

# Introduction to High Performance Computing



*Forschung und Wissenschaftliche Informationsversorgung*  
*IT.SERVICES*

# What is High Performance Computing?

A high performance computing system is any computing system that performs significantly more operations per second than an average desktop computer.

A high performance computing system is any computing system that performs significantly more operations per second than an average desktop computer.

High end personal computer:

AMD 6900HX	3.7 TFlops
Playstation 5	10.0 TFlops
RTX 4090	100.0 TFlops

TFlops =  $10^{12}$  Flops = 1,000,000,000,000 Floating point operations/s

A high performance computing system is any computing system that performs significantly more operations per second than an average desktop computer.

## High end personal computer:

AMD 6900HX	3.7 TFlops
Playstation 5	10.0 TFlops
RTX 4090	100.0 TFlops

## High end supercomputers:

El Capitan	1742.00 PFlops
Frontier	1206.00 PFlops
Aurora	1012.00 PFlops

TFlops =  $10^{12}$  Flops = 1,000,000,000,000 Floating point operations/s

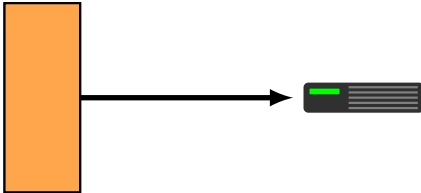
PFlops =  $10^{15}$  Flops = 1,000,000,000,000,000 Floating point operations/s



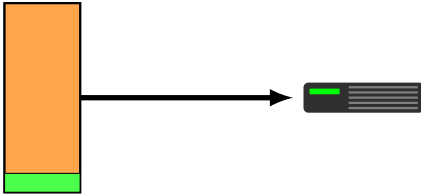
- The performance of the fastest computers grows exponentially.<sup>1</sup>

<sup>1</sup><https://www.top500.org/>

- Solving a problem on one computer takes time.

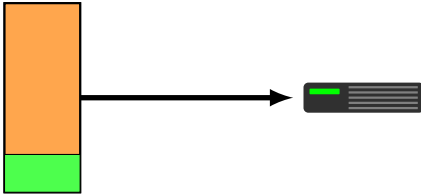


- Solving a problem on one computer takes time.

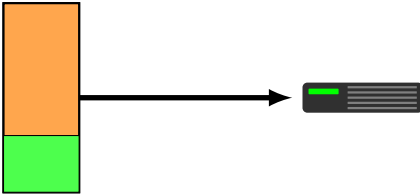




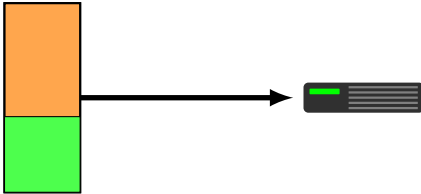
- Solving a problem on one computer takes time.



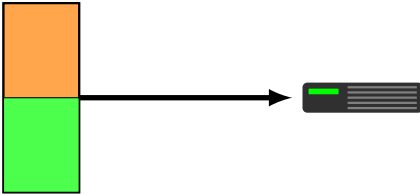
- Solving a problem on one computer takes time.



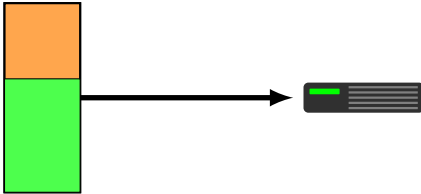
- Solving a problem on one computer takes time.



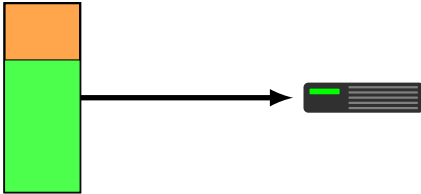
- Solving a problem on one computer takes time.



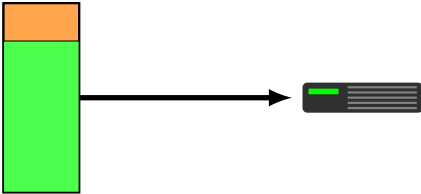
- Solving a problem on one computer takes time.



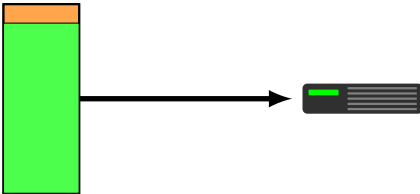
- Solving a problem on one computer takes time.



- Solving a problem on one computer takes time.

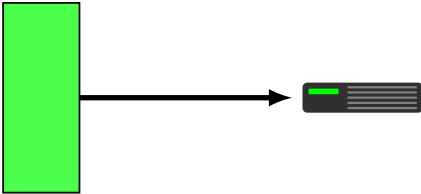


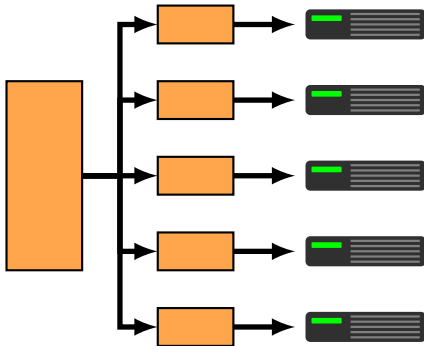
- Solving a problem on one computer takes time.



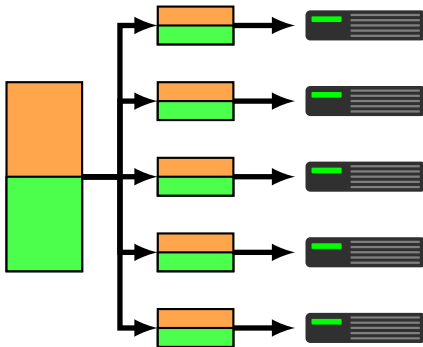


- Solving a problem on one computer takes time.

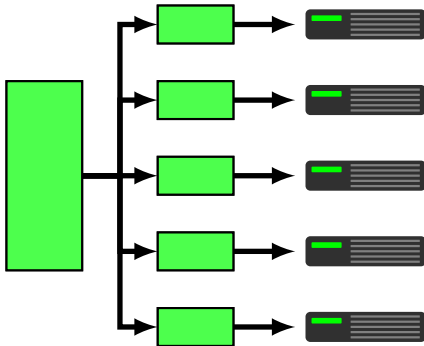




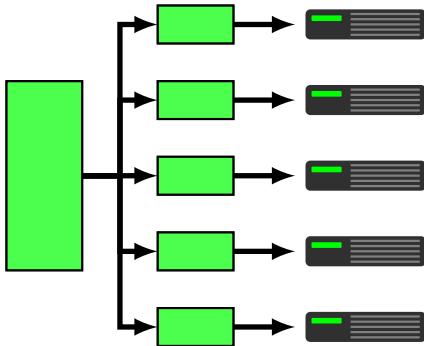
- Solving a problem on one computer takes time.
- Split the problem in several smaller problems and solve each in parallel.



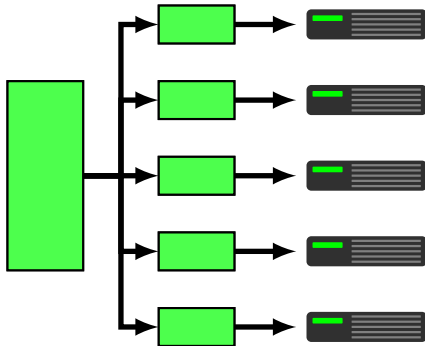
- Solving a problem on one computer takes time.
- Split the problem in several smaller problems and solve each in parallel.



- Solving a problem on one computer takes time.
- Split the problem in several smaller problems and solve each in parallel.
- This can lead to a significant speedup.



- Solving a problem on one computer takes time.
- Split the problem in several smaller problems and solve each in parallel.
- This can lead to a significant speedup.
- The collection of computers (cluster) has a higher combined performance.



- Solving a problem on one computer takes time.
- Split the problem in several smaller problems and solve each in parallel.
- This can lead to a significant speedup.
- The collection of computers (cluster) has a higher combined performance.

Note that software has to be specifically parallelized to run efficiently on HPC-Systems, which is not trivial!

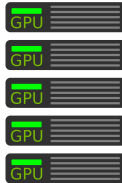
HPC clusters consist of:

- Multiple computers (nodes).

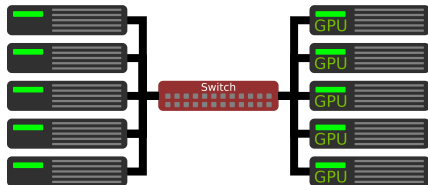


HPC clusters consist of:

- Multiple computers (nodes).
- (Sometimes) Multiple nodes with accelerators (e.g. GPUs).

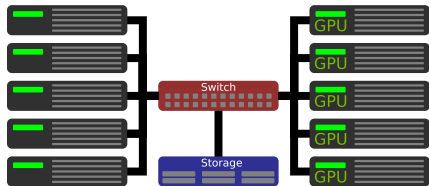






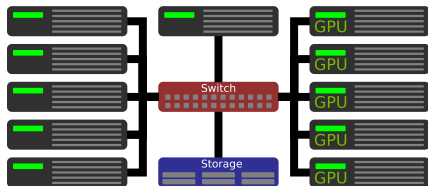
HPC clusters consist of:

- Multiple computers (nodes).
- (Sometimes) Multiple nodes with accelerators (e.g. GPUs).
- Interconnect ( $\text{Lat} \approx 1.2 \mu\text{s}$ ,  $\text{BW} \approx 100\text{-}200 \text{ Gbit/s}$ ).



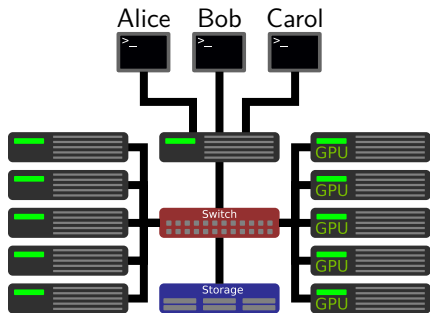
HPC clusters consist of:

- Multiple computers (nodes).
- (Sometimes) Multiple nodes with accelerators (e.g. GPUs).
- Interconnect ( $\text{Lat} \approx 1.2 \mu\text{s}$ ,  $\text{BW} \approx 100\text{-}200 \text{ Gbit/s}$ ).
- Shared mass storage.



HPC clusters consist of:

- Multiple computers (nodes).
- (Sometimes) Multiple nodes with accelerators (e.g. GPUs).
- Interconnect ( $\text{Lat} \approx 1.2 \mu\text{s}$ ,  $\text{BW} \approx 100\text{-}200 \text{ Gbit/s}$ ).
- Shared mass storage.
- Login node for users to connect to.



HPC clusters consist of:

- Multiple computers (nodes).
- (Sometimes) Multiple nodes with accelerators (e.g. GPUs).
- Interconnect ( $\text{Lat} \approx 1.2 \mu\text{s}$ ,  $\text{BW} \approx 100\text{-}200 \text{ Gbit/s}$ ).
- Shared mass storage.
- Login node for users to connect to.

- HPC systems demand a lot of electrical power and cooling

- HPC systems demand a lot of electrical power and cooling  
⇒ Specially cooled serverrooms necessary

- HPC systems demand a lot of electrical power and cooling  
⇒ Specially cooled serverrooms necessary
- Serverrooms host delicate hardware and sensitive data

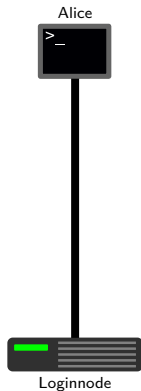
- HPC systems demand a lot of electrical power and cooling  
⇒ Specially cooled serverrooms necessary
- Serverrooms host delicate hardware and sensitive data  
⇒ Access to serverrooms is restricted



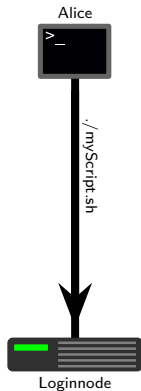
- HPC systems demand a lot of electrical power and cooling  
⇒ Specially cooled serverrooms necessary
- Serverrooms host delicate hardware and sensitive data  
⇒ Access to serverrooms is restricted
- HPC systems have hundreds of simultaneous users

- HPC systems demand a lot of electrical power and cooling  
⇒ Specially cooled serverrooms necessary
- Serverrooms host delicate hardware and sensitive data  
⇒ Access to serverrooms is restricted
- HPC systems have hundreds of simultaneous users  
⇒ Not enough physical space for local user access

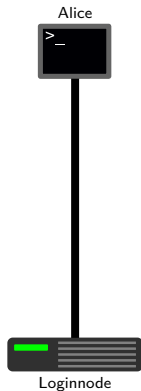
# How to Access an HPC Cluster?



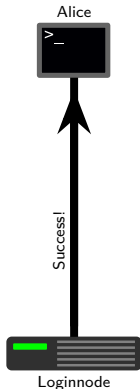
- Linux allows to connect terminals from remote.



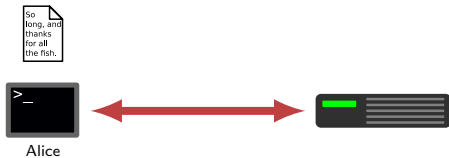
- Linux allows to connect terminals from remote.
- Commands are send over the network to the Cluster



- Linux allows to connect terminals from remote.
- Commands are send over the network to the Cluster
- executed on the cluster

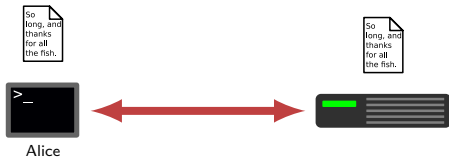


- Linux allows to connect terminals from remote.
- Commands are send over the network to the Cluster
- executed on the cluster
- and output is send back to the users terminal.

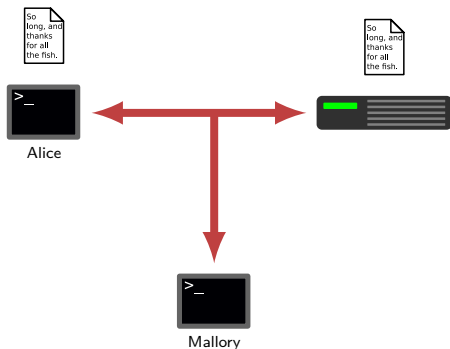


- Alice copies sensitive data to the cluster.

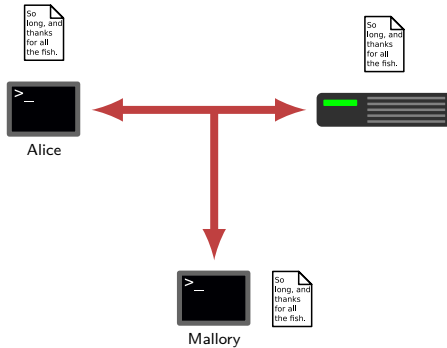




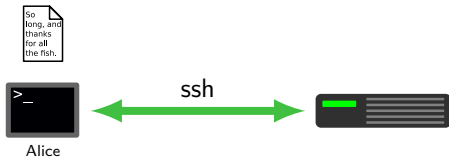
- Alice copies sensitive data to the cluster.



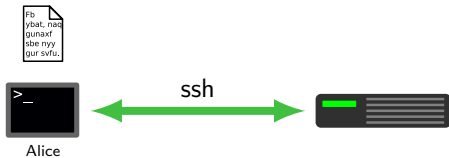
- Alice copies sensitive data to the cluster.
- Mallory snoops the connection to get a copy of the data.



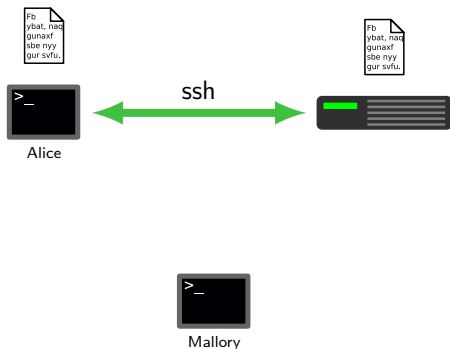
- Alice copies sensitive data to the cluster.
- Mallory snoops the connection to get a copy of the data.



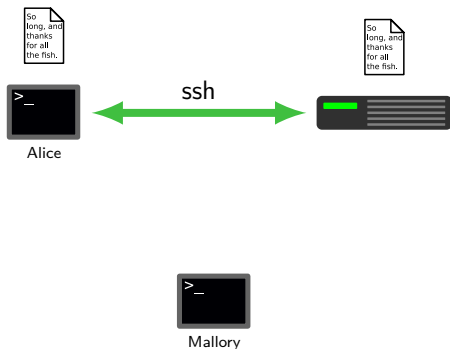
- Alice copies sensitive data to the cluster via a ssh.



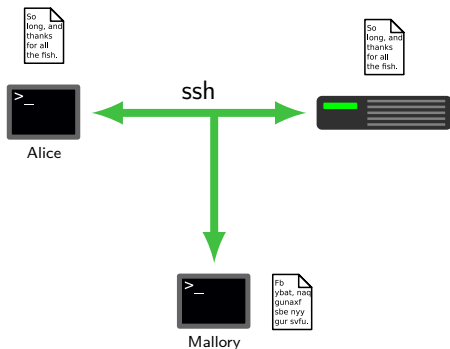
- Alice copies sensitive data to the cluster via a ssh.
- ssh encrypts the file on Alice' computer.



- Alice copies sensitive data to the cluster via a ssh.
- ssh encrypts the file on Alice' computer.
- ssh copies the file to the cluster.



- Alice copies sensitive data to the cluster via a ssh.
- ssh encrypts the file on Alice' computer.
- ssh copies the file to the cluster.
- ssh decrypts the file on the cluster.



- Alice copies sensitive data to the cluster via a ssh.
- ssh encrypts the file on Alice' computer.
- ssh copies the file to the cluster.
- ssh decrypts the file on the cluster.
- Mallory snoops the connection to get a copy of the data.
- Mallory cannot decrypt the file.



An ssh connection can be established with  
`ssh username@ip-address`

### Terminal

```
alice@laptop:$ ssh alice@4.669.201.609
Password:
alice@hpc:$
```

Alternatively:

```
ssh username@hostname
```

## Terminal

```
alice@laptop:$ ssh alice@hpc.rub.de  
Password:  
alice@hpc:$
```

For convenience a connection can be specified in  
~/.ssh/config

### Terminal

```
alice@laptop:~$ cat ~/.ssh/config
Host hpc-rub
  User alice
  Hostname hpc.rub.de
alice@laptop:~$ ssh hpc-rub
Password:
alice@hpc:~$
```

1. Connect to the RUB-Testcluster
  - Username and password are on a paper slip
  - The IP is 134.147.41.133

## 1. Connect to the RUB-Testcluster

- Username and password are on a paper slip
- The IP is 134.147.41.133

### Terminal

```
alice@laptop:$ ssh testuser@134.147.41.133
Password:
alice@hpc:$
```



So long, and  
thanks  
for all  
the fish.



Alice



So long, and  
thanks  
for all  
the fish.



- Alice copied sensitive data to the cluster via a ssh.



Mallory



So long, and  
thanks  
for all  
the fish.



Alice



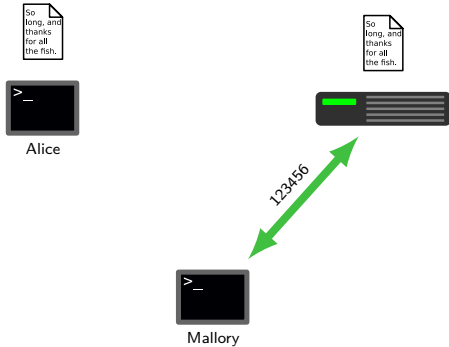
So long, and  
thanks  
for all  
the fish.



- Alice copied sensitive data to the cluster via a ssh.
- Alice used a password for authentication.

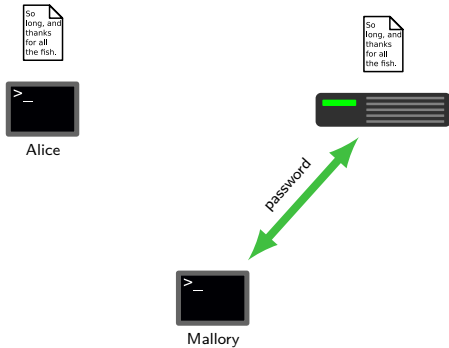


Mallory

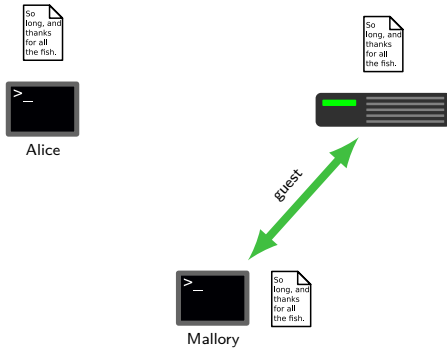


- Alice copied sensitive data to the cluster via a ssh.
- Alice used a password for authentication.
- Mallory tries to guess Alice' weak password.





- Alice copied sensitive data to the cluster via a ssh.
- Alice used a password for authentication.
- Mallory tries to guess Alice' weak password.



- Alice copied sensitive data to the cluster via a ssh.
- Alice used a password for authentication.
- Mallory tries to guess Alice' weak password.
- Mallory obtains an unencrypted copy of the data.



Alice



- Alice generates an "elliptic-curve cryptography key-pair for asymmetric authentication"
  - Public key (green) can only encrypt!
  - Private key (red) can only decrypt!



Alice



- Alice generates an "elliptic-curve cryptography key-pair for asymmetric authentication"
  - Public key (green) can only encrypt!
  - Private key (red) can only decrypt!
- Alice's public key is registered on the HPC cluster.

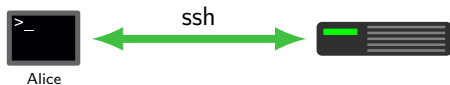


Alice



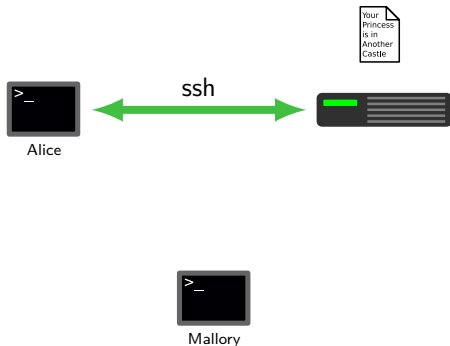
Mallory

- Alice generates an "elliptic-curve cryptography key-pair for asymmetric authentication"
  - Public key (green) can only encrypt!
  - Private key (red) can only decrypt!
- Alice's public key is registered on the HPC cluster.
- Mallory snoops the public key.

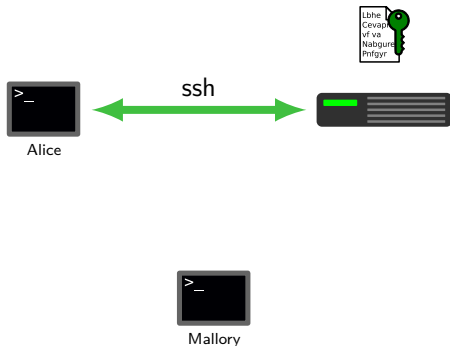


- Alice establishes an ssh-connection with the HPC cluster.



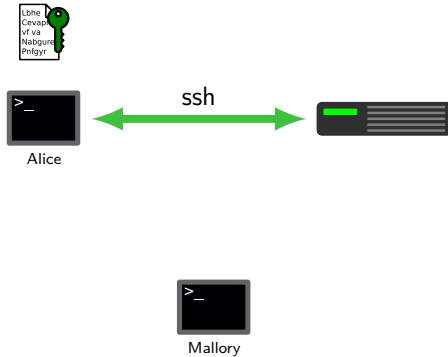


- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.

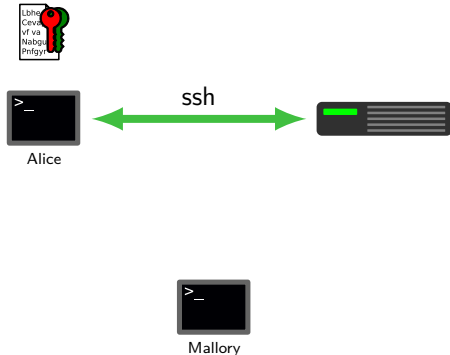


- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.

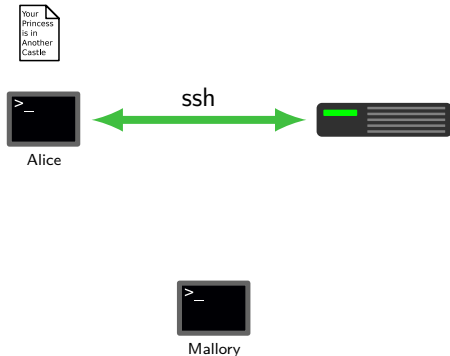




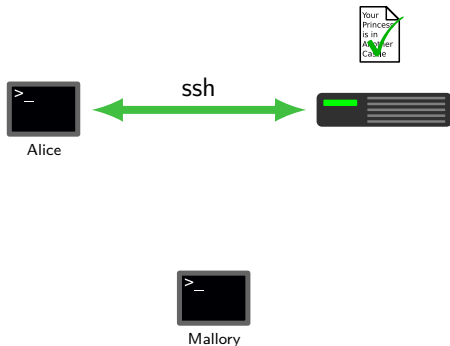
- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.



- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.
- Alice decrypts the text with her private key.



- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.
- Alice decrypts the text with her private key.



- Alice establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Alice to decrypt it.
- Alice decrypts the text with her private key.
- Alice proved her identity.



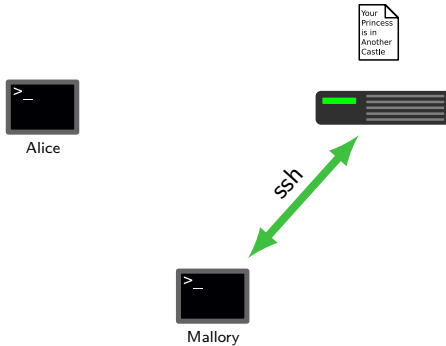
Alice



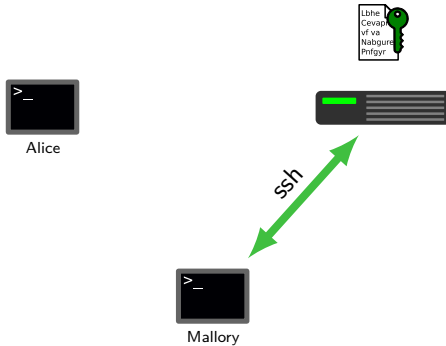
Mallory



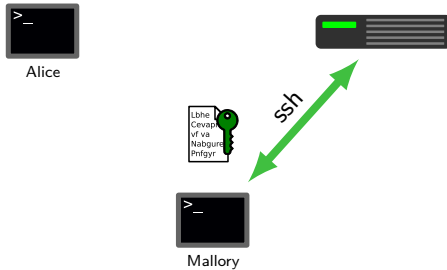
- Mallory establishes an ssh-connection with the HPC cluster.



- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Mallory to decrypt it.

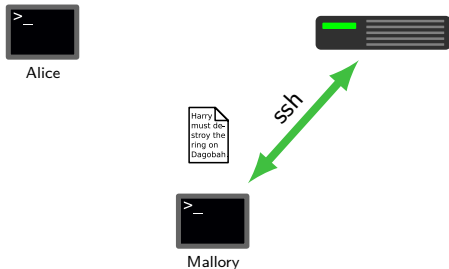


- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Mallory to decrypt it.

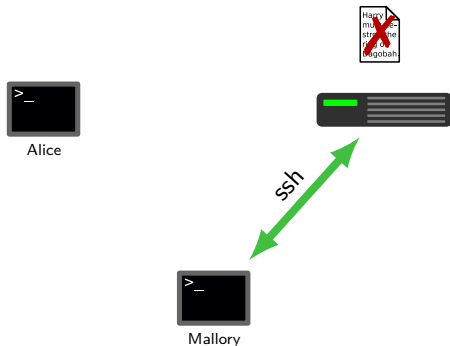


- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Mallory to decrypt it.





- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Mallory to decrypt it.
- Mallory cannot decrypt the text.



- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster encrypts a random text with Alice's public key and challenges Mallory to decrypt it.
- Mallory cannot decrypt the text.
- Mallory is denied access.

Generating and using an "elliptic-curve cryptography key-pair for asymmetric authentication" sounds hard, but is actually quite easy:

## Terminal

```
alice@laptop:~$ ssh-keygen -t ed25519 -N "password123!"
```

Linux provides a key generator: `ssh-keygen`

`-t` selects the key-type (e.g.: `ed25519` = elliptic curve key).

`-f` specify the filename for the key pair.

`-N` specifies additional passphrase.

## Terminal

```
alice@laptop:~$ ssh-keygen -t ed25519 -N "password123!"
Generating public/private ed25519 key pair.
Your identification has been saved in /home/alice/.ssh/id_ed25519
Your public key has been saved in /home/alice/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:yNhMofaDvRHn6AsgI8BNY17yWPRrxGtieQoRa479+SY alice@laptop
...
```

You will find two new files in your `~/ .ssh/` directory:

## Terminal

```
alice@laptop:~$ ls ~/.ssh/  
config id_ed25519 id_ed25519.pub
```

You will find two new files in your `~/ .ssh/` directory:

`id_ed25519` is your private key.

## Terminal

```
alice@laptop:~$ ls ~/.ssh/  
config id_ed25519 id_ed25519.pub
```

## Warning!

Never share your private key!

Not with your coworkers, admins, spouse, or even grandma!

You will find two new files in your `~/.ssh/` directory:  
`id_ed25519` is your private key.

## Terminal

```
alice@laptop:~$ cat ~/.ssh/id_ed25519 #(THIS IS NOT MY ACTUAL KEY)
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXAAAABG5vbmUAAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
...
-----END OPENSSH PRIVATE KEY-----
```

## Warning!

Never share your private key!  
Not with your coworkers, admins, spouse, or even grandma!

You will find two new files in your `~/ .ssh/` directory:

`id_ed25519` is your private key.

`id_ed25519.pub` is your public key.

## Terminal

```
alice@laptop:~$ ls ~/.ssh/  
config id_ed25519 id_ed25519.pub
```

## Not a Warning!

You may share your public key!

Give it to your coworkers, admins, spouse, or even grandma!

Put it on your business card, e-mail signature, t-shirt, billboard.



You will find two new files in your `~/.ssh/` directory:

`id_ed25519` is your private key.

`id_ed25519.pub` is your public key.

## Terminal

```
alice@laptop:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOHvm993oC6kwohuHbV0T2xu/x2INIXS2GxFb84s1KbL
alice@laptop
```

## Not a Warning!

You may share your public key!

Give it to your coworkers, admins, spouse, or even grandma!

Put it on your business card, e-mail signature, t-shirt, billboard.

The key is automatically used by ssh.

Terminal

```
alice@laptop:$ ssh hpc-rub  
alice@hpc:$
```

If you have multiple keys, specify `IdentityFile` in your config to use the correct key for the connection.

## Terminal

```
alice@laptop:~$ cat ~/.ssh/config
Host hpc-rub
  User alice
  Hostname hpc.rub.de
  IdentityFile ~/.ssh/id_ed25519
alice@laptop:~$ ssh hpc-rub
alice@hpc:~$
```

- 1 Generate an elliptic-curve cryptography key-pair for asymmetric authentication
  - Generate in folder `~/.ssh/`
  - Generate an `ed25519` key without passphrase
- 2 Copy the public key to the cluster  
(This is not required for the real HPC-System)
  - use `ssh-copy-id`
  - or write the key in `~/.ssh/authorized_keys` with copy/paste
- 3 Validate the key usage by connecting to the testcluster again

- 1 Generate an elliptic-curve cryptography key-pair for asymmetric authentication
  - Generate in folder `~/.ssh/`
  - Generate an ed25519 key without passphrase

### Terminal

```
alice@laptop:$ ssh-keygen -t ed25519 -N "" -f ~/.ssh/testcluster
```

- 2 Copy the public key to the cluster  
(This is not required for the real HPC-System)
  - use `ssh-copy-id`

### Terminal

```
alice@laptop:~$ ssh-copy-id -i ~/.ssh/testcluster testuser@134.147.41.133
```

- 2 Copy the public key to the cluster  
(This is not required for the real HPC-System)
  - use `ssh-copy-id`
  - or write the key in `~/.ssh/authorized_keys` with copy/paste

### Terminal

```
alice@laptop:~$ ssh-copy-id -i ~/.ssh/testcluster testuser@134.147.41.133
alice@hpc:~$ nano ~/.ssh/authorized_keys
```

3 Validate the key usage by connecting to the testcluster again

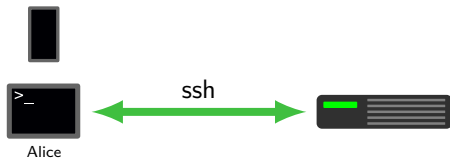
Terminal

```
alice@laptop:$ ssh testuser@134.147.41.133  
alice@hpc:$
```

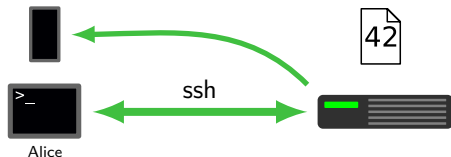




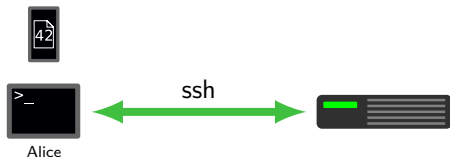
- Alice enables two-factor-authentication with her mobile phone.



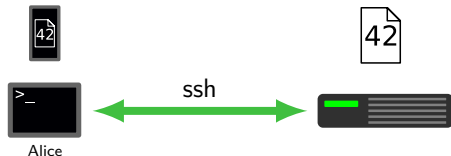
- Alice establishes an ssh-connection with the HPC cluster.



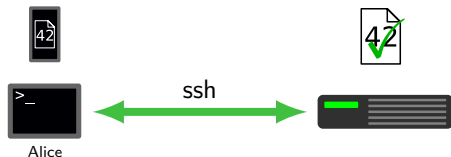
- Alice establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.



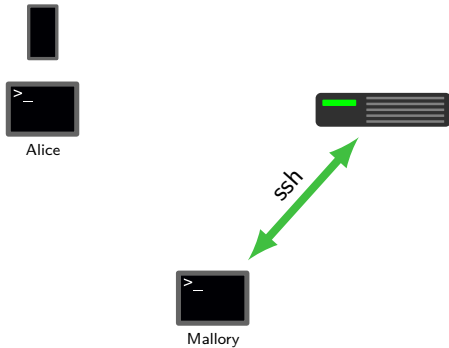
- Alice establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.



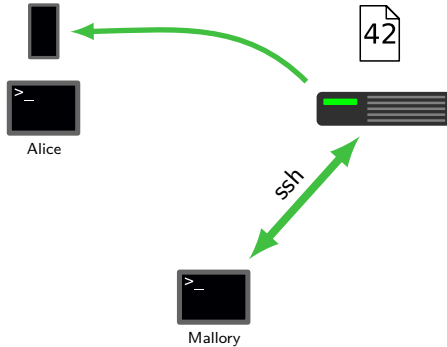
- Alice establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.
- Alice sends the number to the cluster via ssh.



- Alice establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.
- Alice sends the number to the cluster via ssh.
- Alice proved her identity.

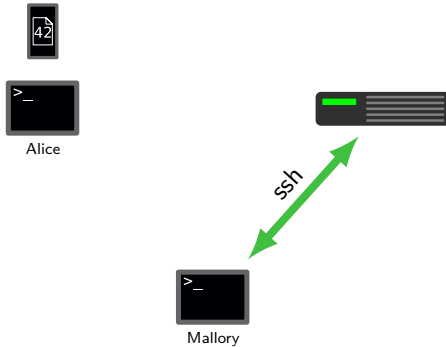


- Mallory establishes an ssh-connection with the HPC cluster.

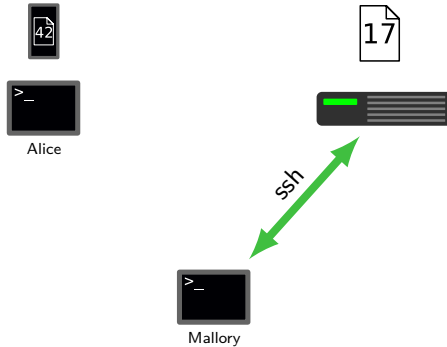


- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.

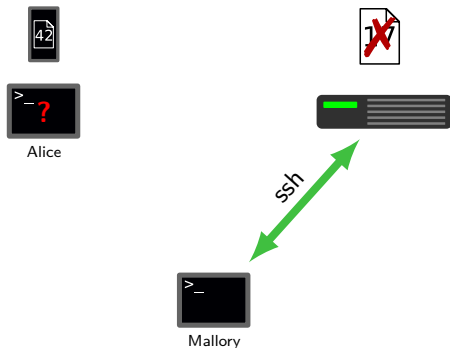




- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.



- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.
- Mallory guesses the number.



- Mallory establishes an ssh-connection with the HPC cluster.
- The cluster sends a random number to Alice's mobile phone.
- Mallory guesses the number.
- Mallory is denied access (and Alice is left confused).

# Datatransfer from/to HPC Clusters

## Terminal

```
alice@laptop:$ cp source destination
```

- The `cp` command copies a file given by `source` to `destination`

## Terminal

```
alice@laptop:$ cp -r source destination
```

- The `cp` command copies a file given by `source` to `destination`
- The `-r` flag copies recursively, thus allowing copying of directories and their content

## Terminal

```
alice@laptop:~$ cp --help
```

- The `cp` command copies a file given by source to destination
- The `-r` flag copies recursively, thus allowing copying of directories and their content
- Consult the `--help` flag for more available options

## Terminal

```
alice@laptop:$ scp source destination
```

- Prepending an “s” to the cp command enables copies from/to remote machines



## Terminal

```
alice@laptop:$ scp myData username@ip-address:~/myProject/
```

- Prepending an “s” to the cp command enables copies from/to remote machines
- Remote locations are prefixed by the username and IP/hostname

## Terminal

```
alice@laptop:~$ scp username@ip-address:~/myProject/myData .
```

- Prepending an “s” to the cp command enables copies from/to remote machines
- Remote locations are prefixed by the username and IP/hostname

## Terminal

```
alice@laptop:~$ scp -r username@ip-address:~/myProject .
```

- Prepending an “s” to the cp command enables copies from/to remote machines
- Remote locations are prefixed by the username and IP/hostname
- The -r flag copies recursively, thus allowing copying of directories and their content

## Terminal

```
alice@laptop:~$ scp --help
```

- Prepending an “s” to the cp command enables copies from/to remote machines
- Remote locations are prefixed by the username and IP/hostname
- The -r flag copies recursively, thus allowing copying of directories and their content
- Consult the --help flag for more available options

## Terminal

```
alice@laptop:$ rsync source destination
```

- `rsync` functions similarly to `scp`, but adds many quality of life features

## Terminal

```
alice@laptop:$ rsync -r username@ip-address:~/myProject .
```

- `rsync` functions similarly to `scp`, but adds many quality of life features

## Terminal

```
alice@laptop:~$ rsync --bwlimit=1024 username@ip-address:~/myProject .
```

- rsync functions similarly to scp, but adds many quality of life features
- `--bwlimit=RATE` limits the transfer speed by RATE KiBit/s

## Terminal

```
alice@laptop:$ rsync --progress username@ip-address:~/myProject .
```

- rsync functions similarly to scp, but adds many quality of life features
- `--bwlimit=RATE` limits the transfer speed by RATE KiBit/s
- `--progress` gives live updates on copy



## Terminal

```
alice@laptop:$ rsync --compress username@ip-address:~/myProject .
```

- rsync functions similarly to scp, but adds many quality of life features
- `--bwlimit=RATE` limits the transfer speed by RATE KiBit/s
- `--progress` gives live updates on copy
- `--compress` compresses data for transfer

## Terminal

```
alice@laptop:$ rsync --checksum username@ip-address:~/myProject .
```

- rsync functions similarly to scp, but adds many quality of life features
- `--bwlimit=RATE` limits the transfer speed by RATE KiBit/s
- `--progress` gives live updates on copy
- `--compress` compresses data for transfer
- `--checksum` checks if data was already transferred with checksums

## Terminal

```
alice@laptop:~$ rsync --help
```

- rsync functions similarly to scp, but adds many quality of life features
- `--bwlimit=RATE` limits the transfer speed by RATE KiBit/s
- `--progress` gives live updates on copy
- `--compress` compresses data for transfer
- `--checksum` checks if data was already transferred with checksums
- Consult the `--help` flag for more available options

## Terminal

```
alice@laptop:~$ ls -l | wc -l  
65537  
alice@laptop:~$
```

- Alice has a lot of files

Warning!

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l  
65537  
alice@laptop:~$ for i in *; do rsync ${i} alice@141.421.356.23:~/ ;done
```

- Alice has a lot of files
- Alice loops over the files and starts an `rsync` for each

Warning!

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l  
65537  
alice@laptop:~$ for i in *; do rsync ${i} alice@141.421.356.23:~/ ;done
```

- Alice has a lot of files
- Alice loops over the files and starts an `rsync` for each
- Each file transfer requires:

Warning!

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l  
65537  
alice@laptop:~$ for i in *; do rsync ${i} alice@141.421.356.23:~/ ;done
```

- Alice has a lot of files
- Alice loops over the files and starts an `rsync` for each
- Each file transfer requires:
  - establishing connection

Warning!

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ for i in *; do rsync ${i} alice@141.421.356.23:~/ ;done
```

- Alice has a lot of files
- Alice loops over the files and starts an `rsync` for each
- Each file transfer requires:
  - establishing connection
  - key exchange and key authentication

**Warning!**

Do not do this, as it is inefficient!



## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ for i in *; do rsync ${i} alice@141.421.356.23:~/ ;done
```

- Alice has a lot of files
- Alice loops over the files and starts an `rsync` for each
- Each file transfer requires:
  - establishing connection
  - key exchange and key authentication
  - file transfer initialization and finalization

**Warning!**

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ rsync * alice@141.421.356.23:~/
```

- Alice has a lot of files

Warning!

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ rsync * alice@141.421.356.23:~/
```

- Alice has a lot of files
- Alice starts rsync with a wildcard for the files

**Warning!**

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ rsync * alice@141.421.356.23:~/
```

- Alice has a lot of files
- Alice starts rsync with a wildcard for the files
- Each file transfer requires:

**Warning!**

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ rsync * alice@141.421.356.23:~/
```

- Alice has a lot of files
- Alice starts rsync with a wildcard for the files
- Each file transfer requires:
  - file transfer initialization and finalization

**Warning!**

Do not do this, as it is inefficient!

## Terminal

```
alice@laptop:~$ ls -l | wc -l  
65537  
alice@laptop:~$
```

- Alice has a lot of files

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
```

- Alice has a lot of files
- Alice creates a tar archive with her files

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
alice@laptop:~$ rsync files.tar alice@141.421.356.23:~/
```

- Alice has a lot of files
- Alice creates a tar archive with her files
- Alice transfers one big file



## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
alice@laptop:~$ rsync files.tar alice@141.421.356.23:~/
alice@laptop:~$ ssh alice@141.421.356.23
alice@hpc:~$ tar -xvf files.tar
```

- Alice has a lot of files
- Alice creates a tar archive with her files
- Alice transfers one big file

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
alice@laptop:~$ rsync files.tar alice@141.421.356.23:~/
alice@laptop:~$ ssh alice@141.421.356.23
alice@hpc:~$ tar -xvf files.tar
```

- Alice has a lot of files
- Alice creates a tar archive with her files
- Alice transfers one big file and extracts it on the cluster
- This is efficient because:

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
alice@laptop:~$ rsync files.tar alice@141.421.356.23:~/
alice@laptop:~$ ssh alice@141.421.356.23
alice@hpc:~$ tar -xvf files.tar
```

- Alice has a lot of files
- Alice creates a tar archive with her files
- Alice transfers one big file and extracts it on the cluster
- This is efficient because:
  - Minimizes rsync and file transfer overhead

## Terminal

```
alice@laptop:~$ ls -l | wc -l
65537
alice@laptop:~$ tar -cvf files.tar *
alice@laptop:~$ rsync files.tar alice@141.421.356.23:~/
alice@laptop:~$ ssh alice@141.421.356.23
alice@hpc:~$ tar -xvf files.tar
```

- Alice has a lot of files
- Alice creates a tar archive with her files
- Alice transfers one big file and extracts it on the cluster
- This is efficient because:
  - Minimizes `rsync` and file transfer overhead
  - Enables cross-file compression

# SLURM: Simple Linux Utility for Resource Management



SLURM is:

---

[https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm\\_logo.svg/2238px-Slurm\\_logo.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm_logo.svg/2238px-Slurm_logo.svg.png)



SLURM is:

- A resource manager.

[https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm\\_logo.svg/2238px-Slurm\\_logo.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm_logo.svg/2238px-Slurm_logo.svg.png)



SLURM is:

- A resource manager.
- A scheduler.

---

[https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm\\_logo.svg/2238px-Slurm\\_logo.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm_logo.svg/2238px-Slurm_logo.svg.png)





SLURM is:

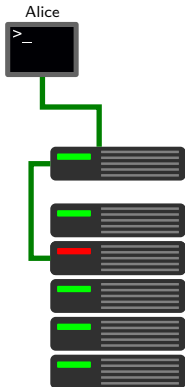
- A resource manager.
- A scheduler.
- An accountant.

[https:](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm_logo.svg/2238px-Slurm_logo.svg.png)

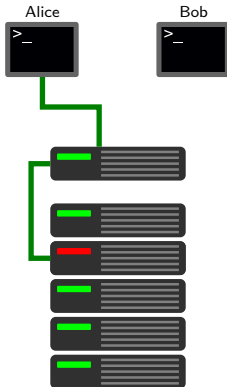
[//upload.wikimedia.org/wikipedia/commons/thumb/  
3/3a/Slurm\\_logo.svg/2238px-Slurm\\_logo.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Slurm_logo.svg/2238px-Slurm_logo.svg.png)



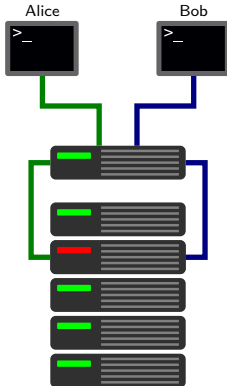
Alice is conducting Research on an HPC cluster.



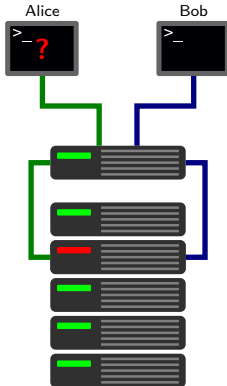
Alice connects to a compute node.



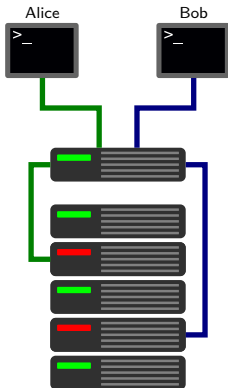
Bob is conducting Research on an HPC cluster.



Bob forgets which node was dedicated to him and accidentally connects to the same one as Alice.



Alice wonders why her calculation slows down. Bob accidentally interferes with Alice's research.



Solution:

Alice and Bob request a free node from the SLURM resource manager on the login node.

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```



`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**PARTITION** Name of a set of nodes of similar type. Allows for hardware specific requests.

A node can be part of multiple partitions.

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc  cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**AVAIL** Availability of a partition (Up, down, ...).

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**TIMELIMIT** Maximum time for a resource request (job) in in this partition. Format: dd-hh:mm:ss

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc  cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**NODES** Number of nodes in this partition.

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**STATE** Current operation status of nodes in this partition:

**idle:** Node is waiting for work.

**alloc:** Node is allocated and busy.

**down:** Node is unreachable.

`sinfo` gives a list of all nodes and their status:

## Terminal

```
alice@hpc:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
first-nodes up 2-00:00:00    2  idle  cpu01,gpu01
cpu-nodes up 2-00:00:00    3  idle  cpu[01,02,04]
cpu-nodes up 2-00:00:00    1  alloc  cpu03
gpu-nodes up 2-00:00:00    2  idle  gpu[01-02]
```

**NODELIST** List of nodes in the partition in compressed form.

**node[1-3,6-7]:** node1, node2, node3, node6, node7

`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

`-p/--partition` Partition to request a node from.



`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

`-p/--partition` Partition to request a node from.

`-t/--time` Maximum time the job will run.

`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

- p/--partition Partition to request a node from.
- t/--time Maximum time the job will run.
- pty Run with pseudoterminal

`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

- `-p/--partition` Partition to request a node from.
- `-t/--time` Maximum time the job will run.
- `--pty` Run with pseudoterminal
- `bash` run a bash shell

`srun` can be used to request resources for an interactive session:

## Terminal

```
alice@hpc:$ srun -p gpu-nodes --time=30:00 --pty bash
alice@gpu1:$
```

`-p/--partition` Partition to request a node from.

`-t/--time` Maximum time the job will run.

`--pty` Run with pseudoterminal

`bash` run a bash shell

All flags at <https://slurm.schedmd.com/srun.html>

`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:$ sbatch -p gpu-nodes --time=30:00 myscript.sh  
Submitted batch job 1337
```

`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:$ sbatch -p gpu-nodes --time=30:00 myscript.sh
Submitted batch job 1337
```

- `myscript.sh` will be executed on the nodes.

`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:~$ sbatch -p gpu-nodes --time=30:00 myscript.sh
Submitted batch job 1337
```

- `myscript.sh` will be executed on the nodes.
- Each batch job is assigned a unique ID (e.g. 1337).

`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:~$ sbatch -p gpu-nodes --time=30:00 myscript.sh
Submitted batch job 1337
```

- `myscript.sh` will be executed on the nodes.
- Each batch job is assigned a unique ID (e.g. 1337).
- Output and errors will be written to `slurm-${SLURM_JOB_ID}.out/err`



`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:~$ sbatch -p gpu-nodes --time=30:00 myscript.sh
Submitted batch job 1337
```

- `myscript.sh` will be executed on the nodes.
- Each batch job is assigned a unique ID (e.g. 1337).
- Output and errors will be written to `slurm-${SLURM_JOB_ID}.out/err`
- Set output/error files with `--output=/--error=` flags.

`sbatch` can be used to request resources for scripted job.

## Terminal

```
alice@hpc:~$ sbatch -p gpu-nodes --time=30:00 myscript.sh
Submitted batch job 1337
```

- `myscript.sh` will be executed on the nodes.
- Each batch job is assigned a unique ID (e.g. 1337).
- Output and errors will be written to `slurm-${SLURM_JOB_ID}.out/err`
- Set output/error files with `--output=/--error=` flags.
- All flags at <https://slurm.schedmd.com/sbatch.html>

All valid `sbatch`-flags can also be incorporated into the script itself:

## Terminal

```
alice@hpc:$ cat myscript.sh
#!/bin/bash
#SBATCH --output=TestJob.out
#SBATCH --error=TestJob.err
#SBATCH --time=00:10:00
#SBATCH --partition=cpu-nodes
...
alice@hpc:$ sbatch myscript.sh
```

All valid `sbatch`-flags can also be incorporated into the script itself:

## Terminal

```
alice@hpc:$ cat myscript.sh
#!/bin/bash
#SBATCH --output=TestJob.out
#SBATCH --error=TestJob.err
#SBATCH --time=00:10:00
#SBATCH --partition=cpu-nodes
...
alice@hpc:$ sbatch myscript.sh
```

This is called the SLURM-Header.

All valid `sbatch`-flags can also be incorporated into the script itself:

## Terminal

```
alice@hpc:$ cat myscript.sh
#!/bin/bash
#SBATCH --output=TestJob.out
#SBATCH --error=TestJob.err
#SBATCH --time=00:10:00
#SBATCH --partition=cpu-nodes
...
alice@hpc:$ sbatch myscript.sh
```

This is called the SLURM-Header.

`#!/bin/bash` The script interpreter should be `bash`.

All valid `sbatch`-flags can also be incorporated into the script itself:

## Terminal

```
alice@hpc:$ cat myscript.sh
#!/bin/bash
#SBATCH --output=TestJob.out
#SBATCH --error=TestJob.err
#SBATCH --time=00:10:00
#SBATCH --partition=cpu-nodes
...
alice@hpc:$ sbatch myscript.sh
```

This is called the SLURM-Header.

`#!/bin/bash` The script interpreter should be `bash`.

`#SBATCH` Ignored by the interpreter, but parsed by `sbatch`.

Mistakes happen to everyone!

If you accidentally submitted a job and want to cancel it (running or pending) you can use the `scancel` command.

## Terminal

```
alice@hpc:$ scancel 1337  
alice@hpc:$
```

Mistakes happen to everyone!

If you accidentally submitted a job and want to cancel it (running or pending) you can use the `scancel` command.

## Terminal

```
alice@hpc:$ scancel 1337  
alice@hpc:$
```

- Job with the jobid 1337 will be stopped (if possible)



Mistakes happen to everyone!

If you accidentally submitted a job and want to cancel it (running or pending) you can use the `scancel` command.

## Terminal

```
alice@hpc:$ scancel 1337  
alice@hpc:$
```

- Job with the jobid 1337 will be stopped (if possible)
- You can only cancel your own jobs!

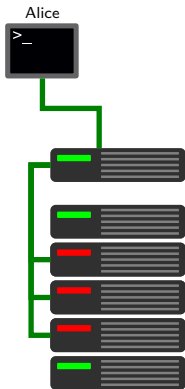
Mistakes happen to everyone!

If you accidentally submitted a job and want to cancel it (running or pending) you can use the `scancel` command.

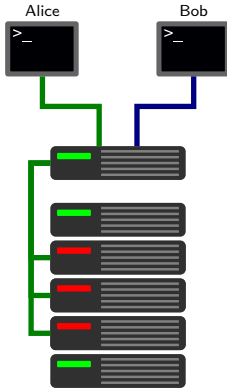
### Terminal

```
alice@hpc:$ scancel 1337
alice@hpc:$
```

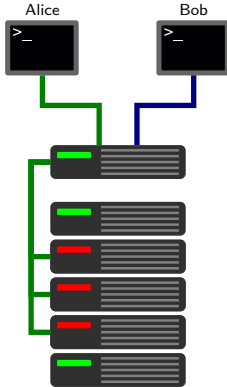
- Job with the jobid 1337 will be stopped (if possible)
- You can only cancel your own jobs!
- All flags at <https://slurm.schedmd.com/scancel.html>



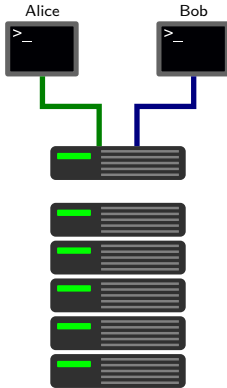
Alice' job occupies several nodes.



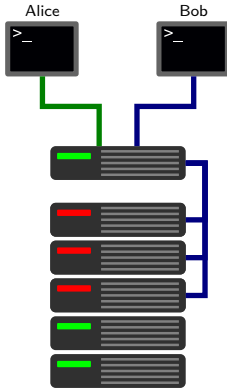
There are not enough free nodes left for Bob's job.



SLURM put Bob's job on hold until the required resources are available.



SLURM put Bob's job on hold until the required resources are available.



SLURM put Bob's job on hold until the required resources are available.

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice  R 1:37 1 cpu01
6883  cpu-nodes  calc carol  R 0:42 1 cpu02 ...
```



squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:~$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice  R 1:37 1 cpu01
6883  cpu-nodes  calc carol  R 0:42 1 cpu02 ...
```

**JOBID** Unique ID for this job

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
6886 cpu-nodes check bob PD 0:00 1 (Resources)
6882 cpu-nodes sim alice R 1:37 1 cpu01
6883 cpu-nodes calc carol R 0:42 1 cpu02 ...
```

**PARTITION** Partition the job is running on.

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice  R 1:37 1 cpu01
6883  cpu-nodes  calc carol  R 0:42 1 cpu02 ...
```

**NAME** Name given to job by user via the `--job-name=` flag.

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice  R 1:37 1 cpu01
6883  cpu-nodes  calc carol  R 0:42 1 cpu02 ...
```

**USER** User who submitted/started the job.

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:~$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
6886   cpu-nodes check bob   PD 0:00      1 (Resources)
6882   cpu-nodes sim  alice R 1:37      1 cpu01
6883   cpu-nodes calc  carol R 0:42      1 cpu02 ...
```

ST Status of the Job. Most important statuses are:

- R Running
- PD Pending
- CG Completing

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00      1 (Resources)
6882  cpu-nodes  sim   alice  R 1:37      1 cpu01
6883  cpu-nodes  calc  carol  R 0:42      1 cpu02 ...
```

**TIME** Time the job is already running.

squeue gives a list of all running and pending jobs:

## Terminal

```
alice@hpc:~$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob   PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice R 1:37 1 cpu01
6883  cpu-nodes  calc carol R 0:42 1 cpu02 ...
```

**NODES** Number of nodes allocated to this job.

squeue gives a list of all running and pending jobs:

## Terminal

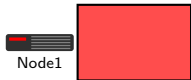
```
alice@hpc:$ squeue
JOBID PARTITION NAME  USER  ST TIME NODES NODELIST(REASON)
6886  cpu-nodes  check bob    PD 0:00 1 (Resources)
6882  cpu-nodes  sim  alice  R 1:37 1 cpu01
6883  cpu-nodes  calc carol  R 0:42 1 cpu02 ...
```

**NODELIST(REASON)** List of nodes allocated to the job.

Or reason why the job is not running (e.g. Resources).



New:



Backfill-Scheduling for one node:

New:



Node1



Backfill-Scheduling for one node:

- Append new jobs as they come in.

New:



Backfill-Scheduling for one node:

- Append new jobs as they come in.

New:



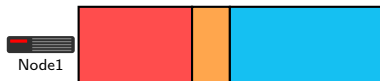
Node1



Backfill-Scheduling for one node:

- Append new jobs as they come in.

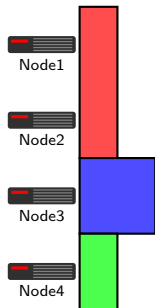
New:



Backfill-Scheduling for one node:

- Append new jobs as they come in.

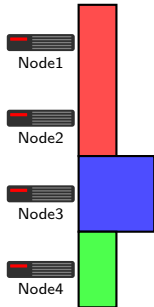
New:



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.

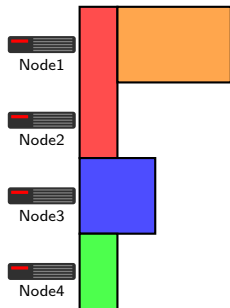
New:



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.

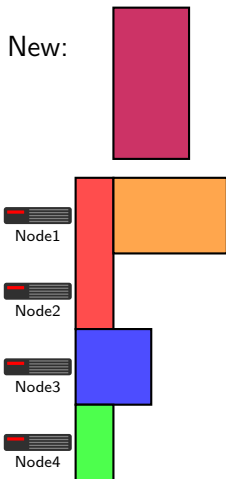
New:



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.

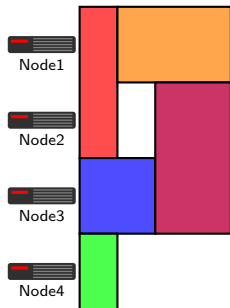




Backfill-Scheduling for multiple nodes:

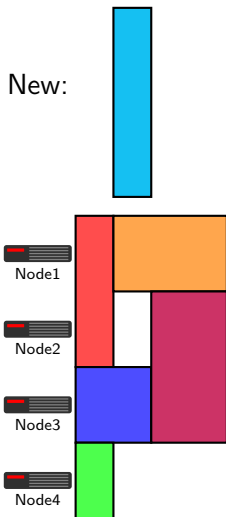
- Append new jobs as they come in.

New:



Backfill-Scheduling for multiple nodes:

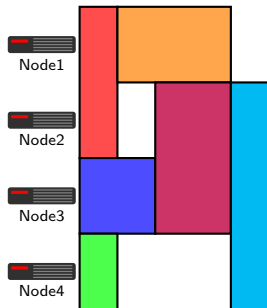
- Append new jobs as they come in.



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.

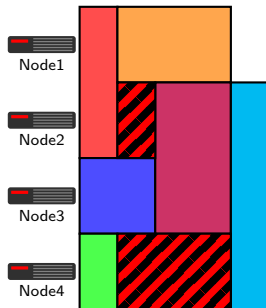
New:



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.

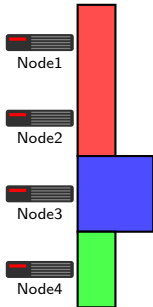
New:



Backfill-Scheduling for multiple nodes:

- Append new jobs as they come in.
- This can leave gaps in the schedule and lead to idle times.

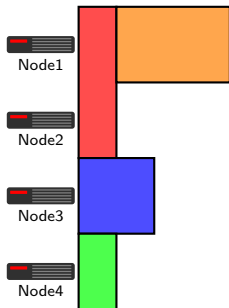
New:



Intelligent scheduling for multiple nodes:

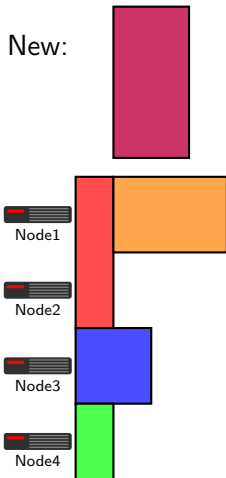
- Append new jobs as they come in.

New:



Intelligent scheduling for multiple nodes:

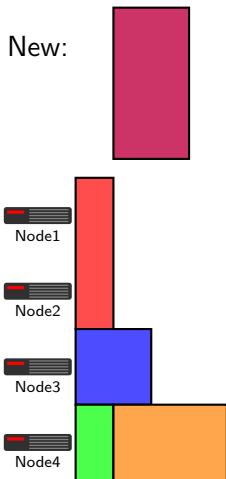
- Append new jobs as they come in.



Intelligent scheduling for multiple nodes:

- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.

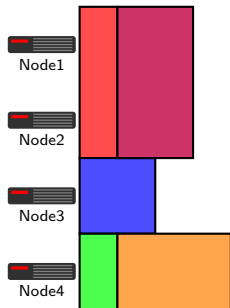




Intelligent scheduling for multiple nodes:

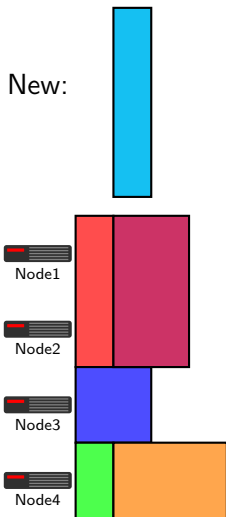
- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.

New:



Intelligent scheduling for multiple nodes:

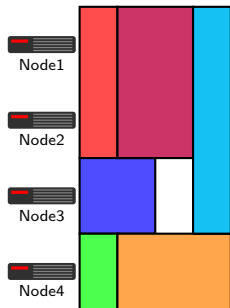
- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.



Intelligent scheduling for multiple nodes:

- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.

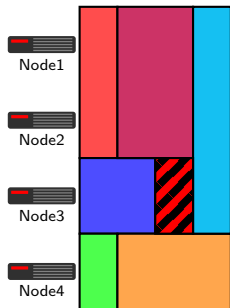
New:



Intelligent scheduling for multiple nodes:

- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.

New:



Intelligent scheduling for multiple nodes:

- Append new jobs as they come in.
- Reorganize queued jobs to minimize idle times.
- Leaves fewer and smaller gaps than Backfilling.
- Relies on realistic runtime estimates by users.



Node1



Node2



Node3



Node4



Node5



Node6

Alice submits a huge amount of jobs.



Node1



Node2



Node3



Node4



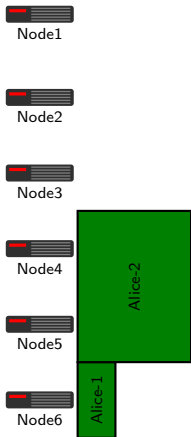
Node5



Node6

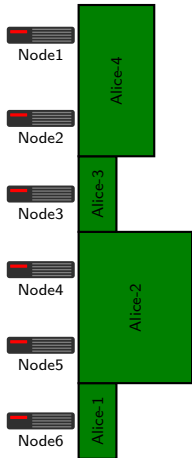


Alice submits a huge amount of jobs.

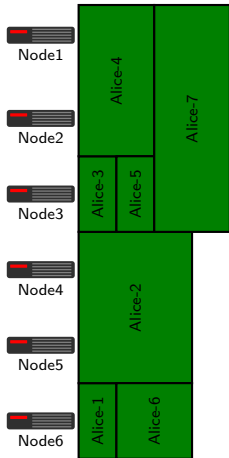


Alice submits a huge amount of jobs.

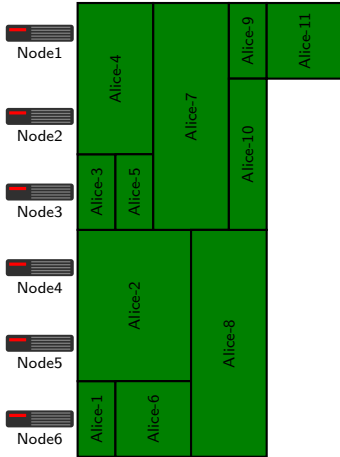




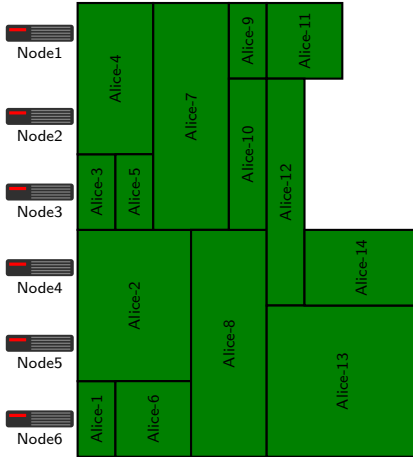
Alice submits a huge amount of jobs.



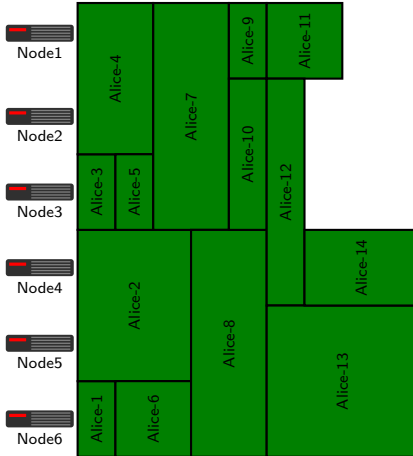
Alice submits a huge amount of jobs.



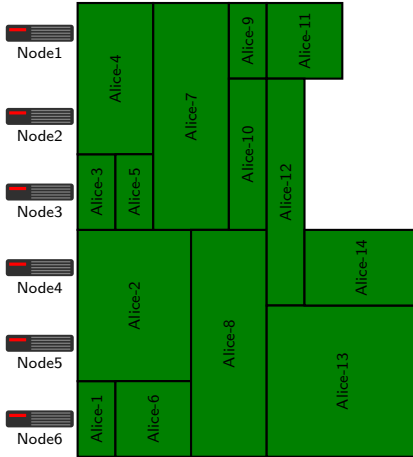
Alice submits a huge amount of jobs.



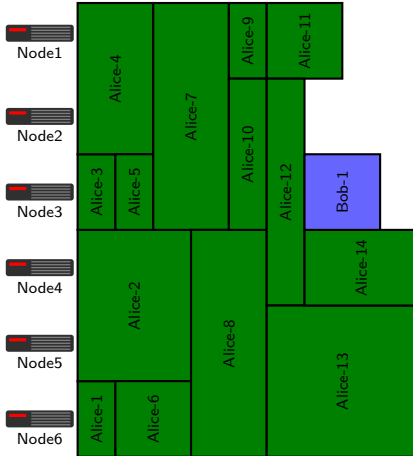
Alice submits a huge amount of jobs.



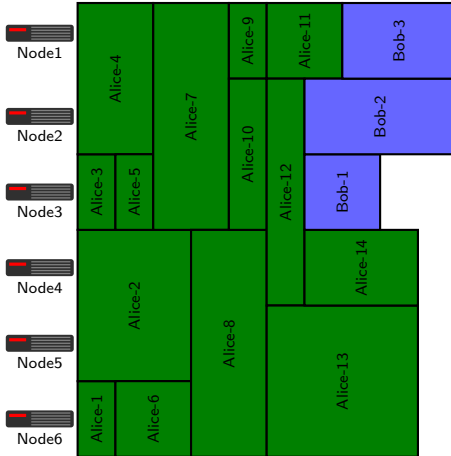
Alice submits a huge amount of jobs.  
She filled the cluster and queue.



Bob submits some jobs.

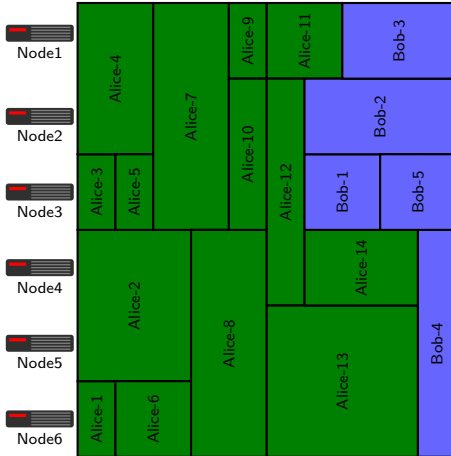


Bob submits some jobs.

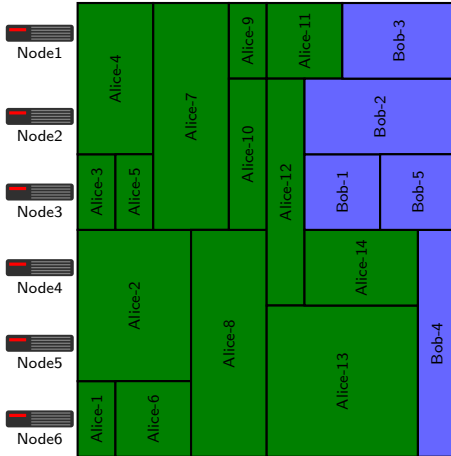


Bob submits some jobs.

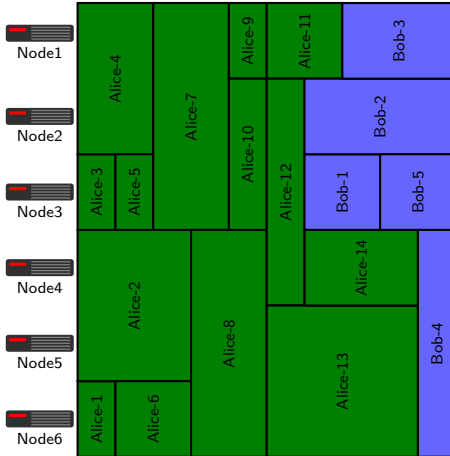




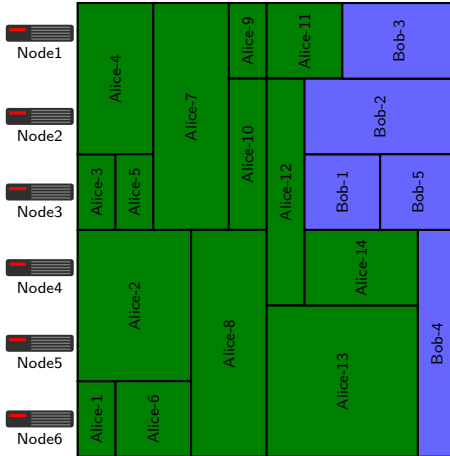
Bob submits some jobs.



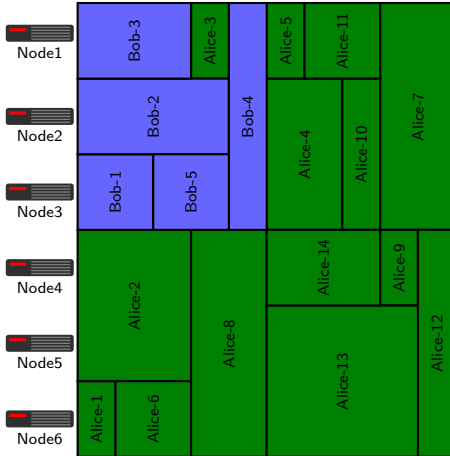
The cluster is not shared fairly, just because Alice spammed jobs quickly and early enough.



SLURM assigns a priority to each job, based on the users past resource usage.



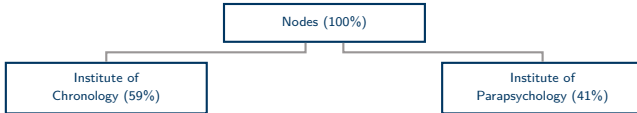
Users who used the cluster less have a higher job priority.



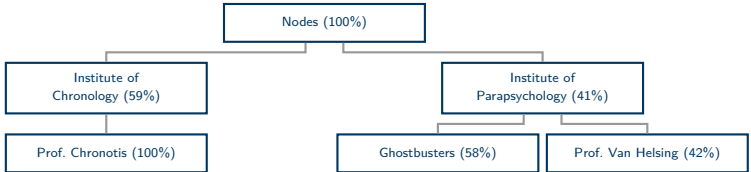
SLURM reorganizes the queue based on the priorities to ensure a fair share of the cluster. Already running jobs are not stopped!

Nodes (100%)

The accounting is done in a tree-like structure.

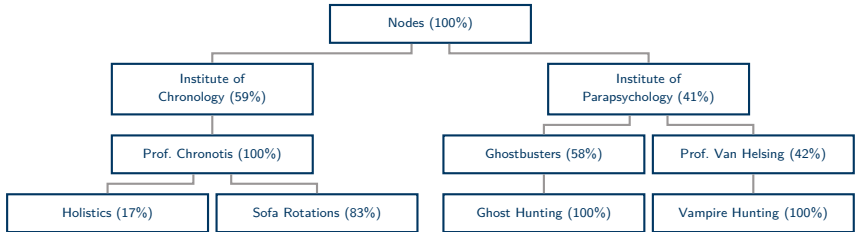


All nodes are shared between all institutes with individual shares.

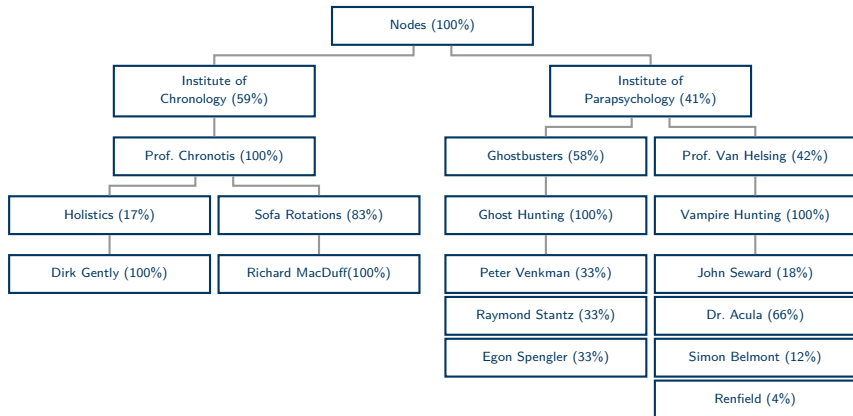


An institutes share is further shared between different groups.

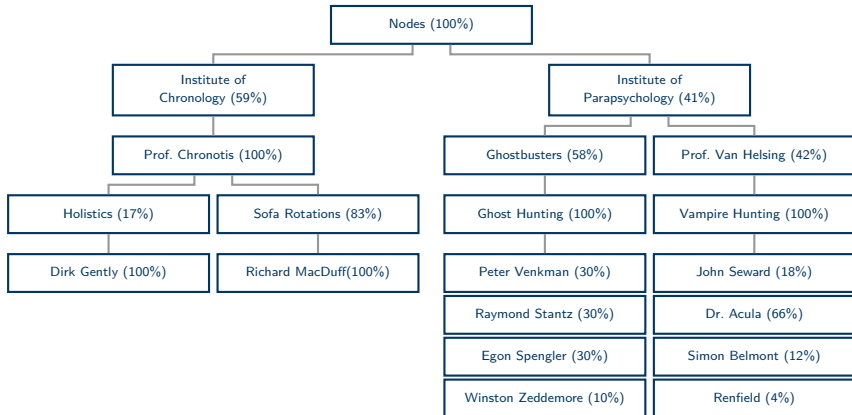




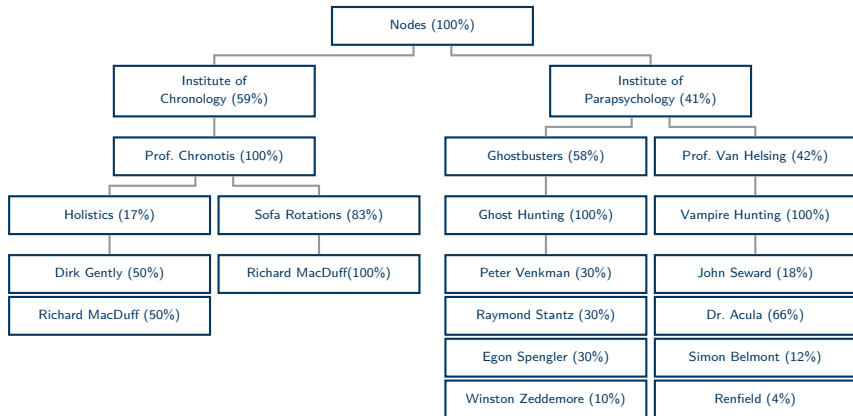
A group can have multiple projects that share the groups share.



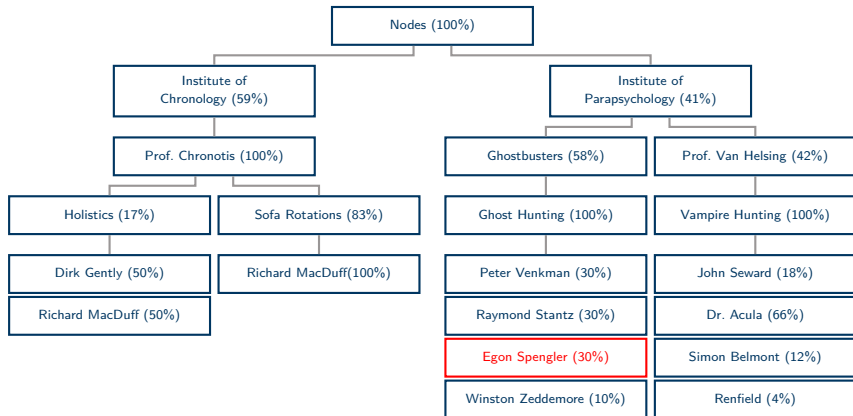
A project can have multiple users, that share the projects share.



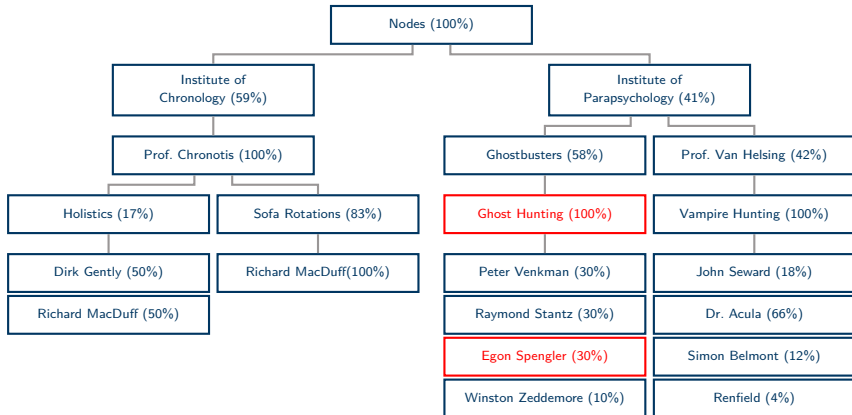
Adding a new user renormalizes the shares of the users in the same project.



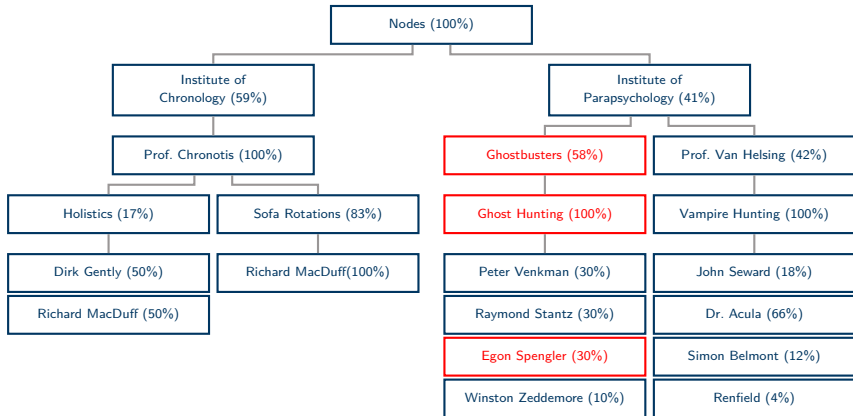
Users can be members of multiple projects. The project needs to be specified during job submission (e.g. SLURM-header).



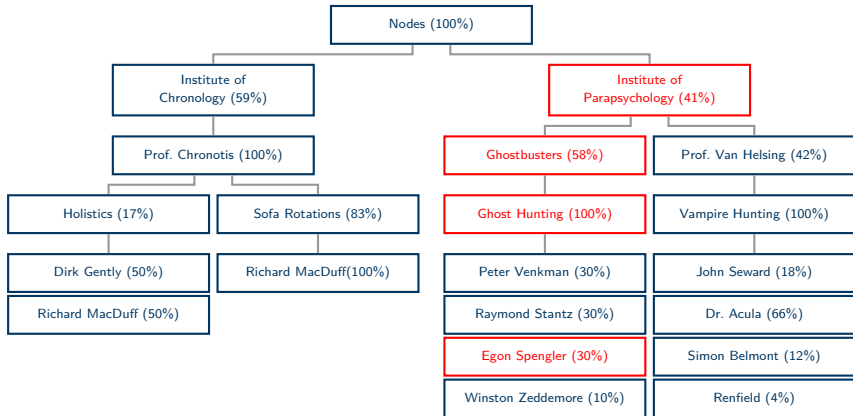
Example: User “Egon Spengler”



Example: User “Egon Spengler” of the “Ghost Hunting” project

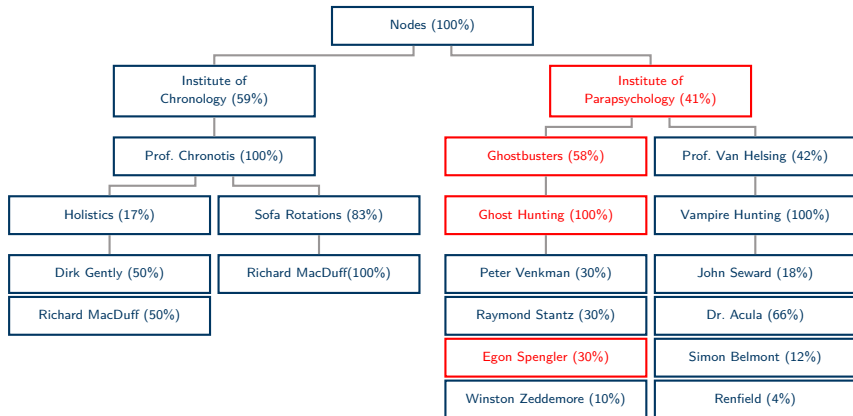


Example: User “Egon Spengler” of the “Ghost Hunting” project of the “Ghostbusters” group



Example: User “Egon Spengler” of the “Ghost Hunting” project of the “Ghostbusters” group in the “Chair of Parapsychology”





Example: User “Egon Spengler” has a real share of:  
 $30\% \times 100\% \times 58\% \times 41\% = 7\%$  of the Nodes.

## Terminal

```

root@hpc:$ sshare
Account          User RawShares NormShares RawUsage EffectvUsage FairShare
-----
...
parapsy          41  0.525641  1432063  0.737318  0.000000
ghostbusters    58  0.580000  917609   0.640760  0.000000
ghosthunting    100 1.000000  917609   1.000000  0.000000
ghosthunting    pvenkman 30  0.300000  3365     0.003667  0.272727
ghosthunting    rstantz  30  0.300000  186275   0.203000  0.181818
ghosthunting    espengle 30  0.300000  727422   0.792736  0.090909
ghosthunting    wzeddemo 10  0.100000  547      0.000596  0.363636
vanhelsing      42  0.420000  514454   0.359240  0.000000
vamphunting     100 1.000000  514454   1.000000  0.000000
vamphunting     jseward  18  0.180000  72698    0.141311  0.545455
vamphunting     acula    66  0.660000  666      0.001295  0.636364
vamphunting     sbelmont 12  0.120000  441050   0.857317  0.454545
vamphunting     renfield  4  0.040000  40       0.000078  0.727273

```

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

**Account** Name of the account.

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

User Name of the user.

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

**RawShares** User's share of the account.

**NormShares** User's share of the account relative to other members of the same account  $\left( S_{norm} = \frac{S_{raw,user}}{\sum S_{raw,siblings}} \right)$ .

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

**RawUsage** Time in seconds nodes were used.

**EffectvUsage** Resource usage relative to other members of the same account  $\left( U_{eff} = \frac{U_{raw,user}}{\sum U_{raw,siblings}} \right)$

sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

**FairShare** [0, 1] Priority rank with which jobs will be scheduled.  
The closer to 1.0 the more the jobs are prioritized.



sshare shows only own shares and usage (privacy reasons):

### Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799    0.321122  1.000000
sofarot     rmduff     100  1.000000  432971    1.000000  0.818182
```

On If a user belongs to multiple accounts the account must be specified at job submission with the `--account` flag.

(On the RUB cluster the flag is always required)

If a user participates in multiple projects with cryptic account names it can be cumbersome to figure out which project to select for a job.

## Terminal

```
rmduff@hpc:$ sshare
Account      User RawShares NormShares RawUsage EffectvUsage FairShare
-----
holistics   rmduff      50  0.500000   24799   0.321122  1.000000
sofarot     rmduff     100  1.000000  432971  1.000000  0.818182
```

If a user participates in multiple projects with cryptic account names it can be cumbersome to figure out which project to select for a job.

### Terminal

```
rmduff@hpc:$ rub-acclist
Project ID   | Project Description
-----+-----
holistics   | The fundamental interconnectedness of all things
sofarot     | The translated quaternion for optimal pivoting
```

The `rub-acclist` (RUB exclusive) lists all projects a user is part of, as well as the description provided by the project manager.

FairShare Assignment:

- Sibling accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.

FairShare Assignment:

- Sibling accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.
- $R = \frac{S_{norm}}{U_{eff}} > 1$ : Used less than entitled to

## FairShare Assignment:

- Sibling accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.
- $R = \frac{S_{norm}}{U_{eff}} > 1$ : Used less than entitled to
- $R = \frac{S_{norm}}{U_{eff}} < 1$ : Used more than entitled to

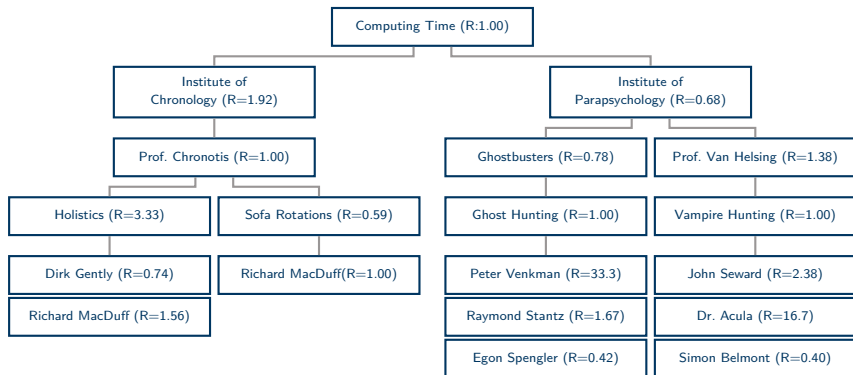
## FairShare Assignment:

- Sibling accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.
- $R = \frac{S_{norm}}{U_{eff}} > 1$ : Used less than entitled to
- $R = \frac{S_{norm}}{U_{eff}} < 1$ : Used more than entitled to
- Parent accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.

## FairShare Assignment:

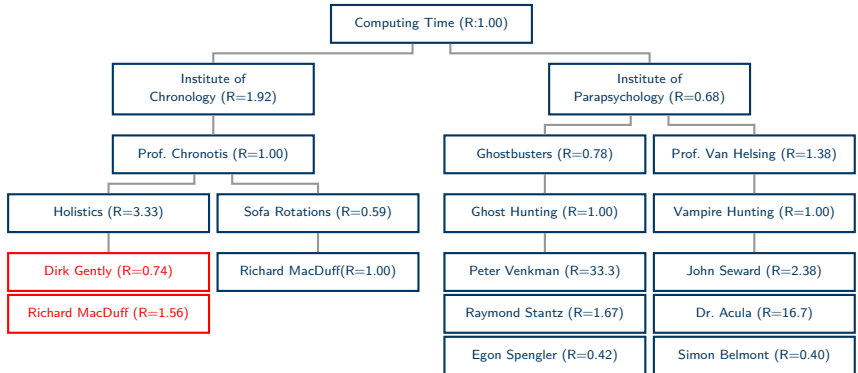
- Sibling accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.
- $R = \frac{S_{norm}}{U_{eff}} > 1$ : Used less than entitled to
- $R = \frac{S_{norm}}{U_{eff}} < 1$ : Used more than entitled to
- Parent accounts are ranked by their  $R = \frac{S_{norm}}{U_{eff}}$  value.
- Child rankings are concatenated accordingly.

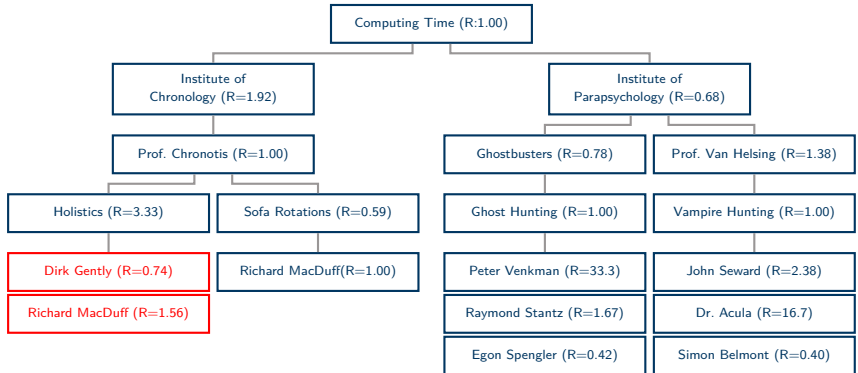




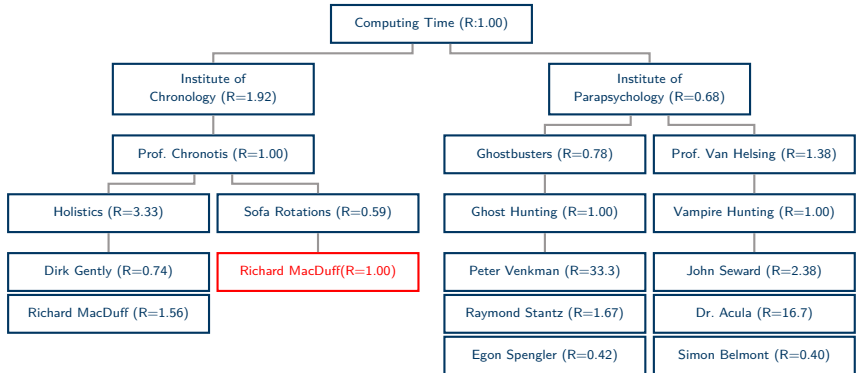
$R = \frac{S_{norm}}{U_{eff}} > 1$ : used fewer resources than entitled to  
 $R = \frac{S_{norm}}{U_{eff}} < 1$ : used more resources than entitled to

# SLURM as Accountant: Prioritization Example

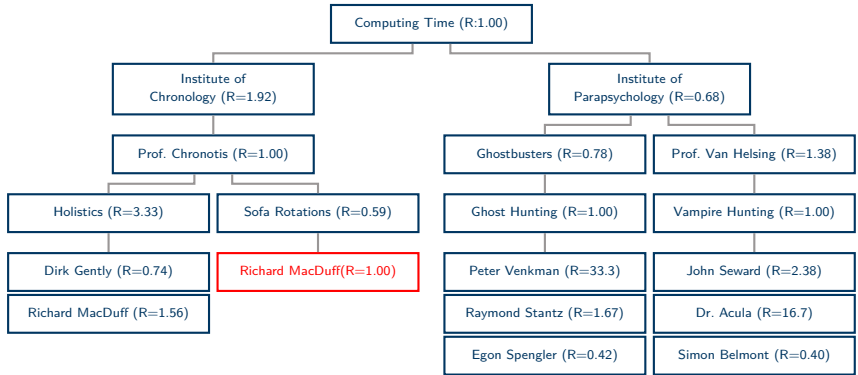




McDuff(Holistics) > Gently

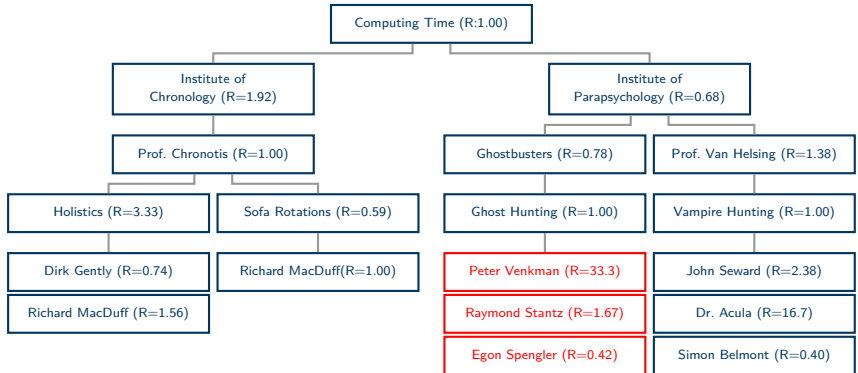


McDuff(Holistics) > Gently



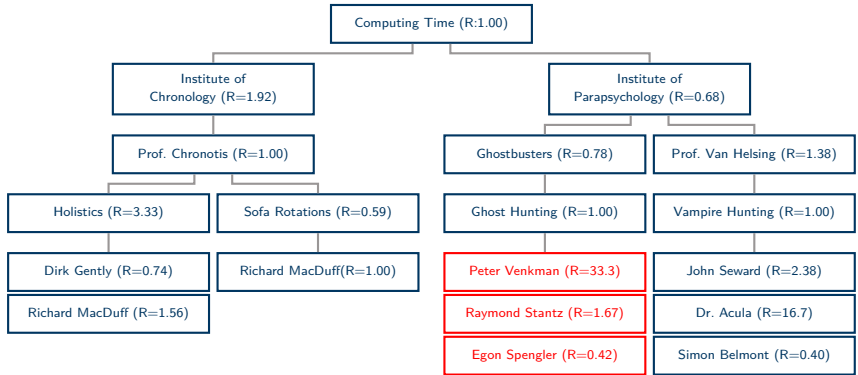
McDuff(Holistics) > Gently

McDuff(SofaRot)



McDuff(Holistics) > Gently

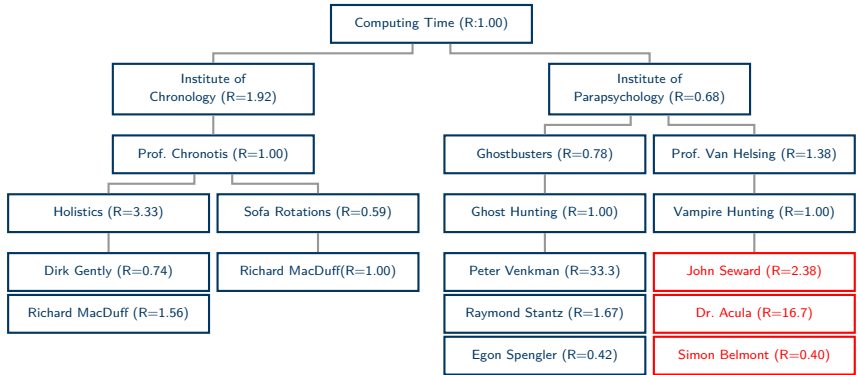
McDuff(SofaRot)



McDuff(Holistics) > Gently

McDuff(SofaRot)

Venkman > Stantz > Spengler

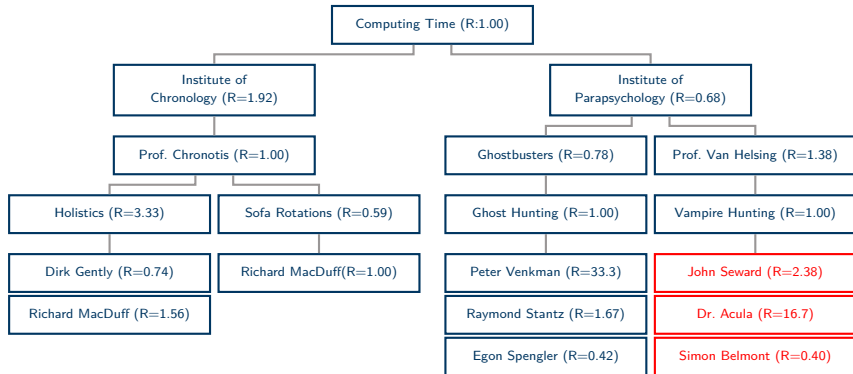


McDuff(Holistics) > Gently

McDuff(SofaRot)

Venkman > Stantz > Spengler



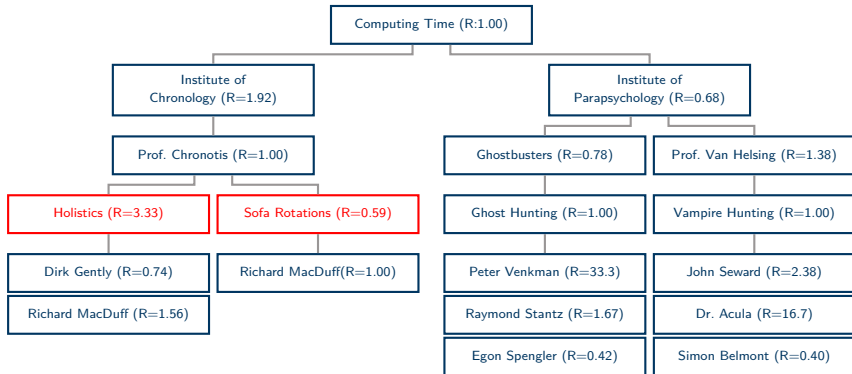


McDuff(Holistics) > Gently

McDuff(SofaRot)

Venkman > Stantz > Spengler

Acula > Seward > Belmont

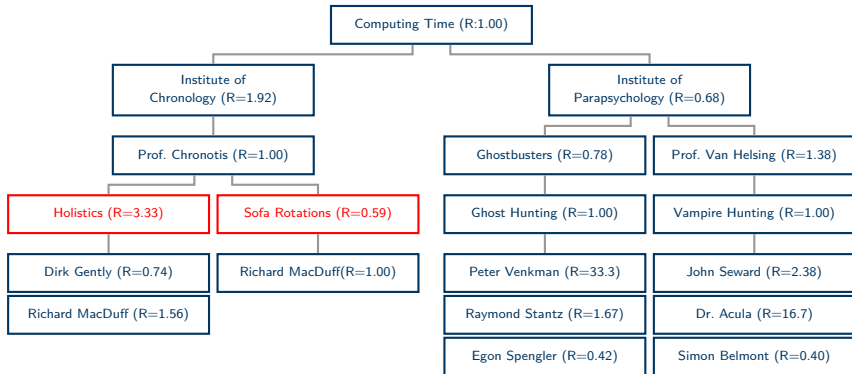


McDuff(Holistics) > Gently

McDuff(SofaRot)

Venkman > Stantz > Spengler

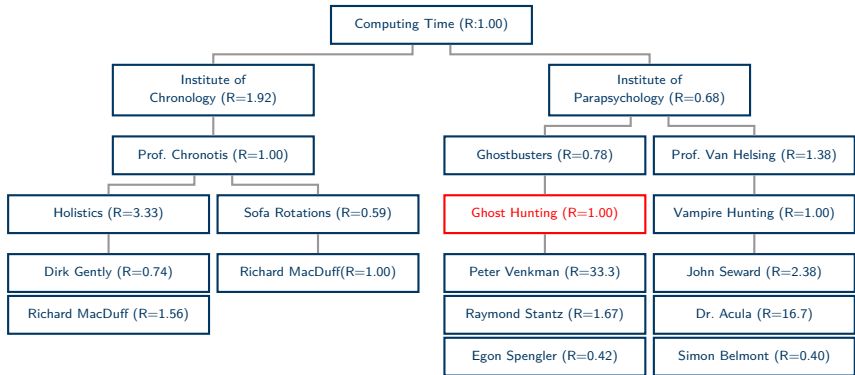
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

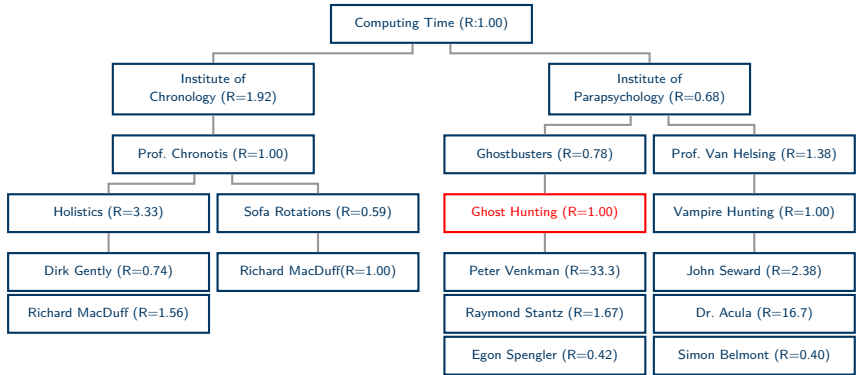
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

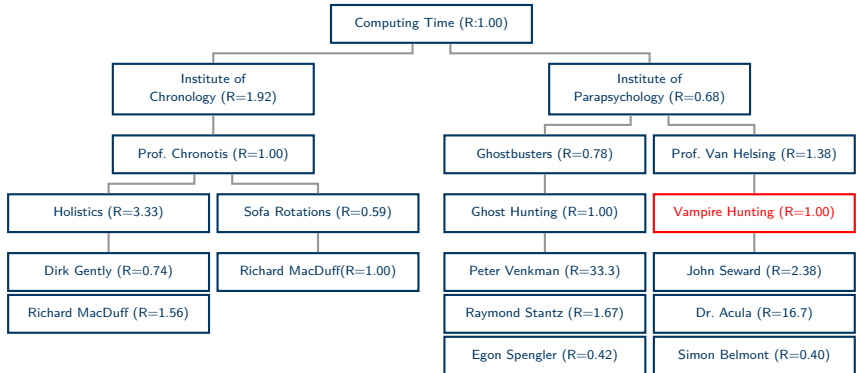
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

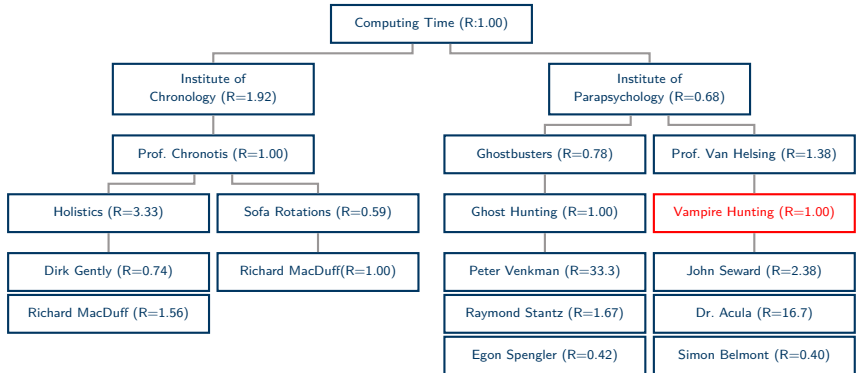
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

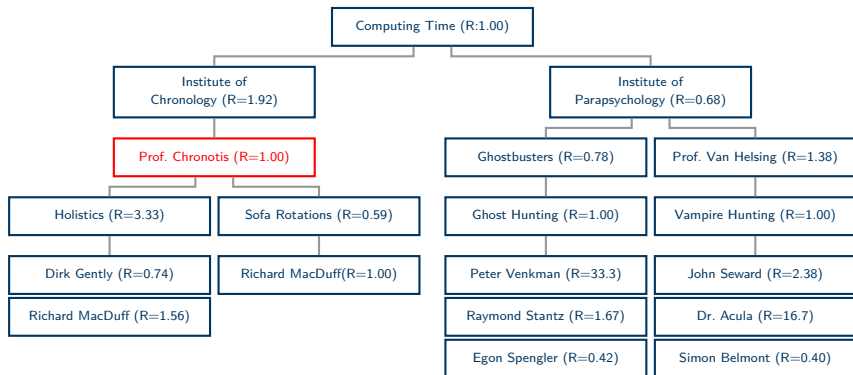
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

Acula > Seward > Belmont

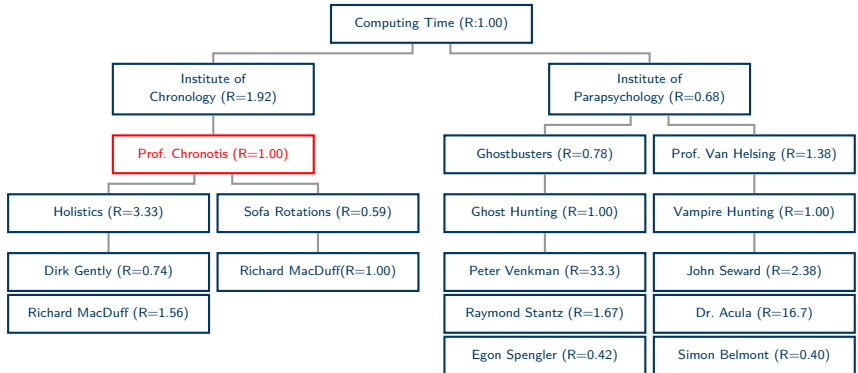


McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

Acula > Seward > Belmont

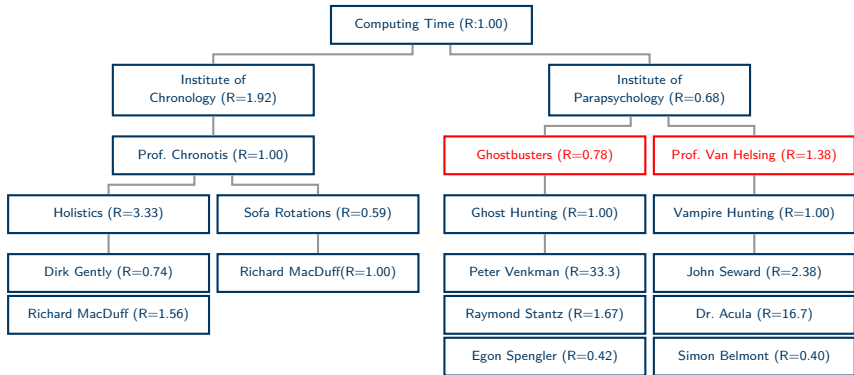




McDuff(Holistics) > Gently > McDuff(SofaRot)

Venkman > Stantz > Spengler

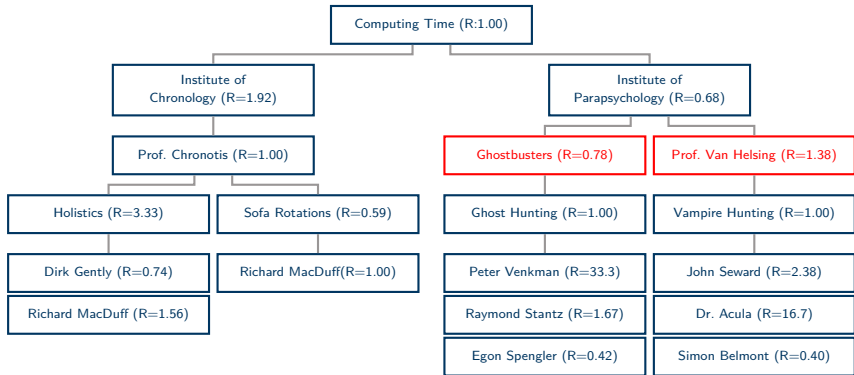
Acula > Seward > Belmont



McDuff(Holistics) > Gently > McDuff(SofaRot)

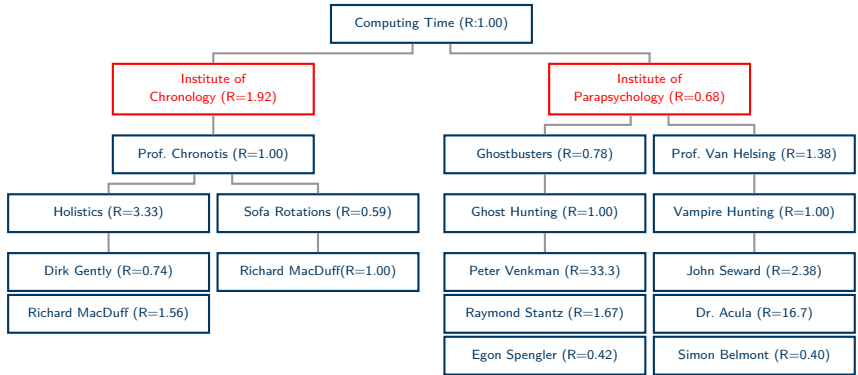
Venkman > Stantz > Spengler

Acula > Seward > Belmont



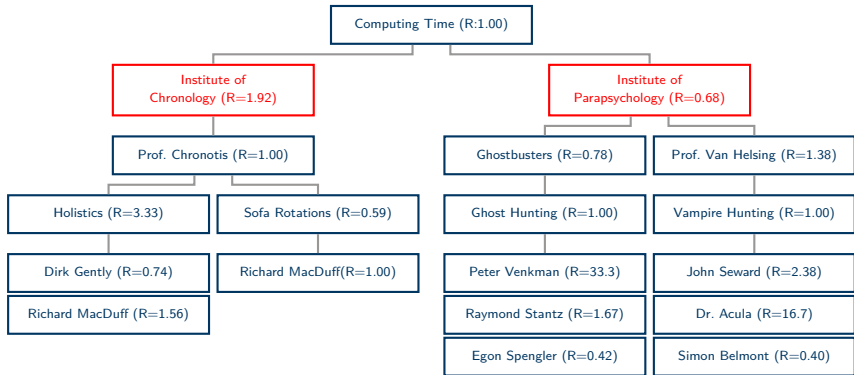
McDuff(Holistics) > Gently > McDuff(SofaRot)

Acula > Seward > Belmont > Venkman > Stantz > Spengler

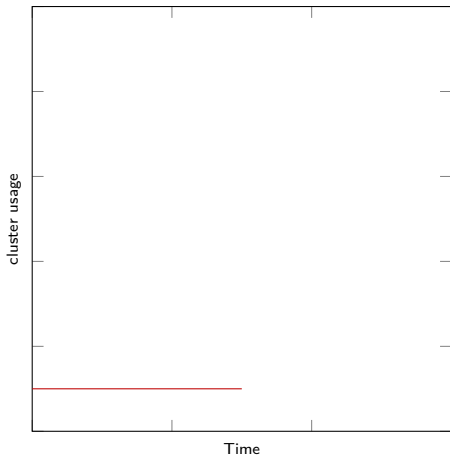


McDuff(Holistics) > Gently > McDuff(SofaRot)

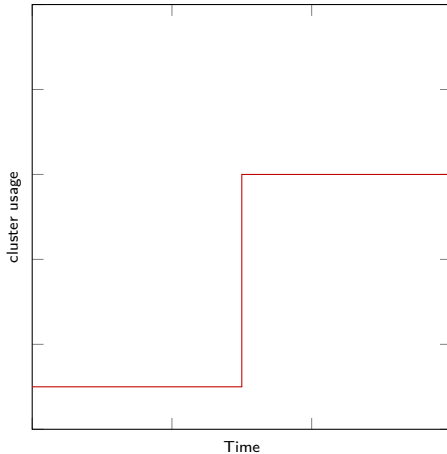
Acula > Seward > Belmont > Venkman > Stantz > Spengler



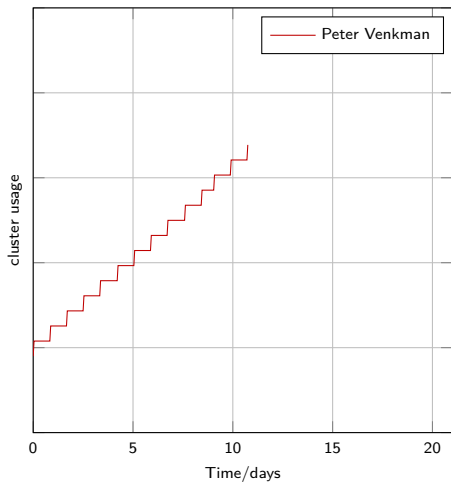
McDuff(Holistics) > Gently > McDuff(SofaRot) >  
Acula > Seward > Belmont > Venkman > Stantz > Spengler



The user's rawUsage is tracked by SLURM over time.

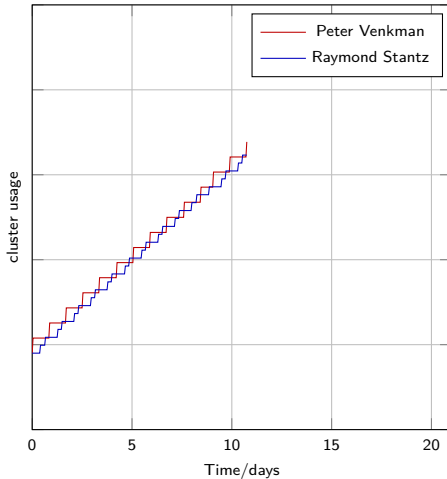


If a job is executed the jobs runtime (multiplied with the number cores/GPUs) is added to the users `rawUsage`.

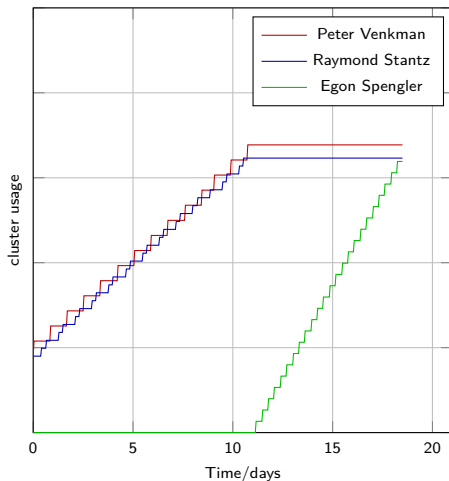


Example: Peter Venkman  
submits jobs.  
The runtime is added to his  
rawUsage.

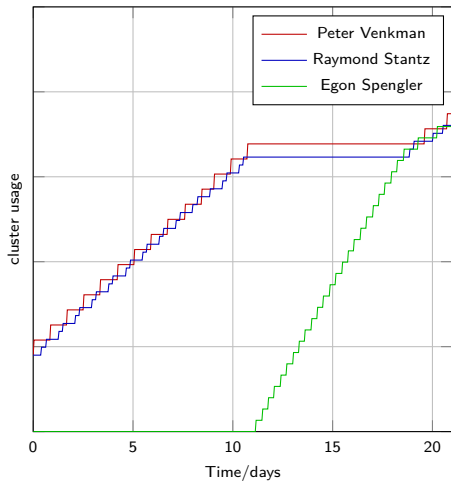




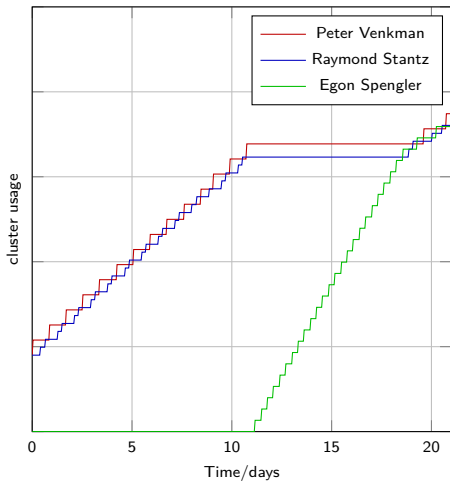
Raymond Stantz also uses the cluster  
Their rawUsage is similar, their jobs take turns. The cluster is shared fairly.



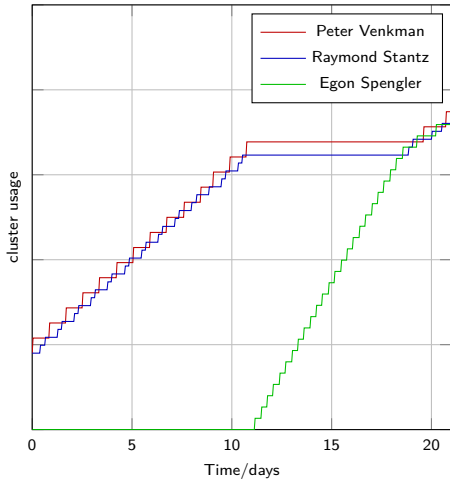
Egon Spengler starts to use the cluster.  
His `rawUsage` is lowest, so he takes over and forces Peter's and Raymond's jobs to wait.



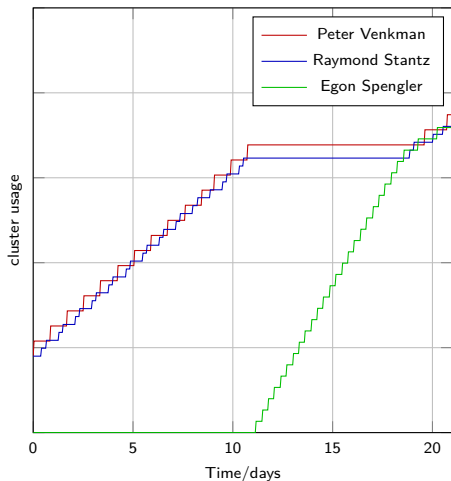
Finally Egon's rawUsage caught up with Peter's and Raymond's and everyone shares the cluster evenly.



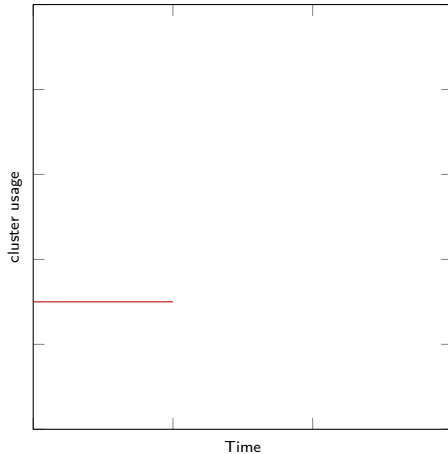
- All of a user's usage history is considered to prioritize jobs.



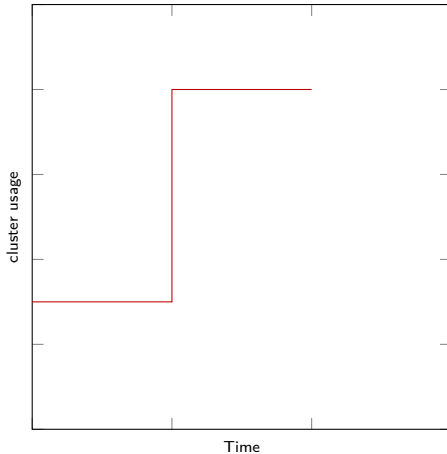
- All of a user's usage history is considered to prioritize jobs.
- This is unfair, especially as new users/groups could take over the complete cluster and block everyone until the rawUsage caught up with everyone else.



- All of a user's usage history is considered to prioritize jobs.
- This is unfair, especially as new users/groups could take over the complete cluster and block everyone until the rawUsage caught up with everyone else.
- Some way to “forget” the past is needed to ensure fair usage.

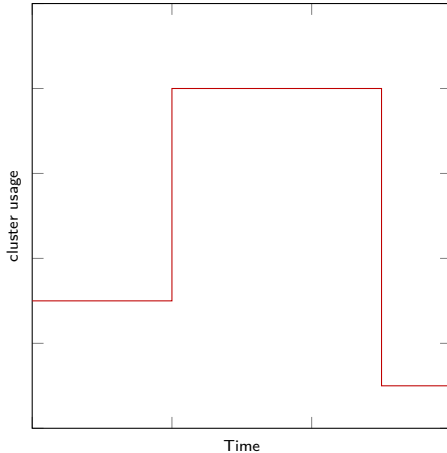


The user's rawUsage is tracked by SLURM over time.

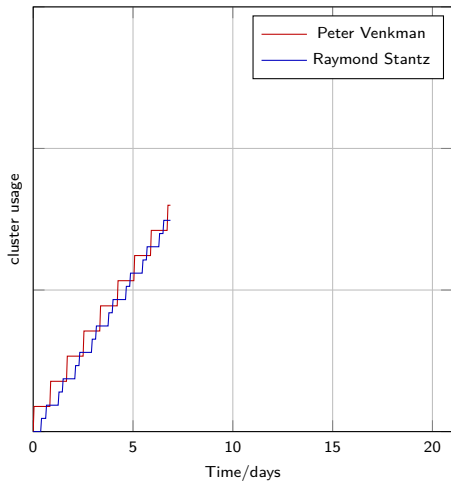


If a job is executed the jobs runtime (multiplied with the number of nodes) is added to the users rawUsage.

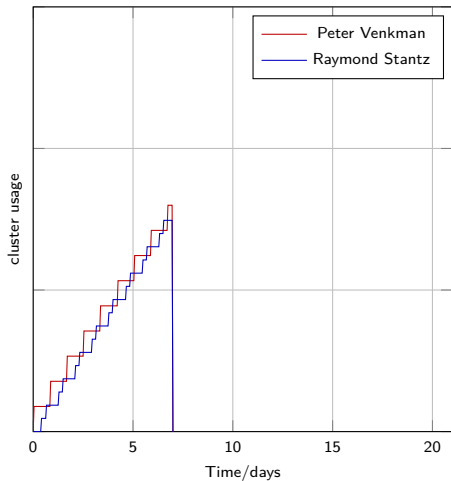




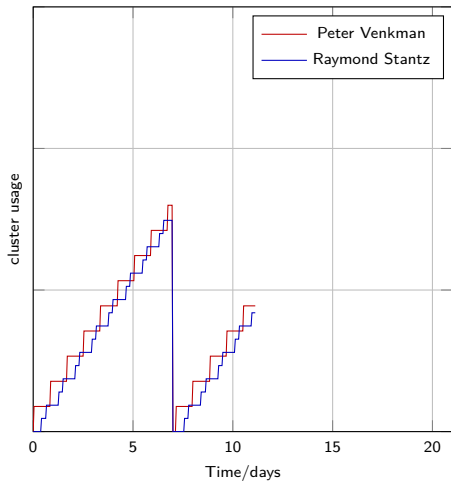
In regular intervals the `rawUsage` of all users is set to zero in order to ignore distant usage history.



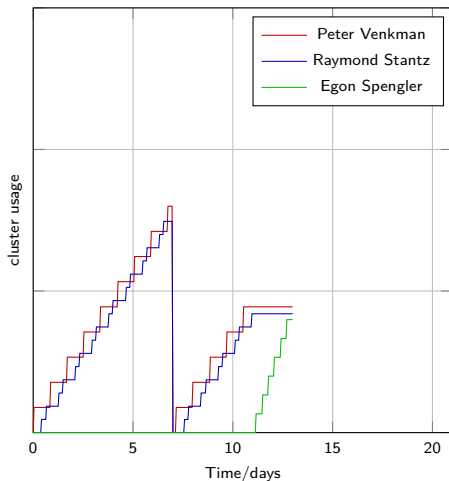
Example: Peter Venkman and Raymond Stantz share the cluster evenly.



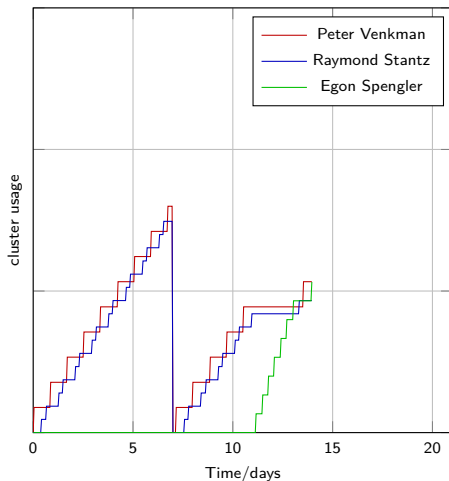
A reset point is reached and everyone's rawUsage is cleared.



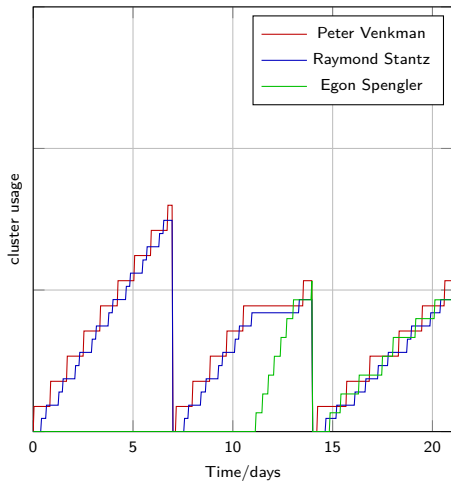
Peter and Raymond still share the cluster evenly.



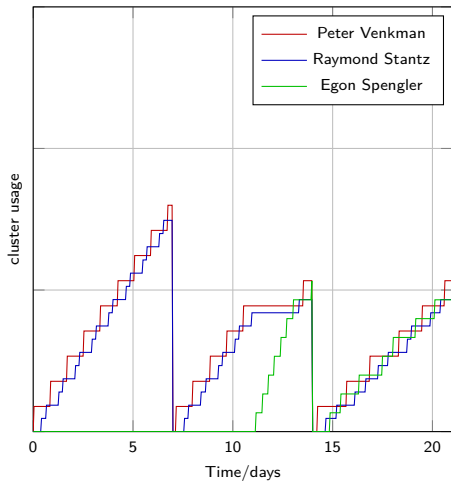
Egon Spengler starts to use the cluster.  
His `rawUsage` is lowest, so he takes over and forces Peter's and Raymond's jobs to wait.



Egon's rawUsage overtakes Raymond's and Peter's rawUsage and they share the cluster fairly.

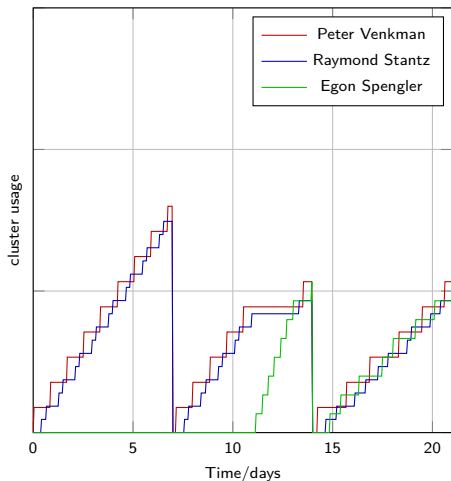


The next reset has no influence on the fair sharing of the cluster.

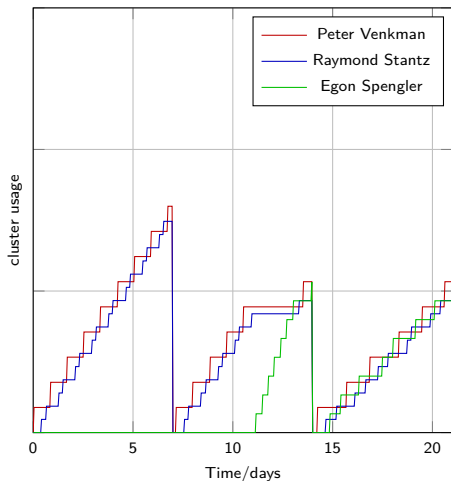


- New users cannot take over the cluster for extended periods of time.

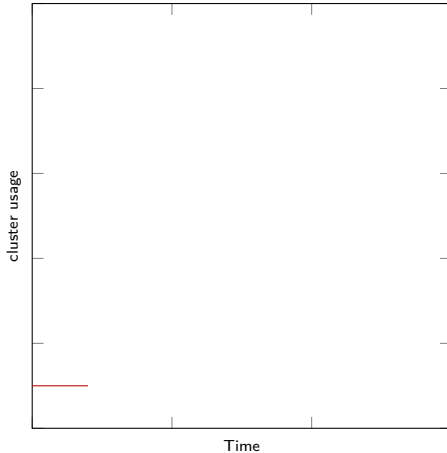




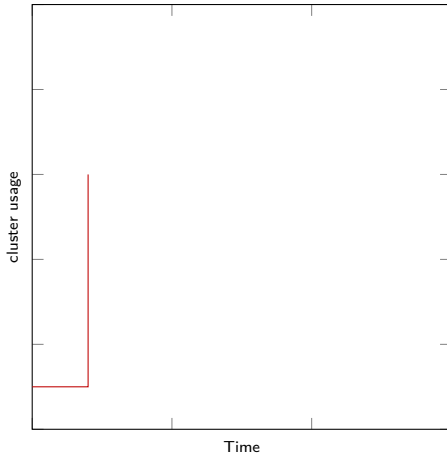
- New users cannot take over the cluster for extended periods of time.
- However, new users can aim at maximizing cluster blockage by choosing a clever starting point.



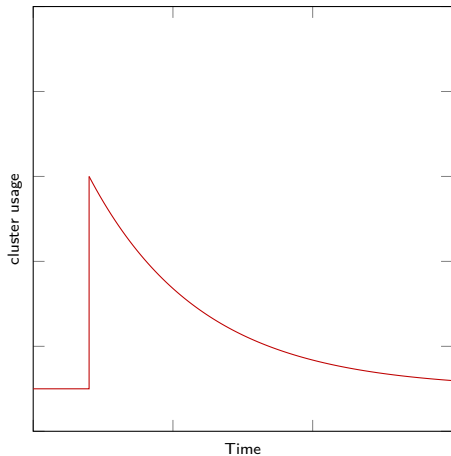
- New users cannot take over the cluster for extended periods of time.
- However, new users can aim at maximizing cluster blockage by choosing a clever starting point.
- Some way to make the distant past less relevant than the recent past is needed to ensure fair usage.



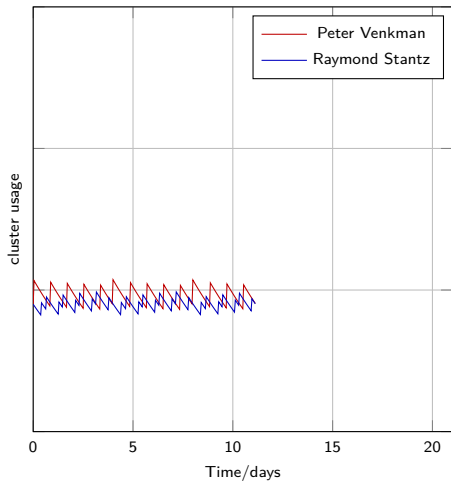
The user's rawUsage is tracked by SLURM over time.



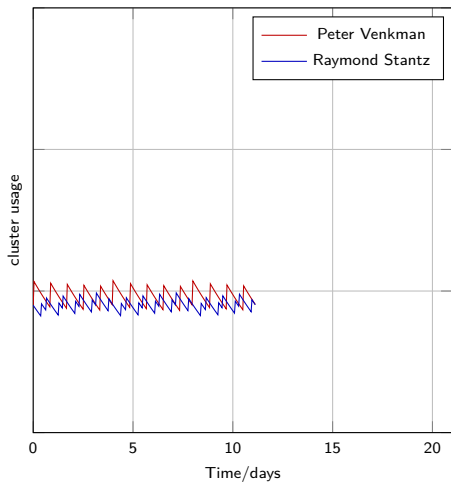
If a job is executed the jobs runtime (multiplied with the number of nodes) is added to the users rawUsage.



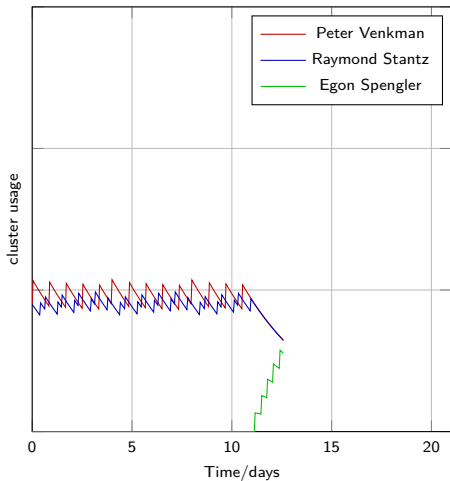
`rawUsage` is subject to constant exponential decay.



Example: Peter Venkman and Raymond Stantz share the cluster evenly.

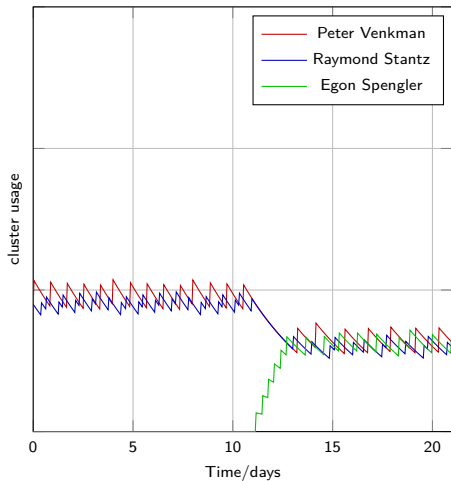


The exponential decay is already visible and ensures that the raw usage does not grow indefinitely.

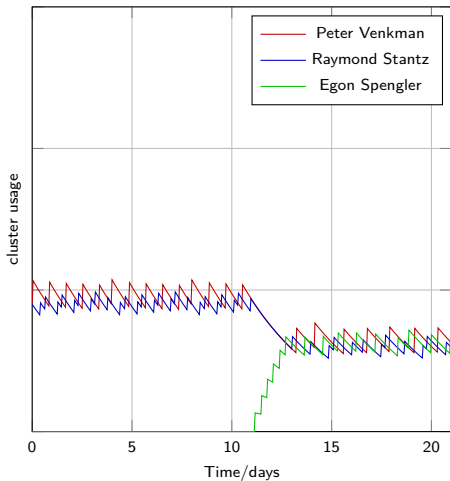


Egon Spengler starts to use the cluster.  
His `rawUsage` is lowest, so he takes over and forces Peter's and Raymond's jobs to wait.

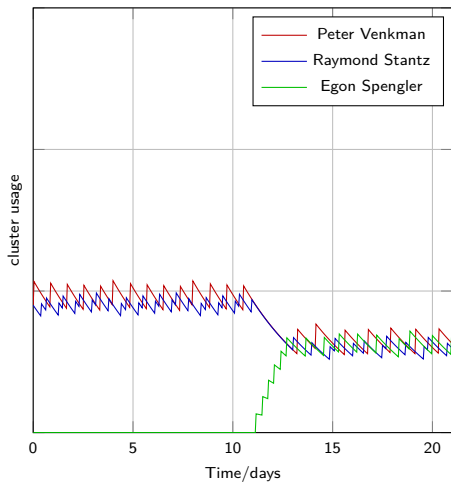




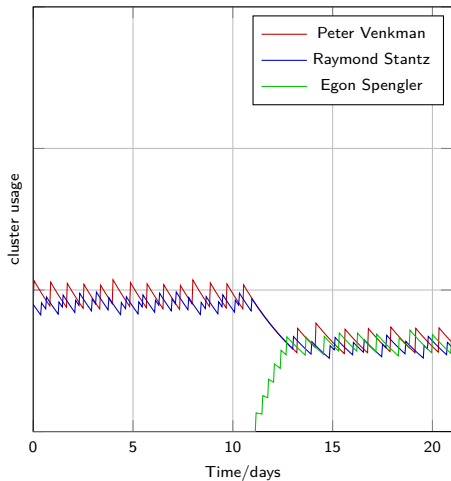
Egon's rawUsage quickly overtakes Raymond's and Peter's rawUsage and they share the cluster fairly.



- Distant usage history is dampened and recent usage history preserved.



- Distant usage history is dampened and recent usage history preserved.
- New users cannot take over the cluster for extended periods of time.



- Distant usage history is dampened and recent usage history preserved.
- New users cannot take over the cluster for extended periods of time.
- Users are not punished for taking advantage of an empty cluster.

- 1 Check the available partitions/queues with `sinfo`
- 2 Start an interactive session with `srun`, and exit again
- 3 Read the `sample_job_sleeper.sh` script and fill in all `FIX_ME`
- 4 Submit the `sample_job_sleeper.sh`
- 5 Check the jobs status with `squeue`

## 1 Check the available partitions/queues with sinfo

### Terminal

```
alice@hpc:~$ sinfo
PARTITION      AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu-nodes      up    2-00:00:00    4    idle  hpc-dev-cpu[01-04]
gpu-nodes      up    2-00:00:00    2    idle  hpc-dev-gpu[01-02]
alice@hpc:~$
```

### 2 Start an interactive session with `srun`, and exit again

#### Terminal

```
alice@hpc:$ srun --partition=cpu-nodes --pty /bin/bash
alice@hpc-cpu01:$ hostname
hpc-dev-cpu01
alice@hpc-cpu01:$ exit
alice@hpc:$
```

### 3 Read the `sample_job_sleeper.sh` script and fill in all `FIX_ME`

#### Terminal

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --partition=          <= FIX_ME
#SBATCH --job-name=          <= FIX_ME
#SBATCH --time=hh:mm:ss     <= FIX_ME
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

# Do nothing for 10 seconds
sleep 10s
```



### 3 Read the `sample_job_sleeper.sh` script and fill in all `FIX_ME`

#### Terminal

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --partition=cpu-nodes
#SBATCH --job-name=                <= FIX_ME
#SBATCH --time=hh:mm:ss           <= FIX_ME
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

# Do nothing for 10 seconds
sleep 10s
```

### 3 Read the `sample_job_sleeper.sh` script and fill in all `FIX_ME`

#### Terminal

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --partition=cpu-nodes
#SBATCH --job-name=sleeper
#SBATCH --time=hh:mm:ss           <= FIX_ME
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

# Do nothing for 10 seconds
sleep 10s
```

3 Read the `sample_job_sleeper.sh` script and fill in all `FIX_ME`

### Terminal

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --partition=cpu-nodes
#SBATCH --job-name=sleeper
#SBATCH --time=00:05:00
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

# Do nothing for 10 seconds
sleep 10s
```

### 4 Submit the `sample_job_sleepter.sh`

#### Terminal

```
alice@hpc:$ sbatch sample_job_sleepter.sh
Submitted batch job 13491
alice@hpc:$
```

## 5 Check the jobs status with squeue

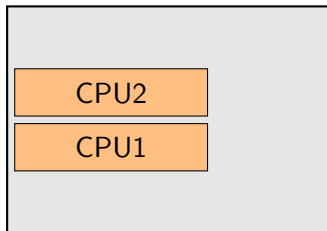
### Terminal

```
alice@hpc:$ squeue
JOBID PARTITION   NAME     USER ST       TIME  NODES NODELIST(REASON)
13491 cpu-nodes  sleeper  alice R        0:07      1 hpc-cpu01
alice@hpc:$
```

# Running Programs in Parallel

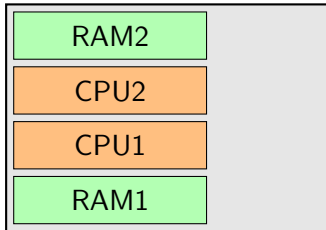


- The most important compute node components:

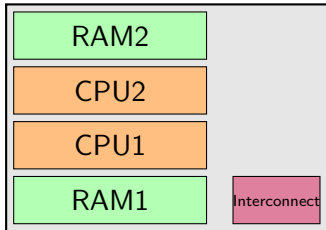


- The most important compute node components:
  - CPUs

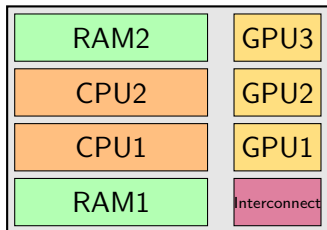




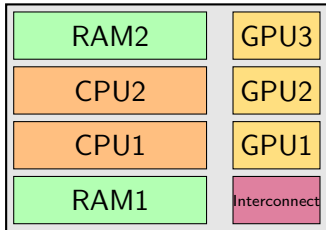
- The most important compute node components:
  - CPUs
  - Memory



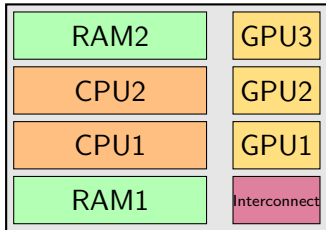
- The most important compute node components:
  - CPUs
  - Memory
  - Interconnect



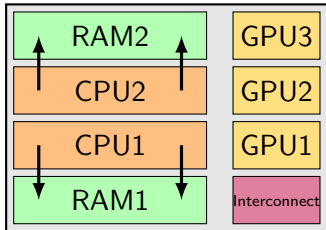
- The most important compute node components:
  - CPUs
  - Memory
  - Interconnect
  - Accelerator(GPU)



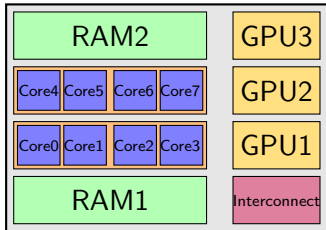
- CPU:



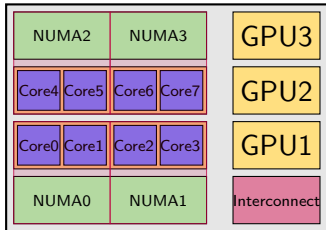
- CPU:
  - sits in a socket



- CPU:
  - sits in a socket
  - only part of memory directly addressable

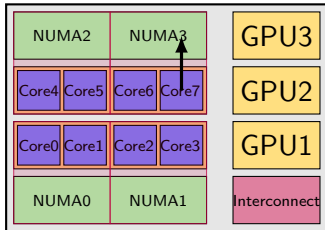


- CPU:
  - sits in a socket
  - only part of memory directly addressable
  - several (4-128) cores



- CPU:
  - sits in a socket
  - only part of memory directly addressable
  - several (4-128) cores
  - cores and memory portions are grouped (NUMA)



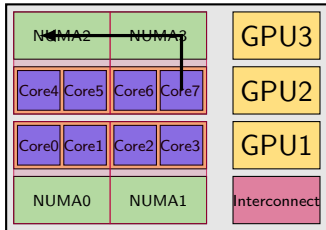


---

Data Location	Latency
Within NUMA	~20 ns

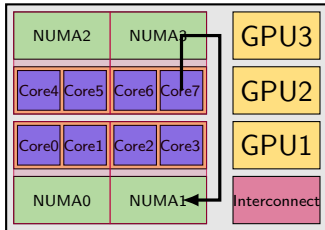
---

To ensure minimal latencies  
pin processes to cores.



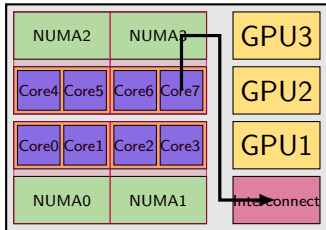
Data Location	Latency
Within NUMA	~20 ns
Within Socket	~100 ns

To ensure minimal latencies  
pin processes to cores.



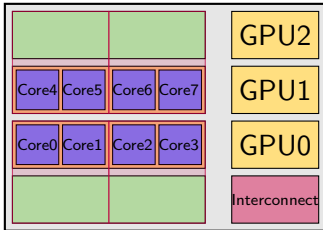
Data Location	Latency
Within NUMA	~20 ns
Within Socket	~100 ns
Other Socket	~200 ns

To ensure minimal latencies  
pin processes to cores.



Data Location	Latency
Within NUMA	~20 ns
Within Socket	~100 ns
Other Socket	~200 ns
Other Node	~1200 ns

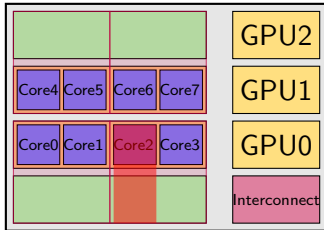
To ensure minimal latencies  
pin processes to cores.



Terminal

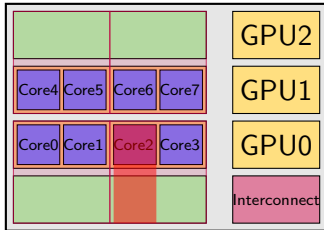
```
alice@hpc:$
```

- A program is executed on a “randomly” selected core



Terminal

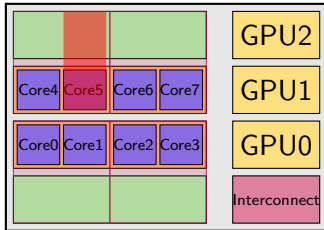
```
alice@hpc:~$ ./myprog.x
```



- A program is executed on a “randomly” selected core
- It accesses memory in its NUMA region

## Terminal

```
alice@hpc:~$ ./myprog.x
```

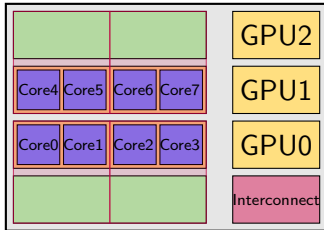


- A program is executed on a “randomly” selected core
- It accesses memory in its NUMA region
- The core assignment might switch during execution

## Terminal

```
alice@hpc:~$ ./myprog.x
```



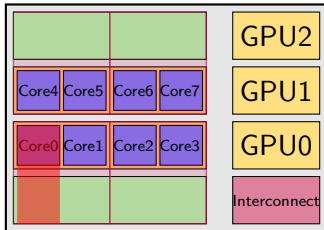


- A program is executed on a “randomly” selected core
- It accesses memory in its NUMA region
- The core assignment might switch during execution
- The terminal awaits its completion before it returns

## Terminal

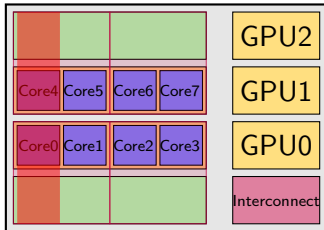
```
alice@hpc:~$ ./myprog.x
alice@hpc:~$
```

- An “&” starts the process in the background and returns immediately



## Terminal

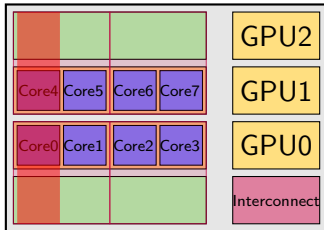
```
alice@hpc:~$ ./myprog.x &  
alice@hpc:~$
```



- An “&” starts the process in the background and returns immediately
- This allows for multiple parallel program executions

## Terminal

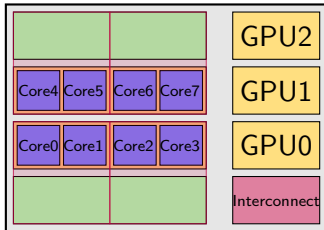
```
alice@hpc:~$ ./myprog.x &  
alice@hpc:~$ ./myprog.x &  
alice@hpc:~$
```



- An “&” starts the process in the background and returns immediately
- This allows for multiple parallel program executions
- `wait` returns as soon as all background processes have returned

## Terminal

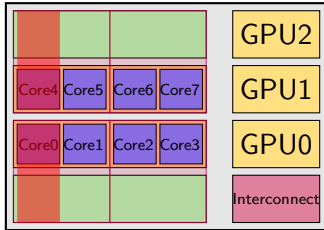
```
alice@hpc:~$ ./myprog.x &
alice@hpc:~$ ./myprog.x &
alice@hpc:~$ wait
```



- An “&” starts the process in the background and returns immediately
- This allows for multiple parallel program executions
- `wait` returns as soon as all background processes have returned

## Terminal

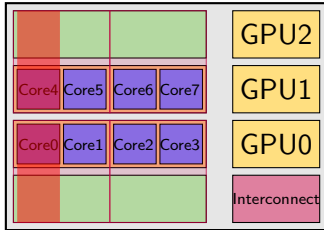
```
alice@hpc:~$ ./myprog.x &
alice@hpc:~$ ./myprog.x &
alice@hpc:~$ wait
alice@hpc:~$
```



- `taskset -c <coreid>`  
pins a process to a core  
and prevents core switching

## Terminal

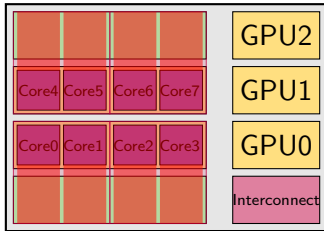
```
alice@hpc:$ taskset -c 0 ./myprog.x &  
alice@hpc:$ taskset -c 4 ./myprog.x &  
alice@hpc:$ wait
```



- `taskset -c <coreid>`  
pins a process to a core  
and prevents core switching
- this allows for optimal  
process placement for  
memory access

## Terminal

```
alice@hpc:~$ taskset -c 0 ./myprog.x &  
alice@hpc:~$ taskset -c 4 ./myprog.x &  
alice@hpc:~$ wait
```

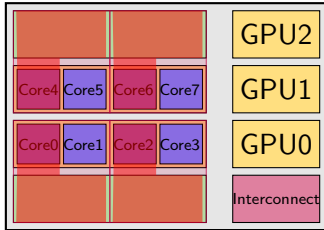


- Loops can be used to start and pin huge amounts of processes

## Terminal

```
alice@hpc:$ for i in $(seq 0 17); do taskset -c ${i} ./myprog.x & done
alice@hpc:$ wait
```





- Loops can be used to start and pin huge amounts of processes
- For memory intensive jobs cores can be skipped

## Terminal

```
alice@hpc:$ for i in $(seq 0 2 7); do taskset -c ${i} ./myprog.x & done
alice@hpc:$ wait
```

- 1 Read the `farming_job.sh` script and fill in all `FIX_ME`
- 2 Read the farming outputfile and check the pinning

1 Read the `farming_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
# loop to pin processes to different cores  
for icore in $(seq <FIX_ME>)  
do  
    taskset -c <FIX_ME> ${pinnalizer} > farming_${icore}.tmpout &  
done  
...
```

1 Read the `farming_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
# loop to pin processes to different cores
for icore in $(seq 4)
do
    taskset -c <FIX_ME> ${pinnalizer} > farming_${icore}.tmpout &
done
...
```

1 Read the `farming_job.sh` script and fill in all `FIX_ME`

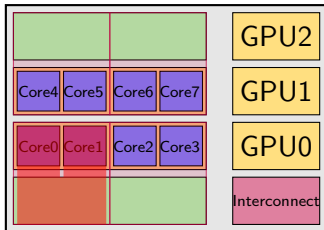
### Terminal

```
...
# loop to pin processes to different cores
for icore in $(seq 4)
do
    taskset -c ${icore} ${pinnalizer} > farming_${icore}.tmpout &
done
...
```

### 2 Read the farming outputfile and check the pinning

#### Terminal

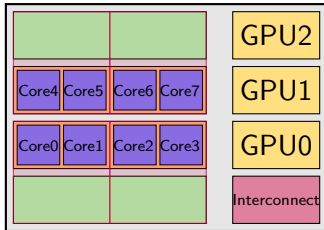
```
alice@hpc:$ cat farming-1234.out
Running process 0/1 on cpu 1/48 on hpc-cpu01
Running process 0/1 on cpu 2/48 on hpc-cpu01
Running process 0/1 on cpu 3/48 on hpc-cpu01
Running process 0/1 on cpu 4/48 on hpc-cpu01
```



- In shared memory parallelized (or threaded) programs the process can spawn threads running on other cores, but accessing the same memory regions

Terminal

```
alice@hpc:~$ ./myprog.x
```

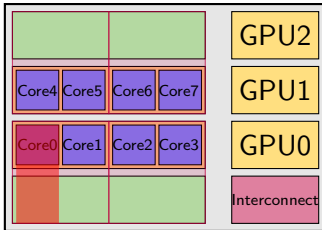


- In shared memory parallelized (or threaded) programs the process can spawn threads running on other cores, but accessing the same memory regions
- Threading is usually done via [OpenMP](#) or [pthreads](#)

Terminal

```
alice@hpc:$
```

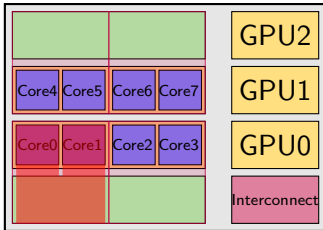




- In pthreaded programs the threading behaviour is part of the program

## Terminal

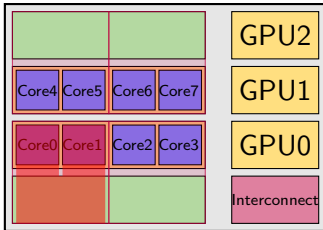
```
alice@hpc:~$ ./my_single_thread_prog.x
```



- In pthreaded programs the threading behaviour is part of the program

## Terminal

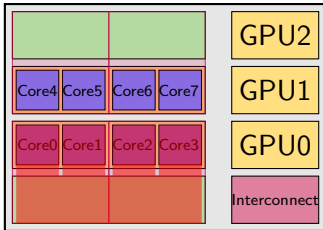
```
alice@hpc:~$ ./my_dual_thread_prog.x
```



- In pthreaded programs the threading behaviour is part of the program

## Terminal

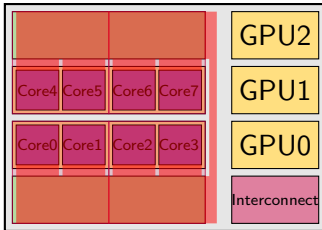
```
alice@hpc:~$ taskset -c 0,1 ./my_dual_thread_prog.x
```



- In pthreaded programs the threading behaviour is part of the program

## Terminal

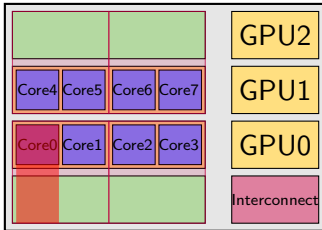
```
alice@hpc:~$ ./my_quad_thread_prog.x
```



- In pthreaded programs the threading behaviour is part of the program

## Terminal

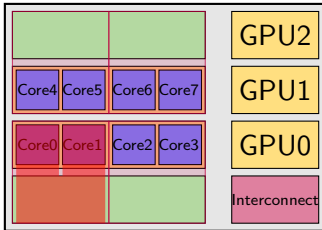
```
alice@hpc:~$ ./my_oct_thread_prog.x
```



- OMP uses the `OMP_NUM_THREADS` environment variable to set the number of threads

## Terminal

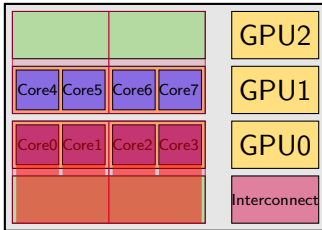
```
alice@hpc:$ export OMP_NUM_THREADS=1
alice@hpc:$ ./my_omp_prog.x
```



- OMP uses the `OMP_NUM_THREADS` environment variable to set the number of threads

## Terminal

```
alice@hpc:$ export OMP_NUM_THREADS=2
alice@hpc:$ ./my_omp_prog.x
```

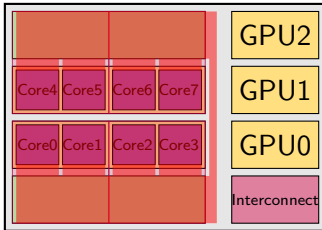


- OMP uses the `OMP_NUM_THREADS` environment variable to set the number of threads

## Terminal

```
alice@hpc:$ export OMP_NUM_THREADS=4
alice@hpc:$ ./my_omp_prog.x
```





- OMP uses the `OMP_NUM_THREADS` environment variable to set the number of threads

## Terminal

```
alice@hpc:$ export OMP_NUM_THREADS=8
alice@hpc:$ ./my_omp_prog.x
```

- 1 Read the `shared_mem_job.sh` script and fill in all `FIX_ME`
- 2 Read the `shared-mem` outputfile and check the pinning

1 Read the `shared_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
# Define number of threads  
export OMP_NUM_THREADS=          <= FIX_ME  
...
```

1 Read the `shared_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

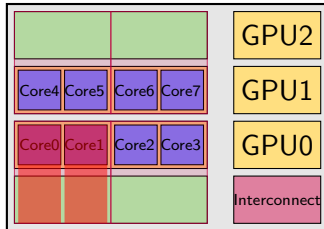
```
...  
# Define number of threads  
export OMP_NUM_THREADS=4  
...
```

### 2 Read the shared-mem outputfile and check the pinning

#### Terminal

```
alice@hpc:$ cat shared_mem-1234.out
Running process 0/1, thread 0/4 on cpu 2/48 on hpc-cpu01
Running process 0/1, thread 1/4 on cpu 0/48 on hpc-cpu01
Running process 0/1, thread 2/4 on cpu 3/48 on hpc-cpu01
Running process 0/1, thread 3/4 on cpu 8/48 on hpc-cpu01
```

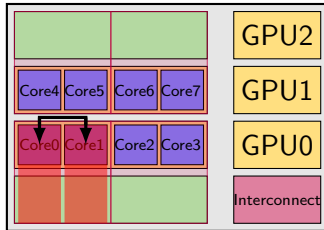
- In distributed memory parallelized programs several processes are started without direct access to the other processes memory



Terminal

```
alice@hpc:$  
alice@hpc:$
```

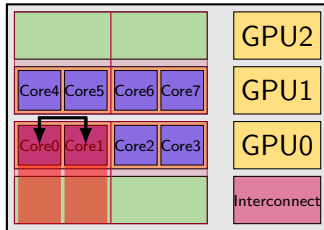
- Exchange of data between processes is usually done via the Message Passing Interface (MPI)



Terminal

```
alice@hpc:$  
alice@hpc:$
```

- Most common MPI implementations are [OpenMPI](#) and [IntelMPI](#)

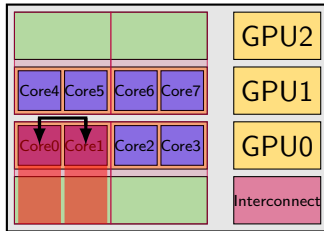


Terminal

```
alice@hpc:$  
alice@hpc:$
```



- `mpirun` is a wrapper that manages process spawning and the communication channels between them

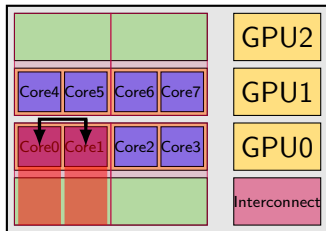


## Terminal

```
alice@hpc:$ mpirun -np 2 ./myprog.x
alice@hpc:$ mpirun -n 2 ./myprog.x
```

```
#OpenMPI
#IntelMPI
```

- OpenMPI: `-np <n>`  $\Rightarrow$  number of processes
- IntelMPI: `-n <n>`  $\Rightarrow$  number of processes

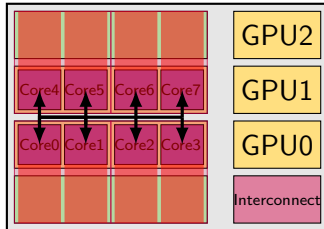


## Terminal

```
alice@hpc:~$ mpirun -np 2 ./myprog.x
alice@hpc:~$ mpirun -n 2 ./myprog.x
```

```
#OpenMPI
#IntelMPI
```

- OpenMPI: `-np <n>`  $\Rightarrow$  number of processes
- IntelMPI: `-n <n>`  $\Rightarrow$  number of processes

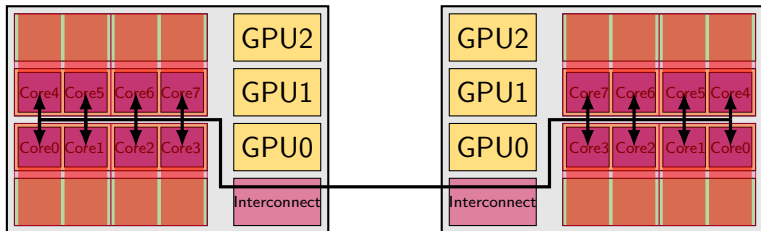


## Terminal

```
alice@hpc:~$ mpirun -np 8 ./myprog.x
alice@hpc:~$ mpirun -n 8 ./myprog.x
```

```
#OpenMPI
#IntelMPI
```

- OpenMPI: `--map-by ppr:<n>:node`  $\Rightarrow$  processes per node
- IntelMPI: `-ppn <n>`  $\Rightarrow$  processes per node

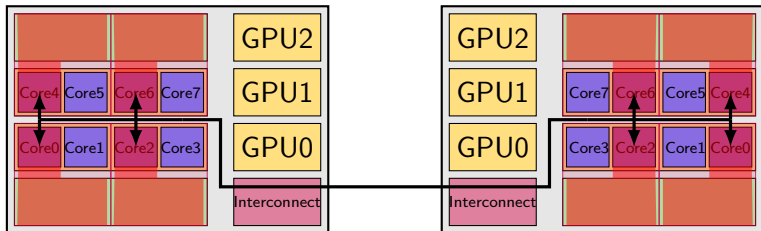


## Terminal

```
alice@hpc:~$ mpirun -np 16 --map-by ppr:8:node ./myprog.x
alice@hpc:~$ mpirun -n 16 -ppn 8 ./myprog.x
```

```
#OpenMPI
#IntelMPI
```

- OpenMPI: `--map-by ppr:<n>:node`  $\Rightarrow$  processes per node
- IntelMPI: `-ppn <n>`  $\Rightarrow$  processes per node



## Terminal

```
alice@hpc:~$ mpirun -np 8 --map-by ppr:4:node ./myprog.x
alice@hpc:~$ mpirun -n 8 -ppn 4 ./myprog.x
```

```
#OpenMPI
#IntelMPI
```

- 1 Read the `distributed_mem_job.sh` script and fill in all `FIX_ME`
- 2 Read the `mpi` outputfile and check the pinning

1 Read the `distributed_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
#SBATCH --nodes=                <= FIX_ME  
...  
# execute program with MPI for distributed memory job  
mpirun -np <FIX_ME> --map-by ppr:<FIX_ME>:node ${pinnalizer}
```

1 Read the `distributed_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
#SBATCH --nodes=2  
...  
# execute program with MPI for distributed memory job  
mpirun -np <FIX_ME> --map-by ppr:<FIX_ME>:node ${pinnalizer}
```



1 Read the `distributed_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
#SBATCH --nodes=2  
...  
# execute program with MPI for distributed memory job  
mpirun -np 4 --map-by ppr:<FIX_ME>:node ${pinnalizer}
```

1 Read the `distributed_mem_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
#SBATCH --nodes=2  
...  
# execute program with MPI for distributed memory job  
mpirun -np 4 --map-by ppr:2:node ${pinnalizer}
```

### 2 Read the mpi outputfile and check the pinning

#### Terminal

```
alice@hpc:$ cat distr_mem-1234.out
Running process 0/4 on cpu 0/48 on hpc-cpu01
Running process 1/4 on cpu 2/48 on hpc-cpu01
Running process 2/4 on cpu 0/48 on hpc-cpu02
Running process 3/4 on cpu 2/48 on hpc-cpu02
```

- 1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`
- 2 Read the hybrid outputfile and check the pinning

## 1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --nodes=                <= FIX_ME
...
# Define number of threads
export OMP_NUM_THREADS=         <= FIX_ME
# execute program with MPI for distributed memory job
mpirun -np <FIX_ME> --map-by ppr:<FIX_ME>:node ${pinnalizer}
```

## 1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --nodes=2
...
# Define number of threads
export OMP_NUM_THREADS=                <= FIX_ME
# execute program with MPI for distributed memory job
mpirun -np <FIX_ME> --map-by ppr:<FIX_ME>:node ${pinnalizer}
```

1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --nodes=2
...
# Define number of threads
export OMP_NUM_THREADS=2
# execute program with MPI for distributed memory job
mpirun -np <FIX_ME> --map-by ppr:<FIX_ME>:node ${pinnalizer}
```

1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --nodes=2
...
# Define number of threads
export OMP_NUM_THREADS=2
# execute program with MPI for distributed memory job
mpirun -np 4 --map-by ppr:<FIX_ME>:node ${pinnalizer}
```



1 Read the `hybrid_job.sh` script and fill in all `FIX_ME`

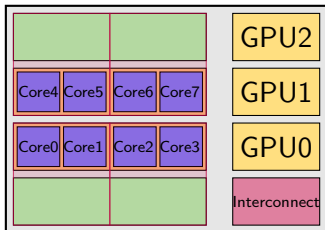
### Terminal

```
...  
#SBATCH --nodes=2  
...  
# Define number of threads  
export OMP_NUM_THREADS=2  
# execute program with MPI for distributed memory job  
mpirun -np 4 --map-by ppr:2:node ${pinnalizer}
```

## 2 Read the hybrid outputfile and check the pinning

### Terminal

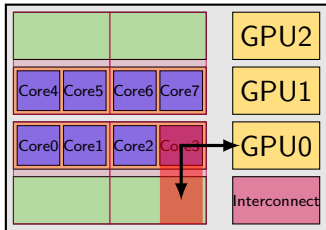
```
alice@hpc:$ cat hybr_mem-1234.out
Running process 0/4, thread 1/2 on cpu 1/48 on hpc-cpu01
Running process 0/4, thread 0/2 on cpu 0/48 on hpc-cpu01
Running process 1/4, thread 0/2 on cpu 3/48 on hpc-cpu01
Running process 1/4, thread 1/2 on cpu 2/48 on hpc-cpu01
Running process 2/4, thread 1/2 on cpu 1/48 on hpc-cpu02
Running process 2/4, thread 0/2 on cpu 0/48 on hpc-cpu02
Running process 3/4, thread 0/2 on cpu 3/48 on hpc-cpu02
Running process 3/4, thread 1/2 on cpu 2/48 on hpc-cpu02
```



- The GPU acceleration is part of the CUDA program.

Terminal

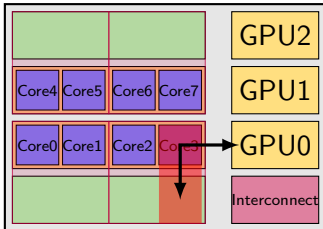
```
alice@hpc:$
```



- The GPU acceleration is part of the CUDA program.
- GPUs need a CPU-Host for data transfer, I/O, ...

Terminal

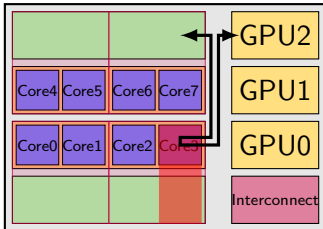
```
alice@hpc:~$ ./my_CUDA_prog.x
```



- The `CUDA_VISIBLE_DEVICES` environment variable selects the GPUs
- GPUs can be closer to one of the CPUs making them better suited for host tasks. Check with `nvidia-smi`

## Terminal

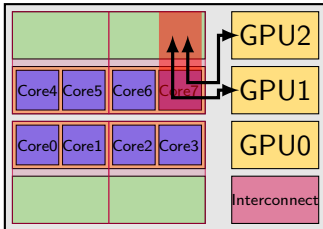
```
alice@hpc:~$ export CUDA_VISIBLE_DEVICES=0
alice@hpc:~$ ./my_CUDA_prog.x
```



- The `CUDA_VISIBLE_DEVICES` environment variable selects the GPUs
- GPUs can be closer to one of the CPUs making them better suited for host tasks. Check with `nvidia-smi`

## Terminal

```
alice@hpc:~$ export CUDA_VISIBLE_DEVICES=2
alice@hpc:~$ ./my_CUDA_prog.x
```



- The `CUDA_VISIBLE_DEVICES` environment variable selects the GPUs
- GPUs can be closer to one of the CPUs making them better suited for host tasks. Check with `nvidia-smi`

## Terminal

```
alice@hpc:~$ export CUDA_VISIBLE_DEVICES=1,2
alice@hpc:~$ ./my_CUDA_prog.x
```

- 1 Read the `cuda_job.sh` script and fill in all `FIX_ME`
- 2 Read the `cuda` outputfile and check the pinning and GPU



1 Read the `cuda_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --partition=                <= FIX_ME
...
# CUDA_VISIBLE_DEVICES is also set by SLURM,
# depending on the allocated gpus
export CUDA_VISIBLE_DEVICES=       <= FIX_ME
...
```

1 Read the `cuda_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...
#SBATCH --partition=gpu
...
# CUDA_VISIBLE_DEVICES is also set by SLURM,
# depending on the allocated gpus
export CUDA_VISIBLE_DEVICES=                <= FIX_ME
...
```

1 Read the `cuda_job.sh` script and fill in all `FIX_ME`

### Terminal

```
...  
#SBATCH --partition=gpu  
...  
# CUDA_VISIBLE_DEVICES is also set by SLURM,  
# depending on the allocated gpus  
export CUDA_VISIBLE_DEVICES=0  
...
```

## 2 Read the cuda outputfile and check the pinning and GPU

### Terminal

```
alice@hpc:$ cat cuda-1234.out
```

```
...
```

```
-----+-----+
| GPU  Name                Persistence-M | ... |
| Fan  Temp    Perf        Pwr:Usage/Cap | ... |
|-----+-----+=====+-----+
|   0  Tesla V100-PCIE-16GB      Off | ... |
| N/A   33C    P0                25W / 250W | ... |
|-----+-----+-----+-----+
| ... |
```

```
...
```

## 2 Read the cuda outputfile and check the pinning and GPU

### Terminal

```
...
+-----+
| Processes:                                     ...
| GPU  GI  CI          PID  Type  Process name  ... GPU Memory
|      ID  ID          |      |      |              | ... Usage
|=====|
| No running processes found                    ...
+-----+
...
```

2 Read the cuda outputfile and check the pinning and GPU

### Terminal

```
...
GPU0   GPU0      CPU Affinity   NUMA Affinity  GPU NUMA ID
        X        0-31,64-95     0                N/A
...
```

- 2 Read the cuda outputfile and check the pinning and GPU (Real life example of a GPU topology map)

## Terminal

```
...
      GPU0   GPU1   GPU2   GPU3   CPU Affinity   NUMA Affinity   GPU NUMA ID
GPU0     X     NV18   NV18   NV18     0-5             0               N/A
GPU1    NV18     X     NV18   NV18    24-29           4               N/A
GPU2    NV18   NV18     X     NV18    48-53           8               N/A
GPU3    NV18   NV18   NV18     X     72-77          12              N/A
```

Legend:

X = Self

...

NV# = Connection traversing a bonded set of # NVLinks

### 2 Read the cuda outputfile and check the pinning and GPU

#### Terminal

```
...  
Running process 0/1 on cpu 79/128 on hpc-gpu01 with 1 X Tesla V100-PCIE-16GB
```



# Software Versions

Alice:  
I need the newest  
GNU-compiler.

Alice:

I need the newest  
GNU-compiler.

Bob:

My CUDA code does  
not work with GNU-  
compilers of version  
> 10.x.

Alice:  
I need the newest  
GNU-compiler.

Carol:  
My OpenMPI needs  
to be tuned to work  
with Intel-compilers.

Bob:  
My CUDA code does  
not work with GNU-  
compilers of version  
> 10.x.

Alice:  
I need the newest

Dave:  
But my OpenMPI  
needs to be tuned  
to work with GNU-  
compilers.

Carol:  
My OpenMPI needs  
to be tuned to work  
with Intel-compilers.

Bob:  
My CUDA code does  
not work with GNU-  
compilers of version  
> 10.x.

Alice:  
I need the

Dave:  
But my OpenMP  
needs to be tuned  
to work with GNU  
compilers.

Carol:  
My OpenMPI needs  
to be tuned to work  
with Intel-compilers.

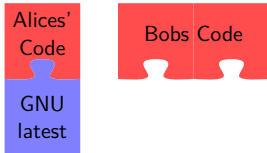
Eve:  
I need this one specific  
combination of compiler/  
mpi/fftw/BLAS/Lapack-  
/Scalapack/. But I need  
everything for two different  
compilers to see which is  
faster.

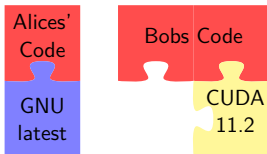
not work with GNU-  
compilers of version  
> 10.x.

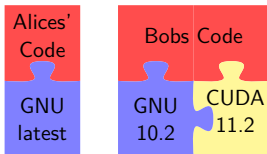


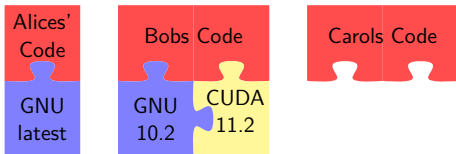


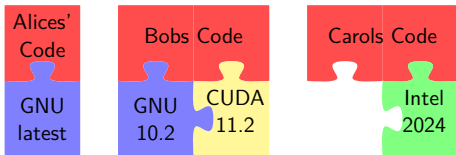


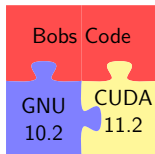


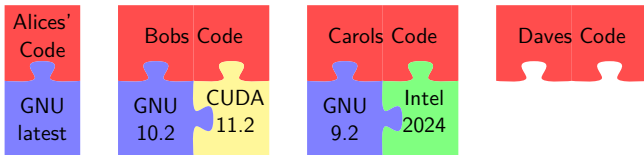


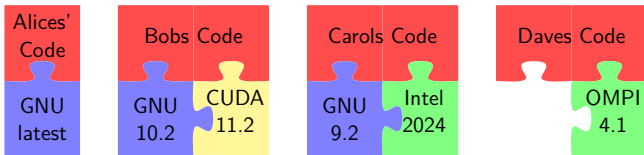




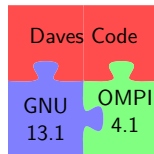
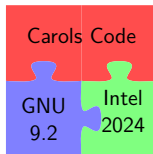
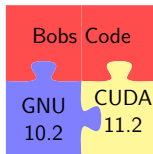
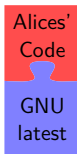


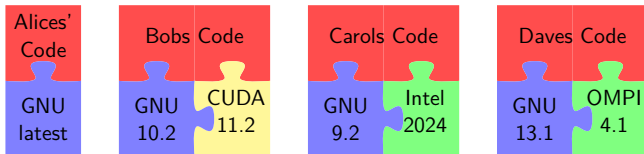




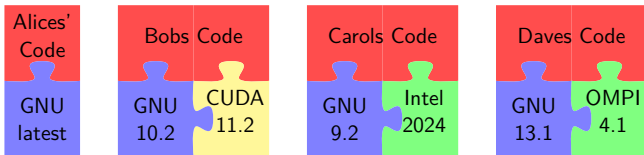




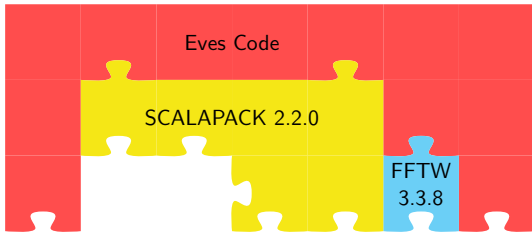
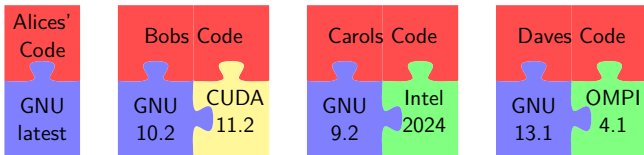




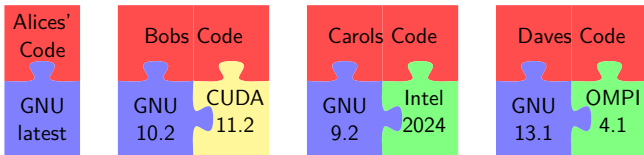
# A Modular Approach



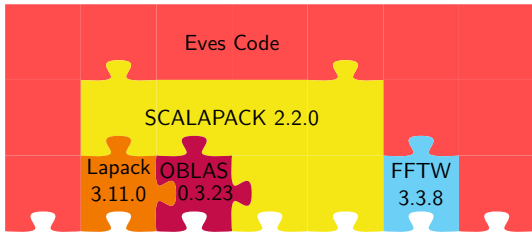
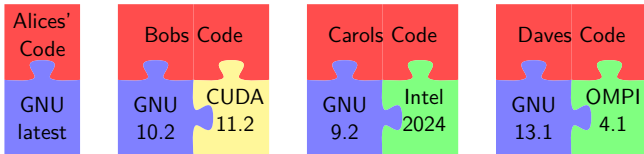
# A Modular Approach



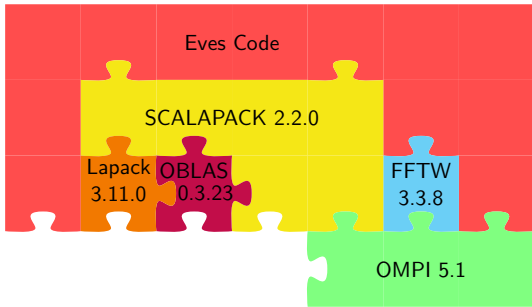
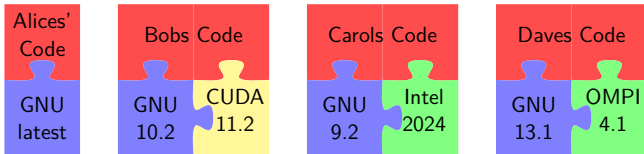
# A Modular Approach



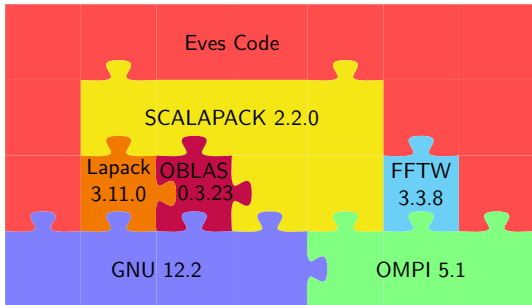
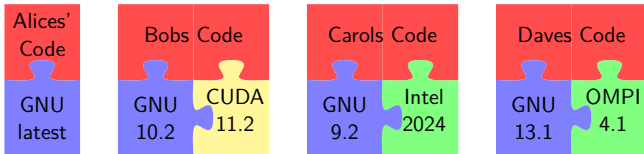
# A Modular Approach



# A Modular Approach

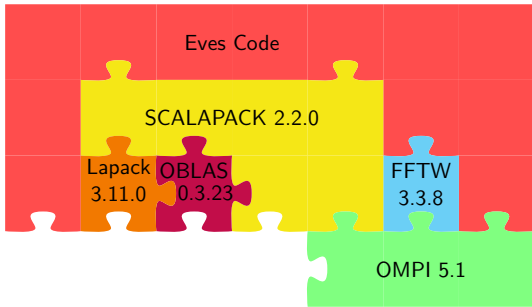
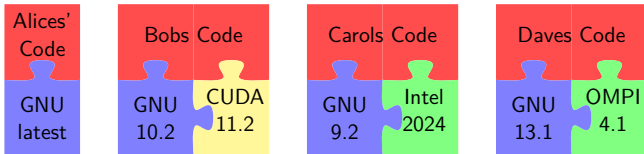


# A Modular Approach

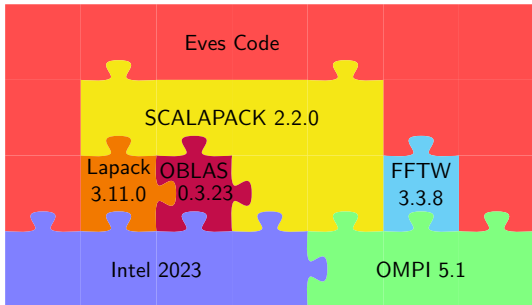
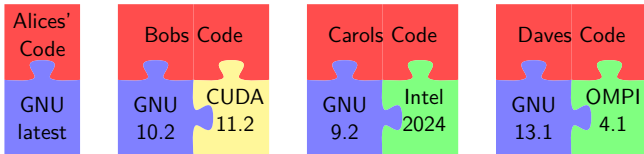




# A Modular Approach



# A Modular Approach



Environment modules allow users to temporarily change default executable/library path.

## Terminal

```
alice@hpc:$ which gcc  
/usr/bin/gcc
```

Environment modules allow users to temporarily change default executable/library path.

`module load <modulename>` loads a module and changes executable/library path.

## Terminal

```
alice@hpc:$ which gcc
/usr/bin/gcc
alice@hpc:$ module load gnu9/9.4.0
alice@hpc:$ which gcc
/opt/ohpc/pub/compiler/gcc/9.4.0/bin/gcc
```

Environment modules allow users to temporarily change default executable/library path.

`module unload <modulename>` removes a loaded module.

## Terminal

```
alice@hpc:$ which gcc
/usr/bin/gcc
alice@hpc:$ module load gnu9/9.4.0
alice@hpc:$ which gcc
/opt/ohpc/pub/compiler/gcc/9.4.0/bin/gcc
alice@hpc:$ module unload gnu9/9.4.0
alice@hpc:$ which gcc
/usr/bin/gcc
```

Environment modules allow users to temporarily change default executable/library path.

`module list` shows the currently loaded modules.

## Terminal

```
alice@hpc:$ module list
```

```
Currently Loaded Modules:  
1) autotools 2) gnu9/9.4.0
```

Environment modules allow users to temporarily change default executable/library path.

`module purge` unloads all modules.

## Terminal

```
alice@hpc:$ module list
Currently Loaded Modules:
 1) autotools 2) gnu9/9.4.0
alice@hpc:$ module purge
```

Environment modules allow users to temporarily change default executable/library path.

`module purge` unloads all modules.

## Terminal

```
alice@hpc:$ module list

Currently Loaded Modules:
 1) autotools 2) gnu9/9.4.0
alice@hpc:$ module purge
alice@hpc:$ module list
No modules loaded
```



Environment modules allow users to temporarily change default executable/library path.

`module avail` shows the available modules.

## Terminal

```
alice@hpc:$ module avail

----- /opt/ohpc/pub/modulefiles -----
EasyBuild/4.6.2          nvhpc/22.11
autotools                os
cmake/3.24.2            pmix/4.2.1
gnu12/12.2.0            prun/2.2
gnu9/9.4.0              singularity/3.7.1
...
```

# Managing Software with Spack

- Spack is a package manager for supercomputers that run Linux.

- Spack is a package manager for supercomputers that run Linux.
- Spack does not deploy precompiled binaries, but provides build instructions and dependency trees.

- Spack is a package manager for supercomputers that run Linux.
- Spack does not deploy precompiled binaries, but provides build instructions and dependency trees.
- Spack allows users to build customized software packages without relying on the administrators to install them.

- Spack is a package manager for supercomputers that run Linux.
- Spack does not deploy precompiled binaries, but provides build instructions and dependency trees.
- Spack allows users to build customized software packages without relying on the administrators to install them.
- Spack makes software packages available to users via a module system.

- Spack is a package manager for supercomputers that run Linux.
- Spack does not deploy precompiled binaries, but provides build instructions and dependency trees.
- Spack allows users to build customized software packages without relying on the administrators to install them.
- Spack makes software packages available to users via a module system.

Spack is well documented on our [website](#).

There are two basic steps to enable users to use Spack:

1. Write several config files:

- upstreams.yaml
- config.yaml
- modules.yaml
- compilers.yaml

Terminal

```
alice@hpc:$
```



There are two basic steps to enable users to use Spack:

1. Write several config files:

We abstracted this task with the RUB-exclusive `rub-deploy-spack-configs` command.

#### Terminal

```
alice@hpc:~$ rub-deploy-spack-configs
Deployment completed.
alice@hpc:~$
```

There are two basic steps to enable users to use Spack:

1. Write several config files:

### Terminal

```
alice@hpc:$ cat ~/.bashrc
...
export \
MODULEPATH=$MODULEPATH:$HOME/spack/share/spack/lmod/linux-almalinux9-x86_64/Core
source /cluster/spack/0.23.0/share/spack/setup-env.sh
...
alice@hpc:$
```

To find available packages, use the `spack list` command.

Terminal

```
alice@hpc:$ spack list openfoam
```

To find available packages, use the `spack list` command.

## Terminal

```
alice@hpc:$ spack list openfoam
openfoam openfoam-org
==> 2 packages
alice@hpc:$
```

To find available packages, use the `spack list` command.

## Terminal

```
alice@hpc:$ spack list openfoam
openfoam openfoam-org
==> 2 packages
alice@hpc:$
```

For a quick search for all available packages in Spack, visit <https://packages.spack.io/>.

`spack info` also lists multiple build variants of one package:

Terminal

```
alice@hpc:$ spack info hdf5
```

`spack info` also lists multiple build variants of one package:

### Terminal

```
alice@hpc:~$ spack info hdf5
CMakePackage:  hdf5
Description:
  HDF5 is a data model, library, and file format for storing and managing
  ...
```

spack info also lists multiple build variants of one package:

## Terminal

```
alice@hpc:~$ spack info hdf5
CMakePackage:  hdf5
Description:
  HDF5 is a data model, library, and file format for storing and managing
  ...
Preferred version:
  1.14.5 https://support.hdfgroup.org/.../downloads/hdf5-1.14.5.tar.gz
```



`spack info` also lists multiple build variants of one package:

## Terminal

```
alice@hpc:~$ spack info hdf5
CMakePackage:  hdf5
Description:
  HDF5 is a data model, library, and file format for storing and managing
  ...
Preferred version:
  1.14.5  https://support.hdfgroup.org/.../downloads/hdf5-1.14.5.tar.gz
Safe versions:
  develop-2.0  [git] https://github.com/HDFGroup/hdf5.git on branch develop
  ...
```

`spack info` also lists multiple build variants of one package:

## Terminal

```
alice@hpc:~$ spack info hdf5
CMakePackage:  hdf5
Description:
  HDF5 is a data model, library, and file format for storing and managing
  ...
Preferred version:
  1.14.5  https://support.hdfgroup.org/.../downloads/hdf5-1.14.5.tar.gz
Safe versions:
  develop-2.0  [git] https://github.com/HDFGroup/hdf5.git on branch develop
  ...
Deprecated versions:
  None
```

`spack info` also lists multiple build variants of one package:

## Terminal

```
alice@hpc:~$ spack info hdf5
CMakePackage:  hdf5
Description:
  HDF5 is a data model, library, and file format for storing and managing
  ...
Preferred version:
  1.14.5 https://support.hdfgroup.org/.../downloads/hdf5-1.14.5.tar.gz
Safe versions:
  develop-2.0 [git] https://github.com/HDFGroup/hdf5.git on branch develop
  ...
Deprecated versions:
  None
Variants:
  api [default]  default, v110, v112, v114
  Choose api compatibility for earlier version
```

In order to check the dependencies of a package, use `spack spec`

## Terminal

```
alice@hpc:~$ spack spec hdf5
[ ] hdf5@1.14.3%gcc@11.4.1+cxx+fortran hl ... build_system=cmake ...
[ ^ ] ^cmake@3.27.9%gcc@11.4.1 doc+ncurses+... build_type=Release...
[ ^ ] ^curl@8.7.1%gcc@11.4.1 gssapi ... libs=shared,static...
[ ^ ] ^nghttp2@1.57.0%gcc@11.4.1... arch=linux-almalinux9-zen4
..
```

`spack install <package>` will build and install the selected package and its dependencies.

## Terminal

```
alice@hpc:$ spack install hdf5  
...
```

`spack install <package>` will build and install the selected package and its dependencies.

## Terminal

```
alice@hpc:$ spack install hdf5  
...
```

Variants can be enabled (+) and disabled (~), dependencies can be specified (^)

## Terminal

```
alice@hpc:$ spack install hdf5 +mpi +cxx ~hl ^openmpi@4.1.5 +threads_multiple  
...
```

Use `spack find -vl` to show all installed variants and their hashes:

## Terminal

```
alice@hpc:~$ spack find -vl hdf5
-- linux-almalinux9-zen4 / gcc@11.4.1 -----
amrsck6 hdf5@1.14.3~cxx~fortran+hl...
2dsgtoe hdf5@1.14.3+cxx+fortran~hl...
==> 2 installed packages
```

Use `spack find -vl` to show all installed variants and their hashes:

## Terminal

```
alice@hpc:~$ spack find -vl hdf5
-- linux-almalinux9-zen4 / gcc@11.4.1 -----
amrsck6 hdf5@1.14.3~cxx~fortran+hl...
2dsgtoe hdf5@1.14.3+cxx+fortran~hl...
==> 2 installed packages
```

The unique installation hashes e.g. `amrsck6` and `2dsgtoe` can be used to view more details of the installed package.

## Terminal

```
alice@hpc:~$ spack spec /2dsgtoe
[^] hdf5@1.14.3%gcc@11.4.1~cxx~fortran+hl~ipo~java~map..
[^]      ^cmake@3.27.9%gcc@11.4.1~doc..
[^]      ^curl@8.7.1%gcc@11.4.1..
```



Use `spack find -vl` to show all installed variants and their hashes:

## Terminal

```
alice@hpc:~$ spack find -vl hdf5
-- linux-almalinux9-zen4 / gcc@11.4.1 -----
amrsk6 hdf5@1.14.3~cxx~fortran+hl...
2dsgtoe hdf5@1.14.3+cxx+fortran-hl...
==> 2 installed packages
```

Two installations can be compared by using the installation hashes.

## Terminal

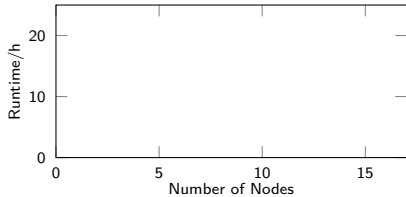
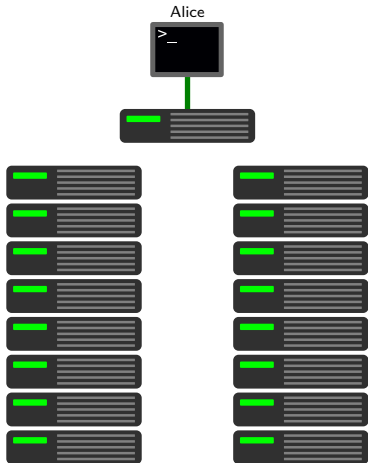
```
alice@hpc:~$ spack diff /amrsk6 /2dsgtoe
-- hdf5@1.14.3/amrsk6mml43sfv4bhvvniwdydaxfgne
+++ hdf5@1.14.3/2dsgtoevoypx7dr45l5ke2dlb56agvz4
@@ virtual_on_incoming_edges @@
- openmpi mpi
+ mpich mpi
```

No longer needed packages or package variants can be removed using the installation hash with the `spack uninstall` command.

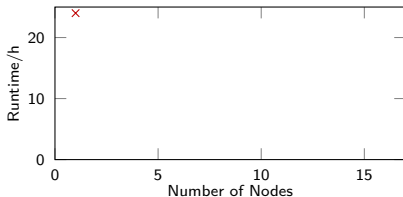
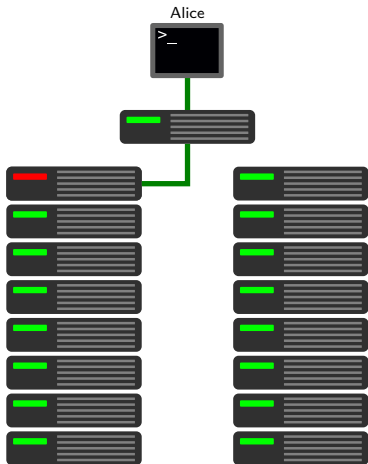
### Terminal

```
alice@hpc:$ spack uninstall /amrsc6
```

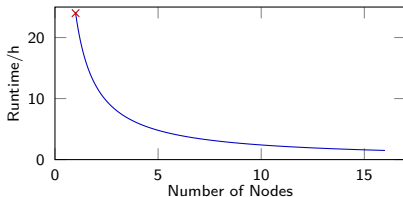
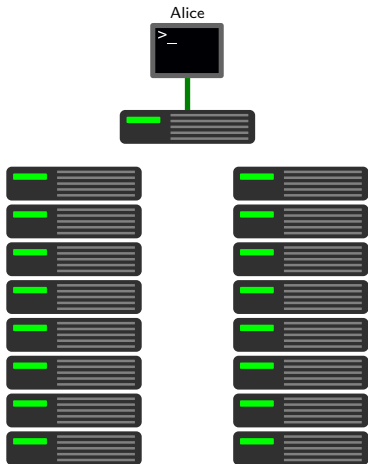
# Performance Scaling



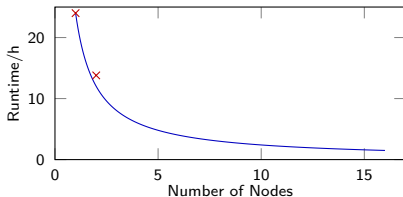
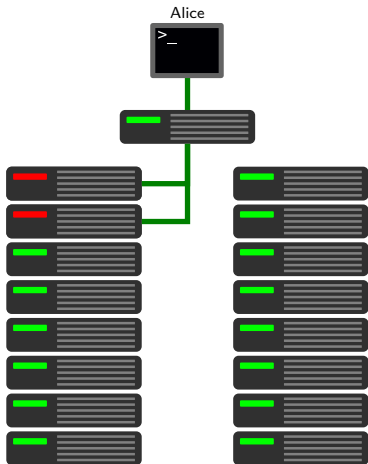
Alice is conducting Research on an HPC cluster.



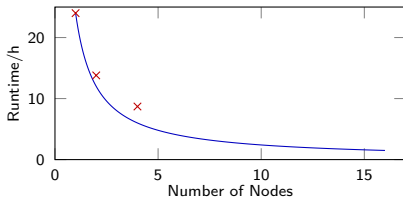
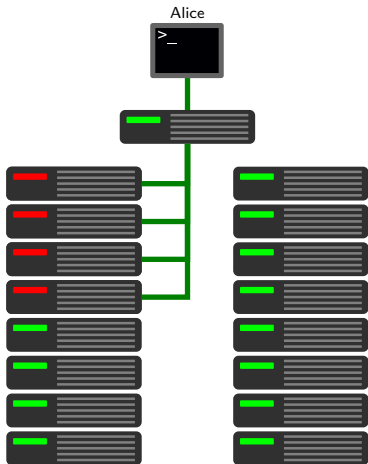
Alice runs a job on one node and measures a runtime of 24 h.



Alice guesses that the runtime scales as  $t_N = \frac{t_1}{N}$ .

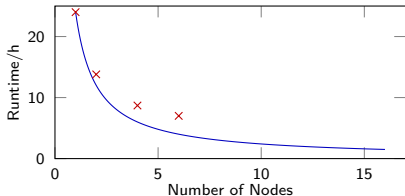
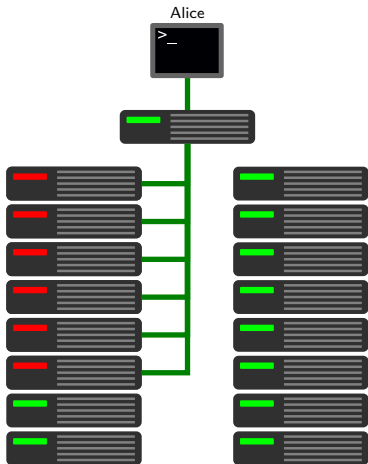


Alice doubles the number of nodes to cut the runtime in half.

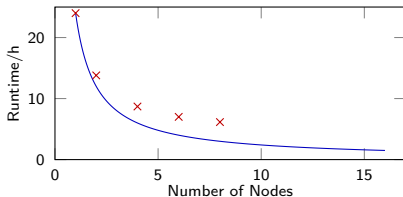
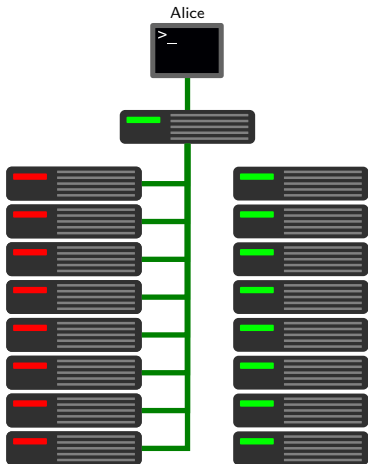


Alice measures a few different node numbers.

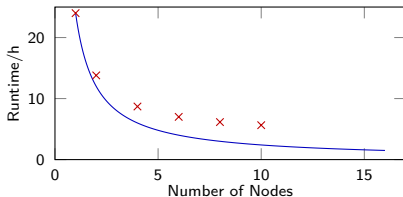
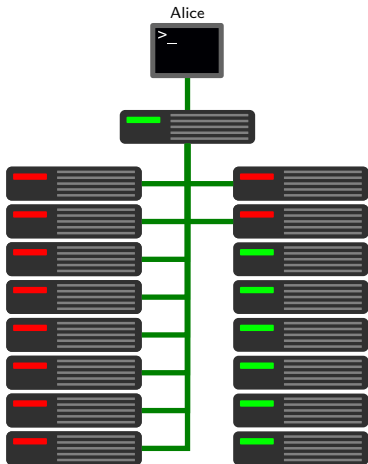




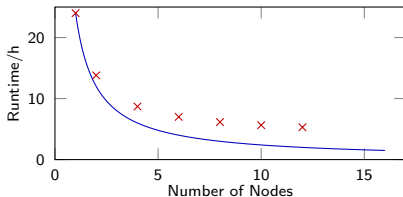
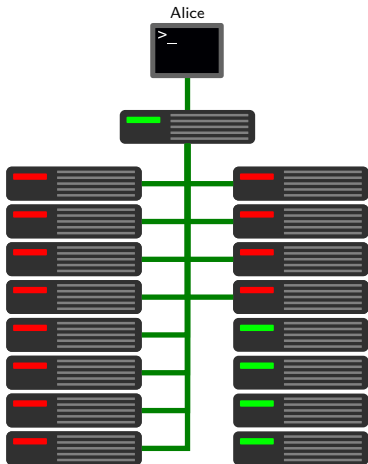
Alice measures a few different node numbers.



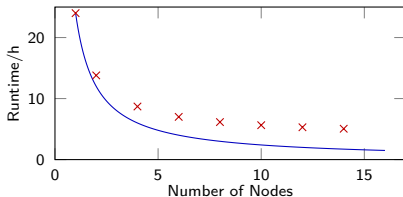
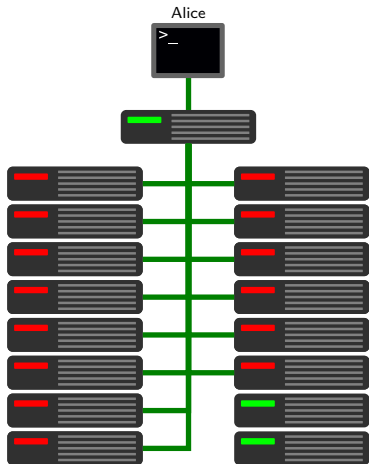
Alice measures a few different node numbers.



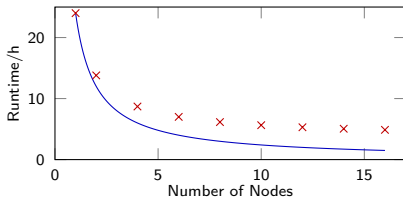
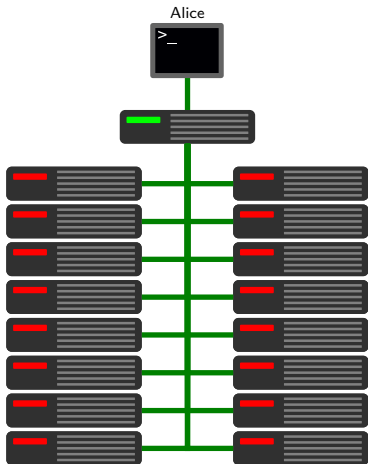
Alice measures a few different node numbers.



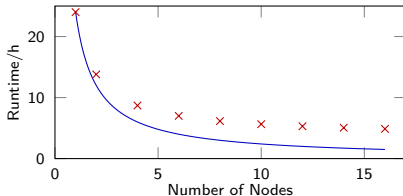
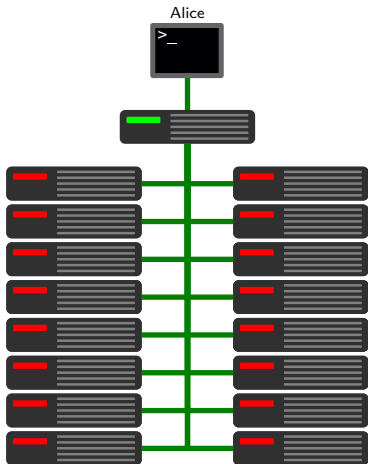
Alice measures a few different node numbers.



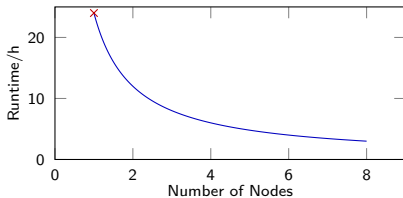
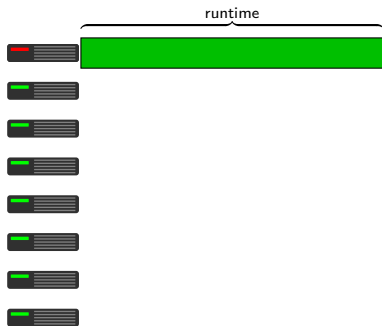
Alice measures a few different node numbers.



Alice measures a few different node numbers.

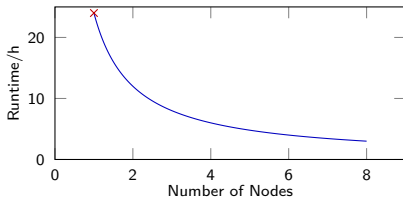
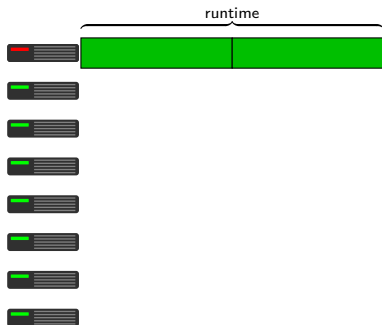


Alice finds a significant discrepancy between her idea of scaling and the measurement.

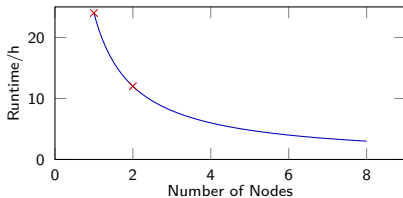
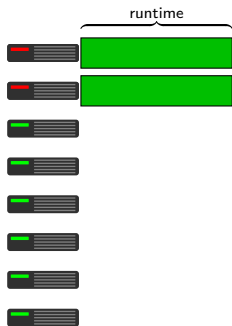


Alice runs a job on one node and measures a runtime of 24 h.

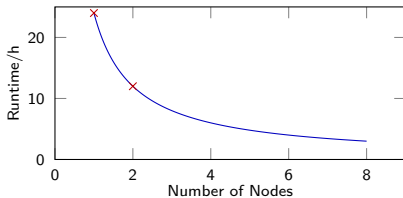
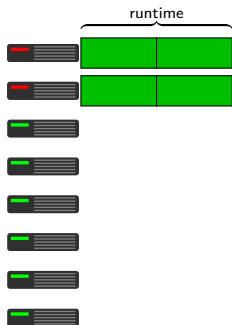




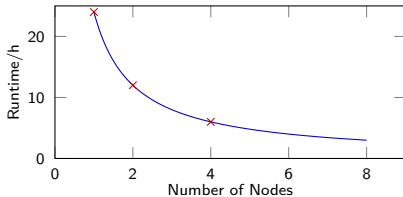
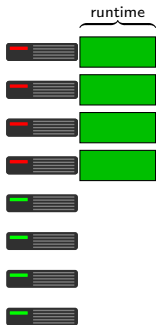
Alice splits the work into two equal parts.



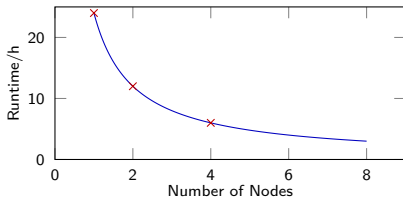
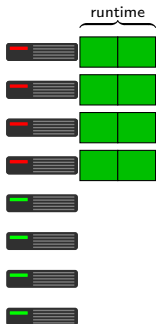
Computing both parts in parallel splits the runtime in two.



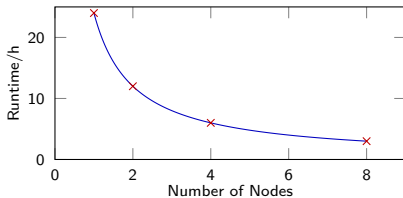
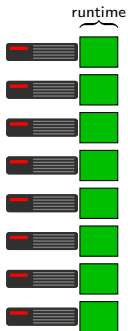
Alice splits each part in two equal parts again.



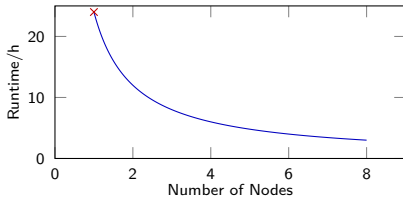
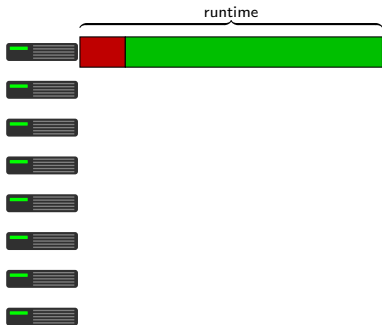
Computing all parts in parallel splits the runtime in two again.



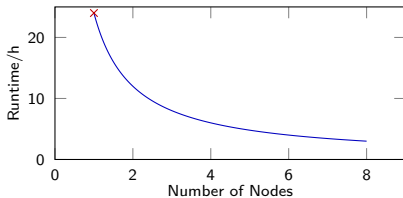
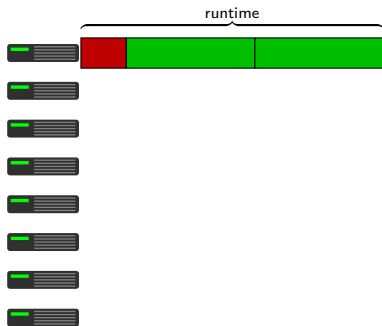
Alice splits each part in two equal parts again.



Computing all parts in parallel splits the runtime in two again.

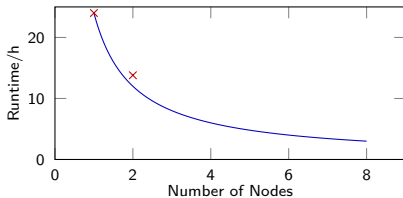
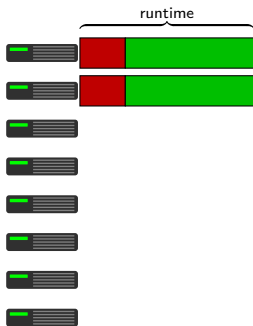


Alice' code contains some unparallelizable (serial) part (red).

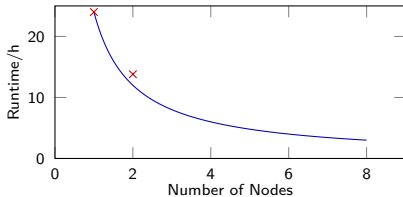
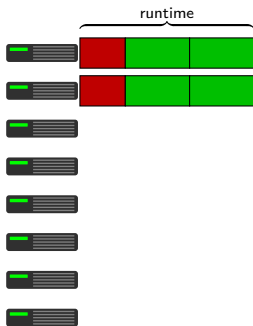


Alice splits the parallelizable work into two equal parts.

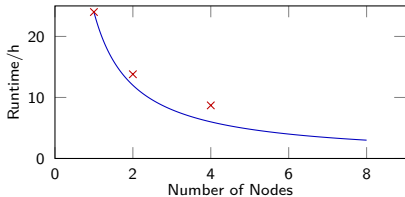
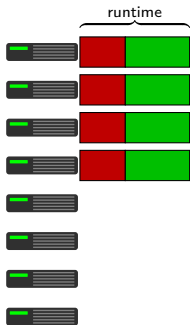




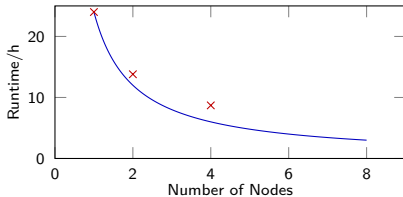
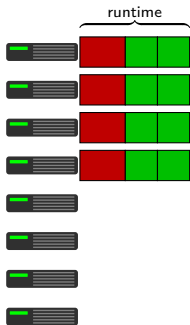
The serial part stays constant.



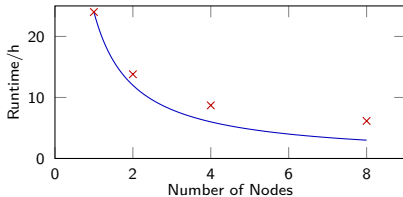
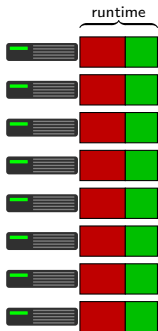
Alice splits the parallelizable work into two equal parts again.



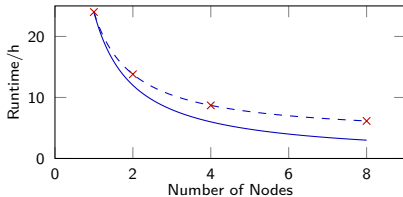
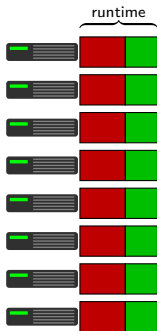
The serial part stays constant.



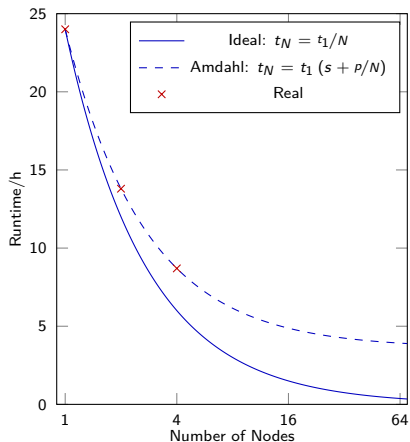
Alice splits the parallelizable work into two equal parts again.



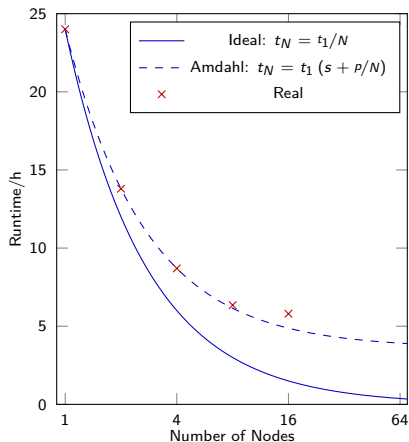
The serial part stays constant.



A partial parallel code scales as:  
$$t_N = t_1 \left( s + \frac{p}{N} \right).$$

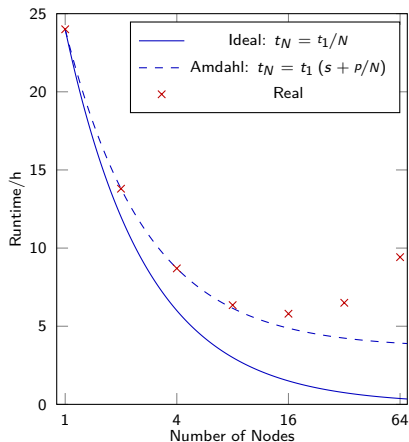


- For small node numbers many HPC codes follow an Amdahl like behavior.

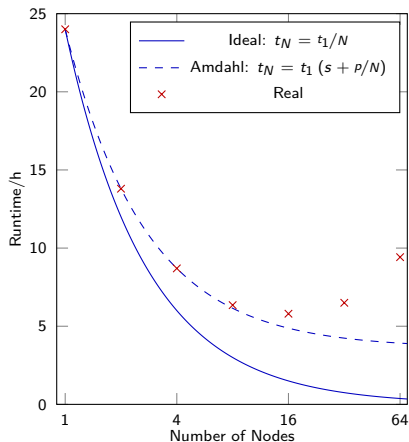


- For small node numbers many HPC codes follow an Amdahl like behavior.
- At some point the real behavior will deviate from Amdahl's law and stagnate.

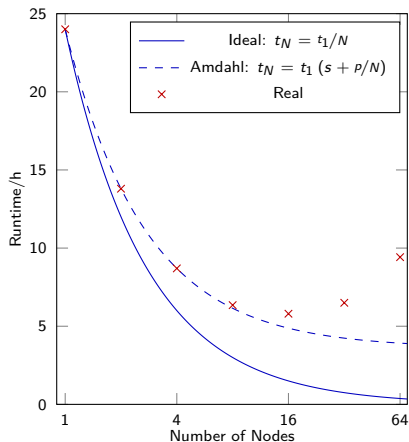




- For small node numbers many HPC codes follow an Amdahl like behavior.
- At some point the real behavior will deviate from Amdahl's law and stagnate.
- In many cases the runtime can worsen with higher node numbers because:



- For small node numbers many HPC codes follow an Amdahl like behavior.
- At some point the real behavior will deviate from Amdahl's law and stagnate.
- In many cases the runtime can worsen with higher node numbers because:
  - Workload cannot be split into smaller sections.



- For small node numbers many HPC codes follow an Amdahl like behavior.
- At some point the real behavior will deviate from Amdahl's law and stagnate.
- In many cases the runtime can worsen with higher node numbers because:
  - Workload cannot be split into smaller sections.
  - Network latency and bandwidth limitations.

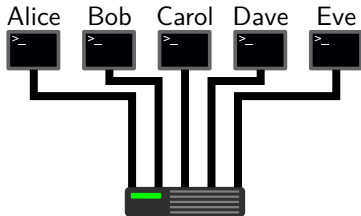
# Code of Conduct

## Rules

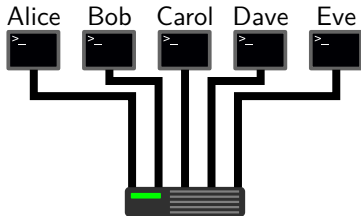
1. Participating on an HPC system requires constant care and mutual respect.
2. A person using an HPC system shall act in such a way as not to hinder or inconvenience any other person more than is unavoidable in the circumstances.

---

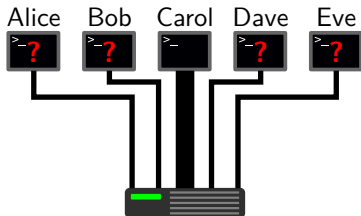
<sup>1</sup>Based on: Deutsche Straßenverkehrs-Ordnung(StVO) § 1 Grundregeln



- Multiple users are connected to the login node.

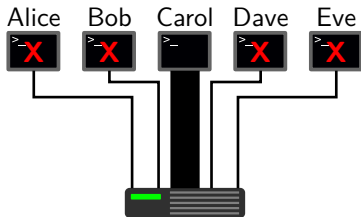


- Multiple users are connected to the login node.
- Carol wants to copy huge amounts of data to/from the cluster.



- Multiple users are connected to the login node.
- Carol wants to copy huge amounts of data to/from the cluster.
- Carol hogs all the bandwidth.

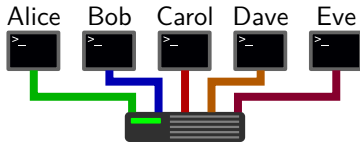




- Multiple users are connected to the login node.
- Carol wants to copy huge amounts of data to/from the cluster.
- Carol hogs all the bandwidth.
- The connection for everyone else gets unusable.

### Copying Data from/to HPC Cluster

1. Utilize a bandwidthlimit (e.g. `rsync -bwlimit=<rate>`)
2. Multiple parallel copy processes are not faster than one process.
3. Copying multiple small files is more demanding than few big ones. Use `tar` to pack files together.



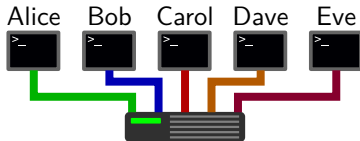
- Multiple users are connected to the login node.

CPU usage:



Memory usage:





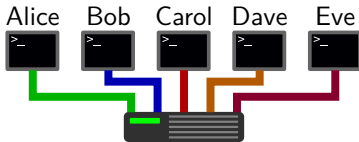
CPU usage:



Memory usage:



- Multiple users are connected to the login node.
- The CPU is idling and memory is free. Everyone has enough resources for their work.



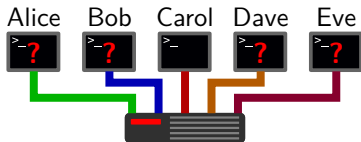
CPU usage:



Memory usage:



- Multiple users are connected to the login node.
- The CPU is idling and memory is free. Everyone has enough resources for their work.
- Carol wants her compilation to go faster and utilizes more processors.



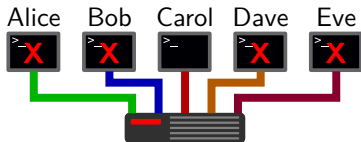
CPU usage:



Memory usage:



- Multiple users are connected to the login node.
- The CPU is idling and memory is free. Everyone has enough resources for their work.
- Carol wants her compilation to go faster and utilizes more processors.
- There are no longer enough resources for everyone.



CPU usage:



Memory usage:

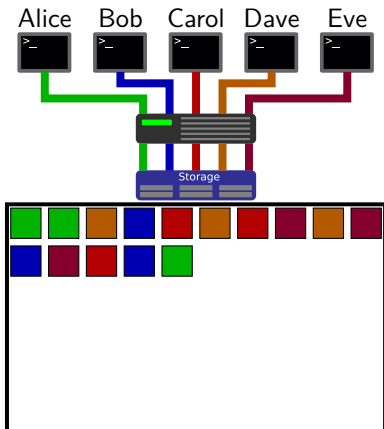


- Multiple users are connected to the login node.
- The CPU is idling and memory is free. Everyone has enough resources for their work.
- Carol wants her compilation to go faster and utilizes more processors.
- There are no longer enough resources for everyone.

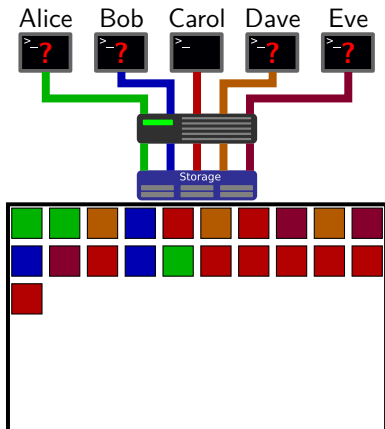
### Exhausting Login Node Resources

1. The login node is only suited for small analysis/compilation tasks.
2. If lots of resources are required for analysis/compilation/... request a compute node.
3. Intense processes on the login node will be killed.
4. Repeated offenders might be banned from using HPC Clusters.

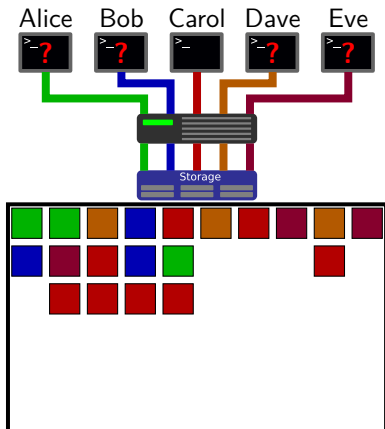




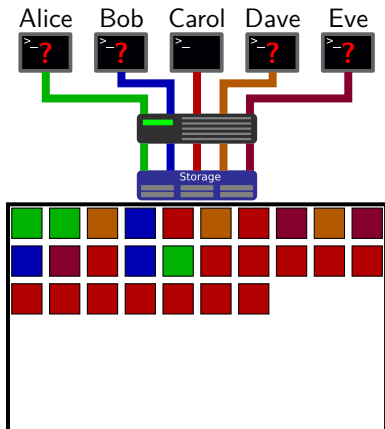
- Multiple users utilize the global storage system.



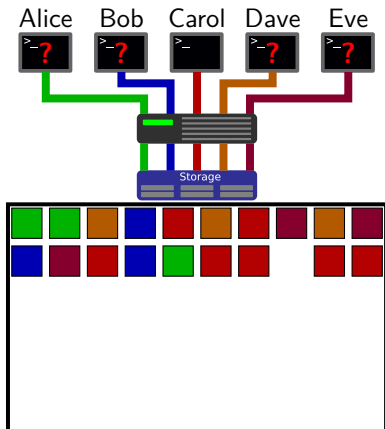
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.



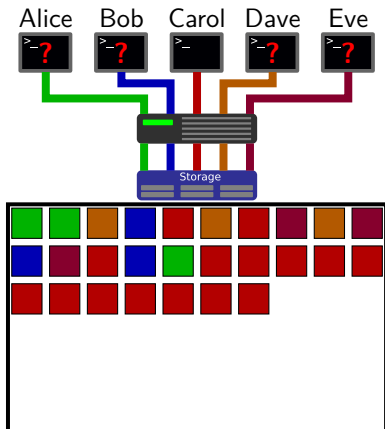
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.



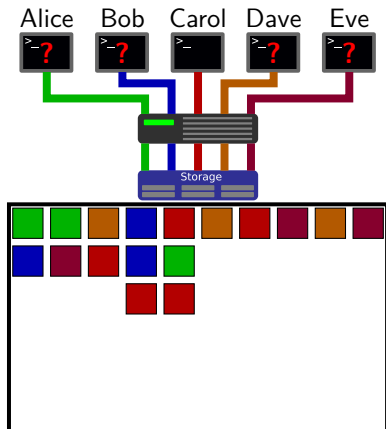
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.



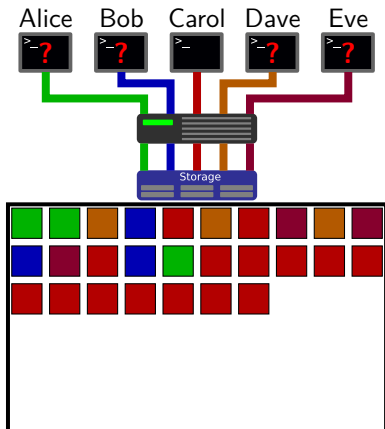
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.



- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.

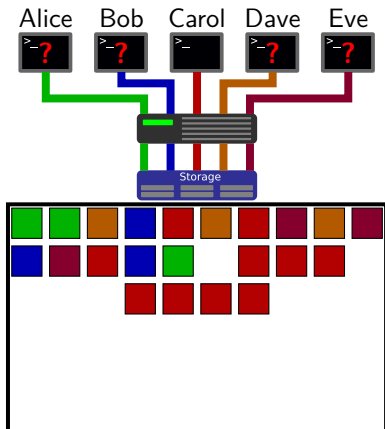


- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.

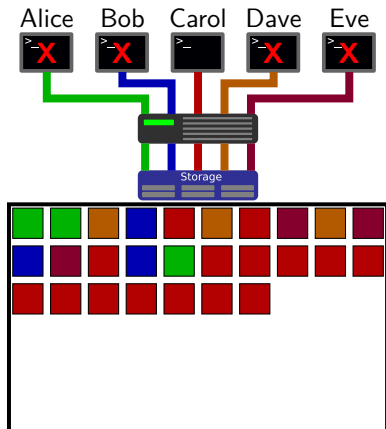


- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.





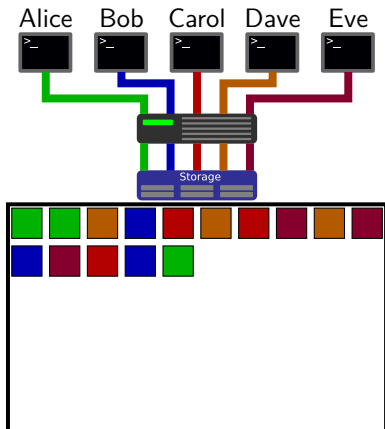
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.



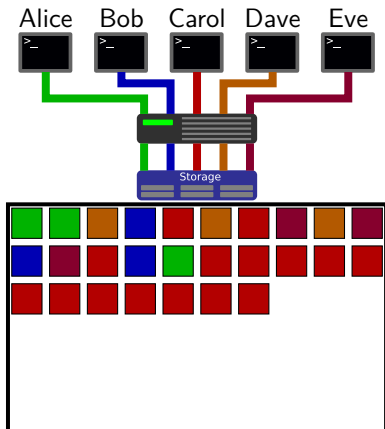
- Multiple users utilize the global storage system.
- Carol's computations quickly writes/deletes/opens/closes a lot of files on the global files system.
- Many quick global operations are poison for shared file systems, rendering it useless for everyone.

### Stress Shared File System

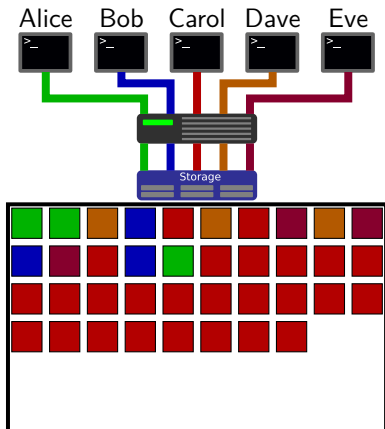
1. Utilize local file systems for temporary data.
2. Write/read to/from the shared file system in few big chunks.



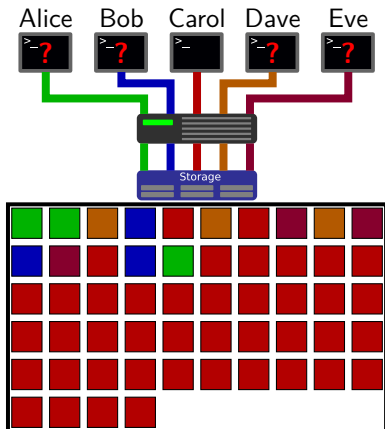
- Multiple users utilize the global storage system.



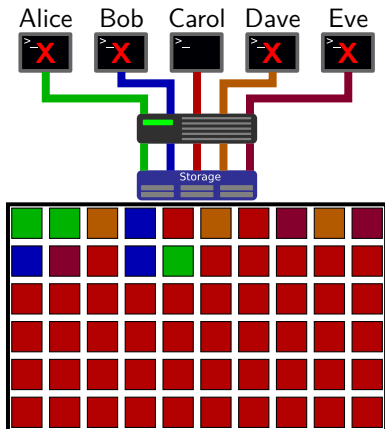
- Multiple users utilize the global storage system.
- Carol's computations dump a lot of data onto the global files system.



- Multiple users utilize the global storage system.
- Carol's computations dump a lot of data onto the global files system.



- Multiple users utilize the global storage system.
- Carol's computations dump a lot of data onto the global files system.

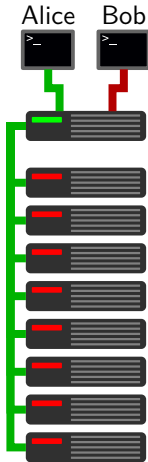


- Multiple users utilize the global storage system.
- Carol's computations dump a lot of data onto the global files system.
- There is no longer enough storage left for everyone to use.

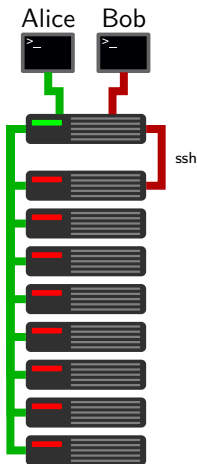


### Fill the Filesystem

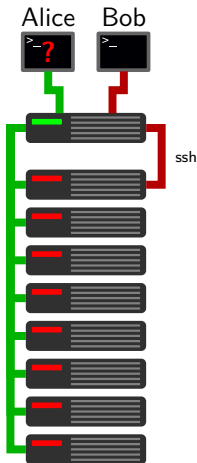
1. The Filesystem utilizes quotas, preventing users from storing more data than a certain share.
2. Regularly tidy up your data to keep the storage free for further computations.



- All nodes are allocated for jobs.



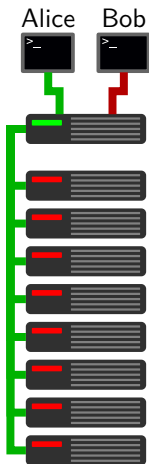
- All nodes are allocated for jobs.
- Bob does not want to wait for his slot, ignores the resource manager and connects to a node directly.



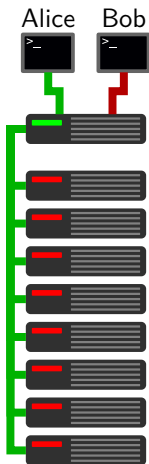
- All nodes are allocated for jobs.
- Bob does not want to wait for his slot, ignores the resource manager and connects to a node directly.
- Alice wonders why her calculation slows down.

### Work on Already Used Nodes

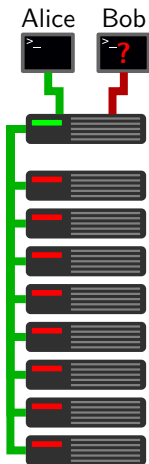
1. Resource allocation should always be done via SLURM.
2. The HPC-Clusters are setup such that ssh connections to nodes are denied unless the user also owns the job running on that node.



- Alice is not computing anything, but she reserved all nodes for an extended period of time just in case.



- Alice is not computing anything, but she reserved all nodes for an extended period of time just in case.
- Bob has urgent research, but cannot get free nodes.



- Alice is not computing anything, but she reserved all nodes for an extended period of time just in case.
- Bob has urgent research, but cannot get free nodes.
- Alice is wasting valuable resources.



### Hugging Compute Nodes

1. Jobs have a maximum time to prevent node-hugging.
2. Node-hugging is punished by the SLURM accountant by lowering the users priority for scheduling new jobs.

# Take Home Messages

- Keep your private key private! Use TFA for secure access!

- Keep your private key private! Use TFA for secure access!
- HPC resources are shared. Be considered of others!

- Keep your private key private! Use TFA for secure access!
- HPC resources are shared. Be considered of others!
- Assist the scheduler with accurate runtime estimates!

- Keep your private key private! Use TFA for secure access!
- HPC resources are shared. Be considered of others!
- Assist the scheduler with accurate runtime estimates!
- The accounting will ensure that everybody gets their share of computing time!

- Keep your private key private! Use TFA for secure access!
- HPC resources are shared. Be considered of others!
- Assist the scheduler with accurate runtime estimates!
- The accounting will ensure that everybody gets their share of computing time!
- More resources will not necessarily speedup your computation!

**Happy Computing!**