

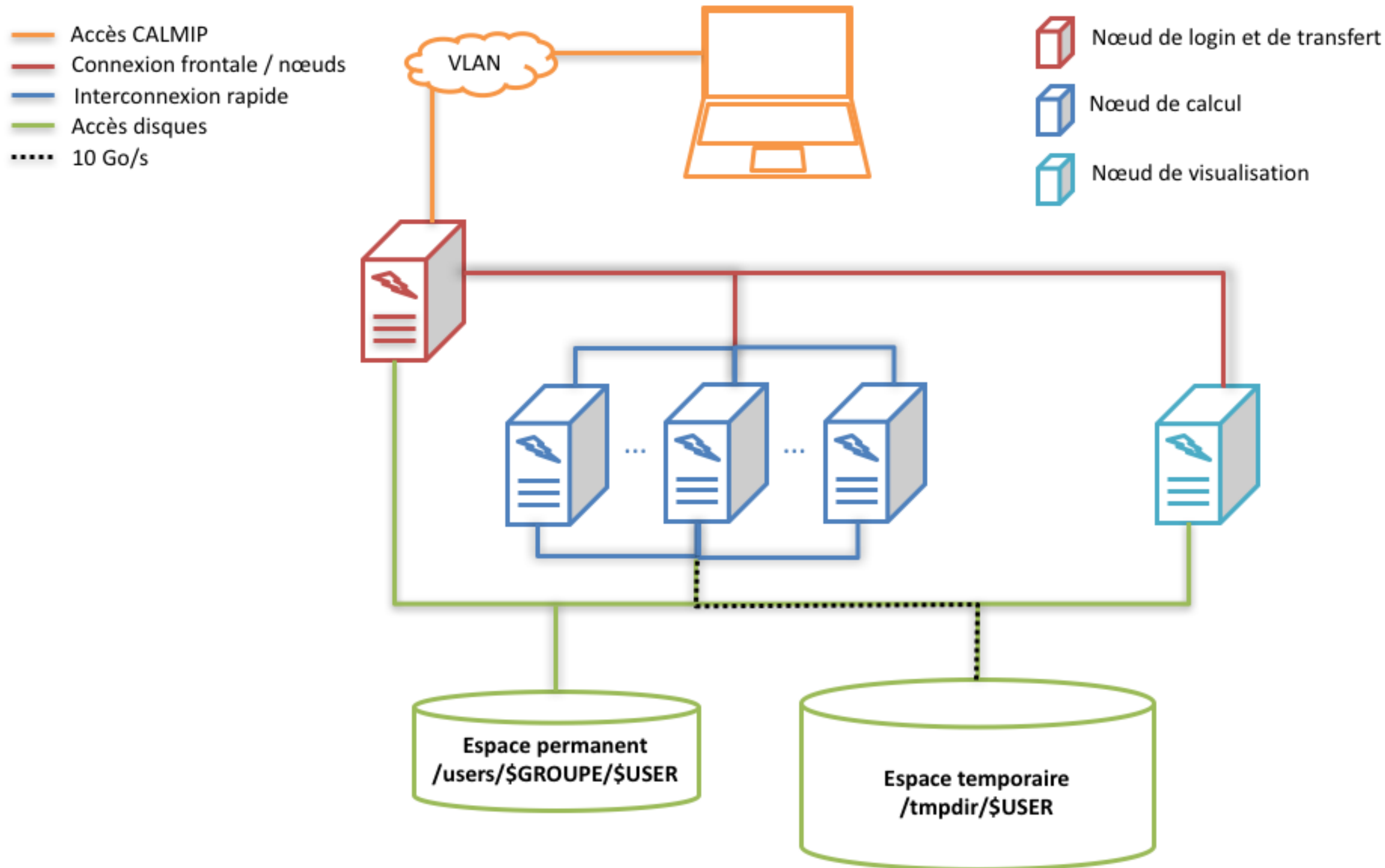
# PLATEFORME DE CALCUL INTENSIF : KIT TECHNIQUE



CALMIP (UMS 3667)  
Espace Clément Ader  
[www.calmip.univ-toulouse.fr](http://www.calmip.univ-toulouse.fr)



# PRÉSENTATION : SCHÉMA DU SYSTÈME DE CALCUL



Frontales de connexion :

- ▶ 4 x (20-cores, 128 GB RAM)

Cluster distribué BULLx DLC :

- ▶ 12240 cores - 612 nodes
- ▶ Intel® Ivybridge 2,8 Ghz 10-cores
- ▶ 64 GB RAM / nœud
- ▶ Interconnexion : Infiniband FDR

Nœud large mémoire :

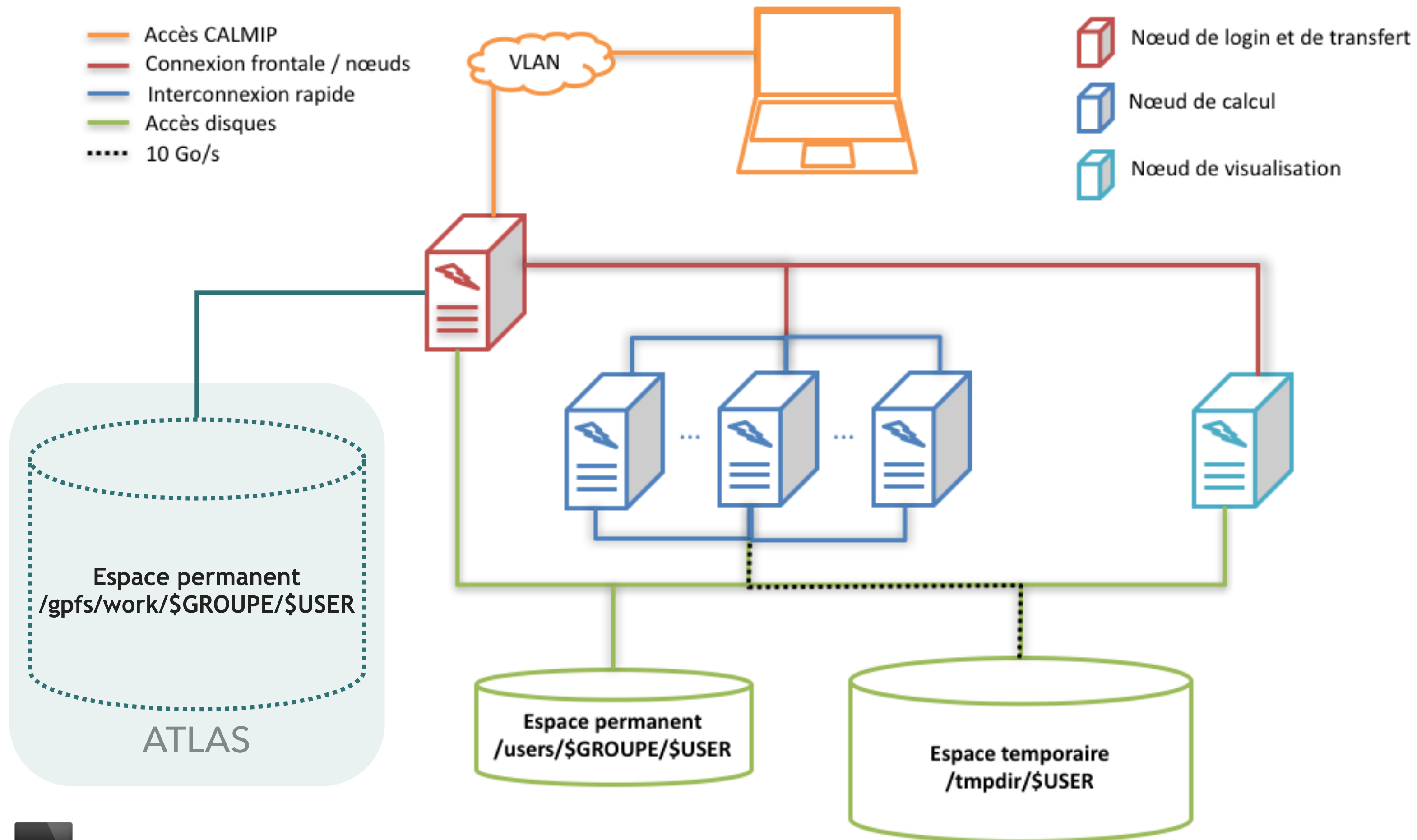
- ▶ 128 cores - 2 TB RAM
- ▶ Intel® Haswell-EX 2,2 Ghz 16-cores

Solution de visualisation à distance :

- ▶ 2 nœuds (20-cores, 128 GB RAM)
- ▶ Cartes Nvidia Quadro 6000
- ▶ TurboVNC / VirtualGL



# PRÉSENTATION : ESPACES FICHIERS



- ▶ Espace permanent (NFS) :
  - 5 Go par utilisateur
  - sauvegardes quotidiennes

- ▶ Espace temporaire (Lustre) :
  - 780 To partagés par tous les utilisateurs
  - effacement des fichiers non accédés après 100 jours

- ▶ Stockage sécurisé ATLAS (GPFS) :
  - 3 Po partagés par tous les utilisateurs
  - dédié au stockage de données massives (sur demande)



# PRISE EN MAIN D'EOS : LA FRONTALE DE CONNEXION

▶ Connexion « Secure Shell » (ssh)

- À partir d'un poste Linux / macOS

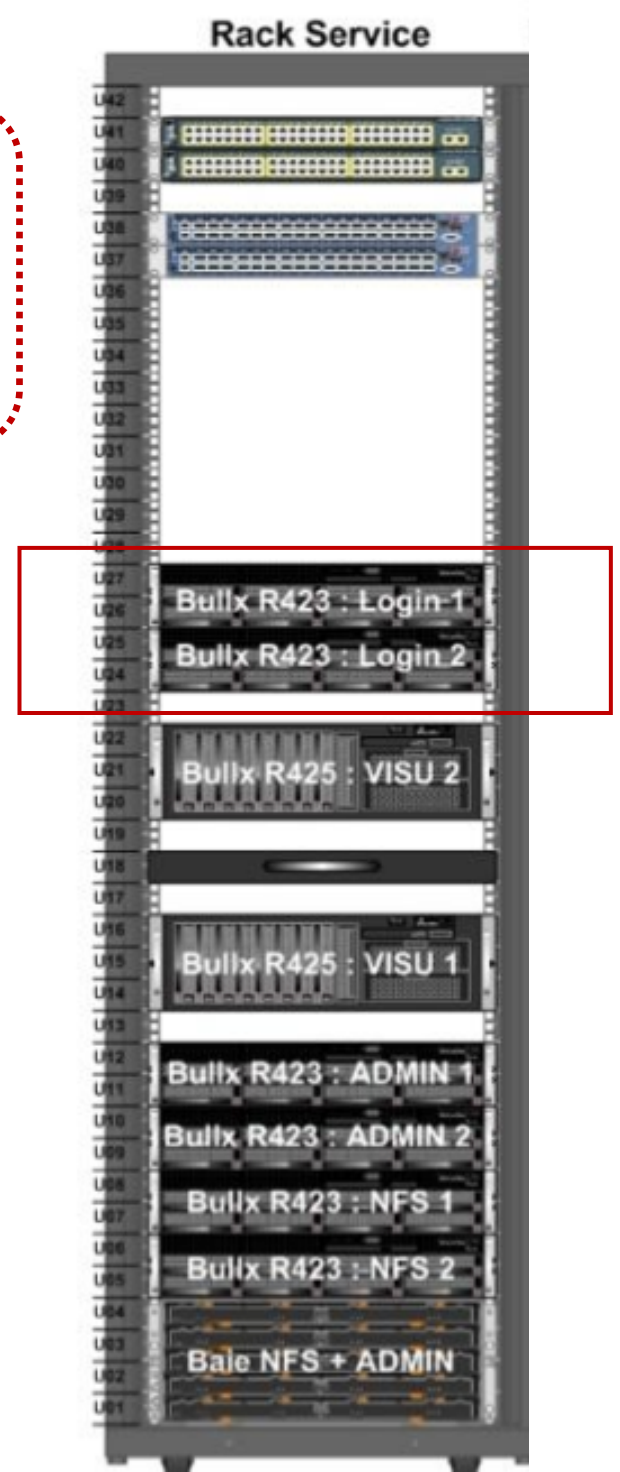
`ssh -X {login}@eos.calmip.univ-toulouse.fr`

- À partir d'un poste Windows

*Client ssh avec serveur X (Putty/Xming, MobaXterm)*

Frontales de connexion :

- ▶ 4 x (20-cores, 128 GB RAM)

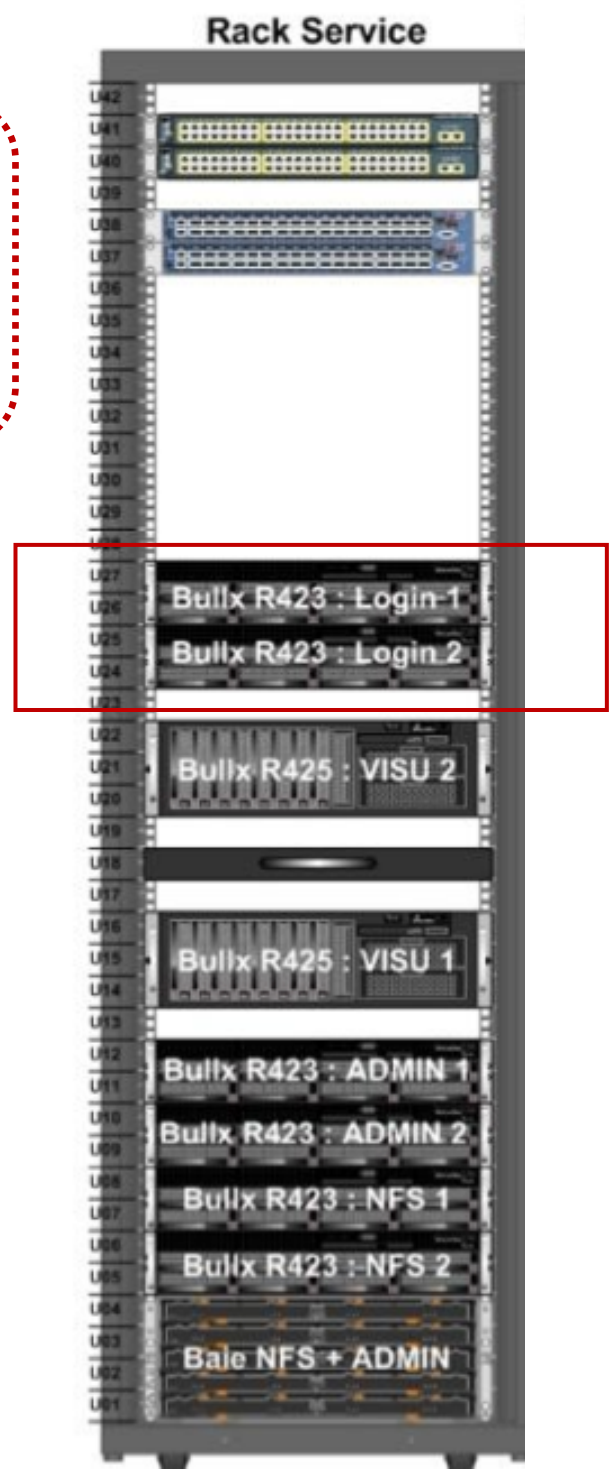


# PRISE EN MAIN D'EOS : LA FRONTALE DE CONNEXION

- ▶ Ce que l'on peut faire sur une frontale
  - compiler, installer un code
  - transférer des fichiers
  - effectuer des (petits) test, debugger
  
- ▶ Ce que l'on ne peut PAS faire sur une frontale
  - faire des calculs
  - mode batch obligatoire

Frontales de connexion :

- ▶ 4 x (20-cores, 128 GB RAM)



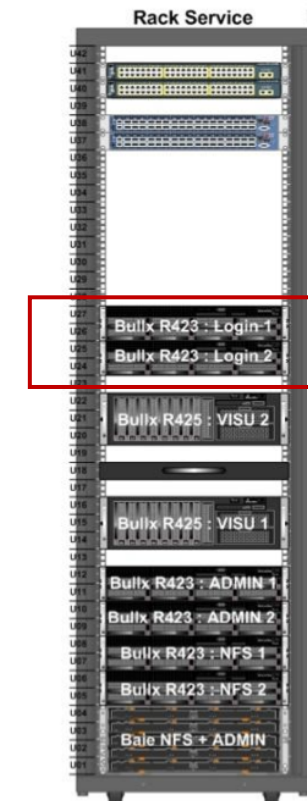
# LANCEMENT DES CALCULS : PRINCIPES

- ▶ Connexion sur la frontale

```
ssh -X {login}@eos.calmip.univ-toulouse.fr
```

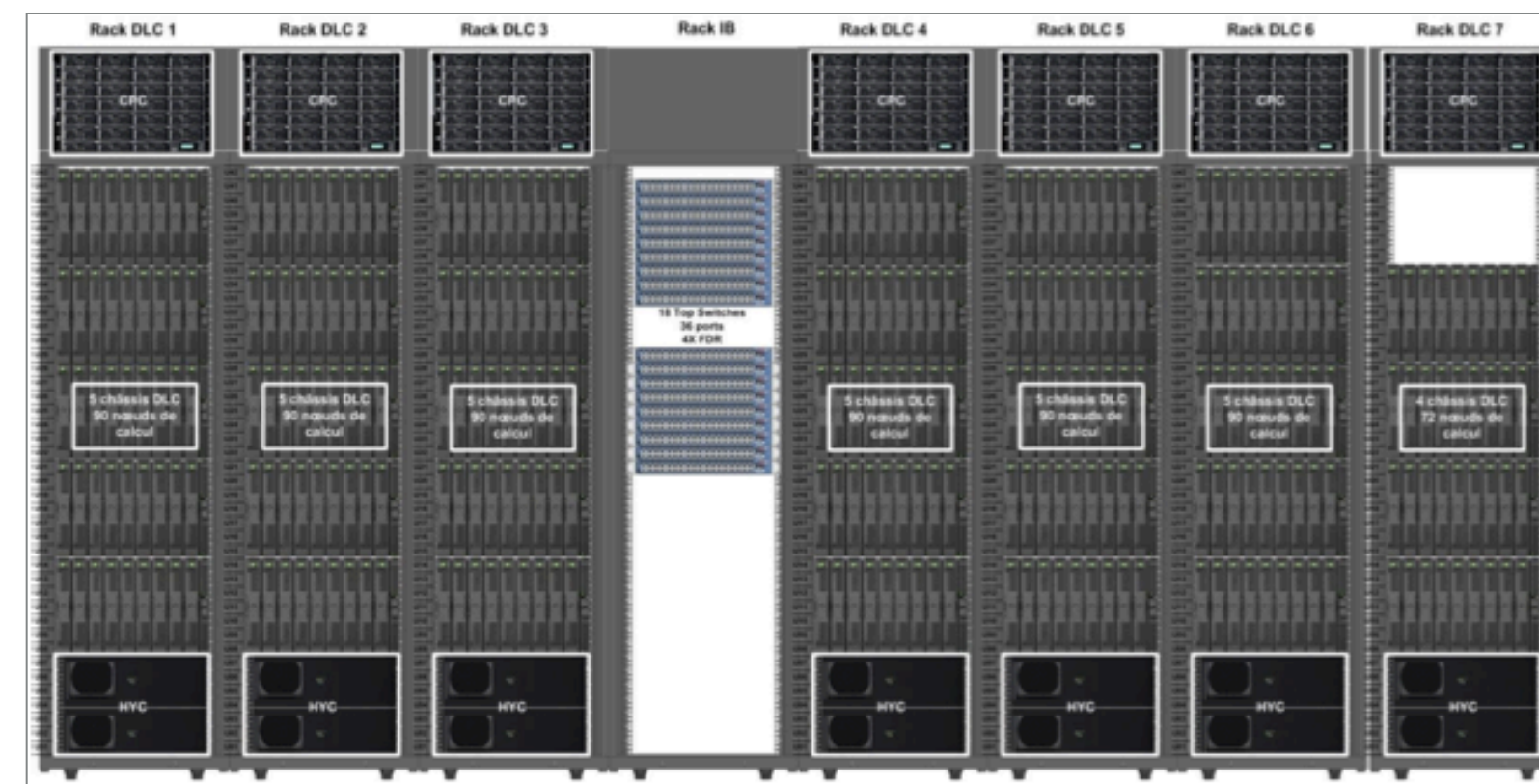
- ▶ Lancement des calculs en différé avec le gestionnaire de batch (SLURM)

```
sbatch mon_job
```



Frontales de connexion :

- ▶ 4 x (20-cores, 128 GB RAM)



- ▶ Distributed Memory Cluster BULLx DLC (12240 cores - 612 nodes)
- ▶ Processeurs Intel® Ivybridge 2,8 Ghz 10-cores
- ▶ 64 GB RAM / nœud
- ▶ Interconnection : Infiniband FDR
- ▶ Topologie : full fat-tree

# LANCEMENT DES CALCULS : COMMANDES SLURM

- ▶ Lancer un calcul

```
sbatch mon_job
```

- ▶ Arrêter un calcul

```
scancel $SLURM_JOBID
```

- ▶ Afficher les informations sur le calcul

```
scontrol show jobid=$SLURM_JOBID
```

- ▶ Afficher la liste des calculs

```
squeue -u $USER
```



# LANCEMENT DES CALCULS : SCRIPT BATCH

```
#!/bin/bash
#SBATCH --job-name=script_UtilisationEOS
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=20
#SBATCH --ntasks-per-core=1
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com
```

```
dirname=${SLURM_JOBID}
mkdir /tmpdir/$USER/$dirname
cp mes_inputs /tmpdir/$USER/$dirname
cd /tmpdir/$USER/$dirname
```

```
module purge
module load module1 module2
module list
```

```
./mon_appli.exe > output_${SLURM_JOBID}.log
```

```
mv mes_outputs $SLURM_SUBMIT_DIR
```

en-tête du script : balises SLURM

balises spécifiques à la réservation des ressources

- nombre de nœuds, de tâches, temps maximum ...





# LANCEMENT DES CALCULS : SCRIPT BATCH

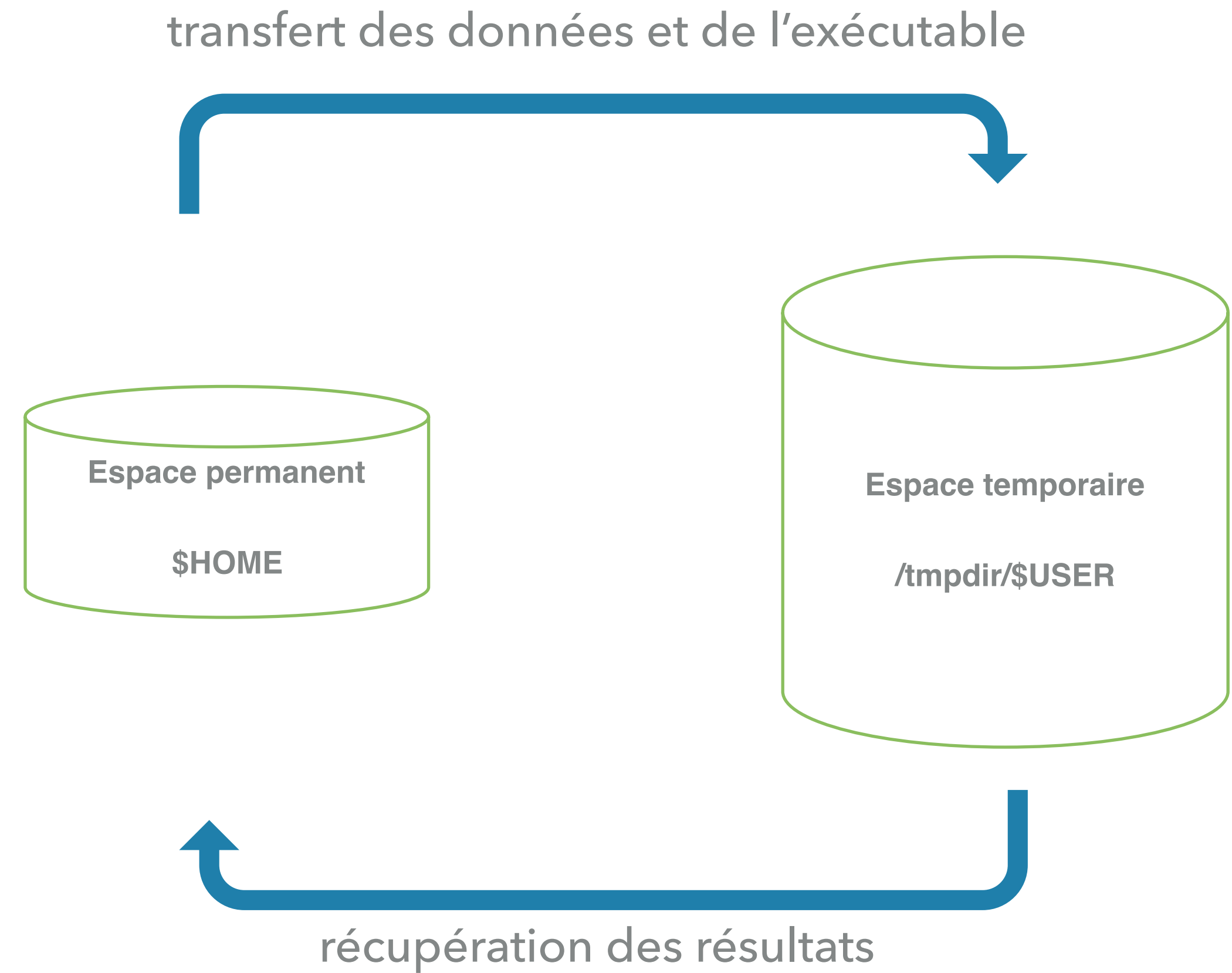
```
#!/bin/bash
#SBATCH --job-name=script_UtilisationEOS
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=20
#SBATCH --ntasks-per-core=1
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com
```

```
dirname=${SLURM_JOBID}
mkdir /tmpdir/$USER/$dirname
cp mes_inputs /tmpdir/$USER/$dirname
cd /tmpdir/$USER/$dirname
```

```
module purge
module load module1 module2
module list
```

```
./mon_appli.exe > output_${SLURM_JOBID}.log
```

```
mv mes_outputs $SLURM_SUBMIT_DIR
```



# LANCEMENT DES CALCULS : SCRIPT BATCH

```
#!/bin/bash
#SBATCH --job-name=script_UtilisationEOS
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=20
#SBATCH --ntasks-per-core=1
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com
```

```
dirname=${SLURM_JOBID}
mkdir /tmpdir/$USER/$dirname
cp mes_inputs /tmpdir/$USER/$dirname
cd /tmpdir/$USER/$dirname
```

```
module purge
module load module1 module2
module list
```

```
./mon_appli.exe > output_${SLURM_JOBID}.log
```

```
mv mes_outputs $SLURM_SUBMIT_DIR
```



chargement des modules requis



# LANCEMENT DES CALCULS : RESERVATION DES RESSOURCES

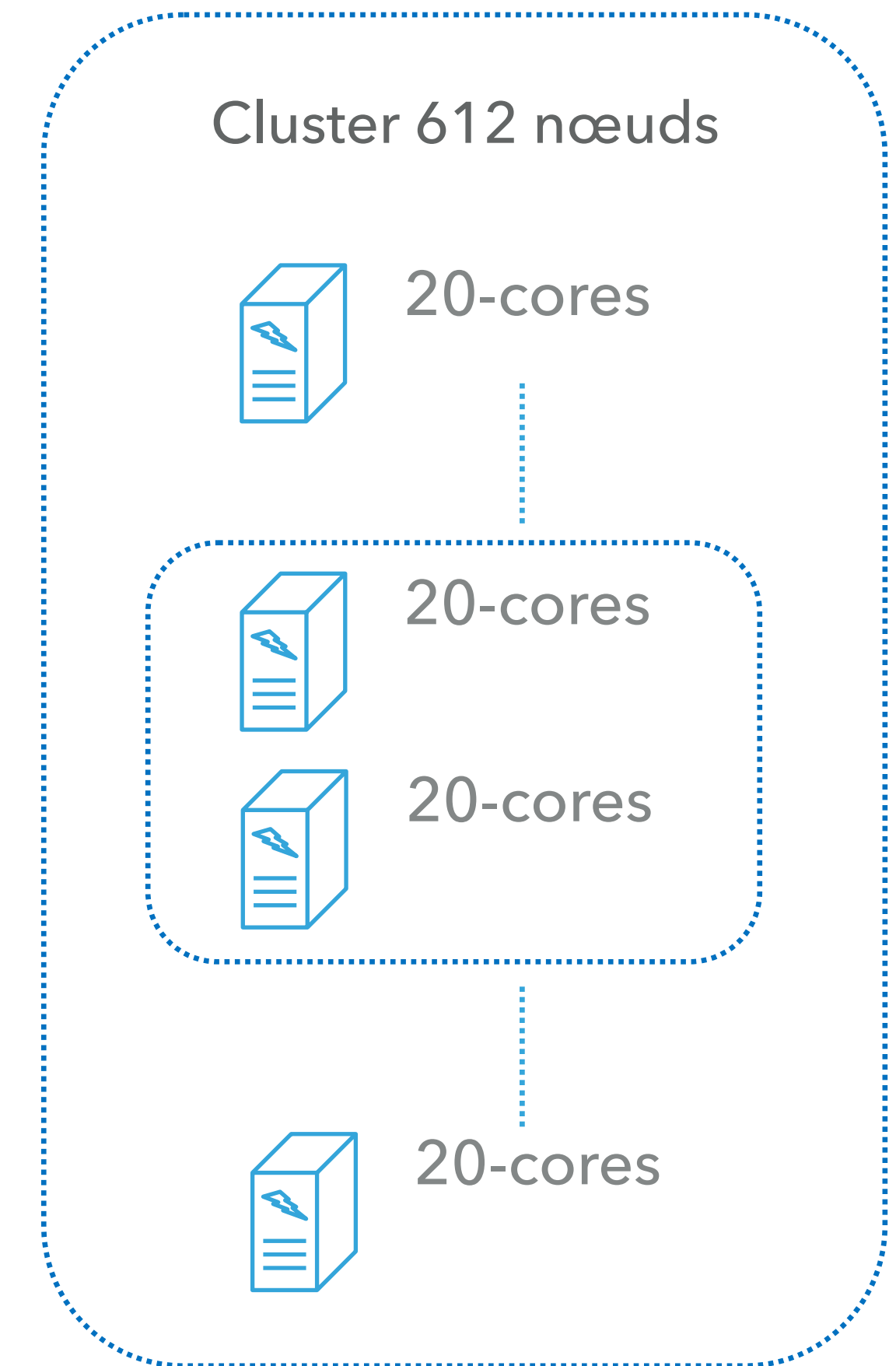
- ▶ Réserve de plus de 10 tâches : nœud entier alloué

```
#SBATCH --nodes=1
#SBATCH --ntasks=20
#SBATCH --ntasks-per-node=20
```

```
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=20
```

- ▶ Réserve de moins de 10 tâches : spécifier la mémoire requise

```
#SBATCH --nodes=1
#SBATCH --ntasks=5
#SBATCH --ntasks-per-node=5
#SBATCH --mem=10000
```



- ▶ Principe général : on réserve un certain nombre de nœuds



# LANCEMENT DES CALCULS : FILES D'ATTENTE

File d'attente	Nombre de cœurs	Nombre de nœuds	Walltime	jobs/user	RAM	Remarque	Partition
mono	< 10	1	400h	3 max	32 Go max	non exclusif - HT	Shared
nœud	20	1	300h	3 max	60 Go	exclusif - HT	Exclusive
nœud9	40 - 180	2 - 9	200h	2 max	60 Go/nœud	exclusif - HT	Exclusive
noeud18	200 - 360	10 - 18	150h	2 max	60 Go/nœud	exclusif - HT	Exclusive
noeud36	380 - 720	19 - 36	100h	1 max	60 Go/nœud	exclusif - HT	Exclusive
noeud72	740 - 1440	37 - 72	48h	1 max	60 Go/nœud	exclusif - HT	Exclusive
noeud90	1460 - 1800	73 - 90	36h	1 max	60 Go/nœud	exclusif - HT	Exclusive
visu	1 - 20	1	2h	1 max	126 Go max	non exclusif - HT	visu
mesca	1 - 64	1	100h	1 max	1 To max	non exclusif	mesca



# LANCEMENT DES CALCULS : ACCOUNTING

- ▶ Pour un calcul de moins de 10 tâches (nœud partagé) :  
*(nombre de CPUS réservés) \* (temps de réservation effectivement utilisé)*
- ▶ Pour un calcul de plus de 10 tâches (nœuds alloués exclusivement) :  
*(nombre de nœuds réservés) \* (20 CPUS) \* (temps de réservation effectivement utilisé)*
- ▶ Pour connaître sa consommation :  
*maconso*
- ▶ Pour connaître la consommation de chacun des collaborateurs du projet :  
*maconso\_detail*



# ENVIRONNEMENT DE CALCUL : LES MODULES

## ► Environnement par défaut :

```
[user@eoslogin1 ~]$ module list
Currently Loaded Modulefiles:
  1) oscar-modules/1.0.3  2) intel/14.0.2.144  3) intelmpi/4.1.3.049

[user@eoslogin1 ~]$ ifort -V
Intel(R) Fortran Intel(R) 64 Compiler XE for applications running on Intel(R) 64, Version
14.0.2.144 Build 20140120
Copyright (C) 1985-2014 Intel Corporation. All rights reserved.
```

## ► Modules disponibles :

```
[user@eoslogin1 ~]$ module available
----- /usr/local/modules/modulefiles
-----
ansys/15.07          intel/11.1.064      openmpi/1.8.2
ansys/16             intel/12.1.5        openmpi/2.0.2
ansys/16.2          intel/13.1.1        paraview/4.1.0
ansys/17.0          intel/14.0.2.144(default)  paraview/4.4.0(default)
ansys/17.2          intel/15.0.090      paraview/5.3.0
...

```



# ENVIRONNEMENT DE CALCUL : LES MODULES

## ▶ Opérations supplémentaires sur les modules

- Charger un module

*module load new\_module*

- Décharger un module

*module unload old\_module*

- Décharger tous les modules

*module purge*

- Echanger un module

*module switch old\_module new\_module*

- Décharger un module

*module unload old\_module*

- Visualiser l'effet d'un module sur les variables d'environnement

*module display new\_module*



# ENVIRONNEMENT DE CALCUL : LES LIBRAIRIES SCIENTIFIQUES

► Intel® Math Kernel Library (MKL) : BLAS, LAPACK, ScaLAPACK ...

- Les modules Intel intègrent la MKL

```
[user@eoslogin1 ~]$ module show intel/17.0.4  
prepend-path INCLUDE      usr/local/intel_new/2017_4/mkl/include
```

- Linker BLAS-LAPACK avec le compilateur Intel

```
ifort mon_prog.f90 -mkl=sequential
```

```
ifort mon_prog.f90 -mkl=parallel
```

- Linker ScaLAPACK avec le compilateur Intel et IntelMPI

```
mpiifort mon_prog.f90 -mkl=cluster
```





# ENVIRONNEMENT DE CALCUL : LES BIBLIOTHÈQUES ET LOGICIELS

- ▶ Bibliothèques scientifiques
  - FFTW, HDF5, MUMPS, NetCFD, PETSc ...
  
- ▶ Logiciels scientifiques
  - Python, R, OpenFOAM, Gaussian, VASP, Quantum Espresso ...
  
- ▶ Logiciels de visualisation
  - Paraview, Salome, Gaussview, Jmol ...



# ENVIRONNEMENT DE CALCUL : LANCER UN CODE MPI

- ▶ Parallélisme en mémoire distribué (avec IntelMPI) :

```
#!/bin/bash
#SBATCH --job-name=script_UtilisationEOS
#SBATCH --nodes=2
#SBATCH --ntasks=40
#SBATCH --ntasks-per-node=20
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com

module load intelmpi/5.1.3.210

export I_MPI_LIBRARY=/usr/lib64/libpmi.so

srun ./mon_appli.exe
```

- application multi-nœud / multi-processus
- spécification du nombre de tâches MPI

chargement du module Intel MPI (d'autres implémentation MPI sont disponibles)



# ENVIRONNEMENT DE CALCUL : LANCER UN CODE OPENMP

## ▶ Parallélisme en mémoire partagée (multithreading) :

```
#!/bin/bash
#SBATCH --job-name=script_UtilisationEOS
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=20
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com

export OMP_NUM_THREADS=20
export OMP_PROC_BIND=true

# Si utilisation de la MKL
export MKL_NUM_THREADS=$OMP_NUM_THREADS

srun ./mon_appli.exe
```

- application mono-nœud / mono-processus
- spécification du nombre de cœurs dédiés au processus

variable OpenMP spécification le nombre de threads



## POUR ALLER PLUS LOIN

- ▶ Améliorer les performances
  - Compiler, Debugger, Mesurer, Vectoriser ...



- [Page web associée](#)

- ▶ Une simulation en vidéo
  - Exemple complet de simulation sur CALMIP



- [Page web associée](#)

