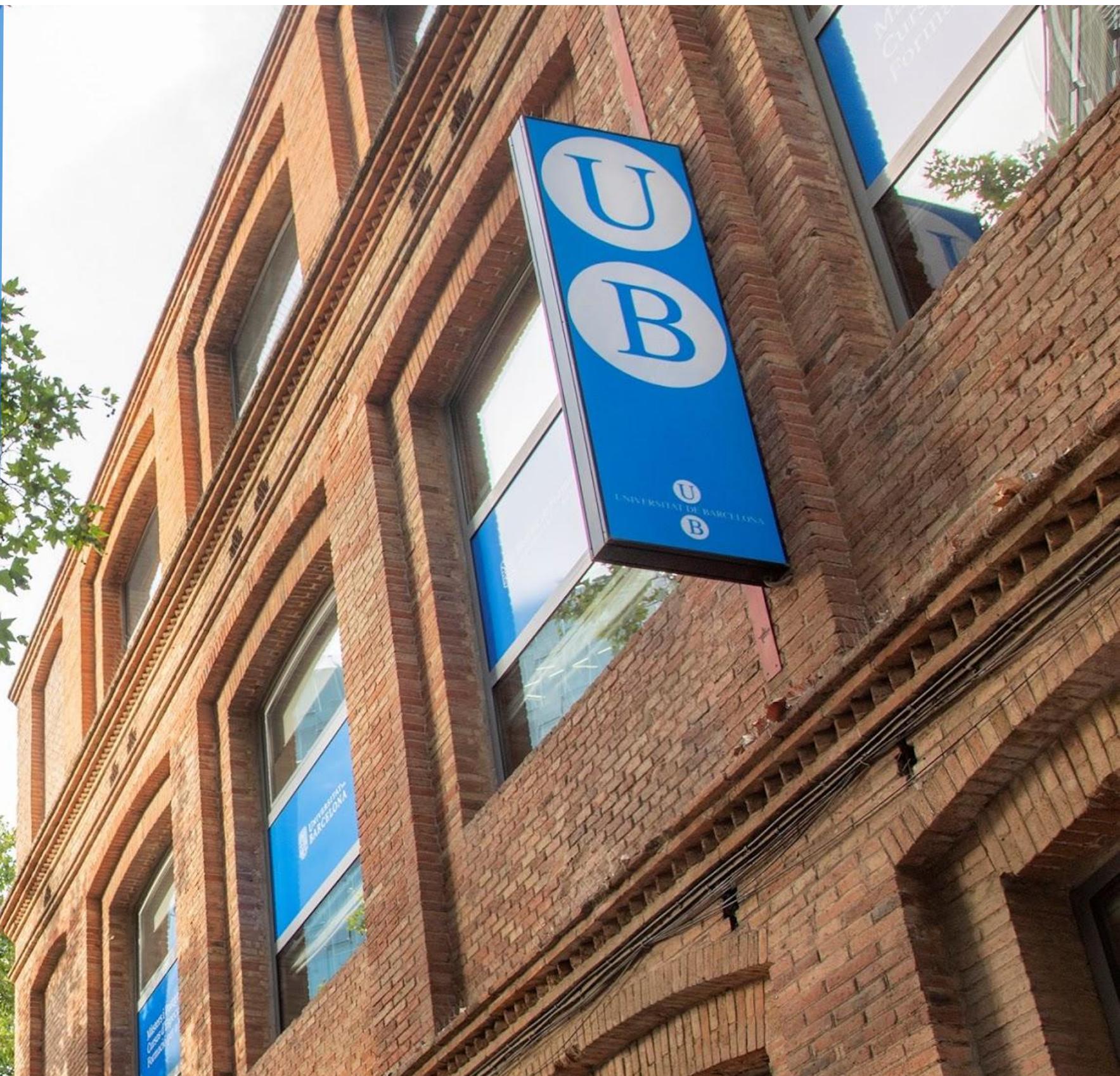


# Introduction and Best Practices for High Performance Computing - HPC

Erica Bianco, PhD  
Computational Scientist @HPCNow!

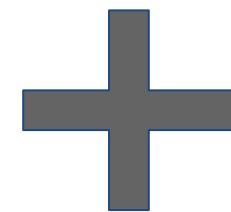


# Contents

1. Quick introduction to the HPC
2. Hands-on @ public HPC

# System Administrators and User Support

CENTRE DE SUPERCOMPUTACIÓ  
DE CATALUNYA



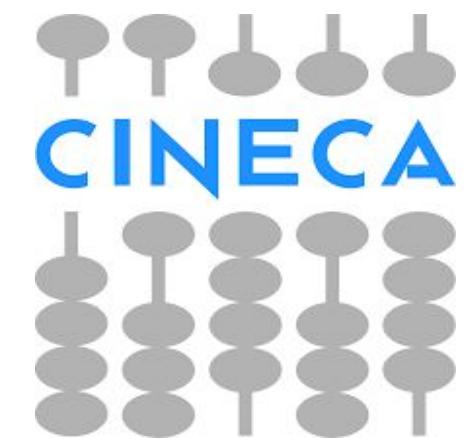
Reference Network on  
Theoretical and Computational Chemistry

# Top500 Supercomputer Users



*Barcelona  
Supercomputing  
Center*

*Centro Nacional de Supercomputación*



- IT + scientific background
- HPC services and solutions
- User-oriented company
- SW & HW agnostic





Consulting



Training



Installation



Support



Administration

**Advanced Supercomputing  
Services for Science  
and Engineering**

# Public sector



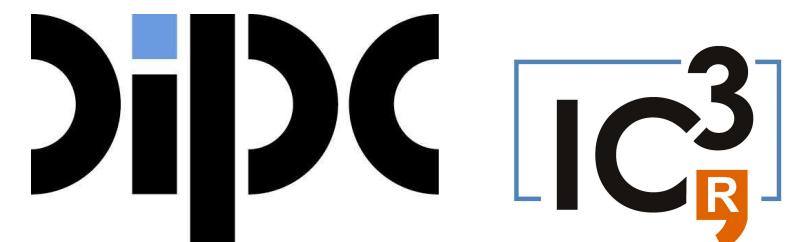
Universitat Autònoma de Barcelona



UNIVERSIDAD AUTONOMA  
DE MADRID



Centro de Supercomputación de Galicia



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea



Universitat  
Pompeu Fabra  
Barcelona



Barcelona  
Supercomputing  
Center

Centro Nacional de Supercomputación



basque center for applied mathematics



CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS



iQAC



Osakidetza



Barcelona  
Biomedical  
Research  
Park



ITA INNOVA  
INSTITUTO TECNOLÓGICO DE ARAGÓN



RSET  
RAJAGIRI SCHOOL OF  
ENGINEERING & TECHNOLOGY



NBD  
INOSTRUM BIODISCOVERY

SIEMENS Gamesa  
RENEWABLE ENERGY

OMEGA  
PERIPHERALS

FotoPRIXTM

# Private Companies

Confidential

Appplus<sup>+</sup>  
IDIADA



renfe

MADE OF  
GENES  
All genomics.  
All yours.

AWS Truepower®

Company overview

# Partners

HW

**Lenovo**

**CRAY**  
THE SUPERCOMPUTER COMPANY

**DELL**

**IBM**

**ASUS**®

**SUPERMICRO**®

SW



**BeeGFS**®  
developed by Fraunhofer

**NICE**

**SchedMD**

**IBM**

**intel**®

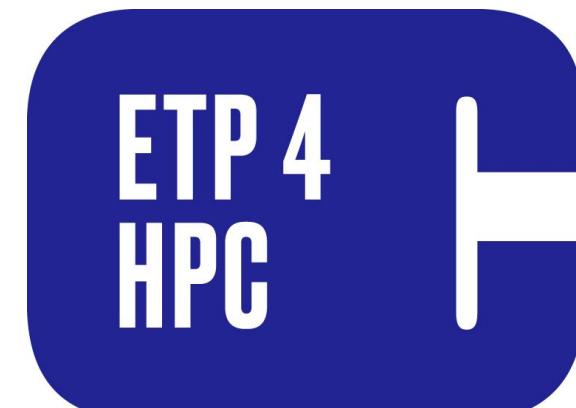
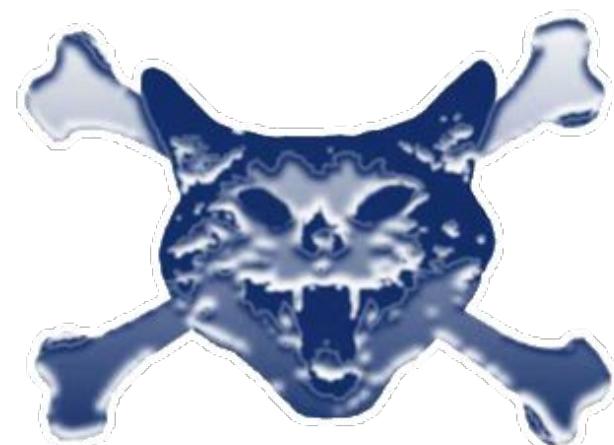
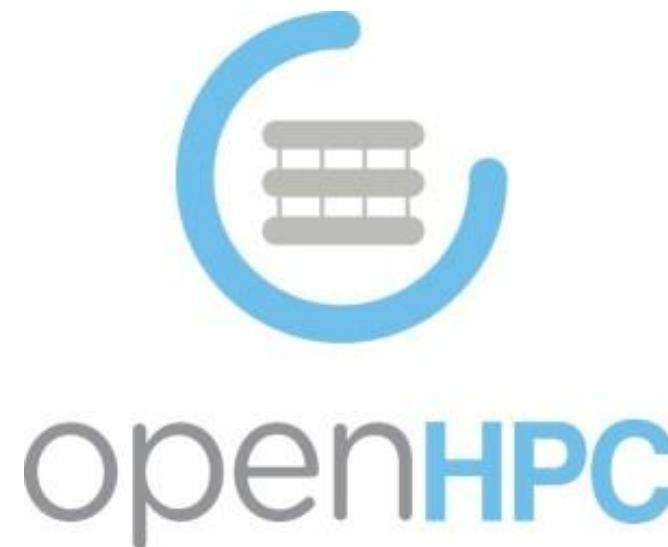


**Bright** Computing

**ARM**®

**UC.**  
**it**  
You See It

# Contributions to HPC Community



EUROPEAN  
TECHNOLOGY  
PLATFORM  
FOR HIGH  
PERFORMANCE  
COMPUTING



## Take home messages

### ★ Basic HPC concepts

- What does HPC means
- Why to use an HPC system and how

### ★ How to submit your first jobs successfully to an HPC system



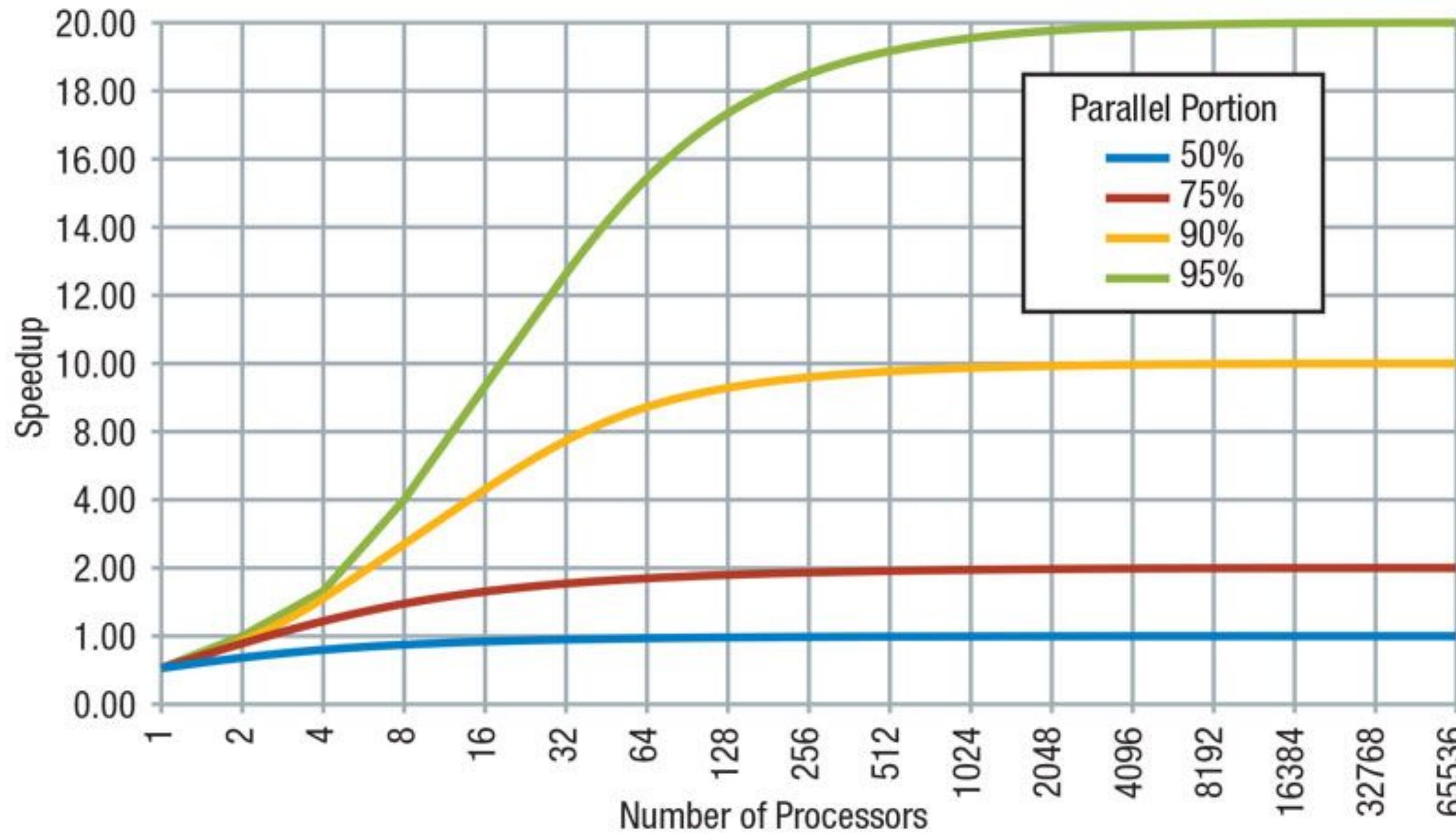
# Contents

- 1. Quick introduction to the HPC**
- 2. Hands-on @ public HPC**

## Why use an HPC?

- The key goal is to solve the problems faster.
- Split the work between several processors
- Parallel computing allows to divide the work among multiple CPUs in several compute nodes.
- Ideally, the program will run  $N$  times faster by using  $N$  processors.
- Unfortunately, in most cases that's not true.

# Amdahl's Law

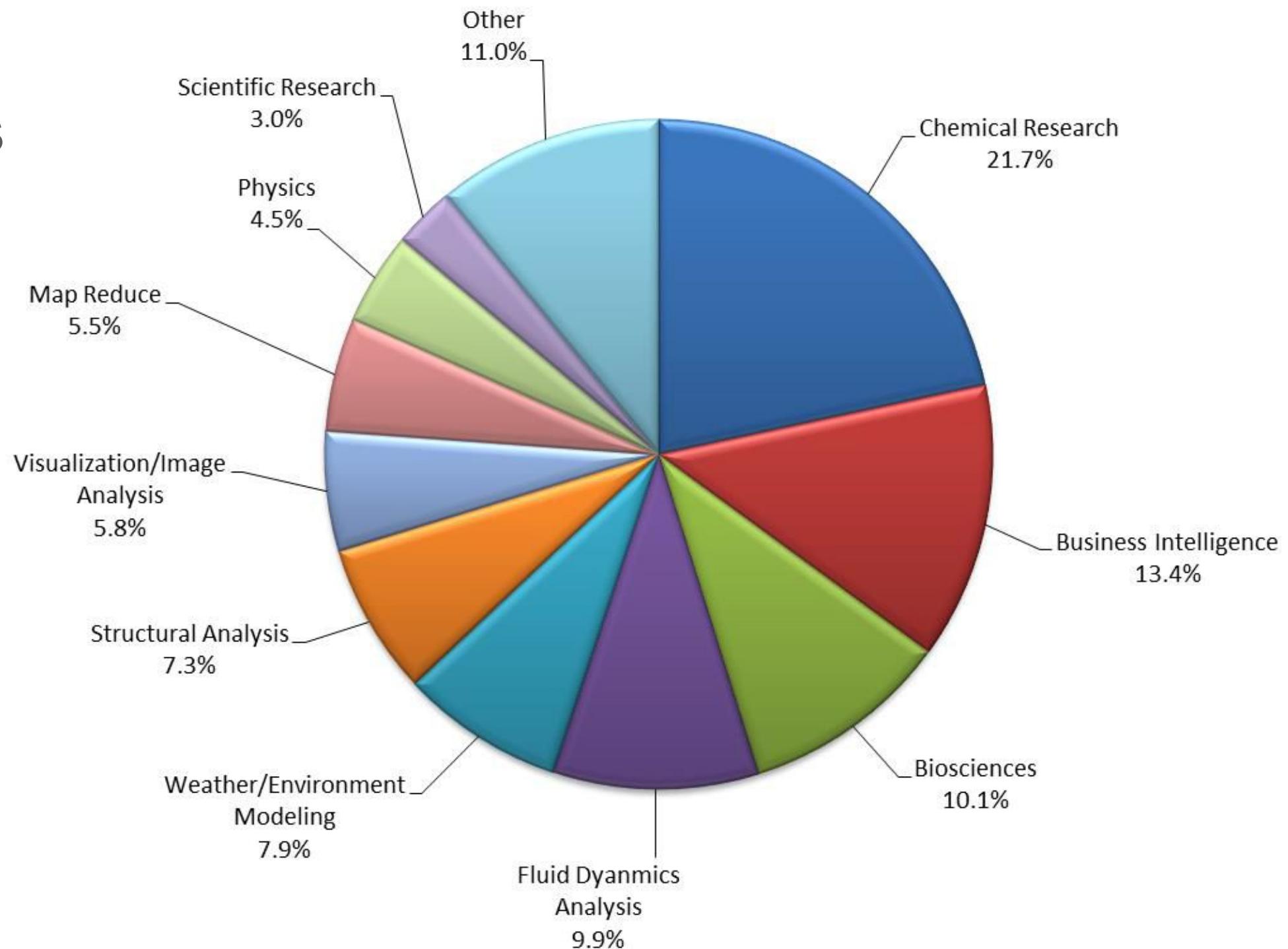


## Why to use HPC?



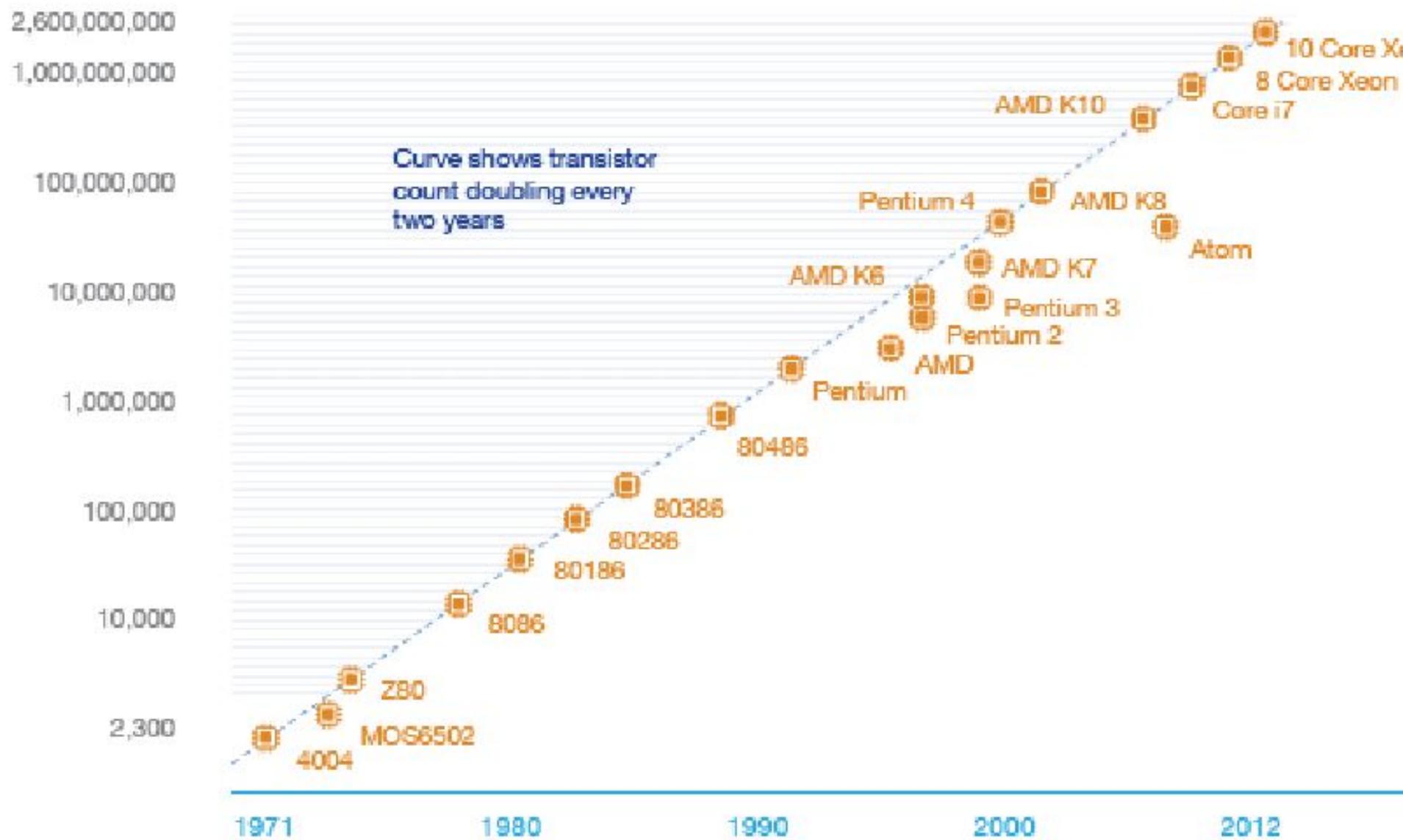
## What HPC used for?

- Scientific and Engineering simulations have a need for massive compute power.
- The clock speed of single-core processors can not be faster due power and heat limitations.
- Memory allocated to a single-core is quite limited. Increasing it is very expensive and slow.



# Moore's law

The number of transistors on a microchip doubles about every two years, though the cost of computers is halved.

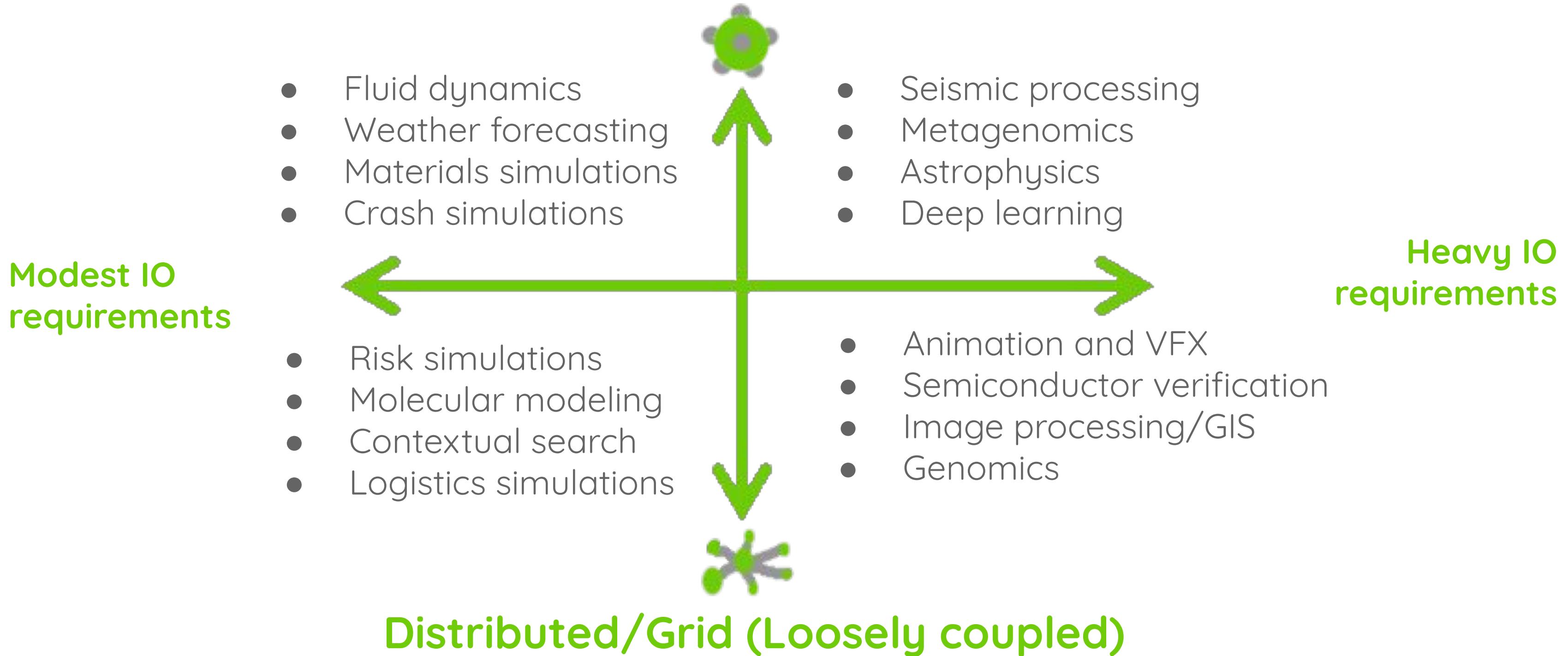


It also applies to:

- processor speeds,
- communication speeds,
- storage space on hard disks,
- pixels on a screen.

# Main Use Cases for HPC

## Clustered (Tightly coupled)



# HPC Suitable Work

## Suitable Work

- When tasks take too long
- When one server is not enough
- When my problem consumes large amounts of memory

## Less Suited Work

- Windows only software
- Interactive software i.e. GUI
- Inefficient, un-optimised software

## HPC Parallel execution time

- Single core computation time: computation only.
- Parallel computation time: computation + communication + waiting.
- E.g. writing results (to one file) is often a bottleneck.
- Small problem on many cores: communication cost will dominate.
- Unbalanced load: one core will mainly wait on the other.

# Performance

The performance of a program depends on:

- Clock speed
- Floating point unit
- Cache size
- Memory latency
- Memory bandwidth
- IO
- Fabric interconnect

$$\text{FLOPS} = \text{sockets} \times \frac{\text{cores}}{\text{socket}} \times \text{clock} \times \frac{\text{FLOPs}}{\text{cycle}}$$

# Paradigms in Parallel Programming

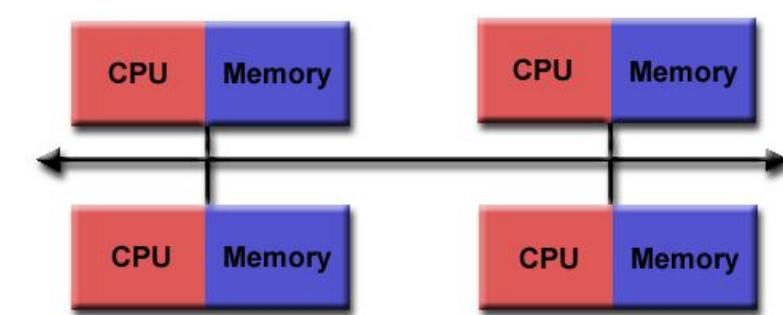
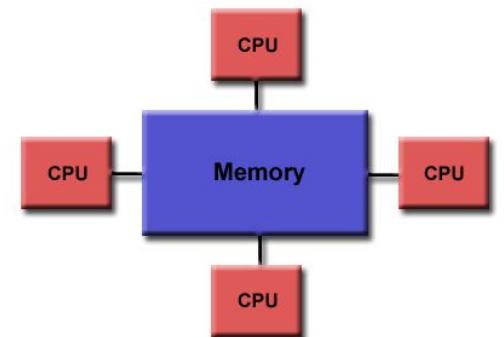
The two parallel programming paradigms most used in HPC are:

- **OpenMP** for shared memory systems (multithreading): on multiple cores of a single node
- **MPI** for distributed memory systems (multiprocessing): on multiple nodes

Parallel programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronisation between the different subtasks are typically some of the greatest obstacles to getting good parallel program performance.

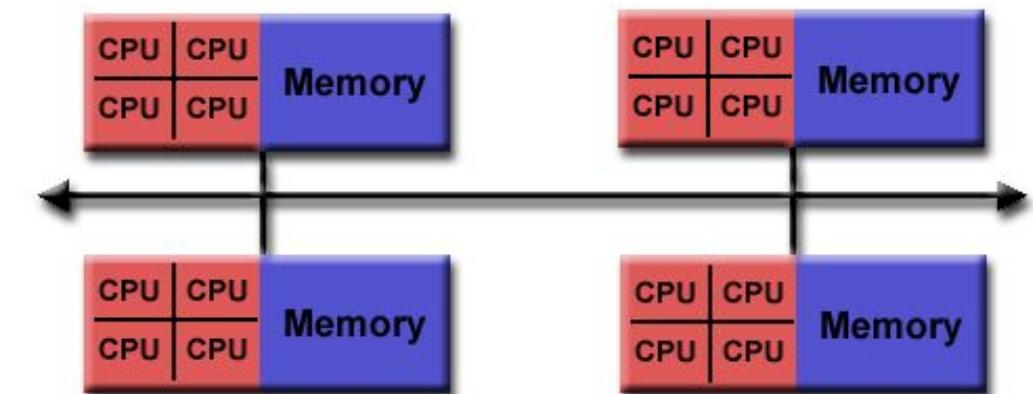
# Architecture

**Shared Memory:** Memory can be simultaneously accessed by multiple programs or threads with an intent to provide communication among them or avoid redundant copies. Shared memory is an efficient means of passing data between programs. **OpenMP**

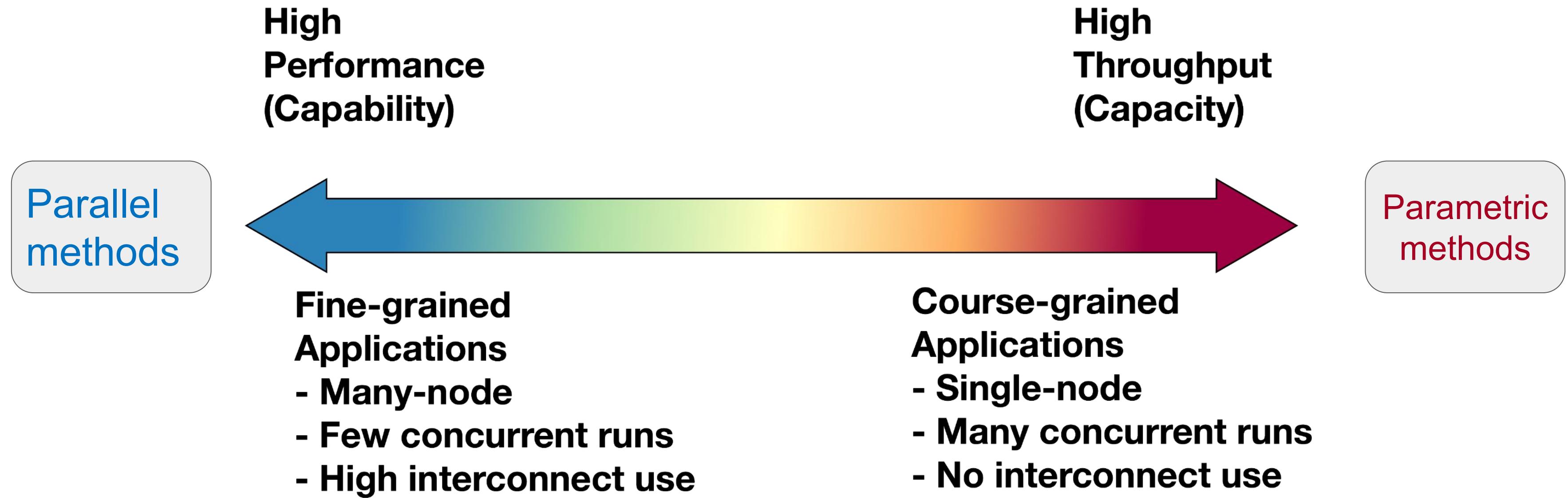


**Distributed memory:** Defined as a multiprocessor computer system in which each processor has its own private memory. Computational tasks can only operate on local data, and if remote data is required, the computational task must communicate with one or more remote processors through the network. **MPI**

**Hybrid:** Nowadays, the clusters are composed of multiple compute nodes, where each node has access to a large shared memory in addition to each node's limited non-shared private memory. **OpenMP and MPI**

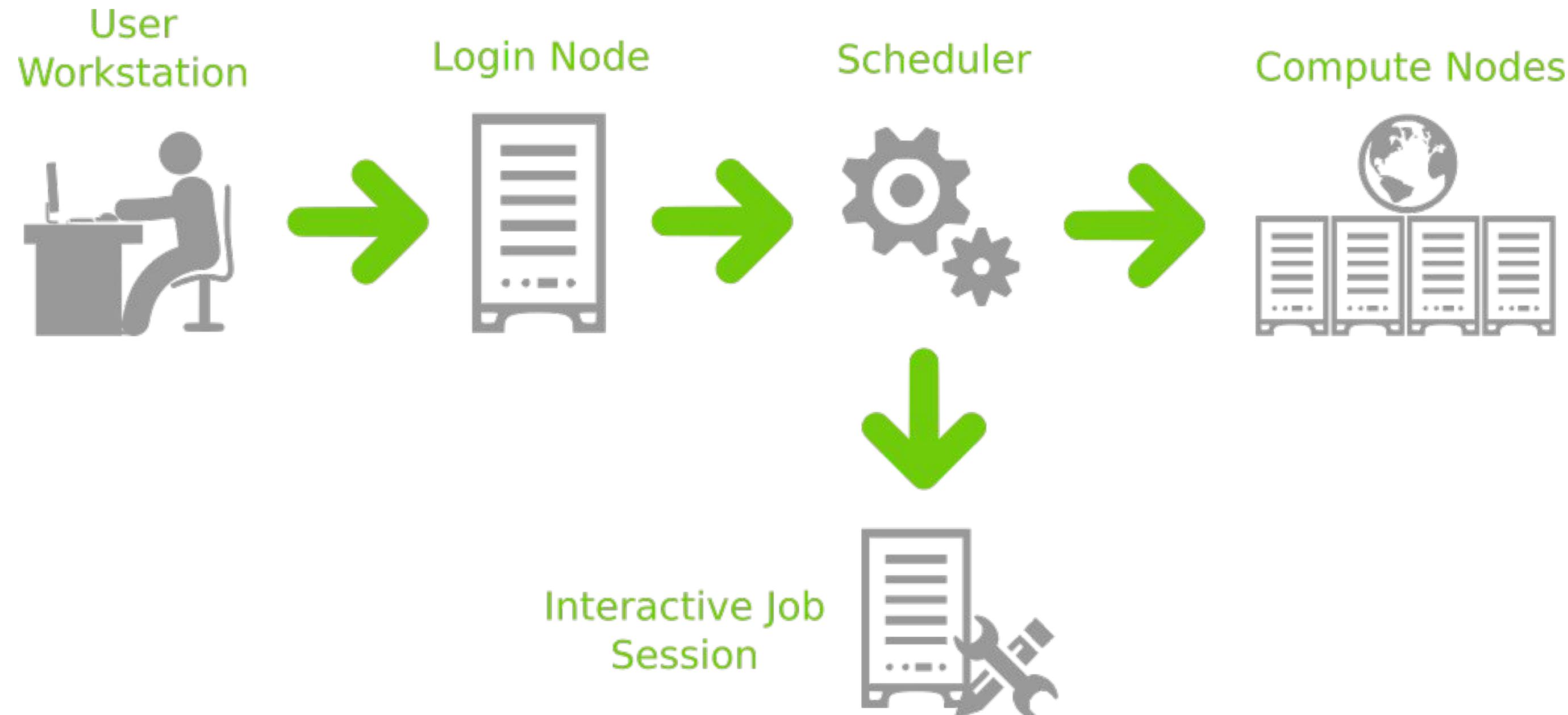


# HPC or HTC?



# User Workflow

On a HPC system you send your script to the queue to be run and use the computational resources



# Difference between Desktop Computer and HPC

- No Graphical User Interface (GUI)
- Programs are executed in batch mode
- Resources are shared between multiple users

- Workstation
- Local Server

Local

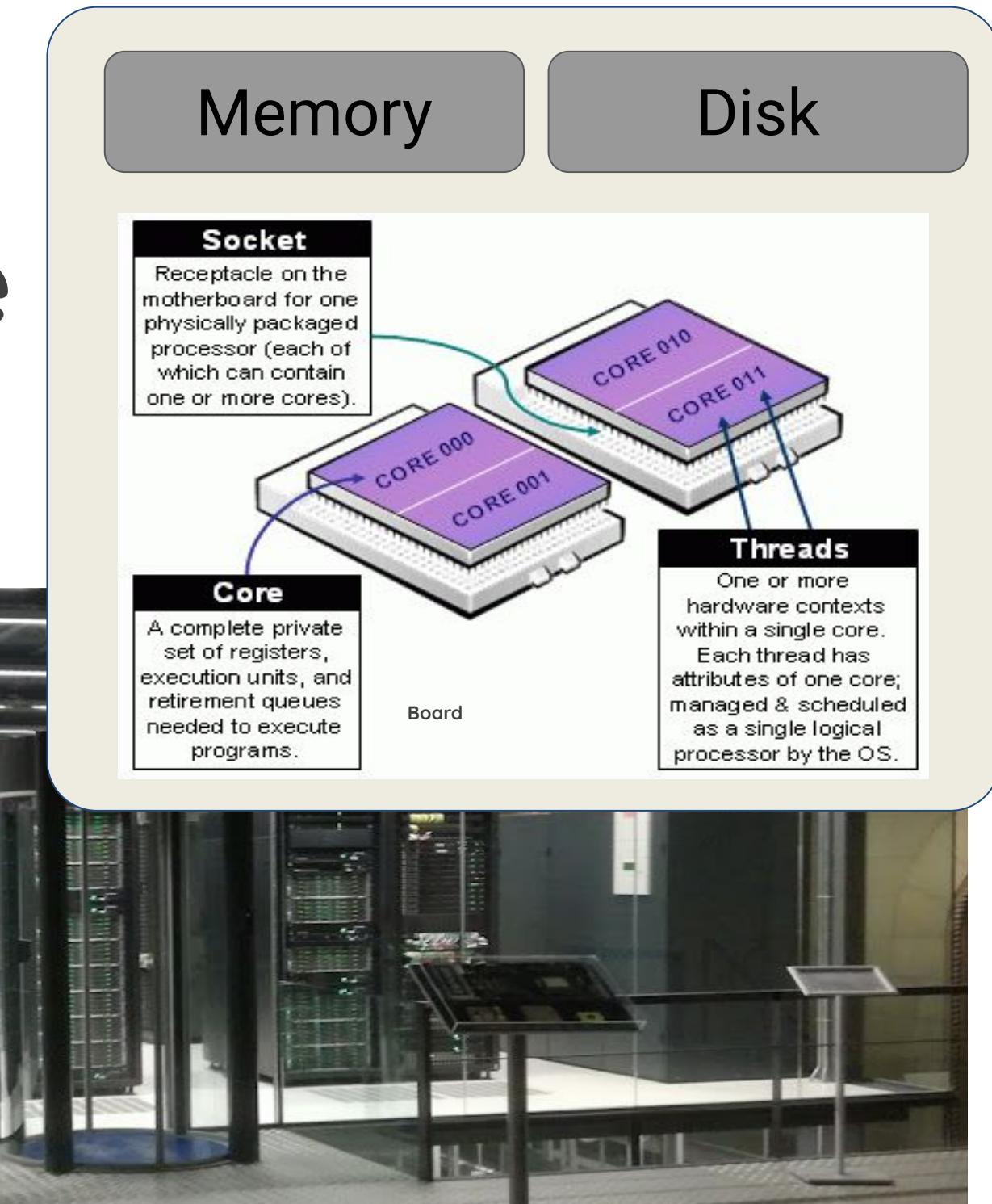
- Cloud HPC
- HPC Provider Company
- HPC Public Center

Remote

## HPC schema

Rack

Node



Data Center

# Storage: HPC Filesystems

/share

/scratch

297 Tb

/project

517 Tb

/home

8 Tb

No quota  
(but we are a lot of users)

short term storage  
**temporary and working files**  
faster I/O  
share files between users

No quota  
(but we are a lot of users)

long term storage (slow I/O)  
**important files storage**  
share files between users of the  
same lab

individual quota: 50 Gb

to store scripts and small files  
symbolic links for project and  
scratch

# Software Applications on an HPC

HPC have shared appstacks and local installed applications.

EESSI is an example of a shared appstack across multiple centers



# Environment Modules

## Configuring Environment:

- Environment Modules package allows for dynamic modification of the user environment
- Modulefiles contain the required information to configure the shell for an application software
  - Loaded / unloaded dynamically
  - Useful in managing different versions of applications



# Lmod cheat-sheet

```
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 to
the environment
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
```

## Load a module:

```
[ hpcnow1 ] pirineus7:~ $ which R
/usr/bin/which: no R in (/opt/gold/bin:/...:/usr/sbin)
[ hpcnow1 ] pirineus7:~ $ module load R
[ hpcnow1 ] pirineus7:~ $ which R
/prod/apps/R/4.0.2/bin/R
[ hpcnow1 ] pirineus7:~ $ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

.
.
.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

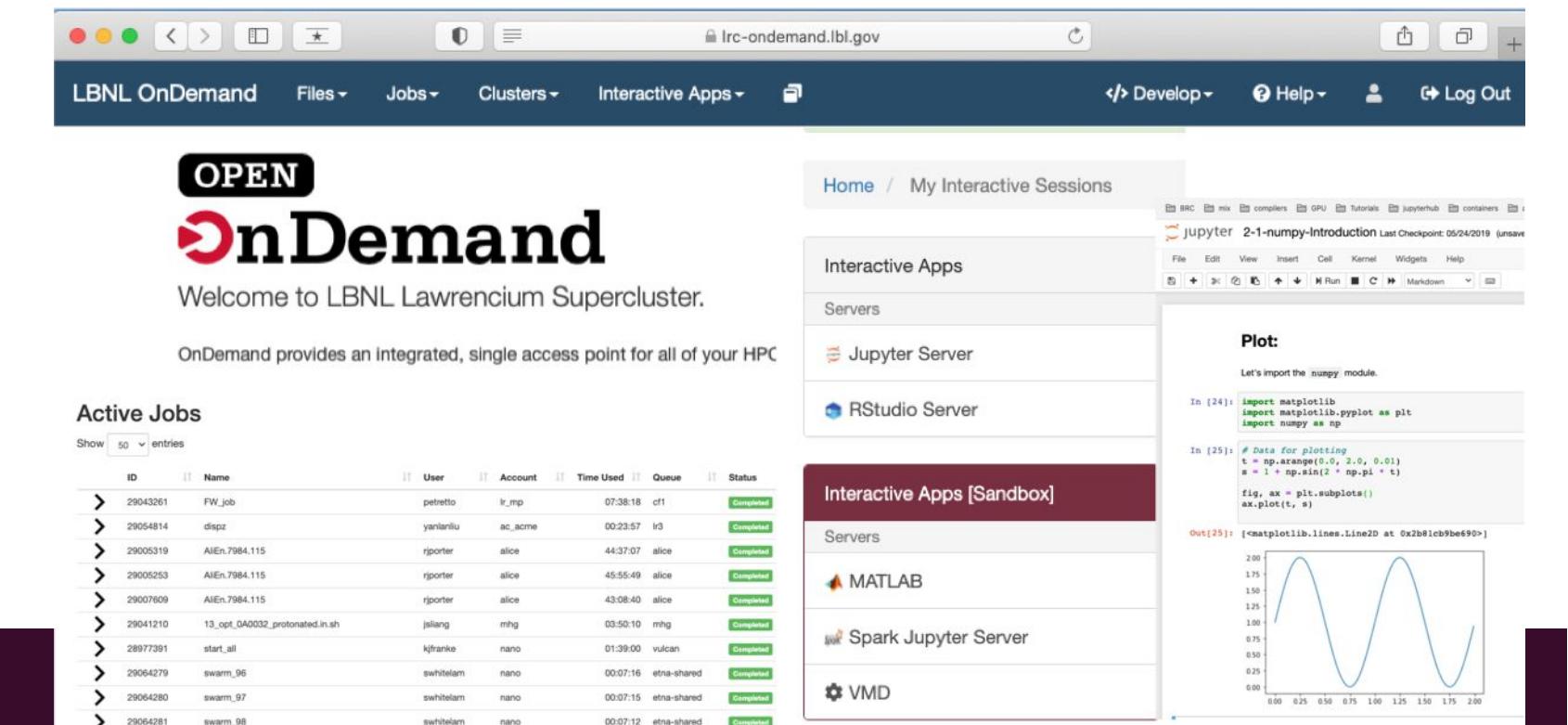
>
```

# Accessing resources: SSH connection

- Linux - use a terminal
- Windows - use a terminal emulator
  - Putty
  - MobaXterm
  - windows powershell
  - WSL (Windows Subsystem Linux)
  - Virtual Machine

```
ssh $USER@$IP -p $PORT
```

- webportal



The screenshot shows the LBNL OnDemand web interface. At the top, there's a navigation bar with links for LBNL OnDemand, Files, Jobs, Clusters, Interactive Apps, and a user icon. Below the header, the main content area has a banner for "OPEN OnDemand" and a welcome message: "Welcome to LBNL Lawrencium Supercluster. OnDemand provides an integrated, single access point for all of your HPC".

On the left, there's a sidebar titled "Interactive Apps" which lists "Jupyter Server" and "RStudio Server".

The central part of the screen displays "Active Jobs" in a table:

ID	Name	User	Account	Time Used	Queue	Status
29043261	FW_job	petretto	lr_mp	07:38:18	cfl	Completed
29054814	dispt	yanliu	ac_acme	00:23:57	lr3	Completed
29005319	AIEn.7984.115	rporter	alice	44:37:07	alice	Completed
29005253	AIEn.7984.115	rporter	alice	45:55:49	alice	Completed
29007609	AIEn.7984.115	rporter	alice	43:08:40	alice	Completed
29041216	13_opt_0A0032_protonated.in.sh	jislang	mhg	03:50:10	mhg	Completed
28977391	start_all	kfrankie	nano	01:39:00	vulcan	Completed
29064279	swarm_96	swhitelam	nano	00:07:16	etna-shared	Completed
29064280	swarm_97	swhitelam	nano	00:07:15	etna-shared	Completed
29064281	swarm_98	swhitelam	nano	00:07:12	etna-shared	Completed

To the right, there's a section for "Interactive Apps [Sandbox]" which includes "MATLAB", "Spark Jupyter Server", and "VMD". A Jupyter notebook cell is shown with Python code to import numpy and matplotlib, and a resulting sine wave plot.

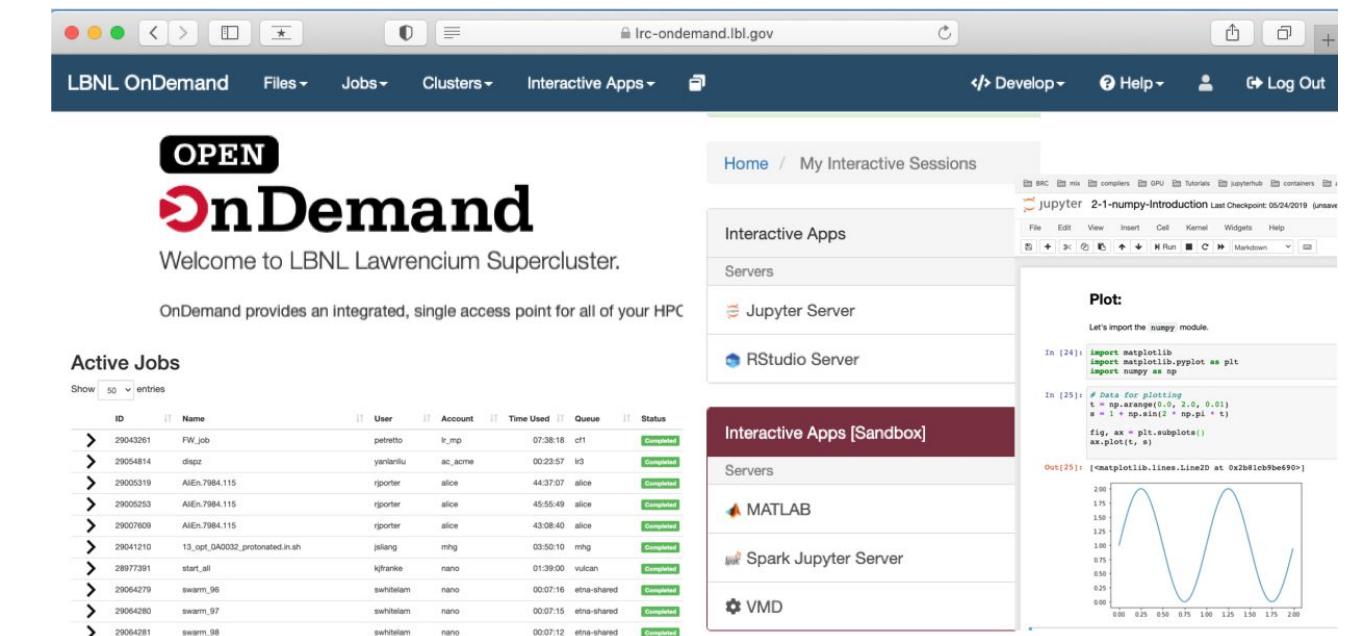
# Accessing resources: move files from/to remote clients

## ● GUI (Graphical User Interfaces):

- WinSCP
- MobaXterm
- FileZilla
- smb mounts, if available
- webportal

## ● CLI (Command Line Interfaces):

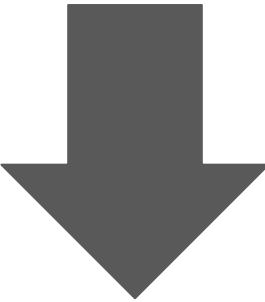
```
scp -oPort $PORT /folder/XX $USER@$IP:/folder/YY  
## or  
sftp -P $PORT $USER@$IP
```



# Compiling

## Running interactive jobs

- **never** compile, work or execute jobs on the login node
- **never** ssh directly into computing nodes



USE INTERACTIVE COMMAND  
or  
LAUNCH THE JOB TO THE QUEUE

## Login nodes dos and don'ts

The login nodes are NOT used for:

- running batch jobs
- performing IO intensive operations
- pre and post processing simulations and/or analysis
- visualize data
- performing memory intensive operations

The login nodes are used for:

- access to the cluster
- access to your home and project directories
- upload your data and/or the code
- download the results
- submit and manage batch jobs
- initiate an interactive job session

# Workload Schedulers

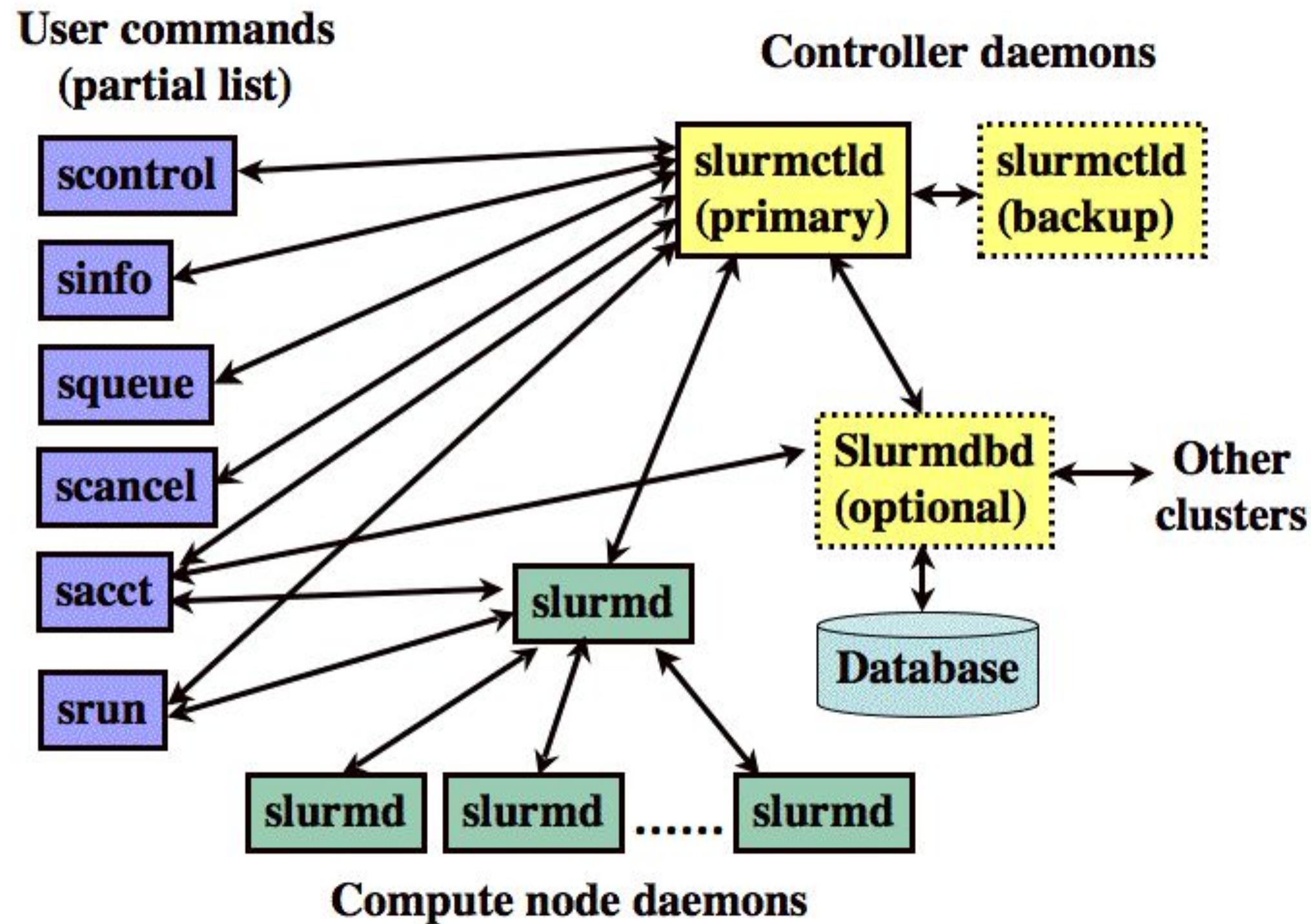
- LSF(IBM Spectrum) --- OpenLAVA
- SGE
- PBS Pro (Altair)
- HTCondor
- Slurm

# Slurm as a workload manager

Slurm is an **open-source workload manager** designed for Linux clusters of all sizes. It provides three key functions. First it **allocates exclusive and/or non-exclusive access to resources** (computer nodes) to users for some duration of time so they can perform work. Second, it provides a framework for starting, executing, and monitoring work (typically a parallel job) on a set of allocated nodes. Finally, it arbitrates contention for resources by managing a queue of pending work.

Slurm's design is very modular with dozens of optional plugins. In its simplest configuration, it can be installed and configured in a couple of minutes (see Caos NSA and Perceus: All-in-one Cluster Software Stack by Jeffrey B. Layton). More complex configurations can satisfy the job scheduling needs of world-class computer centers and rely upon a MySQL database for archiving accounting records, managing resource limits by user or bank account, or supporting sophisticated job prioritization algorithms

<https://slurm.schedmd.com/sbatch.html>



# Slurm cheat-sheet

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
sstat	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 job

# Partitions examples

	Time(H)	Priority	Job Preemption
interactive	9	High	none
gui	-	Urgent	none
gui-extra	-	10	none
short	0,5	100	none
medium	48	50	none
large	120	25	Job suspension
extra	-	10	Job suspension

# Slurm submit script header example

“sbatch script.slurm” to send the job

```
#!/bin/bash
#SBATCH -J GPU_JOB
#SBATCH --time=01:00:00 # Walltime
#SBATCH -A hpcnow # Project Account
#SBATCH --ntasks=4 # number of tasks
#SBATCH --ntasks-per-node=2 # number of tasks per node
#SBATCH --mem-per-cpu=8132 # memory/core (in MB)
#SBATCH --cpus-per-task=4 # 4 OpenMP Threads
#SBATCH --gres=gpu:2 # GPUs per node
#SBATCH -C kepler ml GROMACS/4.6.5-goolfc-2.6.10-hybrid
srun mdrun_mpi -----
```

## Enter the interactive queue or submit the job to the queue

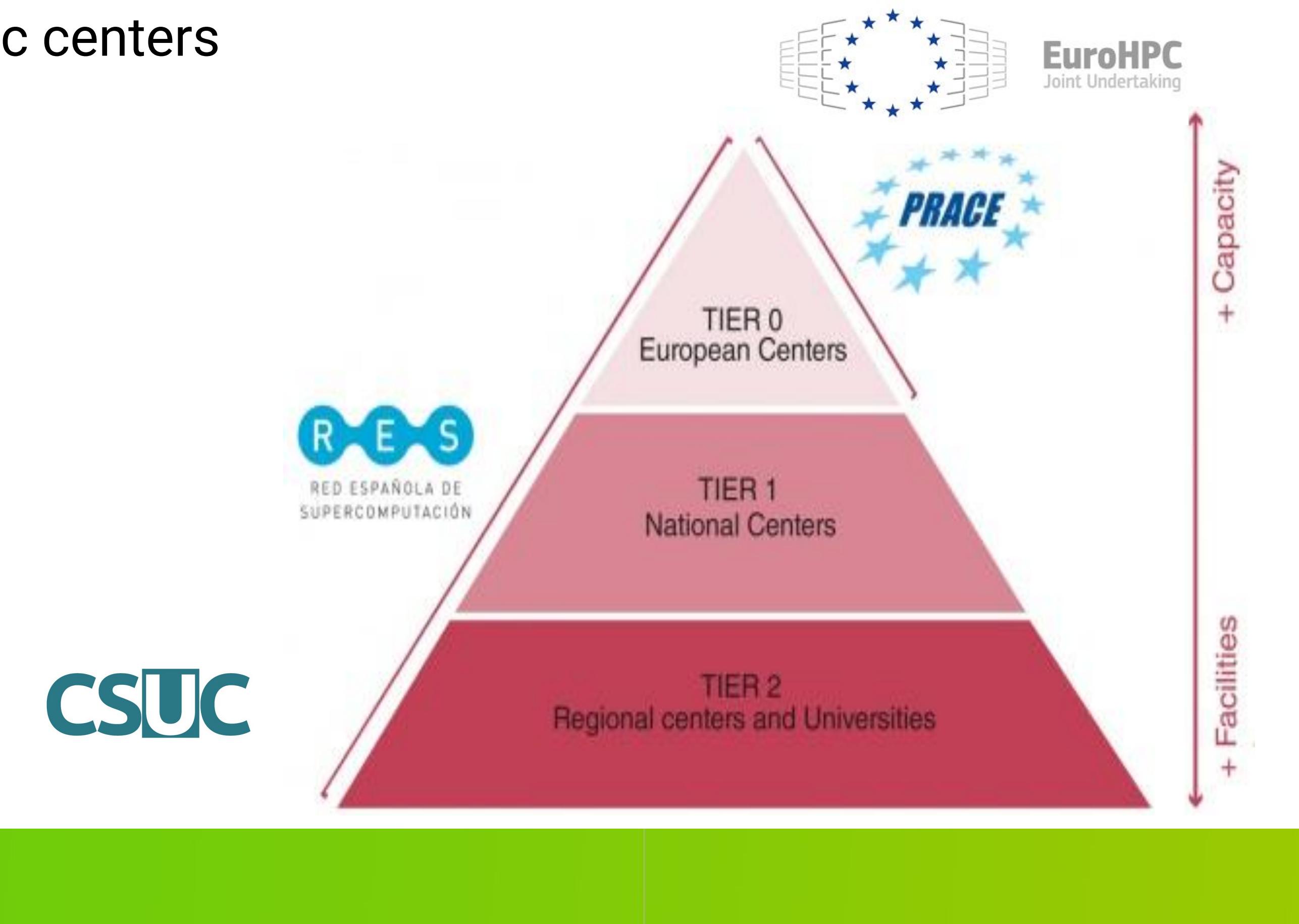
Interactive queue:

```
salloc --time 1:00:00 srun --pty bash
```

Submit to the queue:

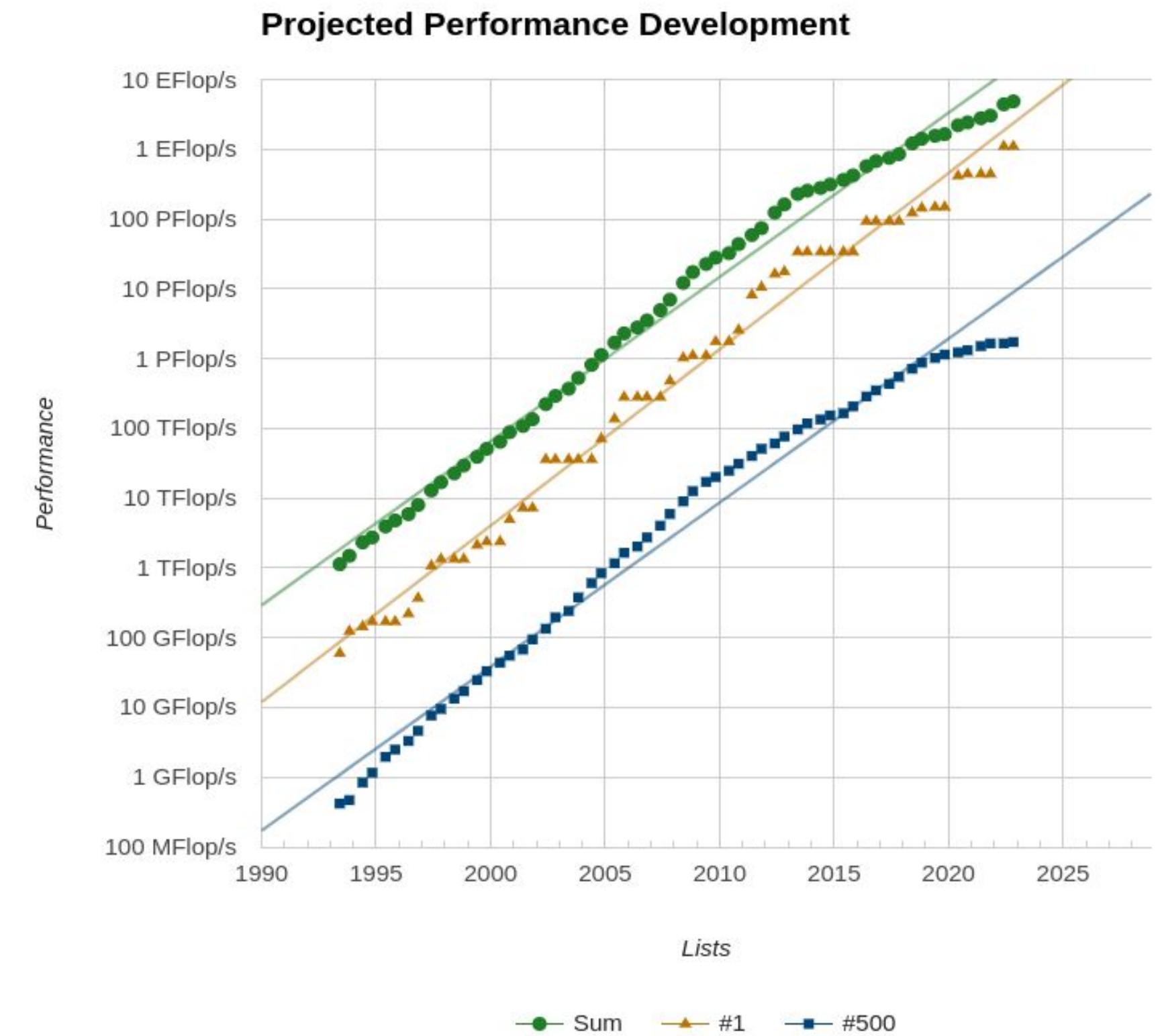
```
sbatch slurm-parallel-example.slm
```

# HPC Public centers



# TOP 500: Linpack benchmark

[www.top500.org](http://www.top500.org)



## HPC Public Center: [CSUC](#)

- SSH Connection
- Different architectures (SMP, DMP)
- MPI and OpenMP
- Software is already installed
- Slurm Scheduler
- Pay-per-use access (When the job is running)

**CSUC**

Càcul Científic

Comunicacions

Portals i Repositoris

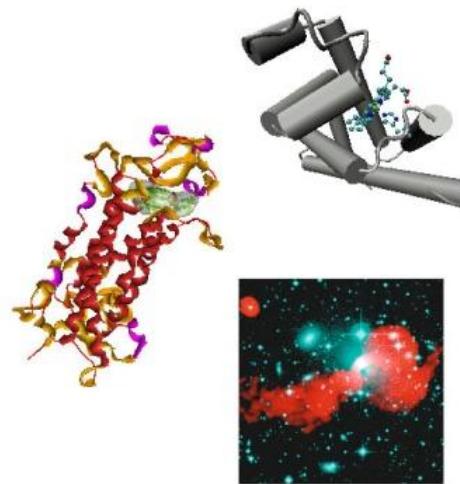
e-Administració

Promoció

# CSUC

CAP

SDF



ANELLA  
CIENTÍFICA

 Red IRIS  
Node Catalunya

 CATNIX

TDX

RACO

mdx

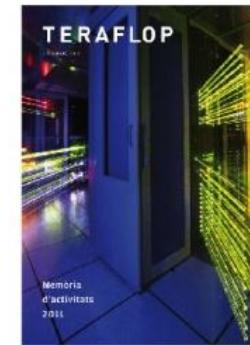
CALAIX

RECERCAT

 RECYT

padicat

Arxiu  
Certificació  
Logs  
Registre  
Vot



JOCS

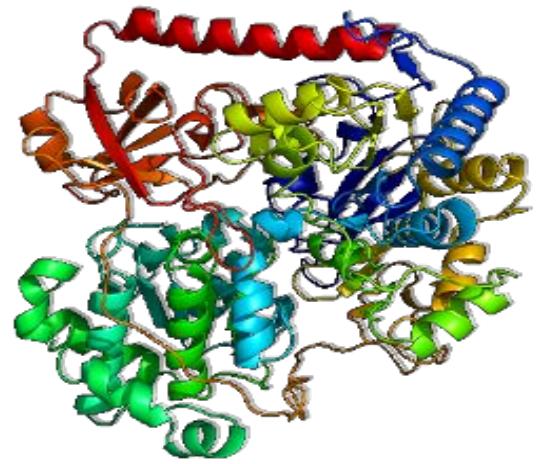
 TAC

 TSIUC

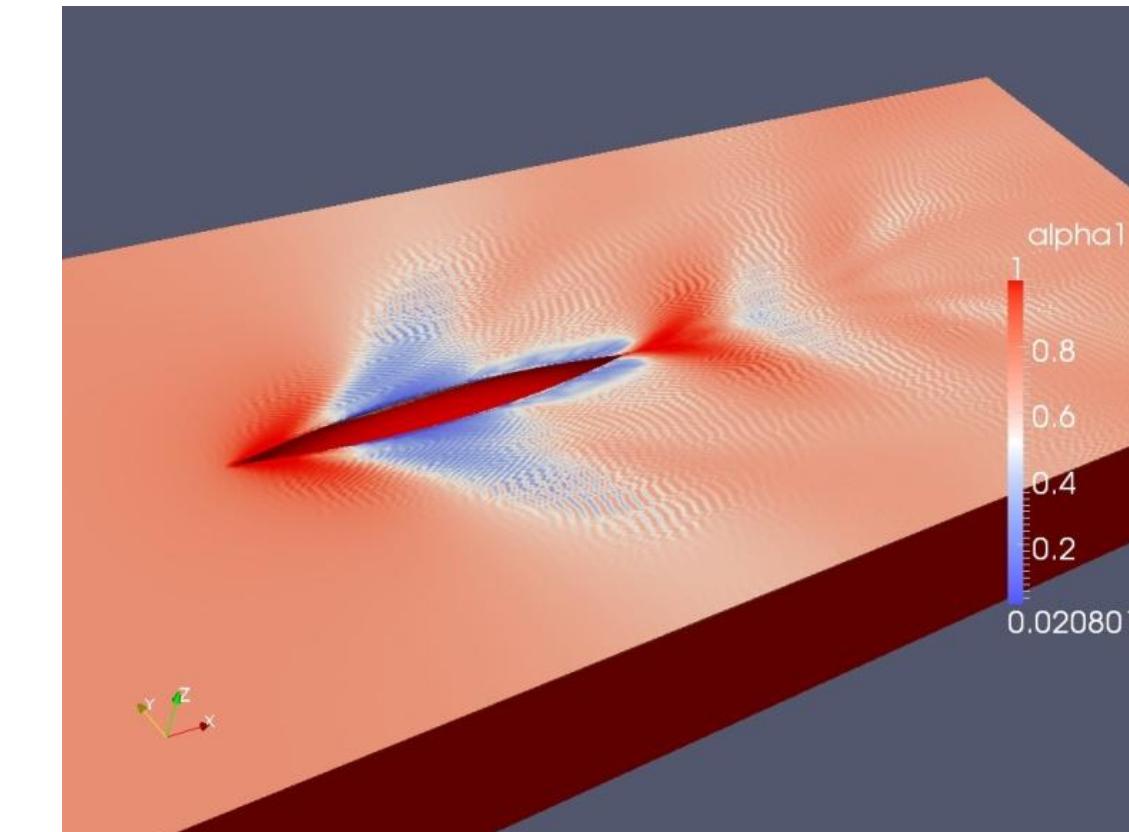
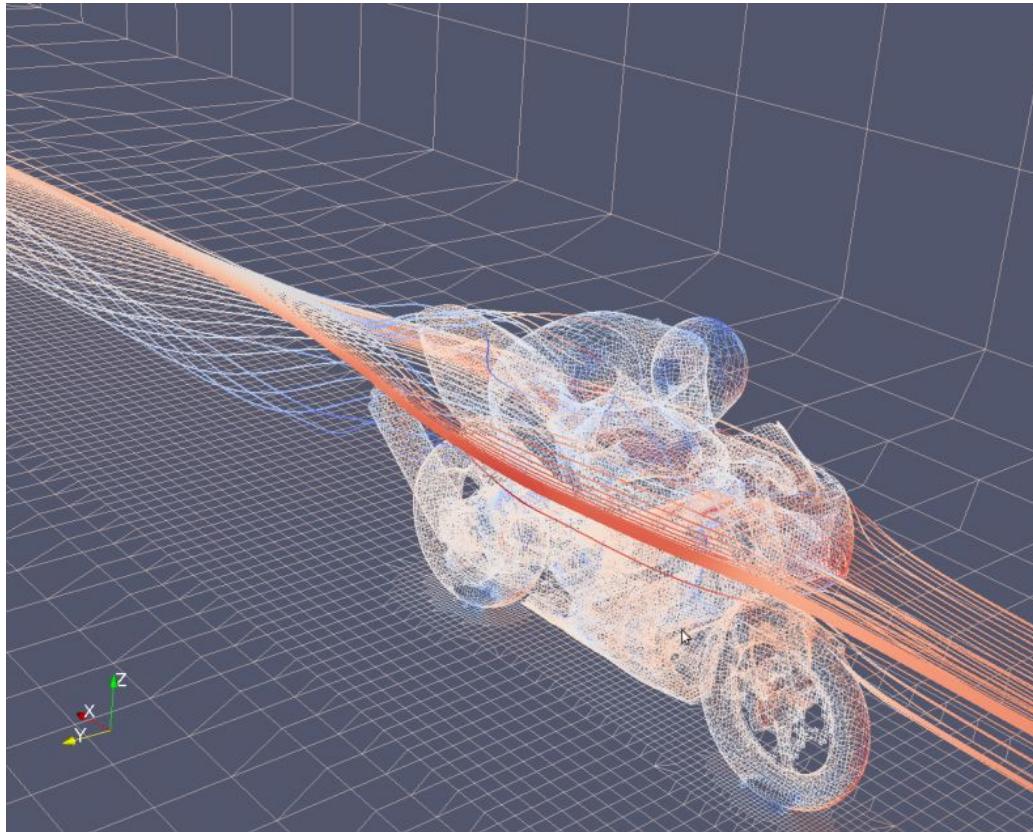
## Operacions i Seguretat



CSIRT  
EC-UR  
ER-CESCA  
SAH  
SED  
S24x7



- ✓ Proveeix de Càlcul d'Altes Prestacions (CAP) a la comunitat científica i industrial.
- ✓ Gestionar el Servei de Disseny de Fàrmacs (SDF), que ofereix eines per modelitzar molècules d'interès biològic.



## Maquinari d'Emmagatzematge

**Collserola (2014)**

- **10 nodes**
  - Coprocessador Intel Phi
  - 24 (2 cpus amb 12 cores) a 2.6 GHz
  - Memòria
  - Collserola1 -> 512 GB
  - Collserola2-9 -> 256 GB
  - Nodes interconnectats amb Infiniband (56 Gbps).

**Bull Sequana X800 - Canigó (2018)**

- **Memòria compartida**
  - 384 cores a 2.7 GHz
  - 9 TB memòria
  - 40 TB emmagatzematge
  - Connectada amb Infiniband (100 Gbps) a clúster BeeGFS (emmagatzematge compartit /scratch).



## Maquinari d'Emmagatzematge

### Bull Sequana X550 - Pirineus 2 (2018)

- Clúster heterogeni
- **46 nodes estàndard**
  - 48 cores (2 cpus de 24 cores) a 2.7 GHz
  - 192 GB memòria
  - 4 TB emmagatzematge
- **6 nodes FAT**
  - 48 cores (2 cpus de 24 cores) a 2.7 GHz
  - 384 GB memòria
  - 4 TB emmagatzematge
- **4 nodes amb 2 GPU's**
  - 48 cores (2 cpus de 24 cores) a 2.7 GHz
  - 192 GB memòria
  - 4 TB emmagatzematge
  - GPU's:
    - 3584 nuclis CUDA
    - 12 GB de memòria
    - Rendiment de 4,7 Tflop/S

- **4 nodes amb processadors Intel KNL 7250**
  - 68 cores a 1,4 GHz
  - 384 GB memòria
  - 4 TB emmagatzematge

Tots els nodes estan interconnectats a clúster BeeGFS (emmagatzematge compartit /scratch) mitjançant Infiniband (100 Gbps).

Aquesta serà la màquina per defecte on s'enviaran tots els càlculs que no tinguin requeriments especials.

## Best practices

- NEVER run scripts outside of Workload scheduler
- Use the short or interactive queue to test your simulations
- Work on /shared or /scratch for light i/o jobs
- Work on /tmp for intensive i/o jobs
- Store temporary files on /scracth
- Move important files to /projects

## Books

- Introduction to high-performance scientific computing (2nd edition), by Victor Eijkhout, Lulu, 2015
- Introduction to High Performance Computing for Scientists and Engineers, Hager, Wellein, CRC Press, 2010.
- High Performance Computing, Sterling, Anderson, Brodowicz, Morgan Kaufmann, 2017.

# Contents

1. Quick introduction to the HPC
2. Hands-on @ public HPC

## Hands-on @ CSUC

- Get data from github
- Access login node
- Explore environment
- Execute scripts through SLURM
  - 1. Serial
  - 2. OpenMP
  - 3. MPI
  - 4. Hybrid



## Get data from GitHub

Get the data from [GitHub](#) - or from the folder 2023UB-formation

Contains all the scripts to run, a README and the list of command to be run.

```
git clone https://github.com/kErica/2023UB-formation.git
```

## Access the cluster

User: curs\$NUM [1..23]

Password: c1rs\$NUM (for example curs24 password c1rs24)

Change the password at first login: 8 characters, 1 uppercase , 1 lowercase, 1 number, 1 special character.

```
## login  
  
ssh curs$NUM@hpc.csuc.cat -p 2122
```

## Access and move the data

Moving the data from source to destination and access CSUC cluster

```
## transfer files
scp -rp -P 2122 2023UB-formation curs$NUM@hpc.csuc.cat:/home/curs$NUM

## or
cd 2023UB-formation
sftp -oPort=2122 <YOUR_USERNAME>@hpc.csuc.cat
mput -R *
```

# Explore the environment

In the login node you can see:

- available resources
- scheduler partition
- access storage

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/vdal	7.9G	6.7G	1.2G	86%	/
devtmpfs	7.8G	0	7.8G	0%	/dev
tmpfs	7.8G	0	7.8G	0%	/dev/shm
tmpfs	7.8G	641M	7.2G	9%	/run
tmpfs	7.8G	0	7.8G	0%	/sys/fs/cgroup
cabreraSas.xe:/cap_prod/prod	3.5T	3.5T	81G	98%	/prod
netapp.xe:/vol/cap_nfs/home	7.0T	6.5T	537G	93%	/home
netapp.xe:/vol/cap_sdf/sdf	240G	189G	52G	79%	/home/sdf
192.168.93.123:/slurm_cap	2.0G	0	2.0G	0%	/etc/slurm
192.168.93.122:/cap_dades	10T	4.1T	6.0T	41%	/dades
192.168.93.201:/ubbiomed/beegfs_nodev	25T	8.9T	17T	36%	/dades/ubbiomed
	175T	159T	17T	91%	/mnt/beegfs

## Explore the scheduler commands

You can also check the scheduler commands and partitions in login nodes

```
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 job
```

## Get an interactive session

Check the options of the application and request 4h of interactive time

```
salloc -h
Usage: salloc [OPTIONS...] [command [args...]]
```

```
salloc --time 4:00:00 srun --pty bash
```

```
## where installed
interactive -t 4
```

# Execute scripts through SLURM

Send a serial job (not parallel) and an array of jobs to the scheduler

```
>cat serial.slm

#!/bin/bash
#SBATCH -J Serial          # Jobname
#SBATCH --ntasks=1           # Processors
#SBATCH --time=10:00          # Walltime
#SBATCH --mem-per-cpu=2G     # memory/cpu
#SBATCH -o serial-%j.out
#SBATCH -e serial-%j.err
#SBATCH --reservation=hpcnow

srun sleep 30
srun echo "My first serial Slurm job"
```

# Execute scripts through SLURM

Send a serial job (not parallel) and an array of jobs to the scheduler

```
>cat serial-array.slm

#!/bin/bash
#SBATCH -J Serial          # Jobname
#SBATCH --ntasks=1           # Processors
#SBATCH --time=10:00          # Walltime
#SBATCH --mem-per-cpu=2G     # memory/cpu
#SBATCH -o serial-%j.out
#SBATCH -e serial-%j.err
#SBATCH --reservation=hpcnow

srun sleep 30
srun echo "My first serial Slurm job - ${SLURM_ARRAY_TASK_ID}"
```

# Execute scripts through SLURM

Send a serial job (not parallel) and an array of jobs to the scheduler

```
## send a job to the queue
sbatch serial.slm
```

```
## send an array of job to the queue
sbatch --array=0-9 serial-array.slm
```

# Execute scripts through SLURM

## Execute an OpenMP job (parallel, no MPI)

```
>cat openmp.slm

#!/bin/bash
#SBATCH -J OMP
#SBATCH --time=10:00          # Walltime
#SBATCH -n 16                 # Number of cores
#SBATCH -N 1                  # Number of nodes
#SBATCH --reservation=hpcnow
#SBATCH --mem-per-cpu=800M # memory/cpu

export OMP_NUM_THREADS=16

/home/$USER/2023UB-formation/02-OpenMP/hello > hello.log
```

# Execute scripts through SLURM

## Execute an OpenMP job (parallel, no MPI)

```
sbatch openmp.slm

## check the output
ls # note that the default outfile is created for slurm
cat hello.log # the output file mentioned in the script
```

# Execute scripts through SLURM

Execute an job with different options: **openMP**, hybrid and **MPI**

```
#!/bin/bash

#SBATCH -J gmx_omp_8
#SBATCH -e gmx_omp_8.%j.err
#SBATCH -o gmx_omp_8.%j.out
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 8
#SBATCH --mem-per-cpu=2G
#SBATCH -t 30
#SBATCH --reservation=hpcnow

module load gromacs/2018.3

OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK:-1}

cp ${SLURM_SUBMIT_DIR}/ion_channel.tpr ${SCRATCH}
cd ${SCRATCH}

srun gmx_mpi mdrun -ntomp $OMP_NUM_THREADS -ntomp_pme 0 -s ion_channel.tpr
cp md.log ${SLURM_SUBMIT_DIR}/omp-md.${SLURM_JOB_ID}.log
```

## Execute scripts through SLURM

Execute an job with different options: **openMP**, hybrid and **MPI**

```
sbatch gmx_omp_8.slm

## check the output
## check the errors and logs
```

# Execute scripts through SLURM

## Execute an job with different options: openMP, hybrid and MPI

```
#!/bin/bash

#SBATCH -J gmx_hyb_8
#SBATCH -e gmx_hyb_8.%j.err
#SBATCH -o gmx_hyb_8.%j.log
#SBATCH -N 2
#SBATCH --ntasks-per-node 1
#SBATCH -c 4
#SBATCH --mem-per-cpu=2G
#SBATCH -t 30
#SBATCH --reservation=hpcnow

module load apps/gromacs/2018.3

OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK:-1}

cp ${SLURM_SUBMIT_DIR}/ion_channel.tpr ${SCRATCH}
cd ${SCRATCH}

srun gmx_mpi mdrun -ntomp $OMP_NUM_THREADS -npme 0 -ntomp_pme 0 -s ion_channel.tpr
cp md.log ${SLURM_SUBMIT_DIR}/hyb-md.${SLURM_JOB_ID}.log
```

## Execute scripts through SLURM

Execute an job with different options: openMP, **hybrid** and MPI

```
sbatch gmx_hyb_8.slm

## check the output
## check the errors and logs
```

# Execute scripts through SLURM

## Execute an job with different options: openMP, hybrid and MPI

```
#!/bin/bash

#SBATCH -J gmx_mpi_8
#SBATCH -e gmx_mpi_8.%j.err
#SBATCH -o gmx_mpi_8.%j.log
#SBATCH -N 2
#SBATCH -n 4
#SBATCH -t 30
#SBATCH --mem-per-cpu=2G
#SBATCH --reservation=hpcnow

module load apps/gromacs/2018.3

OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK:-1}

cp ${SLURM_SUBMIT_DIR}/ion_channel.tpr ${SCRATCH}
cd ${SCRATCH}

srun gmx_mpi mdrun -ntomp $OMP_NUM_THREADS -npme 0 -ntomp_pme 0 -s ion_channel.tpr
cp md.log ${SLURM_SUBMIT_DIR}/mpi-md.${SLURM_JOB_ID}.log
```

# Execute scripts through SLURM

Execute an job with different options: openMP, hybrid and **MPI**

```
sbatch gmx_mpi_8.slm

## check the output
## check the errors and logs
```

# Execute scripts through SLURM

Comparison OpenMP,

hybrid and

MPI job Slurm parameters

```
#SBATCH -J gmx_omp_8
#SBATCH -e gmx_omp_8.%j.err
#SBATCH -o gmx_omp_8.%j.out
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 8
#SBATCH --mem-per-cpu=2G
#SBATCH -t 30
```

```
#SBATCH -J gmx_hyb_8
#SBATCH -e gmx_hyb_8.%j.err
#SBATCH -o gmx_hyb_8.%j.log
#SBATCH -N 2
#SBATCH --ntasks-per-node=1
#SBATCH -c 4
#SBATCH --mem-per-cpu=2G
#SBATCH -t 30
```

```
#SBATCH -J gmx_mpi_8
#SBATCH -e gmx_mpi_8.%j.err
#SBATCH -o gmx_mpi_8.%j.log
#SBATCH -N 2
#SBATCH -n 4
#SBATCH -t 30
#SBATCH --mem-per-cpu=2G
```

1 node (-N)

1 task (-n) - *default*

8 cpus per task (-c)

2 nodes (-N)

1 task per node (--ntasks)

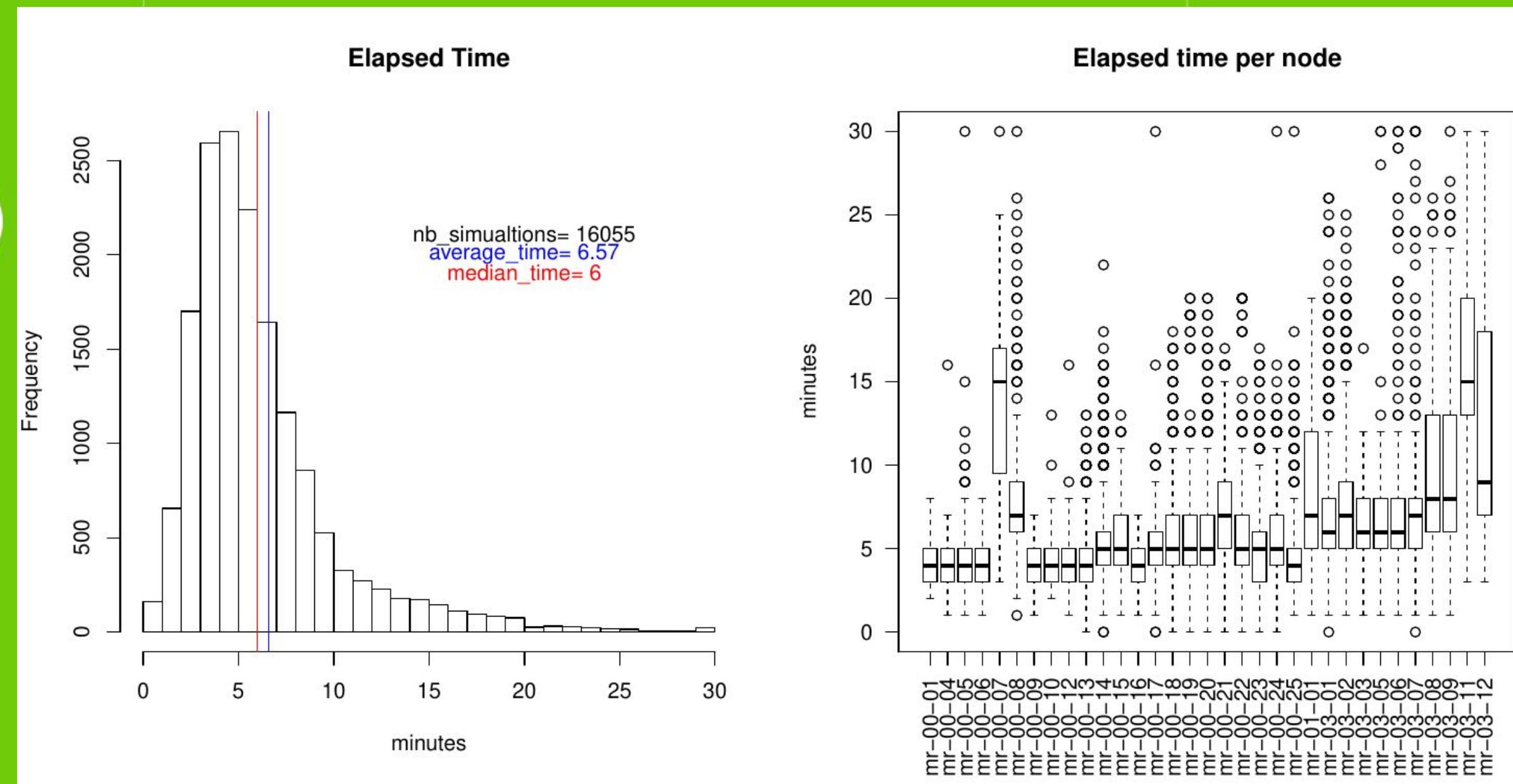
4 cpus per task (-c)

2 nodes (-N)

4 tasks (-n)



# Questions?



Erica Bianco, PhD  
Computational Scientist  
[erica.bianco@hpcnow.com](mailto:erica.bianco@hpcnow.com)  
[www.hpcnow.com](http://www.hpcnow.com)