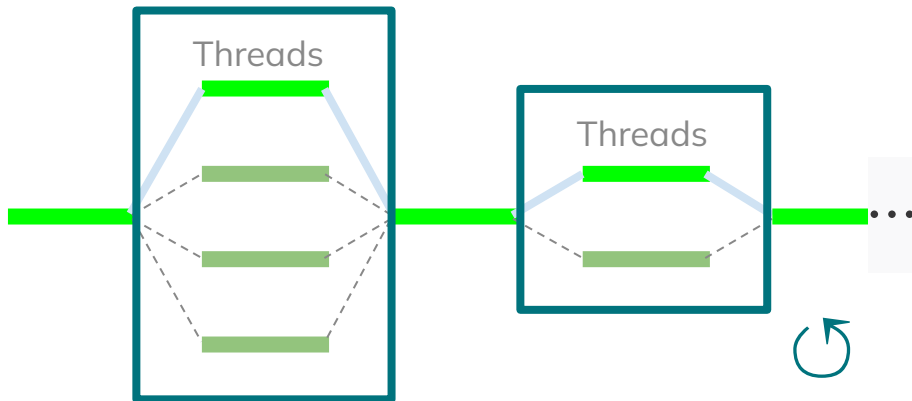
Command	Description
sacct	Displays accounting data for all jobs.
salloc	Allocate resources for interactive use.
sbatch	Submit a job script to a queue
scancel	Signal jobs or job steps that are under the control of SLURM (cancel jobs or job steps)
scontrol	View SLURM configuration and state
sinfo	View information about SLURM nodes and partitions
sjstat	Display statistics of jobs (data from sinfo, squeue and scontrol).
smap	Graphically view information about SLURM jobs, partitions, and set config. param
squeue	View information about jobs located in the SLURM scheduling queue.
srun	Run a parallel task

Computación en Paralelo



Computación en paralelo

Memoria Compartida

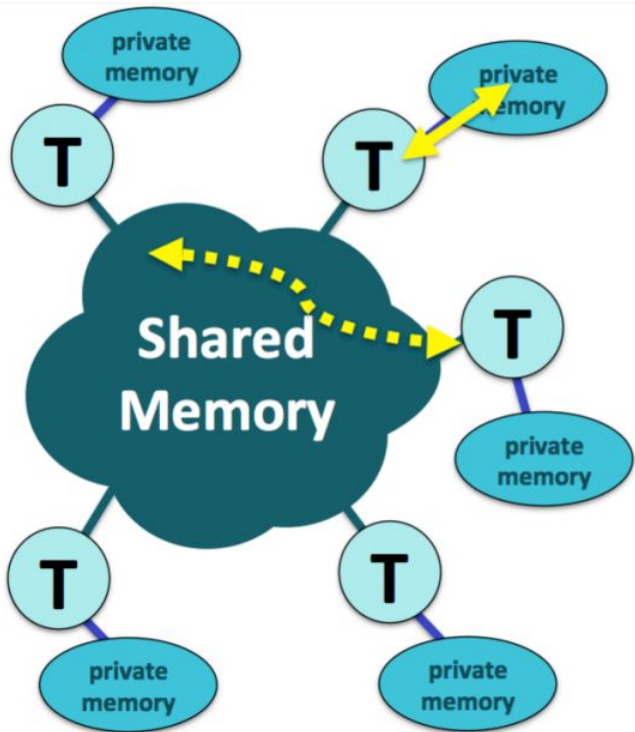
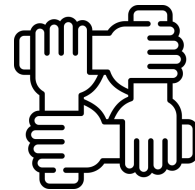


- Cada “thread” tiene acceso toda la memoria
- Inicia con un proceso maestro
- Se crean “threads” que hacen la tarea en paralelo
- Una vez han terminado, se sincronizan los resultados
- Usado en procesos que se ejecutan en el mismo nodo

OpenMP

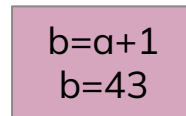
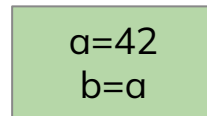
Computación en paralelo

Memoria Compartida

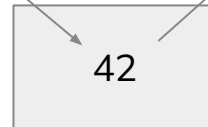


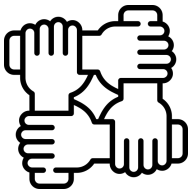
- Cada thread tiene una memoria privada
- La información a compartir se expone en la memoria compartida
- El orden es importante!

Memoria
privada



Memoria
compartida

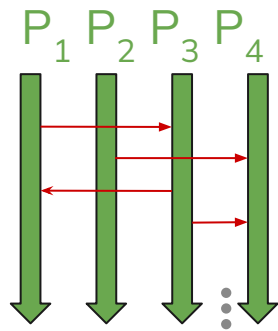
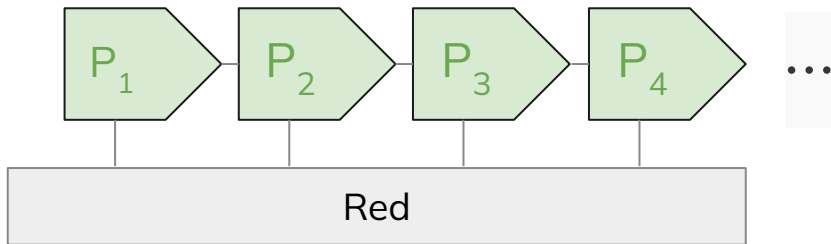


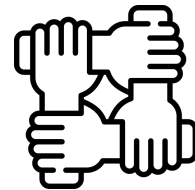


Computación en paralelo

Memoria Distribuida

- Cada proceso tiene acceso la memoria localmente.
- Los procesos se comunican compartiendo información usando **la red**
- MPI (Message Passing Interphase) es el estándar dominante
- Se usa principalmente para procesos internodo.
- Diferentes implementaciones, OpenMPI, Intel MPI, MVAPICH...

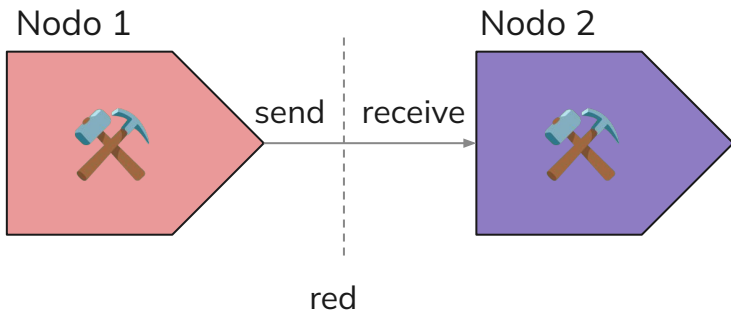




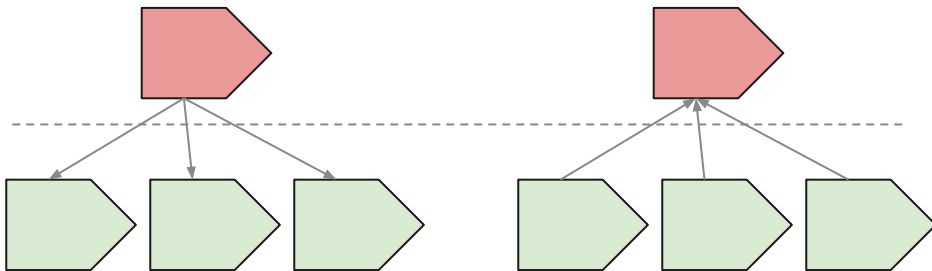
Computación en paralelo

Memoria Distribuida

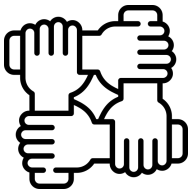
Punto a punto



Colectivo



- Cada proceso tiene acceso a la memoria local del nodo.
- La información se comparte vía mensajes. (orden explícito)



Computación en paralelo

GGPU

General Graphical Processor Unit:

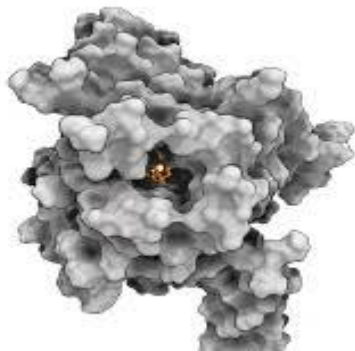
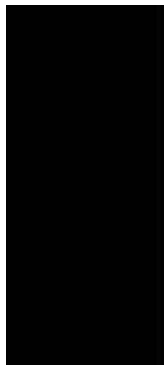
GPUs para uso no gráfico

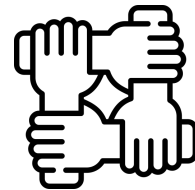
- Muchos cores de potencia limitada
- Eg:
 - AMD EPYC™ 9654: 96 cores, 2 threads por core: **192 threads**
 - NVIDIA A100™: 1080 multiprocessor, 4096 threads por multiprocessor: **4411200 threads!**
- Aplicar una sola instrucción a muchos datos (SIMD)

CPU



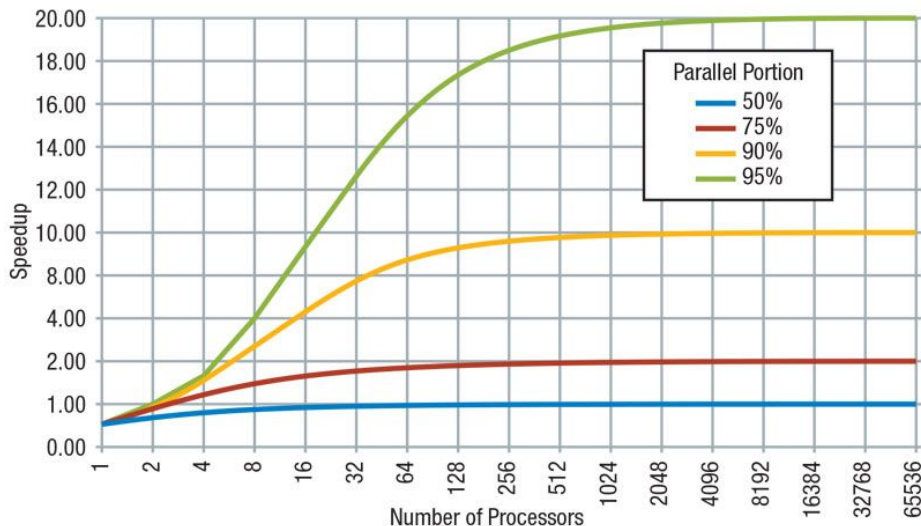
GGPU





Computación en paralelo

Limite (Ley de Amdahl)



$$s_{max} = \frac{1}{(1 - P) + \frac{P}{N}}$$

- A mayor código paralelizado, mas speedup
- Un código no puede ser acelerado infinitamente
- Limitado por las partes no paralelizables (recuperar toda la información)
- Sincronización, Latencias, Velocidad de comunicación....

Software

Appstack centralizado



- Los centros de HPC usualmente tienen un appstack (grupo de aplicaciones) centralizado adaptado para HPC.
- El software puede ser cargado dinámicamente por los usuarios via Environment modules.
- Útil cuando se necesitan diferentes versiones del mismo software

Comandos básicos:

```
man $MODULENAME ## or module help

module list ## currently loaded modules

module avail ## modules available to be loaded

module show $MODULEFILE.lua ## see exactly what a given modulefile will do

module load $MODULENAME ## add a module to the environment

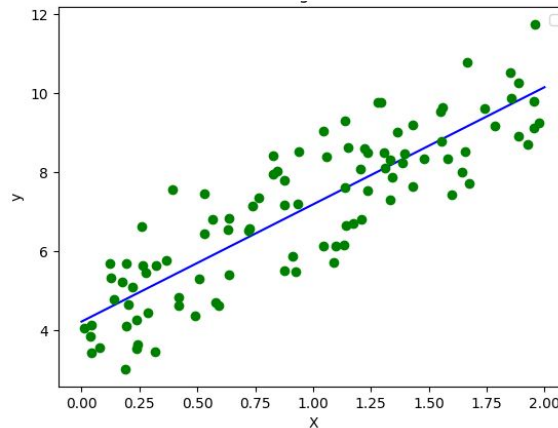
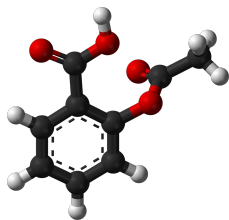
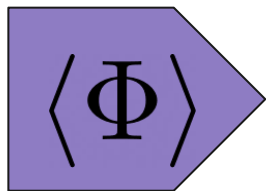
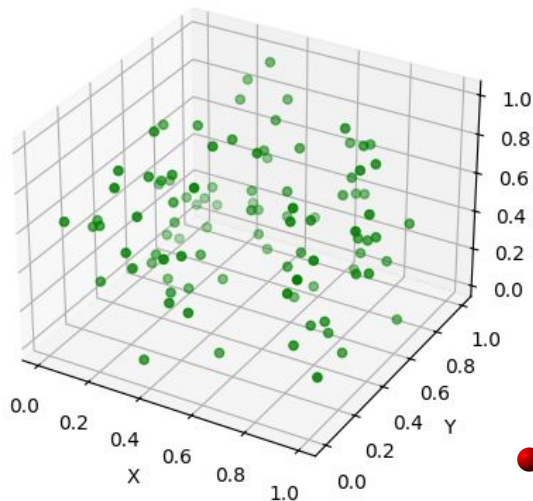
module unload $MODULENAME ## remove a module from the environment

module switch $MODULENAME $NEW_MODULENAME

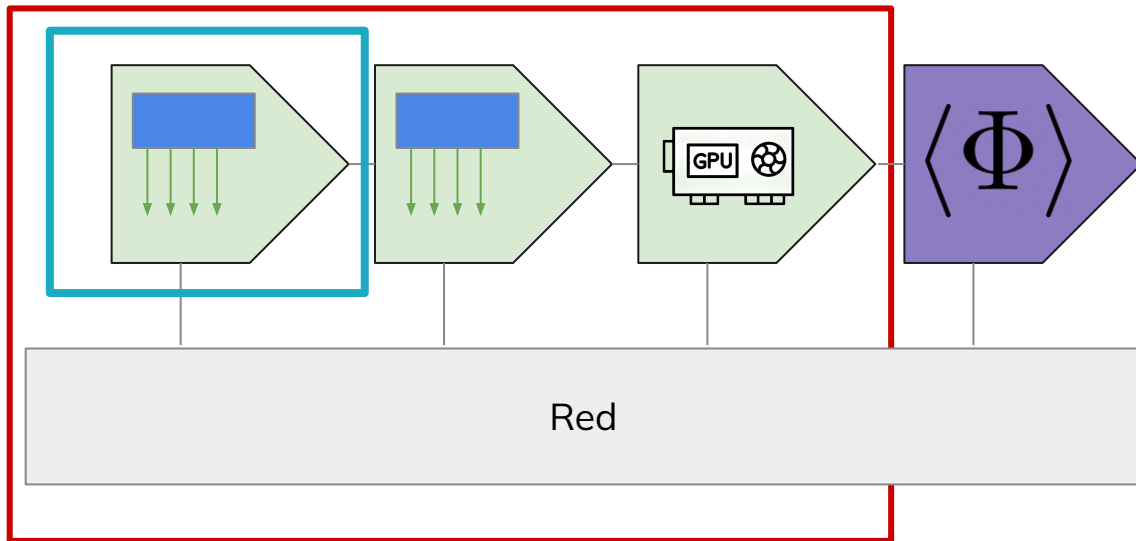
module purge ## unload all active modules
```

Más allá de la paralelización: Quantum Computing

Quantum Reservoir Computing



Visión actual de la supercomputación



Open**MP**

MPI



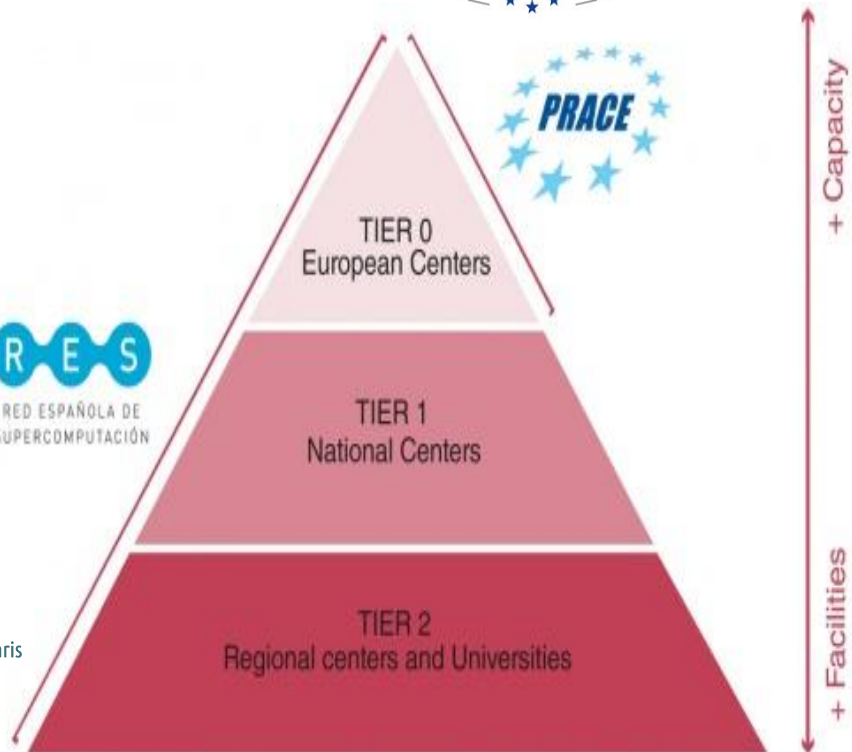
¿Cómo acceder a HPC?

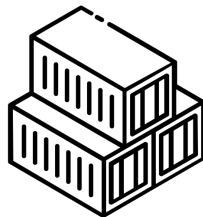
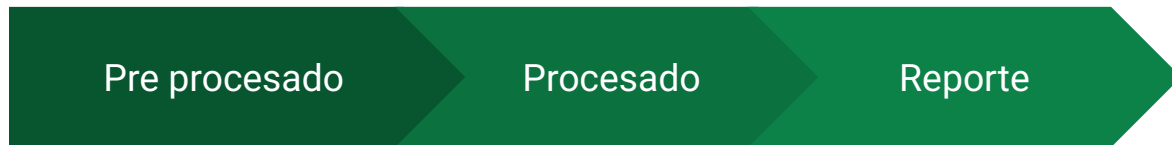


EuroHPC
Joint Undertaking

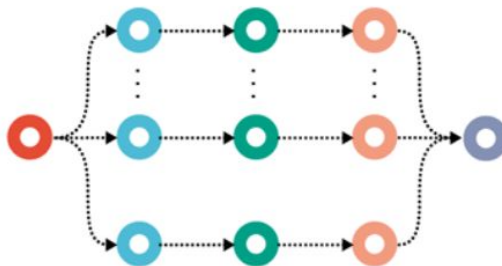


Consorti de
Serveis Universitaris
de Catalunya

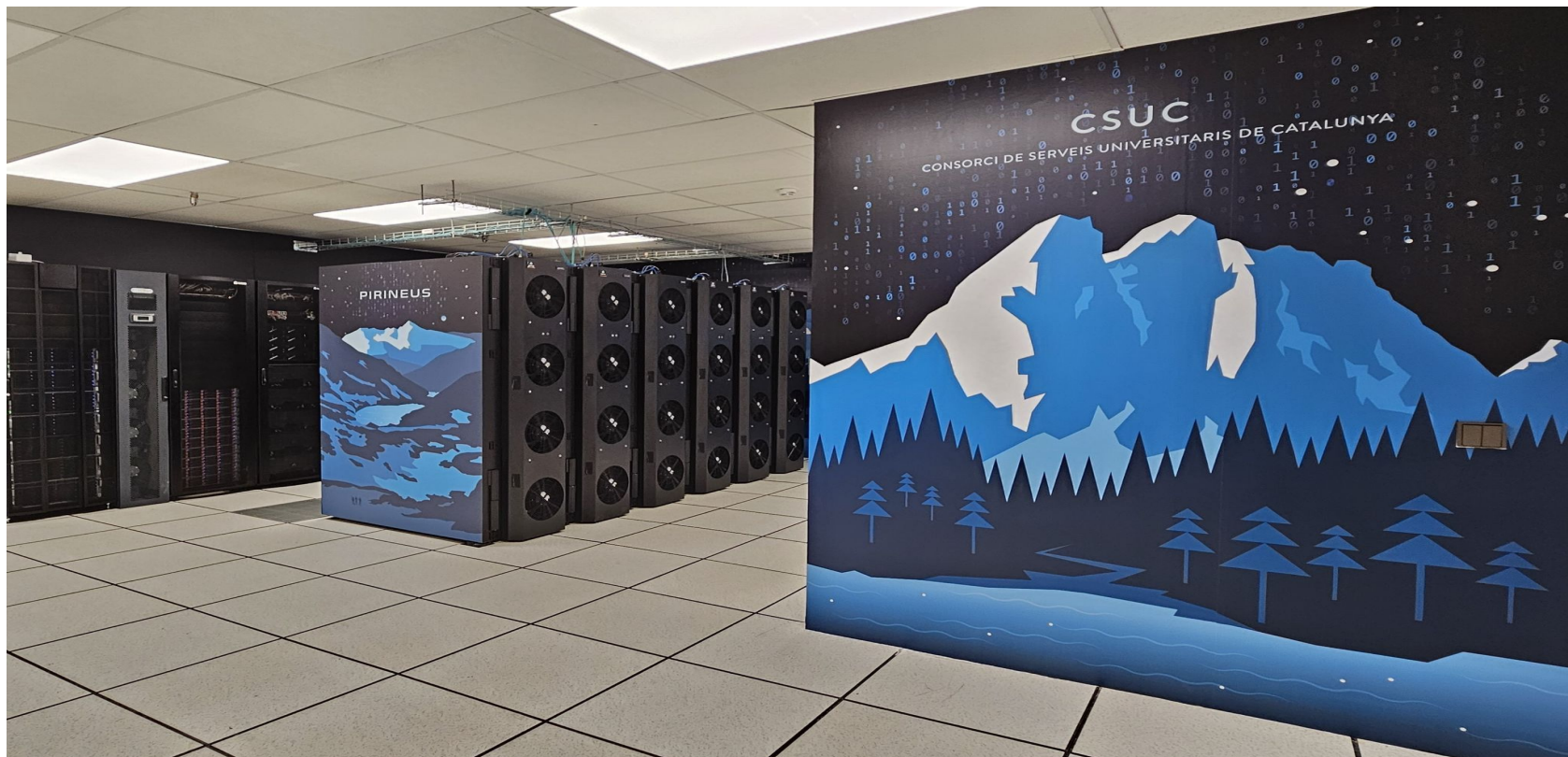




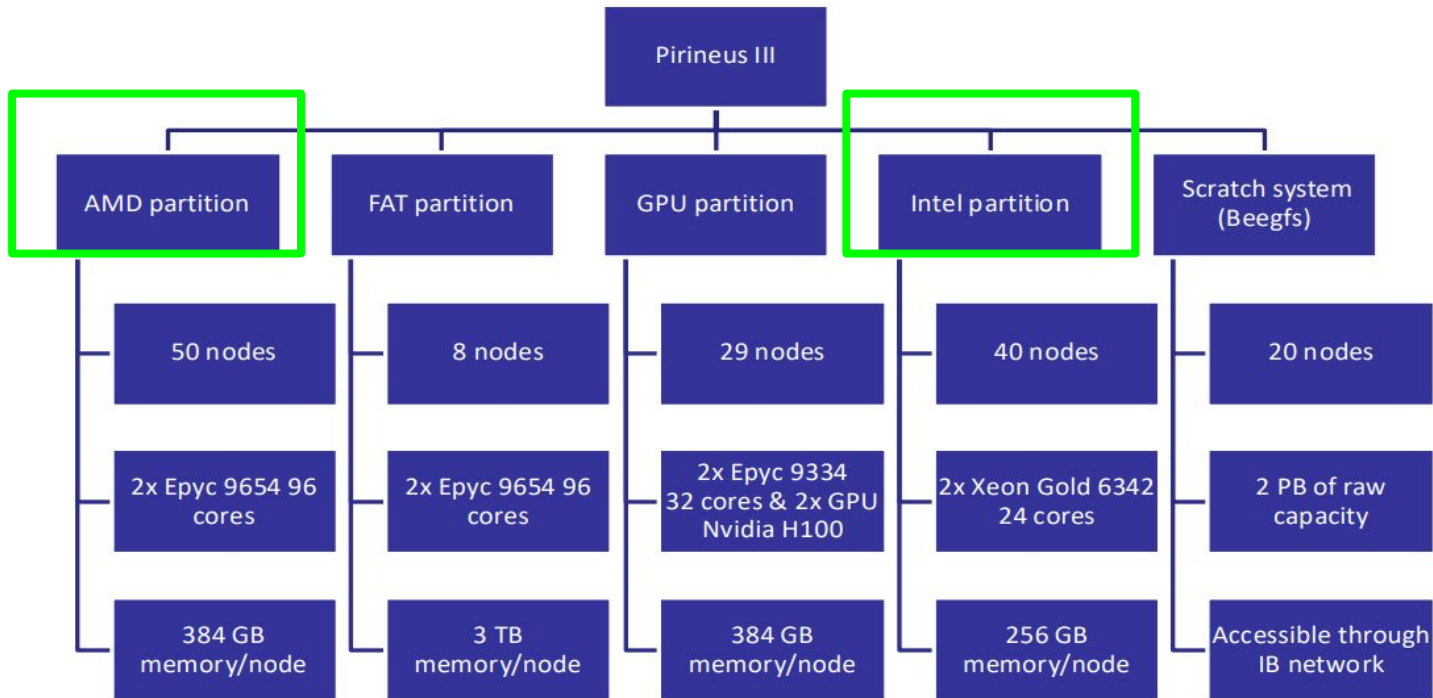
 **nextflow**



Hands On!



Hands On



Hands On

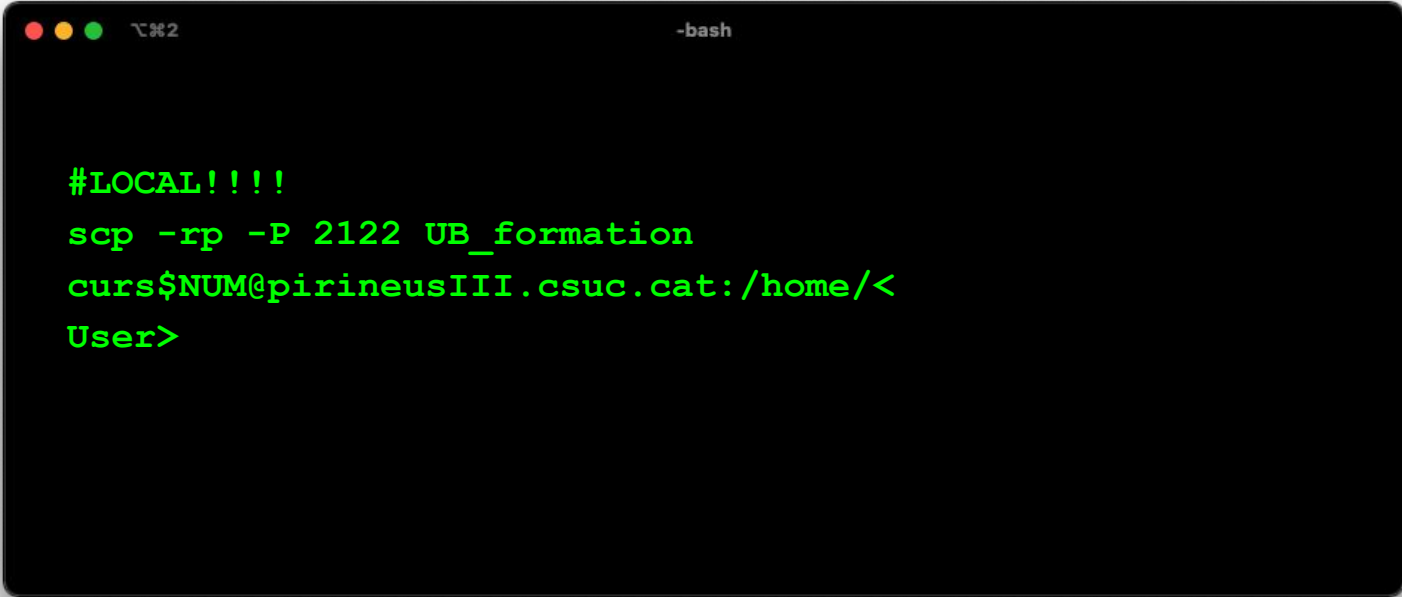
Unit	Path in the sytem	Disk quota	Backup	Observations
Home	/home/\$user	500 GB	Yes (daily)	Not suitable for the execution of jobs
Data	/data/\$group/\$user	2 TB (can be increased according to storage prices)	Not by default (but you can ask for it)	
Scratch	/scratch/\$group/\$user	No limits	No	Only jobs executing under slurm can write in this directory, files are automatically removed after seven days

Hands On

- Clonar repositorio de Github:
https://github.com/HPCNow/UB_formation
- Acceder a Pirineus III

```
ssh <User>@pirineus3.csuc.cat -p 2122
```

- Copiar ficheros a Pirineus III



```
~%2 -bash

#LOCAL!!!!
scp -rp -P 2122 UB_formation
curs$NUM@pirineusIII.csuc.cat:/home/<
User>
```

Hands On

- Explorar Pirineus III
 - particiones de slurm
 - software disponible via modulos
 - verificar que los ficheros copiados estén

Hands On

- Iniciar una sesión interactiva



```
$ salloc --time 4:00:00 -n 1
```

Hands On

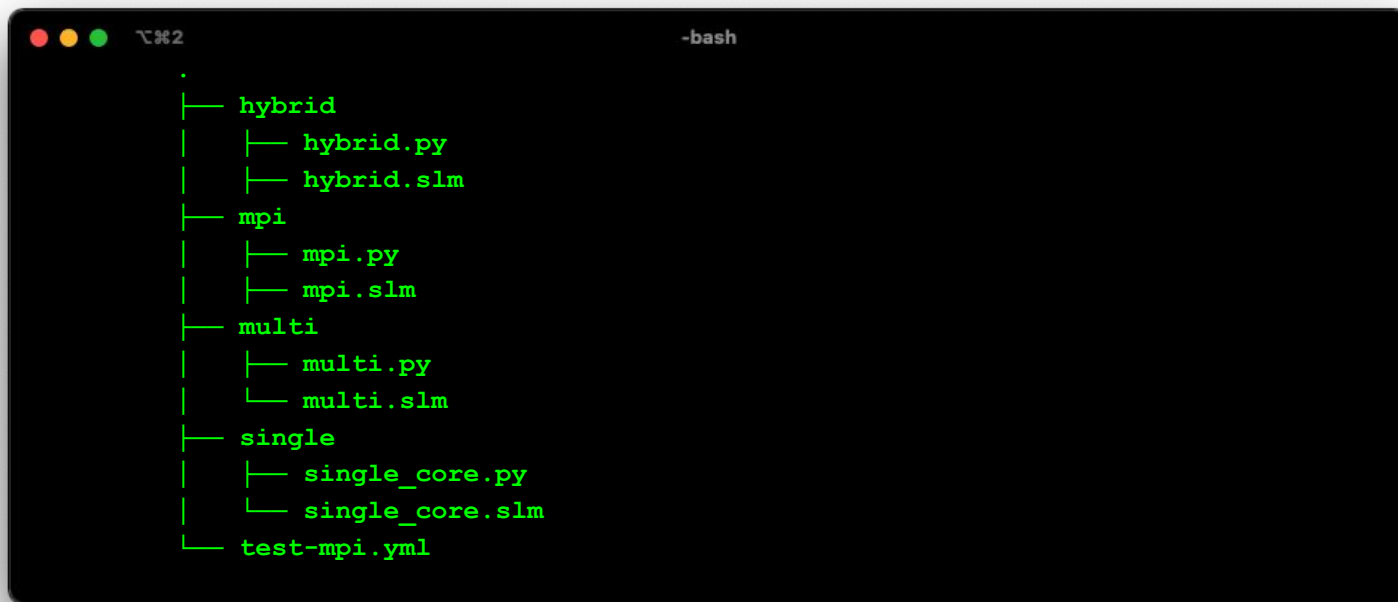
- Preparar entorno de conda



```
$ ml conda  
$ conda env create -f test-mpi.yml -y
```

Hands On

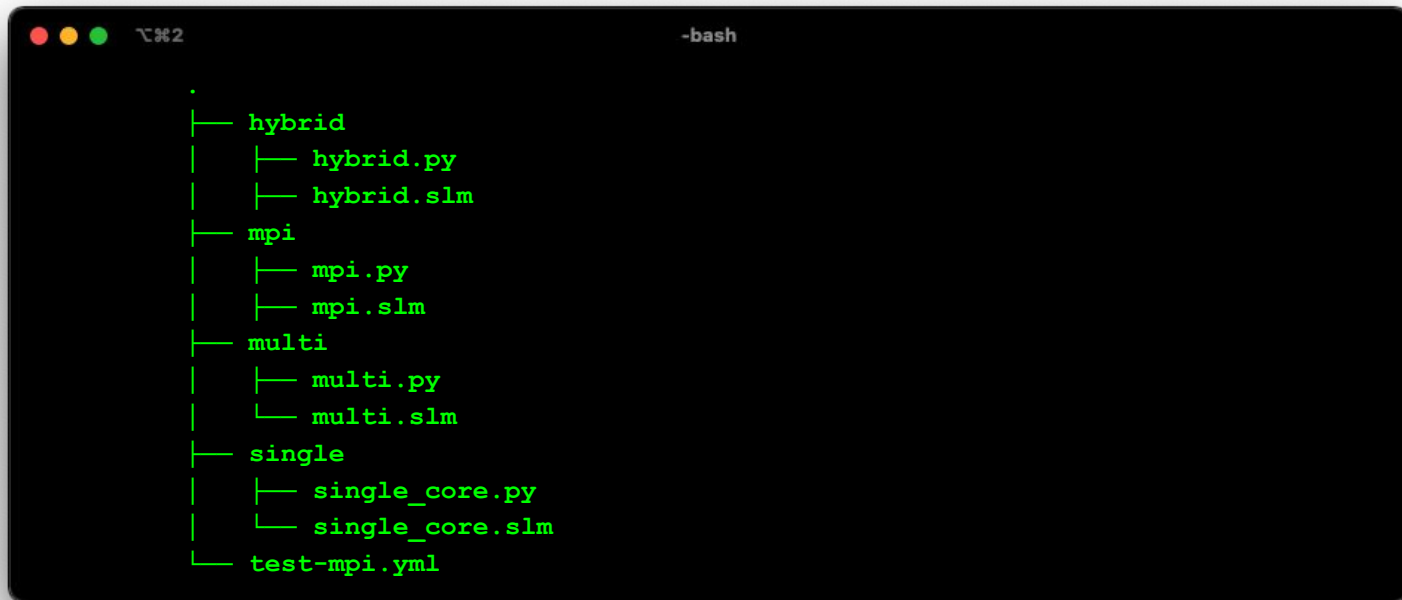
- Scripts de python que:
 - Crea dos arreglos con 1×10^7 elementos
 - Suma los arreglos
 - Suma todos los elementos de la suma y calcula el promedio



```
.  
├── hybrid  
│   ├── hybrid.py  
│   └── hybrid.slm  
├── mpi  
│   ├── mpi.py  
│   └── mpi.slm  
├── multi  
│   ├── multi.py  
│   └── multi.slm  
├── single  
│   ├── single_core.py  
│   └── single_core.slm  
└── test-mpi.yml
```

Hands On

- Lanzar los trabajos
 - Ir a cada carpeta y lanzar los trabajos con **sbatch** **<script>.slm**
 - Revisar los outputs



```
.  
├── hybrid  
│   ├── hybrid.py  
│   └── hybrid.slm  
├── mpi  
│   ├── mpi.py  
│   └── mpi.slm  
├── multi  
│   ├── multi.py  
│   └── multi.slm  
├── single  
│   ├── single_core.py  
│   └── single_core.slm  
└── test-mpi.yml
```

Hands On

- Modificar los scripts de slurm/python cambiando los parámetros del tamaño del arreglo/número de procesos y analizar diferencias

Preguntas