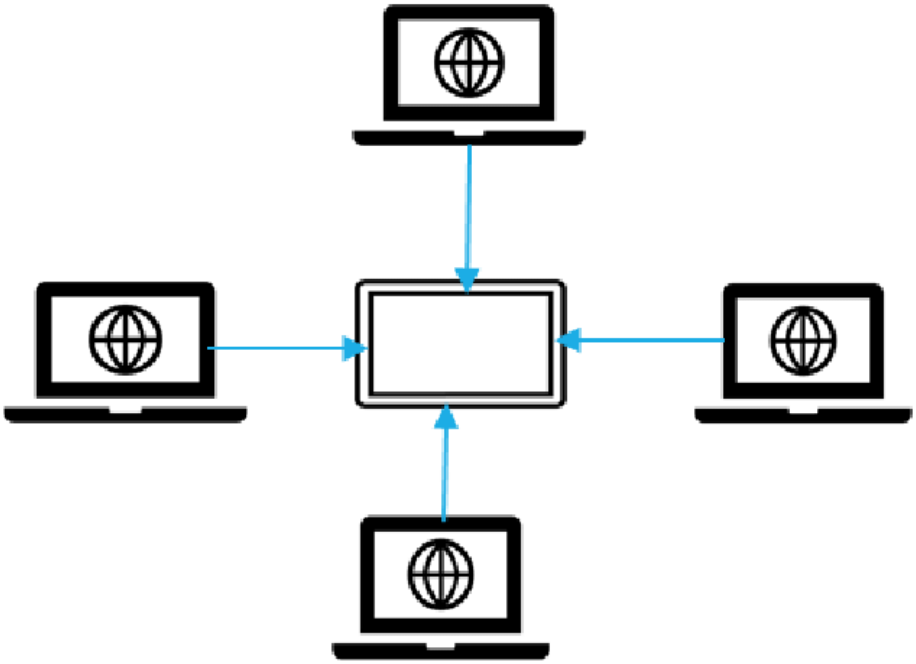
Guide for Creating a Computing Cluster on the Ubuntu Operating System

Jose Luciano Caamal Ayala

December 19, 2025



Contents

1 Introduction

In various academic and research contexts, certain projects require a considerably high level of computational power. Under these conditions, many of them cannot be properly developed due to limitations in the available hardware. However, there are alternatives that allow increasing processing capacity in an accessible and efficient manner, among which computing clusters stand out.

This work proposes the design and implementation of a computing cluster composed of two interconnected systems, using general-purpose hardware and open-source software. The main objective is to demonstrate that it is possible to build a functional parallel computing environment without the need for specialized infrastructure, enabling the execution of applications that benefit from parallelism.

The developed cluster has a primarily educational and experimental focus, aimed at learning fundamental concepts such as distributed computing, communication between nodes, and Linux-based system administration. Throughout this document, the hardware characteristics, operating system configuration, system interconnection, and the steps required for correct deployment and operation are described.

2 Basic Concepts of Clusters

A computing cluster is a set of independent computers, known as *nodes*, that are interconnected and work in a coordinated manner to function as a single system. The main objective of a cluster is to increase the available computational power, improve application performance, or facilitate the study and implementation of distributed computing techniques.

The nodes that make up a cluster can be homogeneous, when they have similar hardware characteristics, or heterogeneous, when they differ in processor, memory, storage, or graphics capabilities. In both cases, cooperation between nodes is achieved through communication mechanisms that allow proper task distribution and efficient information exchange.

There are different types of clusters, among the most common are the following:

- **High Performance Computing (HPC) clusters**: focused on executing applications that require high computational power, such as numerical simulations, scientific analysis, and parallel processing.
- **High availability clusters**: designed to ensure service continuity in the event of failures in one or more nodes.
- **Load balancing clusters**: used to efficiently distribute tasks among multiple nodes, optimizing the use of available resources.

In the context of this project, a small-scale computing cluster is implemented for educational and experimental purposes. This cluster consists of two heterogeneous nodes interconnected through a direct Ethernet connection, without the use of additional network infrastructure such as routers.

Communication between the nodes is carried out using the TCP/IP protocol over the Ethernet connection. Remote administration and command execution are performed using the SSH protocol, while parallel process execution relies on message-passing libraries such as MPI. This direct interconnection establishes a dedicated communication channel between nodes, reducing latency and simplifying network configuration.

3 Components

3.1 Interconnection Elements

- **Category 6 UTP Ethernet cables**: two 90 cm cables used for the physical interconnection of the systems (`https://www.steren.com.mx/cable-ethernet-utp-cat-6-de-90-cm.html`).

  ▲ **Observation:** this component was used because one of the devices does not have an integrated Ethernet port.

- **5-Port Fast Ethernet Switch**: used for the physical interconnection of the cluster nodes (`https://www.steren.com.mx/switch-fast-ethernet-de-5-puertos.html`).

3.2 Computing Equipment

The cluster consists of two laptop computers, each fulfilling a specific role within the system:

- **Computer A (Master node)**: MSI laptop responsible for cluster administration and coordination of computational tasks.
- **Computer B (Worker node)**: Lenovo laptop used as an additional computing node for executing parallel processes.

3.3 Technical Specifications

Table 1: Technical specifications of Computer A (Master node)

Component	**Specification**
Processor	`13th Gen Intel(R) Core(TM) i5-13420H`
RAM Memory	`16 GB`
Internal Storage	`Topesel Expansion X15 -- 1 TB`
Graphics Card (GPU)	`Intel(R) UHD Graphics (2 GB)`
	`NVIDIA GeForce RTX 4060 Laptop GPU (4 GB)`
Network Interface	`Ethernet: Realtek PCIe GbE Family Controller`
	`Wi-Fi: Intel(R) Wi-Fi 6E AX211 160 MHz`
Operating System	`Ubuntu 24.04.3 LTS`

Table 2: Technical specifications of Computer B (Worker node)

Component	**Specification**
Processor	`AMD Ryzen 5 5500U with Radeon Graphics`
RAM Memory	`12 GB`
Graphics Card (GPU)	`AMD Radeon Graphics (Lucienne)`
Network Interface	`Ethernet network adapter via micro USB`
Operating System	`Ubuntu 24.04.3 LTS`

4 Configuration

This section describes all the required configurations, as well as possible errors and their solutions, for the correct creation and operation of the cluster.

4.1 Configuration on Laptop A (Server / Master Node)

Initially, the physical connection of the equipment is performed, as shown in Figure 1.

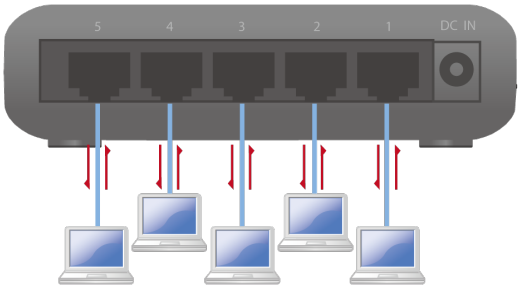


Figure 1: Fast Ethernet switch connection used in the cluster (two nodes).

Once the physical connection is completed, proceed with the network configuration in the operating system. Open the **Ubuntu Settings** and navigate to the **Network** section.

Under **Wired**, click on the **gear icon** to edit the connection.

Follow the steps below:

1. Access the **IPv4** tab.
2. Change the configuration method from **Automatic (DHCP)** to **Manual**.
3. In the **Addresses** section, enter the following values:
   - **IP Address:** 192.168.1.1
   - **Netmask:** 255.255.255.0
   - **Gateway:** leave blank or set to 0.0.0.0

Click **Apply** to save the changes.

4.2 Configuration on Laptop B (Worker Node)

This laptop will receive instructions from the master node and process part of the mesh generated by **OpenFOAM**.

Repeat the previous steps and configure:

- **IP Address:** 192.168.1.2
- **Netmask:** 255.255.255.0
- **Gateway:** leave blank

Apply the changes to complete the configuration.

4.3 Connection Verification

To ensure that **OpenFOAM** can communicate correctly between the cluster nodes, it is necessary to perform a connectivity test using the `ping` command.

Connectivity Test

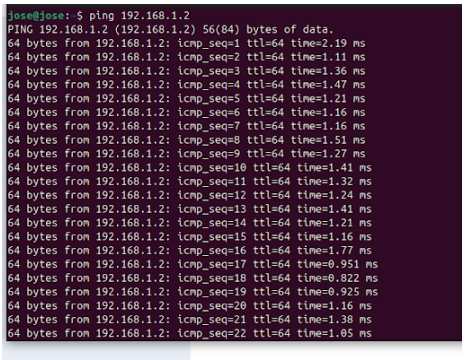
- **From Laptop A (master node):** Open a terminal and execute the following command:

```
ping 192.168.1.2
```

- **From Laptop B (worker node):** Open a terminal and execute the following command:

```
ping 192.168.1.1
```

If the configuration is correct, continuous replies without packet loss should be received, as shown in the following image, confirming communication between both systems.



Important note: If the `ping` test fails, the **Ubuntu Firewall** may be blocking the connection. For testing purposes, it can be temporarily disabled on both laptops by executing the following command:

```
sudo ufw disable
```

Once testing is complete, it is recommended to re-enable the firewall to maintain system security.

4.4 Passwordless SSH Access Configuration

To allow remote communication between the cluster nodes without requiring password entry for each connection, it is necessary to configure access using **SSH public keys**.

1. SSH Service Installation

Ensure that the SSH service is installed on both laptops. Execute the following commands in the terminal on **both systems**:

```
sudo apt update
sudo apt install openssh-server
```

2. Access Key Generation (Laptop A)

On **Laptop A** (master node, IP `192.168.1.1`), generate a security key that will be used to authenticate with Laptop B:

```
ssh-keygen -t rsa
```

When the system prompts for the key storage location or a passphrase, press **Enter** for each prompt (approximately three times), accepting the default values. At the end of the process, a graphical representation of the key will appear in the terminal.

3. Copying the Public Key to Laptop B

To authorize Laptop A to connect to Laptop B, execute the following command **from Laptop A**:

```
ssh-copy-id jose@192.168.1.2
```

The system will request the password for the user `jose` on **Laptop B**. Enter the password (characters will not be displayed) and press **Enter**.

4. Connection Verification (Functionality Test)

Finally, from **Laptop A**, attempt to access Laptop B via SSH:

```
ssh jose@192.168.1.2
```

If the connection is established directly without requesting a password, the configuration has been completed successfully.

To close the remote session and return to the Laptop A terminal, execute the following command:

```
exit
```

Figure 2: Example of a successful connection to Computer B.

4.5 NFS Configuration for Sharing the OpenFOAM Directory

The purpose of this configuration is to allow **Laptop B** to access the **OpenFOAM** directory located on **Laptop A** as if it were a local directory. This allows both nodes to work on the same cases, meshes, and libraries without duplicating files.

Step 1: Laptop A Configuration (NFS Server)

Open a terminal on **Laptop A** (master node) and execute the following commands to install the NFS server:

```
sudo apt update
sudo apt install nfs-kernel-server
```

Shared Directory Configuration

In this case, the entire **OpenFOAM** directory will be shared so that library and case paths match on both systems.

Edit the NFS exports configuration file:

```
sudo nano /etc/exports
```

Scroll to the end of the file and add the following line (verify that the path corresponds to your user):

```
/home/jose/OpenFOAM 192.168.1.2(rw,sync,no_subtree_check)
```

Save the changes by pressing `Ctrl+O`, then `Enter`, and exit the editor with `Ctrl+X`.

Applying the Changes

Finally, apply the new configuration and restart the NFS service for the changes to take effect:

```
sudo exportfs -ra
sudo systemctl restart nfs-kernel-server
```

4.6 Laptop B Configuration (NFS Client)

Step 2: Laptop B Configuration

Open a terminal on **Laptop B** (worker node) and execute the following commands to install the NFS client:

```
sudo apt update
sudo apt install nfs-common
```

Mount Directory Preparation

For **OpenFOAM** to function correctly, it is essential that the shared directory path be **identical** on both systems. If the directory already exists and contains files, they must be temporarily moved elsewhere, since mounting the shared directory will hide the local contents.

Ensure that the directory exists by executing:

```
mkdir -p /home/jose/OpenFOAM
```

Mounting the Shared Directory

Mount the shared directory from **Laptop A** to **Laptop B** using the following command:

```
sudo mount 192.168.1.1:/home/jose/OpenFOAM /home/jose/OpenFOAM
```

4.7 Shared File System Verification

Step 3: "Mirror" Test

The APP2 directory is the one used in this case and will be the working directory for both systems. To verify that the shared file system is functioning correctly, perform the following test:

- **On Laptop A:**

  Access the OpenFOAM case directory:

```
cd ~/OpenFOAM/jose-13/run/tutorials/fluid/APP2
```

  Create a test file:

```
touch connection_test.txt
```

- **On Laptop B:**

  Access the same path:

```
cd ~/OpenFOAM/jose-13/run/tutorials/fluid/APP2
```

List the directory contents:

```
ls
```

If the file `connection_test.txt` also appears on **Laptop B**, the shared file system has been configured correctly and both systems are working on the same file environment.

4.8 Parallel Execution Configuration in OpenFOAM

1. Domain Decomposition Configuration (`decomposeParDict`)

To allow **OpenFOAM** to run the simulation in parallel, it is necessary to specify how many parts the computational domain will be divided into.

Edit the corresponding configuration file from the case directory:

```
nano system/decomposeParDict
```

Modify the `numberOfSubdomains` parameter according to the total number of cores used in the cluster. For example, if **8 cores** from the MSI laptop and **8 cores** from the Lenovo laptop are used, the value should be:

```
numberOfSubdomains 16;
```

Also verify that the decomposition method is set to `hierarchical` or `scotch`, as both are suitable for distributed execution:

```
method scotch;
```

2. Machine File Creation (`hosts`)

The `hosts` file specifies which machines will participate in the parallel execution and how many cores each one contributes.

From the case directory, create and edit the file:

```
nano hosts
```

Add the IP addresses of the laptops along with the number of assigned cores. Assuming **8 cores per machine** are used, the file should contain:

```
192.168.1.1 slots=8
192.168.1.2 slots=8
```

Save the changes by pressing `Ctrl+O`, then `Enter`, and exit with `Ctrl+X`.

4.9 Parallel Simulation Execution

3. Simulation Execution

Once the domain decomposition and machine file are configured, the simulation can be executed in parallel from **Laptop A** (master node).

Execute the following commands in order from the OpenFOAM case directory:

- **Mesh preparation:**

```
blockMesh
```

- **Domain decomposition:** This command will generate directories named `processor0`, `processor1`, etc.

```
decomposePar
```

- **Launching the cluster simulation:**

```
mpirun -np 16 foamRun -parallel
```

4. Resource Usage Verification on Laptop B

To confirm that **Laptop B** is actively participating in the simulation, perform the following check while the process is running:

- On **Laptop B**, open a terminal and execute:

```
htop
```

Multiple OpenFOAM solver processes (for example, `simpleFoam`) should be observed using approximately **100%** of the assigned cores, confirming that the computation is being correctly distributed across the cluster nodes.

4.10 Troubleshooting Common Errors in Parallel Execution

During OpenFOAM execution on the cluster, the following error may appear when launching the parallel simulation:

```
mpirun --hostfile hosts -np 16 foamRun -parallel
Authorization required, but no authorization protocol specified
...
mpirun was unable to find the specified executable file
Node: 192.168.1.2
Executable: /opt/openfoam13/platforms/linux64GccDPInt32Opt/bin/foamRun
8 total processes failed to start
```

This error mainly indicates two issues:

- Lack of authorization to access the graphical interface (X11).
- **Laptop B** is not correctly loading the **OpenFOAM** environment and therefore cannot find the `foamRun` executable.

1. `.bashrc` File Configuration (Laptop B)

This is the most important step. Ubuntu includes a protection mechanism in the `.bashrc` file that prevents environment variables from being loaded when the access is non-interactive (such as SSH connections used by `mpirun`).

On **Laptop B**, edit the file:

```
nano ~/.bashrc
```

Locate the following lines near the beginning of the file (usually within the first 5 to 10 lines):

```
# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac
```

You have two valid options:

- Cut these lines and paste them at the end of the file.
- Or comment out the line `*) return;;` by adding a `#`.

Also ensure that the following line is present at the end of the file:

```
source /opt/openfoam13/etc/bashrc
```

Save the changes and close the editor.

2. Allow Graphical Access (Laptop A)

The `Authorization required` message appears because the remote process attempts to interact with the graphical server (X11) and is blocked.

On **Laptop A**, execute the following command:

```
xhost +
```

This command temporarily allows external processes to interact with the local display.

3. Remote Execution Verification

Before launching the full simulation, it is recommended to verify that **Laptop A** can correctly execute OpenFOAM commands on **Laptop B**.

From **Laptop A**, execute:

```
ssh 192.168.1.2 "foamRun -help"
```

- If the OpenFOAM help is displayed, the environment is correctly configured.
- If the message `command not found` appears, the environment is still not being loaded correctly on Laptop B.

4. Final Simulation Launch

Once correct remote execution is verified, access the case directory on **Laptop A** and launch the parallel simulation again:

```
cd ~/OpenFOAM/jose-13/run/tutorials/fluid/APP2
mpirun --hostfile hosts -np 16 foamRun -parallel
```

If the simulation starts without errors and both systems show CPU load, the cluster is functioning correctly.

5 Disconnection and Future Use

5.1 Cluster Disconnection and Reuse

System Disconnection (After Simulation Completion)

Once the simulation has finished, it is important to properly disconnect the shared file system to avoid errors or data loss.

- **Unmount the shared directory on Laptop B:**

  Before disconnecting the Ethernet cable, Laptop B must stop accessing files located on Laptop A.

```
sudo umount -l /home/jose/OpenFOAM
```

If the error `device is busy` appears, ensure that all terminals or processes using that directory on Laptop B are closed.

- **Physical cable disconnection:**

  Once the file system has been properly unmounted, the Ethernet cable can be safely disconnected.

Cluster Reuse in Future Sessions

Since the main configuration has already been completed, reusing the cluster at a later time is a fast and simple process:

1. Connect the Ethernet cable between both laptops.
2. Verify that both machines retain the configured IP addresses:
   - Laptop A: `192.168.1.1`
   - Laptop B: `192.168.1.2`
3. Mount the shared directory again on **Laptop B** by executing:

```
sudo mount 192.168.1.1:/home/jose/OpenFOAM /home/jose/OpenFOAM
```

4. Perform a quick test to verify that files are visible:

```
ls ~/OpenFOAM/jose-13/...
```

   If the files appear correctly, the shared file system is active.

5. Finally, run the parallel simulation again from **Laptop A** using the `mpirun` command.