High-Performance Mathematics

Building the `steffe` cluster

Progetto Speciale per la Didattica 2023/24

Fabio Durastante (L4)

May 8, 2024

Dipartimento
di Matematica
Università di Pisa

Table of Contents

1 Our cluster

The `steffe` cluster.

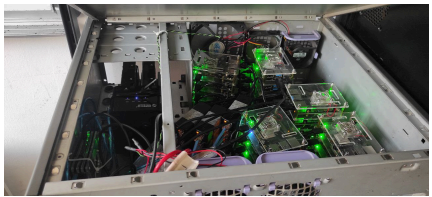
1 Our cluster

Now that we know how to put together some parallel program examples, we can **focus on the machine** we want to run them on.

The steffe cluster.

1 Our cluster

Now that we know how to put together some parallel program examples, we can **focus on the machine** we want to run them on.

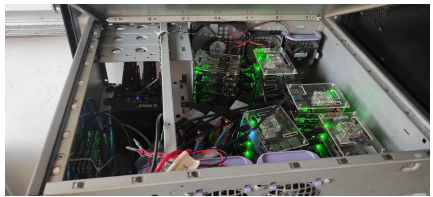


- Last year we assembled this parallel computer prototype,

The `steffe` cluster.

1 Our cluster

Now that we know how to put together some parallel program examples, we can **focus on the machine** we want to run them on.



- Last year we assembled this parallel computer prototype,
- today we want to expand it by increasing the number of nodes.

The `steffe` cluster.

1 Our cluster

Now that we know how to put together some parallel program examples, we can **focus on the machine** we want to run them on.

- Last year we assembled this parallel computer prototype,
- today we want to expand it by increasing the number of nodes.
- But first let's understand exactly **how it is done** and **how we should act**.

Clusters

Cluster

"Clusters are an ensemble of **off-the-shelf computers** integrated by an **interconnection network** and operating within a single administrative domain and usually within a single machine room. Commodity clusters employ **commercially available networks** (e.g., **Ethernet**, Myrinet) as opposed to custom networks (e.g., IBM SP-2). *Beowulf-class* clusters incorporate mass-market PC technology for their compute nodes to achieve the best price/performance."

Beowulf Cluster Computing with Linux

Nodes and Network

1 Our cluster

A **node** is responsible for all activities and capabilities associated with executing an application program and supporting a sophisticated software environment:

1. instruction execution;
2. high-speed temporary information storage;
3. high-capacity persistent information storage, and
4. **communication** with the external environment, including **other nodes**.

A **network** is a combination of *physical transport* and *control mechanisms* associated with a **layered hierarchy** of **message** encapsulation.

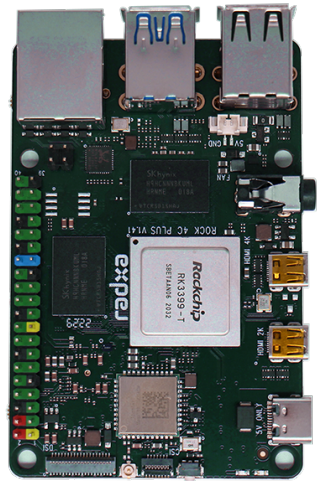
Building our machine

1 Our cluster

Let's start with **the nodes**:

Radxa ROCK 4C+

CPU	Arm® big.LITTLE™ technology: Dual Cortex® A72 frequency 1.5GHz and a Quad Cortex A53 frequency 1.0GHz,
GPU	Arm Mali™ T860MP4 GPU,
RAM	Dual channel 4GB 64bit LPDDR4,
LAN	1x Gigabit Ethernet port,
HD	1x micro SD card slot.



Building our machine

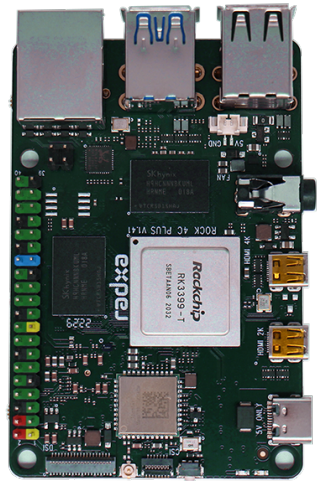
1 Our cluster

Let's start with **the nodes**:

Radxa ROCK 4C+

CPU	Arm® big.LITTLE™ technology: Dual Cortex® A72 frequency 1.5GHz and a Quad Cortex A53 frequency 1.0GHz,
GPU	Arm Mali™ T860MP4 GPU,
RAM	Dual channel 4GB 64bit LPDDR4,
LAN	1x Gigabit Ethernet port,
HD	1x micro SD card slot.

Why two processors?

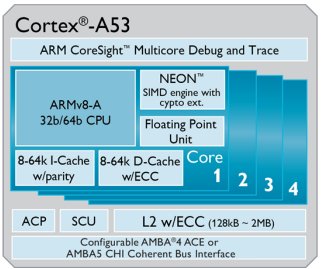


The ARM big.LITTLE architecture

The idea

ARM big.LITTLE is a **heterogeneous computing architecture** coupling *relatively* energy-saving and slower processor cores (LITTLE) with *relatively* more powerful and power-hungry ones (big).

- Only one ”side” or the other will be active at once,
- All cores have access to the same memory regions, so workloads can be swapped between *big* and *LITTLE* cores on the fly.



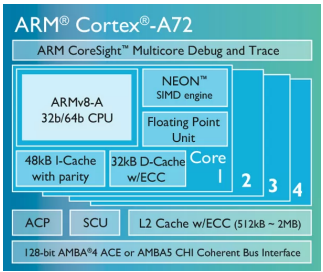
LITTLE

The ARM big.LITTLE architecture

The idea

ARM big.LITTLE is a **heterogeneous computing architecture** coupling *relatively* energy-saving and slower processor cores (LITTLE) with *relatively* more powerful and power-hungry ones (big).

- Only one ”side” or the other will be active at once,
- All cores have access to the same memory regions, so workloads can be swapped between *big* and *LITTLE* cores on the fly.



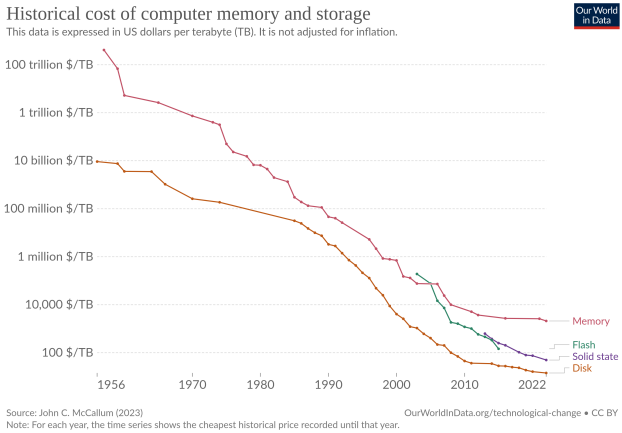
big

The 4GB 64bit LPDDR4 Memory

1 Our cluster

The LPDDR4 acronym stands for **L**ow-**P**ower **D**ouble **D**ata **R**ate **4** dynamic RAM.

Along with processor speed (*Moore's Law*), memory capacity has grown at a phenomenal rate, quadrupling in size approximately every three years.



HD: Storage

1 Our cluster

Each node will use Kingston 64 GB Micro SD (SDHC Class 10) SDCS/32GB with

- OS - Local to the node.
- Home - Where the users files and program will reside, it will be a *shared file system*.

RAID and NFS

1 Our cluster

- A **RAID** (**R**edundant **A**rray of **I**ndependent **D**isks) configuration is a method of storing data across multiple hard drives to improve performance, reliability, or both. It combines *multiple physical disk drives into a single logical unit*, typically offering fault tolerance by mirroring or striping data across the drives.

RAID and NFS

1 Our cluster

- A **RAID** (**R**edundant **A**rray of **I**ndependent **D**isks) configuration is a method of storing data across multiple hard drives to improve performance, reliability, or both. It combines *multiple physical disk drives into a single logical unit*, typically offering fault tolerance by mirroring or striping data across the drives.
- **NFS** (**N**etwork **F**ile **S**ystem) allows multiple remote systems to access shared files over a network, enabling seamless collaboration and centralized data management.

RAID and NFS

- A **RAID** (**R**edundant **A**rray of **I**ndependent **D**isks) configuration is a method of storing data across multiple hard drives to improve performance, reliability, or both. It combines *multiple physical disk drives into a single logical unit*, typically offering fault tolerance by mirroring or striping data across the drives.
- **NFS** (**N**etwork **F**ile **S**ystem) allows multiple remote systems to access shared files over a network, enabling seamless collaboration and centralized data management.

Our configuration

Integrating RAID with NFS enhances data availability and reliability, as RAID's redundancy features protect against disk failures, while NFS facilitates easy access to files across the network.

Our configuration is made of

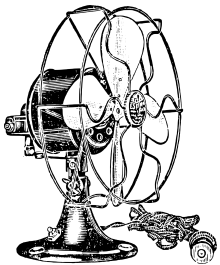
- 1 ORICO Dual Bay docking station with space for two disks that are connected in RAID with replica 1.
- 2 Western Digital Red WD40EFAX - 4 TB Sata 600 256 MB 5400 rpm disks.

Power requirements and cooling

The Radxa ROCK 4C+ is **powered with a 5V source**.

- USB C 5V/3A,
- 5V Power applied to the GPIO PIN 2 & 4.

The recommended power source capacity is at least 5V/3A.



"The Radxa ROCK 4C+ will operate perfectly well without any additional cooling and is ***designed for sprint performance*** *- expecting a light use case on average then ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature (sic.) at full performance, further cooling may be needed."*

OS: What flavor of Linux?

The Radxa ROCK 4C+ has **Debian/Ubuntu Linux support**, images can be obtained from:

https://wiki.radxa.com/Rockpi4/downloads

- Ubuntu 20.04.6 LTS (Focal Fossa),
- Server install image ⇒ No GUI!
- Again, *why Linux*? Linux is the unchallenged champion for building compute engines with commodity parts: www.top500.org.

OS Setup

≈ 10 Gb for the OS:

- Compilers: GCC Suite v10.3.0,
- MPI: OpenMPI v4.0.3,
- OpenBLAS v0.3.8,
- Valgrind v3.15.0.
- NFS

≈ 4 Tb of shared filesystem for the homes.

OS Setup

1 Our cluster

≈ 10 Gb for the OS:

- Compilers: GCC Suite v10.3.0,
- MPI: OpenMPI v4.0.3,
- OpenBLAS v0.3.8,
- Valgrind v3.15.0.
- NFS

≈ 4 Tb of shared filesystem for the homes.
≈ 40 Gb of experimental shared filesystem (doesn't perform very well)

We will use the **Gluster** **F**ile **S**ystem (www.gluster.org)

"Gluster is a distributed scale-out filesystem that allows rapid provisioning of additional storage based on your storage consumption needs. It incorporates automatic failover as a primary feature. All of this is accomplished without a centralized metadata server."

OS Setup

1 Our cluster

≈ 10 Gb for the OS:

- Compilers: GCC Suite v10.3.0,
- MPI: OpenMPI v4.0.3,
- OpenBLAS v0.3.8,
- Valgrind v3.15.0.
- NFS

≈ 4 Tb of shared filesystem for the homes.
≈ 40 Gb of experimental shared filesystem (doesn't perform very well)

We will use the **Gluster** **F**ile **S**ystem (www.gluster.org)

"Gluster is a distributed scale-out filesystem that allows rapid provisioning of additional storage based on your storage consumption needs. It incorporates automatic failover as a primary feature. All of this is accomplished without a centralized metadata server."

All the **nodes** need to have the **same configuration** and **software**!

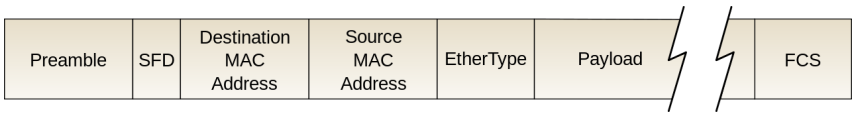
Table of Contents

2 Network architecture

The LAN Network

2 Network architecture

The Ethernet message packet comprises a sequence of **multibit fields**, one of which is variable length. The fields include a **combination** of **network control information** and **data payload**.

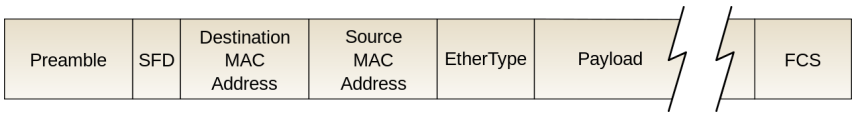


- TCP/IP is the *de facto* standard network communication protocol:
  - The **destination** of an Internet Protocol packet is specified by a 32-bit IP address, e.g., 192.168.1.2.

The LAN Network

2 Network architecture

The Ethernet message packet comprises a sequence of **multibit fields**, one of which is variable length. The fields include a **combination** of **network control information** and **data payload**.



- TCP/IP is the *de facto* standard network communication protocol:
  - The **destination** of an Internet Protocol packet is specified by a 32-bit IP address, e.g., 192.168.1.2.
- Before assigning IP addresses to our nodes, designing the network topology, and booting all the machine, we need to **decide how the system will be accessed**: how a user can log in to a system and use the machine?

How do we access the system?

2 Network architecture

The Standalone System: unattached to any external networks, the user need to be in the same room of the machine.

The Universally Accessible Machine: every node is accessible from the entire Internet.

The Guarded Beowulf: reserved IPs to all internal nodes, and single front-end with an IP address accessible from outside.

How do we access the system?

2 Network architecture

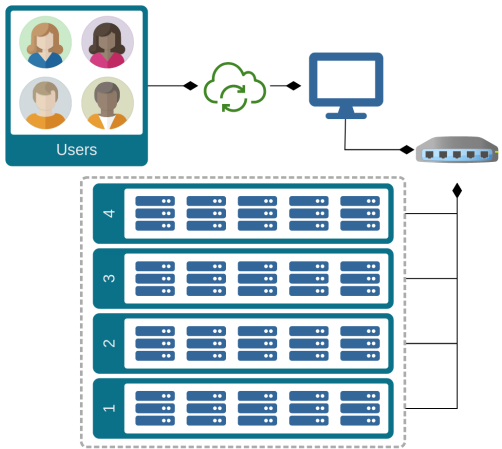
The Standalone System: unattached to any external networks, the user need to be in the same room of the machine.

The Universally Accessible Machine: every node is accessible from the entire Internet.

The Guarded Beowulf: reserved IPs to all internal nodes, and single front-end with an IP address accessible from outside.

Our network architecture

2 Network architecture



We will build a **guarded Beowulf** with an access node, where the user will log-in.

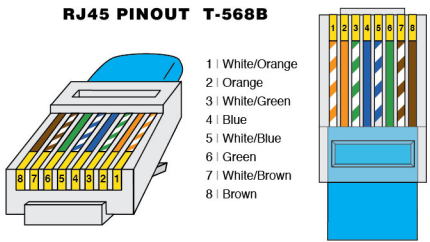
All the **nodes** will be connected to the **switch** via Ethernet connection, and will be powered through it.
The **access node** will be configured on the IP:

```
IP: 131.114.10.121
Gateway: 131.114.10.1
```

and is reachable at the address `steffe.cs.dm.unipi.it`.

Ethernet Cables

2 Network architecture



1. Cut the cable to the length needed,
2. Strip back the cable jacket approximately 2.5 cm,
3. Use the **568-B wiring scheme** on both ends for a standard patch cable.

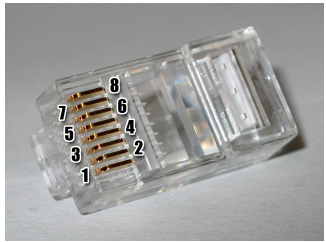
The **maximum length** for a cable segment is **100 meters**. If longer runs are required repeaters or switches are necessary.

Ethernet Cables

Pin	Pair	Wire	Color
1	2	1	
2	2	2	
3	3	1	
4	1	2	
5	1	1	
6	3	2	
7	4	1	
8	4	2	

ANSI/TIA-568 Standard

1. Cut the cable to the length needed,
2. Strip back the cable jacket approximately 2.5 cm,
3. Use the **568-B wiring scheme** on both ends for a standard patch cable.



Power over Ethernet (PoE)

2 Network architecture

Power over Ethernet, or PoE, describes any of *several standards* or ad hoc systems that **pass electric power** *along* with **data** on **twisted-pair Ethernet cabling**.

- 24 10/100/1000Mbps RJ45 PoE+ Ports Switch with 2 SFP Slots,
- **PoE power budget** is up to **250 W** (in *laboratory environment*).

Compliant with IEEE802.3af/at
Isolation: 2.5kV
Power Input: DC44 57V
Power Output: DC5V 3A

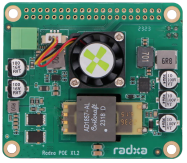


Table of Contents

3 Extending the configuration

Extending the configuration

To **extend our setup** this year we decided to:

1. find a new housing for the cluster,

Extending the configuration

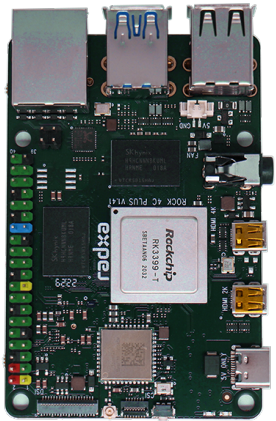
To **extend our setup** this year we decided to:

1. find a new housing for the cluster,
2. specifically for the nodes,

Extending the configuration

To **extend our setup** this year we decided to:

1. find a new housing for the cluster,
2. specifically for the nodes,
3. add **other 15 nodes**,



Extending the configuration

To **extend our setup** this year we decided to:

1. find a new housing for the cluster,
2. specifically for the nodes,
3. add **other 15 nodes**,
4. **but** this will require **another switch** to satisfy both the **network** and **power issues**.

Extending the configuration

To **extend our setup** this year we decided to:

1. find a new housing for the cluster,
2. specifically for the nodes,
3. add **other 15 nodes**,
4. **but** this will require **another switch** to satisfy both the **network** and **power issues**.

What we have already done

We have already moved the cluster from the old case to the new one ($\approx$ 6 h of work). Mounted the supports and inserted the new switch and power supply.

Table of Contents

4 Things to do today

Things to do today

1. Install the OS on the new nodes and configure them like the others,
2. Mount the POE hats on the nodes and place them in the rack slots,
3. Prepare the ethernet cables to connect the nodes to the new switch,
4. Configure the new switch.
5. Create accounts to enable you to use the machine.

Things to do today

1. Install the OS on the new nodes and configure them like the others,
2. Mount the POE hats on the nodes and place them in the rack slots,
3. Prepare the ethernet cables to connect the nodes to the new switch,
4. Configure the new switch.
5. Create accounts to enable you to use the machine.

A command we will use a lot:

The `scp` command (*Secure Copy*) is used to securely transfer files between a local and remote host over a network. Its syntax is: `scp [options] [source] [destination]`

Things to do today

1. Install the OS on the new nodes and configure them like the others,
2. Mount the POE hats on the nodes and place them in the rack slots,
3. Prepare the ethernet cables to connect the nodes to the new switch,
4. Configure the new switch.
5. Create accounts to enable you to use the machine.

A command we will use a lot:

The `scp` command (*Secure Copy*) is used to securely transfer files between a local and remote host over a network. Its syntax is: `scp [options] [source] [destination]`
Where:

- `[source]`: Specifies the file or directory you want to copy from the local host or remote host.

Things to do today

4 Things to do today

1. Install the OS on the new nodes and configure them like the others,
2. Mount the POE hats on the nodes and place them in the rack slots,
3. Prepare the ethernet cables to connect the nodes to the new switch,
4. Configure the new switch.
5. Create accounts to enable you to use the machine.

A command we will use a lot:

The `scp` command (*Secure Copy*) is used to securely transfer files between a local and remote host over a network. Its syntax is: `scp [options] [source] [destination]`
Where:

- `[destination]`: Specifies the destination path where you want to copy the file or directory. This can be a local path or a remote path in the format: `[user@]host:[path]`.

Things to do today

1. Install the OS on the new nodes and configure them like the others,
2. Mount the POE hats on the nodes and place them in the rack slots,
3. Prepare the ethernet cables to connect the nodes to the new switch,
4. Configure the new switch.
5. Create accounts to enable you to use the machine.

A command we will use a lot:

The `scp` command (*Secure Copy*) is used to securely transfer files between a local and remote host over a network. Its syntax is: `scp [options] [source] [destination]`
Options:

- `-r`: Recursively copy entire directories.
- `-P port`: Specifies the port to connect to on the remote host.
- `-v`: Verbose mode, provides more detailed output for debugging.

Install the OS

4 Things to do today

Since all the nodes have to be equal we move in **two steps**:

1. Install and configure the first node,
2. Make copies of the same SD for all the other nodes *or* execute same configuration commands everywhere.

Install the OS

4 Things to do today

Since all the nodes have to be equal we move in **two steps**:

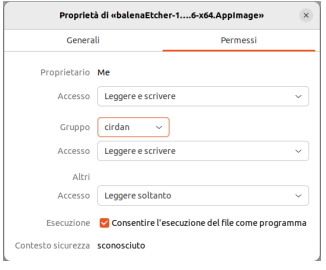
1. Install and configure the first node,
2. Make copies of the same SD for all the other nodes *or* execute same configuration commands everywhere.

Let's start with **step 1**, we go to wiki.radxa.com/Rockpi4/downloads and download

- Etcher - A user friendly Image Writer,
- Ubuntu 20 Server(Linux 4.4).

We use Etcher to write the system image on the SD, on Linux this will be a `.appimage` file, so first of all we have to *make it executable*.

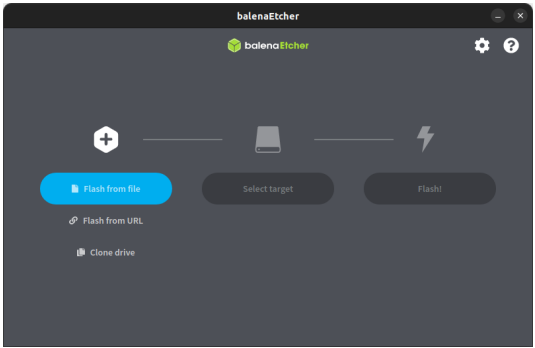
Remark: **we need *root privileges*!**



Install the OS

4 Things to do today

Etcher is intuitive enough to use:



We select the Ubuntu 20 Server(Linux 4.4) image we have downloaded, then select the SD as target and the we flash the SD.

Align installed packages between old and new nodes

4 Things to do today

On one of the old nodes (but **not** the login node) run:

```
dpkg --get-selections > list.txt
```

and the use `scp` to copy the file `list.txt` on the new node.

On the new node run

```
sudo apt-get update
sudo apt-get upgrade
dpkg --clear-selections
sudo dpkg --set-selections < list.txt
```

Package versions

This would probably be a good moment to **update all packages across the cluster**.

Copying SD Card to Hard Disk

4 Things to do today

1. Insert the SD card into your computer's SD card slot.
2. Open a terminal window.
3. Identify the device name of your SD card using the command:
   - `lsblk`
4. Unmount the SD card if it is automatically mounted:
   - `sudo umount /dev/sdX`
5. Copy the SD card to the hard disk using the `dd` command:
   - `sudo dd if=/dev/sdX of=/path/to/destination/image.img bs=4M`
6. Wait for the process to complete.
7. Safely remove the SD card.

Copying Back to Another SD Card

4 Things to do today

1. Insert the destination SD card into your computer's SD card slot.
2. Open a terminal window.
3. Identify the device name of the destination SD card using the command:
   - `lsblk`
4. Unmount the destination SD card if it is automatically mounted:
   - `sudo umount /dev/sdY`
5. Copy the image file back to the SD card using the `dd` command:
   - `sudo dd if=/path/to/source/image.img of=/dev/sdY bs=4M`
6. Wait for the process to complete.
7. Safely remove the destination SD card.

Change the system names and configurations

4 Things to do today

At this point the **node is just a clone**, that is, it has the wrong name, address and other data. . .

To change the system name (hostname) in Linux:

1. Open a terminal window.
2. Check the current system name by running:
   - `hostname`
3. To change the system name temporarily (until the next reboot), use:
   - `sudo hostname new-name`
4. To change the system name permanently:
   - Edit the `/etc/hostname` file and replace the current name with the new name.
   - Edit the `/etc/hosts` file and replace any occurrences of the old hostname with the new hostname.
   - Reboot your system for the changes to take effect.

Homes and users

4 Things to do today

To **be sure that the** RAID **disk** with NFS **are mounted** we should ensure that the `/etc/fstab` file contains

```
steffe0:/data /mnt/data glusterfs defaults,_netdev,nofail 0 0
steffe0:/mnt/raid/ /mnt/raid nfs auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Homes and users

4 Things to do today

To **be sure that the** RAID **disk** with NFS **are mounted** we should ensure that the `/etc/fstab` file contains

```
steffe0:/data /mnt/data glusterfs defaults,_netdev,nofail 0 0
steffe0:/mnt/raid/ /mnt/raid nfs auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

To **copy all users from one Linux system to another**:

1. Open a terminal on the source system.
2. Export user information to a file using the `getent` command:
   - `getent passwd > users.txt`
3. Copy the generated `users.txt` file to the destination system using a secure method such as `scp` or `rsync`.
4. Open a terminal on the destination system.
5. Import users from the file:
   - `sudo cat users.txt | sudo chpasswd`
6. Verify that the users have been copied successfully:
   - `getent passwd`