UL HPC School 2017

PS3a: Advanced Scheduling with SLURM and OAR on UL HPC clusters

UL High Performance Computing (HPC) Team

V. Plugaru

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu
Latest versions available on Github:

UL HPC tutorials: https://github.com/ULHPC/tutorials
UL HPC School: http://hpc.uni.lu/hpc-school/
PS3a tutorial sources: https://github.com/ULHPC/tutorials/tree/devel/advanced/advanced_scheduling
Introduction

Summary

1. Introduction

2. SLURM workload manager
   - SLURM concepts and design for iris
   - Running jobs with SLURM

3. OAR and SLURM

4. Conclusion
Introduction

Main Objectives of this Session

- Design and usage of **SLURM**
  - cluster workload manager of the UL HPC iris cluster

The tutorial will show you:

- the way SLURM was **configured**, **accounting** and **permissions**
- **common** and **advanced** SLURM tools and commands
  - srun, sbatch, squeue etc.
  - job specification
  - SLURM job types
  - comparison of SLURM (iris) and OAR (gaia & chaos)
- SLURM generic **launchers** you can use for your own jobs

Documentation & comparison to OAR

[https://hpc.uni.lu/users/docs/scheduler.html](https://hpc.uni.lu/users/docs/scheduler.html)
Summary

1 Introduction

2 SLURM workload manager
   SLURM concepts and design for iris
   Running jobs with SLURM

3 OAR and SLURM

4 Conclusion
SLURM workload manager

SLURM - core concepts

SLURM manages user jobs with the following key characteristics:

- set of requested resources:
  - number of computing resources: nodes (including all their CPUs and cores) or CPUs (including all their cores) or cores
  - amount of memory: either per node or per (logical) CPU
  - (wall)time needed for the user’s tasks to complete their work

- a requested node partition (job queue)

- a requested quality of service (QoS) level which grants users specific accesses

- a requested account for accounting purposes

Example: run an interactive job

Alias: si [...]

```
(access)$ srunt  -p interactive  --qos qos --interactive  --pty bash
(node)$ echo $SLURM_JOBID
2058
```

Simple interactive job running under SLURM
$ scontrol show job 2058
JobId=2058 JobName=bash
   UserId=vplugaru(5143) GroupId=clusterusers(666) MCS_label=N/A
   Priority =100 Nice=0 Account=ulhpc QOS=qos−interactive
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
   RunTime=00:00:08 TimeLimit=00:05:00 TimeMin=N/A
   SubmitTime=2017−06−09T16:49:42 EligibleTime=2017−06−09T16:49:42
   StartTime=2017−06−09T16:49:42 EndTime=2017−06−09T16:54:42 Deadline=N/A
   PreemptTime=None SuspendTime=None SecsPreSuspend=0
   Partition = interactive AllocNode:Sid=access2:163067
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=iris−081
   BatchHost=iris−081
   NumNodes=1 NumCPUs=1 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*:
   TRES=cpu=1,mem=4G,node=1
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=* 
   MinCPUsNode=1 MinMemoryCPU=4G MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=bash
   WorkDir=/mnt/irisgpfs/users/vplugaru
   Power=

Simple interactive job running under SLURM
Many metrics available during and after job execution

- including energy (J) – but with caveats
- job steps counted individually
- enabling advanced application debugging and optimization

Job information available in easily parseable format (add -p/-P)

```
$ sacct -j 2058 --format=account,user,jobid,jobname,partition,state
Account  User  JobID  JobName  Partition  State
ulhpc    vplugaru  2058  bash interacti + COMPLETED

$ sacct -j 2058 --format=elapsed,elapsedraw,start,end
Elapsed  ElapsedRaw  Start  End
00:02:56  176  2017-06-09T16:49:42  2017-06-09T16:52:38

$ sacct -j 2058 --format=maxrss,maxvmsize,consumedenergy,consumedenergyraw,nnodes,ncpus,nodelist
MaxRSS  MaxVMSize  ConsumedEnergy  ConsumedEnergyRaw  NNodes  NCPUS  NodeList
0  299660K  17.89K  17885.000000  1  1  iris-081
```

Job metrics after execution ended
### SLURM - design for iris (I)

<table>
<thead>
<tr>
<th>Partition</th>
<th># Nodes</th>
<th>Default time</th>
<th>Max time</th>
<th>Max nodes/user</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch*</td>
<td>80 (80%)</td>
<td>0-2:0:0</td>
<td>5-0:0:0</td>
<td>unlimited</td>
</tr>
<tr>
<td>interactive</td>
<td>10 (10%)</td>
<td>0-1:0:0</td>
<td>0-4:0:0</td>
<td>2</td>
</tr>
<tr>
<td>long</td>
<td>10 (10%)</td>
<td>0-2:0:0</td>
<td>30-0:0:0</td>
<td>2</td>
</tr>
</tbody>
</table>
SLURM - design for iris (I)

<table>
<thead>
<tr>
<th>Partition</th>
<th># Nodes</th>
<th>Default time</th>
<th>Max time</th>
<th>Max nodes/user</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch*</td>
<td>80 (80%)</td>
<td>0-2:0:0</td>
<td>5-0:0:0</td>
<td>unlimited</td>
</tr>
<tr>
<td>interactive</td>
<td>10 (10%)</td>
<td>0-1:0:0</td>
<td>0-4:0:0</td>
<td>2</td>
</tr>
<tr>
<td>long</td>
<td>10 (10%)</td>
<td>0-2:0:0</td>
<td>30-0:0:0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QoS</th>
<th>User group</th>
<th>Max nodes</th>
<th>Max jobs/user</th>
</tr>
</thead>
<tbody>
<tr>
<td>qos-besteffort</td>
<td>ALL</td>
<td>no limit</td>
<td></td>
</tr>
<tr>
<td>qos-batch</td>
<td>ALL</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>qos-interactive</td>
<td>ALL</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>qos-long</td>
<td>ALL</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>qos-batch-001</td>
<td>private</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>qos-interactive-001</td>
<td>private</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>qos-long-001</td>
<td>private</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>
Default partition: **batch**, meant to receive most user jobs
   → we hope to see majority of user jobs being able to scale

All partitions have a correspondingly named **QOS**
   → granting resource access (**long** – **qos-long**)
   → for now users required to **always specify QOS**
   → automation to make this even easier may be put in place soon
• **Default partition**: batch, meant to receive most user jobs
  → we hope to see majority of user jobs being able to scale

• All partitions have a correspondingly named **QOS**
  → granting resource access (**long – qos-long**)  
  → for now users required to **always specify QOS**  
  → automation to make this even easier may be put in place soon

• Preemptible **besteffort** QOS available for **batch** and **interactive** partitions (but **not** for **long**)  
  → meant to ensure maximum resource utilization  
  → should be used together with checkpointable software

• **QOSs specific to particular group accounts exist** (discussed later)  
  → granting additional accesses to platform contributors
Backfill scheduling for efficiency
- multifactor job priority (size, age, fairshare, QOS, ...)
- currently QOS weight set
- other factors/decay to be tuned after observation period
  ✓ i.e. with real user jobs – so this starts now

Resource selection: consumable resources
- cores and memory as consumable (per-core scheduling)
- block distribution for cores (best-fit algorithm)
- default memory/core: 4GB (4.1GB maximum, rest is for OS)
SLURM workload manager

SLURM - design for iris (III)

- **Backfill** scheduling for efficiency
  - multifactor job priority (size, age, fairshare, QOS, ...)  
  - currently QOS weight set  
  - other factors/decay to be tuned after observation period
    ✓  i.e. with real user jobs – so this starts now

- Resource selection: **consumable resources**
  - cores and memory as consumable (per-core scheduling)  
  - block distribution for cores (best-fit algorithm)  
  - default memory/core: 4GB (4.1GB maximum, rest is for OS)

- Reliable user process tracking with **cgroups**
  - cpusets used to constrain cores, RAM and swap (none!)  
  - task affinity used to bind tasks to cores (hwloc based)

- Hierarchical tree topology defined (network)
  - for optimized job resource allocation
SLURM workload manager

SLURM - design for iris (III)

- **Backfill** scheduling for efficiency
  - multifactor job priority (size, age, fairshare, QOS, . . . )
  - currently QOS weight set
  - other factors/decay to be tuned after observation period
    - i.e. with real user jobs –
- Resource selection: **consumable resources**
  - cores and memory as consumable (per-core scheduling)
  - block distribution for cores (best-fit algorithm)
  - default memory/core: 4GB (4.1GB maximum, rest is for OS)
- Reliable user process tracking with **cgroups**
  - cpuset used to constrain cores, RAM and swap (none!)
  - tasksets used to bind tasks to cores (hwloc based)
- Hierarchical tree topology defined (network)
  - for optimized job resource allocation

Help will be needed on your part to optimize your job parameters!
SLURM workload manager

A note on job priority

\[
\text{Job\_priority} = \\
(\text{PriorityWeightAge}) \times (\text{age\_factor}) + \\
(\text{PriorityWeightFairshare}) \times (\text{fair\_share\_factor}) + \\
(\text{PriorityWeightJobSize}) \times (\text{job\_size\_factor}) + \\
(\text{PriorityWeightPartition}) \times (\text{partition\_factor}) + \\
(\text{PriorityWeightQOS}) \times (\text{QOS\_factor}) + \\
\text{SUM}(\text{TRES\_weight\_cpu} \times \text{TRES\_factor\_cpu}, \\
\quad \text{TRES\_weight\_<type>} \times \text{TRES\_factor\_<type>}, \\
\quad \ldots)
\]

- TRES – Trackable RESources
  - CPU, Energy, Memory and Node tracked by default All details at slurm.schedmd.com/priority_multifactor.html
- The corresponding weights and reset periods we need to tune
  - we require real application usage in order to set up initial values
Some Details on job permissions...

- Partition limits + association-based rule enforcement
  - association settings in SLURM’s accounting database
- QOS limits imposed, jobs without QOS will not run (no default)
- Only users with existing associations able to run jobs
Some Details on job permissions...

- Partition limits + association-based rule enforcement
  → association settings in SLURM’s accounting database
- QOS limits imposed, jobs without QOS will not run (no default)
- Only users with existing associations able to run jobs

- **Best-effort** jobs possible through preemptible QOS: `qos-besteffort`
  → priority lower and preemptible by all other QOS
  → preemption mode is `requeue`, requeueing enabled by default
Some Details on job permissions...

- Partition limits + association-based rule enforcement
  - association settings in SLURM’s accounting database
- **QOS** limits imposed, jobs without QOS will not run (no default)
- Only users with existing **associations** able to run jobs

- **Best-effort** jobs possible through preemptible QOS: **qos-besteffort**
  - priority lower and preemptible by all other QOS
  - preemption mode is **requeue**, requeueuing enabled by default
- **On metrics**: Accounting & profiling data for jobs sampled every 30s
  - tracked: cpu, mem, energy
  - energy data retrieved through the **RAPL** mechanism
    - **caveat**: for energy not all hw. that may consume power is monitored with RAPL (CPUs, GPUs and DRAM are included)
On tightly coupled parallel jobs (MPI)

→ Process Management Interface (PMI 2) recommended
→ PMI2 used for better scalability and performance

✓ faster application launches
✓ tight integration w. SLURM's job steps mechanism (& metrics)
✓ we are also testing PMIx (PMI Exascale) support
On tightly coupled parallel jobs (MPI)

→ Process Management Interface (PMI 2) recommended
→ PMI2 used for better scalability and performance
  ✓ faster application launches
  ✓ tight integration w. SLURM's job steps mechanism (& metrics)
  ✓ we are also testing PMIx (PMI Exascale) support

→ PMI2 enabled in default software set for IntelMPI and OpenMPI
  ✓ requires minimal adaptation in your workflows
  ✓ replaces mpirun with SLURM’s srun (at minimum)
  ✓ if you compile/install your own MPI you’ll need to configure it

→ Example: https://hpc.uni.lu/users/docs/slurm_launchers.html
SLURM workload manager

SLURM - design for iris (V)

- **On tightly coupled parallel jobs (MPI)**
  - Process Management Interface (PMI 2) recommended
  - PMI2 used for better scalability and performance
    - ✔ faster application launches
    - ✔ tight integration w. SLURM’s job steps mechanism (& metrics)
    - ✔ we are also testing PMIx (PMI Exascale) support
  - PMI2 enabled in default software set for IntelMPI and OpenMPI
    - ✔ requires minimal adaptation in your workflows
    - ✔ replaces mpirun with SLURM’s srun (at minimum)
    - ✔ if you compile/install your own MPI you’ll need to configure it
  - **Example:** https://hpc.uni.lu/users/docs/slurm_launchers.html

- **SSH-based connections** between computing nodes still possible
  - other MPI implementations can still use ssh as launcher
    - ✔ but really shouldn’t need to, PMI2 support is everywhere
  - user jobs are tracked, no job == no access to node
Hierarchical **bank (group) accounts**

UL as root account, then underneath accounts for the 3 Faculties and 3 ICs

All Prof., Group leaders and above have **bank accounts**, linked to a Faculty or IC

→ with their own name: **Name.Surname**

All **user accounts** linked to a bank account

→ including Profs.’s own user

Iris accounting DB initialized with:

→ 70 group accounts from all Faculties/ICs

→ comprising 406 users

**Will allow better usage tracking and reporting than was possible before.**
SLURM workload manager

SLURM - brief commands overview

- `squeue`: view queued jobs
- `sinfo`: view queue, partition and node info,
- `sbatch`: submit job for batch (scripted) execution
- `srun`: submit interactive job, run (parallel) job step
- `scancel`: cancel queued jobs
## SLURM - brief commands overview

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>squeue</code></td>
<td>view queued jobs</td>
</tr>
<tr>
<td><code>sinfo</code></td>
<td>view queue, partition and node info,</td>
</tr>
<tr>
<td><code>sbatch</code></td>
<td>submit job for batch (scripted) execution</td>
</tr>
<tr>
<td><code>srune</code></td>
<td>submit interactive job, run (parallel) job step</td>
</tr>
<tr>
<td><code>scancel</code></td>
<td>cancel queued jobs</td>
</tr>
<tr>
<td><code>scontrol</code></td>
<td>detailed control and info. on jobs, queues, partitions</td>
</tr>
<tr>
<td><code>sstat</code></td>
<td>view system-level utilization (memory, I/O, energy) for running jobs / job steps</td>
</tr>
<tr>
<td><code>sacct</code></td>
<td>view system-level utilization for completed jobs / job steps (accounting DB)</td>
</tr>
<tr>
<td><code>sacctmgr</code></td>
<td>view and manage SLURM accounting data</td>
</tr>
</tbody>
</table>
SLURM workload manager

SLURM - brief commands overview

- `squeue`: view queued jobs
- `sinfo`: view queue, partition and node info,
- `sbatch`: submit job for batch (scripted) execution
- `srune`: submit interactive job, run (parallel) job step
- `scancel`: cancel queued jobs

- `scontrol`: detailed control and info. on jobs, queues, partitions
- `sstat`: view system-level utilization (memory, I/O, energy)
  - for running jobs / job steps
- `sacct`: view system-level utilization
  - for completed jobs / job steps (accounting DB)
- `sacctmgr`: view and manage SLURM accounting data

- `sprio`: view job priority factors
- `sshare`: view accounting share info. (usage, fair-share, etc.)
### SLURM - basic commands

<table>
<thead>
<tr>
<th>Action</th>
<th>SLURM command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit passive/batch job</td>
<td><code>sbatch $script</code></td>
</tr>
<tr>
<td>Start interactive job</td>
<td><code>srun --pty bash -i</code></td>
</tr>
<tr>
<td>Queue status</td>
<td><code>squeue</code></td>
</tr>
<tr>
<td>User job status</td>
<td><code>squeue -u $user</code></td>
</tr>
<tr>
<td>Specific job status (detailed)</td>
<td><code>scontrol show job $jobid</code></td>
</tr>
<tr>
<td>Job metrics (detailed)</td>
<td><code>sstat --job $jobid -l</code></td>
</tr>
<tr>
<td>Job accounting status (detailed)</td>
<td><code>sacct --job $jobid -l</code></td>
</tr>
<tr>
<td>Delete (running/waiting) job</td>
<td><code>scancel $jobid</code></td>
</tr>
<tr>
<td>Hold job</td>
<td><code>scontrol hold $jobid</code></td>
</tr>
<tr>
<td>Resume held job</td>
<td><code>scontrol release $jobid</code></td>
</tr>
<tr>
<td>Node list and their properties</td>
<td><code>scontrol show nodes</code></td>
</tr>
</tbody>
</table>

**QoS specification always necessary, also partition if not "batch"**
### SLURM - basic options for sbatch/srun

<table>
<thead>
<tr>
<th>Action</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request $n distributed nodes</td>
<td>-N $n</td>
</tr>
<tr>
<td>Request $m memory per node</td>
<td>--mem=$mGB</td>
</tr>
<tr>
<td>Request $mc memory per core (logical cpu)</td>
<td>--mem-per-cpu=$mcGB</td>
</tr>
<tr>
<td>Request job walltime</td>
<td>--time=d-hh:mm:ss</td>
</tr>
<tr>
<td>Request $tn tasks per node</td>
<td>--ntasks-per-node=$tn</td>
</tr>
<tr>
<td>Request $ct cores per task (multithreading)</td>
<td>-c $ct</td>
</tr>
<tr>
<td>Request $nt total # of tasks</td>
<td>-n $nt</td>
</tr>
<tr>
<td>Request to start job at specific $time</td>
<td>--begin $time</td>
</tr>
<tr>
<td>Specify job name as $name</td>
<td>-J $name</td>
</tr>
<tr>
<td>Specify job partition</td>
<td>-p $partition</td>
</tr>
<tr>
<td>Specify QOS</td>
<td>--qos $qos</td>
</tr>
<tr>
<td>Specify account</td>
<td>-A $account</td>
</tr>
<tr>
<td>Specify email address</td>
<td>--mail-user=$email</td>
</tr>
<tr>
<td>Request email on event</td>
<td>--mail-type=all[,begin,end,fail]</td>
</tr>
<tr>
<td>Use the above actions in a batch script</td>
<td>#SBATCH $option</td>
</tr>
</tbody>
</table>
### SLURM - basic options for sbatch/srun

<table>
<thead>
<tr>
<th>Action</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request $n distributed nodes</td>
<td>-N $n</td>
</tr>
<tr>
<td>Request $m memory per node</td>
<td>--mem=$mGB</td>
</tr>
<tr>
<td>Request $mc memory per core (logical cpu)</td>
<td>--mem-per-cpu=$mcGB</td>
</tr>
<tr>
<td>Request job walltime</td>
<td>--time=d-hh:mm:ss</td>
</tr>
<tr>
<td>Request $tn tasks per node</td>
<td>--ntasks-per-node=$tn</td>
</tr>
<tr>
<td>Request $ct cores per task (multithreading)</td>
<td>-c $ct</td>
</tr>
<tr>
<td>Request $nt total # of tasks</td>
<td>-n $nt</td>
</tr>
<tr>
<td>Request to start job at specific $time</td>
<td>--begin $time</td>
</tr>
<tr>
<td>Specify job name as $name</td>
<td>-J $name</td>
</tr>
<tr>
<td>Specify job partition</td>
<td>-p $partition</td>
</tr>
<tr>
<td>Specify QOS</td>
<td>--qos $qos</td>
</tr>
<tr>
<td>Specify account</td>
<td>-A $account</td>
</tr>
<tr>
<td>Specify email address</td>
<td>--mail-user=$email</td>
</tr>
<tr>
<td>Request email on event</td>
<td>--mail-type=all[,begin,end,fail]</td>
</tr>
<tr>
<td>Use the above actions in a batch script</td>
<td>#SBATCH $option</td>
</tr>
</tbody>
</table>

- **Diff. between -N, -c, -n, --ntasks-per-node, --ntasks-per-core?**
  - Normally you’d specify **-N** and **--ntasks-per-node**
    - fix the latter to 1 and add **-c** for MPI+OpenMP jobs
  - If your application is scalable, just **-n** might be enough
    - iris is homogeneous (for now)
SLURM workload manager

**SLURM - more options for sbatch/srun**

<table>
<thead>
<tr>
<th>Start job when... (dependencies)</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>these other jobs have started</td>
<td>-d after:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended</td>
<td>-d afterany:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with no errors</td>
<td>-d afterok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with errors</td>
<td>-d afternok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>all other jobs with the same name have ended</td>
<td>-d singleton</td>
</tr>
</tbody>
</table>

*Job dependencies and especially "singleton" can be very useful!*
SLURM workload manager

SLURM - more options for sbatch/srun

<table>
<thead>
<tr>
<th>Start job when... (dependencies)</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>these other jobs have started</td>
<td>-d after:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended</td>
<td>-d afterany:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with no errors</td>
<td>-d afterok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with errors</td>
<td>-d alernok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>all other jobs with the same name have ended</td>
<td>-d singleton</td>
</tr>
</tbody>
</table>

Job dependencies and especially "singleton" can be very useful!

<table>
<thead>
<tr>
<th>Allocate job at... (specified time)</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact time today</td>
<td>--begin=16:00</td>
</tr>
<tr>
<td>tomorrow</td>
<td>--begin=tomorrow</td>
</tr>
<tr>
<td>specific time relative to now</td>
<td>--begin=now+2hours</td>
</tr>
<tr>
<td>given date and time</td>
<td>--begin=2017-06-23T07:30:00</td>
</tr>
</tbody>
</table>

Jobs run like this will wait as PD – Pending with "(BeginTime)" reason
## SLURM workload manager

### SLURM - more options for sbatch/srun

<table>
<thead>
<tr>
<th>Start job when... (dependencies)</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>these other jobs have started</td>
<td>-d after:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended</td>
<td>-d afterany:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with no errors</td>
<td>-d afterok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>these other jobs have ended with errors</td>
<td>-d afternok:$jobid1:$jobid2</td>
</tr>
<tr>
<td>all other jobs with the same name have ended</td>
<td>-d singleton</td>
</tr>
</tbody>
</table>

Job dependencies and especially "singleton" can be very useful!

<table>
<thead>
<tr>
<th>Allocate job at... (specified time)</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact time today</td>
<td>--begin=16:00</td>
</tr>
<tr>
<td>tomorrow</td>
<td>--begin=tomorrow</td>
</tr>
<tr>
<td>specific time relative to now</td>
<td>--begin=now+2hours</td>
</tr>
<tr>
<td>given date and time</td>
<td>--begin=2017-06-23T07:30:00</td>
</tr>
</tbody>
</table>

Jobs run like this will wait as PD – Pending with "(BeginTime)" reason

<table>
<thead>
<tr>
<th>Other scheduling request</th>
<th>sbatch/srun option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ask for minimum/maximum # of nodes</td>
<td>-N minnodes-maxnodes</td>
</tr>
<tr>
<td>Ask for minimum run time (start job faster)</td>
<td>--time-min=d-hh:mm:ss</td>
</tr>
<tr>
<td>Ask to remove job if deadline can’t be met</td>
<td>--deadline=YYYY-MM-DD[THH:MM[:SS]]</td>
</tr>
<tr>
<td>Run job within pre-created (admin) reservation</td>
<td>--reservation=$reservationname</td>
</tr>
<tr>
<td>Allocate resources as specified job</td>
<td>--jobid=$jobid</td>
</tr>
</tbody>
</table>

Can use --jobid to connect to running job (different than sattach!)
SLURM - environment variables

- 53 input env. vars. can be used to define job parameters
  - almost all have a command line equivalent
- up to 59 output env. vars. available within job environment
  - some common ones:

<table>
<thead>
<tr>
<th>Description</th>
<th>Environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>$SLURM_JOBID</td>
</tr>
<tr>
<td>Job name</td>
<td>$SLURM_JOB_NAME</td>
</tr>
<tr>
<td>Name of account under which job runs</td>
<td>$SLURM_JOB_ACCOUNT</td>
</tr>
<tr>
<td>Name of partition job is running in</td>
<td>$SLURM_JOB_PARTITION</td>
</tr>
<tr>
<td>Name of QOS the job is running with</td>
<td>$SLURM_JOB_QOS</td>
</tr>
<tr>
<td>Name of job’s advance reservation</td>
<td>$SLURM_JOB_RESERVATION</td>
</tr>
<tr>
<td>Job submission directory</td>
<td>$SLURM_SUBMIT_DIR</td>
</tr>
<tr>
<td>Number of nodes assigned to the job</td>
<td>$SLURM_NNODES</td>
</tr>
<tr>
<td>Name of nodes assigned to the job</td>
<td>$SLURM_JOB_NODELIST</td>
</tr>
<tr>
<td>Number of tasks for the job</td>
<td>$SLURM_NTASKS or $SLURM_NPROCS</td>
</tr>
<tr>
<td>Number of cores for the job on current node</td>
<td>$SLURM_JOB_CPUS_PER_NODE</td>
</tr>
<tr>
<td>Memory allocated to the job per node</td>
<td>$SLURM_MEM_PER_NODE</td>
</tr>
<tr>
<td>Memory allocated per core</td>
<td>$SLURM_MEM_PER_CPU</td>
</tr>
<tr>
<td>Task count within a job array</td>
<td>$SLURM_ARRAY_TASK_COUNT</td>
</tr>
<tr>
<td>Task ID assigned within a job array</td>
<td>$SLURM_ARRAY_TASK_ID</td>
</tr>
</tbody>
</table>

Outputting these variables to the job log is essential for bookkeeping!
## Usage examples (I)

### Interactive jobs

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>srun -p interactive --qos qos-interactive --time=0:30 -N2 --ntasks-per-node=4 --pty bash -i</code></td>
<td>Interactive job with specific QoS and time limits.</td>
</tr>
<tr>
<td><code>srun -p interactive --qos qos-interactive --pty --x11 bash -i</code></td>
<td>Interactive job with X11 support.</td>
</tr>
<tr>
<td><code>srun -p interactive --qos qos-besteffort --pty bash -i</code></td>
<td>Interactive job with best-effort QoS.</td>
</tr>
</tbody>
</table>
SLURM workload manager

Usage examples (I)

> Interactive jobs

- `srun -p interactive --qos qos-interactive --time=0:30 -N2 --ntasks-per-node=4 --pty bash -i`
- `srun -p interactive --qos qos-interactive --pty --x11 bash -i`
- `srun -p interactive --qos qos-besteffort --pty bash -i`

> Batch jobs

- `sbatch job.sh`
- `sbatch -N 2 job.sh`
- `sbatch -p batch --qos qos-batch job.sh`
- `sbatch -p long --qos qos-long job.sh`
- `sbatch --begin=2017-06-23T07:30:00 job.sh`
- `sbatch -p batch --qos qos-besteffort job.sh`
SLURM workload manager

Usage examples (I)

> Interactive jobs

```
srun -p interactive --qos qos-interactive --time=0:30 -N2 --ntasks-per-node=4 --pty bash -i
srun -p interactive --qos qos-interactive --pty --x11 bash -i
srun -p interactive --qos qos-besteffort --pty bash -i
```

> Batch jobs

```
sbatch job.sh
sbatch -N 2 job.sh
sbatch -p batch --qos qos-batch job.sh
sbatch -p long --qos qos-long job.sh
sbatch --begin=2017-06-23T07:30:00 job.sh
sbatch -p batch --qos qos-besteffort job.sh
```

Status and details for partitions, nodes, reservations

```
squeue / squeue -l / squeue -la / squeue -l -p batch / squeue -t PD
scontrol show nodes / scontrol show nodes $nodename
sinfo / sinfo -s / sinfo -N
sinfo -T
```
Collecting job information, priority, expected start time

```
scontrol show job $jobid  # this is only available while job is in the queue + 5 minutes
sprio -l
squeue --start -u $USER
```
Usage examples (II)

### Collecting job information, priority, expected start time

- `scontrol show job $jobid` # this is only available while job is in the queue + 5 minutes
- `sprio -l`
- `squeue --start -u $USER`

### Running job metrics – sstat tool

- `sstat -j $jobid / sstat -j $jobid -l`
- `sstat -j $jobid1 --format=AveCPU,AveRSS,AveVMSize,MaxRSS,MaxVMSize`
- `sstat -p -j $jobid1,$jobid2 --format=AveCPU,AveRSS,AveVMSize,MaxRSS,MaxVMSize`
SLURM workload manager

Usage examples (II)

Collecting job information, priority, expected start time

scontrol show job $jobid  # this is only available while job is in the queue + 5 minutes
sprio -l
squeue --start -u $USER

Running job metrics – sstat tool

sstat -j $jobid / sstat -j $jobid -l
sstat -j $jobid1 --format=AveCPU,AveRSS,AveVMSize,MaxRSS,MaxVMSize
sstat -p -j $jobid1,$jobid2 --format=AveCPU,AveRSS,AveVMSize,MaxRSS,MaxVMSize

Completed job metrics – sacct tool

sacct -j $jobid / sacct -j $jobid -l
sacct -p -j $jobid --format=account,user,jobid,jobname,partition,state,elapsed,elapsedraw,
\ start,end,maxrss,maxvmsize,consumedenergy,consumedenergysraw,nnodes,ncpus,nodelist
sacct --starttime 2017-06-12 -u $USER
SLURM workload manager

Usage examples (III)

**Controlling queued and running jobs**

- `scontrol hold $jobid`
- `scontrol release $jobid`
- `scontrol suspend $jobid`
- `scontrol resume $jobid`
- `scancel $jobid`
- `scancel -n $jobname`
- `scancel -u $USER`
- `scancel -u $USER -p batch`
- `scontrol requeue $jobid`
Controlling queued and running jobs

- `scontrol hold $jobid`
- `scontrol release $jobid`
- `scontrol suspend $jobid`
- `scontrol resume $jobid`
- `scancel $jobid`
- `scancel -n $jobname`
- `scancel -u $USER`
- `scancel -u $USER -p batch`
- `scontrol requeue $jobid`

Checking accounting links and QOS available for you

- `sacctmgr show user $USER format=user%20s,defaultaccount%30s`
- `sacctmgr list association where users=$USER format=account%30s,user%20s,qos%120s`
Usage examples (III)

Controlling queued and running jobs

scontrol hold $jobid
scontrol release $jobid
scontrol suspend $jobid
scontrol resume $jobid
scancel $jobid
scancel -n $jobname
scancel -u $USER
scancel -u $USER -p batch
scontrol requeue $jobid

Checking accounting links and QOS available for you

sacctmgr show user $USER format=user%20s,defaultaccount%30s sacctmgr list association where users=$USER format=account%30s,user%20s,qos%120s

Checking accounting share info - usage, fair-share, etc.

sshare -U
sshare -A $accountname
sshare -A $(sacctmgr -n show user $USER format=defaultaccount%30s)
sshare -a
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:05:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

echo "Hello from the batch queue on node ${SLURM_NODELIST}"
# Your more useful application can be started below!
SLURM workload manager

Job launchers - basic (II)

```
#!/bin/bash -l
#SBATCH -N 2
#SBATCH --ntasks-per-node=2
#SBATCH --time=0-03:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"  
echo "== Node list: ${SLURM_NODELIST}"  
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"
# Your more useful application can be started below!
```
#!/bin/bash

#SBATCH -J MyTestJob
#SBATCH --mail-type=end,fail
#SBATCH --mail-user=Your.Email@Address.lu
#SBATCH -N 2
#SBATCH --ntasks-per-node=2
#SBATCH --time=03:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

```bash
echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"  
echo "== Node list: ${SLURM_NODELIST}" 
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"
```

# Your more useful application can be started below!
#!/bin/bash

#SBATCH -J MyLargeMemorySequentialJob
#SBATCH --mail-type=end,fail
#SBATCH --mail-user=Your.Email@Address.lu
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=64GB
#SBATCH --time=1-00:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"
echo "== Node list: ${SLURM_NODELIST}"
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"

# Your more useful application can be started below!

Use "mem" to request memory per node for low #core jobs
Job launchers - long jobs

```bash
#!/bin/bash
#SBATCH -J MyLongJob
#SBATCH --mail-type=all
#SBATCH --mail-user=Your.Email@Address.lu
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=3-00:00:00
#SBATCH -p long
#SBATCH --qos=qos-long

echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"
echo "== Node list: ${SLURM_NODELIST}"
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"
# Your more useful application can be started below!
```

Longer walltime now possible but you should not (!) rely on it. Always prefer batch and requeue-able jobs.
Many scientific applications support internal state saving and restart! We will also discuss system-level checkpoint-restart with DMTCP.
SLURM workload manager

Job launchers - threaded parallel

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 28
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
/path/to/your/threaded.app
```

By threaded we mean pthreads/OpenMP shared-memory applications.
SLURM workload manager

Job launchers - MATLAB

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=28
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load base/MATLAB
matlab -nodisplay -nosplash < /path/to/infile > /path/to/outfile
```

MATLAB spawns processes, limited for now to single node execution. We are still waiting for Distributed Computing Server availability.
MATLAB spawns processes, limited for now to single node execution. We are still waiting for Distributed Computing Server availability. 

As of the HPC School - June 2017 edition, the UL Matlab license server is not yet reachable from the iris cluster (dedicated tutorial will use gaia).
Currently the iris cluster is homogeneous. Its core networking is a non-blocking fat-tree.

- For now simply requesting a number of tasks (with 1 core/task) should be performant.
- Different MPI implementations will however behave differently.
  - Very recent/latest versions available on *iris* for IntelMPI, OpenMPI, MVAPICH2.
  - We ask that you let us know any perceived benefit for your applications when using one or the other.
- We will soon make available optimized MPI-layer parameters obtained during tuning executions.
  - And hope they will improve even more your time to solution.
SLURM workload manager

Job launchers - IntelMPI

```bash
#!/bin/bash -l
#SBATCH -n 128
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/intel
srun -n $SLURM_NTASKS /path/to/your/intel-toolchain-compiled-app
```

IntelMPI is configured to use PMI2 for process management (optimal).
Bare mpirun will not work for now.
#!/bin/bash -l
#SBATCH -n 128
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/foss
srun -n $SLURM_NTASKS /path/to/your/foss-toolchain-compiled-app

OpenMPI also uses PMI2 (again, optimal).
Bare mpirun does work but is not recommended.

You can easily generate a hostfile from within a SLURM job with:
srun hostname | sort -n > hostfile
#!/bin/bash -l
#SBATCH -N 10
#SBATCH --ntasks-per-node=1
#SBATCH -c 28
#SBATCH --time=00:01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/intel
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
srun -n $SLURM_NTASKS /path/to/your/parallel-hybrid-app

Compile and use your applications in hybrid MPI+OpenMP mode when you can for best possible performance.
Summary

1 Introduction

2 SLURM workload manager
   SLURM concepts and design for iris
   Running jobs with SLURM

3 OAR and SLURM

4 Conclusion
Notes on OAR

- OAR will remain the workload manager of Gaia and Chaos
  - celebrating 4158964 jobs on Gaia! (2017-06-11)
  - celebrating 1570698 jobs on Chaos! (2017-06-11)

- Many of its features are common to other workload managers, incl. SLURM
  - some things are exactly the same
  - but some things work in a different way
  - ... and some have no equivalent or are widely different

- An adjustment period for you and us is needed
  - next slides show a brief transition guide
## OAR/SLURM - commands guide

<table>
<thead>
<tr>
<th>Command</th>
<th>OAR (gaia/chaos)</th>
<th>SLURM (iris)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit passive/batch job</td>
<td>oarsub -S $script</td>
<td>sbatch $script</td>
</tr>
<tr>
<td>Start interactive job</td>
<td>oarsub -I</td>
<td>srun -p interactive --qos qos-interactive --pty bash -i</td>
</tr>
<tr>
<td>Queue status</td>
<td>oarstat</td>
<td>squeue</td>
</tr>
<tr>
<td>User job status</td>
<td>oarstat -u $user</td>
<td>squeue -u $user</td>
</tr>
<tr>
<td>Specific job status (detailed)</td>
<td>oarstat -f -j $jobid</td>
<td>scontrol show job $jobid</td>
</tr>
<tr>
<td>Delete (running/waiting) job</td>
<td>oardel $jobid</td>
<td>scancel $jobid</td>
</tr>
<tr>
<td>Hold job</td>
<td>oarhold $jobid</td>
<td>scontrol hold $jobid</td>
</tr>
<tr>
<td>Resume held job</td>
<td>oarresume $jobid</td>
<td>scontrol release $jobid</td>
</tr>
<tr>
<td>Node list and properties</td>
<td>oarnodes</td>
<td>scontrol show nodes</td>
</tr>
</tbody>
</table>

**Similar yet different?**

Many specifics will actually come from the way Iris is set up.
OAR/SLURM - job specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>OAR</th>
<th>SLURM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script directive</td>
<td>#OAR</td>
<td>#SBATCH</td>
</tr>
<tr>
<td>Nodes request</td>
<td>-l nodes=$count</td>
<td>-N $min-$max</td>
</tr>
<tr>
<td>Cores request</td>
<td>-l core=$count</td>
<td>-n $count</td>
</tr>
<tr>
<td>Cores-per-node request</td>
<td>-l nodes=$ncount/core=$ccount</td>
<td>-N $ncount</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ntasks-per-node=$ccount</td>
</tr>
<tr>
<td>Walltime request</td>
<td>-l [...] , walltime=hh:mm:ss</td>
<td>-t $min OR -t $days-hh:mm:ss</td>
</tr>
<tr>
<td>Job array</td>
<td>--array $count</td>
<td>--array $specification</td>
</tr>
<tr>
<td>Job name</td>
<td>-n $name</td>
<td>-J $name</td>
</tr>
<tr>
<td>Job dependency</td>
<td>-a $jobid</td>
<td>-d $specification</td>
</tr>
<tr>
<td>Property request</td>
<td>-p &quot;$property=$value&quot;</td>
<td>-C $specification</td>
</tr>
</tbody>
</table>

Job specifications will need most adjustment on your side...
... but thankfully Iris has a homogeneous configuration.
Running things in an optimal way will be much easier.
### OAR/SLURM - env. vars.

<table>
<thead>
<tr>
<th>Environment variable</th>
<th>OAR</th>
<th>SLURM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>$OAR_JOB_ID</td>
<td>$SLURM_JOB_ID</td>
</tr>
<tr>
<td>Resource list</td>
<td>$OAR_NODEFILE</td>
<td>$SLURM_NODELIST</td>
</tr>
<tr>
<td>Job name</td>
<td>$OAR_JOB_NAME</td>
<td>$SLURM_JOB_NAME</td>
</tr>
<tr>
<td>Submitting user name</td>
<td>$OAR_USER</td>
<td>$SLURM_JOB_USER</td>
</tr>
<tr>
<td>Task ID within job array</td>
<td>$OAR_ARRAY_INDEX</td>
<td>$SLURM_ARRAY_TASK_ID</td>
</tr>
<tr>
<td>Working directory at submission</td>
<td>$OAR_WORKING_DIRECTORY</td>
<td>$SLURM_SUBMIT_DIR</td>
</tr>
</tbody>
</table>

**Check available variables:** `env | egrep "OAR|SLURM"`

**Generate hostfile:** `srun hostname | sort -n > hostfile`
Summary

1. Introduction

2. SLURM workload manager
   SLURM concepts and design for iris
   Running jobs with SLURM

3. OAR and SLURM

4. Conclusion
Conclusion and Practical Session start

We’ve discussed

- The design of SLURM for the iris cluster
- The permissions system in use through group accounts and QOS
- Main SLURM tools and how to use them
- Job types possible with SLURM on iris
- SLURM job launchers for sequential and parallel applications
- Transitioning from OAR to SLURM

And now..

Short DEMO time!
Conclusion

Conclusion and Practical Session start

We’ve discussed

- The design of SLURM for the **iris** cluster
- The permissions system in use through group accounts and QOS
- Main SLURM tools and how to use them
- Job types possible with SLURM on **iris**
- SLURM job launchers for sequential and parallel applications
- Transitioning from OAR to SLURM

And now..

Short DEMO time!

Your Turn!
The UL High Performance Computing (HPC) Team
University of Luxembourg, Belval Campus:
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

1 Introduction

2 **SLURM workload manager**
   SLURM concepts and design for *iris*
   Running jobs with SLURM

3 **OAR and SLURM**

4 **Conclusion**