

# Uni.lu HPC School 2018

## PS3a: Job scheduling with SLURM and OAR

---



Uni.lu 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:

[ulhpc-tutorials.rtf.d.io/en/latest/scheduling/advanced/](http://ulhpc-tutorials.rtf.d.io/en/latest/scheduling/advanced/)





# Summary

- 1 **Introduction**
- 2 SLURM workload manager  
SLURM concepts and design for iris  
Running jobs with SLURM
- 3 OAR and SLURM
- 4 Conclusion

# Main Objectives of this Session



- **Design and usage of SLURM**

- ↪ cluster workload manager of the UL HPC iris cluster
- ↪ ... and future HPC systems

## 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>



# Summary

- 1 Introduction
- 2 SLURM workload manager**  
SLURM concepts and design for iris  
Running jobs with SLURM
- 3 OAR and SLURM
- 4 Conclusion

## 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)$ srun -p interactive --qos qos-interactive --pty bash
(node)$ echo $SLURM_JOBID
2058
```

Simple interactive job running under SLURM

# SLURM - job example (I)

```
$ 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
5  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
10 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
15 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
20 Gres=(null) Reservation=(null)
  OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
  Command=bash
  WorkDir=/mnt/irisgpf/users/vplugaru
  Power=
```

Simple interactive job running under SLURM

## SLURM - job example (II)

- 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

5  $ 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

10 $ 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)

Partition	# Nodes	Default time	Max time	Max nodes/user
batch*	132	0-2:0:0	5-0:0:0	unlimited
interactive	10(~6.5%)	0-1:0:0	0-4:0:0	2
long	10(~6.5%)	0-2:0:0	30-0:0:0	2

# SLURM - design for iris (I)

Partition	# Nodes	Default time	Max time	Max nodes/user
batch*	132	0-2:0:0	5-0:0:0	unlimited
interactive	10(~6.5%)	0-1:0:0	0-4:0:0	2
long	10(~6.5%)	0-2:0:0	30-0:0:0	2

QoS	User group	Max cores	Max jobs/user
<b>qos-besteffort</b>	<b>ALL</b>	<b>no limit</b>	
<b>qos-batch</b>	<b>ALL</b>	<b>2294</b>	100
<b>qos-interactive</b>	<b>ALL</b>	<b>224</b>	10
<b>qos-long</b>	<b>ALL</b>	<b>224</b>	10
qos-batch-001	private	1400	100
qos-interactive-001	private	56	10
qos-long-001	private	56	10

## SLURM - design for iris (II)

- **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)
  - ↪ any job is tied to one QOS (user specified or inferred)
  - ↪ automation in place to select QOS based on partition

## SLURM - design for iris (II)

- **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**)
  - ↪ any job is tied to one QOS (user specified or inferred)
  - ↪ automation in place to select QOS based on partition
- 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

## SLURM - design for iris (III)

- **Backfill** scheduling for efficiency
  - ↪ **multifactor job priority** (size, age, fairshare, QOS, ...)
  - ↪ currently weights set for: job age, partition and fair-share
  - ↪ other factors/decay to be tuned **after observation** period
    - ✓ with more user jobs in the queues
- 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 - design for iris (III)

- **Backfill** scheduling for efficiency
  - ↪ **multifactor job priority** (size, age, fairshare, QOS, ...)
  - ↪ currently weights set for: job age, partition and fair-share
  - ↪ other factors/decay to be tuned **after observation** period
    - ✓ with more user jobs in the queues
- 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 and RAM (no swap allowed)
  - ↪ task affinity used to bind tasks to cores (hwloc based)
- Hierarchical tree topology defined (for the network)
  - ↪ for optimized job resource allocation

## SLURM - design for iris (III)

- **Backfill** scheduling for efficiency
  - **multifactor job priority** (size, age, fairness)
  - currently weights set for: job age, job size, share
  - other factors/decay to be tuned for a given time period
    - ✓ with more user jobs in the queue
- Resource selection: **consumption**
  - **cores and memory** as primary (for core scheduling)
  - block distribution for memory (first fit algorithm)
  - default memory/cores ratio (user can set maximum, rest is for OS)
- Reliable user permissions with **cgroups**
  - cpuset for cores and RAM (no swap allowed)
  - task placement and tasks to cores (hwloc based)
- Hierarchical topology defined (for the network)
  - for efficient job resource allocation

**Help will be needed on your part  
to optimize your job parameters!**

## A note on job priority

```
Job_priority =  
    (PriorityWeightAge) * (age_factor) +  
    (PriorityWeightFairshare) * (fair-share_factor) +  
    (PriorityWeightJobSize) * (job_size_factor) +  
    (PriorityWeightPartition) * (partition_factor) +  
    (PriorityWeightQOS) * (QOS_factor) +  
    SUM(TRES_weight_cpu * TRES_factor_cpu,  
        TRES_weight_<type> * TRES_factor_<type>,  
        ...)
```

- TRES - Trackable RESources
  - ↪ CPU, Energy, Memory and Node tracked by default All details at [slurm.schedmd.com/priority\\_multifactor.html](http://slurm.schedmd.com/priority_multifactor.html)
- The corresponding weights and reset periods we **need to tune**
  - ↪ we require (your!) real application usage to optimize them



## SLURM - design for iris (IV)

### Some details on job permissions...

- Partition limits + association-based rule enforcement
  - ↳ association settings in SLURM's accounting database
- **QOS** limits imposed, e.g. you will see (QOSGrpCpuLimit)
- Only users with existing **associations** able to run jobs

## SLURM - design for iris (IV)

### Some details on job permissions...

- Partition limits + association-based rule enforcement
    - ↪ association settings in SLURM's accounting database
  - **QOS** limits imposed, e.g. you will see (QOSGrpCpuLimit)
  - Only users with existing **associations** able to run jobs
- 
- **Best-effort** jobs possible through preemptible QOS: **qos-besteffort**
    - ↪ of lower priority and preemptible by all other QOS
    - ↪ preemption mode is **requeue**, requeueing **disabled** by default

## SLURM - design for iris (IV)

### Some details on job permissions...

- Partition limits + association-based rule enforcement
    - ↪ association settings in SLURM's accounting database
  - **QOS** limits imposed, e.g. you will see (QOSGrpCpuLimit)
  - Only users with existing **associations** able to run jobs
- 
- **Best-effort** jobs possible through preemptible QOS: **qos-besteffort**
    - ↪ of lower priority and preemptible by all other QOS
    - ↪ preemption mode is **requeue**, requeueing **disabled** 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)

## SLURM - design for iris (V)

- **On tightly coupled parallel jobs (MPI)**
  - ↳ Process Management Interface (PMI 2) highly 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

## SLURM - design for iris (V)

- On tightly coupled parallel jobs (MPI)

- ↪ Process Management Interface (PMI 2) highly 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
  - ✓ replace **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](https://hpc.uni.lu/users/docs/slurm_launchers.html)

## SLURM - design for iris (V)

- On tightly coupled parallel jobs (MPI)

- Process Management Interface (PMI 2) highly 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
  - ✓ replace **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](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

## SLURM - design for iris (VI)

### ULHPC customizations through plugins

- Job submission rule / filter
  - ↳ for now: QOS initialization (if needed)
  - ↳ more rules to come (group credits, node checks, etc.)
- Per-job temporary directories creation & cleanup
  - ↳ better security and privacy, using kernel namespaces and binding
  - ↳ /tmp & /var/tmp are /tmp/\$jobid.\$rstcnt/[tmp,var\_tmp]
  - ↳ transparent for apps. ran through srun
  - ↳ **apps. ran with ssh cannot be attached, will see base /tmp!**
- X11 forwarding (GUI applications)
  - ↳ enabled with --x11 parameter to srun/salloc
  - ↳ **being rewritten to play nice with per-job tmpdir**
    - ✓ workaround 1: create job and ssh -X to head node (need to propagate job environment)
    - ✓ workaround 2: create job using salloc and then use ssh -X

## SLURM - design for iris (VII)

### Software licenses in SLURM

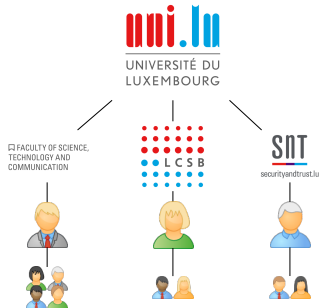
- Allinea Forge and Performance Reports for now
  - ↪ static allocation in SLURM configuration
  - ↪ dynamic checks for FlexNet / RLM based apps. coming later
- Number and utilization state can be checked with:
  - ↪ `scontrol show licenses`
- Use not enforced, **honor system** applied
  - ↪ `srun [...] -L $licname:$licnumber`

```
$> srun -N 1 -n 28 -p interactive -L forge:28 --pty bash -i
```



# SLURM - bank (group) accounts

- 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 contains over
  - ↳ 79 group accounts from all Faculties/ICs
  - ↳ comprising 554 users



**Allows better usage tracking and reporting than was possible before.**

## SLURM - brief commands overview

- **squeue**: view queued jobs
- **sinfo**: view 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

- **squeue**: view queued jobs
  - **sinfo**: view partition and node info.
  - **sbatch**: submit job for batch (scripted) execution
  - **srun**: 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

## SLURM - brief commands overview

- **squeue**: view queued jobs
  - **sinfo**: view partition and node info.
  - **sbatch**: submit job for batch (scripted) execution
  - **srun**: 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

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

**QOS deduced if not specified, partition needs to be set if not "batch"**

# SLURM - basic options for sbatch/srun

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

# SLURM - basic options for sbatch/srun

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

- 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
  - ↪ **beware of running across heterogeneous nodes:use '-C'**

## SLURM - more options for sbatch/srun

Start job when... (dependencies)	sbatch/srun option
these other jobs have started	-d after:\$jobid1:\$jobid2
these other jobs have ended	-d afterany:\$jobid1:\$jobid2
these other jobs have ended with no errors	-d afterok:\$jobid1:\$jobid2
these other jobs have ended with errors	-d afternok:\$jobid1:\$jobid2
all other jobs with the same name have ended	-d singleton

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



# SLURM - more options for sbatch/srun

Start job when... (dependencies)	sbatch/srun option
these other jobs have started	-d after:\$jobid1:\$jobid2
these other jobs have ended	-d afterany:\$jobid1:\$jobid2
these other jobs have ended with no errors	-d afterok:\$jobid1:\$jobid2
these other jobs have ended with errors	-d afternok:\$jobid1:\$jobid2
all other jobs with the same name have ended	-d singleton

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

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

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

# SLURM - more options for sbatch/srun

Start job when... (dependencies)	sbatch/srun option
these other jobs have started	-d after:\$jobid1:\$jobid2
these other jobs have ended	-d afterany:\$jobid1:\$jobid2
these other jobs have ended with no errors	-d afterok:\$jobid1:\$jobid2
these other jobs have ended with errors	-d afternok:\$jobid1:\$jobid2
all other jobs with the same name have ended	-d singleton

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

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

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

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

**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:

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

**Outputting these variables to the job log is essential for bookkeeping!**



## Usage examples (I)

### > Interactive jobs

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



## 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
```

## 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
```



## 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
```

## 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
```



## 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,consumedenergyraw,nnodes,ncpus,nodelist  
sacct --starttime 2017-06-12 -u $USER
```

## 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
```

## 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
```

## 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
```

## Job launchers - basic (I)

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=0-00:05:00
#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!
```

**Submit it with:** sbatch launcher.sh

## 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!
```

## Job launchers - basic (III)

```
#!/bin/bash -l
#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=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!
```

## Job launchers - requesting memory

```
#!/bin/bash -l
#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 (more) memory per node for low #core jobs



## Job launchers - node features selection

```
#!/bin/bash -l
#SBATCH -J MyJobOnSkylakeCPUs
#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
#SBATCH -C skylake
[...]
```

```
$> sinfo -format="%N %f"
NODELIST AVAIL_FEATURES
iris-[001-108] broadwell
iris-[109-168] skylake
```

## Job launchers - long jobs

```
#!/bin/bash -l
#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.**

## Job launchers - besteffort

```
#!/bin/bash -l
#SBATCH -J MyRerunnableJob
#SBATCH --mail-type=end,fail
#SBATCH --mail-user=Your.Email@Address.lu
#SBATCH -N 1
#SBATCH --ntasks-per-node=28
#SBATCH --time=0-12:00:00
#SBATCH -p batch
#SBATCH --qos=qos-besteffort
#SBATCH --requeue

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!
```

**Many scientific applications support internal state saving and restart!**  
**We will also discuss system-level checkpoint-restart with DMTCP.**

## 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.

## 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.**

## 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.**

## A note on parallel jobs

**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 can make available optimized MPI-layer parameters obtained during our tuning runs
  - ↪ and hope they will improve even more your time to solution

## Job launchers - IntelMPI

```
#!/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.**



## Job launchers - OpenMPI

```
#!/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
```

## Job launchers - MPI+OpenMP

```
#!/bin/bash -l
#SBATCH -N 10
#SBATCH --ntasks-per-node=1
#SBATCH -c 28
#SBATCH --time=0-01:00:00
#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 better (best?) possible performance.**

## Notes on optimizing your usage

### A note on CPU affinity

- Processes pinned by default to cores (**CPUs** in SLURM docs.)
- `srun` aware of requested tasks/cores configuration and pins processes/threads accordingly
- Many options to control task affinity exist, see:
  - ↪ [https://slurm.schedmd.com/srun.html#OPT\\_cpu-bind](https://slurm.schedmd.com/srun.html#OPT_cpu-bind)
  - ↪ [https://slurm.schedmd.com/srun.html#OPT\\_hint](https://slurm.schedmd.com/srun.html#OPT_hint)
- Can be disabled with `srun --cpu-bind=none`

If not disabled for 'interactive' jobs, all your processes will be pinned to 1st core!



# 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 **4357055** jobs on Gaia! (2018-06-08)
  - ↪ celebrating **1640883** jobs on Chaos! (2018-06-08)
- 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 is needed if you've only used OAR
  - ↪ next slides show a [brief transition guide](#)

# OAR/SLURM - commands guide

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

Similar yet different?

Many specifics will actually come from the way Iris is set up.

# OAR/SLURM - job specifications

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

**Job specifications will need most adjustment on your side.**  
**Iris more homogeneous than Gaia/Chaos for now.**  
**Running things in an optimal way is easier.**

## OAR/SLURM - env. vars.

Environment variable	OAR	SLURM
Job ID	<code>\$OAR_JOB_ID</code>	<code>\$SLURM_JOB_ID</code>
Resource list	<code>\$OAR_NODEFILE</code>	<code>\$SLURM_NODELIST</code> #List not file! See below.
Job name	<code>\$OAR_JOB_NAME</code>	<code>\$SLURM_JOB_NAME</code>
Submitting user name	<code>\$OAR_USER</code>	<code>\$SLURM_JOB_USER</code>
Task ID within job array	<code>\$OAR_ARRAY_INDEX</code>	<code>\$SLURM_ARRAY_TASK_ID</code>
Working directory at submission	<code>\$OAR_WORKING_DIRECTORY</code>	<code>\$SLURM_SUBMIT_DIR</code>

**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 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!**

# Questions?

<http://hpc.uni.lu>

## High Performance Computing @ uni.lu

Prof. Pascal Bouvry  
Dr. Sebastien Varrette  
Valentin Plugaru  
Sarah Peter  
Hyacinthe Cartiaux  
Clement Parisot

University of Luxembourg, Belval Campus  
Maison du Nombre, 4th floor  
2, avenue de l'Université  
L-4365 Esch-sur-Alzette  
*mail:* [hpc@uni.lu](mailto: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