« Jobs Batch Scheduler »

SLURM

Matthieu Hautreux – matthieu.hautreux@cea.fr
Régine Gaudin-Haugeard - regine.gaudin@cea.fr

Plan

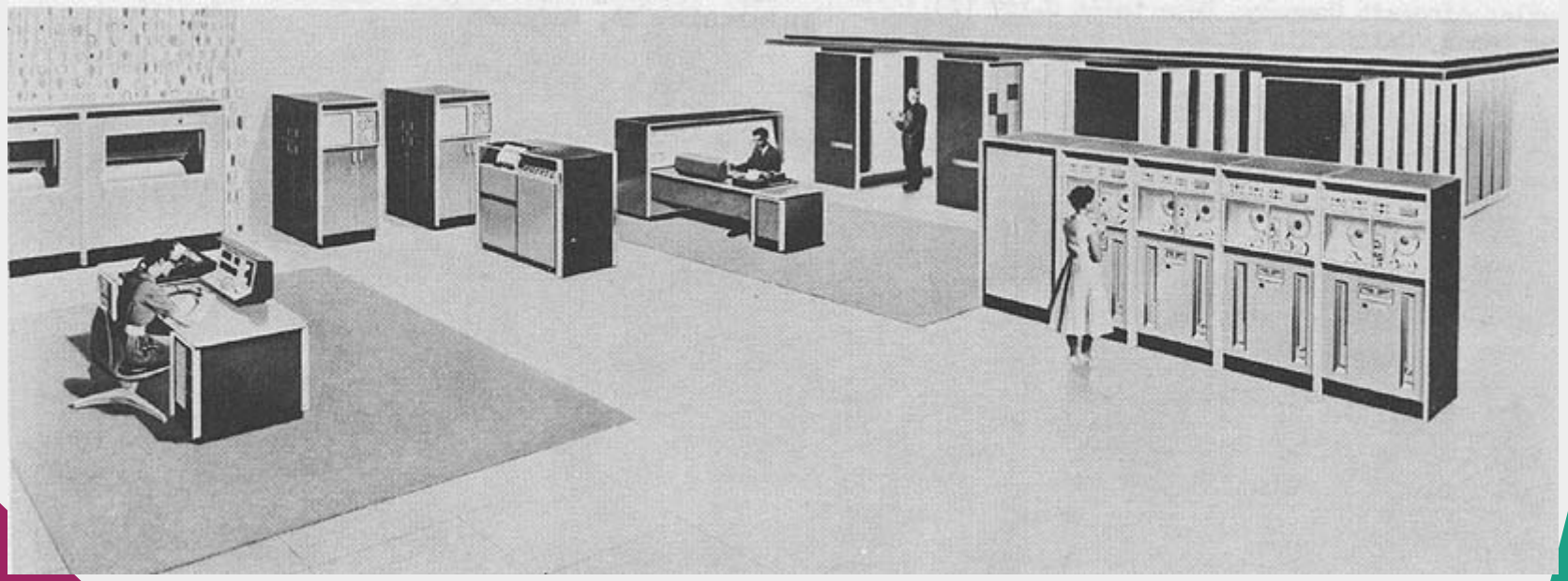
- Un peu d’histoire
- Principe de DRMS
- Cas de SLURM
- Exemple du TGCC ?

Un peu d’histoire

- Premières générations d’ordinateurs

  (50’s - 60s)

- Utilisées pour le calcul scientifique et pour l’automatisation des actions administratives des grandes compagnies



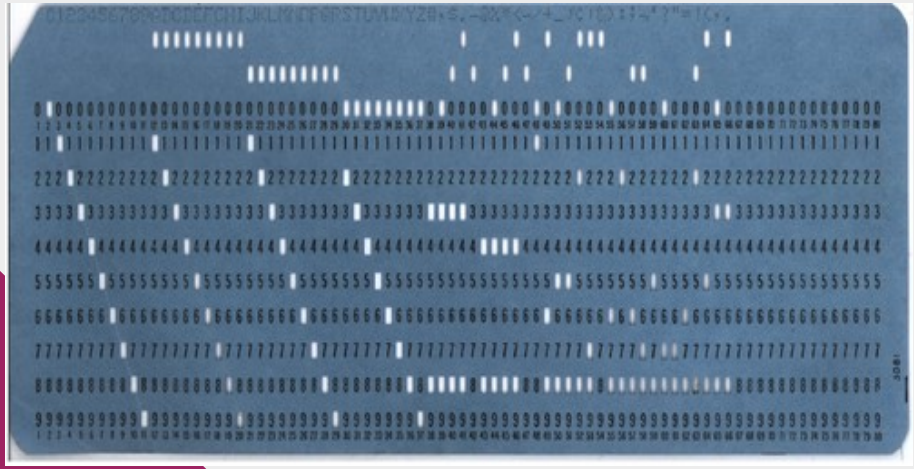
Un peu d’histoire

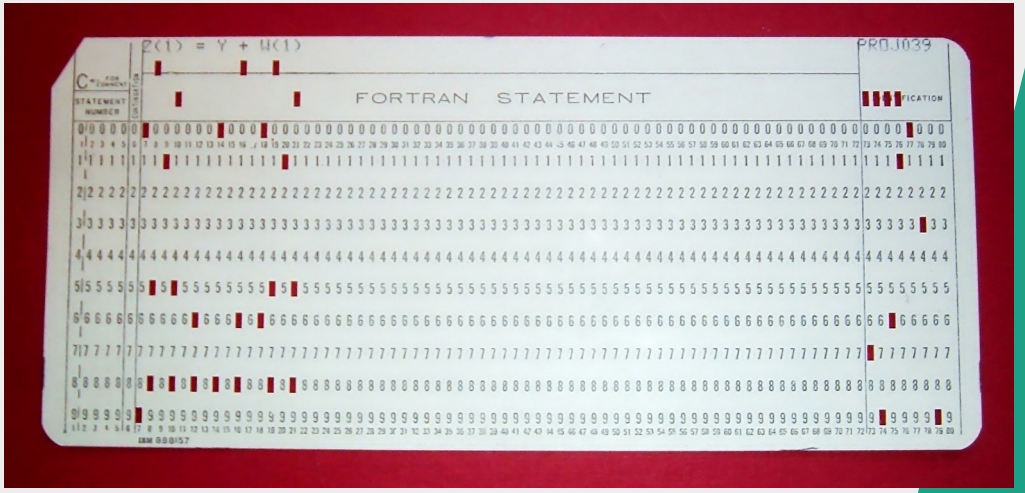
- Premières générations d’ordinateurs

- Les accès interactifs sont inexistants pour les « utilisateurs » de ces machines.

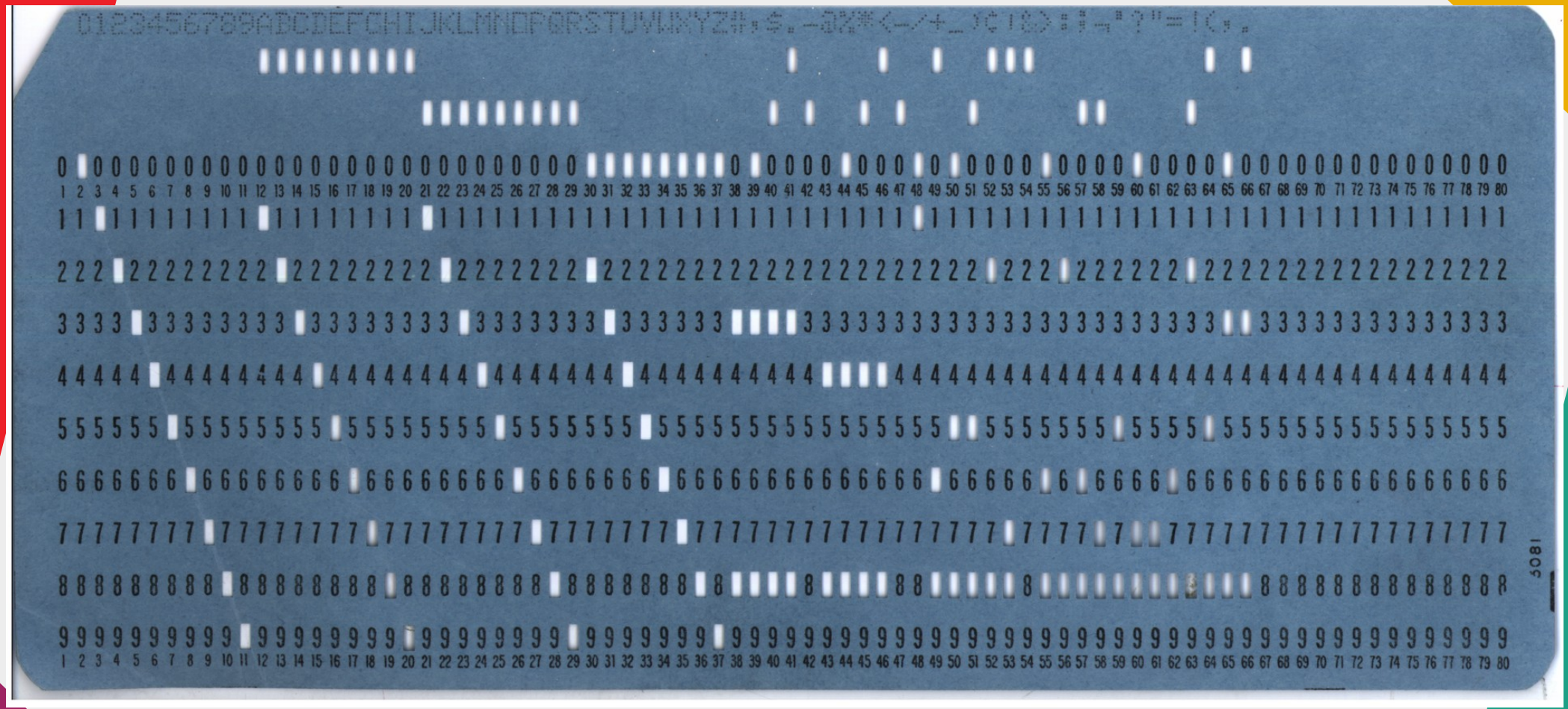
- Les cartes sont leur seul mode d’interaction

-
  - Un coin des cartes est d’ailleurs tronqué pour en simplifier l’organisation et le rangement dans des boites adaptées.



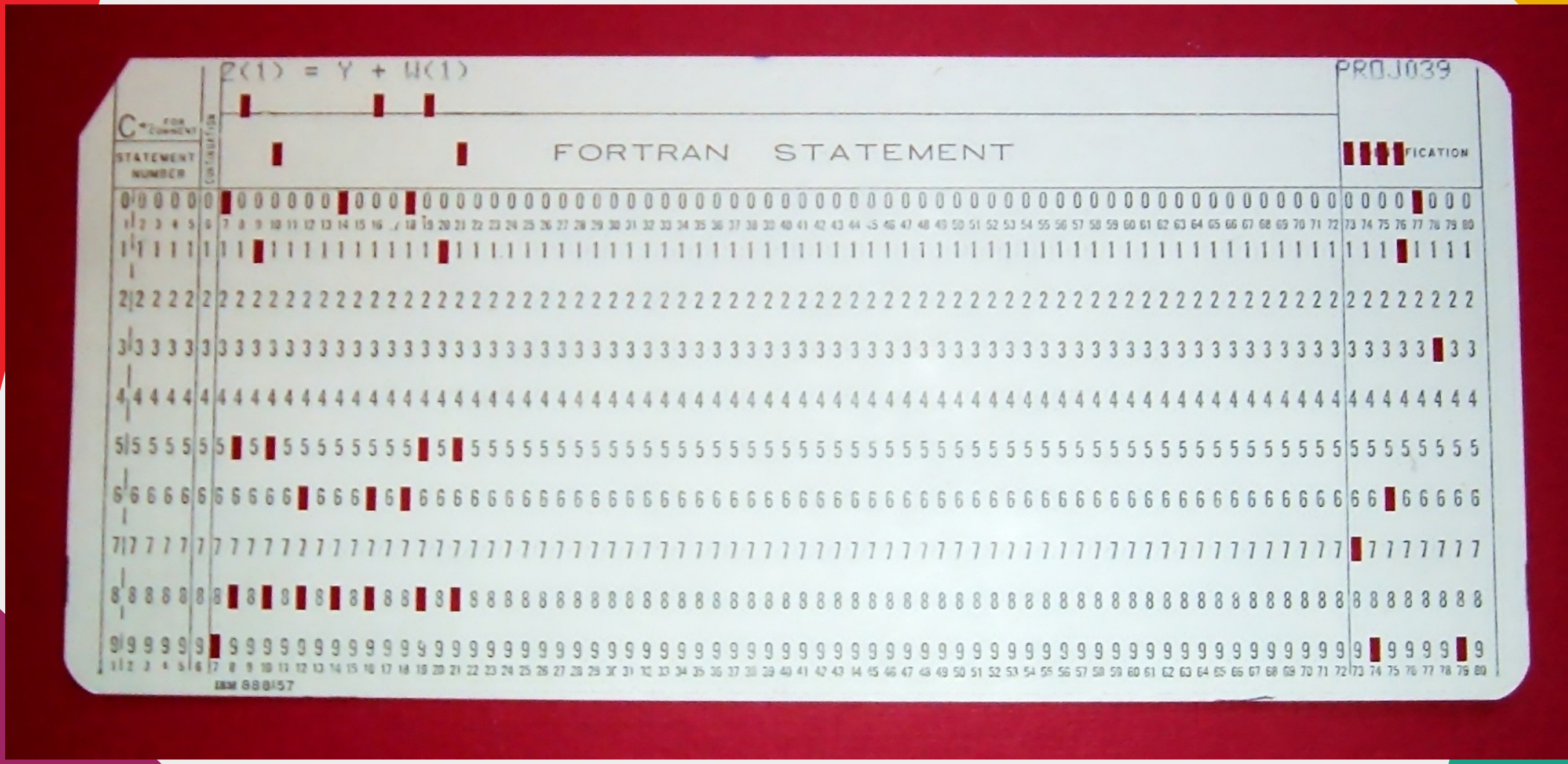


Un peu d’histoire



Codage EBCDIC

Un peu d’histoire



Un peu d’histoire

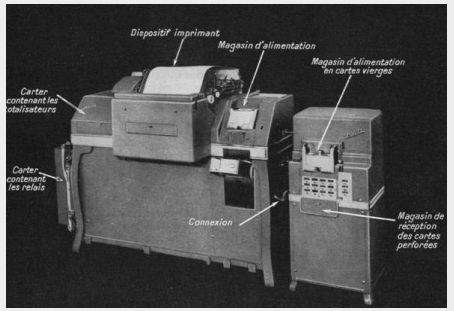
- Premières générations d’ordinateurs

- La programmation est faite à base de cartes perforées, regroupées par lot. Procédure longue.

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.



Un peu d’histoire

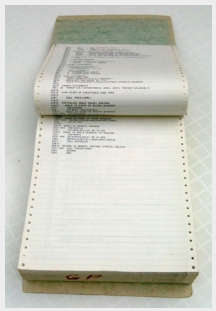
- Premières générations d’ordinateurs

-L’utilisation des machines est assurée par des opérateurs qui chargent les paquets de cartes en machine suivant un planning pré-établi.

« batch scheduling ou planification par lots »

- Les résultats, des « listings » papiers, sont fournis aux utilisateurs après l’exécution de leurs « jobs ». Leur analyse peut être longue.

Les « jobs » en erreur produisent une quantité énorme de sorties ...



Un peu d’histoire

- L’ère « mainframe » ou unité centrale (70’s)

– Les « mainframes » ou unités centrales apparaissent
- Les terminaux « graphiques » font leur entrée.
- Des lecteurs de cartes restent associés...
Pour réutiliser les codes sur cartes et migrer vers des « scripts »

Un peu d’histoire

- L’ère « mainframe » ou unité centrale (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.

Apparaissent les premiers planificateurs de lots de tâches « batch scheduler »...

Un peu d’histoire

- L’ère « mainframe » (70’s)

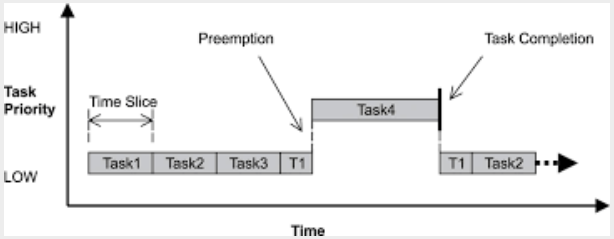
- Le batch scheduling est toujours présent

Notion de **time slicing** (partage de temps)

Actions automatiques en tâches de fond et pendant les périodes d’inactivité (nuit/weekend)

- Des politiques d’ordonnancement doivent être mises en place

Pour automatiser l’exécution des tâches en fonction de leurs priorités



Un peu d’histoire

- L’ère « mainframe » (70’s)

- Soumission de scripts « batch » dits « jobs » par les utilisateurs

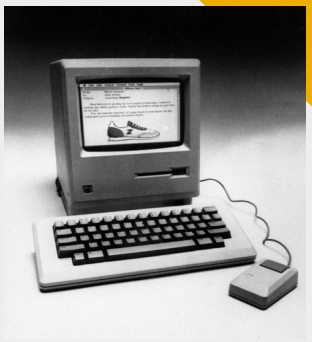
- Ordonnancement automatique de l’exécution des jobs par une application dédiée.

- Résultats par « listings » numérisés : sorties « écran » redirigées dans des fichiers.

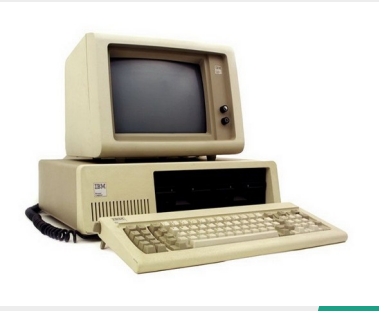
Gain de temps dans la mise au point des programmes, l'éxécution et le dépouillement des résultats.

Un peu d’histoire

- L’ère « PC » (80’s – 00’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 ».

- Les systèmes d’exploitation permettent

une interaction directe via des interfaces graphiques simplifiant l’utilisation des machines

Un peu d’histoire

- L’émergence des clusters (90s)

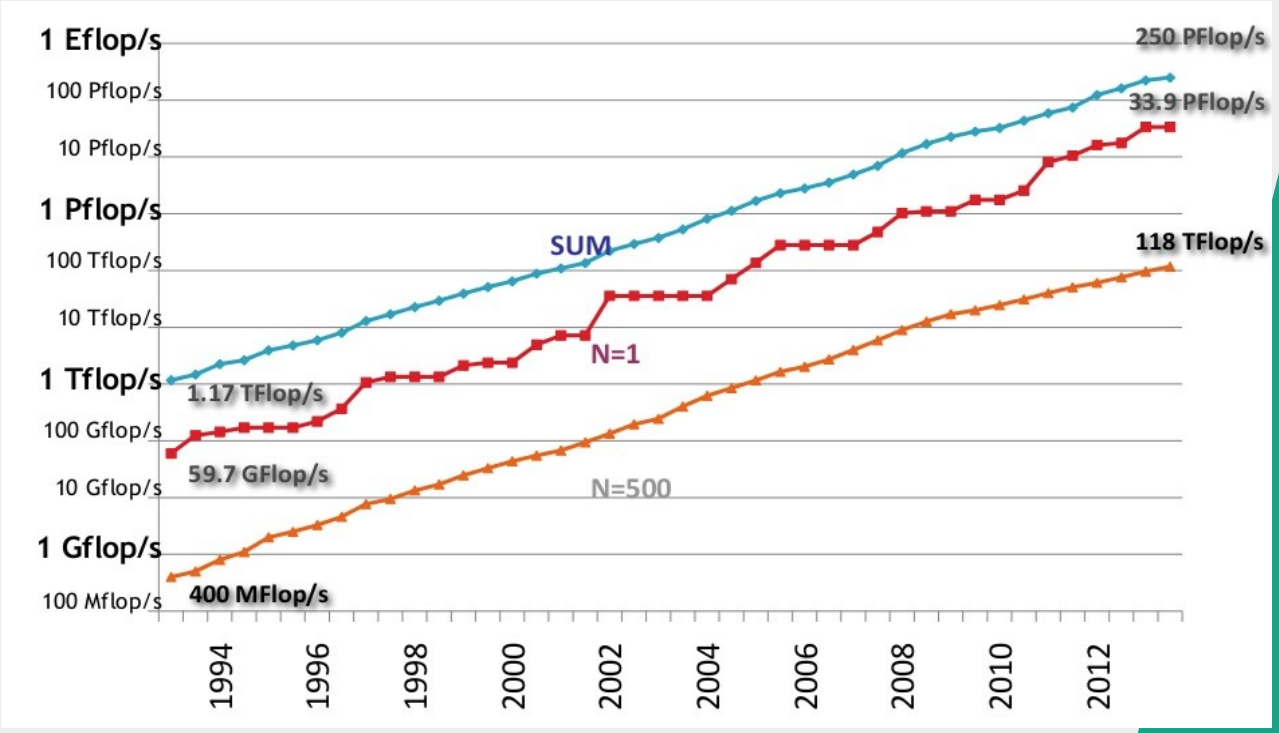
- Les réseaux prennent de l’ampleur et permettent une interconnexion performante d’unités individuelles type PC.

- Le HPC s’engouffre dans cette voie face à la diminution des performances des approches monolithiques des « Mainframe ».

Un peu d'histoire

- L'émergence des clusters (90s)

Le nombre d'unités de calcul connectées ne cessera de croître... https://www.top500.org/



Un peu d’histoire

- L’émergence des clusters (90s)

- L’utilisation des clusters nécessite alors l’orchestration de plusieurs unités de calcul indépendantes.

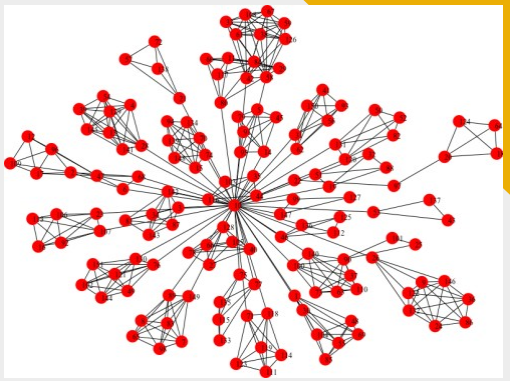
Notion de « **jobs parallèles** » exécutés sur des « systèmes distribués »

- 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 ».

Un peu d’histoire

- L’émergence des clusters (90s)



- Les jobs deviennent hétérogènes

Une ou plusieurs sections parallèles

- Émergence du modèle MPI (Message Passing Interface)

Echanges de données entre les processus

- L’ordonnancement se complexifie

Différents besoins en terme de nombres d’unités de calcul dans les sections parallèles.

Différentes localités.

- L’émergence des clusters (90s)

- Les « batch scheduler » évoluent donc pour traiter efficacement ces « systèmes distribués »

On parle maintenant de **DRMS**

(Distributed Resource Management System)

- **Ces gestionnaires de ressources distribuées feront l’objet de la suite de cette présentation.**

HPC Batch Scheduler

Principe des DRMS

DRMS - Architecture

- 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 »

DRMS - Architecture

- 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

DRMS - Architecture

- 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

- Généralement sur un seul nœud

DRMS - Architecture

- Repose généralement sur un ensemble de « **workers** » distribués

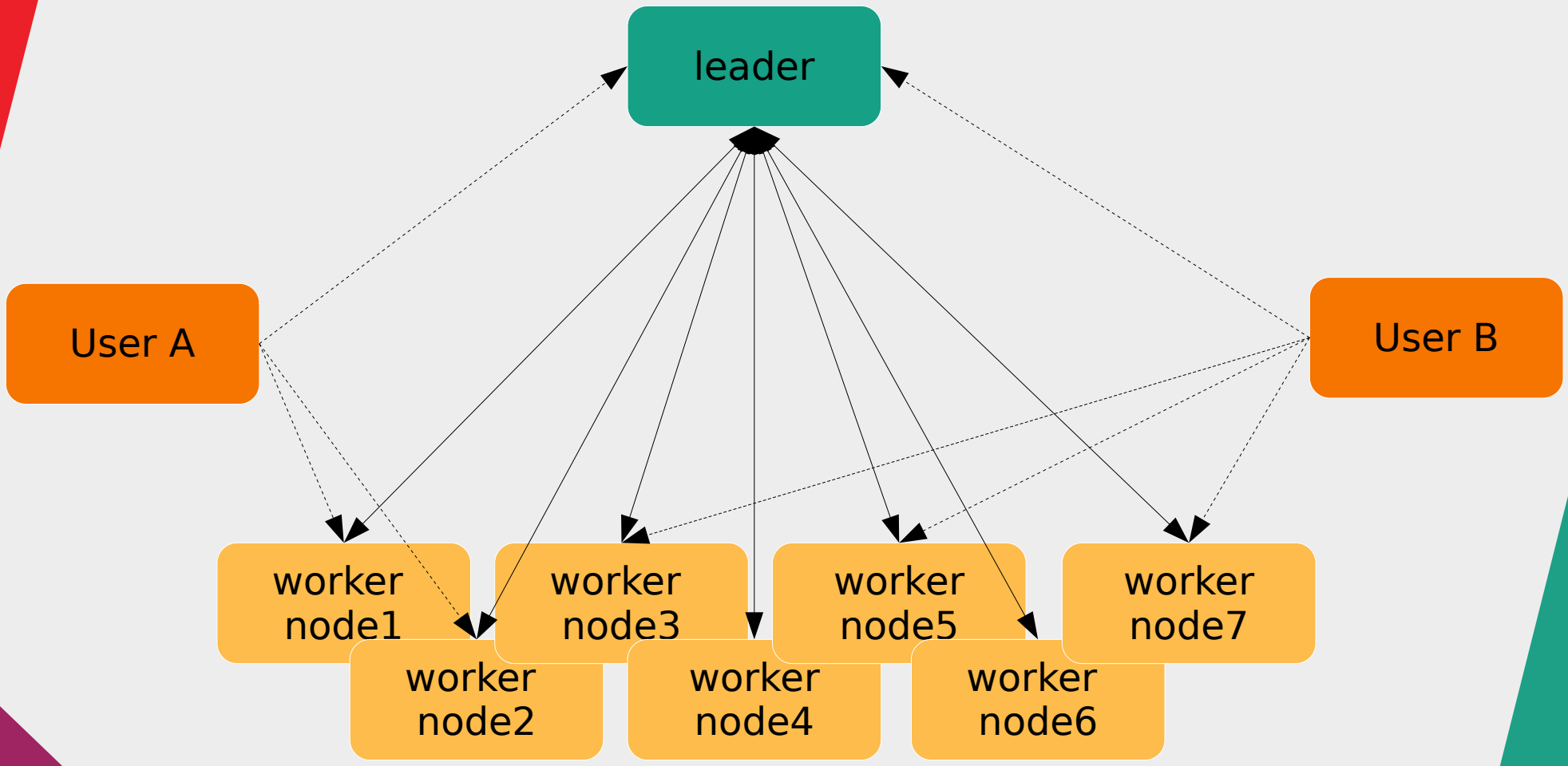
- Généralement un par nœud de calcul

- Fournit l'état du nœud au leader et permet les interactions 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

DRMS - Architecture



SLURM

HPC Batch Scheduler

Principes et utilisation de Slurm

SLURM - Introduction

- **S**imple **L**inux **U**tility for **R**esource **M**anagement

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 de l’époque

https://slurm.schedmd.com/

SLURM - Introduction

- 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

TGCC, CINES, CEA, BSC, NCCS

SLURM - Introduction

- **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

Slurm – Entités élémentaires

- **slurmctld**

Composant « leader » (controleur)

- **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 nécessaire

- **slurmd**

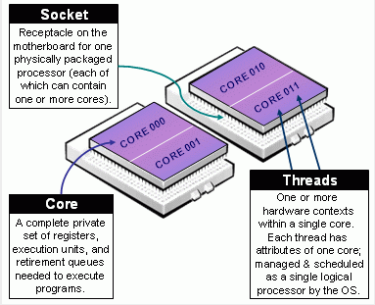
Composant « worker »

Slurm – Notions élémentaires

- **Node**

Unité indépendante fournissant des ressources utilisables par les utilisateurs

Sockets/Cores/Threads, Memory, GPUs, …

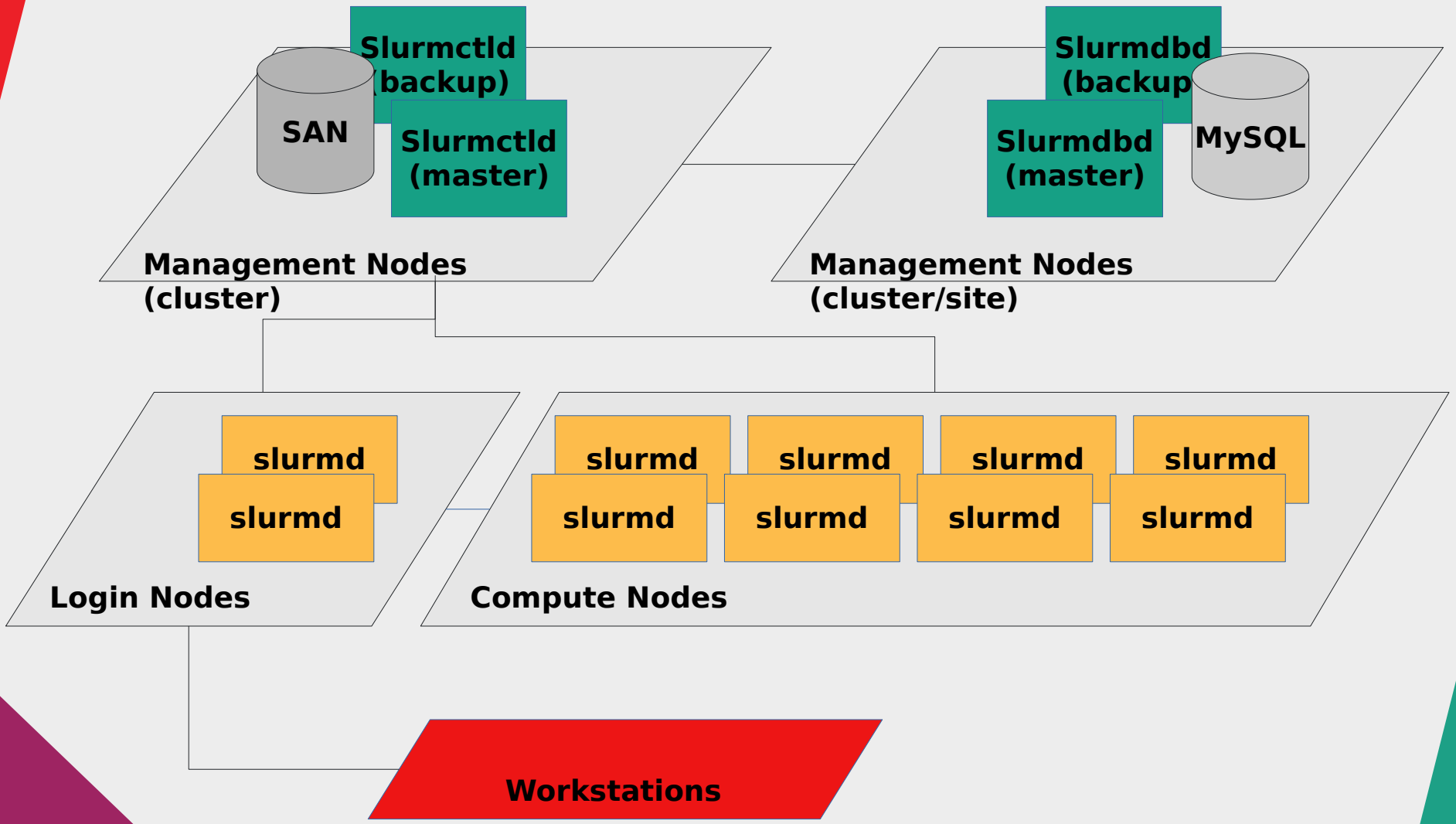


- **Partition**

« Pool » de nœuds utilisables au sein d’un même « job »

Un nœud peut appartenir à plusieurs partitions

Slurm – Entités élémentaires



Slurm – Notions élémentaires

- **Job**

Demande d’allocation de ressources *dans une partition* associée à un utilisateur

Ensemble de ressources réparties sur des nœuds pour un temps défini

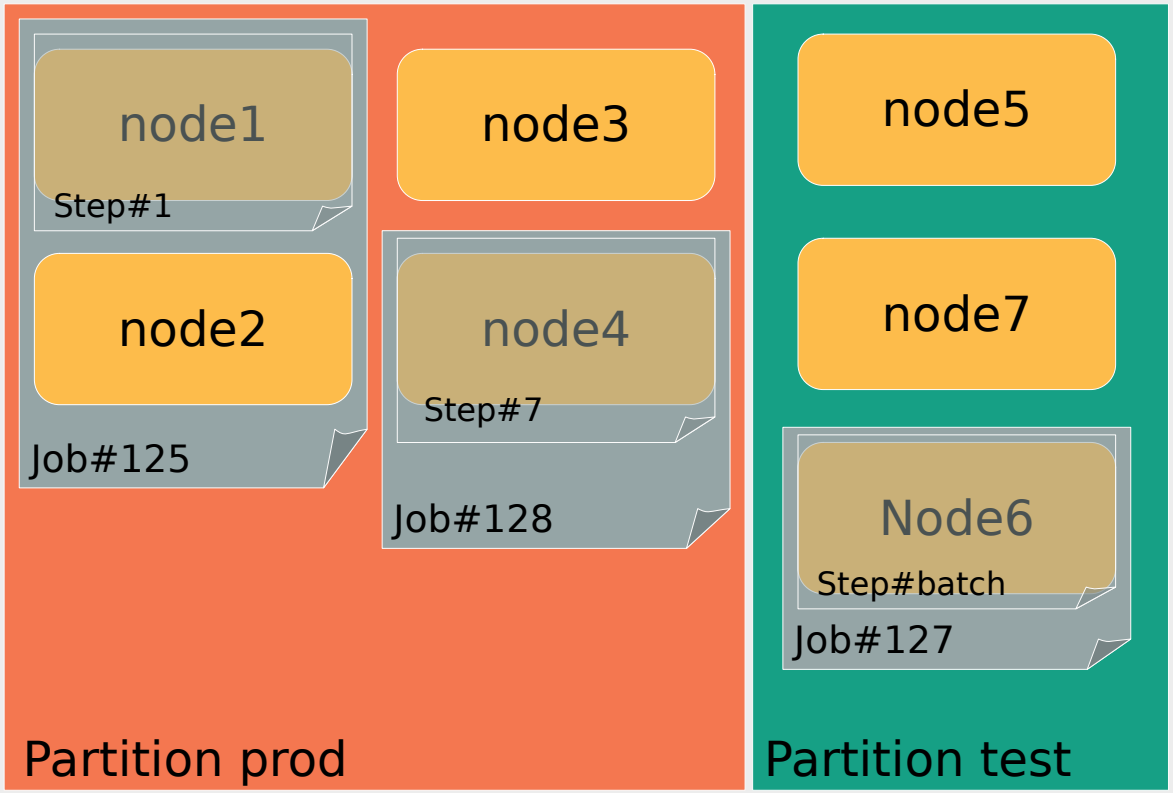
Batch (script fourni) ou Interactif (shell)

- **Jobstep**

Demande de sous-allocation de ressources pour effectuer une tâche particulière

Sous-ensemble de ressources parmi les ressources allouées pour le job associé

Slurm – Notions élémentaires

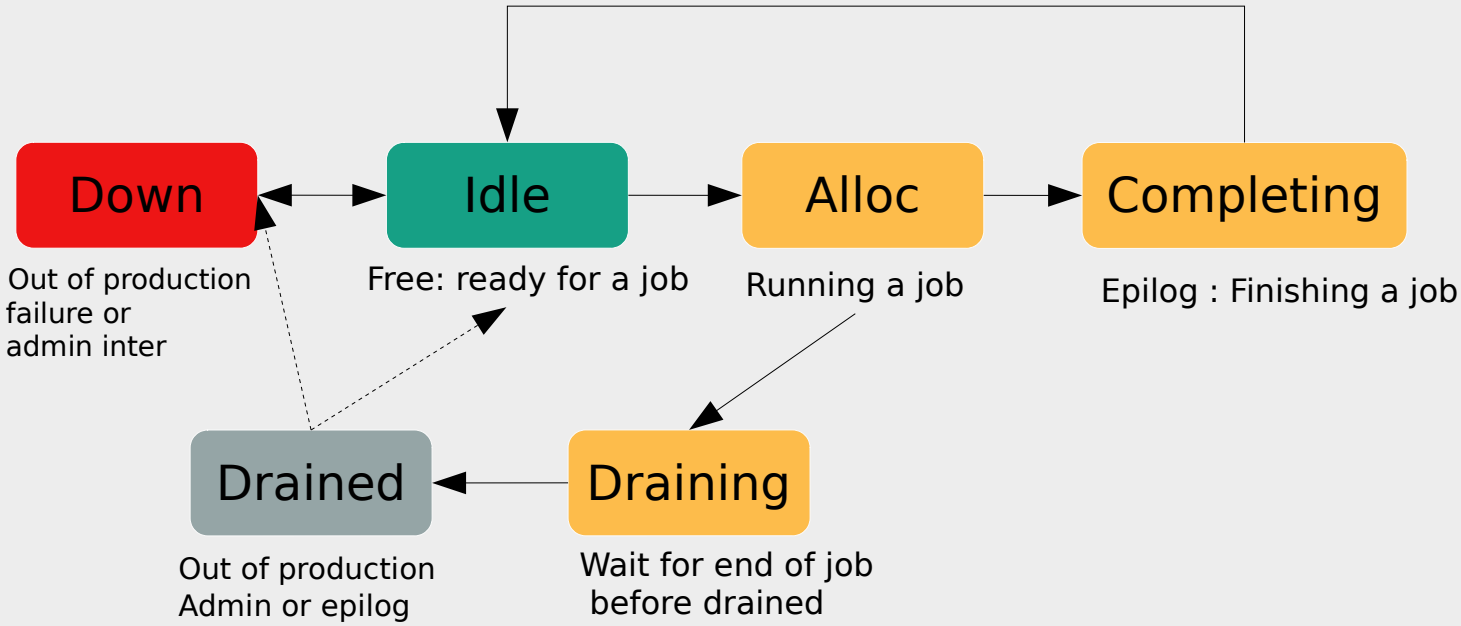


Slurm – Notions élémentaires

- **Node « states » :**

sinfo

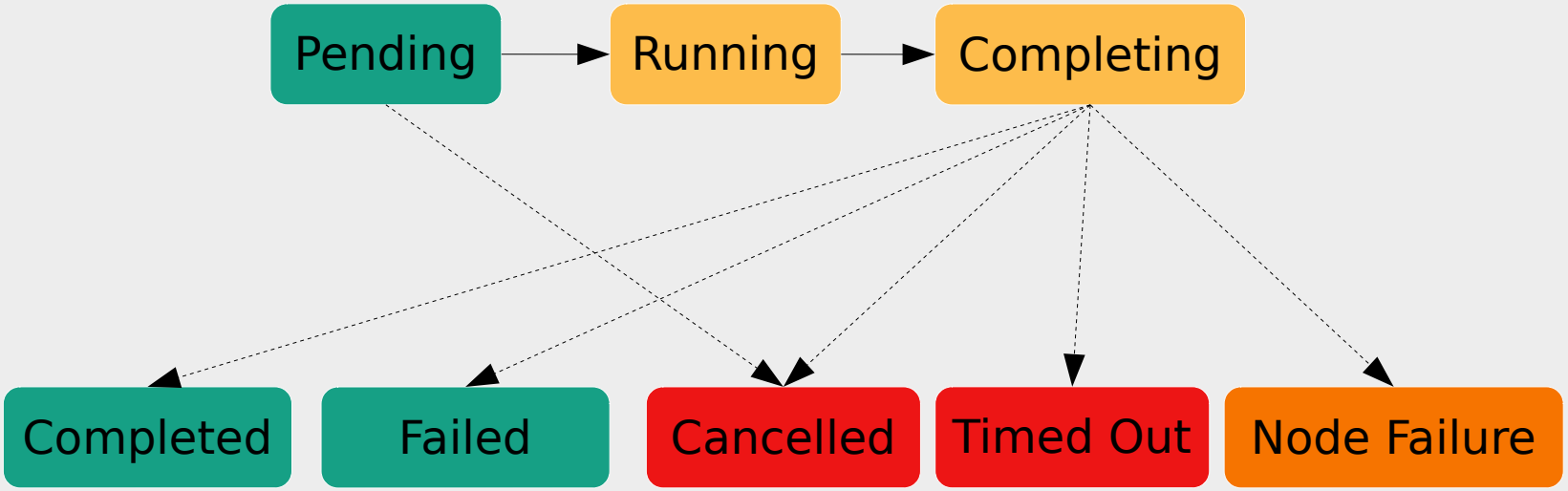
scontrol update state=XXX node=XXX



Slurm – Notions élémentaires

- **Job « states »**

squeue



Slurm – Notions élémentaires

- **Commandes principales de configuration**

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 » ...)

Slurm – Notions élémentaires

- **Commandes principales d’informations**

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

Slurm – Notions élémentaires

- **Commandes intéractives de lancement de « jobs »**

srun

- Soumission d’une demande d’allocation de ressources
- Exécution d’un certain nombre de processus répartis sur les ressources allouées en fonction des détails fournis en argument,

- Libération des ressources à la fin des processus.

$srun -n 1 -p sandy hostname

inti2047

L’utilisateur suit l’exécution du job dans son terminal et peut interagir avec lui (signaux, stdin, ...)

Slurm – Notions élémentaires

- **Commandes interactives de lancement de « jobs »**

salloc (1ère méthode)

- Soumission d’une demande d’allocation de ressources

- Lancement d’un shell interactif associé aux ressources allouées permettant l’exécution de commandes « srun » pour créer des « jobsteps » .

- Libération des ressources par l’utilisateur (ctrl D ou scancel) ou par temps limite de l’intéractif

```
$salloc -n 1 -p sandy
salloc: Granted job allocation 2124281
$srun -n 1 hostname
Inti2047
$ exit
salloc: Relinquishing job allocation 2124281
```

Slurm – Notions élémentaires

- **Commandes intéractives de lancement de « jobs »**

salloc (2ième méthode)

- Soumission d’une demande d’allocation de ressources

- Exécution locale d’un script (contenant srun) passé en argument puis libération des ressources à la fin du script

$salloc -n 1 -p sandy ./script-1.sh

salloc: Granted job allocation 2124277

srun: Step created for job 2124277

inti2047

salloc: Relinquishing job allocation 2124277

Slurm – Notions élémentaires

- **Commandes non interactives de lancement de « jobs » *sbatch***

- Soumission d’une demande d’allocation de ressources

- Exécution du script en argument sur le premier nœud alloué

```
$cat myscript
  #!/bin/sh
  srun hostname
```

Slurm – Notions élémentaires

- **Commandes non interactives de lancement de « jobs »**

sbatch (suite)

$sbatch -n 1 -p sandy myscript
Submitted batch job 212428

L’utilisateur ne peut interagir avec le job sauf avec les commandes slurm :
sinfo, scancel, scontrol, sattach

Les sorties sdout/sderr du script éxécuté sont redirigées vers des fichiers configurables

Slurm – Notions élémentaires

- **Commandes principales d’interaction avec les « jobs »**

sattach

Permet de suivre et/ou d’interagir avec un job batch à la manière d’un job interactif

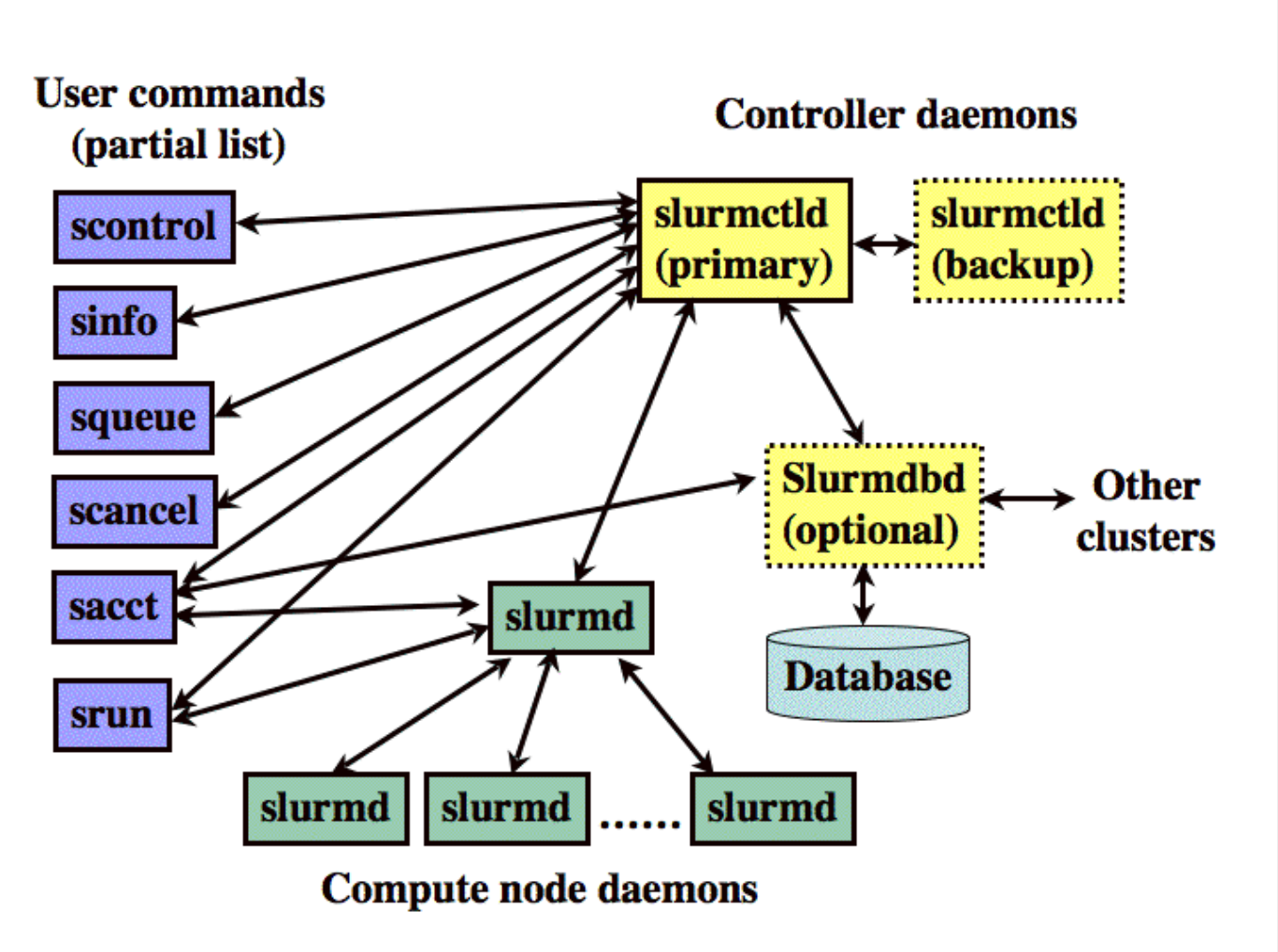
scancel

Permet la transmission d’un signal à un job ou jobstep

Permet de demander la terminaison au plus tôt d’un job ou jobstep

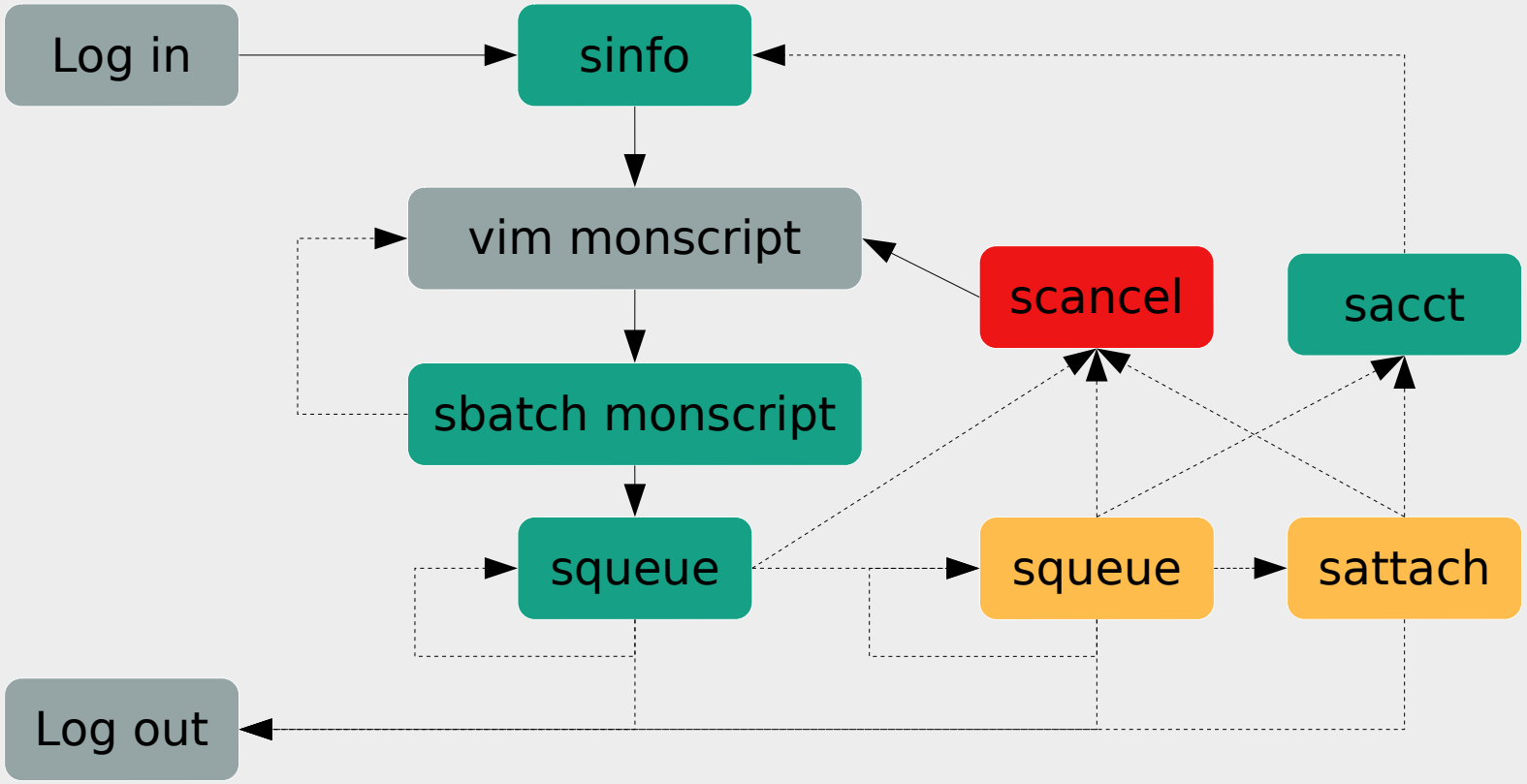
Slurm – Notions élémentaires

- Communication inter démons slurm



Slurm – Notions élémentaires

- **Cas d'utilisation – sbatch**



SLURM – Ordonnancement des jobs

- **Principes utilisés dans Slurm**

  Priority-based scheduling

  Quelque soit la politique d’ordonnancement choisie, celle-ci doit se traduire par une priorité numérique.

Slurm – Ordonnancement des jobs

- **Différents types d’ordonnanceur**

FIFO : First-In First-Out

Jobs soumis par ordre de rentrée

Backfilling

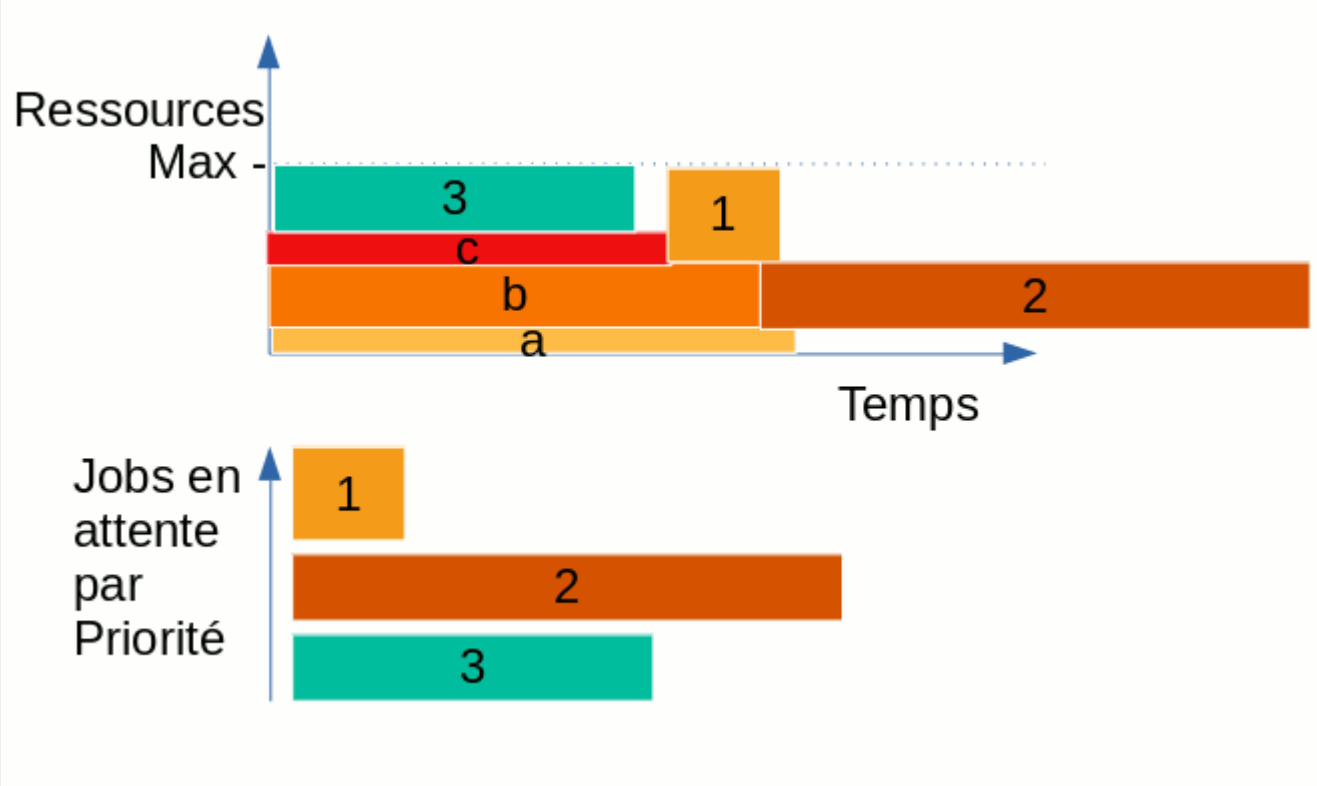
Un job moins prioritaire est exécuté en premier s’ il ne repousse pas la date de démarrage des plus prioritaires.

SLURM - Ordonnancement

- **Différents types d'ordonnanceur**

Backfilling (description)

Un job moins prioritaire est exécuté en premier s'il ne repousse pas la date de démarrage des plus prioritaires.



SLURM - Ordonnancement

- **Politiques de calcul des priorités**

  Size based

  Le plus petit (ou le plus gros) d’abord

  QOS (abordé plus loin)

  Différentes qualités de service avec différentes restrictions.Certaines plus prioritaires que d’autres.

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

FAIRSHARE

Basé sur une notion de parts de ressources attribuées aux différents utilisateurs

Celui qui est le plus prioritaire est celui dont l'utilisation est la plus inférieure à ce qu'il est autorisé à utiliser.

AGING

Le plus ancien est le plus prioritaire

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

Age Factor

Permet de fournir une indication sur le temps d’attente du job

afin de favoriser les plus anciens et éviter les famines (« starvation »)

Job size Factor

Permet de fournir une indication sur la quantité de ressources demandée

Afin de favoriser les plus gros ou les plus petits jobs

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

  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, factor=0.2

partition-B priority=100, factor=1.0

Partition-C priority=70, factor=0.7

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

  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 (factor >0.5 augmente la priorité, < 0.5 baisse la priorité)

Ex

User-A share=0.3 usage=0.2, factor=0.6

User-B share=0.2 usage=0.25, factor=0.45

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

Multifactor priority plugin

Permet de mélanger différentes politiques dans le calcul de la priorité d'un job avec un facteur de pondération :

Factors : Age, Job Size, Partition, QOS, FairShare

Chaque facteur → valeur entre 0 et 1 par job

Somme globale pondérée = priorité du job

SLURM - Ordonnancement

- **Politiques de calcul des priorités**

  Multifactor priority plugin

  Exemple de configuration

  PriorityType=priority/multifactor

  PriorityWeightQOS=100 000
  PriorityWeightAge=10 000

  PriorityWeightFairshare=10 000

  PriorityWeightJobSize=0

  PriorityWeightPartition=0

SLURM - Ordonnancement

- **QOS (Quality Of Service)**

Limitation de l’utilisation des ressources/groupe d’utilisateurs (« account »)

Ou ressources/utilisateurs

Exemples :

Restriction accès ressources/utilisateurs

Restriction quantité de ressources/utilisateurs

Restriction temps d’utilisation/utilisateurs

SLURM - Ordonnancement

- **QOS (Quality Of Service)**

Restriction sur les partitions manque de factorisation

QOS permettent de corriger ce problème avec des restrictions plus fines s’appliquant orthogonalement aux partitions

Une même partition peut être accédée via différentes QOS

Slurm – Ordonnancement

- **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

… nombreux paramètres *man sacctmgr*

Slurm – Ordonnancement

- **QOS – exemple de création**

#sacctmgr create qos debug set priority 100 MaxJobsPU=2 MaxWall=01:00:00

#sacctmgr create qos normal set priority 50 MaxJobsPU=10 MaxWall=08:00:00

#sacctmgr show qos

Name	Priority	MaxJobPu	MaxWall
Debug	100	2	01:00:00
Normal	50	10	08:00:00

SLURM - Ordonnancement

- **QOS – Limitation d’accès aux ressources par associations**

Une association peut correspondre à :

Un « cluster », un « account », qos

Un « cluster », un « account », un utilisateur, qos

Les restrictions s’appliquent hiérarchiquement sur les associations d’un utilisateur pour un « account » donné

Un utilisateur peut être associé à plusieurs comptes

SLURM - Ordonnancement

- **Association – exemple de limites**

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 ...

man sacctmgr

SLURM - Ordonnancement

- **Association – exemple (man sacctmgr)**

#sacctmgr add user gaudinr account=root qos=normal

#sacctmgr modify user gaudinr set qos=test,debug

#sacctmgr show assoc

Cluster account User QOS

inti root gaudinr normal,test,debug

Inti ocre dupond normal

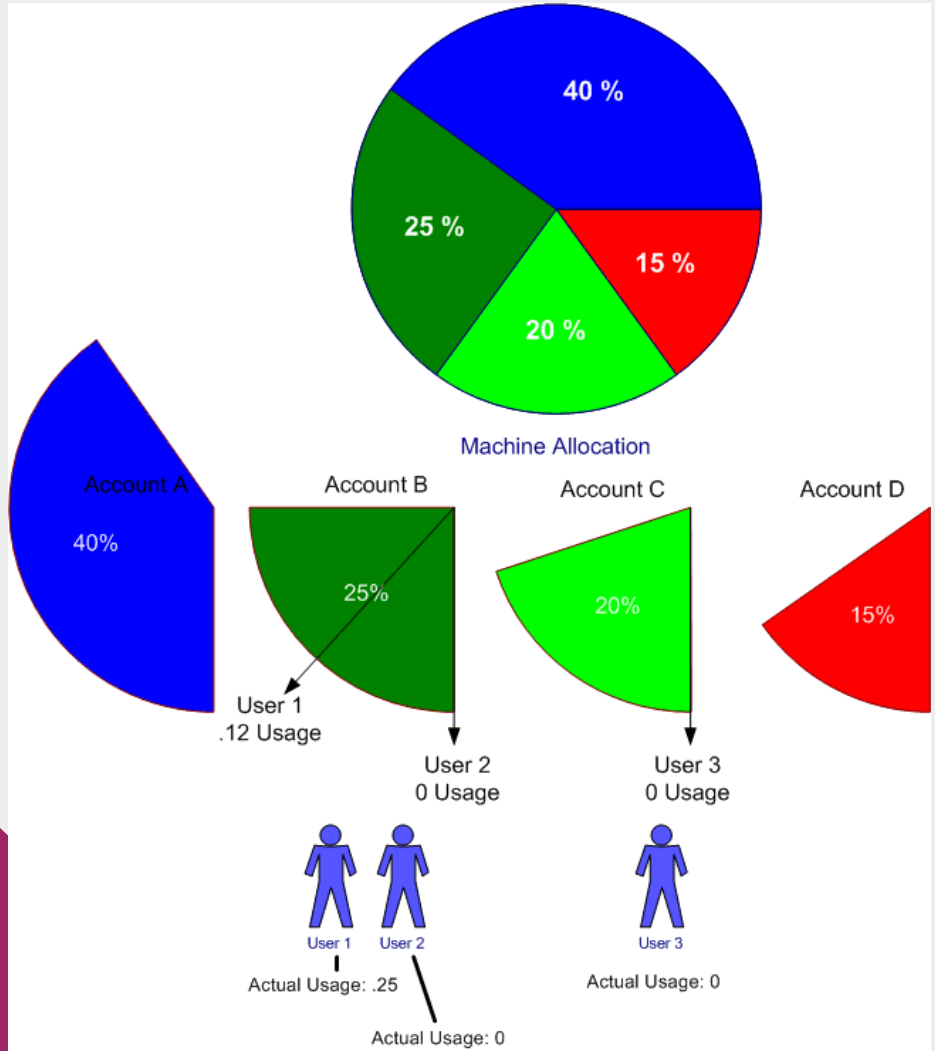
$srun -n 1 –-qos=debug -p vm ~gaudinr/myprogdebug

$sacct -o JobID,JobName,user,qos
61 myprogdebug gaudinr debug

SLURM - Ordonnancement

- **Fairshare**

Groupements hierarchiques d'utilisateurs, notion d' « account »



User 1 et 2 partagent le compte B

Comme le User 1 utilise beaucoup la machine sous le compte B, il impacte la priorité des jobs du User 2 qui sera moindre

Le user 3 passera en priorité

SLURM - Ordonnancement

- **Association – création de faishare**

sacctmgr create cluster=toto

sacctmgr create account name=economie fairshare=40

sacctmgr create account name=science fairshare=60

sacctmgr create account name=chimie parent=science fairshare=40

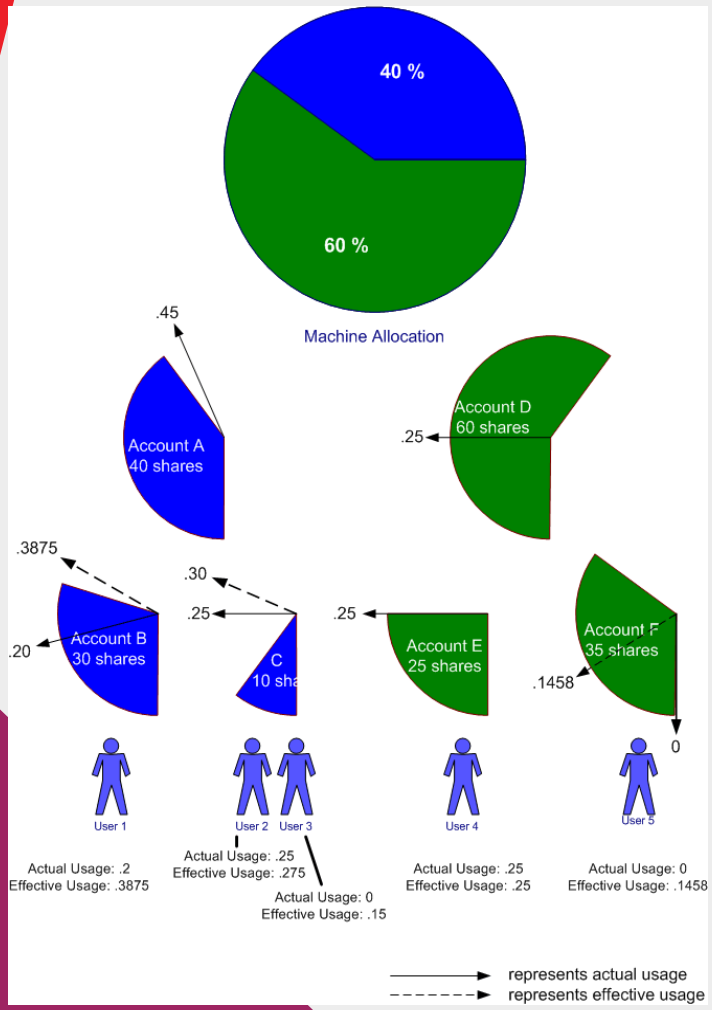
Sacctmgr create account name=physique parent=science fairshare=20

sacctmgr user name=kevin cluster=toto account=chimie

SLURM - Ordonnancement

- **Fairshare Factor (algorithme classique)**

Groupements hierarchiques d'utilisateurs, notion d' « account » :



User fairshare factor=

2** (-Uechild) / Schild)

Compris entre 0 et 1, >0.5 job plus prioritaire, < 0.5 job moins prioritaire

Uechild=Uachild +

((Uaparent - Uachild) *Schild/Sall_siblings)

S : Usage alloué normalisé,

Ua usage actuel, Ue usage effectif

Suser 1 normalized share: 0.3

Suser 2 normalized share: 0.05

Suser 3 normalized share: 0.05

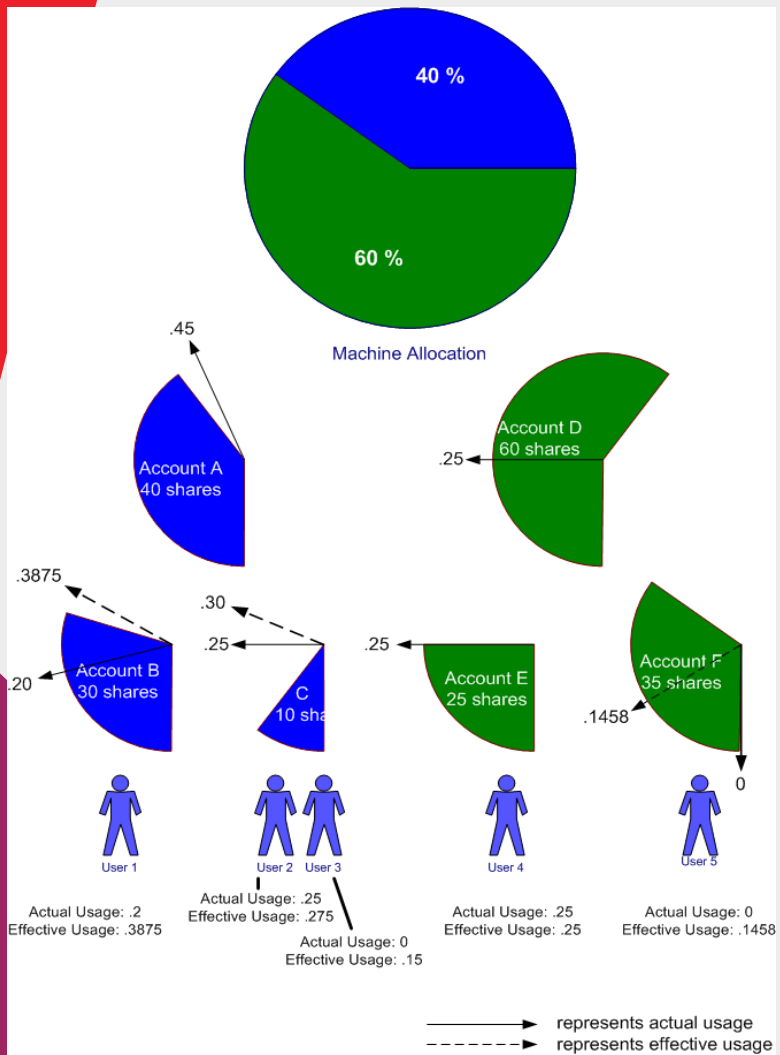
Suser 4 normalized share: 0.25

Suser 5 normalized share: 0.35

SLURM - Ordonnancement

- **Fairshare**

Groupements hierarchiques d'utilisateurs, notion d' « account » : fairshare factor



Tendances :User 1 occupe 20 (/30) %, User2 occupe 25 (/10) % et User4 occupe 25 (/25) % de la machine

Ue= Uachild + ((Ueparent - Uachild) *Schild/Sall_siblings)

S : Usage alloué normalisé, Ua usage actuel, Ue usage effectif

Account A's effective usage : .45

Account D's effective usage : .25

Account B effective usage: 0.2 + ((0.45 - 0.2) * 30 / 40) = 0.3875

Account C effective usage: 0.25 + ((0.45 - 0.25) * 10 / 40) = 0.3

Account E effective usage: 0.25 + ((0.25 - 0.25) * 25 / 60) = 0.25

Account F effective usage: 0.0 + ((0.25 - 0.0) * 35 / 60) = 0.1458

SLURM - Ordonnancement

- **Fairshare**

Groupements hierarchiques d'utilisateurs, notion d' « account »

User 1 occupe 20(/30) %, User2 occupe 25(/10) % et User4 25(/25)% occupe de la machine

User 1 effective usage: 0.2 + ((0.3875 - 0.2) * 1 / 1) = 0.3875

User 2 effective usage: 0.25 + ((0.3 - 0.25) * 1 / 2) = 0.275

User 3 effective usage: 0.0 + ((0.3 - 0.0) * 1 / 2) = 0.15

User 4 effective usage: 0.25 + ((0.25 - 0.25) * 1 / 1) = 0.25

User 5 effective usage: 0.0 + ((.1458 - 0.0) * 1 / 1) = 0.1458

User 1 fair-share factor: 2**(-.3875 / .3) = 0.408479

User 2 fair-share factor: 2**(-.275 / .05) = 0.022097

User 3 fair-share factor: 2**(-.15 / .05) = 0.125000

User 4 fair-share factor: 2**(-.25 / .25) = 0.500000

User 5 fair-share factor: 2**(-.1458 / .35) = 0.749154

Conclusion : User 2 tourne trop (facteur<0.5 ~0), le user 5 n'a pas tourné (facteur>0.5 ~1)

Priorité des jobs du user 3 (qui pourtant n'a rien tourné) influencée à la baisse car son facteur de fairshare diminué par les jobs du user 2 qui partage le compte C

La priorité des jobs du user 5 est influencée à la hausse par le fairshare

SLURM SYNTHESE

Vocabulaire	Signification
Nodes, partition, Job, Jobstep	Ressources et travaux
Users, Account, Qos, Fairshare, Associations	Restrictions
Backfilling	Ordonnanceur

Commandes :	Utilisez le man ! énormément d'options
sinfo	« Show resources »
squeue	« Show jobs »
sacct	« Show stats of jobs »
scontrol	« Admin show/modify config (slurmctld) »
sacctmgr	« Admin show/modify config (slurmdbd) »
srun, salloc	« Interactive launch »
sbatch	« Batch launch »
scancel	« Cancel job »

SLURM

Exemple du tgcc :

http://www-hpc.cea.fr/fr/complexe/tgcc.htm

La suite la semaine prochaine

https://www.schedmd.com/

SLURM : Liens vers cours et TP

ssh hpc.pedago.ensiie.fr

cd /home/shared/regine.gaudin/cours1

$ ls

batch-c1.odp

batch-c1.pdf
tp_slurm1_NOMETUDIANT.odt
tp_slurm1_NOMETUDIANT.pdf