HPC

uni.lu
UNIVERSITÉ DU LUXEMBOURG

High Performance Computing & Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

LUXEMBOURG
LET'S MAKE IT HAPPEN

easybuild

Uni.lu High Performance Computing (ULHPC) Facility

User Guide, 2020

slurm workload manager

UL HPC Team

https://hpc.uni.lu

AMD

intel

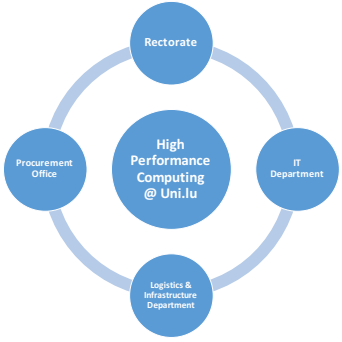
InfiniBand TRADE ASSOCIATION

Summary

Summary

High Performance Computing @ UL

- **Started in 2007** under resp. of Prof P. Bouvry & Dr. S. Varrette
  - ↪ 2nd Largest HPC facility in Luxembourg...
    - ✓ after EuroHPC MeluXina ($\geq$ 15 PFlops) system



https://hpc.uni.lu/

HPC/Computing Capacity

2794.23 TFlops

(incl. 748.8 GPU TFlops)

Shared Storage Capacity

10713.4 TB storage

High Performance Computing & Big Data Services

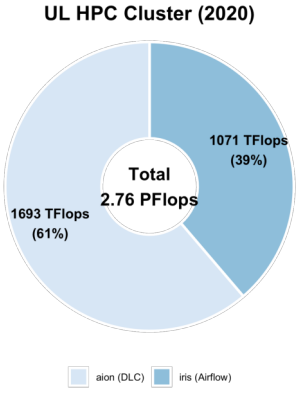
hpc.uni.lu

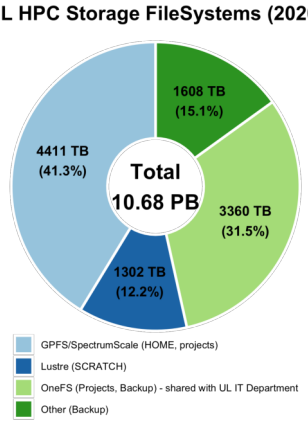
hpc@uni.lu

@ULHPC

LUXEMBOURG
LET'S MAKE IT HAPPEN

High Performance Computing @ UL





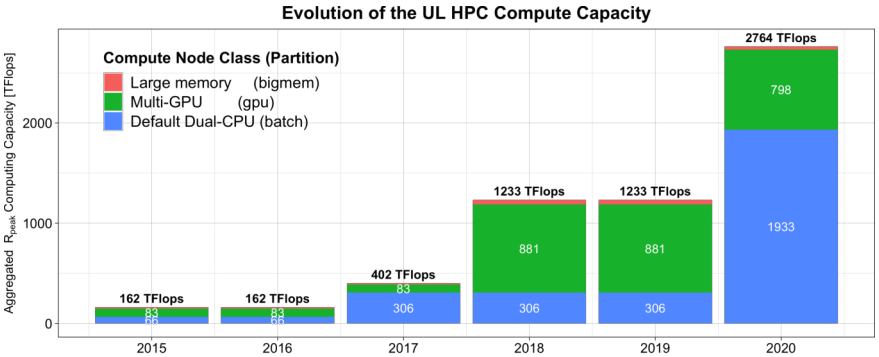
High Performance
Computing &
Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

High Performance Computing @ UL



- 3 types of computing resources across **2 clusters** (`aion`, `iris`)

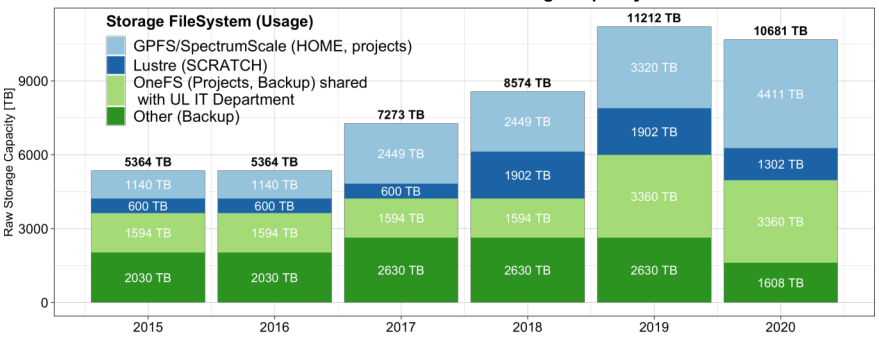
High Performance Computing & Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

High Performance Computing @ UL

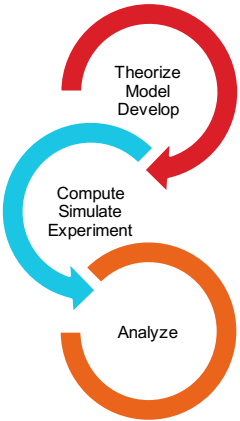


- 4 File Systems commons across the **2 clusters** (`aion`, `iris`)

High Performance Computing & Big Data Services

hpc.uni.lu
hpc@uni.lu
@ULHPC

Accelerating UL Research - User Software Sets

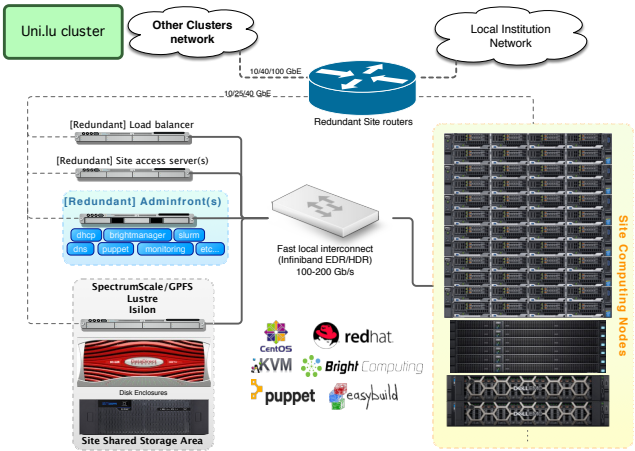


- **Over 230 software packages** available for researchers
  - ↪ software environment generated using Easybuild / LMod
  - ↪ containerized applications delivered with Singularity system

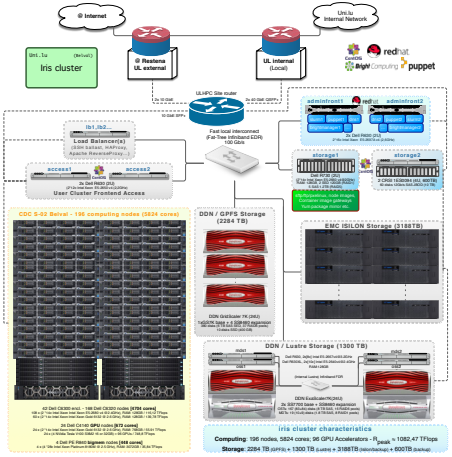
Domain	2019 Software environment
Compiler Toolchains	FOSS (GCC), Intel, PGI
MPI suites	OpenMPI, Intel MPI
Machine Learning	PyTorch, TensorFlow, Keras, Horovod, Apache Spark...
Math & Optimization	Matlab, Mathematica, R, CPLEX, Gurobi...
Physics & Chemistry	GROMACS, QuantumESPRESSO, ABINIT, NAMD, VASP...
Bioinformatics	SAMtools, BLAST+, ABySS, mpiBLAST, TopHat, Bowtie2...
Computer aided engineering	ANSYS, ABAQUS, OpenFOAM...
General purpose	ARM Forge & Perf Reports, Python, Go, Rust, Julia...
Container systems	Singularity
Visualisation	ParaView, OpenCV, VMD, VisIT
Supporting libraries	numerical (arpack-ng, cuDNN), data (HDF5, netCDF)...
...	

https://hpc.uni.lu/users/software/

UL HPC Supercomputers: General Architecture



UL HPC Supercomputers: `iris` cluster

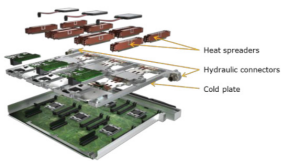


- **Dell/Intel** supercomputer, Air-flow cooling
  - ↪ 196 compute nodes
    - ✓ 5824 compute cores
    - ✓ Total 52224 GB RAM
  - ↪ R_{peak}: **1,072 PetaFLOP/s**
- Fast InfiniBand (IB) EDR network
  - ↪ **Fat-Tree** Topology

blocking factor 1:1.5

Rack ID	Purpose	Description
D02	Network	Interconnect equipment
D04	Management	Management servers, Interconnect
D05	Compute	`iris-[001-056]`, interconnect
D07	Compute	`iris-[057-112]`, interconnect
D09	Compute	`iris-[113-168]`, interconnect
D11	Compute	`iris-[169-177,191-193](gpu)`, `iris-[187-188](bigmem)`
D12	Compute	`iris-[178-186,194-196](gpu)`, `iris-[189-190](bigmem)`

UL HPC Supercomputers: `aion` cluster



- **Atos/AMD** supercomputer, DLC cooling
  - ↪ 4 BullSequana XH2000 adjacent racks
  - ↪ 318 compute nodes
    - ✓ 40704 compute cores
    - ✓ Total 81408 GB RAM
  - ↪ R_{peak}: **1,693 PetaFLOP/s**
- Fast InfiniBand (IB) HDR network
  - ↪ **Fat-Tree** Topology

blocking factor 1:2

	Rack 1	Rack 2	Rack 3	Rack 4	**TOTAL**
Weight [kg]	1872,4	1830,2	1830,2	1824,2	**7357 kg**
#X2410 Rome Blade	28	26	26	26	**106**
#Compute Nodes	84	78	78	78	**318**
#Compute Cores	10752	9984	9984	9984	**40704**
R_{peak} [TFlops]	447,28 TF	415,33 TF	415,33 TF	415,33 TF	**1693.29 TF**

UL HPC Software Stack

Operating System: **Linux CentOS/Redhat**

- **User Single Sign-on**: Redhat IdM/IPA
- **Remote connection & data transfer**: SSH/SFTP
  - ↪ **User Portal**: Open OnDemand
- **Scheduler/Resource management**: Slurm
- **(Automatic) Server / Compute Node Deployment**:
  - ↪ BlueBanquise, Bright Cluster Manager, Ansible, Puppet and Kadeploy
- **Virtualization and Container Framework**: KVM, Singularity
- **Platform Monitoring** (User level): Ganglia, SlurmWeb, OpenOndemand...
- **ISV software**:
  - ↪ ABAQUS, ANSYS, MATLAB, Mathematica, Gurobi Optimizer, Intel Cluster Studio XE, ARM Forge & Perf. Report, Stata, ...

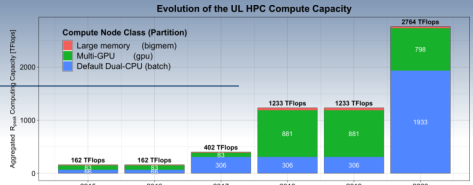
Summary

Slurm on ULHPC clusters

- ULHPC uses Slurm for cluster/resource management and job scheduling
  - ↪ Simple Linux Utility for Resource Management https://slurm.schedmd.com/
  - ↪ Handles submission, scheduling, execution, and monitoring of **jobs**
  - ↪ official documentation, tutorial, FAQ

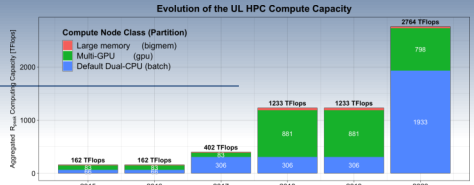
- **User jobs** have the following key characteristics:
  - ↪ set of requested resources:
    - ✓ number of computing resources: **nodes** (including all their CPUs and cores) or **CPUs** (including all their cores) or **cores**
    - ✓ amount of **memory**: either per node or per CPU
    - ✓ **(wall)time** needed for the users 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

Slum on ULHPC clusters



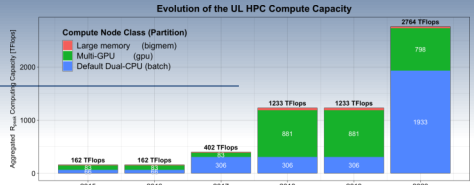
- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) — **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) — **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) — **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for quicks tests) — **Max**: 2 nodes, 2h walltime
    - ✓ for code development, testing, and debugging

Slurm on ULHPC clusters



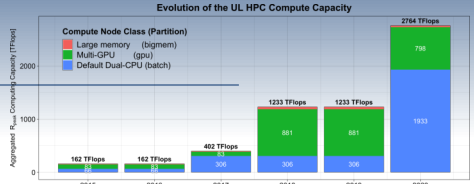
- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) — **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) — **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) — **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for quicks tests) — **Max**: 2 nodes, 2h walltime
    - ✓ for code development, testing, and debugging
- Queue Policy: **cross-partition QOS**, mainly tied to **priority level** (`low` → `urgent`)
  - ↪ `long` QOS with extended Max walltime (`MaxWall`) set to **14 days**
  - ↪ special **preemptible QOS** for best-effort jobs: `besteffort`.

Slurm on ULHPC clusters



- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) — **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) — **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) — **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for quicks tests) — **Max**: 2 nodes, 2h walltime
    - ✓ for code development, testing, and debugging
- Queue Policy: **cross-partition QOS**, mainly tied to **priority level** (`low` → `urgent`)
  - ↪ `long` QOS with extended Max walltime (`MaxWall`) set to **14 days**
  - ↪ special **preemptible QOS** for best-effort jobs: `besteffort`.
- Accounts associated to supervisor (multiple associations possible)
  - ↪ Proper group/user accounting

Slurm on ULHPC clusters



- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) — **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) — **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) — **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for quicks tests) — **Max**: 2 nodes, 2h walltime
    - ✓ for code development, testing, and debugging
- Queue Policy: **cross-partition QOS**, mainly tied to **priority level** (`low` → `urgent`)
  - ↪ `long` QOS with extended Max walltime (`MaxWall`) set to **14 days**
  - ↪ special **preemptible QOS** for best-effort jobs: `besteffort`.
- Accounts associated to supervisor (multiple associations possible)
  - ↪ Proper group/user accounting
- Slurm Federation configuration between `iris` and `aion`
  - ↪ ensures global policy (coherent job ID, global scheduling, etc.) within ULHPC systems
  - ↪ easily submit jobs from one cluster to another `-M, --cluster aion|iris`

Main Slurm Commands: Submit Jobs

```
$> sbatch -p <partition> [--qos <qos>] [-A <account>] [...]  <path/to/launcher.sh>
```

Submitting Jobs

- `sbatch`: Submit batch **launcher script** for later execution **batch/passive mode**
  - ↪ allocate resources (nodes, tasks, partition, etc.)
  - ↪ runs a single **copy** of the batch script on the **first** allocated node

Main Slurm Commands: Submit Jobs

```
$> srun -p <partition> [--qos <qos>] [-A <account>] [...]  --pty bash
```

Submitting Jobs

- `sbatch`: Submit batch **launcher script** for later execution **batch/passive mode**
  - ↪ allocate resources (nodes, tasks, partition, etc.)
  - ↪ runs a single **copy** of the batch script on the **first** allocated node
- `srun`: initiate parallel **job steps within a job OR start an interactive job**
  - ↪ allocate resources (number of nodes, tasks, partition, constraints, etc.)
  - ↪ launch a job that will execute on them.

Main Slurm Commands: Submit Jobs

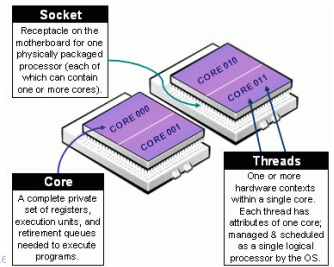
```
$> salloc -p <partition> [--qos <qos>] [-A <account>] [...]  <command>
```

Submitting Jobs

- `sbatch`: Submit batch **launcher script** for later execution **batch/passive mode**
  - ↪ allocate resources (nodes, tasks, partition, etc.)
  - ↪ runs a single **copy** of the batch script on the **first** allocated node
- `srun`: initiate parallel **job steps within a job OR start an interactive job**
  - ↪ allocate resources (number of nodes, tasks, partition, constraints, etc.)
  - ↪ launch a job that will execute on them.
- `salloc`: : request interactive jobs/allocations
  - ↪ allocate resources (nodes, tasks, partition, etc.), either run a command or start a shell.

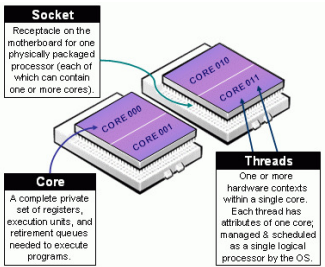
Specific Resource Allocation

- **Beware of Slurm terminology** in Multicore Architecture!
  - ↪ Slurm Node = Physical node — `-N <#nodes>`
    - ✓ **Advice**: explicit number of expected tasks **per node** — `--ntasks-per-node <n>`



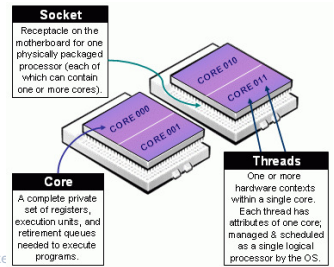
Specific Resource Allocation

- **Beware of Slurm terminology** in Multicore Architecture!
  - ↪ Slurm Node = Physical node — `-N <#nodes>`
    - ✓ **Advice**: explicit number of expected tasks **per node** — `--ntasks-per-node <n>`
  - ↪ Slurm Socket = Physical Socket/**CPU**/Processor — `--ntasks-per-socket <n>`



Specific Resource Allocation

- **Beware of Slurm terminology** in Multicore Architecture!
  - ↪ Slurm Node = Physical node `-N <#nodes>`
    - ✓ **Advice**: explicit number of expected tasks **per node** `--ntasks-per-node <n>`
  - ↪ Slurm Socket = Physical Socket/**CPU**/Processor `--ntasks-per-socket <n>`
  - ↪ **Slurm CPU** = Physical **Core** `-c <#threads>`
    - ✓ Hyper-Threading (HT) Technology is **disabled** on all the compute nodes
    - ✓ #cores = #threads `-c <N>` ⟶ `OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}`
    - ✓ Total number of tasks: `${SLURM_NTASKS}` ⟶ `srun -n ${SLURM_NTASKS} [...]`



Specific Resource Allocation

- **Beware of Slurm terminology** in Multicore Architecture!
  - ↪ Slurm Node = Physical node — `-N <#nodes>`
    - ✓ **Advice**: explicit number of expected tasks **per node** — `--ntasks-per-node <n>`
  - ↪ Slurm Socket = Physical Socket/**CPU**/Processor — `--ntasks-per-socket <n>`
  - ↪ **Slurm CPU** = Physical **Core** — `-c <#threads>`
    - ✓ Hyper-Threading (HT) Technology is **disabled** on all the compute nodes
    - ✓ #cores = #threads — `-c <N>` ⟶ `OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}`
    - ✓ Total number of tasks: `${SLURM_NTASKS}` ⟶ `srun -n ${SLURM_NTASKS} [...]`

- **Important: Always try to align resource specs with physical characteristics**
  - ↪ **Ex: 64 cores per socket and 2 sockets (physical CPUs) per `aion` node**
  - ↪ `[-N <N>] --ntasks-per-node <2n> --ntasks-per-socket <n> -c <thread>`
    - ✓ **Total**: `<N>`×2×`<n>` tasks, each on `<thread>` threads
    - ✓ **Ensure** `<n>`×`<thread>`=64 (#cores) **in this case** (target 14 on `iris`)
    - ✓ Ex: `-N 2 --ntasks-per-node 32 --ntasks-per-socket 16 -c 4` (**Total**: 64 tasks)

Specific Resource Allocation

- **Beware of Slurm terminology** in Multicore Architecture!
  - ↪ Slurm Node = Physical node — `-N <#nodes>`
    - ✓ **Advice**: explicit number of expected tasks **per node** — `--ntasks-per-node <n>`
  - ↪ Slurm Socket = Physical Socket/**CPU**/Processor — `--ntasks-per-socket <n>`
  - ↪ **Slurm CPU** = Physical **Core** — `-c <#threads>`
    - ✓ Hyper-Threading (HT) Technology is **disabled** on all the compute nodes
    - ✓ #cores = #threads — `-c <N>` ⟶ `OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}`
    - ✓ Total number of tasks: `${SLURM_NTASKS}` ⟶ `srun -n ${SLURM_NTASKS} [...]`

Hostname	Node type	#Nodes	#Socket	#Cores	RAM	Features
aion-[0001-0318]	Regular	318	2	128	256 GB	batch,epyc
iris-[001-108]	Regular	108	2	28	128 GB	batch,broadwell
iris-[109-168]	Regular	60	2	28	128 GB	batch,skylake
iris-[169-186]	Multi-GPU	18	2	28	768 GB	gpu,skylake,volta
iris-[191-196]	Multi-GPU	6	2	28	768 GB	gpu,skylake,volta32
iris-[187-190]	Large Memory	4	4	112	3072 GB	bigmem,skylake

- List available features: `sinfo -o "%30N  %.6D %.6c %15F %40P %f"`

Main Slurm Commands: Submit Jobs options

```
$> {sbatch | srun | salloc} [...]
```

Command-line option	Description	Example
-N <N>	<N> **Nodes** request	-N 2
--ntasks-per-socket=<n>	<n> Tasks-per-socket request	--ntasks-per-socket=14
--ntasks-per-node=<n>	<n> Tasks-per-node request	--ntasks-per-socket=28
-c=<c>	<c> Cores-per-task request (multithreading)	-c 1
--mem=<m>GB	<m>**GB memory per node** request	--mem 0
-t [DD-]HH[:MM:SS]>	**Walltime** request	-t 4:00:00
-G <gpu>	<gpu> GPU(s) request	-G 4
-C <feature>	Feature request (Ex: broadwell,skylake,...)	-C skylake
-p <partition>	Specify job partition/queue	
--qos <qos>	Specify job qos	
-A <account>	Specify account	
-J <name>	Job name	-J MyApp
-d <specification>	Job dependency	-d singleton
--mail-user=<email>	Specify email address	
--mail-type=<type>	Notify user by email when certain event types occur.	--mail-type=END,FAIL

Main Slurm Commands: Collect Information

- Partition (queue) and node status
  - ↪ eventually filter on specific job state (**R**:running /**PD**:pending / **F**:failed / **PR**:preempted)

```
$> squeue [-u <user>] [-p <partition>] [--qos <qos>] [-t R|PD|F|PR]
```

Main Slurm Commands: Collect Information

- Partition (queue) and node status
  ↪ eventually filter on specific job state (**R**:running /**PD**:pending / **F**:failed / **PR**:preempted)

```
$> squeue [-u <user>] [-p <partition>] [--qos <qos>] [-t R|PD|F|PR]
```

- Show partition status, summarized status (`-s`), problematic nodes (`-R`), reservations (`-T`)

```
$> sinfo [-p <partition>] {-s | -R | -T |...}
```

Main Slurm Commands: Collect Information

- Partition (queue) and node status
  ↪ eventually filter on specific job state (**R**:running /**PD**:pending / **F**:failed / **PR**:preempted)

```
$> squeue [-u <user>] [-p <partition>] [--qos <qos>] [-t R|PD|F|PR]
```

- Show partition status, summarized status (`-s`), problematic nodes (`-R`), reservations (`-T`)

```
$> sinfo [-p <partition>] {-s | -R | -T |...}
```

- View job, partition, nodes, reservation status

```
$> scontrol show { job <jobid> | partition [<part>] | nodes <node>| reservation...}
```

Main Slurm Commands: Collect Information

Command	Description
`sinfo`	Report system status (nodes, partitions etc.)
`squeue [-u $(whoami)]`	display jobs[steps] and their state
`seff <jobid>`	get efficiency metrics of past job
`scancel <jobid>`	cancel a job or set of jobs.
`scontrol show [...]`	view and/or update system, nodes, job, step, partition or reservation status
`sstat`	show status of running jobs.
`sacct [-X] -j <jobid> [...]`	display accounting information on jobs.
`sprio`	show factors that comprise a jobs scheduling priority
`smap`	graphically show information on jobs, nodes, partitions

```
### Get statistics on past job
slist <jobid>
# sacct [-X] -j <jobid> --format User,JobID,Jobname%30,partition,state,time,elapsed,MaxRss,\
#                                MaxVMSize,nnodes,ncpus,nodelist,AveCPU,ConsumedEnergyRaw
# seff <jobid>
```

ULHPC Slurm Partitions 2.0

`-p, -partition=<partition>`

```
$> {srun|sbatch|salloc|sinfo|squeue...} -p <partition> [...]
```

AION Partition	Type	#Node	PriorityTier	DefaultTime	MaxTime	MaxNodes
interactive	floating	318	100	30min	2h	2
batch		318	1	2h	48h	64

IRIS Partition	Type	#Node	PriorityTier	DefaultTime	MaxTime	MaxNodes
interactive	floating	196	100	30min	2h	2
batch		168	1	2h	48h	64
gpu		24	1	2h	48h	4
bigmem		4	1	2h	48h	1

ULHPC Slurm QOS 2.0

`--qos=<qos>`

```
$> {srun|sbatch|salloc|sinfo|squeue...} [-p <partition>] --qos <qos> [...]
```

QOS	Partition	Allowed [L1] Account	Prio	GrpTRES	MaxTresPJ	MaxJobPU	Flags
besteffort	*	**ALL**	1			100	NoReserve
low	*	**ALL** (default for CRP/externals)	10			2	DenyOnLimit
normal	*	**Default** (UL,Projects,...)	100			10	DenyOnLimit
long	*	UL,Projects,etc.	100	node=6	node=2	1	DenyOnLimit,PartitionTimeLimit
debug	interactive	**ALL**	150	node=8		2	DenyOnLimit
high	*	(**restricted**) UL,Projects,Industry	200			10	DenyOnLimit
urgent	*	(**restricted**) UL,Projects,Industry	1000			100 ?	DenyOnLimit

- **Cross-partition QOS**, mainly tied to **priority level** (low → urgent)
  - ↪ Simpler names than before (i.e. no more `qos-` prefix)
  - ↪ special **preemptible QOS** for best-effort jobs: `besteffort`

Slurm Launchers 2.0

```
#!/bin/bash -l
###SBATCH --job-name=<name>
###SBATCH --dependency singleton
###SBATCH -A <account>
#SBATCH --time=0-01:00:00  # 1 hour
#SBATCH --partition=batch  # If gpu: set '-G <gpus>'
#SBATCH -N 1               # Number of nodes
#SBATCH --ntasks-per-node=2
#SBATCH -c 1               # multithreading per task
#SBATCH -o %x-%j.out       # <jobname>-<jobid>.out
print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
# Load ULHPC modules
[ -f  /etc/profile ] && source /etc/profile
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK:-1}
module purge || print_error_and_exit "No 'module' command"
module load <...>
srun [-n $SLURM_NTASKS] [...]
```

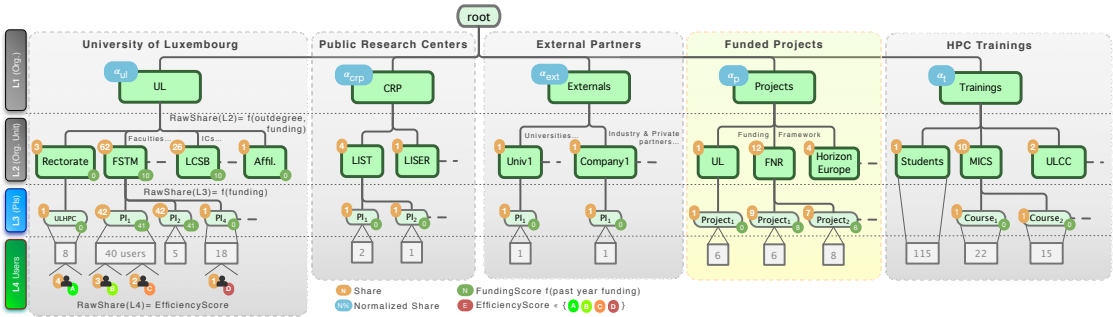
- **Best-Practices**
  - ↪ use `/bin/bash -l` on top
  - ↪ set **reasonable time limits**
  - ↪ set (short) job name
  - ↪ specify account
  - ↪ `--exclusive` allocation?
  - ↪ **Avoid** Job arrays
  - ↪ consider singleton pipelining
    - ✓ job dep. made easy
  - ↪ GPU jobs (gpu partition)
    - ✓ Set #GPUs with `-G <n>`
  - ↪ Use `$SCRATCH` for large/temporary storage
  - ↪ consider night jobs
    - ✓ `--begin=20:00`

Account Hierarchy 2.0

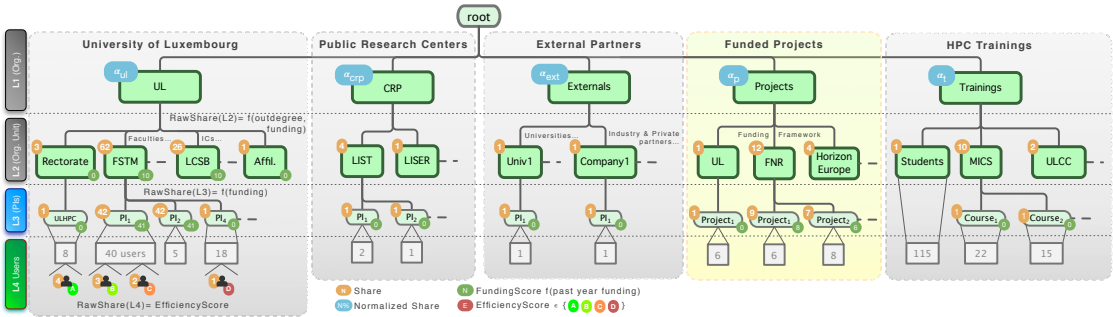
- Every user job runs under a group account
  - ↪ granting access to specific QOS levels.
  - ↪ default raw share for accounts: 1

- **L1**: Organization Level: UL, CRPs, Externals, Projects, Trainings
  - ↪ guarantee 80% of the shares for core UL activities
- **L2**: Organizational Unit (Faculty, ICs, External partner, Funding program. . .)
  - ↪ Raw share depends on outdegree and past year funding
- **L3**: Principal Investigator (PIs), Projects, Course
  - ↪ Raw share depends on past year funding
  - ↪ Eventually restricted **only** to projects and courses
- **L4**: End User (ULHPC login)
  - ↪ Raw share based on efficiency score

Account Hierarchy 2.0



Account Hierarchy 2.0




```
# L1,L2 or L3 account /!\ ADAPT <name> accordingly
sacctmgr show association tree where accounts=<name> format=account,share
# End user (L4)
sacctmgr show association where users=$USER  format=account,User,share,Partition,QOS
```

Efficiency Score (L4)

- **Updated every year based on past jobs efficiency**.
  - ↪ Similar notion of "nutri-score': A(very good - 3), B (good: 2), C (bad, 1), D(very bad - 0)
- Proposed Metric for **user** U: **Average Wall-time Accuracy (WRA)** (higher the better)
  - ↪ Defined for a given time period (past year)

```
sacct -u <U> -X -S <start> -E <end> [...] # --format User,JobID,state,time,elapsed
```

  - ↪ Reduction for N `COMPLETED` jobs:

$$S_{\text{efficiency}}(U, Year) = \frac{1}{N} \sum_{JobID \in (U, Year)} \frac{T_{\text{elapsed}}(JobID)}{T_{\text{asked}}(JobID)}$$

- Default thresholds

Score	Avg. WRA
A	$S_{\text{efficiency}} \geq 75\%$
B	$50\% \leq S_{\text{efficiency}} < 75\%$
C	$25\% \leq S_{\text{efficiency}} < 50\%$
D	$S_{\text{efficiency}} < 25\%$

- **WIP**: integrate other efficiency metrics (CPU, mem, GPU efficiency)

Job Priority, Fairsharing and Fair Tree

- **Fairsharing**: way of ensuring that users get their appropriate portion of a system
  - ↪ **Share**: portion of the system users have been granted.
  - ↪ **Usage**: amount of the system users have actually **used**.
  - ↪ **Fairshare score**: value the system calculates based off of user's usage.
    - ✓ difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
  - ↪ **Priority score**: priority assigned based off of the user's fairshare score.
- ULHPC Slurm configuration with Multifactor Priority Plugin and **Fair tree** algorithm
  - ↪ rooted plane tree (rooted ordered tree) being created then sorted by Level Fairshare
  - ↪ All users from a higher priority account receive a higher fair share factor than all users from a lower priority account

```
$> sshare -l                                                    # See Level FS
```

ULHPC Job Prioritization Factors

- **Age**: length of time a job has been waiting (PD state) in the queue, eligible to be scheduled
- **Fairshare**: difference between the portion of the computing resource that has been promised and the amount of resources that has been consumed
- **Partition**: factor associated with each node partition
  - ↪ Ex: privilege `interactive` over `batch`
- **QOS** A factor associated with each Quality Of Service (`low` ⟶ `urgent`)

```
Job_priority =
    PriorityWeightAge       * age_factor +
    PriorityWeightFairshare * fair-share_factor+
    PriorityWeightPartition * partition_factor +
    PriorityWeightQOS       * QOS_factor +
    - nice_factor
```

Fairshare Factor and Job billing

- Utilization of the University computational resources is charged in Service Unit (SU)
  - ↪ 1 SU $\simeq$ 1 hour on 1 physical processor core on regular computing node
  - ↪ Usage charged **0,03€ per SU (VAT excluded)** (external partners, funded projects etc.)
- A Job is characterized (and thus billed) according to the following elements:
  - ↪ T_{exec}: Execution time (in hours)
  - ↪ N_{Nodes}: number of computing nodes, and **per node**:
    - ✓ N_{cores}: number of CPU cores allocated per node
    - ✓ Mem: memory size allocated per node, in GB
    - ✓ N_{gpus}: number of GPU allocated per node
  - ↪ associated weighted factors $\alpha_{cpu}, \alpha_{mem}, \alpha_{GPU}$ defined as `TRESBillingWeight` in Slurm
    - ✓ account for consumed resources other than just CPUs
    - ✓ taken into account in fairshare factor
    - ✓ α_{cpu}: normalized relative perf. of CPU processor core (reference: skylake 73,6 GFlops/core)
    - ✓ α_{mem}: inverse of the average available memory size per core
    - ✓ α_{GPU}: weight per GPU accelerator

Fairshare Factor and Job billing

Number of SU associated to a job

$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	`n/a`	0	0	0
Iris	Regular	`batch`	28	`broadwell`	1.0*	$\frac{1}{4} = 0,25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0,25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	`epyc`	0,57	$\frac{1}{1.75}$	0

```
#  Billing rate for running job <jobID>
scontrol show job <jobID> | grep -i billing
# Billing rate for completed job <jobID>
sacct -X --format=AllocTRES%50,Elapsed -j <jobID>
```

Fairshare Factor and Job billing

Number of SU associated to a job

$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	n/a	0	0	0
Iris	Regular	`batch`	28	`broadwell`	1.0*	$\frac{1}{4} = 0,25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0,25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	epyc	0,57	$\frac{1}{1.75}$	0

- Continuous use of **2 regular skylake nodes** (56 cores, 224GB Memory) on `iris` cluster
  - ↪ 28 cores per node, 4 GigaByte RAM per core i.e., 112GB per node
  - ↪ **For 30 days**: 2 nodes$\times[\alpha_{cpu} \times 28 + \alpha_{mem} \times 4 \times 28 + \alpha_{gpu} \times 0]\times$ 30 days$\times$ 24 hours
    - ✓ Total: $2 \times [(1.0 + \frac{1}{4} \times 4) \times 28] \times 720 = 80640$ SU $=$ **2419,2€ VAT excluded**

Fairshare Factor and Job billing

Number of SU associated to a job

$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	`n/a`	0	0	0
Iris	Regular	`batch`	28	`broadwell`	1.0*	$\frac{1}{4} = 0,25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0,25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	`epyc`	0,57	$\frac{1}{1.75}$	0

- Continuous use of **2 regular epyc nodes** (256 cores, 448GB Memory) on `aion` cluster
  - ↪ 128 cores per node, 1,75 GigaByte RAM per core i.e., 224 GB per node
  - ↪ **For 30 days**: 2 nodes$\times[\alpha_{cpu} \times 128 + \alpha_{mem} \times 1.75 \times 128 + \alpha_{gpu} \times 0]\times$ 30 days$\times$ 24 hours
    - ✓ Total: $2 \times [(0.57 + \frac{1}{1.75} \times 1.75) \times 128] \times 720 = 289382{,}4$ SU = **8681,47€ VAT excluded**

Fairshare Factor and Job billing

Number of SU associated to a job

$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	`n/a`	0	0	0
Iris	Regular	`batch`	28	`broadwell`	1.0*	$\frac{1}{4} = 0,25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0,25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	`epyc`	0,57	$\frac{1}{1.75}$	0

- Continuous use of **1 GPU nodes** (28 cores, 4 GPUs, 756GB Memory) on `iris` cluster
  - ↪ 28 cores per node, 4 GPUs per nodes, 27 GigaByte RAM per core, 756 GB per node
  - ↪ **For 30 days**: 1 node$\times[\alpha_{cpu} \times 28 + \alpha_{mem} \times 27 \times 28 + \alpha_{gpu} \times 4$ GPUS$]\times$ 30 days$\times$ 24 hours
    - ✓ Total: $1 \times [(1.0 + \frac{1}{27} \times 27) \times 28 + 50.0 \times 4] \times 720 = 184320$ SU $=$ **5529,6€ VAT excluded**

Fairshare Factor and Job billing

Number of SU associated to a job

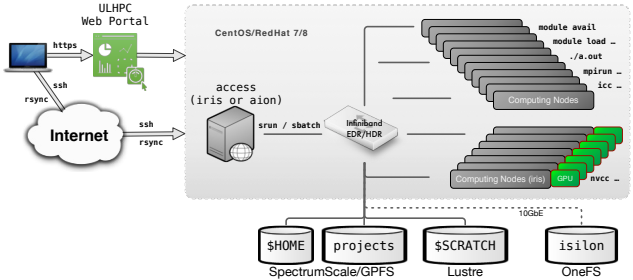
$$N_{\text{Nodes}} \times [\alpha_{cpu} \times N_{\text{cores}} + \alpha_{mem} \times Mem + \alpha_{gpu} \times N_{\text{gpus}}] \times T_{\text{exec}}$$

Cluster	Node Type	Partition	#Cores/node	CPU	α_{cpu}	α_{mem}	α_{GPU}
Iris, Aion	Regular	`interactive`	28/128	`n/a`	0	0	0
Iris	Regular	`batch`	28	`broadwell`	1.0*	$\frac{1}{4} = 0,25$	0
Iris	Regular	`batch`	28	`skylake`	1.0	$\frac{1}{4} = 0,25$	0
Iris	GPU	`gpu`	28	`skylake`	1.0	$\frac{1}{27}$	50
Iris	Large-Mem	`bigmem`	112	`skylake`	1.0	$\frac{1}{27}$	0
Aion	Regular	`batch`	128	`epyc`	0,57	$\frac{1}{1.75}$	0

- Continuous use of **1 Large-Memory nodes** (112 cores, 3024GB Memory) on `iris` cluster
  - ↪ 112 cores per node, 27 GigaByte RAM per core i.e. 3024 GB per node
  - ↪ **For 30 days**: 1 node$\times[\alpha_{cpu} \times 112 + \alpha_{mem} \times 27 \times 112 + \alpha_{gpu} \times 0]\times$ 30 days$\times$ 24 hours
    - ✓ Total: $1 \times [(1.0 + \frac{1}{27} \times 27) \times 112] \times 720 = 161280$ SU = **4838,4€ VAT excluded**

Summary

Compute Nodes / Storage Environment



- **Storage usage**: `df -ulhpc [-i]`
  - ↪ `$HOME`: regular backup policy
  - ↪ `$SCRATCH` **NO** backup & purged
    - ✓ 60 days retention policy
  - ↪ Project quotas attached to group
    - ✓ **not** *(default)* `clusterusers` group
    - ✓ Commands writing in project dir: `sg <group> -c "<command>"`
- **LMod/Environment modules**
  - ↪ **Not** on access, **only** on compute nodes

Directory	FileSystem	Max size	Max #files	Backup
`$HOME` (`iris`)	GPFS	500 GB	1.000.000	YES
`$SCRATCH`	Lustre	10 TB	1.000.000	**NO**
Project	GPFS	*per request*		PARTIALLY (`/backup` subdir)
Project	OneFS	*per request*		PARTIALLY

Software/Modules Management

https://hpc.uni.lu/users/software/

- Based on Environment Modules / LMod
  - ↪ convenient way to dynamically change the users environment $PATH
  - ↪ permits to easily load software through `module` command
- Currently on UL HPC: > **230 software packages**, in *multiple* versions, within **18 categ.**
  - ↪ reworked software set now deployed everywhere
    - ✓ RESIF v3.0, allowing [real] semantic versioning of released (arch-based) builds
  - ↪ hierarchical organization **Ex**: `toolchain/{foss,intel}`

```
$> module avail                                    # List available modules
```

```
$> module spider <pattern>          # Search for <pattern> within available modules
```

```
$> module load <category>/<software>[/<version>]
```

Software/Modules Management

- Key `module` variable: `$MODULEPATH` / where to look for modules.
  - ↪ **default** `iris`: `/opt/apps/resif/iris/<version>/{broadwell,skylake,gpu}/modules/all`
  - ↪ **default** `aion`: `/opt/apps/resif/aion/<version>/{epyc}/modules/all`
    - ✓ altered/prefix new path with `module use <path>`. **Ex** (to use **local** modules):

```
export EASYBUILD_PREFIX=$HOME/.local/easybuild
export LOCAL_MODULES=$EASYBUILD_PREFIX/modules/all
module use $LOCAL_MODULES
```

Software/Modules Management

- Key `module` variable: `$MODULEPATH` / where to look for modules.
  - ↪ **default** `iris`: `/opt/apps/resif/iris/<version>/{broadwell,skylake,gpu}/modules/all`
  - ↪ **default** `aion`: `/opt/apps/resif/aion/<version>/{epyc}/modules/all`
    - ✓ altered/prefix new path with `module use <path>`. **Ex** (to use **local** modules):

```
export EASYBUILD_PREFIX=$HOME/.local/easybuild
export LOCAL_MODULES=$EASYBUILD_PREFIX/modules/all
module use $LOCAL_MODULES
```

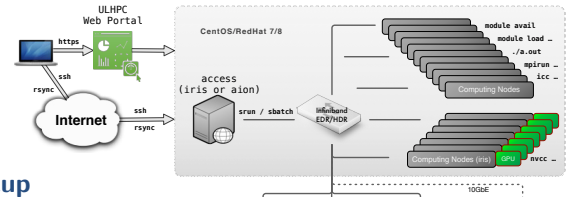
Command	Description
`module avail`	Lists all the modules which are available to be loaded
`module spider <pattern>`	Search for among available modules **(Lmod only)**
`module load <mod1> [mod2...]`	Load a module
`module unload <module>`	Unload a module
`module list`	List loaded modules
`module purge`	Unload all modules (purge)
`module use <path>`	Prepend the directory to the MODULEPATH environment variable
`module unuse <path>`	Remove the directory from the MODULEPATH environment variable

ULHPC Toolchains and Software Set Versioning

- Release based on Easybuid release of toolchains
  - ↪ see Component versions in the foss and intel toolchains
  - ↪ **Yearly** release, fix the version of sub compiler/toolchain
    - ✓ count 6 months of validation/import *after* EB release before ULHPC release

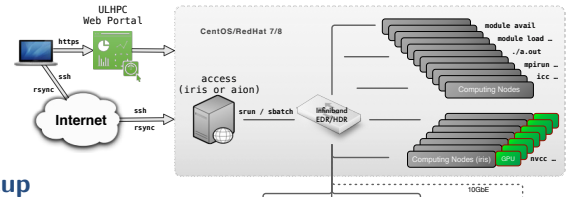
Name	**Type**	2019[a] (old)	**2019b** (prod)	**2020a** (devel)
`GCCCore`	compiler	8.2.0	8.3.0	9.3.0
`foss`	toolchain	2019a	2019b	2020a
`intel`	toolchain	2019a	2019b	2020a
`binutils`		2.31.1	2.32	2.34
LLVM	compiler	8.0.0	9.0.1	9.0.1
Python		3.7.2 (and 2.7.15)	3.7.4 (and 2.7.16)	3.8.2

Typical Workflow on UL HPC resources



- **Preliminary setup**
  1. Connect to the frontend — `ssh, screen`
  2. Synchronize you code — `scp/rsync/svn/git`
  3. Reserve a few interactive resources — `srun -p interactive [...]`
     - ✓ *(eventually)* build your program — `gcc/icc/mpicc/nvcc..`
     - ✓ Test on small size problem — `srun/python/sh...`
     - ✓ Prepare a launcher script — `<launcher>.{sh|py}`

Typical Workflow on UL HPC resources



- **Preliminary setup**
  1. Connect to the frontend — `ssh, screen`
  2. Synchronize you code — `scp/rsync/svn/git`
  3. Reserve a few interactive resources — `srun -p interactive [...]`
     - ✓ *(eventually)* build your program — `gcc/icc/mpicc/nvcc..`
     - ✓ Test on small size problem — `srun/python/sh...`
     - ✓ Prepare a launcher script — `<launcher>.{sh|py}`
- **Real Experiment**
  1. Reserve passive resources — `sbatch [...] <launcher>`
  2. Grab the results — `scp/rsync/svn/git ..`

Summary

General Guidelines

Acceptable Use Policy (AUP) 2.0

`Uni.lu-HPC-Facilities_Acceptable-Use-Policy_v2.0.pdf`

- **UL HPC is a shared** *(and expansive)* **facility: you must practice good citizenship**
  - ↪ **Users are accountable for their actions**
    - ✓ Users are allowed **one account per person** - **user credentials sharing is strictly prohibited**
    - ✓ Use of UL HPC computing resources for personal activities is prohibited
    - ✓ limit activities that may impact the system for other users.
  - ↪ **Do not abuse the shared filesystems**
    - ✓ Avoid too many simultaneous file transfers
    - ✓ regularly clean your directories from useless files
  - ↪ **Do not run programs or I/O bound processes on the login nodes**
  - ↪ Plan large scale experiments during night-time or week-ends
- Resource allocation is done on a **fair-share** principle, with **no guarantee** of being satisfied

General Guidelines

Acceptable Use Policy (AUP) 2.0

- **Data Use / GDPR**
  - ↪ **You** are responsible to ensure the appropriate level of protection, backup & integrity checks
    - ✓ Data Authors/generators/owners are responsible for its correct categorization as sensitive/non-sensitive
    - ✓ Owners of sensitive information are responsible for its secure handling, transmission, processing, storage, and disposal on the UL HPC systems
    - ✓ Data Protection inquiries can be directed to the Uni.lu Data Protection Officer
  - ↪ We make **no guarantee** against loss of data
- We provide [project] **usage report** to user/PI **on-demand** and *(by default)* on a **yearly basis**

- For **ALL** publications having results produced using the UL HPC Facility
  - ↪ **Acknowledge** the UL HPC facility and **cite** reference ULHPC article
    - ✓ using official banner
  - ↪ Tag your publication upon registration on ORBiLu.

ULHPC Websites 2.0 and Documentation

Main Website

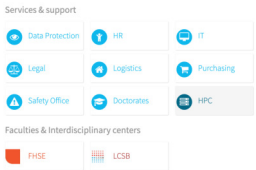
`hpc.uni.lu`

ULHPC Technical Docs

`hpc-docs.uni.lu`

ULHPC HelpDesk

`hpc.uni.lu/support`



ULHPC Tutorials

`ulhpc-tutorials.rtfd.io`

- **Fallback Support**:
  - ↪ `hpc-team@uni.lu`
  - ↪ ULHPC Community: `hpc-users@uni.lu`
    - ✓ moderated

Reporting Problems

ULHPC Helpdesk / Support Ticket Portal

- **First checks**
  1. My issue is probably documented — https://hpc-docs.uni.lu
  2. An event is on-going: **check ULHPC Live status page** — https://hpc.uni.lu/live-status/motd/
     - ✓ Planned maintenance are announced *at least* 2 weeks in advance
     - ✓ The proper SSH banner is displayed during **planned** downtime
  3. check the state of your nodes
     - ✓ `{ scontrol show job <jobid> | sjoin <jobid>}; htop` — *on active jobs*
     - ✓ `{ slist <jobid> | sacct [-X] -j <jobid> -l }` — *post-mortem*

Reporting Problems

ULHPC Helpdesk / Support Ticket Portal

- **First checks**
  1. My issue is probably documented — `https://hpc-docs.uni.lu`
  2. An event is on-going: **check ULHPC Live status page** — `https://hpc.uni.lu/live-status/motd/`
     - ✓ Planned maintenance are announced *at least* 2 weeks in advance
     - ✓ The proper SSH banner is displayed during **planned** downtime
  3. check the state of your nodes
     - ✓ `{ scontrol show job <jobid> | sjoin <jobid>}; htop` — *on active jobs*
     - ✓ `{ slist <jobid> | sacct [-X] -j <jobid> -l }` — *post-mortem*

- **ONLY NOW**, consider the following depending on the severity:
  - ↪ Open an new issue on `https://hpc.uni.lu/support` — (**preferred**)
    - ✓ Uni.lu Service Now Helpdesk Portal: relies on **Uni.lu (≠ ULHPC) credentials**
  - ↪ Mail (only now) us — `hpc-team@uni.lu`
  - ↪ **Ask the help of other users** — `hpc-users@uni.lu`

Reporting Problems

ULHPC Helpdesk / Support Ticket Portal

- **First checks**
  1. My issue is probably documented — https://hpc-docs.uni.lu
  2. An event is on-going: **check ULHPC Live status page** — https://hpc.uni.lu/live-status/motd/
     - ✓ Planned maintenance are announced *at least* 2 weeks in advance
     - ✓ The proper SSH banner is displayed during **planned** downtime
  3. check the state of your nodes
     - ✓ `{ scontrol show job <jobid> | sjoin <jobid>}; htop` — *on active jobs*
     - ✓ `{ slist <jobid> | sacct [-X] -j <jobid> -l }` — *post-mortem*

- **ONLY NOW**, consider the following depending on the severity:
  - ↪ Open an new issue on `https://hpc.uni.lu/support` — (**preferred**)
    - ✓ Uni.lu Service Now Helpdesk Portal: relies on **Uni.lu (≠ ULHPC) credentials**
  - ↪ Mail (only now) us — `hpc-team@uni.lu`
  - ↪ **Ask the help of other users** — `hpc-users@uni.lu`

In all cases: **Carefully describe the problem and the context** Guidelines

Summary

Interactive Jobs

```
# BEFORE
srun -p interactive --qos qos-interactive -C {broadwell|skylake} [...] --pty bash`
# AFTER -- match feature name with target partition ?
srun -p interactive --qos debug -C {batch,gpu,bigmem} [...] --pty bash
```

- **Before**: guaranteed access to interative jobs on regular nodes even if `batch` partition full
  - ↪ **YET** no way to use `qos-interactive` for GPU/bigmem
    - ✓ default node category QOS/partition used, inherits from default limits
    - ✓ `srun -p gpu --qos qos-gpu -G 4 [...] --pty bash` can stay 5 days in a `screen`
- **After**: no guarantee if partition is full **YET** backfilling and priority ensure first served

Node Type	Slurm command	Helper script
regular	`srun -p interactive --qos debug -C batch [-C {broadwell,skylake}] [...] --pty bash`	`si [...]`
gpu	`srun -p interactive --qos debug -C gpu [-C volta[32]] -G 1 [...] --pty bash`	`si-gpu [...]`
bigmem	`srun -p interactive --qos debug -C bigmem [...] --pty bash`	`si-bigmem [...]`

Regular Jobs

- **NO MORE `qos-*` QOS**
  - ↪ **ALL slurm launchers to review to remove/adapt QOS attributes**
  - ↪ all default to `normal` QOS, **except** CRP/externals who default to `low`
  - ↪ thus no need to precise, except to access higher priority QOS if allowed
    - ✓ Ex: `#SBATCH --qos high`
- **NEW: Add `-A <project|lecture>` account when appropriate!**
  - ↪ Non-default L3 meta-account used:
    - ✓ project name `<project>`
    - ✓ lecture/course name: `<lecture>`

```
   #SBATCH -p batch                #SBATCH -p gpu                #SBATCH -p bigmem
-- #SBATCH --qos qos-batch      -- #SBATCH --qos qos-gpu      -- #SBATCH --qos qos-bigmem
++ #SBATCH -A <project>         ++ #SBATCH -A <project>       ++ #SBATCH -A <project>
```

Regular Jobs

- Relatively similar as before, **YET** now restricted to **Max 2 days** / Max 64 nodes
  - ↪ walltime reduction would have affected 1.22% of the jobs completed since July, 1st 2020
  - ↪ default QOS induced by the `job_submit.lua` plugin as before
  - ↪ enforce precision of project/training account (`-A <account>`)

Node Type	Slurm command
regular	`sbatch [-A <project>] -p batch  [--qos {high,urgent}] [-C {broadwell,skylake}] [...]`
gpu	`sbatch [-A <project>] -p gpu  [--qos {high,urgent}] [-C volta[32]] -G 1 [...]`
bigmem	`sbatch [-A <project>] -p bigmem [--qos {high,urgent}] [...]`

- Slurm Federation configuration between `iris` and `aion`
  - ↪ ensures global policy (coherent job ID, global scheduling, etc.) within ULHPC systems
  - ↪ easily submit jobs from one cluster to another `-M, --cluster aion|iris`

```
# Ex (from iris): try first on iris, then on aion
sbatch -p batch -M iris,aion [...]
```

Long Jobs

```
# BEFORE - only on regular nodes
sbatch -p long --qos qos-long [...]
# AFTER -- select target partition to bypass default walltime restrictions
sbatch -p {batch | gpu | bigmem} --qos long [...]
```

- **Before**: extended Max walltime (`MaxWall`) set to **30 days, restricted to regular nodes**
  - ↪ Max 6 nodes, Max 2 nodes per Job, Max 10 Jobs per User
  - ↪ No way to run long jobs on GPU or Large-Memory nodes
- **After**: extended Max walltime (`MaxWall`) set to **14 days** EuroHPC/PRACE Recommendations
  - ↪ Max 6 nodes, Max 2 nodes per Job, Max **1** Job per User

Node Type	Slurm command
regular	`sbatch [-A <project>] -p batch  --qos long [-C {broadwell,skylake}] [...]`
gpu	`sbatch [-A <project>] -p gpu  --qos long [-C volta[32]] -G 1 [...]`
bigmem	`sbatch [-A <project>] -p bigmem --qos long [...]`

Other Misc Changes

- (complex) Depth-Oblivious Fairshare $\Longrightarrow$ **Fair tree** Algorithm
- Special **preemptible QOS** kept for **best-effort Jobs** **YET** renamed: ~~qos-~~`besteffort`
  - ↪ `sbatch -p {batch | gpu | bigmem} --qos besteffort [...]`
- **NO MORE dedicated QOS** `qos-batch-00*` but global **restricted** `high`**(priority) QOS**
  - ↪ Incentives for User groups/Projects contributing to the HPC budget line
    - ✓ **Updated every year** based on past funding amount and depreciation (default: 12 months)
    - ✓ Affect raw share for the L2/L3 account

$$FundingScore(Year) = \left\lfloor \alpha_{\text{level}} \times \frac{Investment(Year-1)}{100 \times \#months} \right\rfloor$$

- **Restricted** `urgent` **QOS** for ultra-high priority jobs (Ex: covid-19)
- End-User raw-share increased based on past year efficiency
  - ↪ Efficiency Score for L4 user, **Average Wall-time Accuracy (WRA)**

Questions?

http://hpc.uni.lu

High Performance Computing @ Uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette
Sarah Peter
Hyacinthe Cartiaux
Dr. Frederic Pinel
Dr. Emmanuel Kieffer
Dr. Ezhilmathi Krishnasamy
Teddy Valette
Abatcha Olloh
University of Luxembourg, Belval Campus:
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu