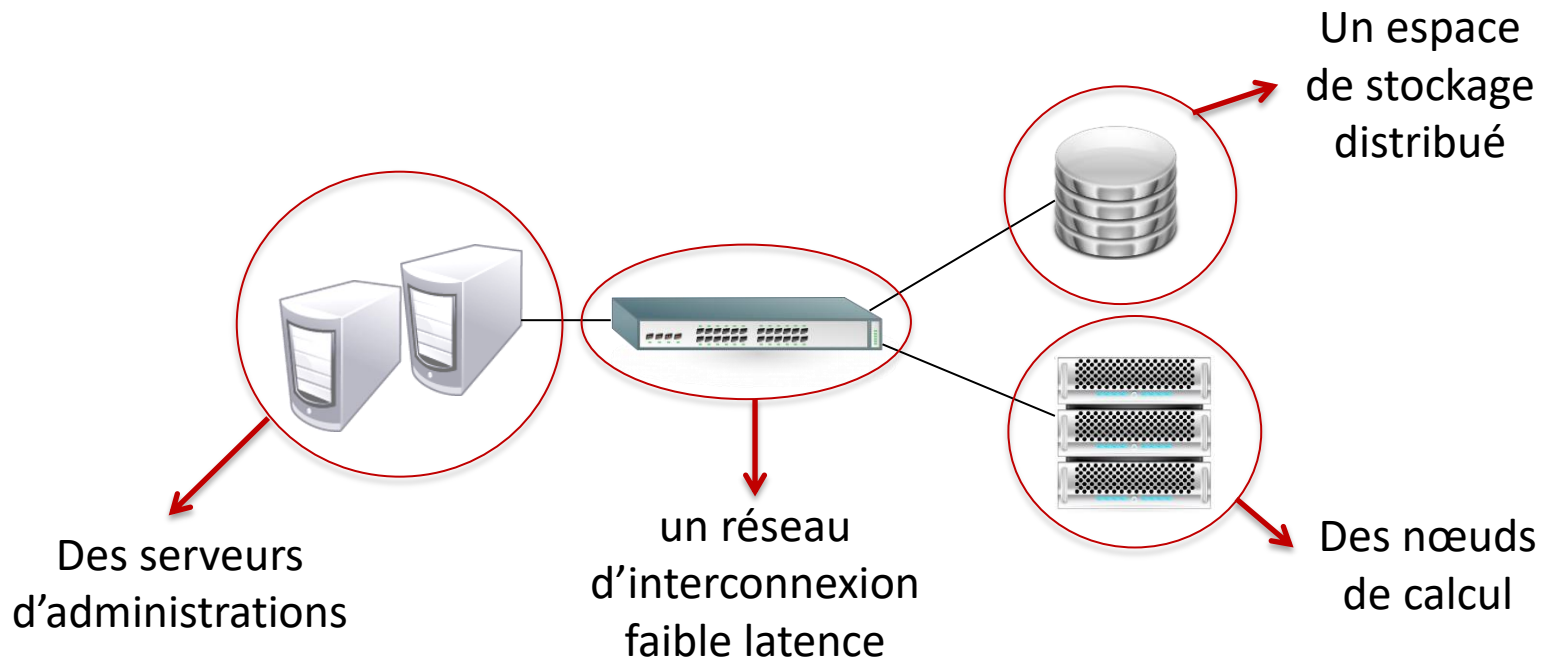


PRÉSENTATION DU CLUSTER MUSE

CYCLE DE FORMATION MESO@LR
7 NOVEMBRE 2019

Baptiste CHAPUISAT

QU'EST CE QU'UN CLUSTER HPC



Une machine unique pour l'utilisateur

LE CLUSTER MUSE

- 308 nœuds de calcul Dell PowerEdge C6320
 - bi processeurs Intel Xeon E5-2680 v4 2,4 Ghz (*broadwell*)
 - 8624 cœurs
 - 128 Go RAM par nœuds
 - 280 Tflops Linpack
- 1 Po de stockage rapide
- 350 To de stockage pérenne
- Réseau d'interconnexion Intel OmniPath 100 Gb/s
- Pas d'accélérateur
- 2 nœuds épais : 80 cœurs, 1To de RAM

HÉBERGEMENT CINES



<https://my.matterport.com/show/?m=zaXMRypj7Hw>



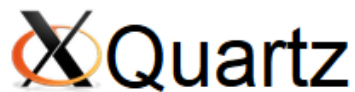
CONNEXION AU SERVEUR FRONTAL

- Depuis Linux

- [chapis@local ~]\$ ssh -X chapis@muse-login.hpc-lr.univ-montp2.fr

- Depuis MacOS X

Pas d'interface X11 par défaut

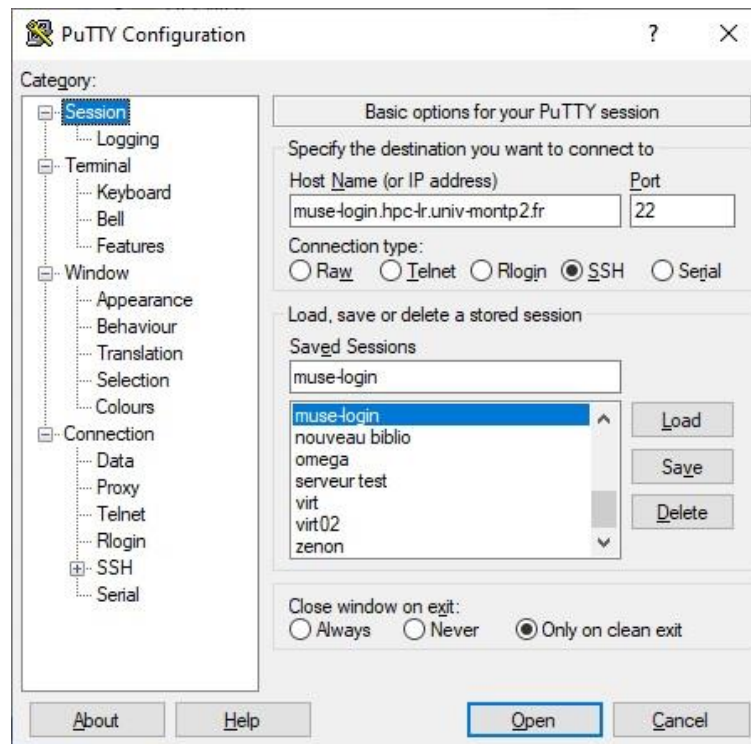


<http://xquartz.macosforge.org/landing/>

CONNEXION SSH DEPUIS WINDOWS

PuTTY

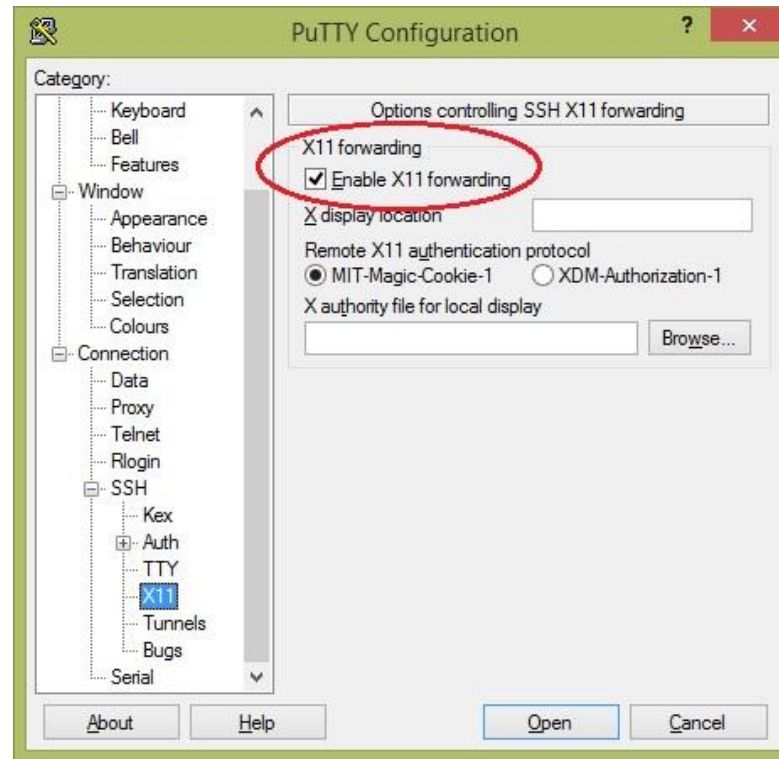
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>



CONNEXION SSH DEPUIS WINDOWS




<http://www.straightrunning.com/XmingNotes/>



ENVIRONNEMENT UTILISATEUR

- Deux serveurs en frontal (muse-login[01-02])
 - Linux CentOS 7.2
- Espace de stockage pérenne avec quota de groupe
 - /home/\$USER (espace utilisateur)
 - /work/\$GROUP (espace groupe)
- Espace de calcul sans quota (durée de validité)
 - /scratch/\$USER (espace utilisateur)



Attention au
groupe multiple !

ENVIRONMENT MODULE

- Environment module permet de configurer l'environnement utilisateur.
- Adapté aux environnements multi-utilisateur.
- Modifie les variables d'environnement.
- Permet de faire cohabiter plusieurs versions d'un même logiciel.

ENVIRONMENT MODULES

- Format de la commande:
 - module [paramètre] <nom_du_module>
 - avail : liste des modules disponibles
 - show : affiche les informations d'un module
 - add / load : charge un module
 - rm / unload : décharge un module
 - switch / swap : remplace un module
 - list : liste des modules chargés
 - purge : décharge tous les modules
 - help
- <https://modules.readthedocs.io/en/latest/>

ENVIRONMENT MODULES

Structure arborescente des modules sur muse.

```
chapuis@muse-login01:~$ module av
----- /usr/share/Modules/modulefiles -----
dot          module-git  module-info modules      null          use.own
----- /trinity/shared/modulefiles/modulegroups -----
cv-admin      cv-advanced cv-local    cv-standard
----- /trinity/shared/modulefiles/cv-standard -----
R/3.3.1                gcc/6.1.0                hdf5/openmpi/icc16/1.8.17  intel/mpi/64/2016.3.210  mvapich2/psm2/icc16/2.2rc2
blas/3.6.0             gdb/7.11                hwloc/1.11.2              intel/runtime/32/2016.3.210  numpy/py27/1.11.2
boost/1.61.0(default)  git/2.9.3              intel/advisor/32/2016.3.210  intel/runtime/64/2016.3.210  openblas/0.2.18(default)
boost/icc16/1.61.0     hdf5/1.8.17            intel/advisor/64/2016.3.210  intel/tbb/32/2016.3.210     openmpi/2.0.1
boost/impi/icc16/1.61.0 hdf5/gcc49/1.8.17(default) intel/clock/64/2016.3.210   intel/tbb/64/2016.3.210     openmpi/icc16/2.0.1
boost/mvapich2/1.61.0  hdf5/gcc53/1.8.17      intel/clock/mic/2016.3.210  intel/vtune/32/2016.3.210   openmpi/psm2/2.0.1
boost/mvapich2/icc16/1.61.0 hdf5/gcc61/1.8.17    intel/compiler/32/2016.3.210  intel/vtune/64/2016.3.210  openmpi/psm2/gcc49/2.0.1(default)
boost/openmpi/1.61.0   hdf5/icc16/1.8.17     intel/compiler/64/2016.3.210  lapack/3.6.1               openmpi/psm2/gcc53/2.0.1
boost/openmpi/icc16/1.61.0 hdf5/impi/icc16/1.8.17 intel/daal/32/2016.3.210    matplotlib/py27/1.5.3      openmpi/psm2/gcc61/2.0.1
cmake/3.6.0(default)  hdf5/mvapich2/1.8.17  intel/daal/64/2016.3.210    mellanox/fca/2.5           openmpi/psm2/icc16/2.0.1
fftw2/2.1.5(default)  hdf5/mvapich2/gcc49/1.8.17 intel/inspector/32/2016.3.210  mellanox/hcoll/3.5        python/2.7.12(default)
fftw3/3.3.5(default)  hdf5/mvapich2/gcc53/1.8.17 intel/inspector/64/2016.3.210  mellanox/mxm/3.4          qt/gcc/4.8.6
fftw3/mvapich2/3.3.5  hdf5/mvapich2/gcc61/1.8.17 intel/ipp/32/2016.3.210     mvapich2/2.2rc2            scalapack/mvapich2/2.0.2
fftw3/openmpi/3.3.5   hdf5/mvapich2/icc16/1.8.17 intel/ipp/64/2016.3.210     mvapich2/icc16/2.2rc2      scalapack/openmpi/2.0.2
fftw3-mvapich2/mvapich2/3.3.5 hdf5/openmpi/1.8.17  intel/itac/64/2016.3.210    mvapich2/psm2/2.2rc2      scilab/5.5.2
fftw3-openmpi/openmpi/3.3.5 hdf5/openmpi/gcc49/1.8.17 intel/mkl/32/2016.3.210    mvapich2/psm2/gcc49/2.2rc2(default)  scipy/py27/0.18.1
gcc/4.9.3(default)    hdf5/openmpi/gcc53/1.8.17 intel/mkl/64/2016.3.210    mvapich2/psm2/gcc53/2.2rc2  valgrind/3.11.0
gcc/5.3.0             hdf5/openmpi/gcc61/1.8.17 intel/mkl/mic/2016.3.210    mvapich2/psm2/gcc61/2.2rc2
[chapuis@muse-login01 ~]$
```

```
[chapuis@muse-login ~]$ echo $MODULEPATH
/usr/share/Modules/modulefiles:/etc/modulefiles:/trinity/shared/modulefiles/
modulegroups:/trinity/shared/modulefiles/
```

ENVIRONMENT MODULES

Structure arborescente

➤ cv-standard

- La majorité des compilateurs et bibliothèques standards

➤ cv-advanced

- Logiciels plus spécifiques à certaines spécialités

➤ local

- Logiciel installé par muse@lr

➤ modulefiles

- Modules spéciaux

LES MODULES INTEL

- intel/advisor : Intel Advisor - <https://software.intel.com/en-us/intel-advisor-xe>
Vectorization Optimization and Thread Prototyping
- intel/cilk : Intel Cilk Plus - <https://www.cilkplus.org/>
Intel Cilk Plus is an extension to the C and C++ languages to support data and task parallelism.
- intel/compiler : Intel C/C++/Fortran compilers - <https://software.intel.com/en-us/intel-compilers>
- intel/daal : Intel Data Analytics Acceleration Library - <https://software.intel.com/en-us/articles/opendaal>
Intel DAAL helps accelerate big data analytics by providing highly optimized algorithmic building blocks for all data analysis stages [...].
- intel/inspector : Intel Inspector - <https://software.intel.com/en-us/intel-inspector-xe>
Memory and Thread Debugger
- intel/ipp : Intel Integrated Performance Primitives - <https://software.intel.com/en-us/intel-ipp>
Intel® IPP offers developers high-quality, production-ready, low-level building blocks for image processing, signal processing, and data processing (data compression/decompression and cryptography) applications.
- intel/itac : Intel Trace Analyzer and Collector - <https://software.intel.com/en-us/intel-trace-analyzer>
Understand MPI application behavior, quickly find bottlenecks, and achieve high performance for parallel cluster applications
- intel/mkl : Intel Math Kernel Library - <https://software.intel.com/en-us/intel-mkl>
Fastest and most used math library for Intel and compatible processors.
- intel/mpi : Intel MPI library - <https://software.intel.com/en-us/intel-mpi-library>
Making applications perform better on Intel® architecture-based clusters with multiple fabric flexibility
- intel/runtime : Intel runtime libs
- intel/tbb : Intel Threading Building Blocks - <https://software.intel.com/en-us/intel-tbb>
Intel® Threading Building Blocks (Intel® TBB) is a widely used C++ library for shared-memory parallel programming and heterogeneous computing (intra-node distributed memory programming). The library provides a wide range of features for parallel programming, including generic parallel algorithms, concurrent containers, a scalable memory allocator, work-stealing task scheduler, and low-level synchronization primitives. Intel TBB is a library only solution for task-based parallelism and does not require any special compiler support. It ports to multiple architectures including Intel® architectures, ARM, and POWER*.*
- intel/vtune : Intel VTune Amplifier – <https://software.intel.com/en-us/intel-vtune-amplifier-xe>
Performance Profiler

LES MODULEFILES

```
#%Module
#
# @name:  monprog
# @version: 1.0
# @packaging: Baptiste Chapuisat
#

# Definie les variables internes au module
#-----
set name    monprog
set version 1.0
set prefix  $::env(HOME)/F_HPC@LR

# S'affiche avec l'option help
#-----
proc ModulesHelp { } {
    puts stderr "\tExemple de documentation"
    puts stderr "\tpour le module [module-info name]"
    puts stderr ""
}
```

```
# Test le repertoire
# -----
if {[file exists $prefix]} {
    puts stderr "\tLoad Error: $prefix does not exist"
    break
    exit 1
}

# Dependance
#
# prereq python/3.7.2
module load python/3.7.2

# Update common variables in the environment
# -----
prepend-path PATH          $prefix/bin
prepend-path LD_LIBRARY_PATH $prefix/lib

setenv      MP_HOME      $version
```

INSTALLATION DE LOGICIEL

- Compilation depuis les sources
- Les systèmes "packagés" : Nix, Miniconda
 - inconvénients : espace disque important
- Système de conteneur Singularity

LE GESTIONNAIRE SLURM

Simple Linux Utility for Resource Management

- gestionnaire de cluster sous Linux
- Les +
 - Open source
 - Très répandu
 - Tout en un
- Configuration des grappes de nœuds (droit d'accès, quotas, ...)
- Accès aux ressources par les utilisateurs
- Ordonnancement de tâches dans les files d'attente (arbitrage)
- Accounting



LE GESTIONNAIRE SLURM

SLURM	LoadLeveler	PBS / Torque
sbatch	lsubmit	qsub
squeue	llq	qstat
scancel	llcancel	qdel

VOCABULAIRE

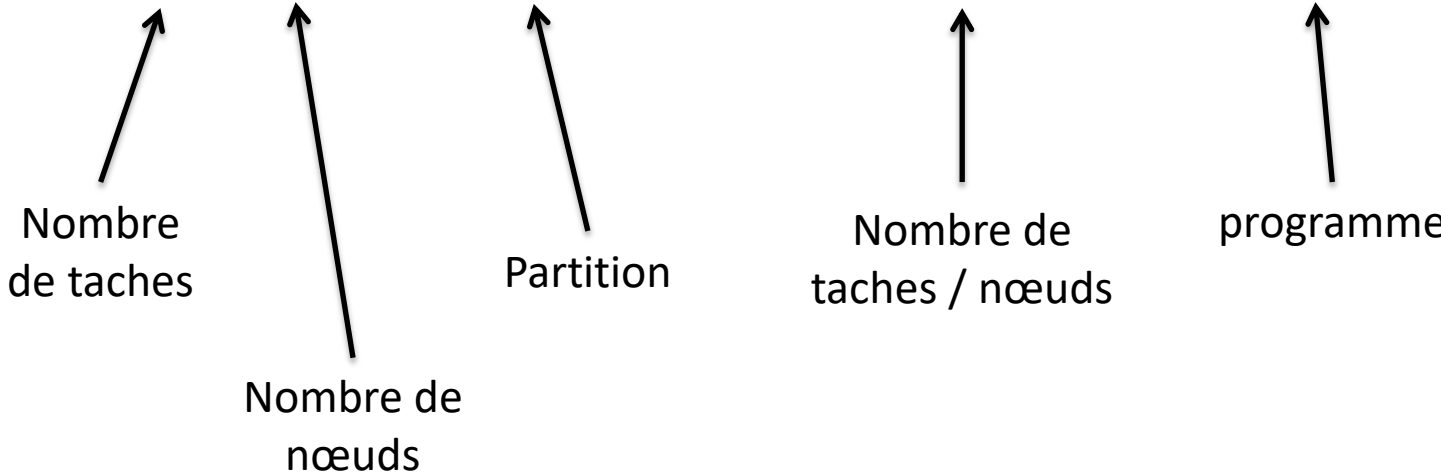
- Des utilisateurs appartenant à des accounts exécutent des jobs sur des partitions
- account = groupe (commande id)
- partition = un ensemble de nœuds
- Chaque nœud est composé de 28 cœurs
- Un cœur est une unité de traitement physique
- Une tâche (ou processus, ou thread) est une unité de traitement logique

LA COMMANDE SRUN

La commande srun permet d'exécuter plusieurs jobs en parallèles sur les nœuds de calcul

```
$ srun -n4 -N2 -p fmuse1 --ntasks-per-node=2 hostname
```

Nombre
de taches



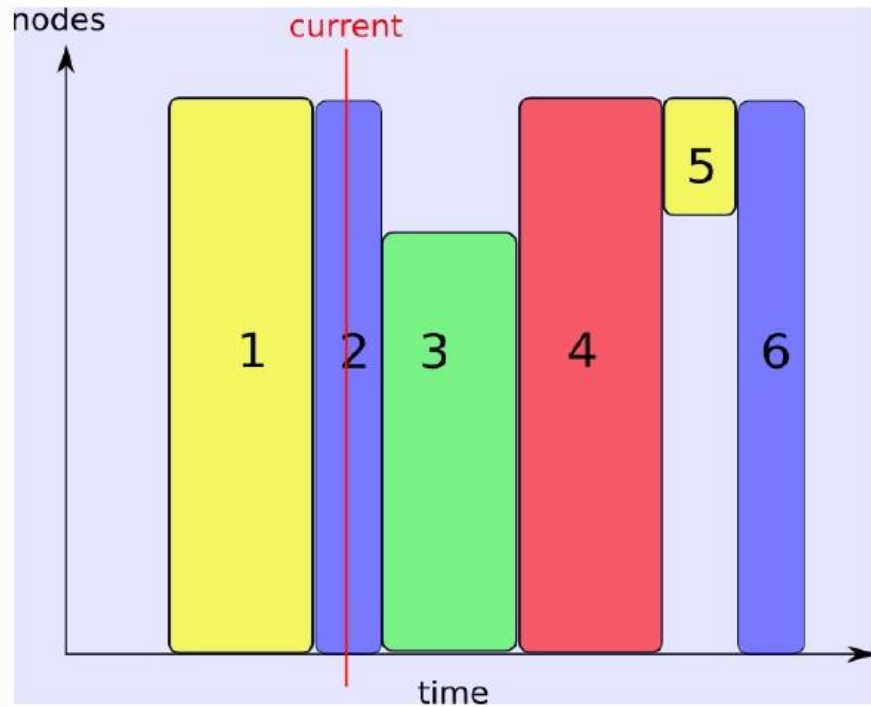
Nombre de
nœuds

Partition

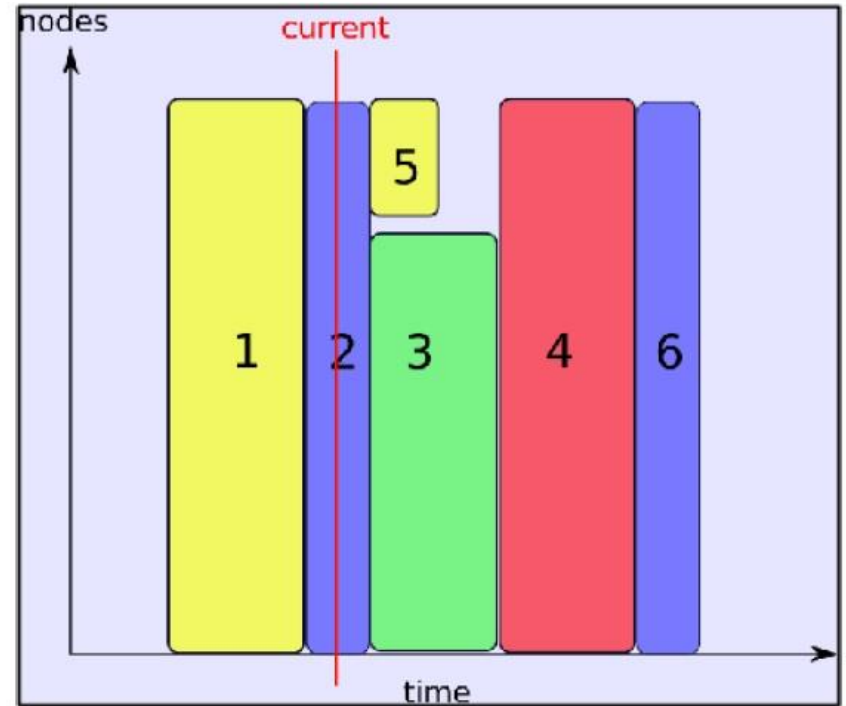
Nombre de
taches / nœuds

programme

SCHEDULER MODE



FIFO Scheduler



Backfill Scheduler

SCHEDULER MODE

FIFO Scheduler

Node

0	C1	C2	C2	C2	C2	C2	C3	C4	C4	C4	C5			
1	C1						C3	C4	C4	C4	C5			
2	C1						C3				C5			
3	C1						C3				C6	C6	C6	C6
	1	2	3	4	5	6	7	8	9	10	11	12	13	14

BackFill Scheduler

Node

0	C1	C2	C2	C2	C2	C2	C3
1	C1	C4	C4	C4		C5	C3
2	C1	C4	C4	C4		C5	C3
3	C1	C6	C6	C6	C6	C5	C3
	1	2	3	4	5	6	7

SBATCH

- La commande sbatch permet de spécifier via un fichier de script les caractéristiques d'un job (ressources, programme à exécuter, ...).
- Bien que l'on retrouve des paramètres semblables pour les deux commandes, srun n'est pas la version en ligne de commande de sbatch.
- Les paramètres relatifs aux ressources sont :
 - des paramètres d'exécution pour srun.
 - des paramètres de réservation pour sbatch.

SBATCH

```
#!/bin/bash
#
#SBATCH -N 2
#SBATCH -n 4
#SBATCH --ntasks-per-node=2
#SBATCH --partition=fmuse1

srun hostname
```

← script de
soumission du job

```
$ sbatch fichier.sh
```

```
$ sbatch -n4 -N2 -p fmuse1 fichier.sh
```

PRINCIPALES OPTIONS SBATCH

- -A, --account=<account>
- -p, --partition=<partition_names>
- -J, --job-name=<jobname>
- -o, --output=<filename pattern>
- -e, --error=<filename pattern>
- --mail-type=<type>
- --mail-user=<user>
- -d, --dependency=<dependency_list>
- -t, --time=<time>
- -b, --begin=<time> # Format time/date :
- --deadline=<date> [YYYY-MM-DDT]HH:MM:SS

PRINCIPALES OPTIONS SBATCH

- -n, --ntasks=<number>
- -N, --nodes=<minnodes[-maxnodes]>
- --ntasks-per-node=<ntasks>
- --ntasks-per-core=<ntasks>
- --ntasks-per-socket=<ntasks>
- -c, --cpus-per-task=<ncpus>
- --cores-per-socket=<cores>
- --mem=<size[units]>
- --mem-per-cpu=<size[units]>
- --exclusive[=user|mcs]
- -w, --nodelist=<node name list>
- -x, --exclude=<node name list>

OPTIONS COMMUNES

- Les options ne sont pas utilisables avec toutes les commandes mais lorsqu'elles le sont, on retrouvera (presque) toujours la même syntaxe.

-A, --account=<account>
-p, --partition=<partition_names>
-u, --user=<user_list>
-w, --nodelist=<node name list>

SCANCEL, SQUEUE ET SINFO

- **scancel** permet de tuer un job

```
$ scancel <num_job>
```

```
$ scancel -u <uid>
```

```
$ scancel -t PENDING
```

- **squeue** affiche l'état des jobs

```
$ squeue -u <uid>
```

- Principaux états :

- RUNNING (R) : En cours d'exécution
 - PENDING (PD) : En attente de ressource
 - COMPLETED (CD) : Terminé
 - CANCELED (CA) : Tué
 - FAILED (F) : Echec

- **sinfo** affiche l'état des partitions

```
$ sinfo -p <nom_partition>
```

SALLOCC

- La commande `salloc` permet de réserver des ressources auxquelles on peut ensuite accéder en `ssh`.

```
$ ssh muse078
```

```
$ salloc -w muse078 -p fmuse1 -t 1:30:00
```

```
$ ssh muse078
```

- L'utilisation des ressources allouées ne se limitent pas à `ssh`. Elles peuvent être utilisées par l'utilisateur pour lancer des jobs.

LES VARIABLES D'ENVIRONNEMENT

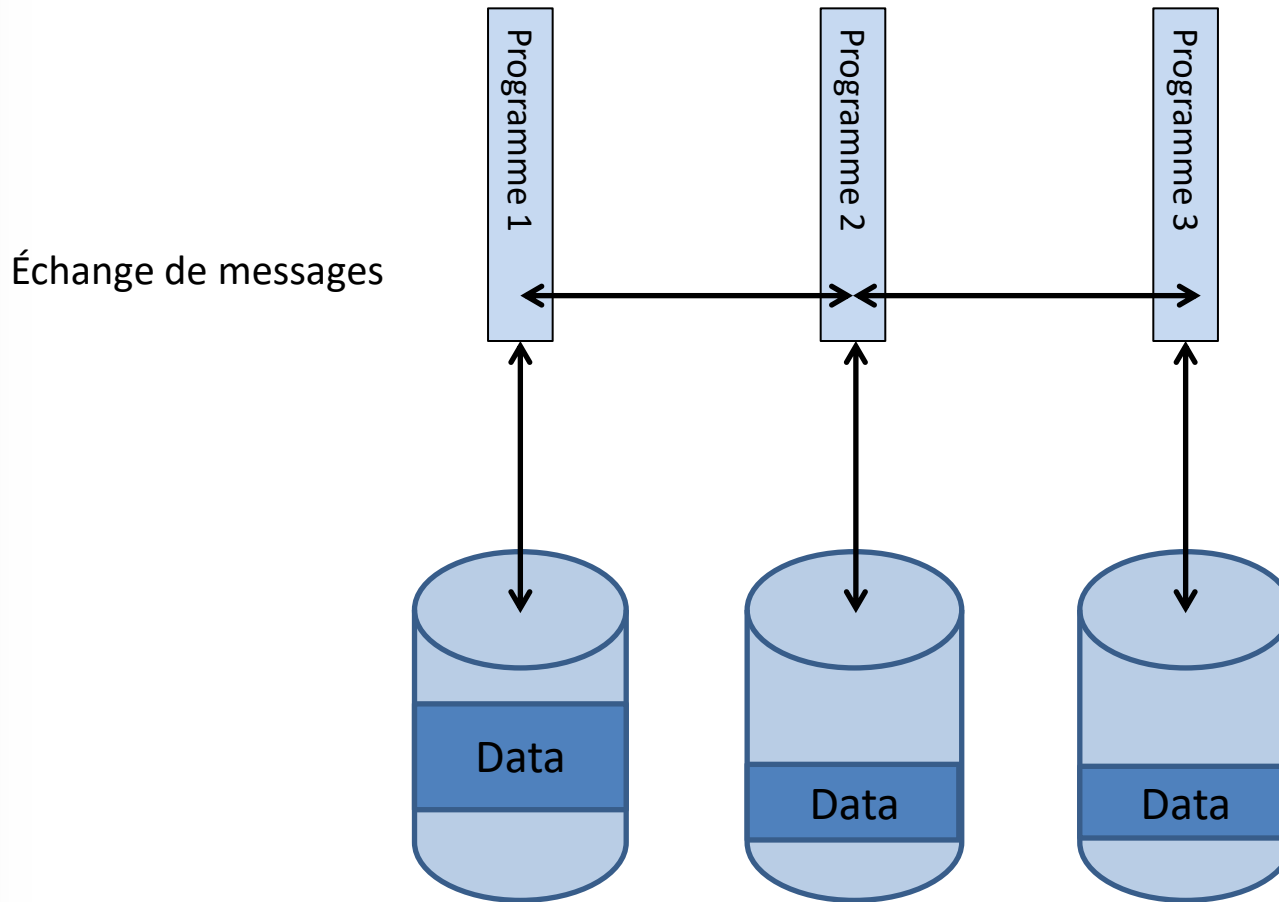
- SLURM_JOB_ID : Numéro du job
- SLURM_NODELIST : Nœuds alloués au job
- SLURM_JOB_NAME : Nom du job
- SLURM_SUBMIT_DIR : Répertoire courant
- SLURM_SUBMIT_HOST : Machine d'où est lancé le job
- SLURM_JOB_NUM_NODES : Nombre de nœuds
- SLURM_CPUS_PER_TASK : Nombre de CPU par tâche
- SLURM_NTASKS : Nombre de tâches
- SLURM_JOB_PARTITION : Partition utilisée

LES DIFFÉRENTS TYPES DE CALCUL

- Calcul en mémoire distribué (MPI)
- Calcul en mémoire partagée (Multi-thread, OpenMP)
- Calcul réparti (multi séquentiel)

- Historiquement C, C++, Fortran
- Aujourd'hui Python, R, Matlab

MODÈLE DE PROGRAMMATION EN MÉMOIRE DISTRIBUÉE



EXEMPLE DE FICHER SBATCH

```
#!/bin/sh
```

```
#SBATCH -N 2
```

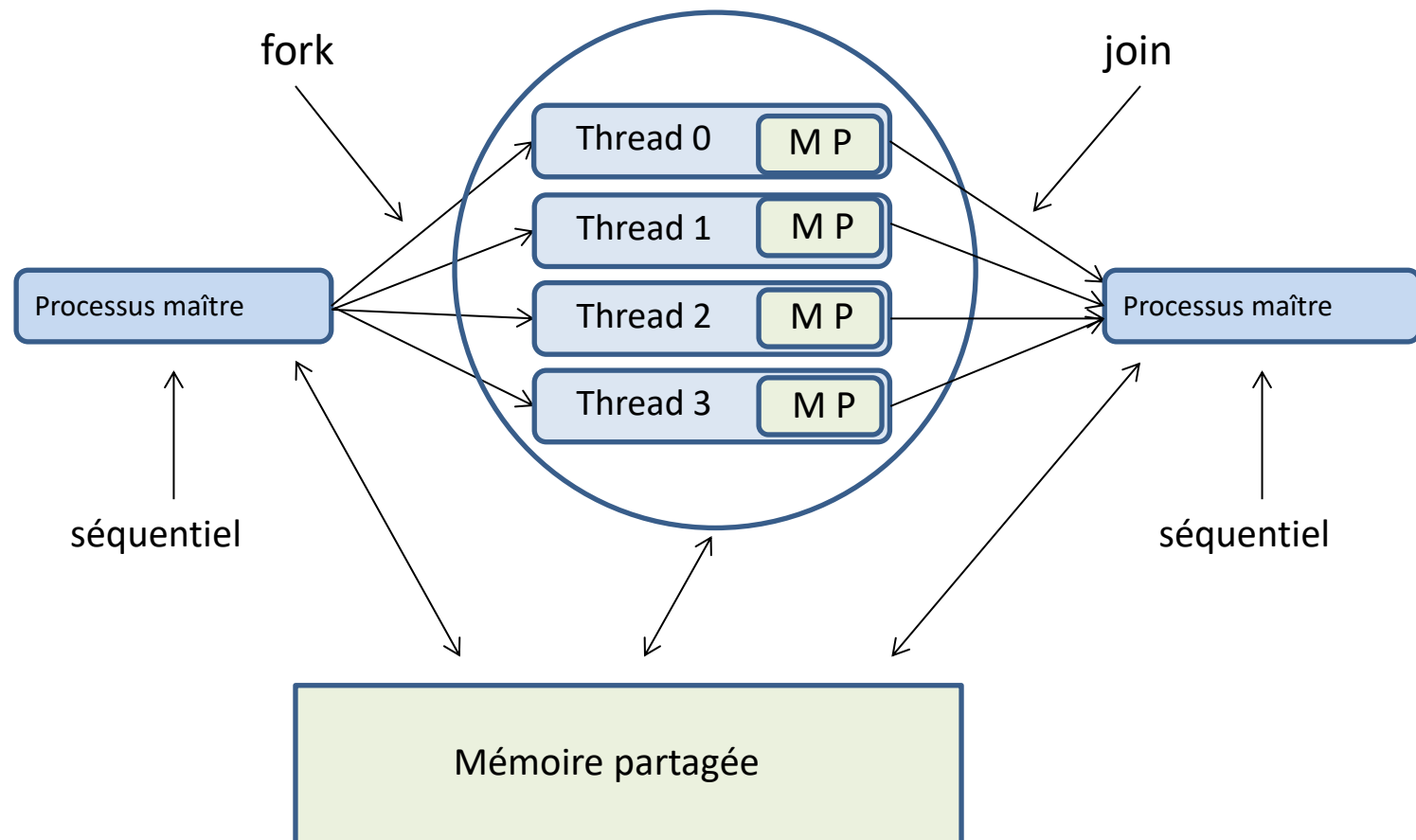
```
#SBATCH -n 56
```

```
#SBATCH --partition=fmuse1
```

```
module load cv-standard openmpi python
```

```
mpiexec -n 56 python prog.py
```


MODÈLE DE PROGRAMMATION MULTITHREAD



EXEMPLE DE FICHER SBATCH

```
$ gcc -fopenmp -o prog.exe prog.c
```

```
#!/bin/sh
#SBATCH -n 1
#SBATCH -c 4
#SBATCH --partition=fmuse1

export OMP_NUM_THREADS=4

./prog.exe
```

LE CALCUL RÉPARTI AVEC SLURM

- Exécuter un programme séquentiel plusieurs fois sur des jeux de données différents => Les processus sont indépendants.
- L'option job array : -a, --array
Exemple : --array=0-27
 --array=0-59%28
 --array=1,2,5-10
 --array=0-99:4

Variables d'environnements :

SLURM_JOB_ID, SLURM_ARRAY_JOB_ID, SLURM_ARRAY_TASK_ID

- Nombre de job limité à 5000 sur le cluster muse

LE CALCUL RÉPARTI AVEC SLURM

```
#!/bin/bash
#
#SBATCH --job-name=test_array
#SBATCH -N 2
#SBATCH --array=1-56
#SBATCH -o array-%a.out
#SBATCH --partition=fmuse1

module purge
module load cv-standard R
echo $SLURM_NODELIST
R CMD BATCH --no-save prog.R
```

LES COMMANDES D'ACCOUNTING

- Plusieurs commandes SLURM permettent d'afficher des informations sur les jobs et sur l'utilisation du cluster. Les deux principales sont sreport et sacct.
- Un utilisateur peut voir les informations sur les jobs des partitions et des groupes auquel il appartient.
- Il existe beaucoup d'option pour chacune des deux commandes.
- Le problème est dans l'interprétation des résultats.

LES COMMANDES D'ACCOUNTING

- Nombre d'heures par partition sur une période pour un utilisateur

```
$ sreport cluster UserUtilizationByAccount  
user=chapuis start=2019-10-01 end=2019-10-  
16T23:59:59 -t hours
```

- Liste des jobs sur une période pour un utilisateur

```
$ sacct --  
format="JobID,CPUTime,User,Account,JobName,Star  
t,End,Node" -u chapuis -X -S 2019-10-15 -E  
2019-10-16T23:59:59
```

LES COMMANDES D'ACCOUNTING

- Afficher les informations sur un job

```
$ sacct --  
format="JobID,CPUTime,AllocCPUS,User,Account,JobName  
,Start,End,State%20,Node,ExitCode" -j 1080739
```

- Afficher les informations sur un job actif

```
$ scontrol show jobid <job_id>
```