

Cluster for the Laboratory: KAHAN

J. M. Alonso, P. Alonso, F. Alvarruiz, I. Blanquer,
J. Ibáñez, E. Ramos, J. E. Román

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Year 2025/26



1

Contents

- 1** Laboratory Cluster: KAHAN
 - Cluster for the Laboratory
 - Execution of Parallel Programs

2

Section 1

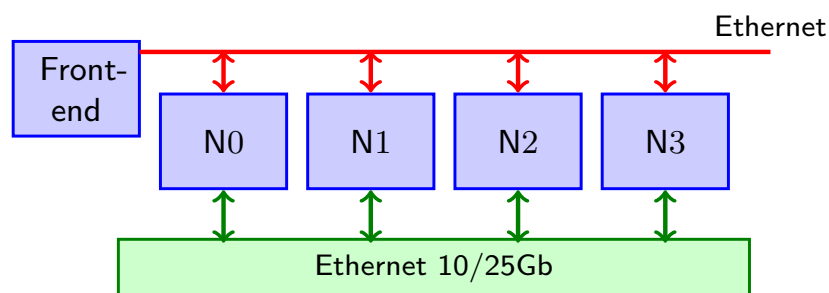
Laboratory Cluster: KAHAN

- Cluster for the Laboratory
- Execution of Parallel Programs

3

Cluster for the Laboratory

Hardware configuration: 4 nodes



Each node:

- 1 AMD EPYC 7551P 32 Cores (64 virtuals)
- 64GB RAM memory
- Disc SSD 240GB
- Ethernet 10/25Gb 2-port 622FLR -SFP28

In total: 4 processors, 128 cores (256 virtuals), 256 GB

4

Lab Cluster: Front-End

The front-end enables users to interact with the cluster

Accessing the front-end:

```
$ ssh -Y -l login@alumno.upv.es kahan.dsic.upv.es
```

For routine tasks (do not run costly executions)

- Editing and compiling programs
- Short test runs

Useful commands:

- Files/directories: cd, pwd, ls, cp, mkdir, rm, mv, scp, less, cat, chmod, find
- Processes: w, kill, ps, top
- Editors and other: vim, emacs, pico, man

5

Lab Cluster: Network

Gigabit Ethernet

- Auxiliary network, only for the O.S. traffic (ssh, NFS)

Ethernet 10/25Gb

- High-bandwidth low-latency network, perfect for clusters
- Ethernet card 10/25Gb 2-port 622FLR -SFP28
- Supports RDMA, RoCE, iWarp
- Can operate at 25 Gbps

6

Running Parallel Programs

OpenMP: executables can be run directly

Usually the number of threads should be indicated

```
$ OMP_NUM_THREADS=4 ./prgomp
```

Another option is to export the variables

```
$ OMP_NUM_THREADS=4; OMP_SCHEDULE=dynamic  
$ export OMP_NUM_THREADS OMP_SCHEDULE  
$ ./prgomp
```

MPI: use the command `mpiexec` (or `mpirun`)

Options: choose hosts, architecture

```
$ mpiexec -n 4 prgmpi <args>  
$ mpiexec -n 6 -host node1,node2,node5 prgmpi
```

7

Queue System

The **queue system** (or **job scheduler** or **resource manager**) is a software that enables sharing a cluster among different users

- The user can launch “jobs” usually in *batch* mode (non interactive) using one or several nodes
- A **job** is a particular execution with a set of attributes (nodes, maximum execution time, etc.)
- Job **scheduling** policies are defined
- The system accounts the resources used (hours)
- Objective: maximize usage and minimize waiting time

Working procedure:

- 1 A job is defined and then submitted to a queue, which returns an identifier
- 2 Depending on the workload, after a waiting time the job is run
- 3 The output is obtained when the job finishes

8

Lab Cluster: Queues (1)

We will use the SLURM queue system

Example of a job file jobopenmp.sh

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=5:00
#SBATCH --partition=cpa
OMP_NUM_THREADS=3 ./pintegral 1
```

- `--nodes`: number of requested nodes
- `--time`: requested execution time
- `--partition`: partition to be used in the queue system

For MPI, use `mpiexec` (no need to specify `-n`)

9

Lab Cluster: Queues (2)

For the submission of a job:

```
$ sbatch jobopenmp.sh
Submitted batch job 728
```

At the end, one file is created in the current directory:
`slurm-728.out`

To check the status of the job:

```
$ squeue
JOBID PARTITION NAME          USER ST TIME NODES NODELIST
728    cpa      jobopenmp.sh ramos R 0:01  1    kahan01
```

Possible status: queued (PD), running (R), completing (CD)

Job cancelling: `scancel`

10