

# PLATEFORME DE CALCUL INTENSIF : KIT TECHNIQUE



CALMIP (UMS 3667)  
Espace Clément Ader

[www.calmip.univ-toulouse.fr](http://www.calmip.univ-toulouse.fr)

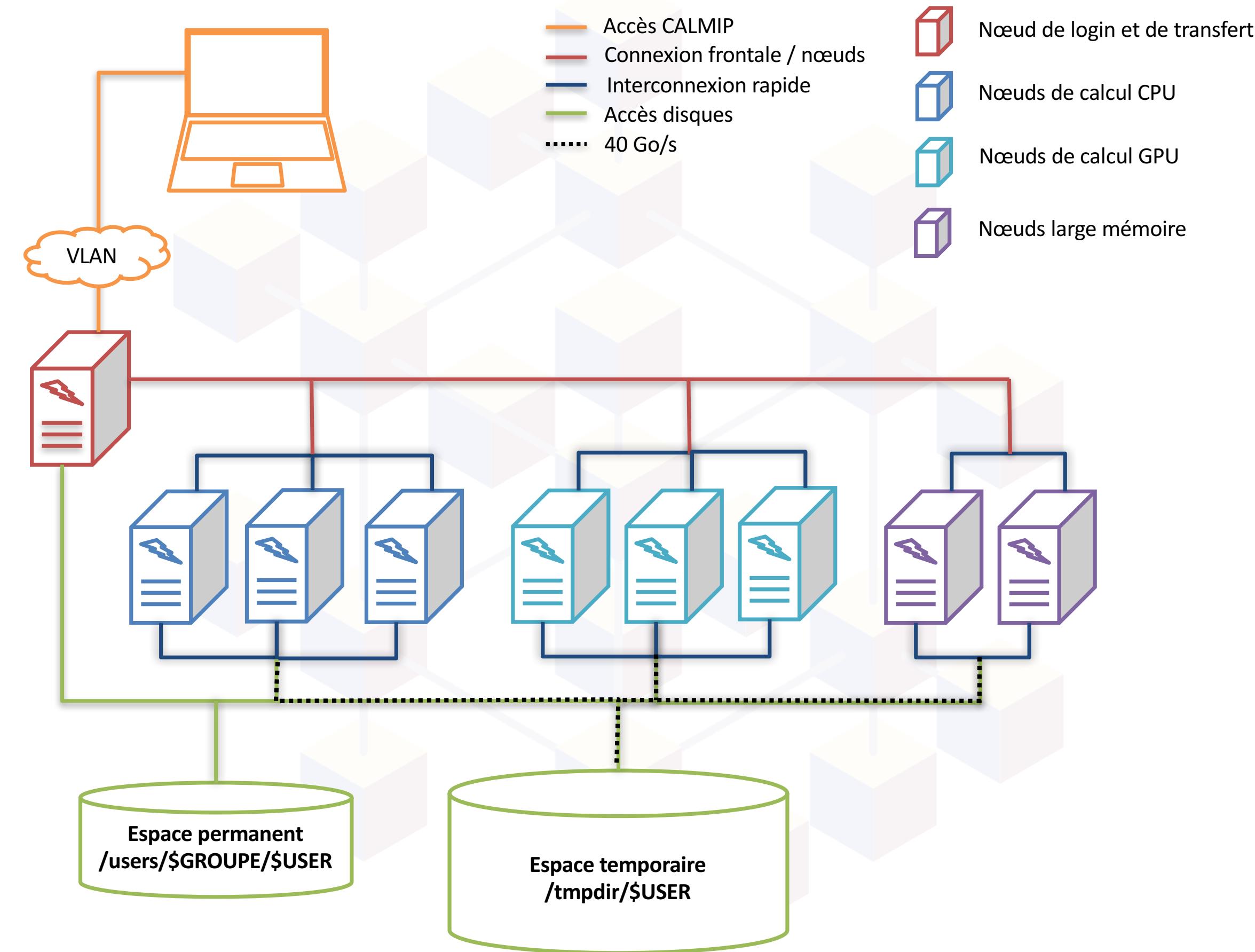


**INSA** INSTITUT NATIONAL  
DES SCIENCES APPLIQUÉES  
TOULOUSE



UNIVERSITÉ  
TOULOUSE III  
PAUL SABATIER

# PRÉSENTATION : SCHÉMA DU SYSTÈME DE CALCUL



Frontales de connexion :

- ▶ 3 x (36-cores, 192 GB RAM)

Cluster distribué Sequana (Atos-Bull) :

- ▶ 12 960 cores - 360 nodes
- ▶ Intel® Skylake 2,3 Ghz 2x18-cores
- ▶ 192 GB RAM / nœud
- ▶ Interconnection : Infiniband EDR

Noeuds GPU :

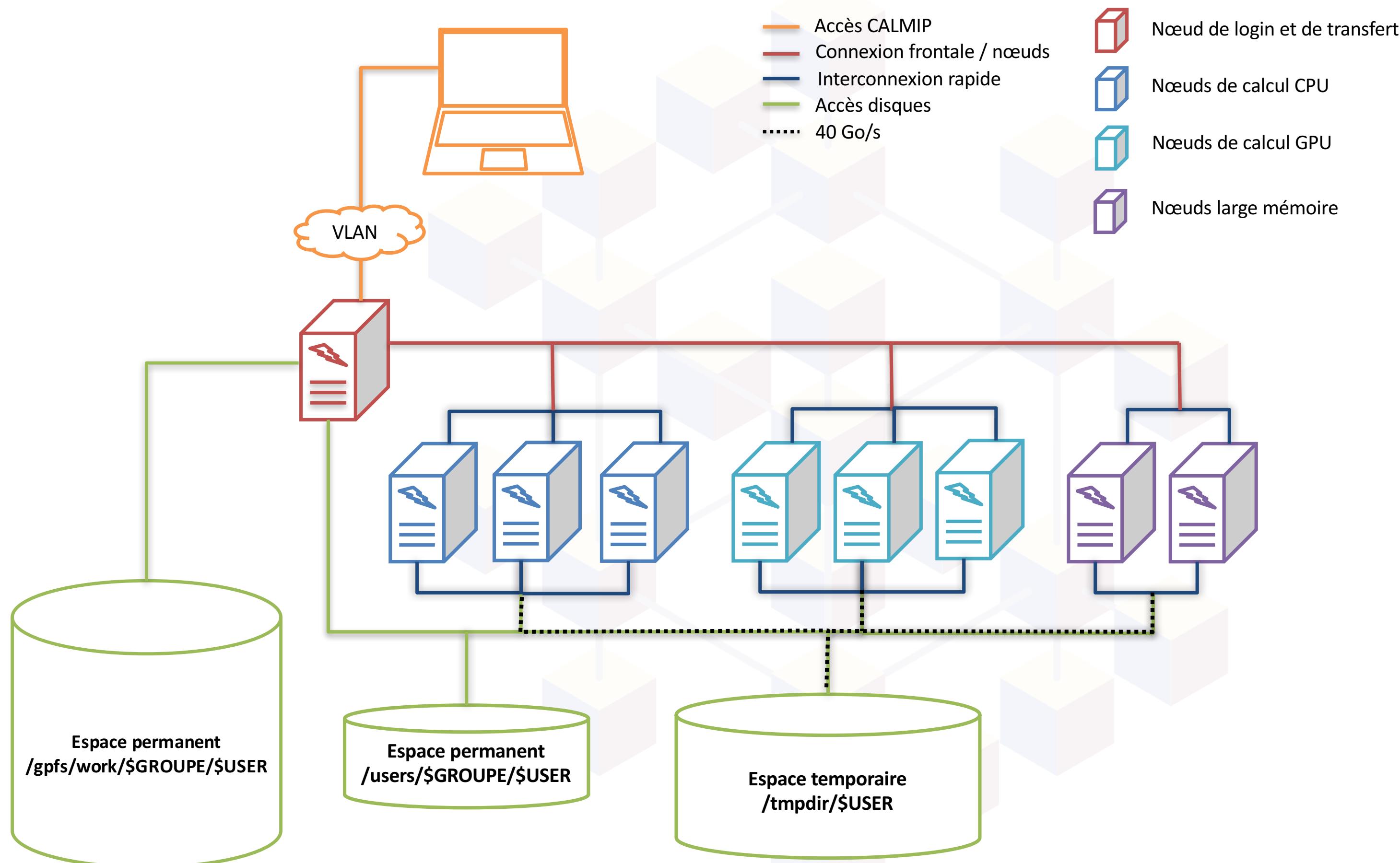
- ▶ Intel® Skylake 2,3 Ghz 2x18-cores
- ▶ 12 nœuds (4 GPU, 384 GB RAM)
- ▶ Cartes GPU Nvidia Volta (V100)

Nœuds large mémoire :

- ▶ Intel® Skylake 2,3 Ghz 2x18-cores
- ▶ 1,5 TB RAM / noeud



# PRÉSENTATION : ESPACES FICHIERS



- Page web associée

## ▶ Espace permanent (NFS) :

- 5 Go par utilisateur
- sauvegardes quotidiennes

## ▶ Espace temporaire (Lustre) :

- 1,5 Po 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'OLYMPE : LA FRONTALE DE CONNEXION

## ▶ Connexion « Secure Shell » (ssh)

- À partir d'un poste Linux / macOS

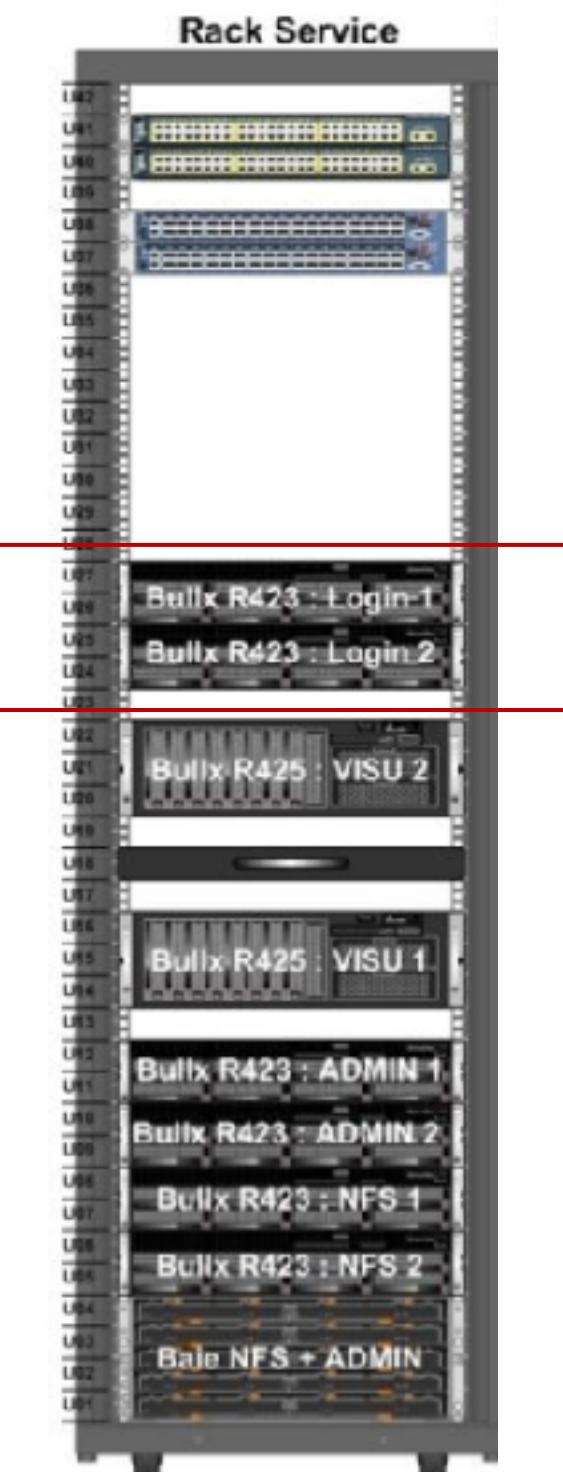
`ssh -X {login}@olympe.calmip.univ-toulouse.fr`

- À partir d'un poste Windows

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

Frontales de connexion :

- ▶ 3 x (36-cores, 192 GB RAM)



- Page web associée

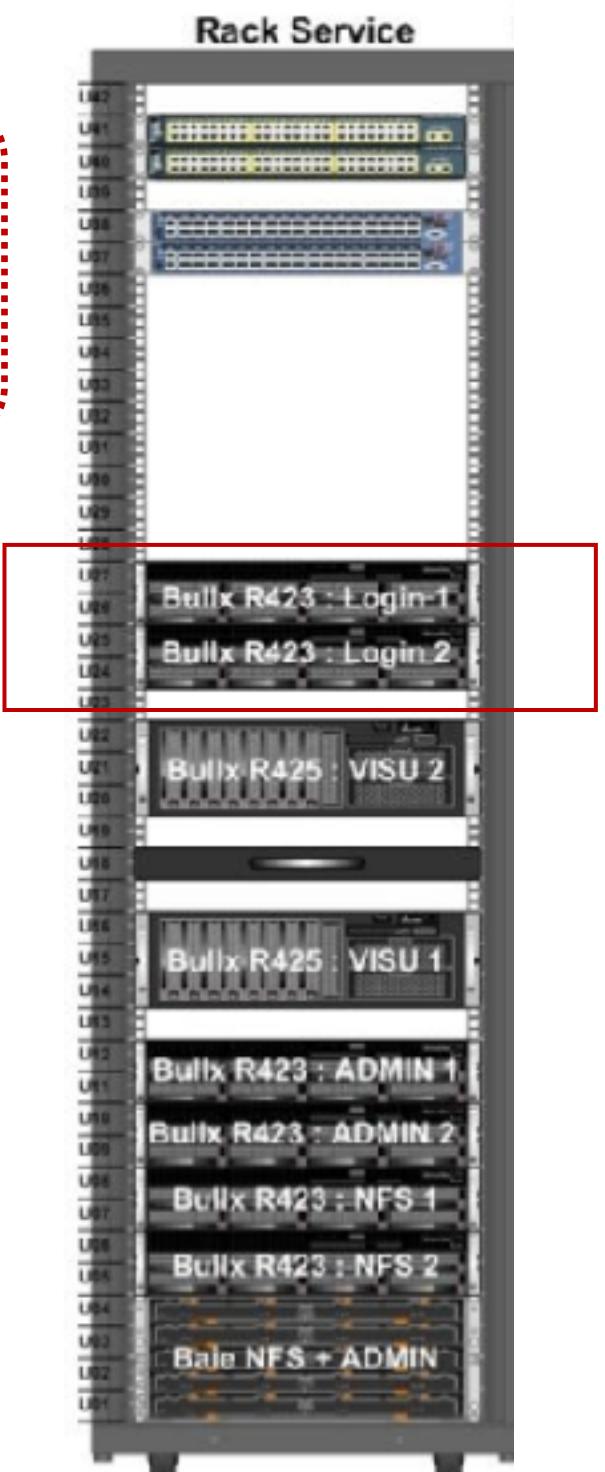
# PRISE EN MAIN D'OLYMPE : LA FRONTALE DE CONNEXION

## ▶ Ce que l'on peut faire sur une frontale

- compiler, installer un code
- transférer des fichiers
- effectuer des (petits) test, debugger

Frontales de connexion :

- ▶ 3 x (36-cores, 192 GB RAM)



## ▶ Ce que l'on ne peut PAS faire sur une frontale

- faire des calculs
- mode batch obligatoire



- Page web associée

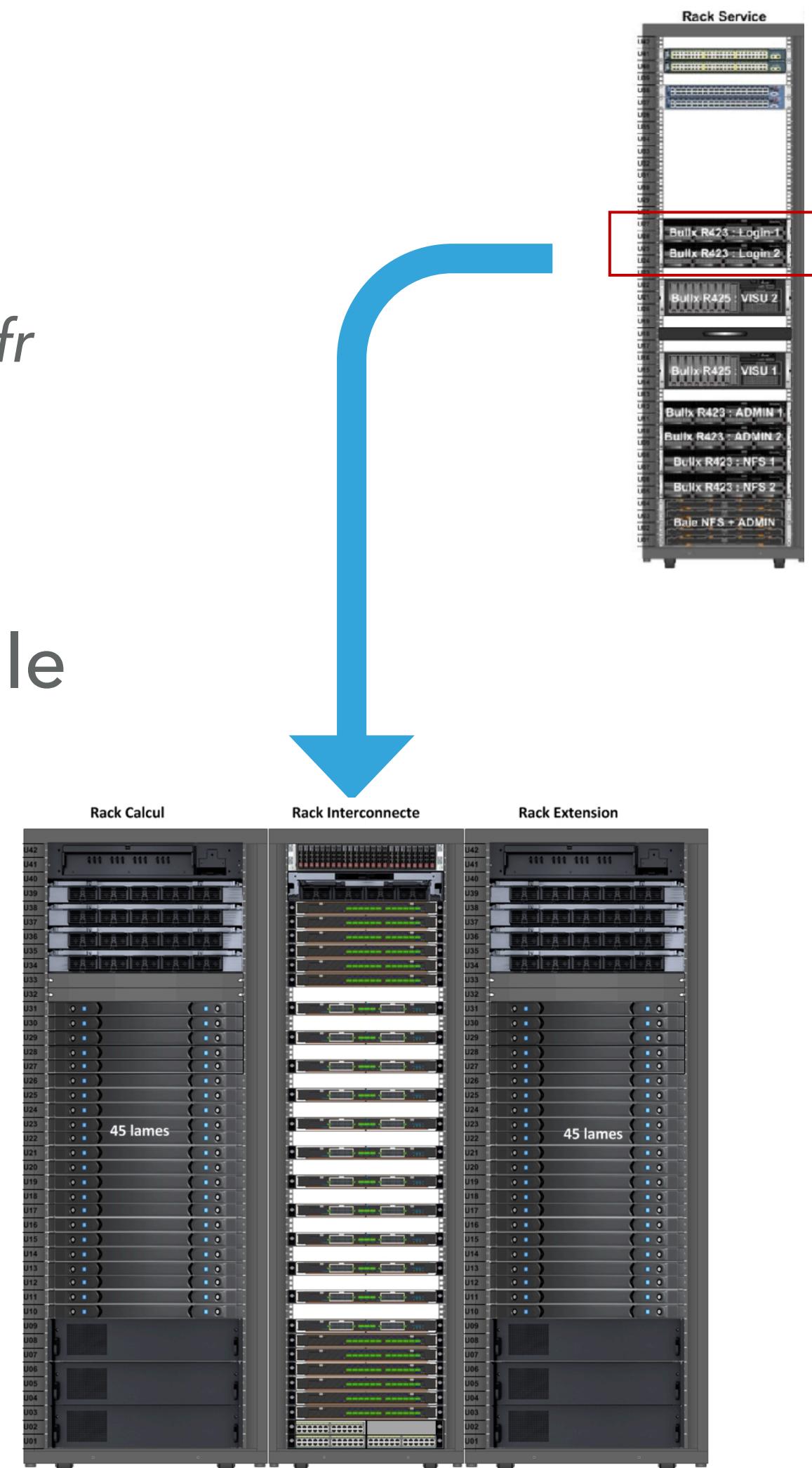
# LANCLEMENT DES CALCULS : PRINCIPES

## Connexion sur la frontale

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

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

`sbatch mon_job`



Frontales de connexion :

- 3 x (36-cores, 192 GB RAM)

Atos-Bull Sequana (13 464 cores - 374 nodes)

Processeurs Intel® Skylake 2,3 Ghz 2x18-cores

12 x 4 GPU Nvidia Volta V100

192-384 GB RAM / nœud

Interconnection : Infiniband EDR

# LANCLEMENT 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*



# LANCLEMENT DES CALCULS : SCRIPT BATCH

```

#!/bin/bash
#SBATCH --job-name=script_BATCH
#SBATCH --nodes=2
#SBATCH --ntasks=72
#SBATCH --ntasks-per-node=36
#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 ...



# LANCLEMENT DES CALCULS : SCRIPT BATCH

```

#!/bin/bash
#SBATCH --job-name=script_BATCH
#SBATCH --nodes=2
#SBATCH --ntasks=72
#SBATCH --ntasks-per-node=36
#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

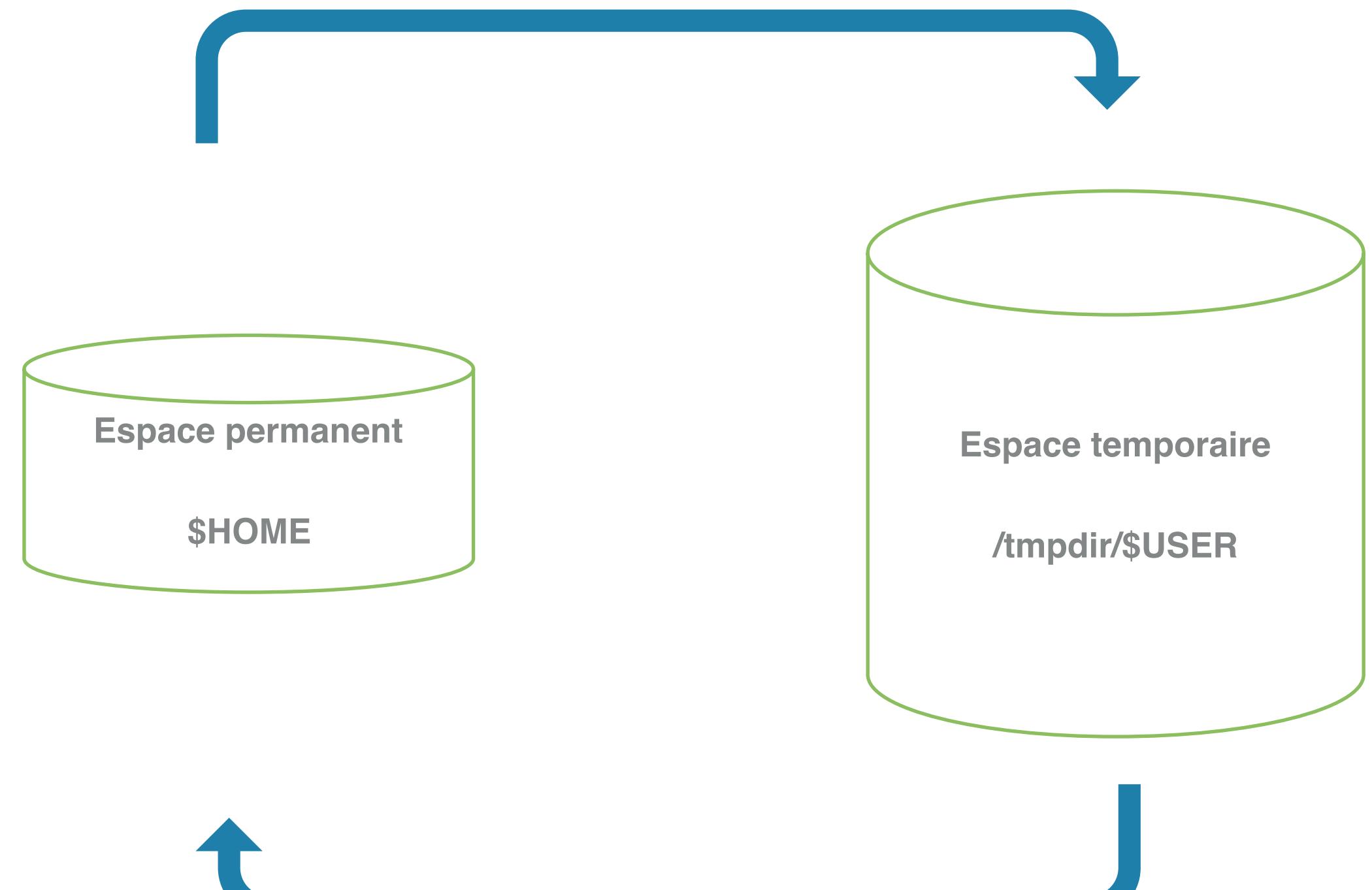
module purge
module load module1 module2
module list

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

mv mes_outputs $SLURM_SUBMIT_DIR

```

transfert des données et de l'exécutable



# LANCLEMENT DES CALCULS : SCRIPT BATCH

```

#!/bin/bash
#SBATCH --job-name=script_batch
#SBATCH --nodes=2
#SBATCH --ntasks=72
#SBATCH --ntasks-per-node=36
#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



- Page web associée

# LANCLEMENT DES CALCULS : RESERVATION DES RESSOURCES

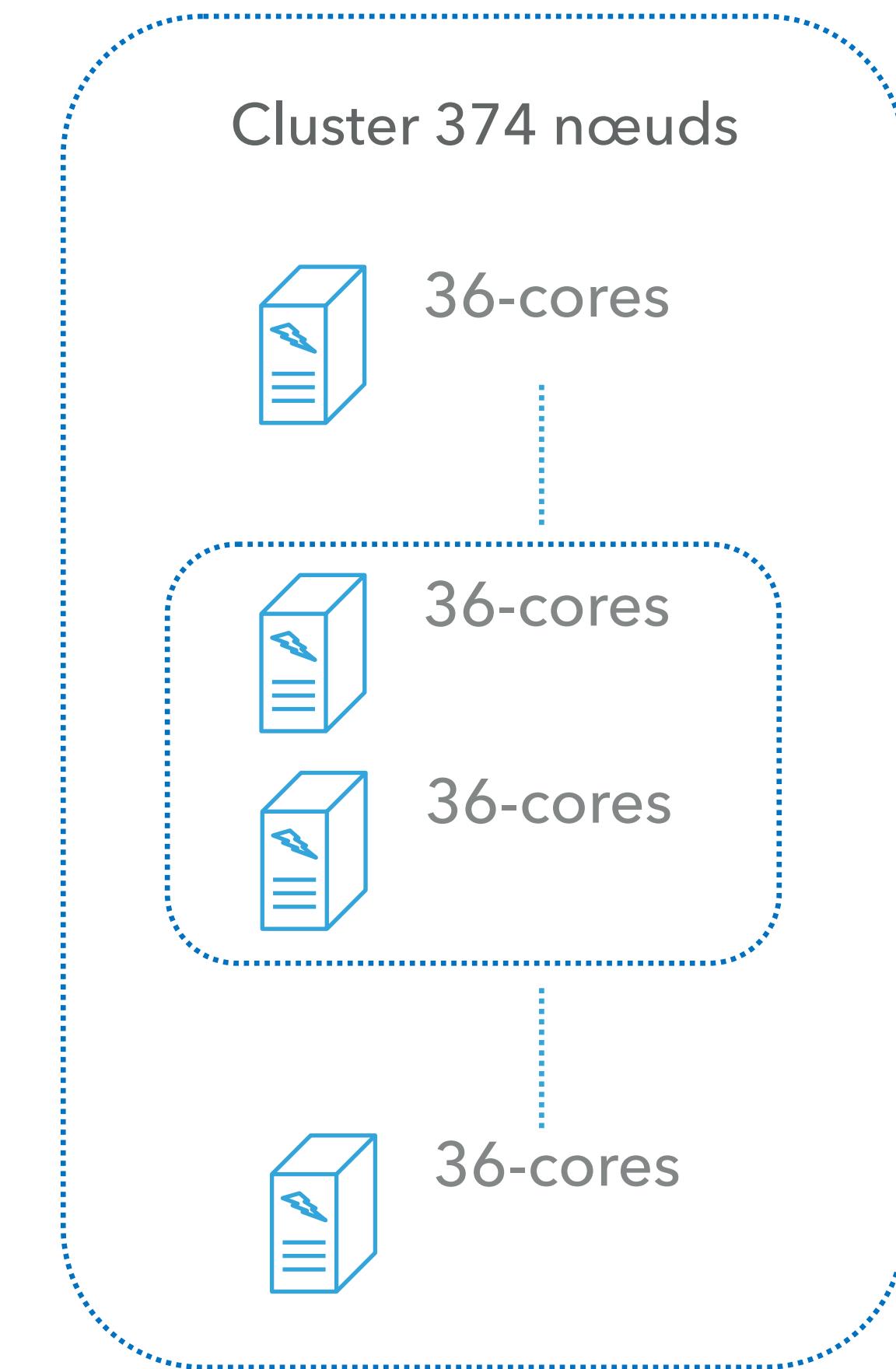
- ▶ Réservation de plus de 18 tâches : nœud entier alloué

```
#SBATCH --nodes=1  
#SBATCH --ntasks=36  
#SBATCH --ntasks-per-node=36
```

```
#SBATCH --nodes=2  
#SBATCH --ntasks=72  
#SBATCH --ntasks-per-node=36
```

- ▶ Réservation de moins de 18 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 noeuds



# LANCLEMENT DES CALCULS : RESERVATION GPU

## ▶ Réservation GPU partagée

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=9
#SBATCH --ntasks-per-node=9
#SBATCH --ntasks-per-core=1
#SBATCH --gres=gpu:1
#SBATCH --mem=20000
#SBATCH --time=01:00:00

module load cuda/9.1.85.3
```

- Réservation jusqu'à 2 GPU
- Réservation jusqu'à 18 coeurs CPU
- Réservation mémoire jusqu'à 192 Go

## ▶ Réservation GPU exclusive

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks=72
#SBATCH --ntasks-per-node=36
#SBATCH --ntasks-per-core=1
#SBATCH --gres=gpu:4
#SBATCH --time=01:00:00

module load cuda/9.1.85.3
```

- Réservation jusqu'à 44 GPU
- Réservation jusqu'à 396 coeurs CPU
- Réservation mémoire jusqu'à 377 Go/noeud



# LANCLEMENT DES CALCULS : FILES D'ATTENTE

File d'attente	Nombre de cœurs	Nombre de nœuds	Nombre de GPU	Walltime	jobs/user	RAM	Partition
mono	< 18	1	0	400 h	3 max	96 Go max	shared
nœud	36	1	0	250 h	2 max	192 Go	exclusive
nœud5	72 - 180	2 - 5	0	150 h	2 max	187 Go / nœud	exclusive
noeud10	216 - 360	6 - 10	0	110 h	2 max	187 Go / nœud	exclusive
noeud20	396 - 720	11 - 20	0	75 h	1 max	187 Go / nœud	exclusive
noeud40	756 - 1440	21 - 40	0	36 h	1 max	187 Go / nœud	exclusive
noeud50	1476 - 1800	41 - 50	0	24 h	1 max	187 Go / nœud	exclusive
visu	1 - 36	1	0	4 h	1 max	192 Go max	visu
mesca	1 - 18	1	0	100 h	1 max	750 Go max	mesca
voltam	1 - 18	1	1 - 2	100 h	1 max	192 Go max	volta
volta	18 - 396	1 - 11	1 - 44	100 h	1 max	377 Go max / noeud	volta



## LANCLEMENT DES CALCULS : ACCOUNTING

- ▶ Pour un calcul de moins de 18 tâches (nœud partagé) :  
*(nombre de CPUS réservés) \* (temps de réservation effectivement utilisé)*
- ▶ Pour un calcul de plus de 18 tâches (nœuds alloués exclusivement) :  
*(nombre de nœuds réservés) \* (36 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@olympelogin1 ~]$ module list
Currently Loaded Modulefiles:
 1) intel/18.2  2) intelmpi/18.2

[user@olympelogin1 ~]$ ifort -V
Intel(R) Fortran Intel(R) 64 Compiler for applications running on Intel(R) 64, Version
18.0.2.199 Build 20180210
Copyright (C) 1985-2018 Intel Corporation. All rights reserved.
```

## ► Modules disponibles :

```
[user@olympelogin1 ~]$ module available
----- /usr/share/Modules/modulefiles -----
openmpi-gnu/mt/ilp64/2.0.2.10 openmpi/icc/mt/ilp64/2.0.2.10
openmpi-gnu/2.0.2.10          openmpi/icc/2.0.2.10      use.own
openmpi-gnu/ilp64/2.0.2.10    openmpi/icc/ilp64/2.0.2.10
openmpi-gnu/mt/2.0.2.10       openmpi/icc/mt/2.0.2.10

----- /usr/local/modules/modulefiles/compilers_and_libraries -----
cuda/8.0.61.2    gcc/7.3.0      intel/16.4      intelmpi/13.2   intelmpi/17.1    pgi/18.3
cuda/9.0.176.2   intel/09.1     intel/17.1      intelmpi/14.0   intelmpi/18.2    tcltk/8.6.3
cuda/9.1.85.3    intel/12.1.5   intel/18.2      intelmpi/15.0   petsc/3.7.4/impi
gcc/5.4.0        intel/14.0     intel/18.2.199  intelmpi/16.4   petsc/3.7.4/ompi
```



- Page web associée

# 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@olympelogin1 ~]$ module show intel/18.2
prepend-path INCLUDE      /usr/local/intel/2018.2.046/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 ...



- [Page web associée](#)

# ENVIRONNEMENT DE CALCUL : LANCER UN CODE MPI

## ▶ Parallélisme en mémoire distribué (avec IntelMPI) :

```
#!/bin/bash
#SBATCH --job-name=script_batch
#SBATCH --nodes=2
#SBATCH --ntasks=72
#SBATCH --ntasks-per-node=36
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com

module load intelmpi/18.2

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_batch
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=36
#SBATCH --time=0-01:00:00
#SBATCH --mail-user=toto@mail.com

export OMP_NUM_THREADS=36
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 cores 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](#)

