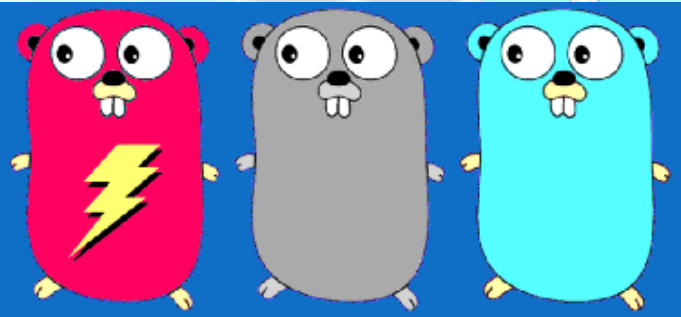
Génie Logiciel pour le Calcul Scientifique

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY



#7

17/01/2025

jean-michel.batto@cea.fr

cea

https://gogs.eldarsoft.com/M2_IHPS

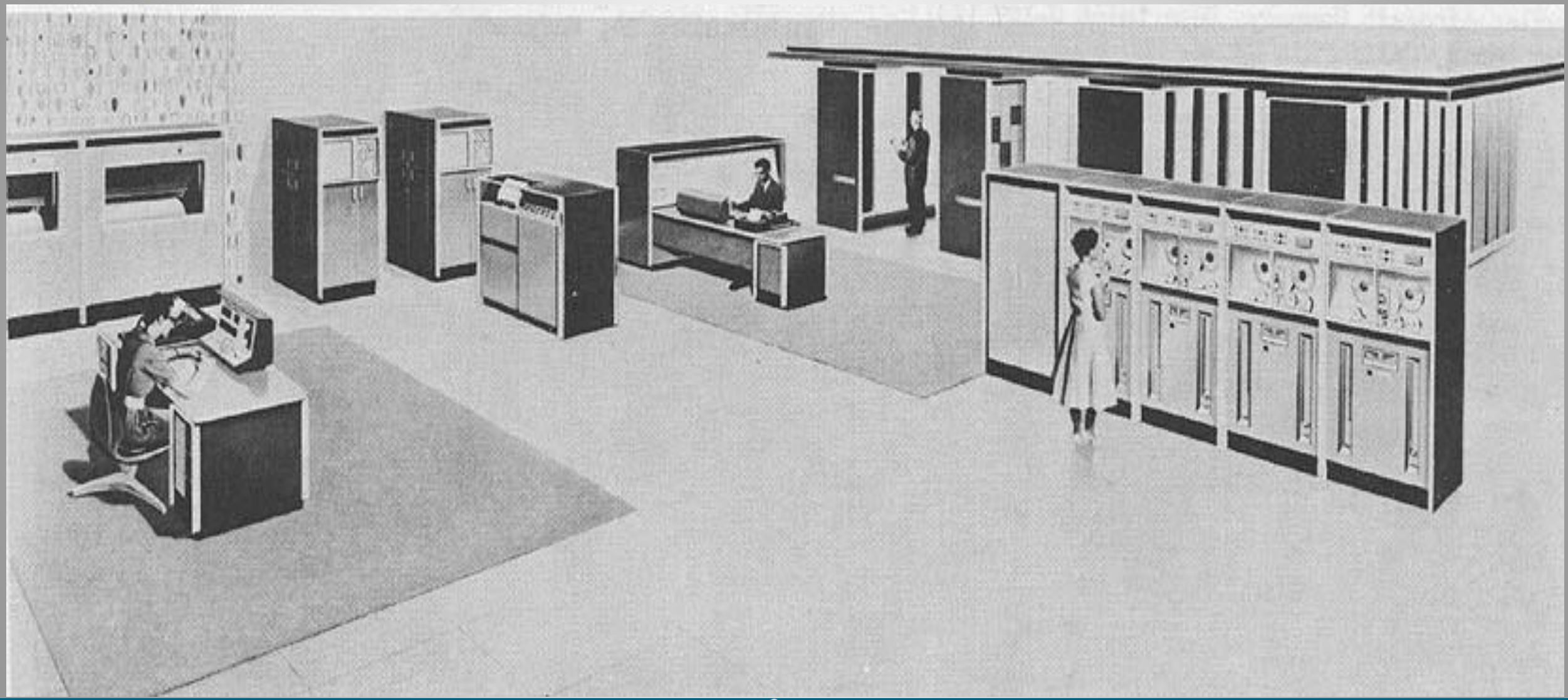
Réponse pour le TD GLCS-CM6-TDXMP

Comment faire la réduction entre les data_out associés aux « partitions » XMP ?

Le code doit être indépendant du nombre de partition.

```
  for (int i = 0; i < 20; i++) {
    int tmp = data_out[i];
#pragma xmp reduction(+ : tmp)
    data_out[i] = tmp;
  }
```

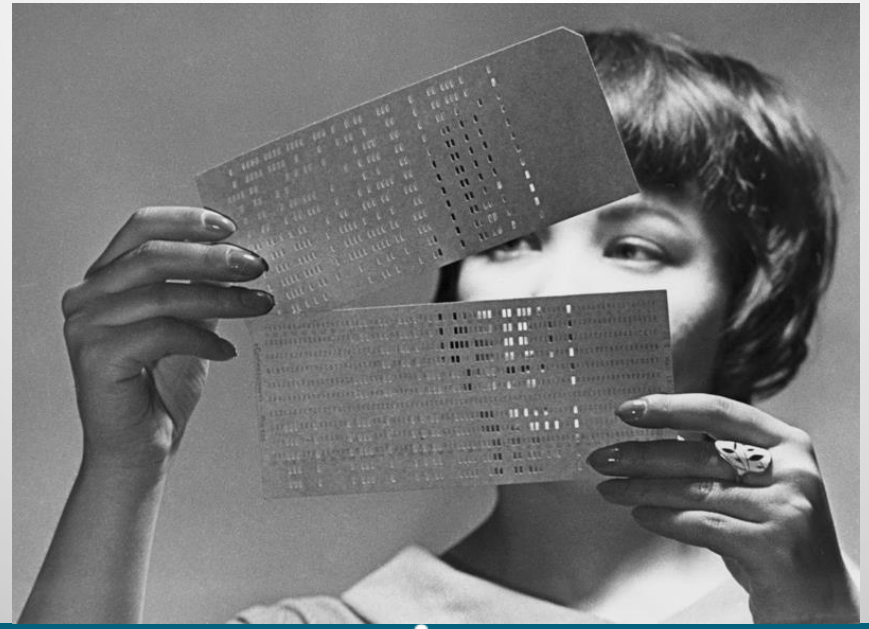
- https://slurm.schedmd.com



50's

- Premières générations d'ordinateurs
  - La programmation est faite à base de cartes perforées, regroupées par lot (batch).

    Les cartes les plus répandues ont 80 colonnes et 12 « lignes ».
  - Les entrées sont elles aussi fournies au travers de cartes, puis de bandes magnétiques.
  - Les sorties sont réalisées sur cartes ou imprimantes.



60 ’s

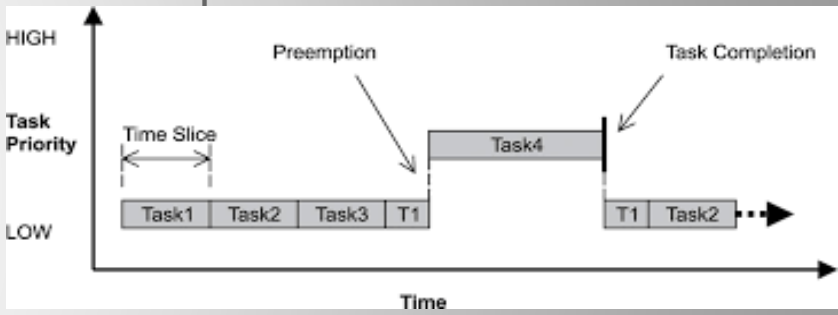
- Premières générations d’ordinateurs
  - L’utilisation des machines est assurée par des opérateurs qui chargent les paquets de cartes en machine suivant un planning pré-établi.
    - → batch scheduling
  - Les résultats, des « listings » papiers, sont fournis aux utilisateurs après l’exécution de leurs « jobs ».
    - Les « jobs » en erreur produisent une quantité pharamineuse de sorties

60's

- Premières générations d'ordinateurs
  - Mode d'utilisation
    - On planifie (schedule) l'exécution de jobs par paquets consécutifs de cartes perforées.
    - On génère des « listings » papiers.
  - On passe un temps certain à mettre au point les « cartes » de ses « jobs » et à en traiter les « listings »
- L'émergence de l'informatique moderne
  - L'arrivée des transistors, des bandes magnétiques et des mémoires permet la conception de nouvelles machines.
  - Les « mainframes » apparaissent
    - Les terminaux « graphiques » 80 colonnes font leur entrée.
  - Des lecteurs de cartes restent associés…
    - Pour réutiliser les codes…Et migrer vers des « scripts »

L'ère « mainframe » (70's)

- Les scripts et programmes se numérisent
  - Les cartes sont mises au placard après numérisation.
  - Les données sont enregistrées sur bandes magnétiques et chargées/écrites depuis les programmes.
- Les « jobs » sont planifiés par des opérateurs puis par des applications spécialisées.
  - Les premiers « batch scheduler »…
- Mode d'utilisation
  - « Soumission » de scripts batch par les utilisateurs (jobs).
  - Ordonnancement automatique de l'exécution des jobs par une application dédiée.
  - On génère des « listings » numérisés : sorties « écran » redirigées dans des fichiers.
- On gagne du temps dans la mise au point des scripts et programmes et le dépouillement des résultats.
- On attend en fonction de l'importance du programme plus ou moins longtemps avant son exécution.

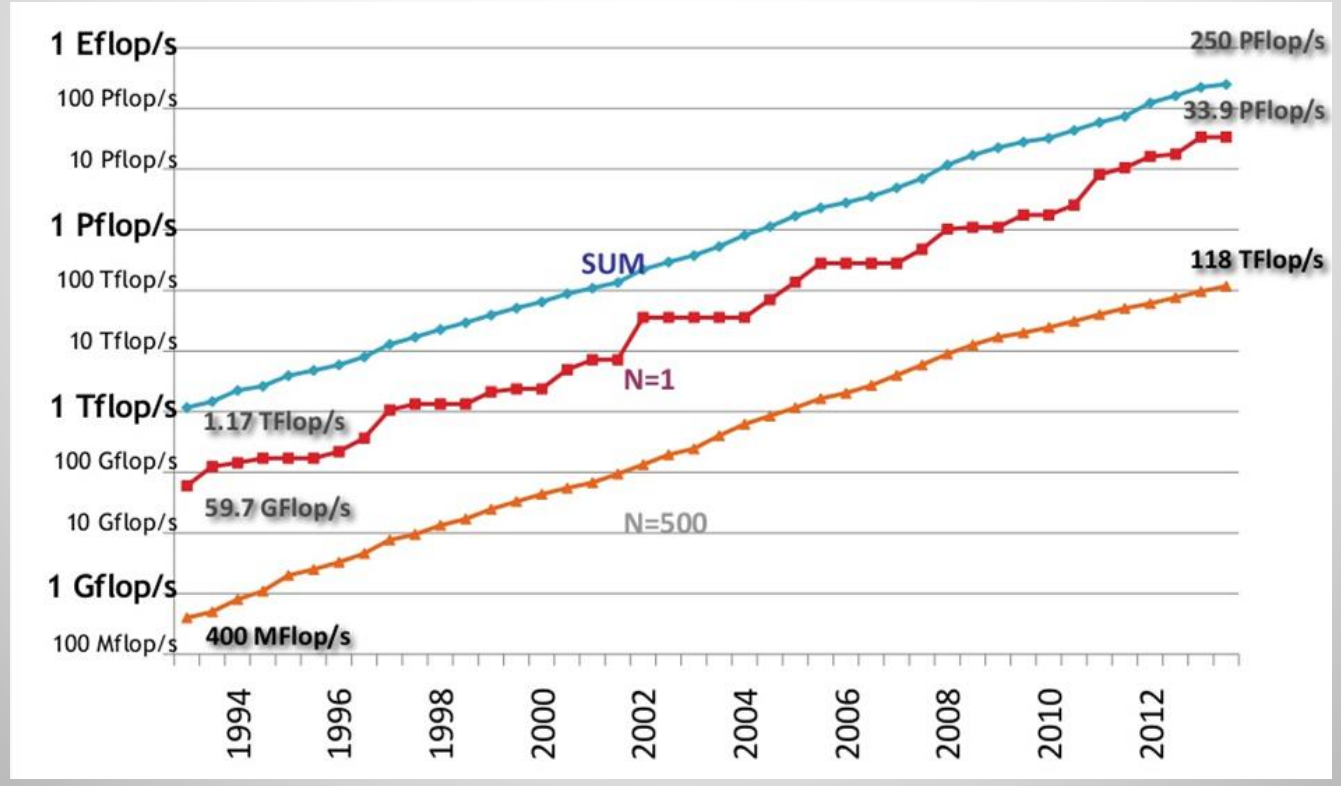


L'ère « PC » (80's - 2000's)

- L'informatique se miniaturise et se démocratise par le canal des « personnal computeur »
- Qui reprennent les concepts des « mainframe » en associant directement le terminal à l' « unité centrale ».
- L'évolution est forte et rapide.
  - Les interfaces graphiques
  - font vite leur entrée
- Les systèmes d'exploitation permettent (Unix 1971, Linux 1991)
  - Une interaction directe via des interfaces graphiques simplifiant l'utilisation des machines
  - Une interaction en mode ligne de commandes et/ou scripts.
    - Ex : fichiers script « .bat » de Windows
  - Les « scripts » restent exécutables en arrière plan (crontab) pour les traitements « batch ».

L'émergence des clusters (90's)

- Les réseaux prennent de l'ampleur et permettent une interconnexion performante d'unités individuelles type PC.
- Le HPC s'engouffre dans cette voie face à la diminution des performances des approches monolithiques des « Mainframe ».
  - Le nombre d'unités de calcul connectées ne cessera de croître...



L'émergence des clusters (90's)

- L'utilisation des clusters nécessite alors l'orchestration de plusieurs unités de calcul indépendantes.
  - Notion de « **jobs parallèles** » exécutés sur des « systèmes distribués »
- Un ordonnanceur central est en charge de la répartition « spatiale » et « temporelle » des travaux.
  - Dédier un certain nombre de « nœuds » pour une période de temps donnée à un « job ».
- Les jobs deviennent hétérogènes
  - Une ou plusieurs sections parallèles permettant l'exécution de codes de calcul optimisés pour l'utilisation de plusieurs unités de calcul
    - Émergence du modèle MPI ! (CSP)
  - Encapsulée(s) dans le déroulement classique du script « batch »
- L'ordonnancement se complexifie
  - Différents besoins en terme de nombres d'unités de calcul dans les sections parallèles.
  - Différentes localités.
- Les « batch scheduler » évoluent donc pour traiter efficacement ces « systèmes distribués »
  - On parle maintenant de **DRMS (Distributed Resource Management System) – PBS, Torque, SGE**
  - **Concept de pilotage (Pilot-Job)**

Principe d'un DRMS

- Rappels
  - Evolution des batchs scheduler « initiaux »
    - Gérant principalement des « jobs » en **time slicing**

      → **composant « job manager »**
    - Prise en charge d'une quantité de ressources de calcul grandissante et distribuée

      → **composant « resource manager »**

Usage d'un DRMS

- Repose généralement sur un composant **leader** central
  - Permettant aux utilisateurs d'enregistrer leurs « jobs » pour exécution ultérieure
  - Fournissant un statut des ressources disponibles et en cours d'utilisation
  - Fournissant un statut des jobs en cours de calcul ou en attente de ressources
  - Fournissant l'historique et les statistiques d'utilisation des ressources

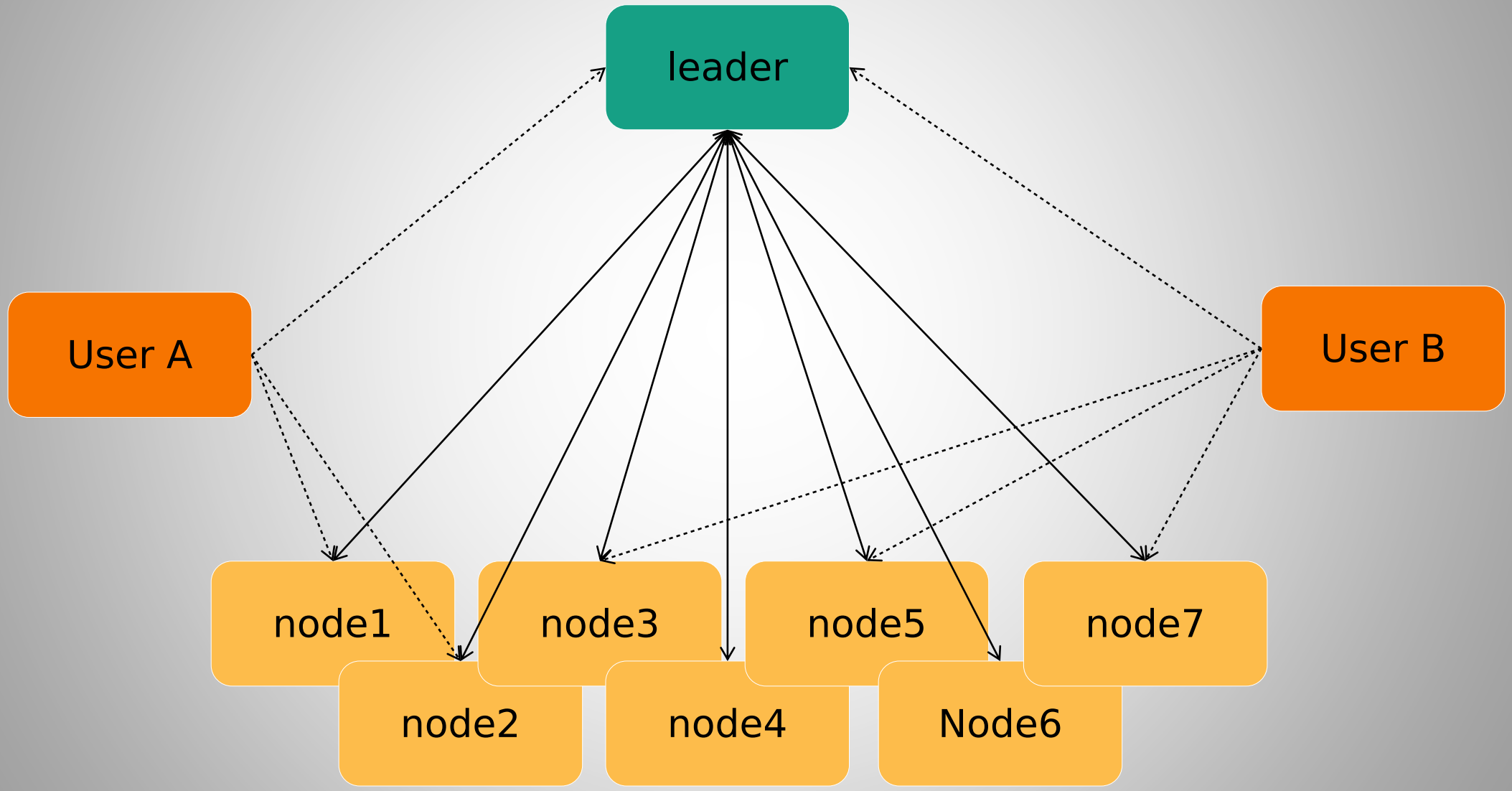
Architecture d'un DRMS

- Repose généralement sur un composant **leader** central
  - Orchestrant la répartition des ressources entre les « jobs » au cours du temps
  - Orchestrant la mise en exécution, l'arrêt des jobs ainsi que le suivi de la bonne utilisation et la libération des ressources utilisées

Architecture d'un DRMS

- Repose généralement sur un ensemble de **workers** distribués
  - Généralement un par nœud de calcul
  - Fournit l'état du nœud au leader et permet les intéractions directes avec celui-ci ou les utilisateurs
  - En charge du démarrage des exécutions de scripts et ou d'applications pour les utilisateurs
  - Assure le suivi de la bonne utilisation et la libération des ressources utilisées

Architecture DRMS (Pilot-Job)



Un DRMS : SLURM

- Simple Linux Utility for Resource Management
  - Simple → Scalable
- Projet démarré au LLNL en 2002
  - Lawrence Livermore National Laboratory, Livermore, CA, USA
- Continué par SchedMD depuis 2010
  - Entreprise créé par les deux développeurs principaux
- Produit OpenSource écrit en C : Licence GPLv2
- Utilisable sur la majorité des environnements de type UNIX
  - AIX, Linux, BSD, ...
- Utilisé sur une multitude de grands calculateurs à travers le monde
  - parmi les plus grands

slurm workload manager

Objectifs

- Scalable
  - Permet la gestion de plusieurs dizaines de milliers de nœuds
  - Permet la gestion de plusieurs centaines de milliers de cœurs de calcul
- Modulaire
  - Basé sur la notion de plugins pour spécialiser différentes parties du produit en fonction des besoins

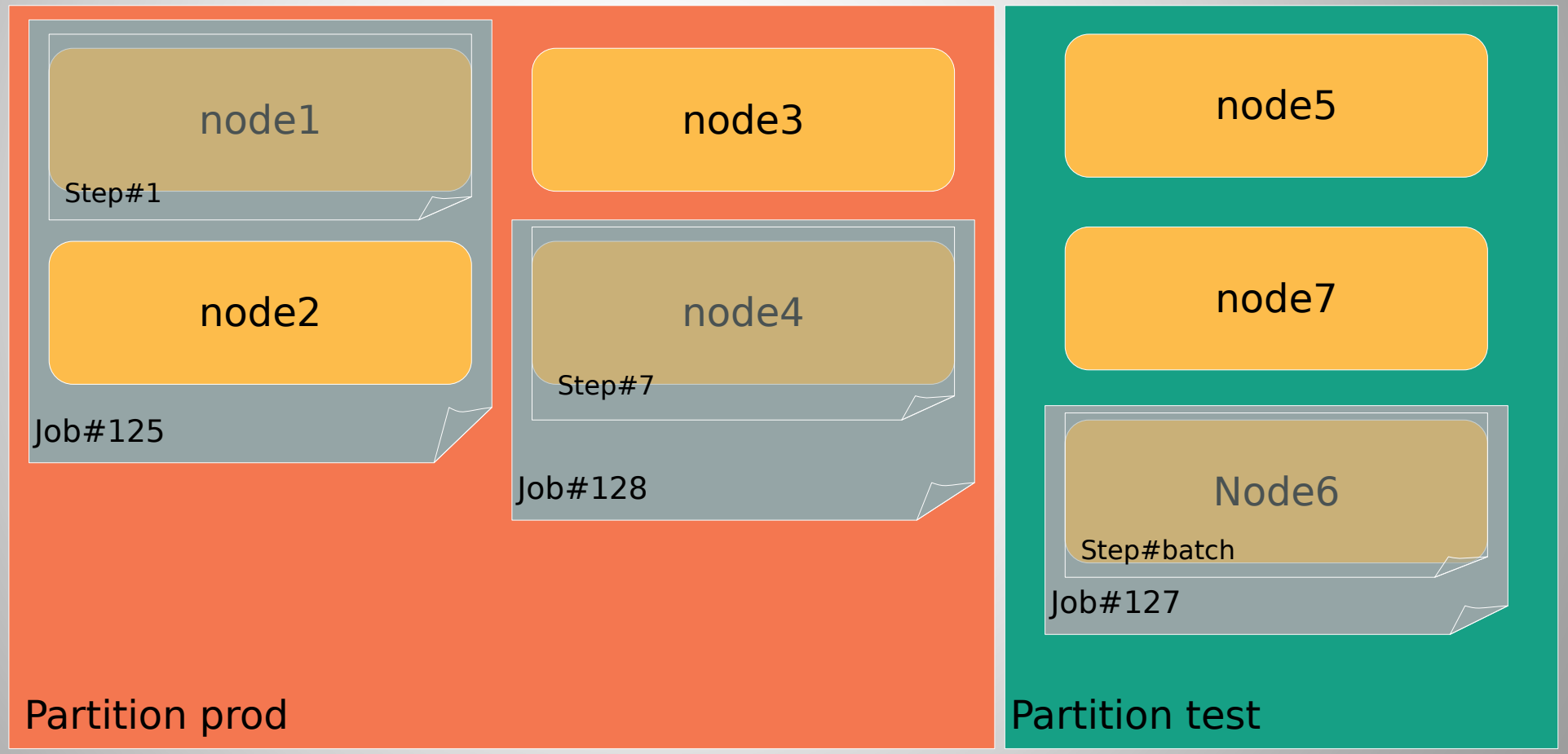
Composants

- slurmctld
  - Composant « leader » (controler)
- slurmdbd
  - Composant additionnel au «leader » pour la persistence des données de comptabilité sur les jobs et la gestion des utilisateurs et de leurs droits
    - Backend MariaDB/Mysql nécessaire
- slurmd
  - Composant « worker »

Concepts

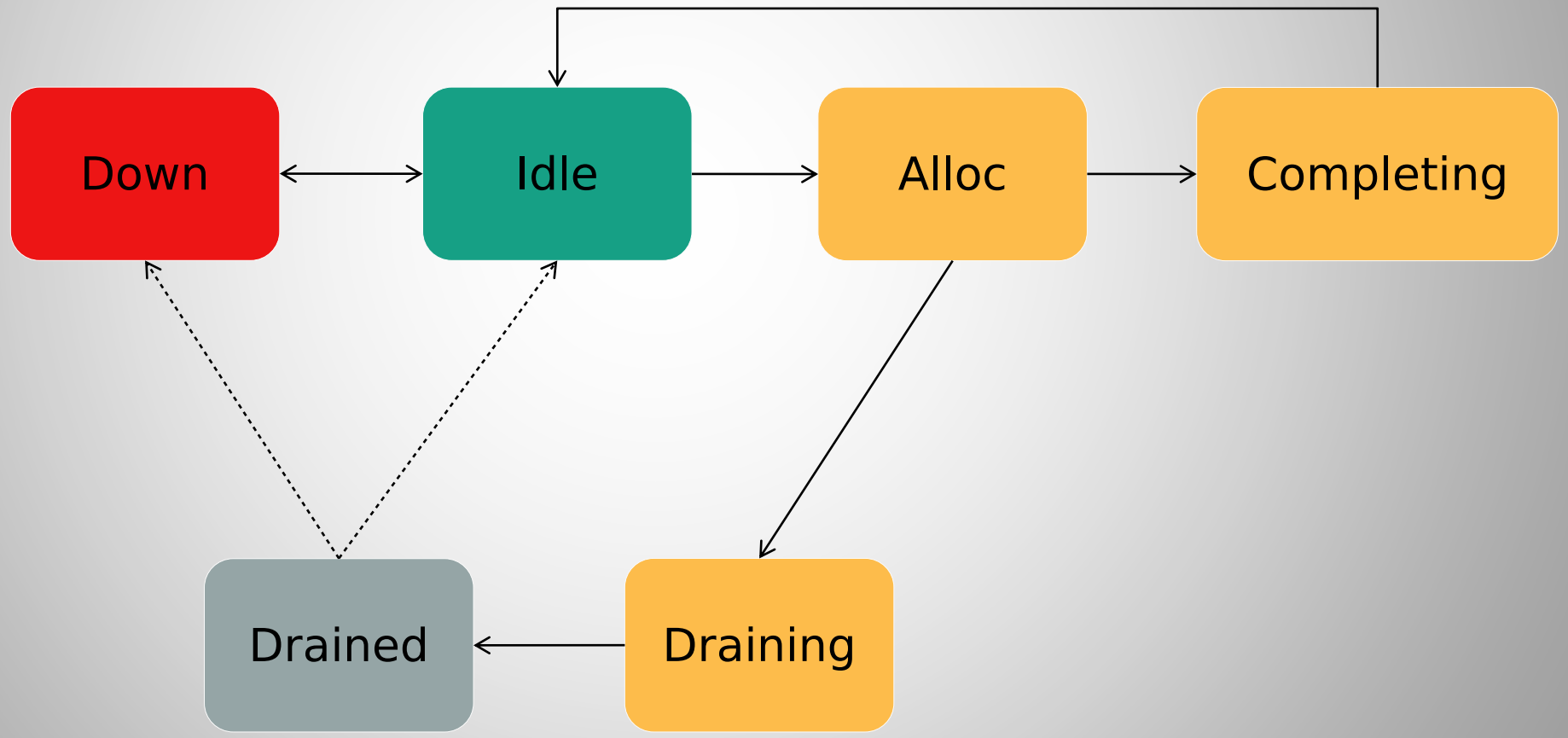
- Partition
  - « Pool » de nœuds utilisables au sein d'un même « job »
    - Un nœud peut appartenir à plusieurs partitions
- Node
  - Unité indépendante fournissant des ressources utilisables par les utilisateurs
    - Sockets/Cores/Threads, Memory, GPUs, …
- Job
  - Demande d'allocation de ressources *dans une partition* associée à un utilisateur
    - Ensemble de ressources réparties sur des nœuds pour un temps défini
    - Batch (script fourni) ou Interactif (shell)
- Jobstep
  - Demande de sous-allocation de ressources pour effectuer une tâche particulière
    - Sous-ensemble de ressources parmi les ressources allouées pour le job associé

Articulation



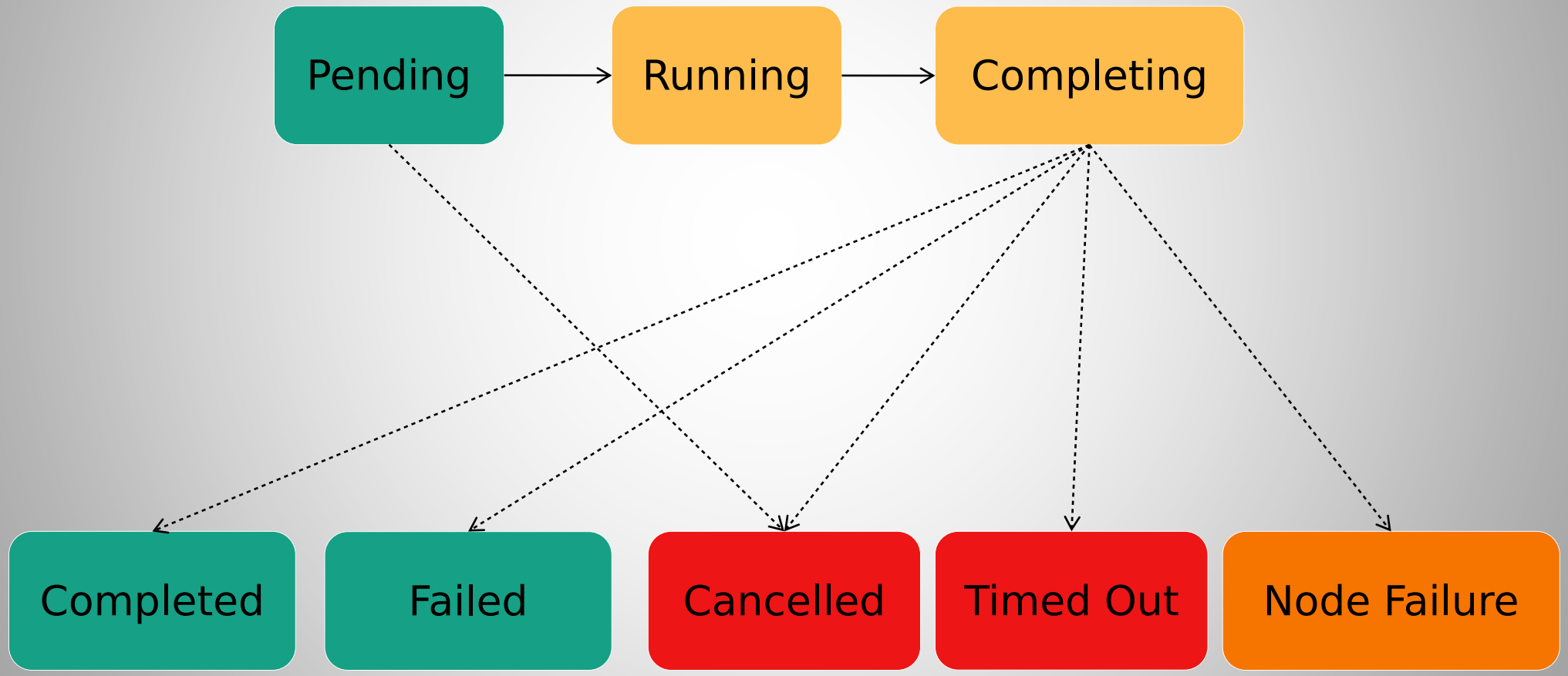
Node

Node « states »

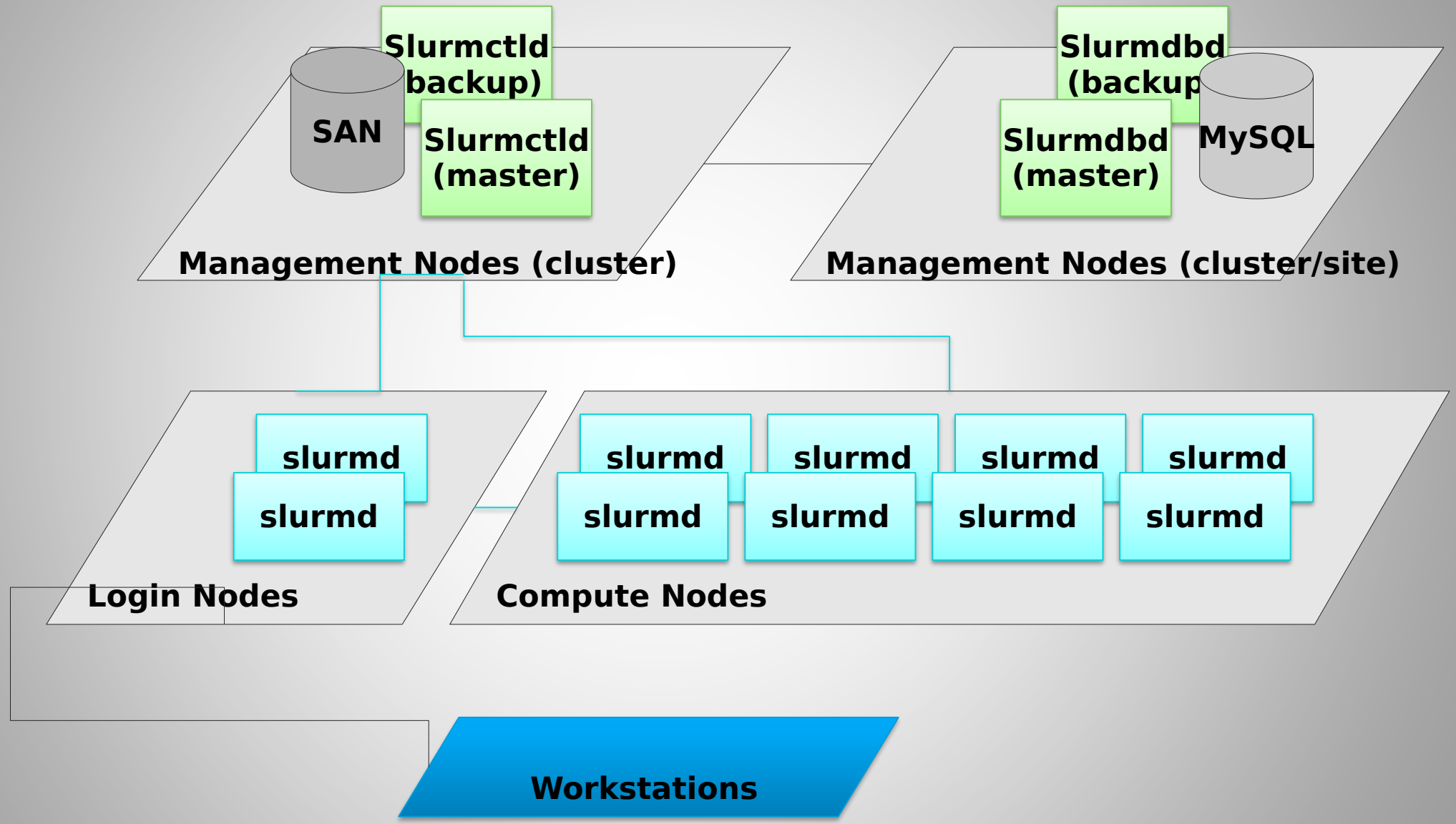


Job

Job « states »



Organisation physique

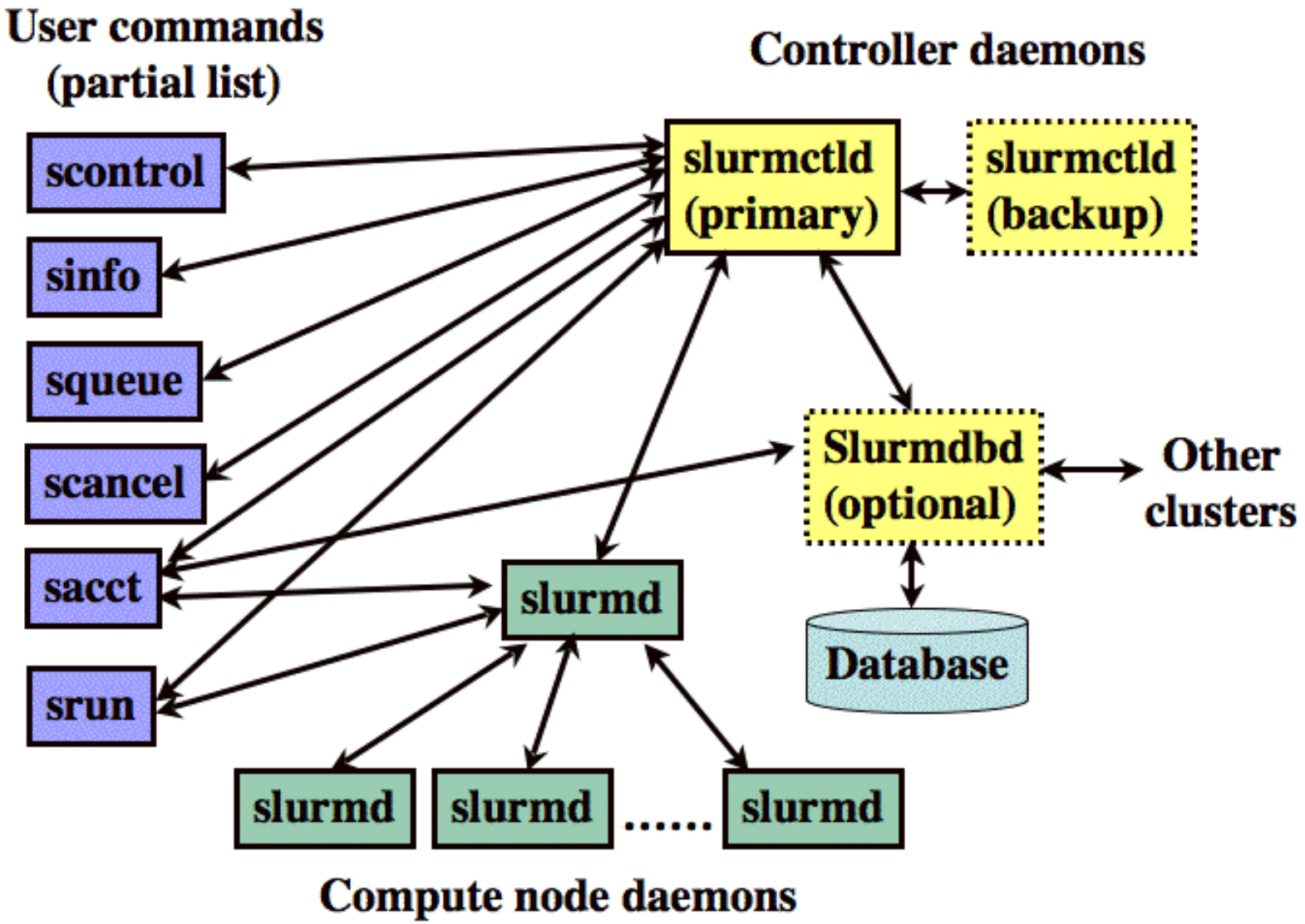


Commandes principales

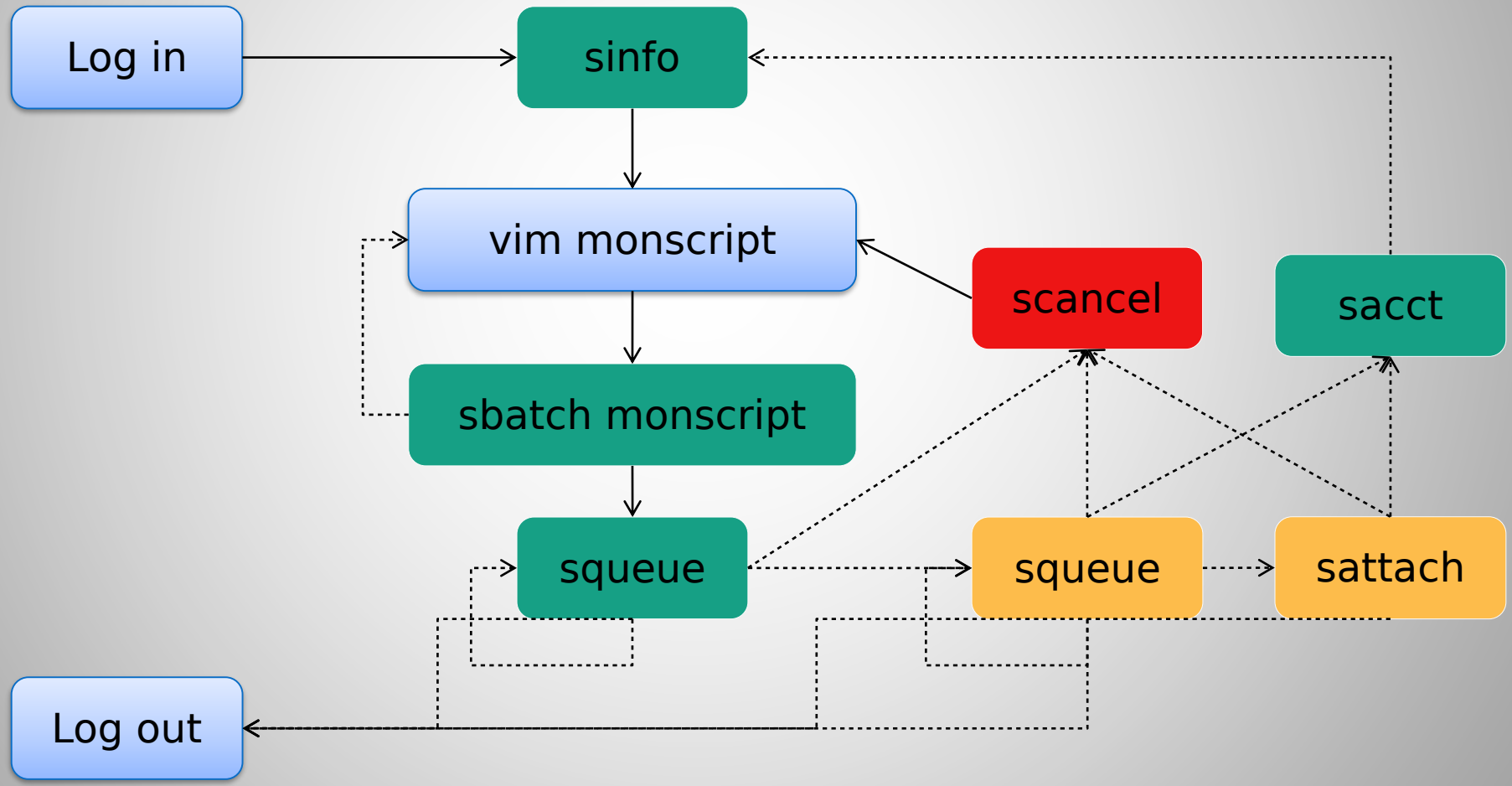
- **scontrol**
  - obtention & modification de la configuration
  - obtention & modification des états des éléments (nodes, partitions, jobs, …)
- **sacctmgr**
  - obtention & modification de la configuration des éléments stockés en BD (« users », « accounts », « qos » …)
- **sinfo**
  - Information sur l'état des partitions
- **squeue**
  - Information sur l'état des « jobs »
- **sacct**
  - Information sur l'exécution de jobs en cours ou passés
- **sstat**
  - Information détaillée sur l'exécution de jobs en cours

- **sbatch**
  - « Soumission » d'une demande d'allocation de ressources *détaillant les ressources nécessaires*
  - Fourniture du « script batch » associé
    - Exécution du script sur les ressources disponibles sur le premier nœud « alloué »
  - Mode « batch » (non interactif)
    - L'utilisateur ne peut plus interagir directement avec son job et doit utiliser les commandes Slurm adhoc pour cela
    - Les sorties stdout/stderr du script exécuté sont redirigés vers des fichiers (configurables)
- **salloc**
  - « Soumission » d'une demande d'allocation de ressources *détaillant les ressources nécessaires*
  - Lancement d'un shell interactif associé aux ressources allouées dès réalisation ou exécution locale d'un script passé en argument
  - Permet l'exécution de commandes « srun » ultérieures pour créer des « jobstep » dans le job réalisé
    - Facilite les tests en évitant l'attente « pending→running » inhérente à chaque soumission
- **srun**
  - « Soumission » d'une demande d'allocation de ressources *détaillant les ressources nécessaires*
  - Exécution d'un certain nombre de processus répartis sur les ressources allouées
    - en fonction des détails fournis en argument
  - Mode d'utilisation interactif (-s)
    - L'utilisateur suit l'exécution du job dans son terminal et peut interagir avec lui (signaux, stdin, ...)

- **sattach**
  - Permet de suivre et/ou d’interagir avec un job batch à la manière d’un job interactif
- **scancel**
  - Permet la transmission d’un signal à un job ou jobstep
  - Permet de demander la terminaison au plus tôt d’un job ou jobstep

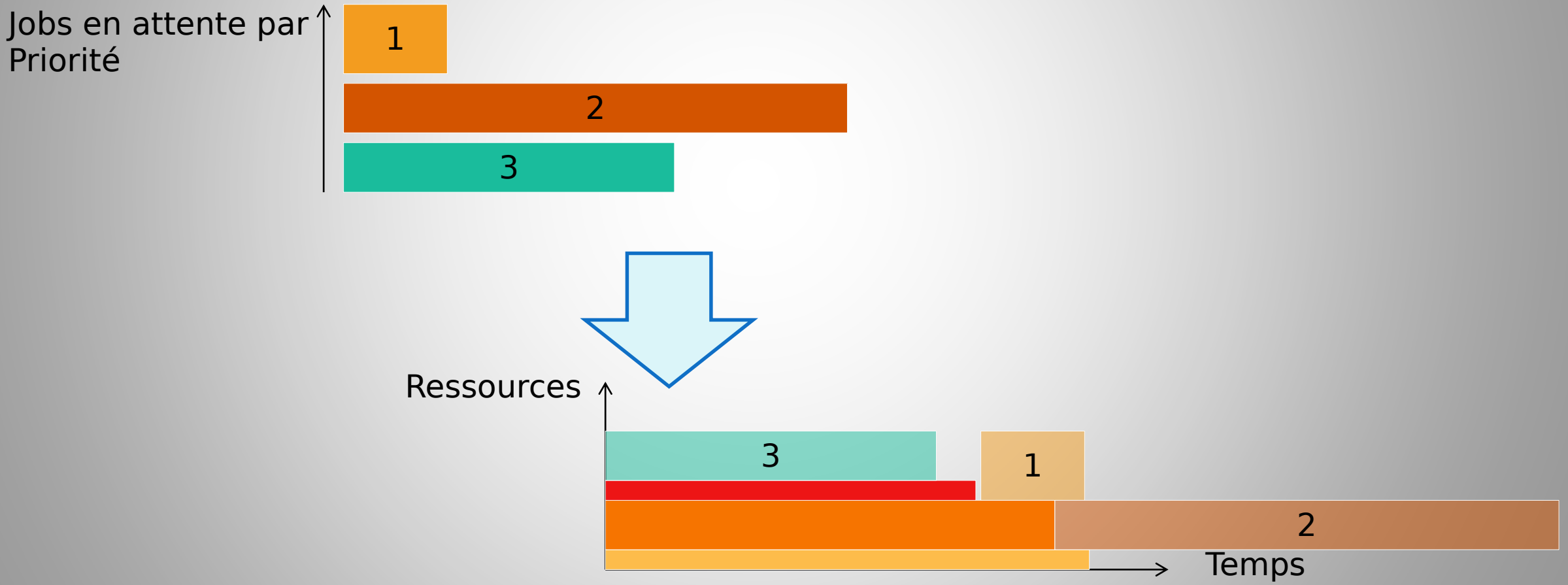

User commands
(partial list)
Controller daemons
scontrol
sinfo
squeue
scancel
sacct
srun
slurmctld
(primary)
slurmctld
(backup)
Slurmdbd
(optional)
Other
clusters
Database
slurmd
slurmd
slurmd
.....
slurmd
Compute node daemons

sbatch


Log in
sinfo
vim monscript
scancel
sacct
sbatch monscript
squeue
squeue
sattach
Log out

Ordonnancement

Comment organiser les priorités ?



Politiques d'ordonnancement classiques

- FIFO : First-In First-Out
  - Premier arrivé, premier servi
- ~~First-Fit~~
  - ~~Le premier qui tient dans l'espace disponible rentre en machine~~
- FairSharing
  - Basé sur une notion de parts de ressources attribués aux différents utilisateurs
  - Celui qui rentre est celui dont l'utilisation est la plus inférieure à ce qu'il est autorisé à utiliser

Politiques d'ordonnancement classiques

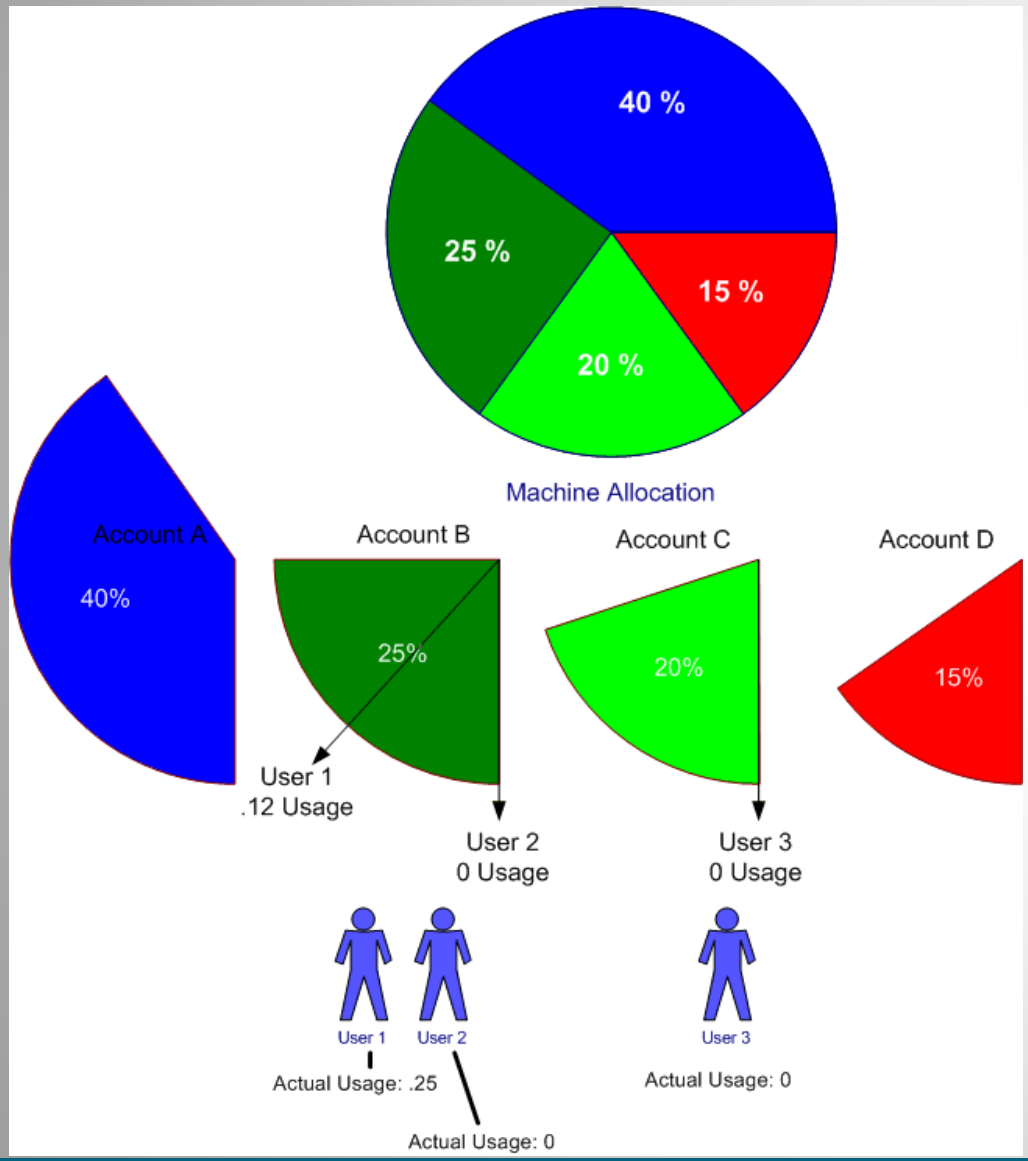
- Aging
  - Le plus ancien est le plus prioritaire
- Size based
  - Le plus petit (ou le plus gros) d'abord
- QOS (Qualité de service)
  - Différentes qualités de service avec différentes restrictions
  - Certaines plus prioritaires que d'autres.
- Backfilling
  - Un job moins prioritaire est exécuté en premier si il ne repousse pas la date de démarrage des plus prioritaires.

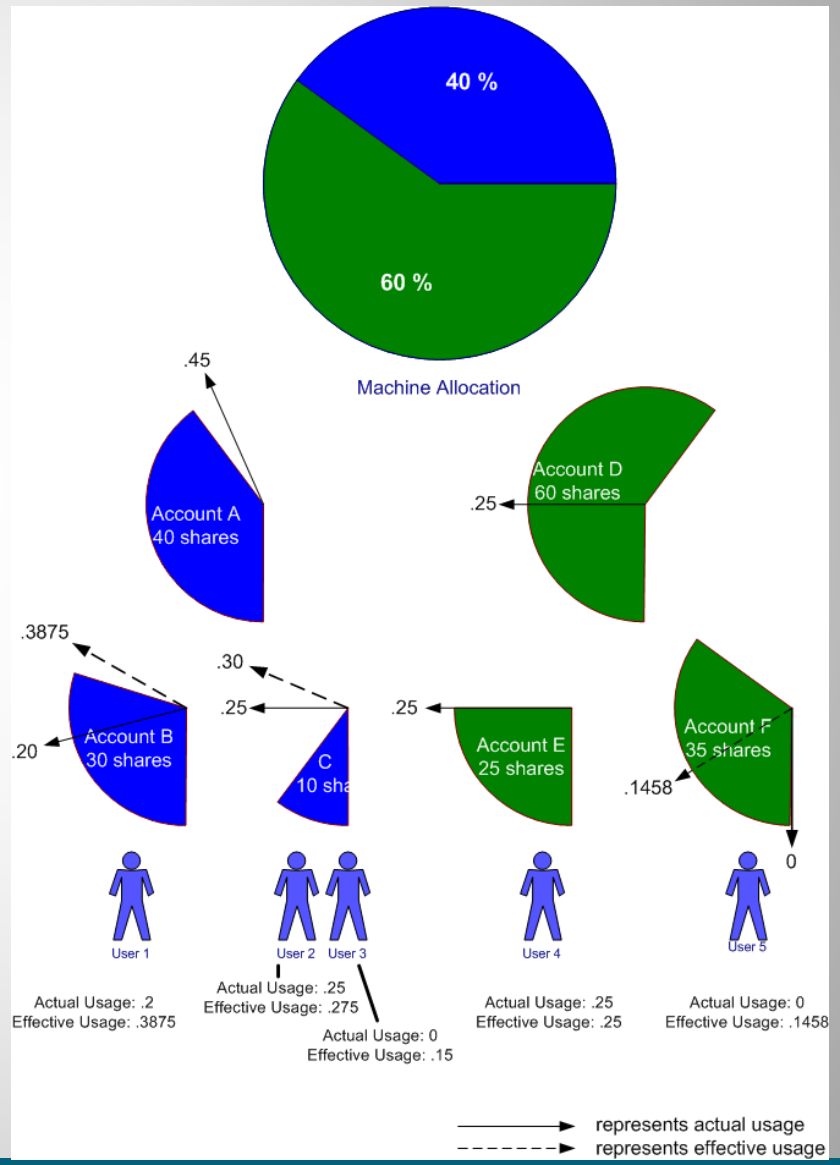
Politiques d'optimisation classique

- Preemption
  - Un job moins prioritaire laisse sa place à un plus prioritaire lorsqu'il doit s'exécuter
    - (suspension d'exécution ou remise en file d'attente (queue))
- Best-effort
  - Un job moins prioritaire est annulé si un plus prioritaire à besoin des ressources

Fairshare

– Groupements hiérarchiques d'utilisateurs, notion d' « account »





FairShare factor

- L’écart entre part utilisable et utilisée des utilisateurs pondère la valeur de chaque job permettant de revenir à l’équilibre souhaité au plus vite
- Ex
  - User-A share=0.3 usage=0.2, fact=0.6
  - User-B share=0.2 usage=0.25, fact=0.45

Partition / QOS Factors

- Chaque partition/qos dispose d'une priorité
- La valeur renvoyée correspond à la normalisation de la valeur de la partition ciblée par rapport à la priorité maximum observée
  - Ex :
    - partition-A priority=20, fact=0.2
    - partition-B priority=100, fact=1.0
    - Partition-C priority=70, fact=0.7

Multifactor priority plugin

- Exemple de configuration

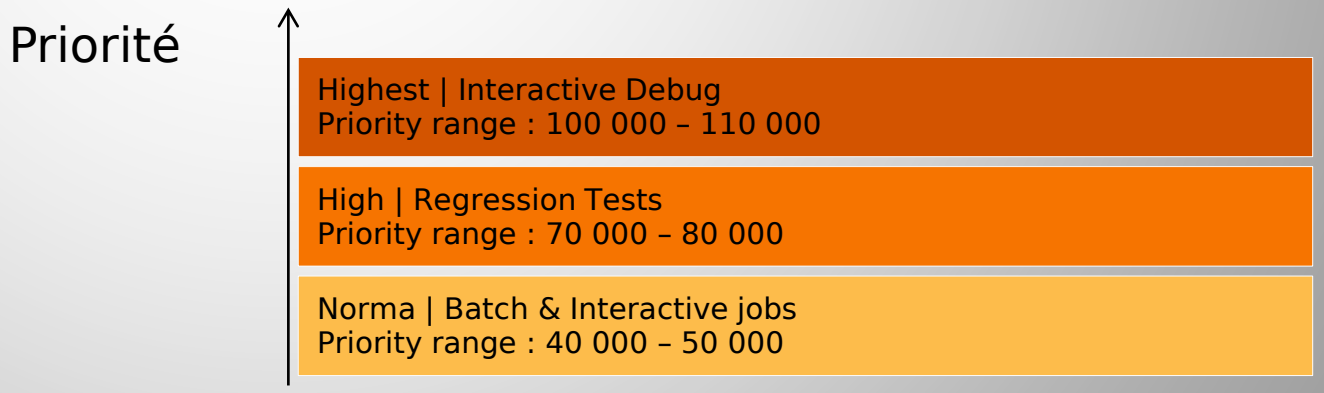
  PriorityWeightQOS=100 000

  PriorityWeightAge=10 000

  PriorityWeightFairshare=10 000

  PriorityWeightJobSize=0

  PriorityWeightPartition=0



Limitation d'accès aux ressources

- Il peut être nécessaire de restreindre l'accès à certaines ressources à certains utilisateurs
- Il peut être nécessaire de restreindre la quantité disponible de ressources pour certains utilisateurs
- Il peut être nécessaire de restreindre le temps d'utilisation maximum possible en fonction des utilisateurs
- Les partitions disposent d'un certain nombre de possibilités de restriction qui ne s'avèrent pas toujours pratiques ou manquent de factorisation
- Les QOS permettent de corriger ce problème en fournissant des restrictions s'appliquant orthogonalement aux partitions
  - Une même partition peut être accédées via différentes QOS
- Les « associations » permettent de raffiner encore la granularité de configuration des limitations
  - Une association peut correspondre à :
    - Un cluster et un account
    - Un cluster, un account et un utilisateur
    - Un cluster, un account, un utilisateur et une partition
  - Les restrictions s'appliquent hiérarchiquement sur les associations d'un utilisateur pour un account donné
    - Un utilisateur peut être associé à plusieurs account

QOS – exemple de limites

- MaxJobsPerUser
  - Quantité maximale de jobs en exécution
- MaxSubmitJobsPerUser
  - Quantité maximale de jobs enregistrés
- MaxNodes
  - Quantité maximale de nœuds utilisables dans un job
- MaxWall
  - Temps d'exécution maximum d'un job
- MaxJobs
  - Quantité maximale de jobs en exécution
- MaxSubmitJobs
  - Quantité maximale de jobs enregistrés
- GrpJobs
  - Quantité maximale de jobs en exécution incluant les jobs de toutes les associations filles
- GrpSubmitJobs : effectif max des jobs en attente

Etude bibliographique

- Depuis 2002 où en sommes nous ?
- 2 articles :
- A Comprehensive Perspective on Pilot-Job Systems
  - https://arxiv.org/abs/1508.04180
- Autopilot: workload autoscaling at Google
  - EuroSys '20: Proceedings of the Fifteenth European Conference on Computer Systems April 2020,
  - https://doi.org/10.1145/3342195.3387524
- Article : autopilot_google_2020.pdf
  - ➔Quelle est la limite à SLURM (par rapport à la disponibilité)
  - ➔Quelle contrainte introduit Autopilot sur le code ?

Prise en main de SLURM

- //swarm init ➔ déjà fait dans les TD précédents
- ➔Docker créé un contexte préfixé par le nom du répertoire…
- Le dossier qui contient le fichier docker-compose porte le nom GLCS-CM5-2024 ➔ afin de rester dans le contexte du CM5 et d'avoir vscode
- Explications sur l'image Docker ici :
  - https://github.com/jmbatto/slurm-docker-cluster
- installation de
  - 1 noeud slurmdbd,
  - 1 noeud de contrôle,
  - 2 nœuds de calcul, le tout sous MPI
- `docker compose up -d // interactions avec CM5`
- Ouvrir 2 shells (un pour le nœud de contrôle, un pour le nœud C1)

Quelques commandes SLURM

sinfo	interrogation des files d'attente
sbatch	soumission d'un job dans une file d'attente (appelées partitions dans SLURM)
salloc	réservation de ressources en interactif
srun	crée une allocation de ressources, à utiliser avec sbatch ou salloc run parallel jobs
scancel	suppression d'un job
squeue	liste des jobs dans les files d'attente
sprio	prorités relatives des jobs en attente
scontrol	affiche/modifie des données relatives aux tâches : jobs, nodes, partitions, reservations, etc.
sacct	affiche les données des jobs
sacctmgr	affiche et modifie les informations des comptes Slurm
sattach	s'attacher à une étape de travail en cours
sdiag	afficher les statistiques d'ordonnancement et les paramètres de synchronisation
sreport	rapports à partir des données de comptabilisation des travaux et des statistiques d'utilisation
sshare	afficher les parts et l'utilisation pour chaque compte de charge et chaque utilisateur
sstat	afficher les statistiques d'un travail ou d'une étape en cours d'exécution
sbcast	transmettre un fichier aux nœuds alloués à un travail Slurm.
scrontab	gestion de la crontab associée à slurm

TD élémentaire

3 étapes (step0/1/2) de découvertes par rapport aux contraintes (pas d'aspect MPI)

On travaille dans le conteneur slurmctld

- on vérifie s'il y a une partition pour root

```
sacctmgr show association -p user=root
scontrol show partition
scontrol show nodes
sinfo -Nel
```

- Sinon

```
sacctmgr --immediate add cluster name=linux
```

  - Puis un restart des images

- Dans le répertoire /usr/local/var/mpishare faire
- git clone http://gogs.eldarsoft.com/jmbatto/glcs slurm.git

```
mpicc -g3 -o elementary elementary.c
```

Etapes du TD

- Explorer les répertoires step0, step1, step2

`salloc -n 2 mpirun ./elementary` ➔ attention il faut que le binaire soit visible des nœuds (pb de partage du binaire – modifier les scripts…)

`sbatch script0.sh`

`sacct -j %jobid obtenu au moment du sbatch%`

- Dans le répertoire step0, modifier le script pour avoir un code retour différent de 0
- Vérifier le résultat du batch

`sbatch -n 2 xxx.sh` //(selon le répertoire)

`squeue -s -j %jobid%`

`sacct -j%jobid%`

`squeue -s -i 30 -j %jobid%`

`sacct -j %jobid%`

- sbatch
  - -N, --nodes=‹minnodes›[-maxnodes]|‹size_string›
  - Request that a minimum of minnodes nodes be allocated to this job.
  - -n, --ntasks=‹number›
  - sbatch does not launch tasks, it requests an allocation of resources and submits a batch script. This option advises the Slurm controller that job steps run within the allocation will launch a maximum of number tasks and to provide for sufficient resources. The default is one task per node, but note that the --cpus-per-task option will change this default.

```
sbatch –n2 –N2 –exclusive xxx.sh
squeue -O jobid,state,qos,timeused
```

Idée du TD

- SLURM « envoie » un signal SIGTERM avant la limite du temps du batch
- On veut observer la fin du batch ➔ raison du répertoire step2

```
sacctmgr modify qos normal set MaxWall=00:00:02
```