HPC Induction

Part III: Software

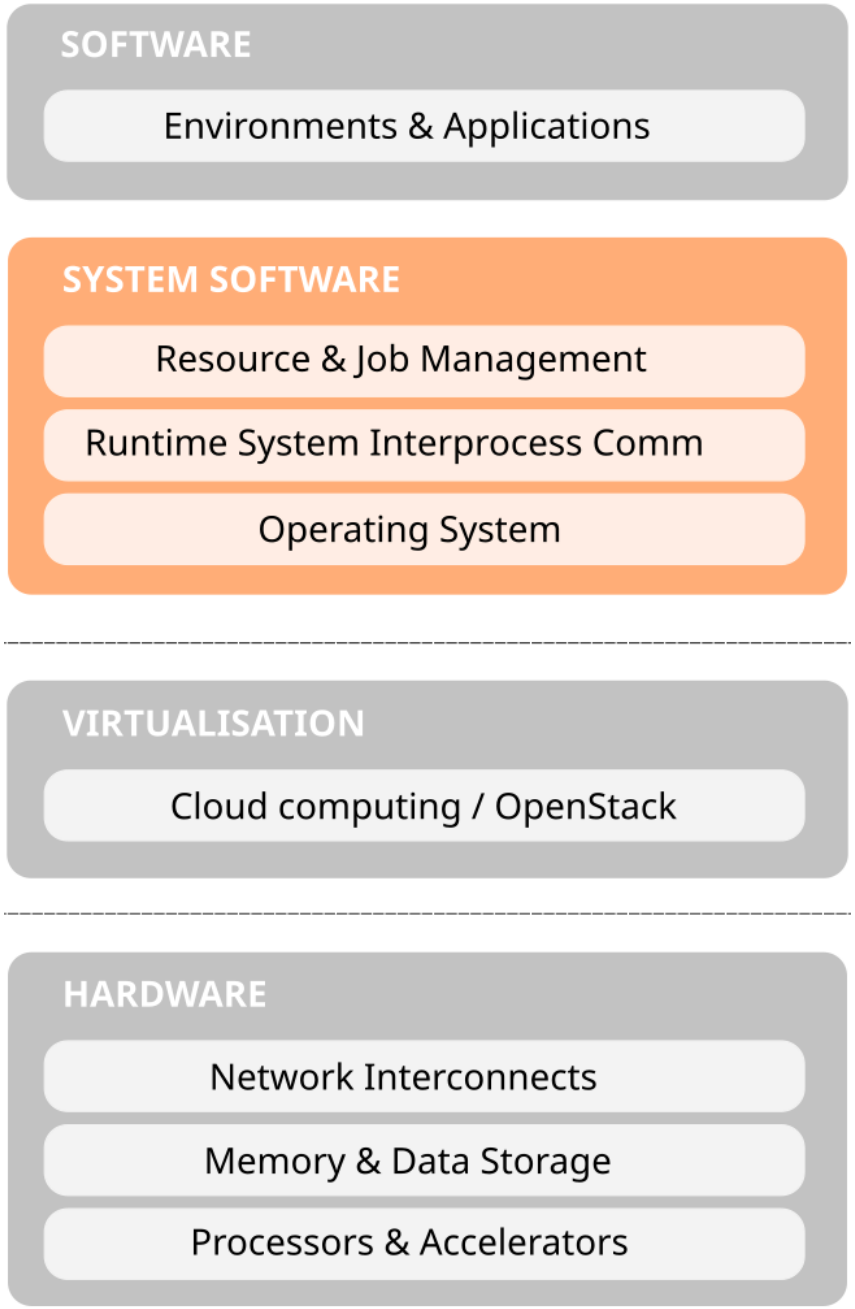
Jascha Schewtschenko

Royal Observatory of Edinburgh, University of Edinburgh

May 14, 2025

Outline

1. System Software
   - Operating System
   - Inter-Process Communication
   - Resource & Job Management
2. Software: Environments & Applications
3. Etiquette


SOFTWARE
Environments & Applications
SYSTEM SOFTWARE
Resource & Job Management
Runtime System Interprocess Comm
Operating System
VIRTUALISATION
Cloud computing / OpenStack
HARDWARE
Network Interconnects
Memory & Data Storage
Processors & Accelerators

Operating System

OS: Basics

- While there are many server OS out there (FreeBSD, z/OS, MS Windows Server, etc.), there is one dominating the HPC market

OS: Basics

- While there are many server OS out there (FreeBSD, z/OS, MS Windows Server, etc.), there is one dominating the HPC market

Linux Runs on All of the Top 500 Supercomputers, Again!

Last updated June 21, 2019 By Abhishek Prakash — 15 Comments

Linux might be struggling for a decent desktop market share but it is definitely ruling the world of supercomputers. Linux is the supercomputer operating system by choice.

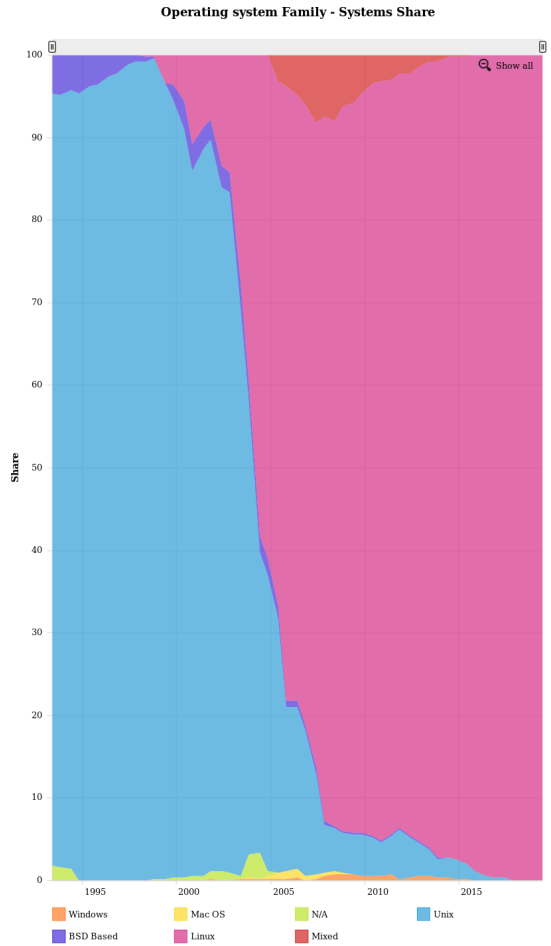
OS: Basics

- While there are many server OS out there (FreeBSD, z/OS, MS Windows Server, etc.), there is one dominating the HPC market

Linux Runs on All of the Top 500 Supercomputers, Again!

Last updated June 21, 2019 By Abhishek Prakash — 15 Comments

Linux might be struggling for a decent desktop market share but it is definitely ruling the world of supercomputers. Linux is the supercomputer operating system by choice.



OS: Basics (cont.)

- Few distros dominate the market

Top five operating systems (November 2018)

Vendor	Systems
Linux	233
CentOS	139
Cray Linux Environment	48
bullx SCS	13
SUSE Linux Enterprise	10

Note: All operating systems of the TOP500 systems use Linux, but Linux above is *generic* Linux

OS: Basics (cont.)

- Few distros dominate the market

Top five operating systems (November 2018)

Vendor	Systems
Linux	233
CentOS	139
Cray Linux Environment	48
bullx SCS	13
SUSE Linux Enterprise	10

Note: All operating systems of the TOP500 systems use Linux, but Linux above is *generic* Linux

- That gives you an environment you are familiar with when logging to new HPC infrastructure

OS: Basics (cont.)

- Few distros dominate the market

Top five operating systems (November 2018)

Vendor	Systems
Linux	233
CentOS	139
Cray Linux Environment	48
bullx SCS	13
SUSE Linux Enterprise	10

Note: All operating systems of the TOP500 systems use Linux, but Linux above is *generic* Linux

- That gives you an environment you are familiar with when logging to new HPC infrastructure
- For a tutorial on how to use Linux (or Unix), please see:

Linux Induction Lecture [Link]

(it is **ESSENTIAL** that you know these very few basics BEFORE you start working on the system, especially if you are using the command-line interfaces (CLI))

OS: Authentication

- When logging into a (remote) system you have to provide verification of your identity as a user

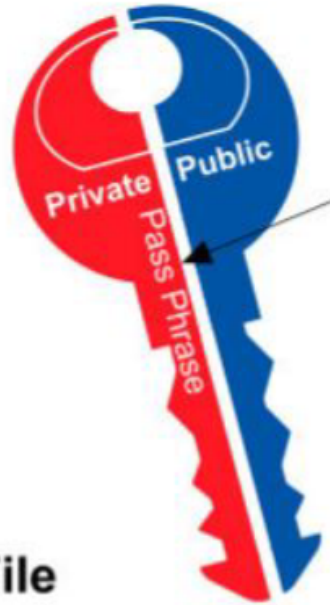
OS: Authentication

- When logging into a (remote) system you have to provide verification of your identity as a user
- traditionally, passwords were used, but disadvantage is that they can easily be stolen (phishing/spoofing, guessing, key loggers, "looking over shoulder")

OS: Authentication

- When logging into a (remote) system you have to provide verification of your identity as a user
- traditionally, passwords were used, but disadvantage is that they can easily be stolen (phishing/spoofing, guessing, key loggers, "looking over shoulder")
- nowadays many system use SSH key authentication (e.g. RSA)

OS: Authentication



Pass Phrase

Associated with the key
is a Pass Phrase.
It is mandatory to use
a Pass Phrase.

Private Key File

Stored on your desktop
or laptop

The pass phrase
protects the
private key

Public Key File

Stored on remote server

TOP SECRET!!

NEVER share a private key !!

OS: Authentication

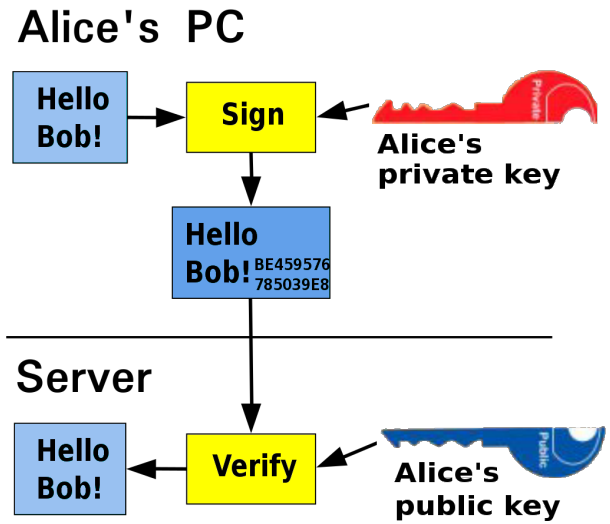
- consists of an asymmetric key pair (like for PGP): a **SECRET** private key and a public key

OS: Authentication

- consists of an asymmetric key pair (like for PGP): a **SECRET** private key and a public key
- private key stored on your computer, public key on the server
  (on Linux in ~/.ssh/authorized_keys)

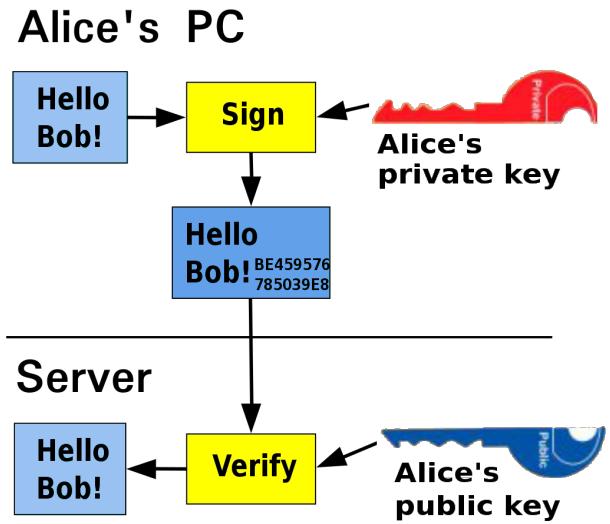
OS: Authentication

- consists of an asymmetric key pair (like for PGP): a **SECRET** private key and a public key
- private key stored on your computer, public key on the server
  (on Linux in ~/.ssh/authorized_keys)
- the private key can sign a message (e.g. login request), while the public key can be used to verify the signature (but not to create it)



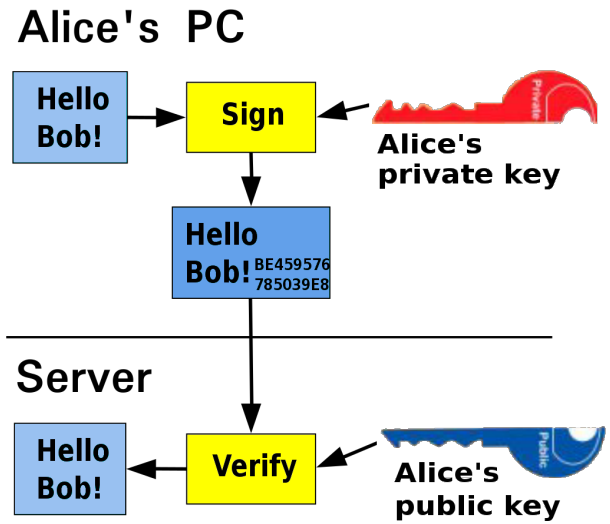
OS: Authentication

- consists of an asymmetric key pair (like for PGP): a **SECRET** private key and a public key
- private key stored on your computer, public key on the server
  (on Linux in ~/.ssh/authorized_keys)
- the private key can sign a message (e.g. login request), while the public key can be used to verify the signature (but not to create it)
- Avoids identity theft by methods listed above as no secret authentication data ever leaves user's computer



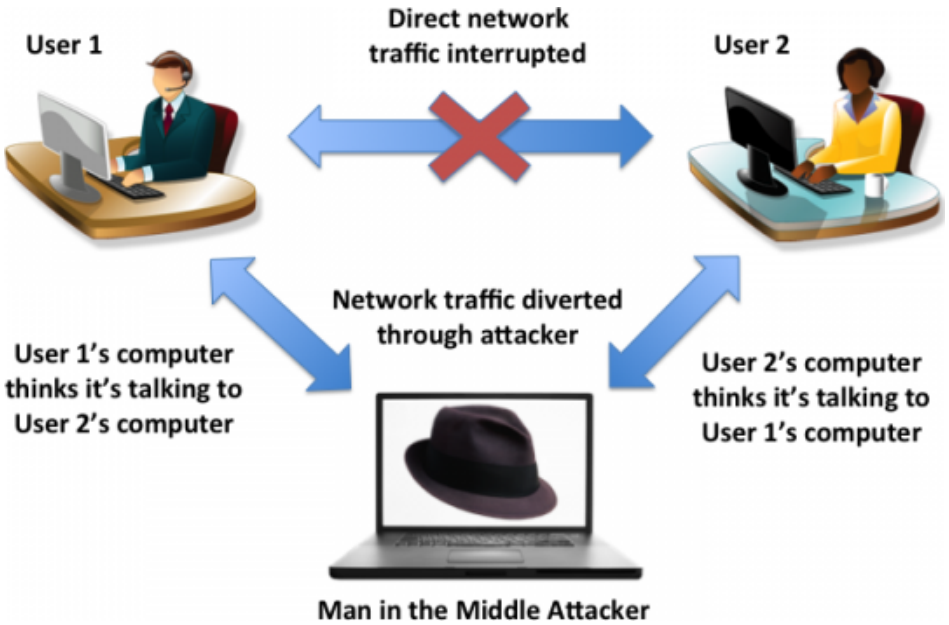
OS: Authentication

- consists of an asymmetric key pair (like for PGP): a **SECRET** private key and a public key
- private key stored on your computer, public key on the server
  (on Linux in ~/.ssh/authorized_keys)
- the private key can sign a message (e.g. login request), while the public key can be used to verify the signature (but not to create it)
- Avoids identity theft by methods listed above as no secret authentication data ever leaves user's computer
- Yet, to avoid key theft/misuse, you **MUST** protect the private key with a passphrase



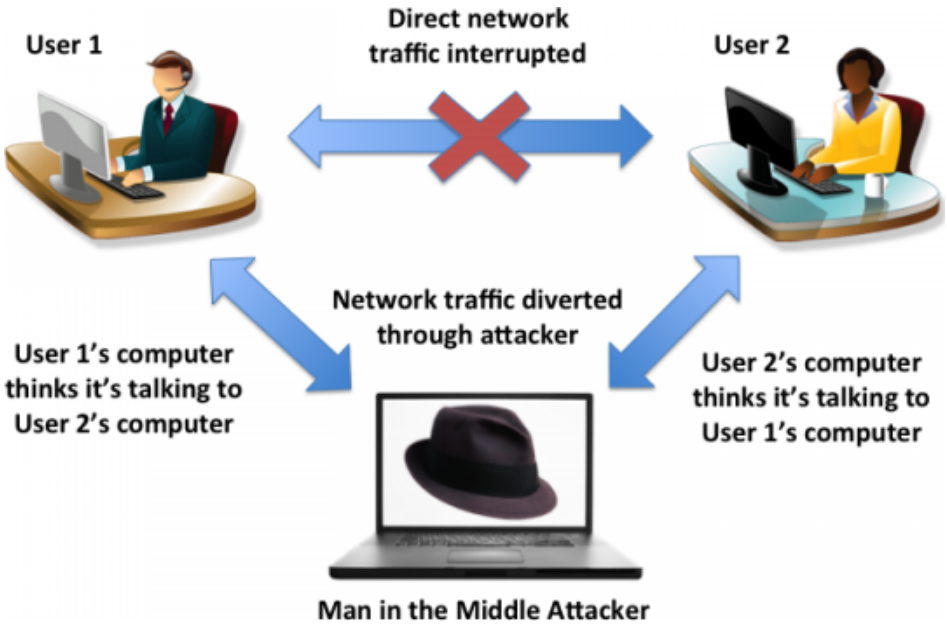
OS: SSH Fingerprints

- So-called SSH fingerprints are used to ensure that your connection to the server is secure (i.e. that you are connected to the real server and/or not a 'Man-in-the-Middle' (MitM))



OS: SSH Fingerprints

- So-called SSH fingerprints are used to ensure that your connection to the server is secure (i.e. that you are connected to the real server and/or not a 'Man-in-the-Middle' (MitM))



- It is a hash based on the public SSH key of the server.

OS: SSH Fingerprints (cont.)

- When you log in for the first time to a server (e.g. login node), you will be notified that the server is not known yet to your system and this fingerprint will be shown

```
[schewtsj@angB-158 ~]$ ssh -Y jschewts@login1.sciama.icg.port.ac.uk
The authenticity of host 'login1.sciama.icg.port.ac.uk (148.197.5.17)' can't be established.
ECDSA key fingerprint is SHA256:JZMx5thY7zQv7dVfGuG+PKcUvUVuXrdrv3nWrzJM4sw.
ECDSA key fingerprint is MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49.
Are you sure you want to continue connecting (yes/no)?
```

OS: SSH Fingerprints (cont.)

- When you log in for the first time to a server (e.g. login node), you will be notified that the server is not known yet to your system and this fingerprint will be shown

```
[schewtsj@angB-158 ~]$ ssh -Y jschewts@login1.sciama.icg.port.ac.uk
The authenticity of host 'login1.sciama.icg.port.ac.uk (148.197.5.17)' can't be established.
ECDSA key fingerprint is SHA256:JZMx5thY7zQv7dVfGuG+PKcUvUVuXrdrv3nWrzJM4sw.
ECDSA key fingerprint is MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49.
Are you sure you want to continue connecting (yes/no)?
```

- After logging in, you should confirm the validity of the server key

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login1.sciama.icg.port.ac.uk,148.197.5.17' (ECDSA) to the list of known hosts.
Last login: Sun Oct 20 19:17:47 2019 from 148.197.150.18
 =============== Welcome to Sciama 4 =================
For any questions, please consult the knowledge base at
     http://icg.port.ac.uk/support-kbtopic/sciama

[jschewts@login1(sciama) ~]$ cd /etc/ssh
[jschewts@login1(sciama) ssh]$ for file in *pub; do ssh-keygen -E md5 -lf $file; done
256 MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49 no comment (ECDSA)
256 MD5:4a:5e:80:88:85:5d:2a:c5:cd:5f:89:88:5b:21:59:d4 no comment (ED25519)
2048 MD5:05:49:7a:73:de:bb:e5:af:ef:72:8b:04:03:81:5c:b4 no comment (RSA)
[jschewts@login1(sciama) ssh]$
```

OS: SSH Fingerprints (cont.)

- When you log in for the first time to a server (e.g. login node), you will be notified that the server is not known yet to your system and this fingerprint will be shown

```
[schewtsj@angB-158 ~]$ ssh -Y jschewts@login1.sciama.icg.port.ac.uk
The authenticity of host 'login1.sciama.icg.port.ac.uk (148.197.5.17)' can't be established.
ECDSA key fingerprint is SHA256:JZMx5thY7zQv7dVfGuG+PKcUvUVuXrdrv3nWrzJM4sw.
ECDSA key fingerprint is MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49.
Are you sure you want to continue connecting (yes/no)?
```

- After logging in, you should confirm the validity of the server key

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login1.sciama.icg.port.ac.uk,148.197.5.17' (ECDSA) to the list of known hosts.
Last login: Sun Oct 20 19:17:47 2019 from 148.197.150.18
 =============== Welcome to Sciama 4 =================
For any questions, please consult the knowledge base at
     http://icg.port.ac.uk/support-kbtopic/sciama

[jschewts@login1(sciama) ~]$ cd /etc/ssh
[jschewts@login1(sciama) ssh]$ for file in *pub; do ssh-keygen -E md5 -lf $file; done
256 MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49 no comment (ECDSA)
256 MD5:4a:5e:80:88:85:5d:2a:c5:cd:5f:89:88:5b:21:59:d4 no comment (ED25519)
2048 MD5:05:49:7a:73:de:bb:e5:af:ef:72:8b:04:03:81:5c:b4 no comment (RSA)
[jschewts@login1(sciama) ssh]$
```

- This protects well against MitM attacks. Any such attempt will result in non-matching SSH fingerprints

OS: SSH Fingerprints (cont.)

- When you log in for the first time to a server (e.g. login node), you will be notified that the server is not known yet to your system and this fingerprint will be shown

```
[schewtsj@angB-158 ~]$ ssh -Y jschewts@login1.sciama.icg.port.ac.uk
The authenticity of host 'login1.sciama.icg.port.ac.uk (148.197.5.17)' can't be established.
ECDSA key fingerprint is SHA256:JZMx5thY7zQv7dVfGuG+PKcUvUVuXrdrv3nWrzJM4sw.
ECDSA key fingerprint is MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49.
Are you sure you want to continue connecting (yes/no)?
```

- After logging in, you should confirm the validity of the server key

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'login1.sciama.icg.port.ac.uk,148.197.5.17' (ECDSA) to the list of known hosts.
Last login: Sun Oct 20 19:17:47 2019 from 148.197.150.18
 =============== Welcome to Sciama 4 =================
For any questions, please consult the knowledge base at
     http://icg.port.ac.uk/support-kbtopic/sciama

[jschewts@login1(sciama) ~]$ cd /etc/ssh
[jschewts@login1(sciama) ssh]$ for file in *pub; do ssh-keygen -E md5 -lf $file; done
256 MD5:f1:8e:7f:e5:b6:03:62:77:9a:b8:8d:65:fe:ac:59:49 no comment (ECDSA)
256 MD5:4a:5e:80:88:85:5d:2a:c5:cd:5f:89:88:5b:21:59:d4 no comment (ED25519)
2048 MD5:05:49:7a:73:de:bb:e5:af:ef:72:8b:04:03:81:5c:b4 no comment (RSA)
[jschewts@login1(sciama) ssh]$
```

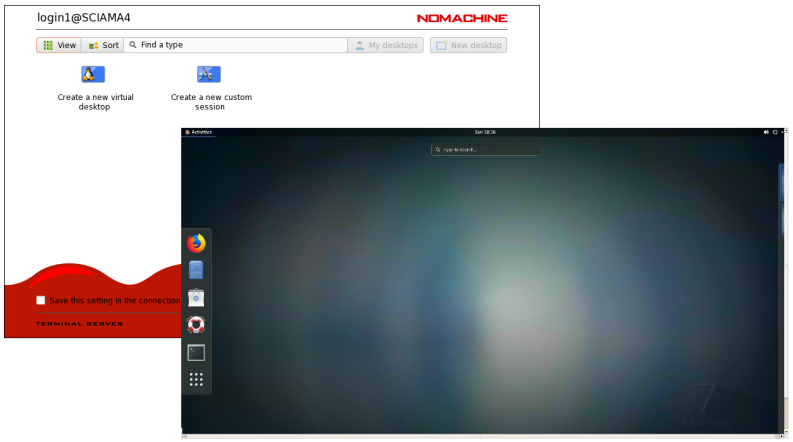
- This protects well against MitM attacks. Any such attempt will result in non-matching SSH fingerprints
- But it is less reliably against spoofing if this happens on first login and you do not know the fingerprints in advance

OS: Login

- Usually, HPC system have a couple of special login nodes accessible from the outside

OS: Login

- Usually, HPC system have a couple of special login nodes accessible from the outside
- Login nodes may support graphical (NoMachine,X2Go,etc.) and/or command-line based (rlogin,telnet,SSH,etc.) remote shells to access them




```
[schewtsj@angB-158 ~]$ ssh -Y jschewts@login1.sciama.icg.port.ac.uk
Last login: Sun Oct 20 19:17:47 2019 from 148.197.150.18
 ================ Welcome to Sciama 4 =================
For any questions, please consult the knowledge base at
     http://icg.port.ac.uk/support-kbtopic/sciama

[jschewts@login1(sciama) ~]$
```

OS: Login (Artemis)

- For Artemis, the login nodes can be found at `ood.artemis.hrc.sussex.ac.uk`

OS: Login (Artemis)

- For Artemis, the login nodes can be found at `ood.artemis.hrc.sussex.ac.uk`
- Artemis's login node supports CLI-based (SSH) remote shell access as well as a graphical user interface with remote desktop access

OS: Login (Artemis)

- For Artemis, the login nodes can be found at `ood.artemis.hrc.sussex.ac.uk`
- Artemis's login node supports CLI-based (SSH) remote shell access as well as a graphical user interface with remote desktop access
- Login nodes can be used for any work with a **SMALL** resource footprint (both memory and CPU-wise) i.e. coding, compiling, plotting, etc.

OS: Login (Artemis)

- For Artemis, the login nodes can be found at `ood.artemis.hrc.sussex.ac.uk`
- Artemis's login node supports CLI-based (SSH) remote shell access as well as a graphical user interface with remote desktop access
- Login nodes can be used for any work with a **SMALL** resource footprint (both memory and CPU-wise) i.e. coding, compiling, plotting, etc.
- Anything else **MUST** be run on the compute nodes (via `slurm`)

OS: Data Storage/Access

- Usually, HPC system have network storage to share data among the nodes as well as to provide space to store results

OS: Data Storage/Access

- Usually, HPC system have network storage to share data among the nodes as well as to provide space to store results
- Home directories with their config files are often network-mounted; so are folders containing applications & their modules

OS: Data Storage/Access

- Usually, HPC system have network storage to share data among the nodes as well as to provide space to store results
- Home directories with their config files are often network-mounted; so are folders containing applications & their modules
- Either hard- or soft-quotas may apply to all provided storage

OS: Data Storage/Access

- Usually, HPC system have network storage to share data among the nodes as well as to provide space to store results
- Home directories with their config files are often network-mounted; so are folders containing applications & their modules
- Either hard- or soft-quotas may apply to all provided storage
- Part of the storage may be backup-ed automatically in regular intervals (in which case, system admins will ask you to keep your storage footprint to the essentials to avoid wasting storage space on keeping backups on unimportant data)

OS: Data Storage/Access

- Usually, HPC system have network storage to share data among the nodes as well as to provide space to store results
- Home directories with their config files are often network-mounted; so are folders containing applications & their modules
- Either hard- or soft-quotas may apply to all provided storage
- Part of the storage may be backup-ed automatically in regular intervals (in which case, system admins will ask you to keep your storage footprint to the essentials to avoid wasting storage space on keeping backups on unimportant data)
- Additionally, many data centres also provide the possibility to do long-term backups on data tapes

OS: Data Transfer

- (For non-cloud storage,) there are various methods for transferring files between Artemis and other computers

OS: Data Transfer

- (For non-cloud storage,) there are various methods for transferring files between Artemis and other computers
- The simplest way to transfer data is using the CLI tool `scp` and the protocol of the same name (based on SSH) e.g.

```
scp
juser@ood.artemis.hrc.sussex.ac.uk:/home/juser/some.file
/Documents/
```

OS: Data Transfer

- (For non-cloud storage,) there are various methods for transferring files between Artemis and other computers
- The simplest way to transfer data is using the CLI tool `scp` and the protocol of the same name (based on SSH) e.g.

```
scp
juser@ood.artemis.hrc.sussex.ac.uk:/home/juser/some.file
/Documents/
```

- `scp` can also compress the data, thus speeding up the transfer.

OS: Data Transfer

- (For non-cloud storage,) there are various methods for transferring files between Artemis and other computers
- The simplest way to transfer data is using the CLI tool `scp` and the protocol of the same name (based on SSH) e.g.

```
scp
juser@ood.artemis.hrc.sussex.ac.uk:/home/juser/some.file
/Documents/
```

- `scp` can also compress the data, thus speeding up the transfer.
- For backups/synchronizing (remote) folders, the SSH-based CLI tool `rsync` not only compresses transfers, but also only transfers the differences between files, which may significantly reduce the amount of data transferred.

OS: Data Transfer

- (For non-cloud storage,) there are various methods for transferring files between Artemis and other computers
- The simplest way to transfer data is using the CLI tool `scp` and the protocol of the same name (based on SSH) e.g.

```
scp
juser@ood.artemis.hrc.sussex.ac.uk:/home/juser/some.file
/Documents/
```

- `scp` can also compress the data, thus speeding up the transfer.
- For backups/synchronizing (remote) folders, the SSH-based CLI tool `rsync` not only compresses transfers, but also only transfers the differences between files, which may significantly reduce the amount of data transferred.
- Alternatively, you can use the SFTP protocol, either via CLI tools or using GUI-based tools (e.g. many Linux file managers support it natively, FileZilla on Windows)

OS: Data Storage/Transfer - Cloud

- For backups, you have to store your data off-site.

OS: Data Storage/Transfer - Cloud

- For backups, you have to store your data off-site.
- A convenient way to do this is to use cloud storage e.g. GoogleDrive storage

OS: Data Transfer - Globus

- For larger amounts of data (e.g. exchange of simulation data $\sim$ TB between data centres), "standard" transfer methods are not feasible.

OS: Data Transfer - Globus

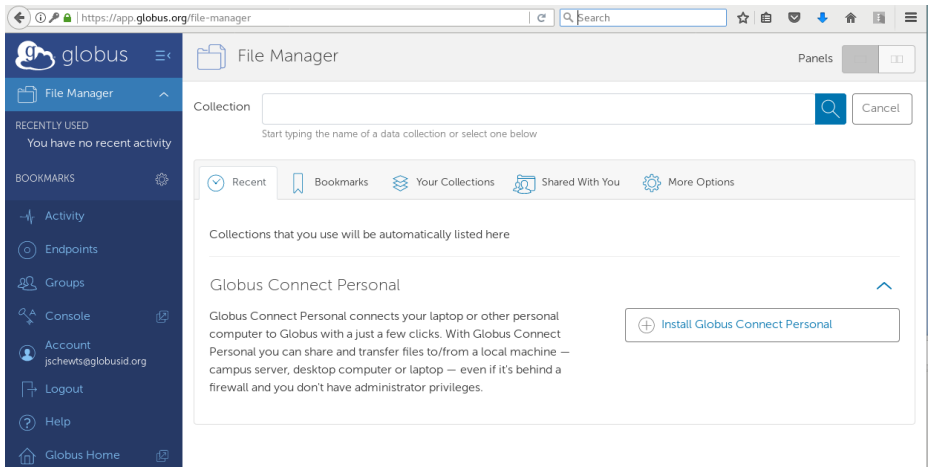
- For larger amounts of data (e.g. exchange of simulation data $\sim$ TB between data centres), "standard" transfer methods are not feasible.
- Globus provides framework to transfer large amounts of research data "efficiently, securely & reliably" (using parallel transfer protocol)

OS: Data Transfer - Globus

- For larger amounts of data (e.g. exchange of simulation data $\sim$ TB between data centres), "standard" transfer methods are not feasible.
- Globus provides framework to transfer large amounts of research data "efficiently, securely & reliably" (using parallel transfer protocol)
- Many research data centres have Globus nodes

OS: Data Transfer - Globus

- For larger amounts of data (e.g. exchange of simulation data $\sim$ TB between data centres), "standard" transfer methods are not feasible.
- Globus provides framework to transfer large amounts of research data "efficiently, securely & reliably" (using parallel transfer protocol)
- Many research data centres have Globus nodes
- Uses a web interface to manage transfers (works as a download/upload manager)



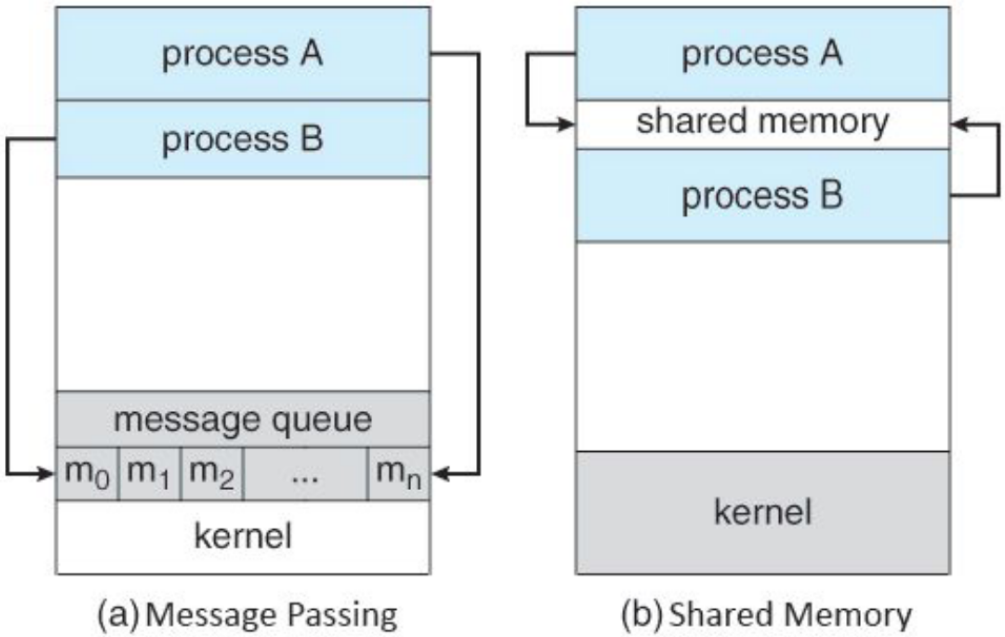
Inter-Process Communication

Inter-Process Communication

Two models of IPC

Message Passing - communication takes place by means of messages exchanged between the cooperating process.

Shared Memory - a region of memory that is shared by cooperating processes is established then exchange information takes place by reading and writing data to the shared area



MUCH more on this on Day 2 & 3 !!

Resource & Job Management

Role of Resource Manager

- Allocate resources within a cluster

Role of Resource Manager

- Allocate resources within a cluster
- Launch and manage jobs

Role of Resource Manager

- Allocate resources within a cluster
- Launch and manage jobs
- If resources required for jobs exceed available resources at the moment, a scheduling strategy is needed

Role of Job Scheduler

- When there is more work than resources, the job scheduler manages queue(s) of work

Role of Job Scheduler

- When there is more work than resources, the job scheduler manages queue(s) of work
- Usually supports complex scheduling algorithms to decide which jobs in the queue(s) are executed to optimize usage of resources:

Role of Job Scheduler

- When there is more work than resources, the job scheduler manages queue(s) of work
- Usually supports complex scheduling algorithms to decide which jobs in the queue(s) are executed to optimize usage of resources:
  - Optimized for network topology, fair-share scheduling, advanced reservations, preemption, gang scheduling, backfill scheduling, etc.

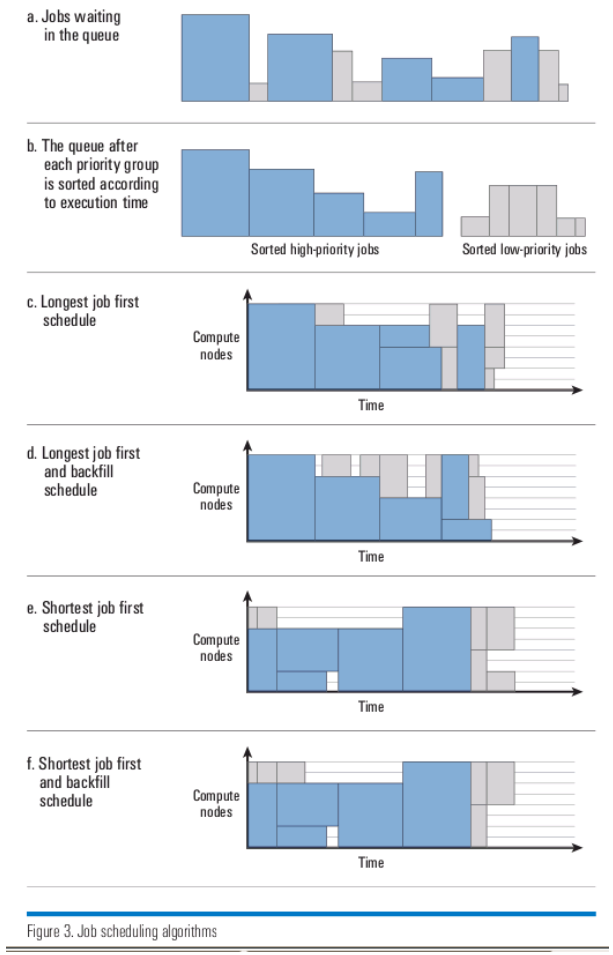


Figure 3. Job scheduling algorithms

Role of Job Scheduler

- When there is more work than resources, the job scheduler manages queue(s) of work
- Usually supports complex scheduling algorithms to decide which jobs in the queue(s) are executed to optimize usage of resources:
  - Optimized for network topology, fair-share scheduling, advanced reservations, preemption, gang scheduling, backfill scheduling, etc.
  - Job can be prioritized by e.g. job age, job partition, job size, etc.

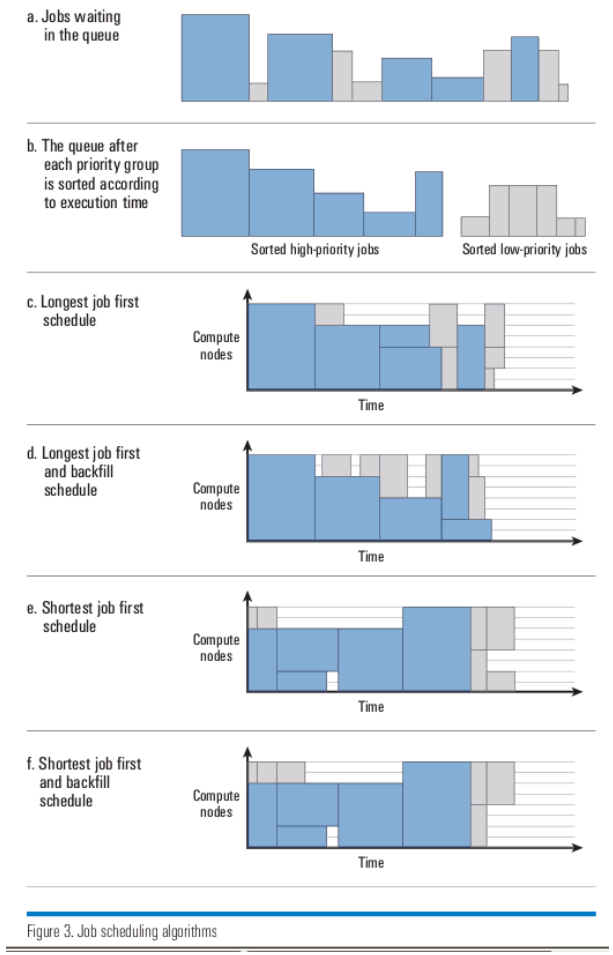


Figure 3. Job scheduling algorithms

Role of Job Scheduler

- When there is more work than resources, the job scheduler manages queue(s) of work
- Usually supports complex scheduling algorithms to decide which jobs in the queue(s) are executed to optimize usage of resources:
  - Optimized for network topology, fair-share scheduling, advanced reservations, preemption, gang scheduling, backfill scheduling, etc.
  - Job can be prioritized by e.g. job age, job partition, job size, etc.
- Supports resource limits (by queue, user, group/project, etc.)

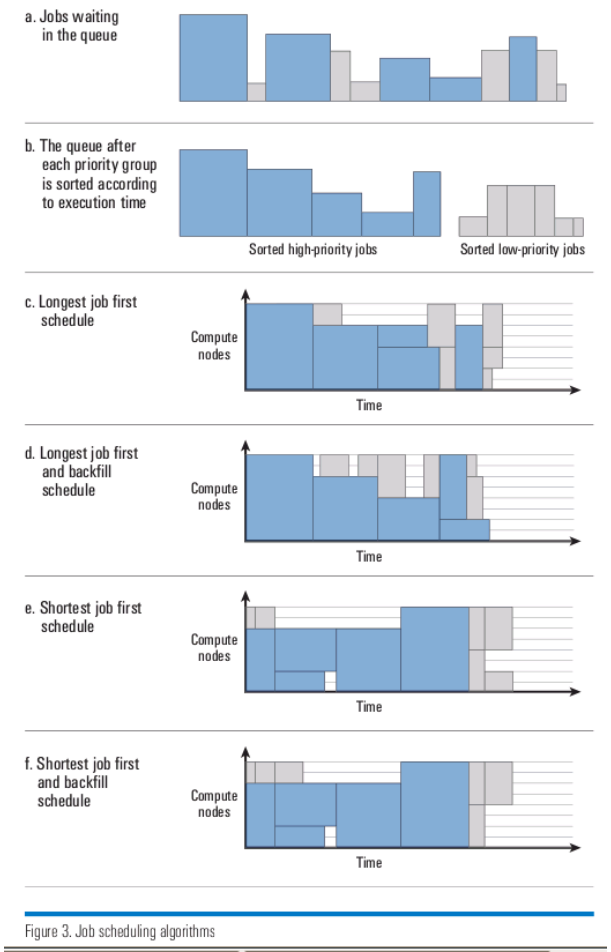


Figure 3. Job scheduling algorithms

Resource Management / Scheduling Software

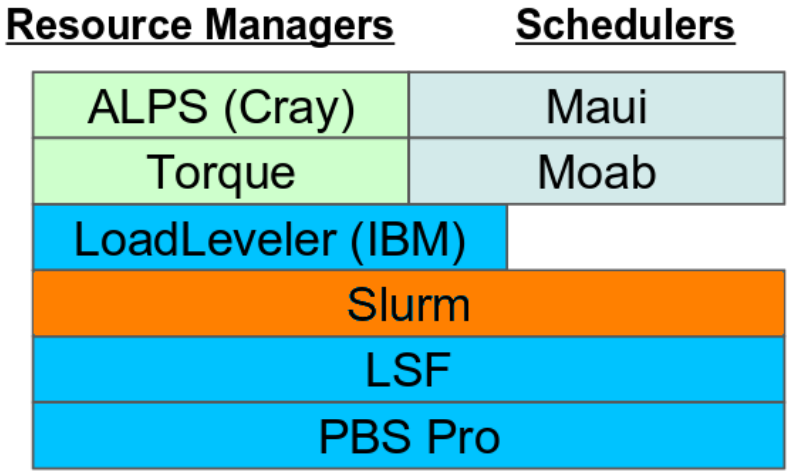
- There is a variety of software packages for HPC resource management and Job scheduling:

Resource Managers	Schedulers
ALPS (Cray)	Maui
Torque	Moab
LoadLeveler (IBM)	
Slurm	
LSF	
PBS Pro	

Many packages cover both roles.

Resource Management / Scheduling Software

- There is a variety of software packages for HPC resource management and Job scheduling:



  Many packages cover both roles.

- While you can encounter any of them out there on HPC systems, we focus here on open-source software `slurm` in particular as it is used by Artemis.

slurm: General

- Historically Slurm was an acronym standing for **S**imple **L**inux **U**tility for **R**esource **M**anagement

slurm: General

- Historically Slurm was an acronym standing for **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Development started in 2002 at Lawrence Livermore National Laboratory as a resource manager for Linux clusters

slurm: General

- Historically Slurm was an acronym standing for **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Development started in 2002 at Lawrence Livermore National Laboratory as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008

slurm: General

- Historically Slurm was an acronym standing for **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Development started in 2002 at Lawrence Livermore National Laboratory as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- Used on many of the world's largest computers (e.g. managing 3.1 million core Tianhe-2)

DISCnet

`slurm`: General

- Historically Slurm was an acronym standing for **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Development started in 2002 at Lawrence Livermore National Laboratory as a resource manager for Linux clusters
- Sophisticated scheduling plugins added in 2008
- Used on many of the world's largest computers (e.g. managing 3.1 million core Tianhe-2)
- Plugins for various MPI libraries available (i.e. MPI "talks" to `slurm` to determine number of tasks)

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

Task A job has at least one task. A task can be thought of as a single process. You may be running several processes/tasks in tandem within a job (such as with MPI)

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

Task A job has at least one task. A task can be thought of as a single process. You may be running several processes/tasks in tandem within a job (such as with MPI)

Step A job may or may not consist of one or more steps started with `srun` which run sequentially, but each step may have multiple tasks running in parallel. If started from CLI, there will be one step, in a new job. If included in a batch script, each `srun` will be a new step. Useful to attach different input (cf. `sattach`).

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

Task A job has at least one task. A task can be thought of as a single process. You may be running several processes/tasks in tandem within a job (such as with MPI)

Step A job may or may not consist of one or more steps started with `srun` which run sequentially, but each step may have multiple tasks running in parallel. If started from CLI, there will be one step, in a new job. If included in a batch script, each `srun` will be a new step. Useful to attach different input (cf. `sattach`).

Array A job may be an array job, i.e. several tasks that do not need to run in parallel and that are submitted through a single command.

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

Task A job has at least one task. A task can be thought of as a single process. You may be running several processes/tasks in tandem within a job (such as with MPI)

Step A job may or may not consist of one or more steps started with `srun` which run sequentially, but each step may have multiple tasks running in parallel. If started from CLI, there will be one step, in a new job. If included in a batch script, each `srun` will be a new step. Useful to attach different input (cf. `sattach`).

Array A job may be an array job, i.e. several tasks that do not need to run in parallel and that are submitted through a single command.

Partition A logical grouping of nodes. Partitions may overlap.

`slurm`: Terminology

Job Each `srun`, `sbatch` or `salloc` that is started from the CLI and not already part of a job creates a new job.

Task A job has at least one task. A task can be thought of as a single process. You may be running several processes/tasks in tandem within a job (such as with MPI)

Step A job may or may not consist of one or more steps started with `srun` which run sequentially, but each step may have multiple tasks running in parallel. If started from CLI, there will be one step, in a new job. If included in a batch script, each `srun` will be a new step. Useful to attach different input (cf. `sattach`).

Array A job may be an array job, i.e. several tasks that do not need to run in parallel and that are submitted through a single command.

Partition A logical grouping of nodes. Partitions may overlap.

CPU here used as a synonym for core (e.g. in `--cpus-per-task`)

`slurm`: Job Submission

sbatch This submits a background/*batch job* to the cluster (the job doesn't stay connected to the terminal); requires a job script

`slurm`: Job Submission

sbatch This submits a background/*batch job* to the cluster (the job doesn't stay connected to the terminal); requires a job script

srun This command is used for starting jobs that may be single tasks or multiple tasks in parallel. When run from CLI, `srun` blocks while job runs on compute nodes. When run inside a job, it creates a new step.

`slurm`: Job Submission

sbatch This submits a background/*batch job* to the cluster (the job doesn't stay connected to the terminal); requires a job script

srun This command is used for starting jobs that may be single tasks or multiple tasks in parallel. When run from CLI, `srun` blocks while job runs on compute nodes. When run inside a job, it creates a new step.

salloc When the system is able to allocate the requested resources, `salloc` will run the command supplied to it (by default a shell) on the system calling `salloc` (!).

`slurm`: Job Submission

sbatch This submits a background/*batch job* to the cluster (the job doesn't stay connected to the terminal); requires a job script

srun This command is used for starting jobs that may be single tasks or multiple tasks in parallel. When run from CLI, `srun` blocks while job runs on compute nodes. When run inside a job, it creates a new step.

salloc When the system is able to allocate the requested resources, `salloc` will run the command supplied to it (by default a shell) on the system calling `salloc` (!).

slurm: Job Submission / Interactive Jobs

- You can execute programs on the resources interactively by using srun e.g.

  ```
  $ srun --pty bash
  ```

  If you run it without salloc, this will try to allocate a single slot on the cluster. Otherwise, it will use on of the slots requested by salloc. You will then have to wait until the resources are available.

slurm: Job Submission / Interactive Jobs

- You can execute programs on the resources interactively by using `srun` e.g.

  ```
  $ srun --pty bash
  ```

  If you run it without `salloc`, this will try to allocate a single slot on the cluster. Otherwise, it will use on of the slots requested by `salloc`. You will then have to wait until the resources are available.
- Once the resources are allocated, a new shell on a compute node opens. The resources stay allocated until you close this shell (or when you hit your defined time limit).

slurm: Job Submission / Interactive Jobs

- You can execute programs on the resources interactively by using srun e.g.

  ```
  $ srun --pty bash
  ```

  If you run it without salloc, this will try to allocate a single slot on the cluster. Otherwise, it will use on of the slots requested by salloc. You will then have to wait until the resources are available.
- Once the resources are allocated, a new shell on a compute node opens. The resources stay allocated until you close this shell (or when you hit your defined time limit).
- The waiting time can be substantial and any loss of your ssh connection to the login node would results in loss of the allocation (request). You can use e.g. screen to prevent that (see exercises).

  ```
  $ screen -S my_useful_name
  [user@artemis-login-0 ~]$ srun --pty bash
  ```

`slurm`: Job Submission / Batch job (simple job)

- `sbatch` requires a batch script to specify the request of resources and commands to be run on these resources

slurm: Job Submission / Batch job (simple job)

- sbatch requires a batch script to specify the request of resources and commands to be run on these resources
- example script for a simple single-threaded, single-process program:

```
#!/bin/bash
#SBATCH --job-name=test_simple
#SBATCH --output=test_simple.log
#SBATCH --partition=discnet
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --time=1:00

echo "$SLURM_JOB_NODELIST #:$SLURM_NTASKS"
hostname
echo "== srun"
srun hostname
echo "== srun -n1 -N1"
srun -n1 -N1 hostname
echo "== srun -n$SLURM_NTASKS"
srun -n$SLURM_NTASKS hostname
```

slurm: Job Submission / Batch job (simple job)

- sbatch requires a batch script to specify the request of resources and commands to be run on these resources
- example script for a simple single-threaded, single-process program:

```
#!/bin/bash
#SBATCH --job-name=test_simple
#SBATCH --output=test_simple.log
#SBATCH --partition=discnet
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --time=1:00

echo "$SLURM_JOB_NODELIST #:$SLURM_NTASKS"
hostname
echo "== srun"
srun hostname
echo "== srun -n1 -N1"
srun -n1 -N1 hostname
echo "== srun -n$SLURM_NTASKS"
srun -n$SLURM_NTASKS hostname
```

Content of test_simple.log:

```
node[111,114,172,194] #:4
==
node111.pri.sciama3.alces.network
== srun
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
== srun -n1 -N1
node111.pri.sciama3.alces.network
== srun -n4
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
```

slurm: Job Submission / Batch job (simple job)

- sbatch requires a batch script to specify the request of resources and commands to be run on these resources
- example script for a simple single-threaded, single-process program:

```
#!/bin/bash
#SBATCH --job-name=test_simple
#SBATCH --output=test_simple.log
#SBATCH --partition=discnet
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --time=1:00

echo "$SLURM_JOB_NODELIST #:$SLURM_NTASKS"
hostname
echo "== srun"
srun hostname
echo "== srun -n1 -N1"
srun -n1 -N1 hostname
echo "== srun -n$SLURM_NTASKS"
srun -n$SLURM_NTASKS hostname
```

Content of test_simple.log:

```
node[111,114,172,194] #:4
==
node111.pri.sciama3.alces.network
== srun
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
== srun -n1 -N1
node111.pri.sciama3.alces.network
== srun -n4
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
```

- there is a login (here running on node111) for bootstraping

slurm: Job Submission / Batch job (simple job)

- sbatch requires a batch script to specify the request of resources and commands to be run on these resources
- example script for a simple single-threaded, single-process program:

```
#!/bin/bash
#SBATCH --job-name=test_simple
#SBATCH --output=test_simple.log
#SBATCH --partition=discnet
#SBATCH --nodes=4
#SBATCH --ntasks=4
#SBATCH --time=1:00

echo "$SLURM_JOB_NODELIST #:$SLURM_NTASKS"
hostname
echo "== srun"
srun hostname
echo "== srun -n1 -N1"
srun -n1 -N1 hostname
echo "== srun -n$SLURM_NTASKS"
srun -n$SLURM_NTASKS hostname
```

Content of test_simple.log:

```
node[111,114,172,194] #:4
==
node111.pri.sciama3.alces.network
== srun
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
== srun -n1 -N1
node111.pri.sciama3.alces.network
== srun -n4
node111.pri.sciama3.alces.network
node114.pri.sciama3.alces.network
node172.pri.sciama3.alces.network
node194.pri.sciama3.alces.network
```

- there is a login (here running on node111) for bootstraping
- Make sure to wrap your commands into srun if you want them to run on any other than the login node (actually, nothing but VERY light-weight tasks should be run without srun)

slurm: Job Submission / Batch job (multi-threading)

- example script for a multi-threaded, single-process program:

```
#!/bin/bash
#
#SBATCH --job-name=test_threading
#SBATCH --output=test_threading.log.%j
#
#SBATCH --partition=discnet
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --time=1:00

module purge
module load intel/2022a

echo "#:$SLURM_NTASKS *:$SLURM_CPUS_PER_TASK"
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo "=="
./test_threading
echo "== srun"
srun ./test_threading
```

slurm: Job Submission / Batch job (multi-threading)

- example script for a multi-threaded, single-process program:

```
#!/bin/bash
#
#SBATCH --job-name=test_threading
#SBATCH --output=test_threading.log.%j
#
#SBATCH --partition=discnet
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --time=1:00

module purge
module load intel/2022a

echo "#:$SLURM_NTASKS *:$SLURM_CPUS_PER_TASK"
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

echo "=="
./test_threading
echo "== srun"
srun ./test_threading
```

- The %j in the output file pattern will be substituted by the JOB_ID (i.e. make it easier to manage output file from multiple submissions of the same script)

slurm: Job Submission / Batch job (multi-processing)

- example script for a multi-threaded, single-process program:

```
#!/bin/bash
#
#SBATCH --job-name=test_mpi
#SBATCH --output=test_mpi.log.%j.%t
#
#SBATCH --partition=discnet
#SBATCH --ntasks=4
#SBATCH --time=1:00

module purge
module load intel/2022a
module load OpenMPI/4.1.4-GCC-11.3.0

echo $SLURM_JOB_NODELIST
echo "#:$SLURM_NTASKS *: $SLURM_NTASKS"

echo "=="
mpirun ./test_mpi
echo "== srun"
srun --mpi=pmi2 ./test_mpi
```

slurm: Job Submission / Batch job (multi-processing)

- example script for a multi-threaded, single-process program:

```
#!/bin/bash
#
#SBATCH --job-name=test_mpi
#SBATCH --output=test_mpi.log.%j.%t
#
#SBATCH --partition=discnet
#SBATCH --ntasks=4
#SBATCH --time=1:00

module purge
module load intel/2022a
module load OpenMPI/4.1.4-GCC-11.3.0

echo $SLURM_JOB_NODELIST
echo "#:$SLURM_NTASKS *: $SLURM_NTASKS"

echo "=="
mpirun ./test_mpi
echo "== srun"
srun --mpi=pmi2 ./test_mpi
```

- both methods work without explicitly passing on the number of tasks (MPI "talks" to slurm)

slurm: Job Submission / Batch job (multi-processing)

- example script for a multi-threaded, single-process program:

```
#!/bin/bash
#
#SBATCH --job-name=test_mpi
#SBATCH --output=test_mpi.log.%j.%t
#
#SBATCH --partition=discnet
#SBATCH --ntasks=4
#SBATCH --time=1:00

module purge
module load intel/2022a
module load OpenMPI/4.1.4-GCC-11.3.0

echo $SLURM_JOB_NODELIST
echo "#:$SLURM_NTASKS *: $SLURM_NTASKS"

echo "=="
mpirun ./test_mpi
echo "== srun"
srun --mpi=pmi2 ./test_mpi
```

- both methods work without explicitly passing on the number of tasks (MPI "talks" to `slurm`)
- it is preferable to use `srun` rather than `mpirun` for bootstrapping (but skip the `--mpi=pmi2` for Intel MPI as it is not supported)

slurm: Job Submission / Batch job (Arrays)

- To submit a lot of similar tasks efficiently, you can use job arrays:

```
# Submit a job array with index values between 0 and 31
$ sbatch --array=0-31  <batch file>

# Submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 <batch file>

# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
$ sbatch --array=1-7:2  <batch file>

# Submit a job array with index values between 1 and 7
# but limit the number of simultaniously running tasks to 4
$ sbatch --array=1-7%4  <batch file>
```

slurm: Job Submission / Batch job (Arrays)

- To submit a lot of similar tasks efficiently, you can use job arrays:

```
# Submit a job array with index values between 0 and 31
$ sbatch --array=0-31  <batch file>

# Submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 <batch file>

# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
$ sbatch --array=1-7:2  <batch file>

# Submit a job array with index values between 1 and 7
# but limit the number of simultaniously running tasks to 4
$ sbatch --array=1-7%4  <batch file>
```

- you can then use e.g. the env variable SLURM_ARRAY_JOB_ID to assign the right data to each of the jobs inside the array's batch script or your program

slurm: Job Submission / Batch job (Arrays)

- To submit a lot of similar tasks efficiently, you can use job arrays:

```
# Submit a job array with index values between 0 and 31
$ sbatch --array=0-31  <batch file>

# Submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 <batch file>

# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
$ sbatch --array=1-7:2  <batch file>

# Submit a job array with index values between 1 and 7
# but limit the number of simultaniously running tasks to 4
$ sbatch --array=1-7%4  <batch file>
```

- you can then use e.g. the env variable SLURM_ARRAY_JOB_ID to assign the right data to each of the jobs inside the array's batch script or your program
- it also helps to keep the scheduler/queues not being overwhelmed by 100s of single submissions/job requests

`slurm`: Job Submission / Batch job (Dependencies)

- Sometimes, jobs may require other jobs to run, their results or are only required to run if another job fails

slurm: Job Submission / Batch job (Dependencies)

- Sometimes, jobs may require other jobs to run, their results or are only required to run if another job fails
- slurm allows to define dependencies for those cases

```
# Wait for specific job to be started
sbatch --depend=after:123 my.job

# Wait for jobs to complete
sbatch --depend=afterany:123:126 my.job

# Wait for jobs to complete successfully
sbatch --depend=afterok:123 my.job

# Wait for job / entire job array to complete and at least one task fails
sbatch --depend=afternotok:123 my.job
```

slurm: Job Submission / Batch job (Dependencies)

- Sometimes, jobs may require other jobs to run, their results or are only required to run if another job fails
- slurm allows to define dependencies for those cases

```
# Wait for specific job to be started
sbatch --depend=after:123 my.job

# Wait for jobs to complete
sbatch --depend=afterany:123:126 my.job

# Wait for jobs to complete successfully
sbatch --depend=afterok:123 my.job

# Wait for job / entire job array to complete and at least one task fails
sbatch --depend=afternotok:123 my.job
```

- You can create complex dependencies by combining conditions e.g.

```
# Wait for specific jobs to be started and another to fail
sbatch --depend=after:123:126,afternotok:125 my.job
```

`slurm`: Additional arguments

There are many additional arguments that can be passed to the resource manager:

- Scheduling/resource allocation:

  `--nodes=<N>` / `--nodes=<N-M>` Request that a minimum of N (and a maximum of M) nodes be allocated to this job

  `--tasks-per-node=<N>` Requests that (a maximum of) N tasks be invoked on each node

  `--mem=<size>` / `--mem-per-cpu=<size>` Specify the real memory required per node / allocated core

  `--exclusive` Requests, that nodes must ont be shared with other running jobs

`slurm`: Additional arguments

There are many additional arguments that can be passed to the resource manager:

- Scheduling/resource allocation:
  - `--nodes=<N>` / `--nodes=<N-M>` Request that a minimum of N (and a maximum of M) nodes be allocated to this job
  - `--tasks-per-node=<N>` Requests that (a maximum of) N tasks be invoked on each node
  - `--mem=<size>` / `--mem-per-cpu=<size>` Specify the real memory required per node / allocated core
  - `--exclusive` Requests, that nodes must ont be shared with other running jobs
- Logging:
  - `--error=<filename>` Instruct Slurm to connect stderr directly to the file specified (by default same as `--output`)
  - `--mail-type=<type>` Requests notifications by email to user (email address stored in system)

slurm: Resources/Accounting (sinfo)

- The command `sinfo` lists the nodes and their states belonging to the various partitions (aka queues) of the computational resources.

```
[jschewts@login1(sciama) ~]$ sinfo
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
sciama4.q       up   infinite      1    mix node304
sciama4.q       up   infinite      8  alloc node[300-303,308-311]
sciama4.q       up   infinite      3   idle node[305-307]
sciama4-12.q    up   infinite      1  drain node312
sciama4-12.q    up   infinite      1    mix node315
sciama4-12.q    up   infinite      2  alloc node[313-314]
sciama4-12.q    up   infinite     12   idle node[316-327]
sciama2.q*      up   infinite      1 drain* node125
sciama2.q*      up   infinite      2  down* node[100,194]
sciama2.q*      up   infinite      1  drain node137
sciama2.q*      up   infinite     17    mix node[101-105,127-129,158,162,169,172,178-180,190-191]
sciama2.q*      up   infinite     73  alloc node[106-124,126,130-136,138-157,159-161,163-168,170-171,173-177,181-189,192]
sciama2.q*      up   infinite      1   idle node193
sciama3.q       up   infinite      1  drain node200
sciama3.q       up   infinite     16    mix node[201-212,225-228]
sciama3.q       up   infinite     16  alloc node[213-224,229-232]
sciama3.q       up   infinite     15   idle node[233-247]
himem.q         up   infinite      1   idle vhmem01
rsm1.q          up   infinite      2    mix node[190-191]
rsm1.q          up   infinite      5  alloc node[186-189,192]
rsm1.q          up   infinite      1   idle node193
```

slurm: Resources/Accounting (sinfo)

- The command `sinfo` lists the nodes and their states belonging to the various partitions (aka queues) of the computational resources.

```
[jschewts@login1(sciama) ~]$ sinfo
PARTITION    AVAIL  TIMELIMIT  NODES  STATE NODELIST
sciama4.q       up   infinite      1    mix node304
sciama4.q       up   infinite      8  alloc node[300-303,308-311]
sciama4.q       up   infinite      3   idle node[305-307]
sciama4-12.q    up   infinite      1  drain node312
sciama4-12.q    up   infinite      1    mix node315
sciama4-12.q    up   infinite      2  alloc node[313-314]
sciama4-12.q    up   infinite     12   idle node[316-327]
sciama2.q*      up   infinite      1 drain* node125
sciama2.q*      up   infinite      2  down* node[100,194]
sciama2.q*      up   infinite      1  drain node137
sciama2.q*      up   infinite     17    mix node[101-105,127-129,158,162,169,172,178-180,190-191]
sciama2.q*      up   infinite     73  alloc node[106-124,126,130-136,138-157,159-161,163-168,170-171,173-177,181-189,192]
sciama2.q*      up   infinite      1   idle node193
sciama3.q       up   infinite      1  drain node200
sciama3.q       up   infinite     16    mix node[201-212,225-228]
sciama3.q       up   infinite     16  alloc node[213-224,229-232]
sciama3.q       up   infinite     15   idle node[233-247]
himem.q         up   infinite      1   idle vhmem01
rsm1.q          up   infinite      2    mix node[190-191]
rsm1.q          up   infinite      5  alloc node[186-189,192]
rsm1.q          up   infinite      1   idle node193
```

- There are various states, nodes can be in: e.g. `alloc`/`mixed`/`idle`, `drng`/`drain`/`down`; For the latter, reasons are provided

```
[jschewts@login1(sciama) ~]$ sinfo -R
REASON               USER      TIMESTAMP           NODELIST
Investigation        jschewts  2019-10-01T20:39:31 node312
Infiniband fault     root      2019-07-30T14:56:26 node125
Not responding       nobody    2019-08-23T19:36:50 node100
Not responding       nobody    2019-10-11T14:44:56 node194
batch job complete f nobody    2019-08-28T07:20:28 node137
investigation lustre jschewts  2019-10-19T12:44:19 node200
```

slurm: Resources/Accounting (sacct)

- There is a record of each job a user submits/runs

```
[jschewts@login1(sciama) ~]$ sacct --format=User,JobID,Jobname,partition,state,time,start,end,elapsed,MaxRss,MaxVMSize,nnodes,ncpus,nodelist -u [user]
     User        JobID    JobName  Partition      State  Timelimit               Start                 End    Elapsed     MaxRSS  MaxVMSize   NNodes      NCPUS        NodeList
--------- ------------ ---------- ---------- ---------- ---------- ------------------- ------------------- ---------- ---------- ---------- -------- ---------- ---------------
   [user] 1135959         starccm  sciama4.q    RUNNING 2-00:00:00 2019-10-22T15:17:11             Unknown 1-02:14:01                             1          8         node300
          1135959.0          bash               RUNNING            2019-10-22T15:17:40             Unknown 1-02:13:32                             1          8         node300
   [user] 1153763         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:44:31 2019-10-23T14:47:47   00:03:16                             1          8         node308
          1153763.bat+      batch             CANCELLED            2019-10-23T14:44:31 2019-10-23T14:47:48   00:03:17      7492K    414804K        1          8         node308
          1153763.0          bash             CANCELLED            2019-10-23T14:44:54 2019-10-23T14:47:47   00:02:53    375156K   3085040K        1          8         node308
   [user] 1153770         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:47:31 2019-10-23T14:49:59   00:02:28                             1          8         node308
          1153770.bat+      batch             CANCELLED            2019-10-23T14:47:31 2019-10-23T14:50:00   00:02:29      7496K    414804K        1          8         node308
          1153770.0          bash             CANCELLED            2019-10-23T14:47:52 2019-10-23T14:50:00   00:02:08    356672K   3075008K        1          8         node308
   [user] 1153778         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22                             1          8         node308
          1153778.bat+      batch              COMPLETED            2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22      1360K    157252K        1          8         node308
          1153778.0          bash              COMPLETED            2019-10-23T14:50:22 2019-10-23T14:50:24   00:00:02      1152K    225628K        1          8         node308
   [user] 1153805         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22                             1          8         node308
          1153805.bat+      batch              COMPLETED            2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22      1360K    157252K        1          8         node308
          1153805.0          bash              COMPLETED            2019-10-23T14:53:22 2019-10-23T14:53:24   00:00:02      1148K    225628K        1          8         node308
   [user] 1153825         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T15:08:04 2019-10-23T15:08:30   00:00:26                             1          8         node309
```

slurm: Resources/Accounting (sacct)

- There is a record of each job a user submits/runs

```
[jschewts@login1(sciama) ~]$ sacct --format=User,JobID,Jobname,partition,state,time,start,end,elapsed,MaxRss,MaxVMSize,nnodes,ncpus,nodelist -u [redacted]
     User        JobID    JobName  Partition      State  Timelimit               Start                 End    Elapsed     MaxRSS  MaxVMSize   NNodes      NCPUS        NodeList
--------- ------------ ---------- ---------- ---------- ---------- ------------------- ------------------- ---------- ---------- ---------- -------- ---------- ---------------
[redacted] 1135959         starccm  sciama4.q    RUNNING 2-00:00:00 2019-10-22T15:17:11             Unknown 1-02:14:01                               1          8         node300
           1135959.0          bash               RUNNING            2019-10-22T15:17:40             Unknown 1-02:13:32                               1          8         node300
[redacted] 1153763         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:44:31 2019-10-23T14:47:47   00:03:16                               1          8         node308
           1153763.bat+      batch             CANCELLED            2019-10-23T14:44:31 2019-10-23T14:47:48   00:03:17      7492K    414804K        1          8         node308
           1153763.0          bash             CANCELLED            2019-10-23T14:44:54 2019-10-23T14:47:47   00:02:53    375156K   3085040K        1          8         node308
[redacted] 1153770         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:47:31 2019-10-23T14:49:59   00:02:28                               1          8         node308
           1153770.bat+      batch             CANCELLED            2019-10-23T14:47:31 2019-10-23T14:50:00   00:02:29      7496K    414804K        1          8         node308
           1153770.0          bash             CANCELLED            2019-10-23T14:47:52 2019-10-23T14:50:00   00:02:08    356672K   3075008K        1          8         node308
[redacted] 1153778         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22                               1          8         node308
           1153778.bat+      batch              COMPLETED            2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22      1360K    157252K        1          8         node308
           1153778.0          bash              COMPLETED            2019-10-23T14:50:22 2019-10-23T14:50:24   00:00:02      1152K    225628K        1          8         node308
[redacted] 1153805         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22                               1          8         node308
           1153805.bat+      batch              COMPLETED            2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22      1360K    157252K        1          8         node308
           1153805.0          bash              COMPLETED            2019-10-23T14:53:22 2019-10-23T14:53:24   00:00:02      1148K    225628K        1          8         node308
[redacted] 1153825         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T15:08:04 2019-10-23T15:08:30   00:00:26                               1          8         node309
```

- on public HPC framework, this accounting is used to keep track/limit resource usage for charging users/project

slurm: Resources/Accounting (sacct)

- There is a record of each job a user submits/runs

```
[jschewts@login1(sciama) ~]$ sacct --format=User,JobID,Jobname,partition,state,time,start,end,elapsed,MaxRss,MaxVMSize,nnodes,ncpus,nodelist -u [redacted]
     User        JobID    JobName  Partition      State  Timelimit               Start                 End    Elapsed     MaxRSS  MaxVMSize   NNodes      NCPUS        NodeList
--------- ------------ ---------- ---------- ---------- ---------- ------------------- ------------------- ---------- ---------- ---------- -------- ---------- ---------------
[redacted] 1135959         starccm  sciama4.q    RUNNING 2-00:00:00 2019-10-22T15:17:11             Unknown 1-02:14:01                               1          8         node300
           1135959.0          bash               RUNNING            2019-10-22T15:17:40             Unknown 1-02:13:32                               1          8         node300
[redacted] 1153763         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:44:31 2019-10-23T14:47:47   00:03:16                               1          8         node308
           1153763.bat+      batch             CANCELLED            2019-10-23T14:44:31 2019-10-23T14:47:48   00:03:17      7492K    414804K        1          8         node308
           1153763.0          bash             CANCELLED            2019-10-23T14:44:54 2019-10-23T14:47:47   00:02:53    375156K   3085040K        1          8         node308
[redacted] 1153770         starccm  sciama4.q CANCELLED+ 2-00:00:00 2019-10-23T14:47:31 2019-10-23T14:49:59   00:02:28                               1          8         node308
           1153770.bat+      batch             CANCELLED            2019-10-23T14:47:31 2019-10-23T14:50:00   00:02:29      7496K    414804K        1          8         node308
           1153770.0          bash             CANCELLED            2019-10-23T14:47:52 2019-10-23T14:50:00   00:02:08    356672K   3075008K        1          8         node308
[redacted] 1153778         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22                               1          8         node308
           1153778.bat+      batch              COMPLETED            2019-10-23T14:50:02 2019-10-23T14:50:24   00:00:22      1360K    157252K        1          8         node308
           1153778.0          bash              COMPLETED            2019-10-23T14:50:22 2019-10-23T14:50:24   00:00:02      1152K    225628K        1          8         node308
[redacted] 1153805         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22                               1          8         node308
           1153805.bat+      batch              COMPLETED            2019-10-23T14:53:02 2019-10-23T14:53:24   00:00:22      1360K    157252K        1          8         node308
           1153805.0          bash              COMPLETED            2019-10-23T14:53:22 2019-10-23T14:53:24   00:00:02      1148K    225628K        1          8         node308
[redacted] 1153825         starccm  sciama4.q  COMPLETED 2-00:00:00 2019-10-23T15:08:04 2019-10-23T15:08:30   00:00:26                               1          8         node309
```

- on public HPC framework, this accounting is used to keep track/limit resource usage for charging users/project
- on Artemis, it can be used for debugging or obtaining useful statistics to optimize your resource requests (e.g. `MaxRSS` size tells you about your actual memory requirements)

Resource Management / Scheduling: Controlling

- Once you submitted a job, you can keep track of its process using the command `squeue` (e.g. use `squeue -u <username>` to list all of the active (i.e. queued or running) jobs of a specific user

Resource Management / Scheduling: Controlling

- Once you submitted a job, you can keep track of its process using the command `squeue` (e.g. use `squeue -u <username>` to list all of the active (i.e. queued or running) jobs of a specific user
- You can also cancel steps within jobs (or if there is only one step), using `scancel <jobid>`.

Resource Management / Scheduling: Controlling

- Once you submitted a job, you can keep track of its process using the command `squeue` (e.g. use `squeue -u <username>` to list all of the active (i.e. queued or running) jobs of a specific user
- You can also cancel steps within jobs (or if there is only one step), using `scancel <jobid>`.
- Furthermore, `scontrol` allows you to make changes to your submitted jobs (e.g. holding them in the queue)

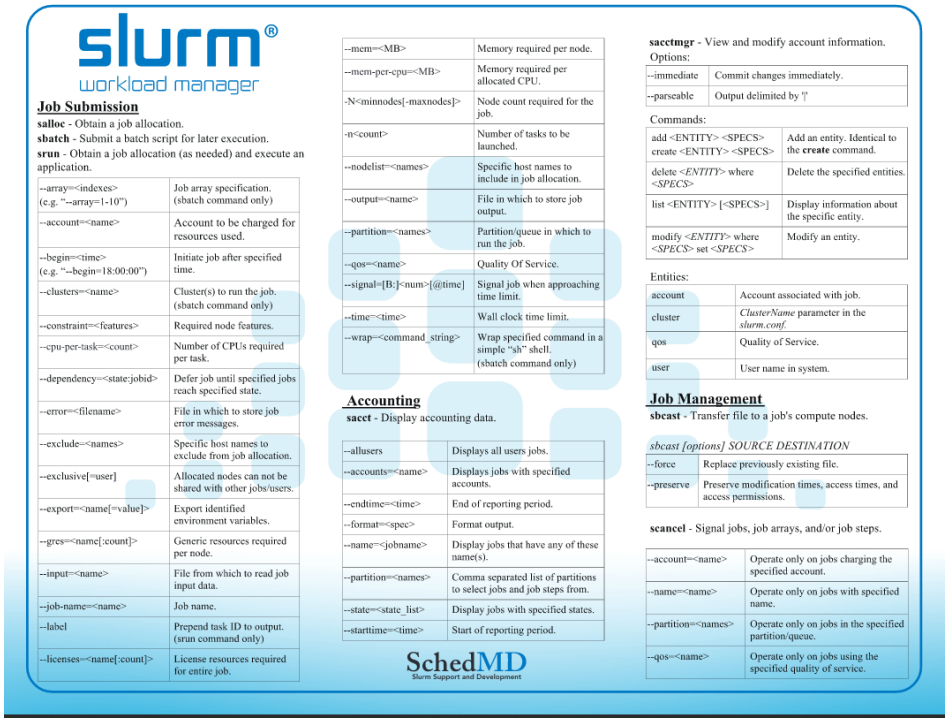
`slurm`: Man Pages / Cheat Sheet

- For more details on all the presented commands, please check their man pages (e.g. `man sbatch`)

slurm: Man Pages / Cheat Sheet

- For more details on all the presented commands, please check their man pages (e.g. `man sbatch`)
- There is a also handy slurm cheat sheet which you can download from e.g.

`https://slurm.schedmd.com/pdfs/summary.pdf`



slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

--array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
--account=<name>	Account to be charged for resources used.
--begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
--clusters=<name>	Cluster(s) to run the job. (sbatch command only)
--constraint=<features>	Required node features.
--cpu-per-task=<count>	Number of CPUs required per task.
--dependency=<state:jobid>	Defer job until specified jobs reach specified state.
--error=<filename>	File in which to store job error messages.
--exclude=<names>	Specific host names to exclude from job allocation.
--exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
--export=<name[=value]>	Export identified environment variables.
--gres=<name[:count]>	Generic resources required per node.
--input=<name>	File from which to read job input data.
--job-name=<name>	Job name.
--label	Prepend task ID to output. (srun command only)
--licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by |

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete <*ENTITY*> where <*SPECS*>	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify <*ENTITY*> where <*SPECS*> set <*SPECS*>	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

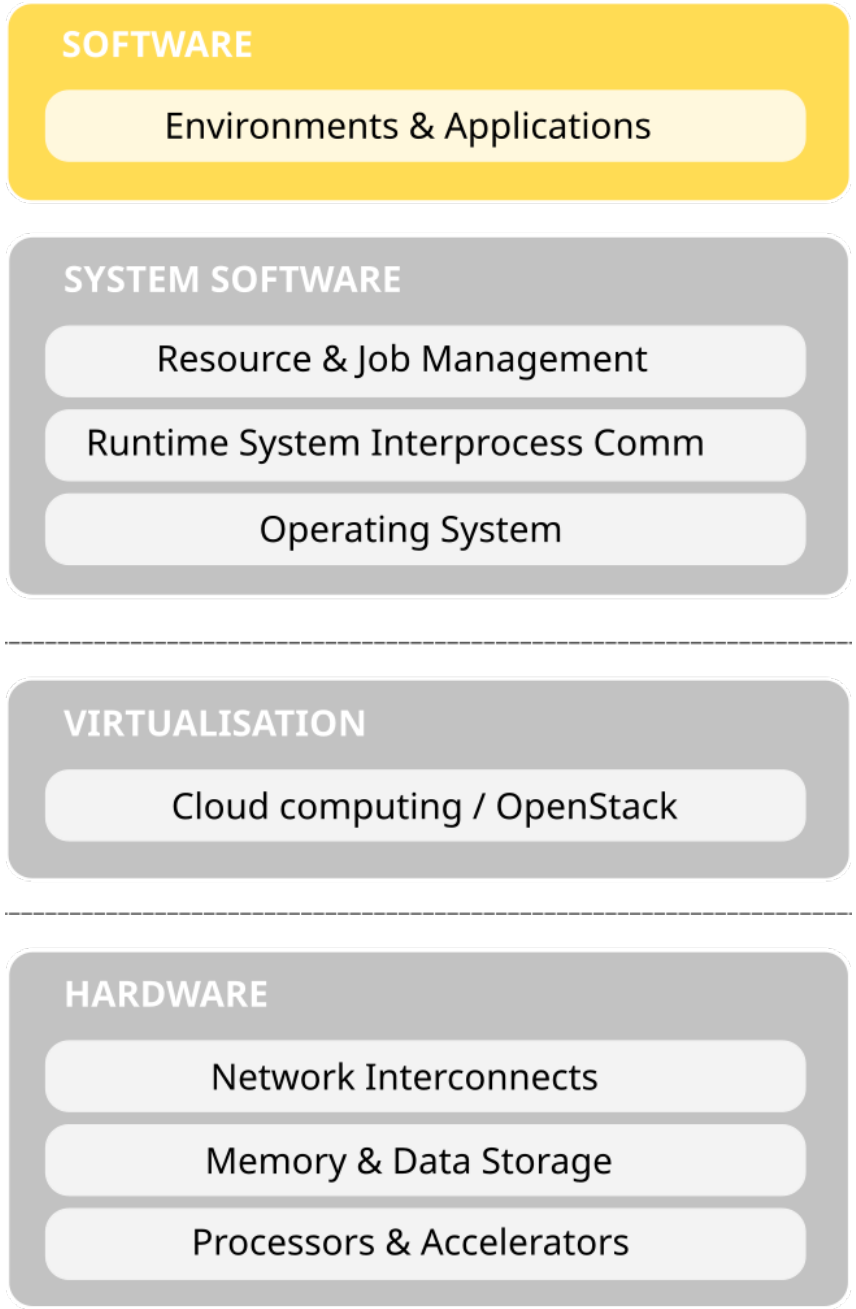
sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.


SOFTWARE
Environments & Applications
SYSTEM SOFTWARE
Resource & Job Management
Runtime System Interprocess Comm
Operating System
VIRTUALISATION
Cloud computing / OpenStack
HARDWARE
Network Interconnects
Memory & Data Storage
Processors & Accelerators

Environments & Applications

System Environment & Multi-user systems

- Single-user computer usually have a single system environment (i.e. each program and library exists in a single version/configuration)

System Environment & Multi-user systems

- Single-user computer usually have a single system environment (i.e. each program and library exists in a single version/configuration)
- Multi-user systems require a more complex setup: Users may require either different conflicting libraries or different versions of the same program/library

System Environment & Multi-user systems

- Single-user computer usually have a single system environment (i.e. each program and library exists in a single version/configuration)
- Multi-user systems require a more complex setup: Users may require either different conflicting libraries or different versions of the same program/library
- Often, even the same user faces this problem to need different setups for different software

System Environment & Multi-user systems

- Single-user computer usually have a single system environment (i.e. each program and library exists in a single version/configuration)
- Multi-user systems require a more complex setup: Users may require either different conflicting libraries or different versions of the same program/library
- Often, even the same user faces this problem to need different setups for different software
- Thus, it requires a way to adapt a system environment for each user/purpose: Environment variables / modules

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)
- In `bash`, these are set by a statement

```
export <VARNAME>=<VALUE>
```

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)
- In `bash`, these are set by a statement

  `export <VARNAME>=<VALUE>`
- Environmental variables are only visible within the same shell, they were defined in (`.bashrc` offers a way to set 'default' variables).

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)
- In `bash`, these are set by a statement

  `export <VARNAME>=<VALUE>`
- Environmental variables are only visible within the same shell, they were defined in (`.bashrc` offers a way to set 'default' variables).
- The `export` statement is optional, but necessary, if this environmental variable should be also visible within processes spawned by that shell (e.g. programs) (if in doubt, use it)

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)
- In `bash`, these are set by a statement

  `export <VARNAME>=<VALUE>`
- Environmental variables are only visible within the same shell, they were defined in (`.bashrc` offers a way to set 'default' variables).
- The `export` statement is optional, but necessary, if this environmental variable should be also visible within processes spawned by that shell (e.g. programs) (if in doubt, use it)
- `$<VARNAME>` returns the value of the variable (i.e. `echo $<VARNAME>` would then print its value)

Environment variables: General

- Environment variables are used in Unix-like shells to store configuration settings (for the shell and programs to be used at runtime)
- In `bash`, these are set by a statement

  `export <VARNAME>=<VALUE>`
- Environmental variables are only visible within the same shell, they were defined in (`.bashrc` offers a way to set 'default' variables).
- The `export` statement is optional, but necessary, if this environmental variable should be also visible within processes spawned by that shell (e.g. programs) (if in doubt, use it)
- `$<VARNAME>` returns the value of the variable (i.e. `echo $<VARNAME>` would then print its value)
- `env` prints all environmental variables

Environment variables: `PATH`

- One of the most important variables for the shell itself is `PATH`

Environment variables: `PATH`

- One of the most important variables for the shell itself is `PATH`
- It contains a colon-seperated list of folders that are searched by the shell for commands/programs to be executed

Environment variables: `PATH`

- One of the most important variables for the shell itself is `PATH`
- It contains a colon-seperated list of folders that are searched by the shell for commands/programs to be executed
- If you install a new program in a folder that is not already listed, but want it available in the shell as command (without providing the path to it), you have to prepend the folder containing the program binary/script to the path

Environment variables: `PATH`

- One of the most important variables for the shell itself is `PATH`
- It contains a colon-seperated list of folders that are searched by the shell for commands/programs to be executed
- If you install a new program in a folder that is not already listed, but want it available in the shell as command (without providing the path to it), you have to prepend the folder containing the program binary/script to the path
- the first occurence found in `PATH` is used

```
export PATH=<INSTALL PATH>/bin:$PATH
```

Environment variables: `LDFLAGS`,`CFLAGS`,`FFLAGS`,etc.

- For compilers, there are a set of variables containing compiler arguments to point to headers, libraries, etc.

Environment variables: `LDFLAGS`,`CFLAGS`,`FFLAGS`,etc.

- For compilers, there are a set of variables containing compiler arguments to point to headers, libraries, etc.
- Similar to `PATH`, if you want to make custom-installed libraries available to the compiler, the resp. `include`, `lib`, etc. folders have to be added to the variables e.g.

```
export CFLAGS="-I<INSTALL PATH>/include $CFLAGS"
```

or

```
export LDFLAGS="-L<LIB PATH> -Wl,-rpath=<LIB PATH> $LDFLAGS"
```

Modules: General

- So-called (environmental) modules in Linux/Unix are (mostly) a convenient way to customize these variables (and thus the system environment) automatically

Modules: General

- So-called (environmental) modules in Linux/Unix are (mostly) a convenient way to customize these variables (and thus the system environment) automatically
- Loading the module for a specific program or library will amend the variables as described above, while unloading it, removes these alterations again

Modules: General

- So-called (environmental) modules in Linux/Unix are (mostly) a convenient way to customize these variables (and thus the system environment) automatically
- Loading the module for a specific program or library will amend the variables as described above, while unloading it, removes these alterations again
- This allows to switch out e.g. different versions of the same library

Modules: General

- So-called (environmental) modules in Linux/Unix are (mostly) a convenient way to customize these variables (and thus the system environment) automatically
- Loading the module for a specific program or library will amend the variables as described above, while unloading it, removes these alterations again
- This allows to switch out e.g. different versions of the same library
- Additionally, some built-in commands and additional Tcl scripting allows to check for potential conflicts of libraries and missing dependencies

Modules: General

- So-called (environmental) modules in Linux/Unix are (mostly) a convenient way to customize these variables (and thus the system environment) automatically
- Loading the module for a specific program or library will amend the variables as described above, while unloading it, removes these alterations again
- This allows to switch out e.g. different versions of the same library
- Additionally, some built-in commands and additional Tcl scripting allows to check for potential conflicts of libraries and missing dependencies
- `MODULESPATH` is a list of folders checked for modules (similar to `PATH` for commands). You can append it to add custom modules

Modules: Usage

- There are a few important commands:

  `module av` lists available modules
  `module load <modulename>` loads a module
  `module unload <modulename>` unloads a module
  `module list` lists all currently loaded modules
  `module spider <string>` lists all modules matching search string
  `module purge` unloads all loaded modules
  `module help <modulename>` shows a help page for the module
  `module show <modulename>` lists content of module

Modules: Usage

- There are a few important commands:

  `module av` lists available modules
  `module load <modulename>` loads a module
  `module unload <modulename>` unloads a module
  `module list` lists all currently loaded modules
  `module spider <string>` lists all modules matching search string
  `module purge` unloads all loaded modules
  `module help <modulename>` shows a help page for the module
  `module show <modulename>` lists content of module

- if you want to have certain modules loaded by default, simply modify the `.bashrc` and/or `.bash_profile` in your home directory accordingly

Modules: Artemis

- Artemis uses LMod (i.e. Lua-based reimplementation of original TCL/L-based module system)

```
[js2347@artemis-login-0 ~]$ module av

------------------------------------------------------------------------------ /opt/ohpc/pub/modulefiles -------------------------------------------------------------
   gnu12/12.2.0    hwloc/2.11.1 (D)    libfabric/1.18.0    os    papi/6.0.0    pmix/4.2.9    prun/2.2    ucx/1.17.0

---------------------------------------------------------------------------- /mnt/shared/easybuild/modules/all ----------------------------------------------------------
   ANSYS/2023R1                                 OpenMPI/4.0.6-GCC-10.3.0                     groff/1.22.4-GCCcore-11.2.0
   ATK/2.36.0-GCCcore-11.2.0                    OpenMPI/4.0.7-GCC-10.3.0                     groff/1.22.4-GCCcore-11.3.0
   ATK/2.38.0-GCCcore-11.3.0                    OpenMPI/4.1.1-GCC-10.3.0                     groff/1.22.4-GCCcore-12.2.0
   ATK/2.38.0-GCCcore-12.2.0                    OpenMPI/4.1.1-GCC-11.2.0                     groff/1.22.4-GCCcore-12.3.0
   ATK/2.38.0-GCCcore-12.3.0                    OpenMPI/4.1.4-GCC-11.3.0                     groff/1.23.0-GCCcore-13.2.0                (D)
   ATK/2.38.0-GCCcore-13.2.0             (D)    OpenMPI/4.1.4-GCC-12.2.0                     grpcio/1.57.0-GCCcore-12.3.0
   Abseil/20230125.2-GCCcore-12.2.0             OpenMPI/4.1.5-GCC-12.3.0                     gzip/1.10-GCCcore-10.2.0
   Abseil/20230125.3-GCCcore-12.3.0      (D)    OpenMPI/4.1.6-GCC-13.2.0                     gzip/1.10-GCCcore-10.3.0
   Anaconda3/2022.10                            OpenMPI/5.0.3-GCC-13.3.0              (D)    gzip/1.10-GCCcore-11.2.0
   Armadillo/11.4.3-foss-2022b                  OpenMolcas/22.10-foss-2021b                  gzip/1.12-GCCcore-11.3.0
   Armadillo/12.6.2-foss-2023a                  OpenPGM/5.2.122-GCCcore-12.3.0               gzip/1.12-GCCcore-12.2.0
   Armadillo/12.8.0-foss-2023b           (D)    OpenSSL/1.1                           (L)    gzip/1.12-GCCcore-12.3.0
   Autoconf/2.69-GCCcore-9.3.0                  OpenSSL/1.1.1e-GCCcore-9.3.0                 gzip/1.13-GCCcore-13.2.0                   (L)
   Autoconf/2.69-GCCcore-10.2.0                 OpenSSL/1.1.1h-GCCcore-10.2.0                gzip/1.13-GCCcore-13.3.0                   (D)
   Autoconf/2.71-GCCcore-10.3.0                 OpenSSL/1.1.1k-GCCcore-10.3.0                h5py/3.2.1-foss-2021a
   Autoconf/2.71-GCCcore-11.2.0                 OpenSSL/1.1.1n-GCCcore-11.3.0                h5py/3.9.0-foss-2023a                      (D)
   Autoconf/2.71-GCCcore-11.3.0                 OpenSSL/1.1.1q-GCCcore-12.2.0                hatchling/1.18.0-GCCcore-12.3.0
   Autoconf/2.71-GCCcore-12.2.0                 OpenSSL/3                             (D)    hatchling/1.18.0-GCCcore-13.2.0            (D)
   Autoconf/2.71-GCCcore-12.3.0                 PBZIP2/1.1.13-GCCcore-12.3.0                 help2man/1.47.12-GCCcore-9.3.0
   Autoconf/2.71-GCCcore-13.2.0                 PCRE/8.44-GCCcore-10.3.0                     help2man/1.47.16-GCCcore-10.2.0
   Autoconf/2.71                                PCRE/8.45-GCCcore-11.2.0                     help2man/1.48.3-GCCcore-10.3.0
   Autoconf/2.72-GCCcore-13.3.0          (D)    PCRE/8.45-GCCcore-11.3.0                     help2man/1.48.3-GCCcore-11.2.0
   Automake/1.16.1-GCCcore-9.3.0                PCRE/8.45-GCCcore-12.2.0                     help2man/1.49.2-GCCcore-11.3.0
   Automake/1.16.2-GCCcore-10.2.0               PCRE/8.45-GCCcore-12.3.0                     help2man/1.49.2-GCCcore-12.2.0
   Automake/1.16.3-GCCcore-10.3.0               PCRE/8.45-GCCcore-13.2.0              (D)    help2man/1.49.3-GCCcore-12.3.0
```

- Modules resolve dependencies automatically, **BUT** loading a new module with conflicts (e.g. dependency with different version) replaces already loaded(!)

```
[js2347@artemis-login-0 ~]$ module load intel
[js2347@artemis-login-0 ~]$ module load GCC

The following have been reloaded with a version change:
  1) GCCcore/11.3.0 => GCCcore/13.3.0     2) binutils/2.38-GCCcore-11.3.0 => binutils/2.42-GCCcore-13.3.0     3) zlib/1.2.12-GCCcore-11.3.0 => zlib/1.3.1-GCCcore-13.3.0
```

Containers: Singularity/Apptainer

- Containers are self-contained, standard units of software
- contain code and all its dependencies, allowing applications to run consistently and reliably across different computing environments

Containers: Singularity/Apptainer

- Containers are self-contained, standard units of software
- contain code and all its dependencies, allowing applications to run consistently and reliably across different computing environments
- images can be easily built/customized via scripts from base images
- visualisation "light" e.g. unlike VMs restricted to host's architecture

Containers: Singularity/Apptainer

- Containers are self-contained, standard units of software
- contain code and all its dependencies, allowing applications to run consistently and reliably across different computing environments
- images can be easily built/customized via scripts from base images
- visualisation "light" e.g. unlike VMs restricted to host's architecture
- popular software:

  Docker (requires root privileges)
  Singularity/Apptainer (supported on Artemis)

DISCnet

Etiquette

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system
- Do not clog up the login nodes

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system
- Do not clog up the login nodes
- When submitting jobs, try to request only the resources you really need:

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system
- Do not clog up the login nodes
- When submitting jobs, try to request only the resources you really need:
  - give good estimate of walltime (for efficient scheduling)

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system
- Do not clog up the login nodes
- When submitting jobs, try to request only the resources you really need:
  - give good estimate of walltime (for efficient scheduling)
  - try not to waste resources i.e. especially at busy times, only request the cores you really need, free up licenses again

Etiquette: Being a good user

When using an HPC system (e.g. Artemis) make sure to follow these simple rules:

- Know the basics from this course
- pay attention to the "house rules" & guidelines of the visiting system
- Do not clog up the login nodes
- When submitting jobs, try to request only the resources you really need:
  - give good estimate of walltime (for efficient scheduling)
  - try not to waste resources i.e. especially at busy times, only request the cores you really need, free up licenses again
- Again, do **NOT** clog up the login nodes

Recap: HPC as a service