GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Instituto de Salud Carlos III

# Scripting & Parallelization SLURM

## Sbatch, JobArray, OpenMP/MPI


**Daniel Valle Millares**
**(Bioinformatics Platform - CIBERINFEC)**


**BU-ISCIII**
**29 sept - 03 de octubre de 2025**
**1ª edición**

# Index

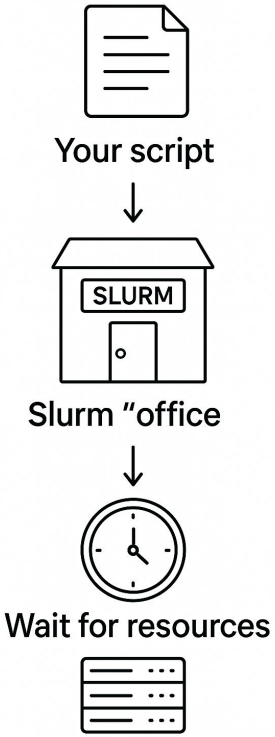
1. **Scripting on the cluster — Slurm: sbatch & job arrays**
2. Parallelization on our cluster — OpenMP vs MPI (when to use each)
3. From scripts to workflows — building a reproducible pipeline (Nextflow preview)
4. Wrap-up & Q&A

# Sbatch File

# Scripting on the cluster — Slurm: sbatch

## What is sbatch?

- Command to submit a job script to Slurm; runs in the background on compute nodes.
- Delegate the work to de cluster:
  - Submit once, let the cluster work
- Example:
  - Your script → Slurm "office" → Wait for resources → Cluster runs it



# Scripting on the cluster — Slurm: sbatch

## Set up a sbatch script file

- 0) Open a <file-name>.sbatch file
- 1) Init with shebang

```
#!/bin/bash
#SBATCH --option=value
#SBATCH --option=value
#SBATCH --option=value
# From here on, the commands
command_1
command_2
```

# Scripting on the cluster — Slurm: sbatch

## Set up a sbatch script file

- 0) Open a <file-name>.sbatch file
- 1) Init with shebang
- 2) Set up slurm directives at the beginning of the script by using “#SBATCH”

```
#!/bin/bash
#SBATCH --option=value
#SBATCH --option=value
#SBATCH --option=value
# From here on, the commands
command_1
command_2
```

# Scripting on the cluster — Slurm: sbatch

## Set up a sbatch script file

- 0) Open a <file-name>.sbatch file
- 1) Init with shebang
- 2) Set up slurm directives at the beginning of the script by using “#SBATCH”
- 3) **After directives, you can:**
  - Load dependencies with ‘module load’
  - Set your commands you want to run

```
#!/bin/bash
#SBATCH --option=value
#SBATCH --option=value
#SBATCH --option=value
# From here on, the commands
command_1
command_2
```

# Scripting on the cluster — Slurm: sbatch

```
#!/bin/bash
#SBATCH --chdir=/path/to/working/directory   # Folder where the analysis will run
#SBATCH --job-name=my_first_slurm_job        # A recognizable job name
#SBATCH --cpus-per-task=1                    # Number of CPU cores (threads) for this job
#SBATCH --mem=1G                             # RAM to reserve
#SBATCH --time=00:10:00                      # Time limit (HH:MM:SS)
#SBATCH --partition=short_idx                # Queue/partition to run in
#SBATCH --output=slurm-%j.out                # File for standard output
#SBATCH --error=slurm-%j.err                 # File for standard error
# From here on, the commands we want to run:
command_1
command_2
```

```
#!/bin/bash
#SBATCH --job-name=fastqc_demo
#SBATCH --partition=short_idx
#SBATCH --cpus-per-task=1
#SBATCH --mem=4G
#SBATCH --time=00:05:00
#SBATCH --output=logs/%x-%j.out
#SBATCH --error=logs/%x-%j.err

# Carga dependencias
module load FastQC/0.11.9-Java-11

# Crea la carpeta de resultados
mkdir -p 01-fastqc-demo-results

# Ejecuta fastqc
fastqc \
  /scratch/hpc_course/*HPC-COURSE_${USER}/ANALYSIS/00-reads/virus1_R1.fastq.gz \
  /scratch/hpc_course/*HPC-COURSE_${USER}/ANALYSIS/00-reads/virus1_R2.fastq.gz \
  -o 01-fastqc-demo-results

echo "[INFO] Finished at $(date)"
```

# Scripting on the cluster — Slurm: sbatch

Lanza el trabajo y monitoriza:

```
sbatch fastqc_overask.sbatch
squeue -u $USER -o "%8i %22j %4t %10u %20q %20P %10Q %5D %11l %11L %50R %10C %c"
scontrol show job <JOBID> | egrep 'Reason|Req|MinCPUs|TRES|Nodes|Partition|QOS'
```

# Scripting on the cluster — Slurm: sbatch

Lanza el trabajo y monitoriza:

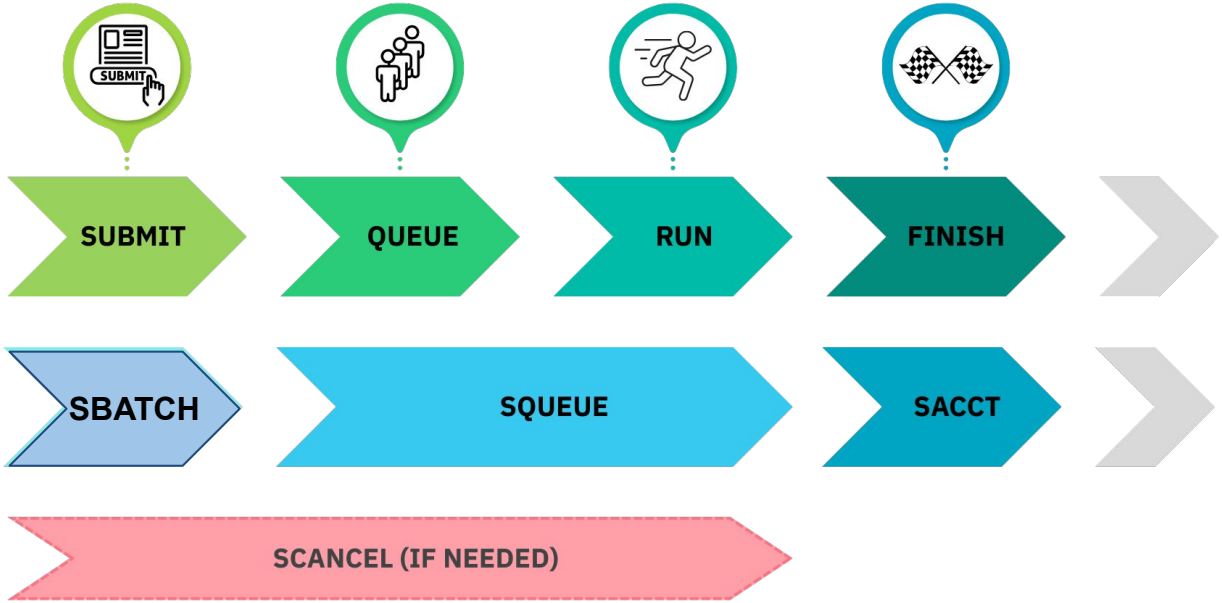
```
sbatch fastqc_overask.sbatch
squeue -u $USER -o "%8i %22j %4t %10u %20q %20P %10Q %5D %11l %11L %50R %10C %c"
scontrol show job <JOBID> | egrep 'Reason|Req|MinCPUs|TRES|Nodes|Partition|QOS'
```

```
sbatch --chdir=/path_to/07-scripting-and-parallelization fastqc_overask.sbatch
squeue -u $USER -o "%8i %22j %4t %10u %20q %20P %10Q %5D %11l %11L %50R %10C %c"
scontrol show job <JOBID> | egrep 'Reason|Req|MinCPUs|TRES|Nodes|Partition|QOS'
```

# Scripting on the cluster — Slurm: sbatch

```
# Submit
$ sbatch fastqc_slurm.sbatch
Submitted batch job 12345

# See the queue
$ squeue --me
             JOBID PARTITION     NAME     USER ST   TIME  NODES NODELIST(REASON)
           12345_2 short_idx fastqc_2     dani  R  00:01:12     1 ideafix03
           12345_3 short_idx fastqc_3     dani PD  00:00:00     1 (Resources)
             12344      long   spades     dani CG  01:59:58     1 ideafix07
             12343      long   spades     dani  R  00:10:21     2 ideafix05,ideafix06
             ▲          ▲        ▲         ▲  ▲     ▲        ▲             ▲
             |          |        |         |  |     |        |             └─ Node(s) assigned;
             |          |        |         |  |     |        └─ Number of nodes reserved
             |          |        |         |  |     └─ Elapsed time (HH:MM:SS)
             |          |        |         |  └─ Short state (PD/R/CG/…)
             |          |        |         └─ User
             |          |        └─ Job name (--job-name)
             |          └─ Partition / queue (e.g., short_idx)
             └─ JobID (arrays: JobID_task, e.g., 12345_2)
```

# Monitoring and job states



# Scripting on the cluster — Slurm: sbatch


**Bigger asks = longer wait, not faster.**

# Scripting on the cluster — Slurm: sbatch

## Once the job/s have finished → Output files

- **Standard output**: Standard outputs of your program.
  For example, if you used: **--output=slurm-%j.out**

**slurm-12345.out**

- **Standard error**: Potential errors you face during execution appear here.
  For example, if you used: **--error=slurm-%j.err**

**slurm-12345.err**

# Scripting on the cluster — Slurm: sbatch

## Good Practices:

- ✅Descriptive **--job-name**; comment your script.
- ❌Never run heavy jobs on the login node !!!! Use **sbatch/srun**.
- ❌Do not request more resources than need!!!!
- ✅Test small first, then scale.
- ✅Version your scripts (Git).

# Thank you for your attention

Questions?

# JobArrays

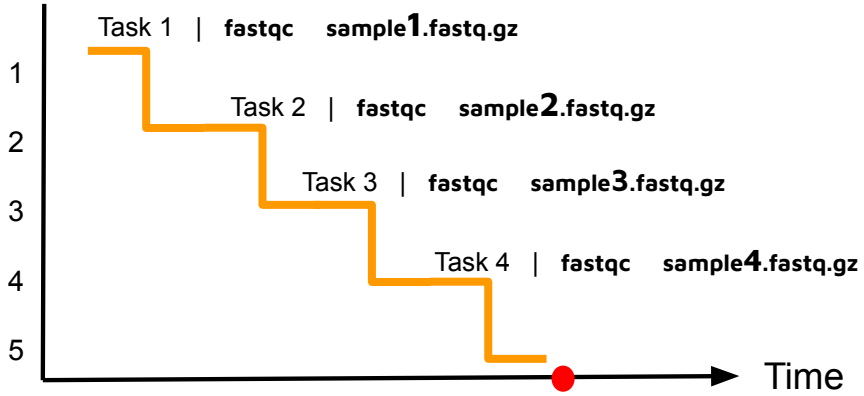
# Scripting on the cluster — Slurm: job arrays

## What are JobArrays and what are they used for?

- Sbatch script designed to launch many jobs
  - Same operation / multiple inputs
- Instead of creating 50 scripts for 50 samples, you create one script and tell Slurm to run it N times → **In Parallel**
- If your shell for loop only changes a filename/ID, make it a job array.

# Scripting on the cluster — Slurm: job arrays

## For Loop




## JobArray

# Scripting on the cluster — Slurm: job arrays

## Create a JobArray script?

- As simple as using the `**--array**` inside the sbatch script
- **Plus env variables**

```
#!/bin/bash
#SBATCH --chdir=/path/to/project
#SBATCH --job-name=fastqc_array
#SBATCH --partition=short_idx
#SBATCH --array=1-20
#SBATCH --cpus-per-task=1
#SBATCH --mem=5G
#SBATCH --time=00:15:00
#SBATCH --output=fastqc_%A_%a.out # %A: array JobID, %a: task index
#SBATCH --error=fastqc_%A_%a.err

module load fastqc/0.12.1

mkdir fastqc_results
fastqc -o fastqc_results muestra_${SLURM_ARRAY_TASK_ID}.fq.gz
```

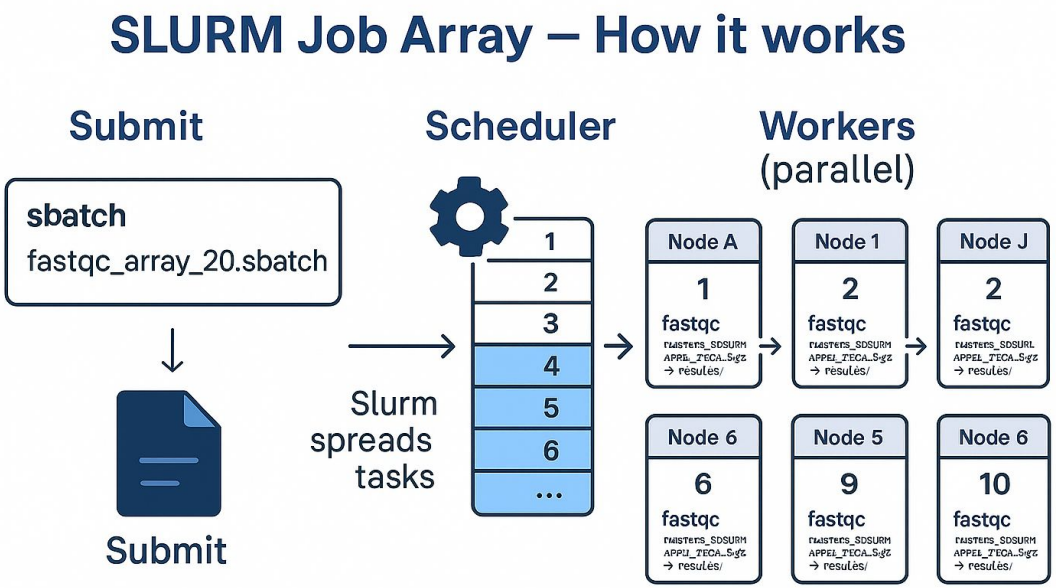
https://docs.hpc.shef.ac.uk/en/latest/referenceinfo/scheduler/SLURM/SLURM-environment-variables.html#gsc.tab=0

# Scripting on the cluster — Slurm: job arrays

## Monitoring Arrays

```
$ sbatch fastqc_array_20.sbatch
Job Submitted
$ squeue --me
             JOBID PARTITION     NAME       USER ST   TIME  NODES NODELIST(REASON)
          78901_1   short_idx fastqc_array   dani  R   0:41      1 ideafix02
          78901_2   short_idx fastqc_array   dani  R   0:39      1 ideafix03
          78901_3   short_idx fastqc_array   dani  R   0:36      1 ideafix04
          78901_4   short_idx fastqc_array   dani PD   0:00      1 (Resources)
          78901_5   short_idx fastqc_array   dani PD   0:00      1 (Resources)
          78901_6   short_idx fastqc_array   dani PD   0:00      1 (Resources)
          78901_7   short_idx fastqc_array   dani PD   0:00      1 (Priority)
          78901_8   short_idx fastqc_array   dani PD   0:00      1 (Priority)
          78901_9   short_idx fastqc_array   dani PD   0:00      1 (Priority)
         78901_10   short_idx fastqc_array   dani PD   0:00      1 (Priority)
```

# Scripting on the cluster — Slurm: job arrays




- $SLURM_ARRAY, TASK_ID = task index
- %A = array JobID, %a = task index (used in log file names)

# Thank you for your attention

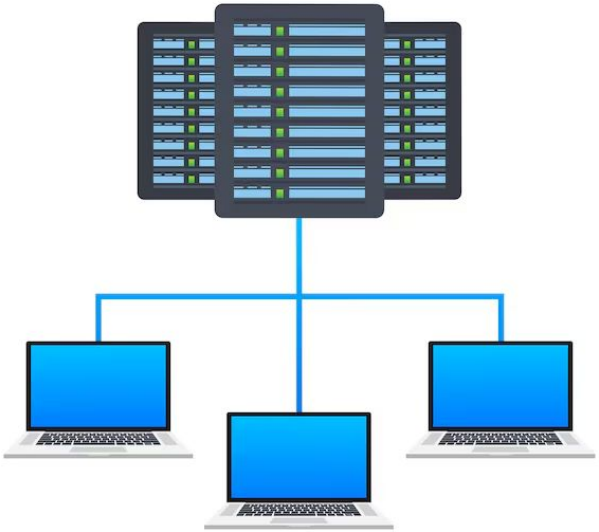
Questions?

# OpenMP v.s. MPI

# Parallelization on our cluster — OpenMP vs MPI

## Parallel programing -- OpenMP and MPI

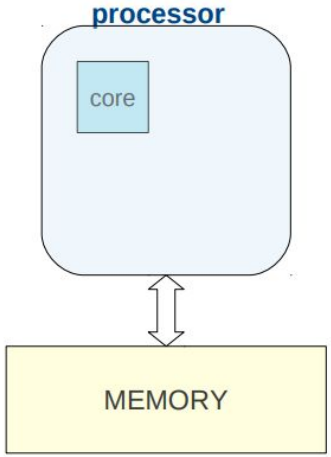
- We use it to finish faster.
- It **reduces time-to-results**, makes better use of cluster resources, and cuts wall-clock time for analyses.
- The two most used HPC models are **OpenMP** and **MPI**.
- They target different scenarios:
  - OpenMP (shared memory, one node)
  - vs MPI (distributed memory, many nodes).



# Parallelization on our cluster — OpenMP vs MPI

## Why parallel computing?

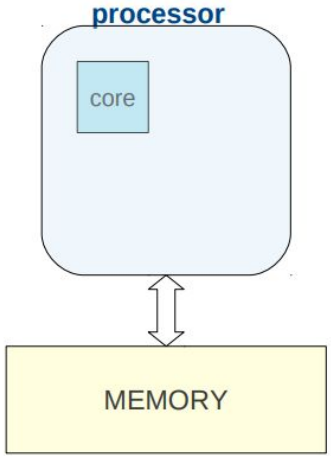
Older processor had only one cpu core to execute instructions



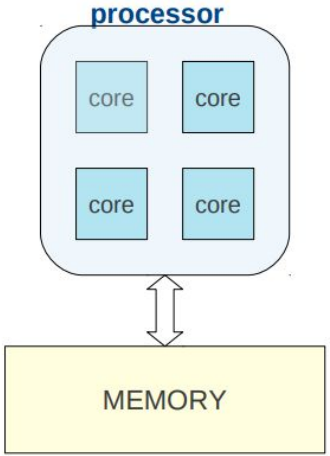
# Parallelization on our cluster — OpenMP vs MPI

## Why parallel computing?

Older processor had only one cpu core to execute instructions




Modern processors have 4 or more independent cpu cores to execute instructions

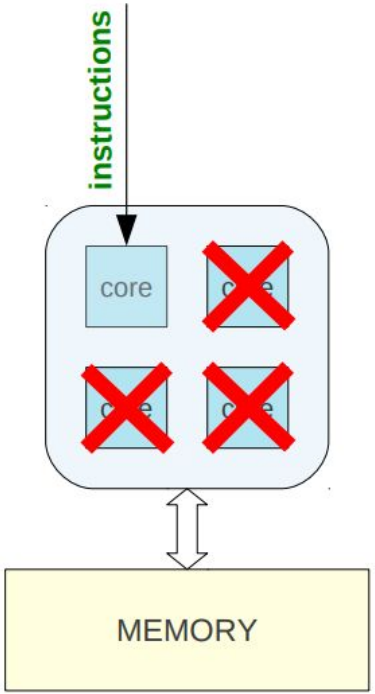


# Parallelization on our cluster — OpenMP vs MPI

## Why parallel computing?

When you run a sequential program it has an instruction to run a task on 1 core.

Other cores are idle



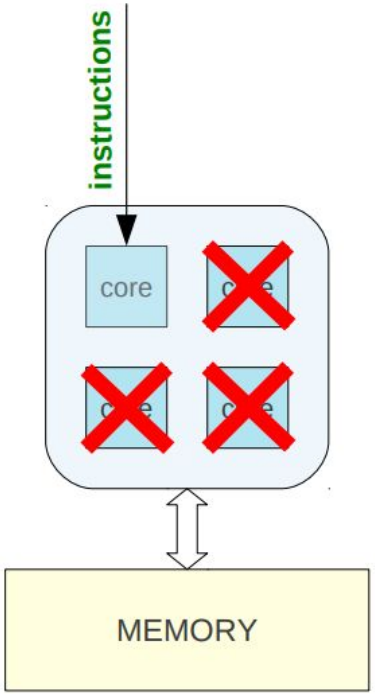
# Parallelization on our cluster — OpenMP vs MPI

## Why parallel computing?

When you run a sequential program it has an instruction to run a task on 1 core.

Other cores are idle

Waste of available resources…



# Parallelization on our cluster — OpenMP

## OpenMP Essentials

- Shared-memory parallel model
- Lets a program to use **multiple threads** inside a **single process** running **on a node** (usually has >>> CPUs).
- That node shares data in a common RAM

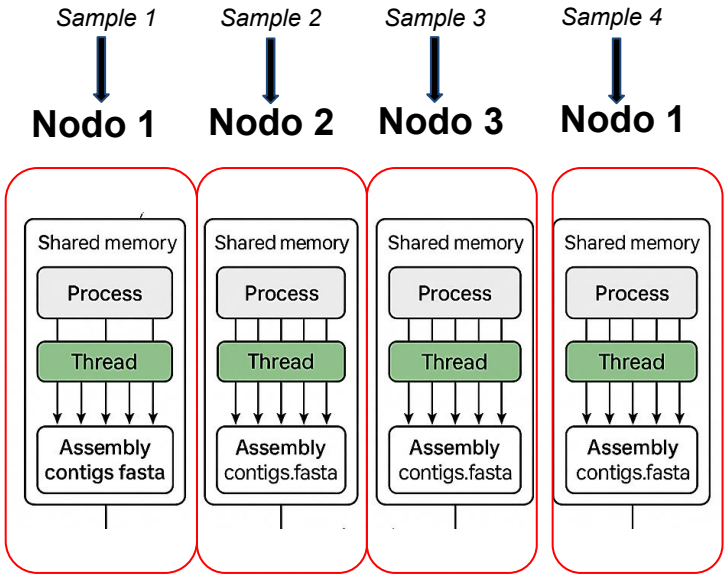
**Most bioinformatics tools use the OpenMP model**

- **Aligners**
- **Assemblers**
- **Read Processing**
- **Variant Callers**
- **...**

# Parallelization on our cluster — OpenMP

## OpenMP Essentials

- Shared-memory parallel model
- Lets a program to use **multiple threads** inside a **single process** running **<u>on a node</u>** (usually has >>> CPUs).
- That node shares data in a common RAM




**Outputs:**
sample1/conttigs.fasta
sample2/contigs.fasta
sample3/contigs.fasta
sample4/contigs.fasta

# Parallelization on our cluster — OpenMP

```
#!/bin/bash
#SBATCH --job-name=spades_openmp
#SBATCH --partition=short_idx
#SBATCH --cpus-per-task=16
#SBATCH --mem=32G
#SBATCH --time=02:00:00
#SBATCH --output=logs/%x-%j.out
#SBATCH --error=logs/%x-%j.err

module load SPAdes/3.15.2-GCC-10.2.0

mkdir -p 04-openmp-spades-results

R1=../00-reads/ERR2261314_R1.fastq.gz
R2=../00-reads/ERR2261314_R2.fastq.gz

spades.py -1 "$R1" -2 "$R2" -o 04-openmp-spades-results/spades_sample01 \
    --threads "$SLURM_CPUS_PER_TASK" \
    --mem $SLURM_MEM_PER_NODE
```

# Parallelization on our cluster — MPI

## MPI Essentials

- It is designed to run an application using **multiple separate processes**, possibly on **different nodes** of a cluster
- When: multinode scaling & very large memory needs.
- Shares resources by passing messages over the cluster's network → network issues can stop the job

**Only a few bioinformatics tools use the MPI model**

**RAxML**

**IQ-Tree**

# Parallelization on our cluster — MPI

## MPI Essentials

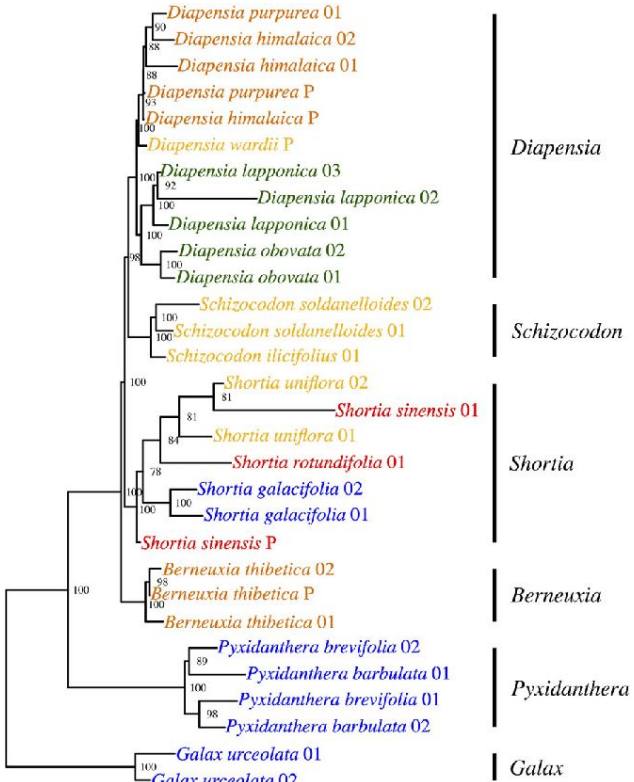
- It is designed to run an application using **multiple separate processes**, possibly on **different nodes** of a cluster
- When: multinode scaling & very large memory needs.
- Shares resources by passing messages over the cluster's network → network issues can stop the job

**Otras áreas**

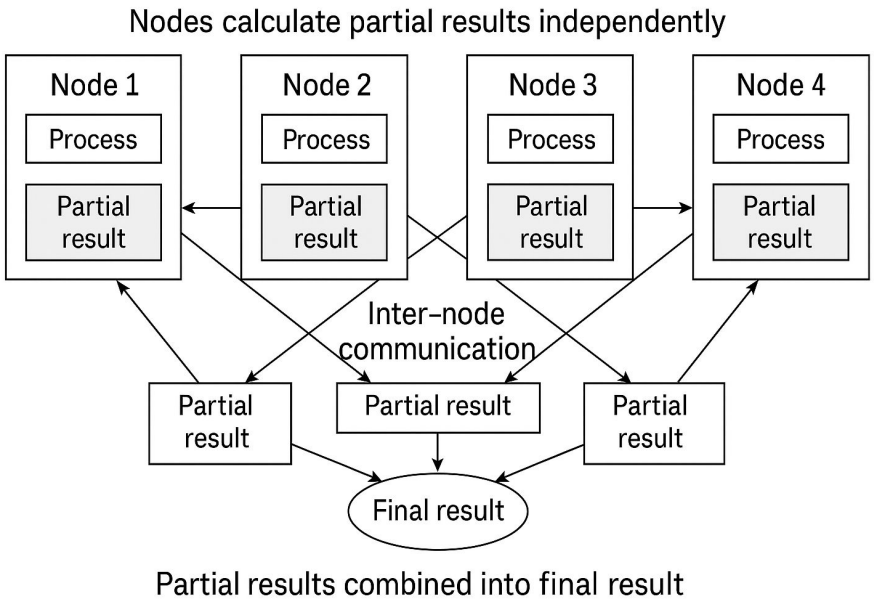
- **Modelos de propagación de enfermedades**
- **Reconstrucción de imágenes 3D**
- **Modelos climáticos y de contaminación**
- **Sincronización de modelos ML (gradient descent trees)**

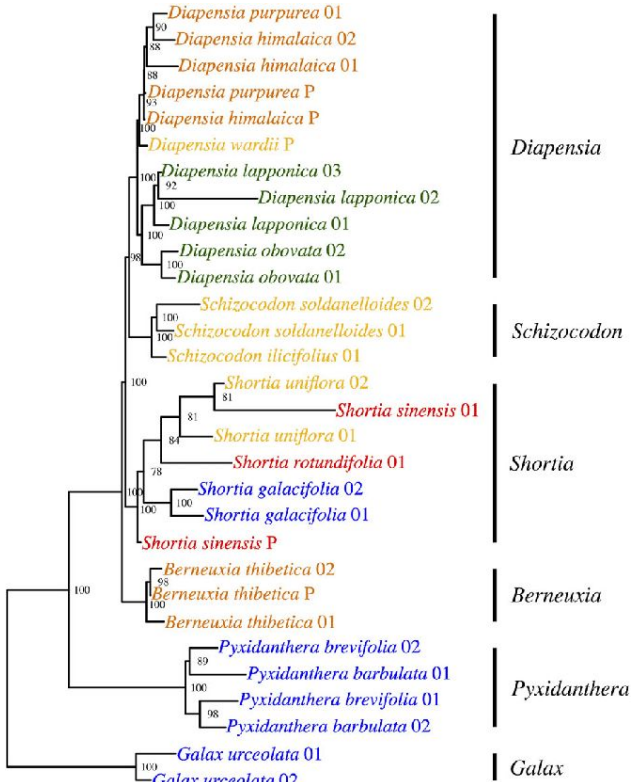
# Ejemplo de **MPI** con herramienta bioinfo **RAxML**

- Imagina un alineamiento de 100.000 secuencias.
- No cabe en memoria de un solo nodo.
- Necesitas usar **varios nodos**, cada uno procesa una parte del árbol y comparten información vía mensajes.



# Ejemplo de **MPI** con herramienta bioinfo **RAxML**



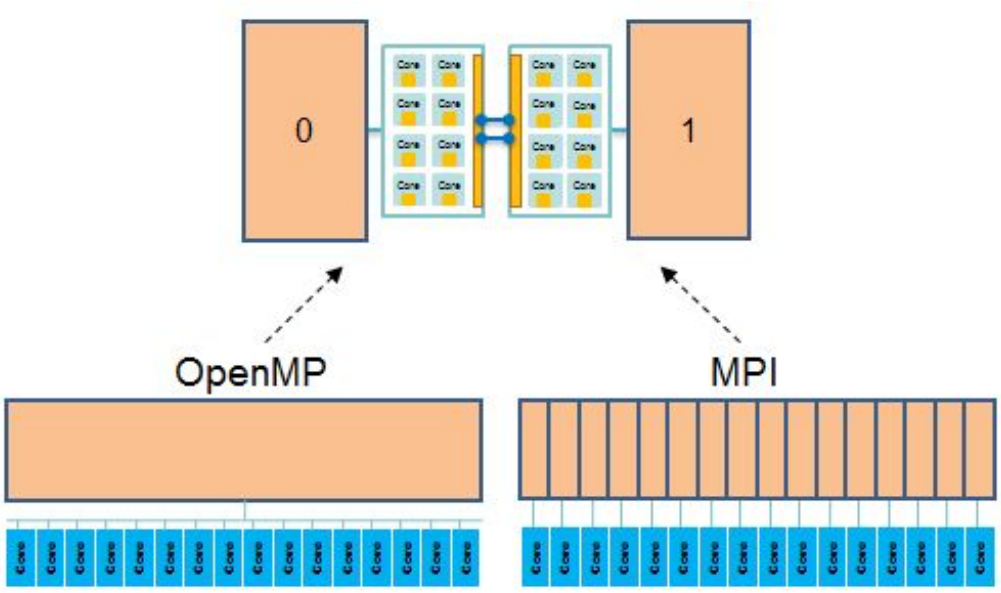
# Ejemplo de **MPI** con herramienta bioinfo **RAxML**

```
#!/bin/bash
#SBATCH --job-name=raxml_mpi
#SBATCH --partition=short_idx
#SBATCH --nodes=2                # <-- nº de nodos
#SBATCH --ntasks=8               # total procesos MPI
#SBATCH --ntasks-per-node=4      # <-- nº procesos MPI por nodo
#SBATCH --mem=8G
#SBATCH --time=00:30:00
#SBATCH --output=logs/%x-%j.out
#SBATCH --error=logs/%x-%j.err

module load RAxML/8.2.12-gompi-2020a-hybrid-avx2  # el módulo puede traer varios binarios

mpirun -np "$SLURM_NTASKS" raxmlHPC \
  -s data/datos.phy \
  -m GTRGAMMA \
  -p 12345 \
  -# 20 \
  -n "$RUNNAME" \
  -w "$RESULTS_DIR"
```

# Summary

- OpenMP
  - launch one <u>process</u> *per node*, **just on a single node**
  - share data using <u>shared memory</u>
  - can't share data with a different process

- MPI
  - launch one process *per core*, **on one node or on many nodes.**
  - **pass messages** among processes without concern for node location: allows intra and inter node communication.




https://cvw.cac.cornell.edu/hybrid-openmp-mpi/intro/node-views

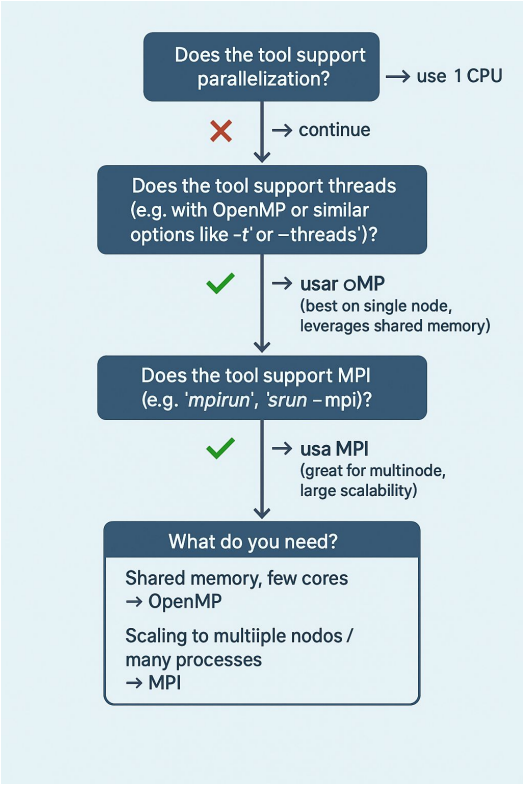
# Parallelization on our cluster — OpenMP vs MPI

## KEY DIFFERENCES

| Technology | Parallelization level | Communication between processes | Typical use case | Usage (sbatch script) |
|---|---|---|---|---|
| **OpenMP** | Within **one node** | Shared memory: all threads access the same RAM | Align 200 million reads on one node using all its cores | **--cpus-per-task (threads)**<br>**--mem (Ram for the whole process)** |
| **MPI** | Across **multiple nodes** | Distributed memory: each node has its own RAM; communication by network | Build a very large phylogenetic tree by splitting the work over 4 nodes | **--nodes**<br>**--ntasks**<br>**--ntasks-per-node** |

# Parallelization on our cluster — OpenMP vs MPI

## When to use OpenMP vs MPI?



# Thank you for your attention

Questions?

# HANDS-ON