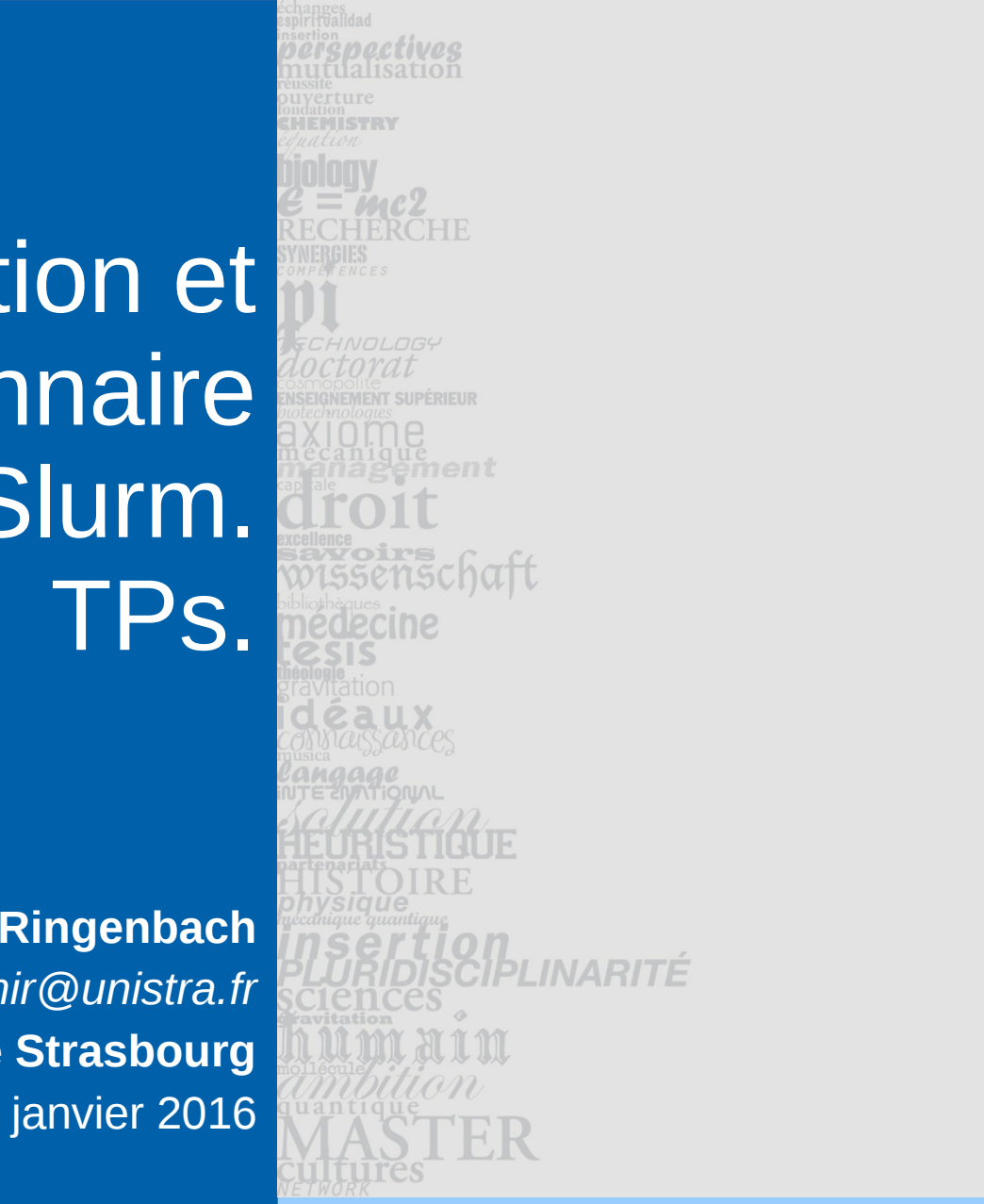
Installation, configuration et utilisation du gestionnaire batch Slurm. TPs.

Michel Ringenbach
mir@unistra.fr
Centre de calcul, Université de Strasbourg
19 janvier 2016



UNIVERSITÉ DE STRASBOURG

Sujet

- Ce que n'est pas cette formation
  - Un comparatif entre gestionnaires batch
- But de cette formation
  - Savoir répondre aux contraintes posées par les utilisateurs (je veux des ressources, tout de suite !)
  - Voir les grands principes communs à tous les gestionnaires batch, vus par le prisme Slurm
  - Savoir soumettre des jobs dans différents contextes
- 3 parties :
  - Cours : Principes et configuration de Slurm
  - TP guidé : Installation de Slurm
  - TPs et cours alternés : configuration et utilisation

UNIVERSITÉ DE STRASBOURG

Outline

- Introduction (brief)
- Unistra and the HPC Center
- From users needs to partitioning and access rules
- Associations and examples
- Utilities to manage accounts and quotas
- Prorities / Fairshare
- Accounting / Reparing DB incoherencies

UNIVERSITÉ DE STRASBOURG

Introduction

- The aim of this presentation is to show you all the specific Slurm configuration to achieve the needs of users of Unistra HPC Center
- We have several kind of users, with all their specific needs: high priority with guaranteed restitution, access to quotas of hours, instant execution on submission, public users with no rights at all...
- Why do we have such a diversity ?
- The reason is in the funding of the servers

UNIVERSITÉ DE STRASBOURG

HPC in Strasbourg

- University of Strasbourg (Unistra) is one of the major universities in France :
  - 45 000 students
  - more than 4800 employees
  - major research institute in many fields...
  - ...some of them need HPC
- Researchers in need for hpc use:
  - basement computers
  - **the HPC Center of the University**
  - French/Europe/Worldwide HPC computers

UNIVERSITÉ DE STRASBOURG

HPC Center

A small team composed of 3 people:

- Operating all the HPC facilities (clusters, datacenter)
- Supporting more than 50 HPC scientific software by:
  - Defining a standard set of tools we strongly support: Intel compilers + in-house built OpenMPI, Cuda
    When possible: building/linking the scientific apps against these standard tools
  - Writing and optimizing code
- Doing the training and support for the users

UNIVERSITÉ DE STRASBOURG

HPC Center: an history

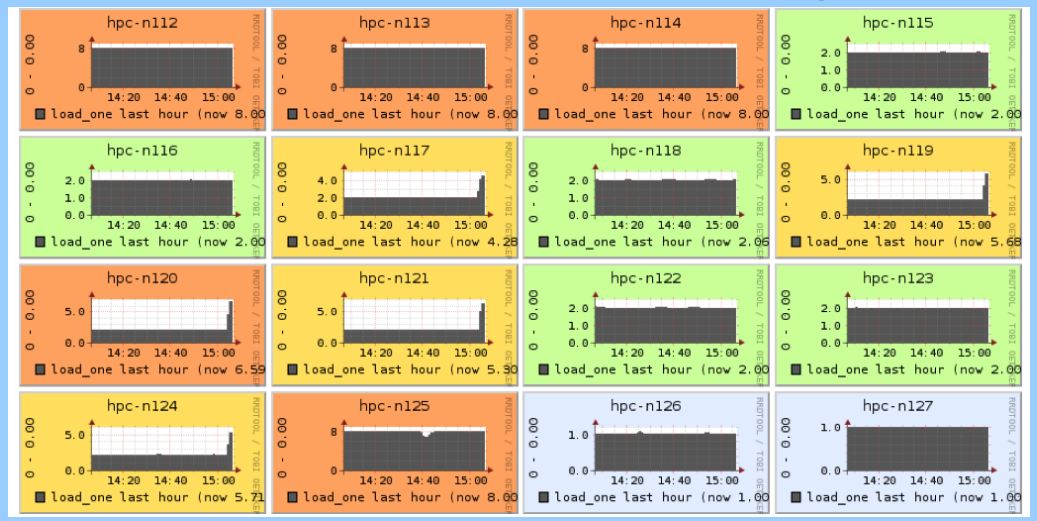
- As a not-so-new HPC computing center, we had a wide range of HPC machines (we probably tried all of the architectures since 1997):
  - 1997: SGI Origin 2k (MIPS)
  - 2003 - 2007: Cluster of Intel IA 64 nodes
  - 2005 - 2014: Cluster of AMD Opterons (several vendors)
  - 2009 - now: Cluster of Intel 64 nodes

HPC Center ressources: in short

A quick overview:

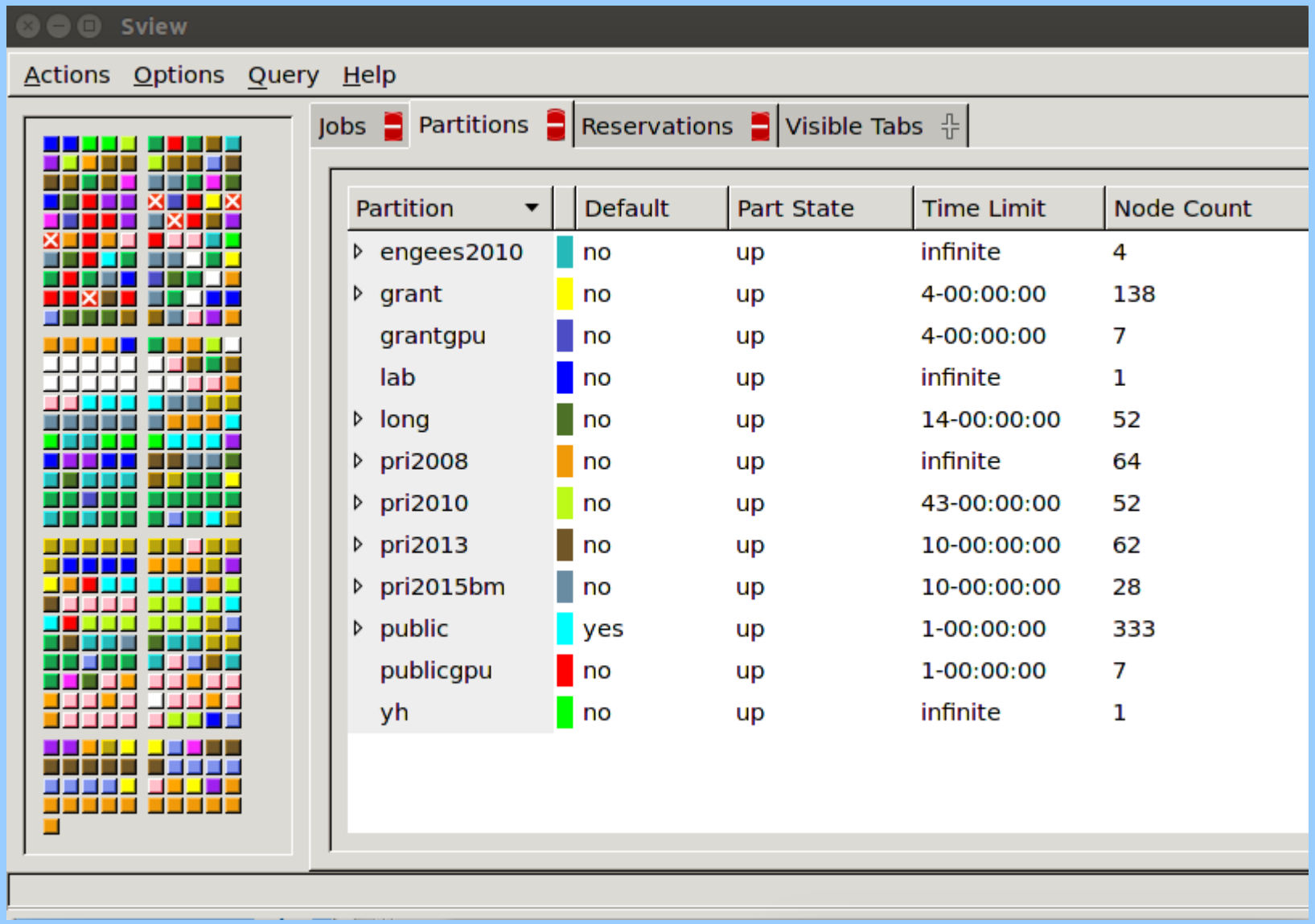
- 4700 compute cores in 341 compute nodes
- 18 TB memory
- Linpack performance: 75 TFlops
- 450 TB GPFS
- QDR Infiniband
- 30 Nvidia K40,K20,M20
  16xK80 coming soon
- Scientific Linux 6.5
- Slurm (of course!)

Load avg.: 85%



UNIVERSITÉ DE STRASBOURG

HPC Center ressources: in short



UNIVERSITÉ DE STRASBOURG

HPC Center ressources: in short

UNIVERSITÉ DE STRASBOURG

HPC Center: Fundings

- The research labs fundings: until 2013, 100% of the compute servers had been bought by the labs. Labs are located not only in Strasbourg, but in all the Alsace region (too many logos to show). Currently **20 different labs**
- The French national initiative *Investissements d'Avenir*, *via* a national project: Equip@Meso 1 million € = 145 nodes (plus datacenter, network, GPFS)

Cluster partitioning / access rules

- public: all nodes (334) except GPU nodes
  Non priority access for all users. Preempted.
  Equal fairshare. No fee.
- grant: national initiative nodes (145)
  Priority access. Projects quota allocation.
  Equal fairshare. No fee.
- pri*: nodes funded by labs (189)
  pri2008=Opteron, pri2010=Westmere, pri2013=SandyBridge
  Priority access. Proportional fairshare.
  Fee = buy the nodes.
- publicgpu + grantgpu : 7 nodes
- Some exceptions

UNIVERSITÉ DE STRASBOURG

Cluster partitioning / access rules


```
PartitionName=public MaxNodes=4 Default=YES \
   DefaultTime=0-10:0 MaxTime=1-0:0 \
   Nodes=hpc-n[101-220,228-441]
PartitionName=grant MaxNodes=32 \
   DefaultTime=1-0:0 MaxTime=4-0:0 \
   Nodes=hpc-n[228-365]
PartitionName=pri2010 \
   DefaultTime=0-10:0 MaxTime=43-0:0 \
   Nodes=hpc-n[165-220]
PartitionName=pri2013 \
   DefaultTime=0-10:0 MaxTime=10-0:0 \
   Nodes=hpc-n[366-385,387-441]
```

UNIVERSITÉ DE STRASBOURG

Cluster partitioning / access rules

- All access combinations must be possible. A single user can:
  - own nodes in different partitions
  - hold one or more grants (hours quotas)
  - being a public user (allways the case)
- Slurm configuration:
  - Access/deny to partitions for a **group or user**
  - Hours quotas for a **group or user**
  - Fairshare on all partitions (equal or proportional)
  - All the possible combinations per user

UNIVERSITÉ DE STRASBOURG

Associations

- To address all the contraints (except preemption), creating **associations** in the Slurm DB:
  cluster / user / account / partition (last is optional)
- Using associations attributes:
  - Fairshare (equal or proportional)
  - GrpCPUMins: quota in minutes. Setting by group.
  - GrpSubmitJobs: maximum number of jobs that can be submitted (reason is Fairshare, see later)

Associations

- One association at least is mandatory for submitting
  Reason is *slurm.conf*:AccountingStorageEnforce=associations
- We derivate this association from the login and group and associate it to the public partition
- The default with `sbatch` or `srun` is to search an association derivated from the login and group: it simplifies submitting
- 2nd advantage: all the access rules are centralized (in the Slurm DB).
  No extra rules depending on the OS or slurm.conf
  ⇒ no `scontrol reconfig`

UNIVERSITÉ DE STRASBOURG

Preemption

- To address preemption constraints, creating 2 QOS :
  - The first with no specific rights
  - The second, with preempting rights on the first one
  - All associations belong to one of these qos
- Jobs are canceled when preempted:
  In fact, it preempts only jobs in partition public, limited to 24 hours CPU: we do not waste too many hours.
  Jobs not canceled can disturb new jobs (memory conflict, ...)

```
sacctmgr create qos defaultqos
sacctmgr create qos preemptqos
   preempt=defaultqos preemptmode=cancel
```

UNIVERSITÉ DE STRASBOURG

Example

- User *john* belongs to the unix group *GRP*. He bought:
  - 4 Opteron nodes (32 cores) in 2008
  - 5 SandyBridge nodes (80 cores) in 2013
- It gives him access to partition pri2008 and pri2013
- Fairshares respectively 3200 and 8000 (ratios will be computed by Slurm)
- He decides to grant his group access to these nodes

UNIVERSITÉ DE STRASBOURG

Example

```
sacctmgr add...
    account qosgrp08 QOS=preemptqos   share=3200
    account qosgrp13 QOS=preemptqos   share=8000

sacctmgr create...
    user john account=qosgrp08 partition=pri2008
    user john account=qosgrp13 partition=pri2013
```

Example

- User *john* got a grant of compute hours:
  500 000 hours = 30 000 000 minutes
- It gives him access to partition grant
- Fairshare equal between all grant holders

```
sacctmgr add account grantjohn
  GrpCPUMins=30000000 QOS=preemptqos  share=1000

sacctmgr create user john
  account=grantjohn  partition=grant
```

UNIVERSITÉ DE STRASBOURG

Example

- But first of all, user *john (group grp)* is an ordinary user
- He can access partition *public*
- Fairshare equal between all public users
- The public jobs are preempted by grant or pri* jobs

```
sacctmgr add account grp qos=defaultqos  share=1000
sacctmgr create user john account=grp partition=public
sacctmgr create user john account=grp partition=publicgpu
```

UNIVERSITÉ DE STRASBOURG

Example

- How can *John* submit jobs using these associations ?
  - This is default (user belong to groupe grp):

```
#SBATCH -p public
#SBATCH -A grp
```

  - Going to partition grant, preempting public jobs:

```
#SBATCH -p grant
#SBATCH -A grantjohn
```

  - Going to partition pri2013, preempting public jobs:

```
#SBATCH -p pri2013
#SBATCH -A qosgrp13
```

UNIVERSITÉ DE STRASBOURG

Accessing the Slurm DB


```
sacctmgr list associations tree \
format=Account,User,Partition,Qos,Fairshare,GrpCPUMins,GrpSubmitJobs
```

Account	User	Partition	QOS	Share	GrpCPUMins	GrpSubmit
grp			defaultqos	1000		
grp	john	public	defaultqos	100		
grp	john	publicgpu	defaultqos	100		
grp	jim	public	defaultqos	100		
grp2			defaultqos	1000		
grp2	doe	public	defaultqos	100		
grantjohn			preemptqos	1000	30000000	
grantjohn	john	grant	preemptqos	100		
grant2			preemptqos	1000	6000000	0
grant2	doe	grant	preemptqos	100		
qosgrp13			preemptqos	8000		
qosgrp13	john	pri2013	preemptqos	100		
qosgrp13	jim	pri2013	preemptqos	100		
qosgrp08			preemptqos	3200		
qosgrp08	john	pri2008	preemptqos	100		
qosgrp08	jim	pri2008	preemptqos	100		

UNIVERSITÉ DE STRASBOURG

Accessing the Slurm DB


```
sacctmgr list qos  format=Name,Preempt,PreemptMode

      Name    Preempt PreemptMode
---------- ---------- -----------
defaultqos                cluster
preemptqos defaultqos      cancel


sacctmgr list associations where fairshare=8000

sacctmgr list associations where GrpSubmit=0

sacctmgr list associations user=john

etc. (sacctmgr modify...)
```

UNIVERSITÉ DE STRASBOURG

Utilities to manages accounts

- Needs an utility to manages accounts, because
  - It is combersome to create all default accounts (login based)
  - It is hard to know who has access to what (partitions, grants) → **description file**
- A script can automate processing rules
  - The script can translate the OS login DB into defaults accounts (each login = default association, public partition)
  - The same script can process a **description file** to generate all the other accounts
  - An adduser-like script can use the same **description file** to generate Slurm accounts for new users

UNIVERSITÉ DE STRASBOURG

Utilities to manages accounts

- Translating the OS login DB into defaults accounts
  - based on *getent passwd* and *getent group*
- Description file for non default accounts
  - `accounts.txt`: matching between an account and the partitions + attributes

```
<ACCOUNT>;<PARTITIONs>;<ATTRIBUTES>
grantjohn;grant,grantgpu;fairshare=1000 \
                                  GrpCPUMins=30000000

qosgrp13;pri2013;fairshare=6000
```

  - `login2accounts.txt:` matching between a login and accounts

```
<LOGIN>;<ACCOUNTs>
john;qosgrp08,qosgrp13,grantjohn
```

UNIVERSITÉ DE STRASBOURG

Utilities to manages accounts

- Description file for non default accounts (continue)
  - `groupe2accounts.txt`: matching between a group and accounts

    `<GROUP>;<ACCOUNTs>`

    `grp;qosgrp08,qosgrp13` # John gives access at his nodes to his group

    `astro1;grant1` # Group astro1 members have access to account grant1
- Utility developped at Unistra :
  - `slurm_initdb`: process OS DB + description file, output on stdout. Using it with « | `grep`»
  - `slurm_initdb.pm`: library used by above script and by adduser-like

UNIVERSITÉ DE STRASBOURG

Utility for quota checking

- Problem with quotas when **fairshare is on**
  - Our ***annuals*** quotas are stored in GrpCPUMins attribute
  - Value of GrpCPUMins is compared to the consumption over the period PriorityDecayHalfLife (2 weeks based)
    ⇒ quota never reached
- Solution: a script launched by cron
  - Check consumption with sreport and compare to GrpCPUMins
  - When over quota: set GrpSubmitJobs = 0
  - `slurm_quota_check`

UNIVERSITÉ DE STRASBOURG

Priority for pending jobs

- If resources become available, and more than one job is eligible, the one with the highest **priority** is launched
- Settings in slurm.conf based on factors with weights:

```
PriorityWeightJobSize      PriorityWeightAge
PriorityWeightFairshare
PriorityWeightPartition    PriorityWeightQOS
PriorityType=priority/multifactor
```

- Factor values: 0 to 1, multiplied by Weight

UNIVERSITÉ DE STRASBOURG

Priority for pending jobs

- We want to favor large jobs (on many nodes):

```
PriorityFavorSmall      = NO
PriorityWeightJobSize = 500000
```

Job on all the cluster nodes ⇒ JobSize factor = 1

- We want to favor old jobs (waiting for a long time) :

```
PriorityMaxAge      = 7-0     # days-hours
PriorityWeightAge = 400000
```

Job waiting for more than 7 days ⇒ Age factor = 1

- We want fairshare use between users, with higher weight than Size and Age factors :

```
PriorityDecayHalfLife    = 14-0
PriorityCalcPeriod       = 5 # refresh time in minutes
PriorityWeightFairshare = 900000
```

UNIVERSITÉ DE STRASBOURG

Check up fairshare


```
$ scontrol show job 1805062,1805813
JobId=1805062 Priority=121547 Account=grant24
   JobState=PENDING Reason=Resources Partition=grant
JobId=1805813 Priority=476688 Account=grant12
   JobState=PENDING Reason=Resources Partition=grant
```

- sprio -l

```
JOBID       USER   PRIORITY        AGE  FAIRSHARE    JOBSIZE
1805062      doe     121547     115214          0       6334
1805813      jim     476688      19600     455516       1573
# 455516 = 900000*0.506129 (900000=PriorityWeightFairshare)
```

- sshare -a -l

```
Account      User    Raw Usage  Norm Usage  FairShare
grant24       doe    193151415    0.030721   0.000000
grant12       jim      6686098    0.001063   0.506129
```

Check up fairshare

- From man sshare:
  « Norm Usage » = The Raw Usage normalized to the total number of cpu-seconds of all jobs run on the cluster, subject to the **PriorityDecayHalfLife** decay when defined

UNIVERSITÉ DE STRASBOURG

Accounting (specifically for quotas)

- Per jobs statistics can be obtained with **`sacct`** or directly by querying the database

```
sacct -u doe -S 2015-02-01T00:00:00 -E 2015-02-01T23:59:59
   --format Start,End,JobID,AllocCPUS,CPUTime
```

```
              Start                 End         JobID  AllocCPUS    CPUTime
2015-01-31T17:05:52 2015-02-01T00:06:04 1799410              240 70-00:48:00
2015-01-31T17:05:52 2015-02-01T00:06:04 1799410.bat+           1   07:00:12
2015-01-31T17:05:55 2015-02-01T00:06:04 1799410.0             14 4-02:02:06
2015-02-01T00:06:04 2015-02-01T07:06:38 1799411              240 70-02:16:00
2015-02-01T00:06:04 2015-02-01T07:06:38 1799411.bat+           1   07:00:34
2015-02-01T00:06:06 2015-02-01T07:06:38 1799411.0             14 4-02:07:28
2015-02-01T07:06:38 2015-02-01T14:06:56 1799412              240 70-01:12:00
2015-02-01T07:06:38 2015-02-01T14:06:56 1799412.bat+           1   07:00:18
2015-02-01T07:06:41 2015-02-01T14:06:56 1799412.0             14 4-02:03:30
2015-02-01T14:06:56 2015-02-01T21:07:14 1799413              240 70-01:12:00
2015-02-01T14:06:56 2015-02-01T21:07:14 1799413.bat+           1   07:00:18
2015-02-01T14:06:58 2015-02-01T21:07:14 1799413.0             14 4-02:03:44
2015-02-01T21:07:14 2015-02-02T04:08:02 1799414              240 70-03:12:00
```

UNIVERSITÉ DE STRASBOURG

Accounting (specifically for quotas)

- Consolidated statistics are periodically computed.
  Needs **slurmdbd** daemon to be started.
  Used to control the quotas use.
  Access with **`sreport:`**

```
sreport cluster UserUtilizationByAccount
   format=Login,Account,Used -t hours
   start=2015-02-01T00:00:00 end=2015-02-01T23:59:59

     Login            Account        Used
--------- --------------- ----------
      paul        g2014a136        5779
      john        g2014a110        5760
      jean         g2014b24        5054
   valjean   qosibmc2015bm        4608
     lupin     qosimfs2013        4236
```

DB incoherencies

- Going deep into job DB to repair incoherencies
- Real cases:
  - Can not supress an association: a **ghost job**, not visible with scancel was blocking sacctmgr

```
sacctmgr -i delete user x where account=y partition=z
Error: Job active, cancel job before remove
  JobID = 15653
scancel: error: job id 15653: Invalid job id specified
```

  - An account was debited by another **ghost job**. The user was paniqued when he saw, with sreport, that his quota was going down without any running job (no job id to process)

UNIVERSITÉ DE STRASBOURG

DB incoherencies

- How to repair the job DB (caution: Slurm version dependant):

```
mysql -u root –password
    mysql> use slurm_acct;
```

- The job can be present in 3 tables:

```
- hpc_job_table      : id_job, job_db_inx
- hpc_step_table     :         job_db_inx
- hpc_suspend_table  :         job_db_inx
```

- If the job id is not known, to find it, you have to select lines on userid criteria, and compare with eventuel running jobs (on the dates, node names,...) :

```
select * from hpc_job_table where id_user=...;
```

Bingo: if no *time_end* and not running/seen with squeue.

UNIVERSITÉ DE STRASBOURG

DB incoherencies

- When job id is known, ***check*** `job_db_inx`, and find all entries in the 3 concerned tables:
  `select * from hpc_step_table where job_db_inx=...;`
- If OK, then delete:
  `delete * from hpc_step_table where job_db_inx=...;`
- Real case (continuing):
  After deleting the ghost job in account debit problem, sreport stopped ***debiting***, but didn't recredit
  ⇒ had to manually credit the account (GrpCPUMins) with sacctmgr

UNIVERSITÉ DE STRASBOURG

Accounting (continuing)

- Querying directly the job database can help in some cases, when in doubt or when there are conflicts (users regularly check their remaining quota)

```
mysql -u root –password
use slurm_acct;
select * from hpc_job_table where id_user=...;

# You can verify CPU time consumed on job id basis:
select cpus_alloc*(time_end - time_start) from hpc_job_table
where job_db_inx=...;

# or on user basis:
where id_user=...;
```

UNIVERSITÉ DE STRASBOURG

Specific needs (for specific users)

- Some users (which funded nodes) want instant execution on submission. This is not compatible with fairshare
  ⇒ 2 specials partitions. Private nodes.
- Some users have limited time based requests
  ⇒ Slurm **reservations** on partitions subsets
- Some users funded « bigmem » nodes (>64GB)
  ⇒ 1 special partition. Private nodes.
- For some bigmem or GPU nodes, we use the **Weight** parameter in
  slurm.conf :**NodeName=**
  so they will be choosen last

UNIVERSITÉ DE STRASBOURG

The end

- All scripts accessible on
  http://hpc-web.u-strasbg.fr/SlurmTraining
- Questions ?

UNIVERSITÉ DE STRASBOURG

TP guidé : Installation de Slurm

- Pré-requis : couche d'authentification (ici Munge)
  - clef partagée sur l'ensemble du cluster
  - utilisée par les daemons slurmd et slurmctld
- Pré-requis : base de donnée (ici mysql)
  - accounting / reporting
  - gestion avancée des utilisateurs
  - slurmdbd : cache d'accès entre slurmctld et la BdD, nécessaire pour certaines commandes (sreport)

UNIVERSITÉ DE STRASBOURG

sbatch, srun, salloc

- http://slurm.schedmd.com/man_index.html
- Obtenir des ressources : dilemne du "trop" vs "pas assez"
  - -n : nombre de tâches à exécuter
  - -N min[-max] : nombre de noeuds (si range, attribution du max si possible)
  - -t days-hours:minutes:seconds
  - --mem= : mémoire réelle par noeud (Mo)
    - --mem-per-cpu= : par cpu
- Orienter le job
  - -p <partition> : autorisations nécessaires le cas échéant
- Obtenir les autorisations
  - -A <account>

UNIVERSITÉ DE STRASBOURG

sbatch, srun, salloc ; TP1

- sbatch : on soumet un script.
  - Paramètres spécifiés avec #SBATCH. Peuvent être écrasés à l'appel de sbatch.

```
vi test.slu
   #! /bin/bash       -> votre language préféré
   #SBATCH -n 1
   sleep 30 # mon application...
sbatch -n 2 test.slu
```

- srun : utilisation dans un script sbatch, ou appel en interactif d'une application (quand ressources dispo)
  - lance le programme autant de fois que spécifié par "-n"
  - dans sbatch, récupération automatique des paramètres
- salloc : allocation dynamique des ressources (si possible)

Supervision de jobs ; TP2

- squeue -p <partition> -u <user> -A <account> --nodelist=<node_list> -j <job_list> -o ...
- sinfo -p <partition> -n <node_list> -o ...
  - sinfo -N : par noeud
- node_list format : node[1-9] node[1,3,5-8,10-20]
- job_list format : 101-120 101,103,120-129,133
- sview
- scontrol show job : voir EligibleTime
- scancel <jobid>

ex: squeue -A lab1 --nodelist=node[10,15-27,30]

sinfo -o "%10P %.11L %.11l" : limites

Allocation des ressources ; TP3

- Segmentation des ressources : noeud ? socket ? cpu ? thread ? RAM ?
- Algorithme de sélection des ressources
  - Détermine l'unité de base d'allocation (+ compta)
  - slurm.conf : restart nécessaire. Reset de la compta !
  - SelectType=select/linear : 1 job = 1 noeud (par défaut)
    select/cons_res : 1 job consomme 1 ressource
  - SectTypeParameters=CR_CPU CR_CPU_Memory
    CR_Socket CR_Socket_Memory
  - NodeName... CPUs= Sockets= CoresPerSocket=
- TP3 (SectTypeParameters=CR_CPU)

UNIVERSITÉ DE STRASBOURG

Arrays, env vars, outputs ; TP4

- Exécutions par séries
- Array : exécution d'un groupe de jobs identiques, mais différentiés par un identifiant
  - --arrays=100-199 10,20,50,100-109 100-190:10
  - identifiant d'un job accessible via une variable d'environnement : SLURM_ARRAY_TASK_ID

    SLURM_ARRAY_JOB_ID : job id propre au groupe

    SLURM_JOB_ID : job id propre à chaque task id
  - Exemples d'utilisations :
    - if [$SLURM_ARRAY_TASK_ID -gt 10]; then ...
    - echo $resultat >> mon_fichier.$SLURM_ARRAY_JOB_ID
    - param = fonction ($SLURM_ARRAY_TASK_ID)

UNIVERSITÉ DE STRASBOURG

Arrays, env vars, outputs ; TP4

- Contrôle de l'environnement d'un job
- Variables d'environnement diverses :
  - SLURM_NPROCS : nombre de coeurs alloués
  - SLURM_NNODES : nombre de noeuds alloués
  - SLURM_JOB_ID : job id
  - SLURM_JOB_NODELIST : liste des noeuds alloués, sous une forme synthétique. Liste détaillée :

    scontrol show hostname $SLURM_JOB_NODELIST
  - Liste détaillée :
    - man sbatch, chapitre "INPUT ENVIRONMENT VARIABLES"
    - salloc puis "echo $SLURM <touche TAB>"

      ou ...$SBATCH...

UNIVERSITÉ DE STRASBOURG

Arrays, env vars, outputs ; TP4

- Contrôler les entrées/sorties
  - -o <fic> -e <fic> : sorties standard et erreur du script batch
    - -o mon_job.%J.std
  - Défaut : slurm-<JOBID>.out (std + err)

    arrays : slurm-<JOBID_ARRAYID>.out

Enchaînements conditionnels ; TP5

- L'exécution de plusieurs jobs peut être liée conditionnellement
- --dependency=
  - after:job_id[:jobid...] : après le démarrage de tous
  - afterany:job_id[:jobid...] : après le démarrage d'un seul
  - afternotok:job_id[:jobid...] : après l'échec de tous
  - afterok:job_id[:jobid...] : après la fin de tous
- Peut être utilisé lors d'un batch :
  - sbatch -d afterok:$SLURM_JOB_ID script_final

UNIVERSITÉ DE STRASBOURG

Preemption ; TP6

- 1 job prends la place d'un autre en cours d'exécution
- Plusieurs modes : suspend, requeue, cancel
- Configuration slurm.conf (restart nécessaire) :
  - PreemptType=preempt/qos
  - PreemptMode=requeue
  - JobRequeue=1
  - AccountingStorageEnforce=Associations,limits
    - Dans ce cas, seule une association valide permet de soumettre
    - limits : nécessaire pour les quotas (GrpCPUMins=)
  - ClusterName=hpc

UNIVERSITÉ DE STRASBOURG

Preemption ; TP6


```
sacctmgr -i create cluster hpc
sacctmgr -i create qos defaultqos
   preempt="" description="pas prio"
sacctmgr -i create qos preemptqos
   preempt=defaultqos preemptmode=requeue
   description="pas prio"
```

UNIVERSITÉ DE STRASBOURG

Priorités, fairshare : TP7

- **FIFO par défaut**

```
PriorityType=priority/multifactor
PriorityDecayHalfLife = 14-0
PriorityCalcPeriod = 5 # La consommation pondérée est recalculée toutes les PriorityCalcPeriod minutes
PriorityFavorSmall = NO # Priorité aux gros jobs
PriorityMaxAge=7-0 # Le facteur age vaudra 1 au bout de d'un temps d'attente de PriorityMaxAge
PriorityWeightAge       =  400000
PriorityWeightFairshare = 1000000
PriorityWeightJobSize   =  500000
#PriorityWeightPartition =
#PriorityWeightQOS       =
```

UNIVERSITÉ DE STRASBOURG

Priorités, fairshare : TP7

- Principal arbitre entre utilisateurs : le fairshare
  - Pourquoi ses jobs passent-ils avant les miens ?
- sshare ... -u user1,user2 ... -A group1,group2 ...
- sprio -l

UNIVERSITÉ DE STRASBOURG

Réservations ; TP8

- Je souhaite faire un big challenge... Je peux avoir tout le cluster pour moi tout seul ?

```
scontrol create reservation=ma_reservation
    starttime=
       2025-05-22T09:00:00
       now+10minutes
    endtime=...
    accounts=A1,A2,...
    users=U1,U2,...
    nodes=...
    partition=...
    flags=...
```

Suivi fin de jobs, compta ; TP9

- sacct a accès à tous les paramètres de l'accounting :
  - suivi fin de la consommation mémoire, cpus, etc.
- sreport génère des rapports synthétiques :
  - décideurs, financeurs, utilisateurs, admins
  - sreport cluster AccountUtilizationByUser -t hours

Suivi fin de jobs, compta ; TP9

Ecole cluster

20 janvier 2016

```
sacct -helpformat
```

AllocCPUS	Account	AssocID	AveCPU	AveCPUFreq	AveDiskRead	AveDiskWrite	AvePages
AveRSS	AveVMSize	BlockID	Cluster	Comment	ConsumedEnergy	ConsumedEnergyRaw	CPUTime
CPUTimeRAW	DerivedExitCode	Elapsed	Eligible	End	ExitCode	GID	Group
JobID	JobName	Layout	MaxDiskRead	MaxDiskReadNode	MaxDiskReadTask	MaxDiskWrite	MaxDiskWriteNode
MaxDiskWriteTask	MaxPages	MaxPagesNode	MaxPagesTask	MaxRSS	MaxRSSNode	MaxRSSTask	MaxVMSize
MaxVMSizeNode	MaxVMSizeTask	MinCPU	MinCPUNode	MinCPUTask	NCPUS	NNodes	NodeList
NTasks	Priority	Partition	QOS	QOSRAW	ReqCPUFreq	ReqCPUS	ReqMem
Reserved	ResvCPU	ResvCPURAW	Start	State	Submit	Suspended	SystemCPU
Timelimit	TotalCPU	UID	User	UserCPU	WCKey	WCKeyID	

sacct

--starttime= --endtime YYYY-MM-DD[THH:MM[:SS]]

--jobs=

--nodelist= --partition

--qos

--user= --accounts=

--state=PENDING,CONFIGURING,RUNNING,PREEMPTED,CANCELLED,FAILED,NODE_FAILED,COMPLETED...

--format --parsable

-X : cumul des steps

UNIVERSITÉ DE STRASBOURG