Uni.lu High Performance Computing (ULHPC) Facility

User Guide, 2022

UL HPC Team

https://hpc.uni.lu

Summary

Summary

Uni.lu HPC (UL HPC) Facility

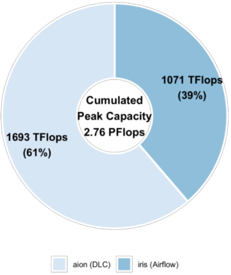
- **Managed and operated since 2007** *(by Digital Platforms Team)*
  - ↪ 2nd Largest HPC facility in Luxembourg after EuroHPC MeluXina
  - ↪ Team led by Dr. S. Varrette until Aug. 2022 (Now: H. Cartiaux)

hpc.uni.lu

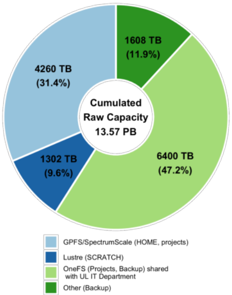
Technical Docs:
hpc-docs.uni.lu

ULHPC Tutorials:
ulhpc-tutorials.rtfd.io

UL HPC Supercomputers (2022)



UL HPC Storage FileSystems (2022)



Uni.lu HPC (UL HPC) Facility

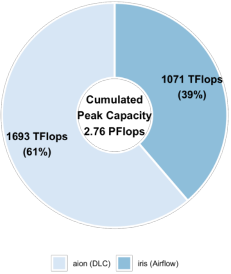
- **Managed and operated since 2007** *(by Digital Platforms Team)*
  - ↪ 2nd Largest HPC facility in Luxembourg after EuroHPC MeluXina
  - ↪ Team led by Dr. S. Varrette until Aug. 2022 (Now: H. Cartiaux)

`hpc.uni.lu`

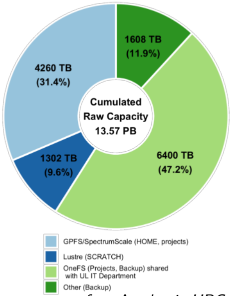
Technical Docs:
`hpc-docs.uni.lu`

ULHPC Tutorials:
`ulhpc-tutorials.rtfd.io`

UL HPC Supercomputers (2022)



UL HPC Storage FileSystems (2022)



[HPCCT22] S. Varrette, H. Cartiaux, S. Peter, E. Kieffer, T. Valette, and A. Olloh. "*Management of an Academic HPC & Research Computing Facility: The ULHPC Experience 2.0*". In 6th HPC and Cluster Technologies Conference (ACM HPCCT 2022) doi – orbilu

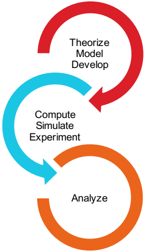
HPC in Luxembourg and Around in EU

Tier 0: EU
Tier 1: National
Tier 2: Regional | Univ.

Country	System(s)	Type	Institute	(CPU) #Nodes	#Cores	#[GPU]Accelerators	R_{peak}	Shared Storage
Luxembourg	MeluXina (2021)	Euro-HPC **Petascale** *Tier 0/1 (EU,Nat)*	LuxProvide	824	≃ 88 000	764 NVidia A100	**17.57 PF**	≃ 20 PB
	aion,iris	*Tier 2 (Univ)*	**Uni.lu HPC**	**552**	46896	**96** NVidia V100	**2.77 PF**	**10.71 PB**
		Tier 2 (local)	LIST	40	1280	8 Nvidia V100	0.126 PF	0.58 PB
France	TGCC (Joliot-Curie)	*Tier 0 (EU)*	GENCI/CEA	4808	430 448	828 Xeon Phi, 128 NVidia V100	22.26 PF	35PB
	JeanZay	*Tier 1 (Nat.)*	GENCI/Idris	1 528	61 120	1292 NVidia V100	14.97 PF	31.2 PB
	ROMEO	*Tier 2 (Reg.)*	Univ. Reims	115	3 220	280 NVidia P100	1.75 PF	0.634
Belgium	Vlaams	*Tier 1 (Nat.)*	VSC	988	27 664	n/a	1.63 PF	1.3PB
	zenobe	*Tier 1 (Nat.)*	Cenaero	584	14 016	4 NVidia K40	0.41 PF	0.356PB
	Hortense	*Tier 2 (Reg.)*	Gent Univ.	n/a	≃ 40 000	88 NVidia V100	3.3PF	3PB
Germany	JUWELS	*Tier 0 (EU)*	JSC	2571	122 768	224 Nvidia V100	12.3 PF	130.3PB
	JURECA	*Tier 0 (EU)*	JSC	3524	156 736	1640 Xeon Phi	7.24 PF	(as above)
	Hawk	*Tier 0 (EU)*	HLRS, Univ. Stuttgart	5632	720 896	n/a	26 PF	≃25PB
	SuperMUC-NG	*Tier 0 (EU)*	LRZ, Munich	6480	311 040	n/a	26.9 PF	70.16PB
	CLAIX-2018	*Tier 2 (Univ)*	Univ. Aachen	1307	61 200	108 Nvidia V100	4.11 PF	3PB
Bulgaria	PetaSC (2021)	Euro-HPC **Petascale** *Tier 0/1 (EU,Nat)*	SofiaTech	n/a	n/a	n/a	**4.5 PF**	n/a
Czech Republic	Barbora	*Tier 1 (Nat.)*	IT4Innovation	201	7232	32 NVidia V100	0.85 PF	≃ 1PB
	Karolina (2021)	Euro-HPC **Petascale** *Tier 0/1 (EU,Nat)*	IT4Innovation	826	≃ 100K	560 NVidia A100	**9.4 PF**	1PB
Finland	LUMI (2021)	Euro-HPC **Pre-exascale** *Tier 0 (EU)*	CSC	n/a	≃ 200K (LUMI-C)	n/a	**375 PF**	127PB
Italy	Marconi-A3	*Tier 0 (EU)*	Cineca	3216	154 368	n/a	10.37 PF	10PB
	Galileo	*Tier 1 (Nat.)*	Cineca	1022	36792	n/a	1.35 PF	1.92PB
	Leonardo (2021)	Euro-HPC **Pre-exascale** *Tier 0 (EU)*	Cineca	4992	n/a	13824 Nvidia A100	**249.5 PF**	100PB
Portugal	Deucalion (2021)	Euro-HPC **Petascale** *Tier 0/1 (EU,Nat)*	MACC	n/a	n/a	n/a	**7.2 PF**	n/a
Slovenia	VEGA (2021)	Euro-HPC **Petascale** *Tier 0/1 (EU,Nat)*	Maribor SC	960	122,8K	240 NVidia A100	**10.1 PF**	24 PB
Spain	MareNostrum 4	*Tier 0 (EU)*	BSC	3456	165 888	n/a	11.15 PF	14PB
	MareNostrum 5 (2021)	Euro-HPC **Pre-exascale** *Tier 0 (EU)*	BSC	n/a	n/a	n/a	**≃ 200 PF**	n/a
Switzerland	Piz-Daint	*Tier 0 (EU)*	CSCS, ETH Zürich	7517	387 872	5704 NVidia P100	29.34 PF	8.8PB

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>

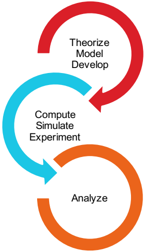


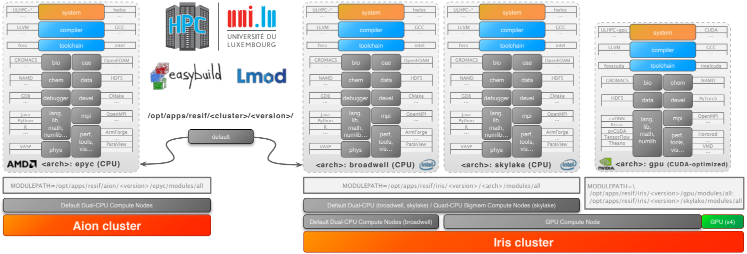
	Software set release <version>		
Component	**2019b** legacy	**2020b** prod	**2021b** devel
binutils	2.32	2.35	2.37
GCCCore	8.3.0	10.2.0	11.2.0
foss	2019b	2020b	2021b
- OpenMPI	3.1.4	4.0.5	4.1.2
intel	2019b	2020b	2021a
- Compilers/MKL	2019.5.281	2020.1.217	2021.4.0
- Intel MPI	2018.5.288	2019.7.217	2021.4.0
Python	3.7.4	3.8.6	3.9.6
RESIF version	3.0	3.0	3.1
#Software Modules	<arch>: 269 gpu: 135	<arch>: 274 gpu: 151	<arch>: 282 gpu: 157

[PEARC21] S. Varrette, E. Kieffer, F. Pinel, E. Krishnasamy, S. Peter, H. Cartiaux, and X. Besseron. "*RESIF 3.0: Toward a Flexible & Automated Management of User Software Environment on HPC facility*". In ACM Practice & Experience in Advanced Research Computing (PEARC'21) pdf – code

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>

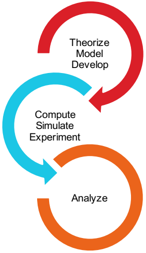


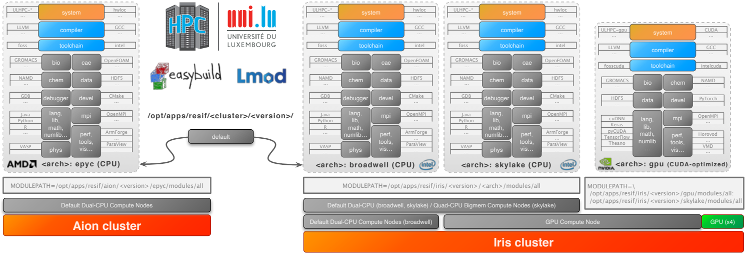


[PEARC21] S. Varrette, E. Kieffer, F. Pinel, E. Krishnasamy, S. Peter, H. Cartiaux, and X. Besseron. "*RESIF 3.0: Toward a Flexible & Automated Management of User Software Environment on HPC facility*". In ACM Practice & Experience in Advanced Research Computing (PEARC'21) pdf – code

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>





  - ↪ containerized applications delivered with Singularity system
  - ↪ user web/application portal (ouside regular SSH access): Open OnDemand

Uni.lu Data Center

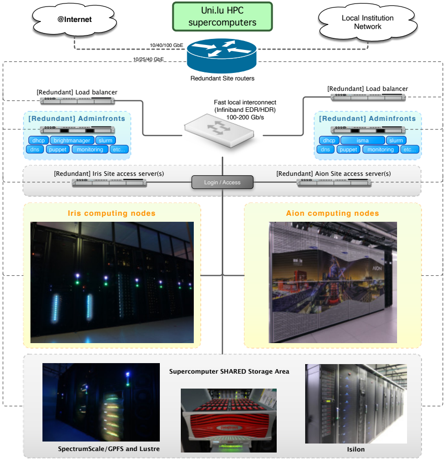
Belval Campus

Centre De Calcul (CDC)

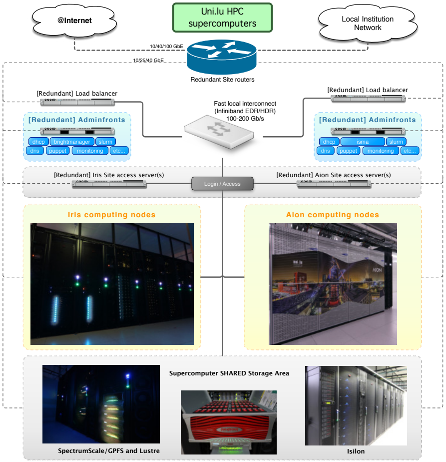
- Power generation station for HPC floor:
  - ↪ up to **3 MW of electrical power**
  - ↪ **2.4 MW of cold water** at a 12-18°C regime
    - ✓ used for traditional Airflow with In-Row cooling.
  - ↪ Separate hot water circuit (between 30 and 40°C)
    - ✓ used for Direct Liquid Cooling (DLC): `aion`

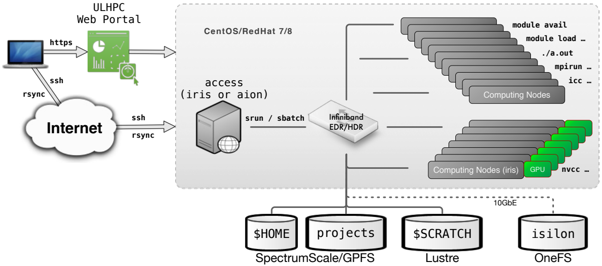
Location	Cooling	Usage
CDC S-02-001	Airflow	**Future extension**
CDC S-02-002	Airflow	**Future extension**
CDC S-02-003	DLC	**Future extension** - High Density/Energy efficient HPC
CDC S-02-004	**DLC**	**High Density/Energy efficient HPC**: `aion`
CDC S-02-005	**Airflow**	**Storage / Traditional HPC**: `iris` and common equipment

UL HPC Supercomputers: General Architecture



UL HPC Supercomputers: General Architecture



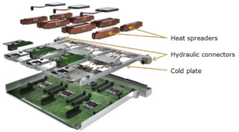


UL HPC Supercomputers: `iris` cluster

`hpc-docs.uni.lu/systems/iris/`

- **Dell/Intel** supercomputer *Air-flow cooling*
  - ↪ 196 compute nodes, **5824 cores**, 52.2 TB RAM
  - ↪ R_{peak}: **1,07 PetaFlop/s**
    - ✓ **regular** nodes (Dual CPU, 128 to 256 GB of RAM)
    - ✓ **GPU** nodes (Dual CPU, 4 NVidia accelerators, 768 GB RAM)
    - ✓ **Large-memory** nodes (Quad-CPU, 3072 GB RAM)
- Fast InfiniBand (IB) EDR network
  - ↪ **Fat-Tree** Topology blocking factor 1:1.5
- Stepwise deployment since 2017
  - ↪ two major upgrade phases (2018 and 2019)

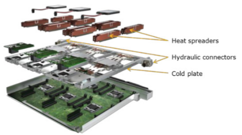
UL HPC Supercomputers: aion cluster



hpc-docs.uni.lu/systems/aion/

- **Atos/AMD** supercomputer, DLC cooling
  - ↪ 4 BullSequana XH2000 adjacent racks
  - ↪ 318 **regular** nodes, **40704 cores**, 81.4 TB RAM
  - ↪ R_{peak}: **1,693 PetaFLOP/s**
- Fast InfiniBand (IB) HDR network
  - ↪ **Fat-Tree** Topology — blocking factor 1:2
- Acquisition by European Tender in 2020
  - ↪ **production release** in Oct 2021 — (delayed by COVID)

UL HPC Supercomputers: aion cluster



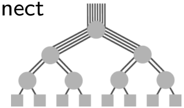
hpc-docs.uni.lu/systems/aion/

- **Atos/AMD** supercomputer, DLC cooling **(EOY update)**
  - ↪ 4 BullSequana XH2000 adjacent racks
  - ↪ 354 **regular** nodes, **45312 cores**, 90.6 TB RAM
  - ↪ R_{peak}: **1,885 PetaFLOP/s**
- Fast InfiniBand (IB) HDR network
  - ↪ **Fat-Tree** Topology — blocking factor 1:2
- Acquisition by European Tender in 2020
  - ↪ **production release** in Oct 2021 — (delayed by COVID)
  - ↪ **First upgrade EOY 2022** — +36 **regular** nodes

Interconnect Networks (Infiniband and Ethernet)

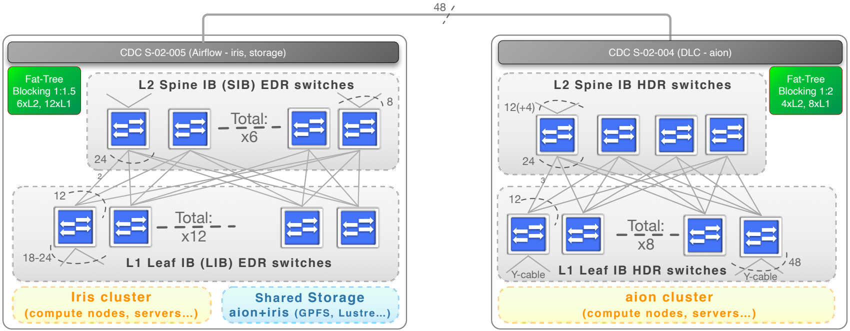
- HPC interconnect technologies nowadays divided into three categories
  1. **Ethernet**: dominant interconnect standard yet underlying protocol has inherent limitations
     - ✓ preventing low-latency deployments expected in real HPC environment
  2. **InfiniBand**: predominant interconnect technology in the HPC market
  3. Vendor specific interconnects: Cray/HPC Slingshot, Intel Omni-Path, Bull BXI...

- **On ULHPC Supercomputers**:
  - ↪ **InfiniBand (IB)** in a **Fat-Tree** Topology as *Ultra-Fast* local interconnect
    - ✓ `iris`: IB **EDR** Fabric
    - ✓ `aion`: IB **HDR100** Fabric
  - ↪ Complementary **Ethernet network**
    - ✓ Consolidated as a 2-layers topology (Gateway / Switching Layers)



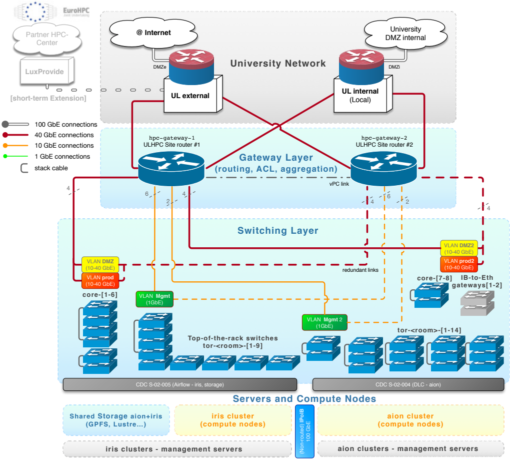
Fast Local Infiniband Interconnect Network

hpc-docs.uni.lu/interconnect/ib/



[PEARC22] S. Varrette, H. Cartiaux, T. Valette and A. Olloh, "*Aggregating and Consolidating two High Performant Network Topologies: The ULHPC Experience*" in ACM Practice and Experience in Advanced Research Computing (PEARC'22). pdf – orbilu

Complementary Ethernet Network



`hpc-docs.uni.lu/interconnect/ethernet/`

- Flexibility of Ethernet-based networks still required
- **2-layers** topology
  - ↪ *Upper* level: **Gateway Layer**
    - ✓ routing, switching features, network isolation and filtering (ACL) rules
    - ✓ meant to interconnect only switches.
    - ✓ allows to interface University network (LAN/WAN)
  - ↪ *bottom* level: **Switching Layer**
    - ✓ [stacked] core switches
    - ✓ TOR (Top-the-rack) switches
    - ✓ meant to interface HPC servers and compute nodes

[PEARC22] S. Varrette, H. Cartiaux, T. Valette and A. Olloh, "*Aggregating and Consolidating two High Performant Network Topologies: The ULHPC Experience*" in ACM Practice and Experience in Advanced Research Computing (PEARC'22). pdf – orbilu

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, ...

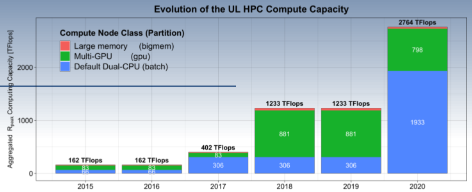
Summary

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

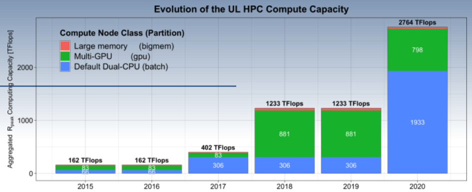
Slurm on ULHPC clusters



- 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

[ISPDC22] S. Varrette, E. Kieffer and F. Pinel, "*Optimizing the Resource and Job Management System of an Academic HPC and Research Computing Facility*" in 21st IEEE Intl. Symp. on Parallel and Distributed Computing (ISPDC'22) – orbilu

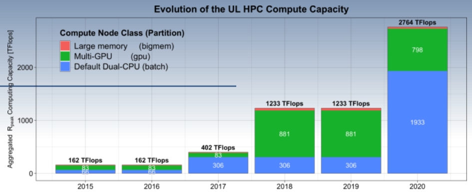
Slurm on ULHPC clusters



- 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`.

[ISPDC22] S. Varrette, E. Kieffer and F. Pinel, "*Optimizing the Resource and Job Management System of an Academic HPC and Research Computing Facility*" in 21st IEEE Intl. Symp. on Parallel and Distributed Computing (ISPDC'22) – orbilu

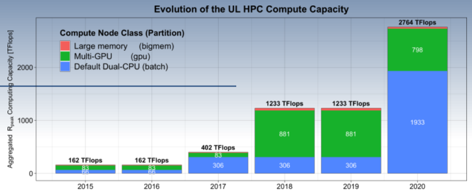
Slurm on ULHPC clusters



- 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

[ISPDC22] S. Varrette, E. Kieffer and F. Pinel, "*Optimizing the Resource and Job Management System of an Academic HPC and Research Computing Facility*" in 21st IEEE Intl. Symp. on Parallel and Distributed Computing (ISPDC'22) – orbilu

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 Multi-cluster configuration between `iris` and `aion`
  - ↪ easily **query** state on remote cluster `{squeue...} -M, --cluster aion|iris`

[ISPDC22] S. Varrette, E. Kieffer and F. Pinel, "*Optimizing the Resource and Job Management System of an Academic HPC and Research Computing Facility*" in 21st IEEE Intl. Symp. on Parallel and Distributed Computing (ISPDC'22) – orbilu

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>
```

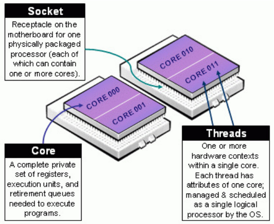
Submitting Jobs

- **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.

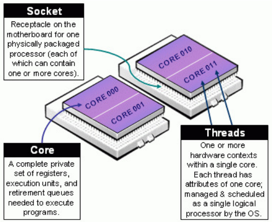
Specific Resource Allocation

- **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>`



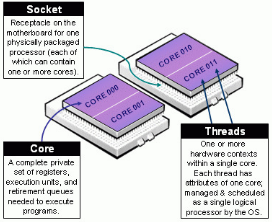
Specific Resource Allocation

- **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 align resource specs with physical NUMA characteristics**
  - ↪ **Ex (AION): 16 cores per socket, 8 sockets ("physical" CPUs) per node** (128c/node)
  - ↪ `[-N <N>] --ntasks-per-node <8n> --ntasks-per-socket <n> -c <thread>`
    - ✓ **Total**: `<N>`×8×`<n>` tasks, each on `<thread>` threads
    - ✓ **Ensure** `<n>`×`<thread>`= 16 on aion
    - ✓ Ex: `-N 2 --ntasks-per-node 32 --ntasks-per-socket 4 -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} [...]`

- **Important: Always align resource specs with physical NUMA characteristics**
  - ↪ **Ex (IRIS): 14 cores per socket, 2 sockets ("physical" CPUs) per node** (28c/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>`= 14 on iris
    - ✓ Ex: `-N 2 --ntasks-per-node 4 --ntasks-per-socket 2 -c 7` (**Total**: 8 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	8	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: `sfeatures`

```
# OR: sinfo -o '%20N %.6D %.6c %15F %12P %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-node=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			50	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 [Generic] Launchers 2.0

```bash
#!/bin/bash -l             # <--- DO NOT FORGET '-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] [...]
```

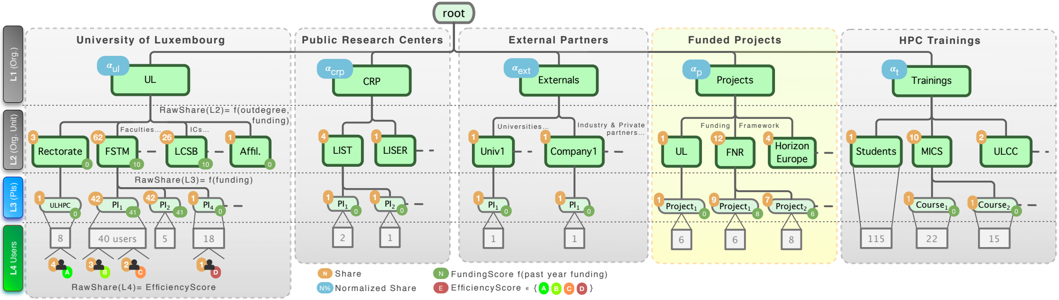
- **Best-Practices**
  - ↪ use `/bin/bash -l` on top
  - ↪ set **reasonable time limits**
  - ↪ set (short) job name
  - ↪ specify account
  - ↪ `--exclusive` allocation?
  - ↪ **Avoid** Job arrays
  - ↪ consider singleton pipelining
    - ✓ job dep. made easy
  - ↪ GPU jobs (gpu partition)
    - ✓ Set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

Account Hierarchy 2.0

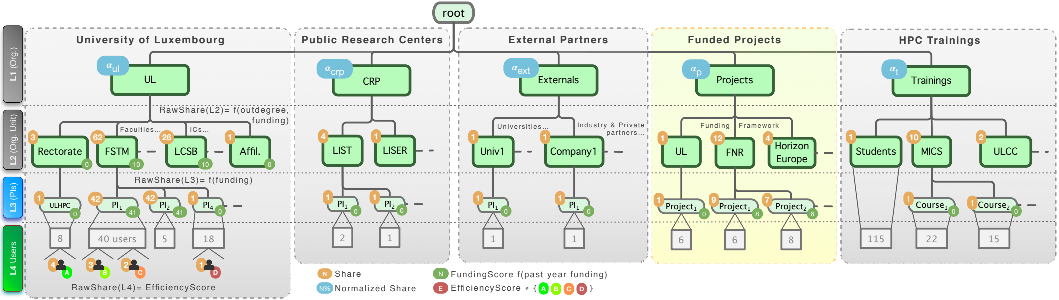
- Every user job runs under a group account (default: PI/line manager)
  - ↪ granting access to specific QOS levels, default raw share for accounts: 1
  - ↪ you **MUST** request a dedicated **Slurm project account** for accounting monitoring
    - ✓ HPC Support (Service Now) / HPC / Storage & projects / New Slurm project account

- **L1**: Organization Level: UL, CRPs, Externals, Projects, Trainings
  - ↪ guarantee 85% of the shares for core UL activities
- **L2**: Organizational Unit (Faculty, ICs, External partner, Funding program...)
  - ↪ Raw share depends on **outdegree** and **funding score**
- **L3**: Principal Investigator (PIs), Projects, Course
  - ↪ Raw share depends on **funding score** (different weight)
  - ↪ Eventually restricted **only** to projects and courses
- **L4**: End User (ULHPC login)
  - ↪ Raw share based on **efficiency score**

Account Hierarchy 2.0

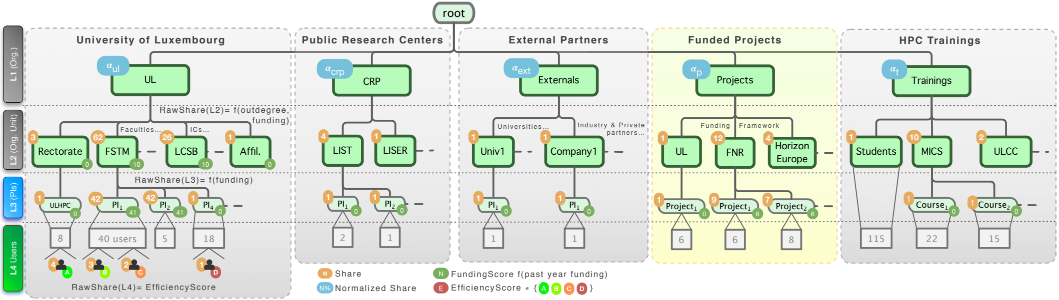


Account Hierarchy 2.0




```
# /!\ ADAPT <name> accordingly
sassoc <name>
```

Account Hierarchy 2.0




```
# Regular submission as End user L4
{srun | sbatch | salloc } [...]
# Accounting associated to **project account** <project> -- required for auditing
{srun | sbatch | salloc } -A <project> [...]
```

Funding Score (L2/L3)

- Associated with an account A belonging to a level L in the hierarchy
  - $\hookrightarrow$ yearly updated at the beginning of the year
  - $\hookrightarrow$ depreciation based on contribution type, weighted by level threshold β_L

$$\texttt{FundingScore}_L(A) = \left\lfloor \beta_L \frac{Investment_A(Year - 1)}{\#months} \right\rfloor$$

Funding Score (L2/L3)

- Associated with an account A belonging to a level L in the hierarchy
  - ↪ yearly updated at the beginning of the year
  - ↪ depreciation based on contribution type, weighted by level threshold β_L

$$\texttt{FundingScore}_L(A) = \left\lfloor \beta_L \frac{Investment_A(Year-1)}{\#months} \right\rfloor$$

- **Ex1**: Exceptional contribution of 120K€ performed in 2021 by a faculty (L2 account A)
  - ↪ depreciation: 12 months (*default*)
  - ↪ **funding score in 2022**: $\left\lfloor \beta_{L_2} \frac{120000}{12} \right\rfloor = \lfloor \beta_{L_2} \times 10000 \rfloor$.
- **Ex2**: let P be a project granted in 2021 to start in 2022 for a duration of 36 months
  - ↪ **budget**: 27K€ allocated for HPC costs
  - ↪ **funding score for the years 2022, 2023 and 2024**: $\left\lfloor \beta_{L_3} \frac{27000}{36} \right\rfloor = \lfloor \beta_{L_3} \times 750 \rfloor$

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:

$$
\begin{aligned}
S_{\text{efficiency}}(U, Year) &= \text{WRA}(U, Year) \\
&= \frac{1}{N} \sum_{JobID} \frac{T_{\text{elapsed}}(JobID)}{T_{\text{asked}}(JobID)}
\end{aligned}
$$

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:

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

- U raw share: $1 + S_{\text{efficiency}}(U, Year)$

Score	Avg. WRA
A (3) very good	$S_{\text{efficiency}} \geq 75\%$
B (2) good	$50\% \leq S_{\text{efficiency}} < 75\%$
C (1) bad	$25\% \leq S_{\text{efficiency}} < 50\%$
D (0) very bad	$S_{\text{efficiency}} < 25\%$

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:

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

- U raw share: $1 + S_{\text{efficiency}}(U, Year)$

Score	Avg. WRA
A (3) very good	$S_{\text{efficiency}} \geq 75\%$
B (2) good	$50\% \leq S_{\text{efficiency}} < 75\%$
C (1) bad	$25\% \leq S_{\text{efficiency}} < 50\%$
D (0) very bad	$S_{\text{efficiency}} < 25\%$

- **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
- **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
```

ULHPC Job Prioritization Factors

- **Age**: length of time a job has been waiting (PD state) in the queue
- **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
```

```
# Show current weights
sprio -w
# List pending jobs, sorted by jobid
sprio [-n]
# List pending jobs, sorted by  priority
sprio [-n] -S+Y
sprio [-n] | sort -k 3 -n
sprio [-n] -l | sort -k 4 -n
```

hpc-docs.uni.lu/policies/usage-charging/

ULHPC Cost Model

- ULHPC **free of charge for UL staff** for their **internal work and training activities**
  - ↪ Yet data storage extension **above** default capacity is charged
- HPC Resource Allocations for **funded research project** **MUST** comply with ULHPC Cost Model policy approved July 7, 2020 (Rectorate, FNR).
  - ↪ You must prepare your budget plan to support HPC costs within your FNR project proposals
    - ✓ UL HPC Cost Budget Plans for Project Proposals [xlsx] provided for help
- **When charged** ULHPC computing resources are billed as follows:
  - ↪ project PI will receive a usage report
  - ↪ **new** SAP Workflow to facilitate auditing and charging from HPC usage report

Cluster	**Node Type**	**#Cores/node [#GPUs]**	Billing Rate	**Hourly Price [€, HT]**
Aion	Regular	128	200.96	6.03€
Iris	Regular	28	56	1.68€
Iris	GPU	28 [+4 NVidia V100]	256	7.68€
Iris	Large-Mem	112	224	6.72€

hpc-docs.uni.lu/policies/usage-charging/

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

Job Billing and Budget Planning

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

```
scontrol show job <jobID> | grep -i billing   # running job
#  Billing rate for completed job <jobID>
sbill <jobID>
```

Job Billing and Budget Planning

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**

Job Billing and Budget Planning

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**

Job Billing and Budget Planning

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**

Job Billing and Budget Planning

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**

Job Billing and Budget Planning

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

- **Not able to anticipate the type and amount of resources needed?**
  - ↪ we suggest a simple rule based on the total number of funded persons
    - ✓ account 5529.60€ for every 12 PM of funded personnel

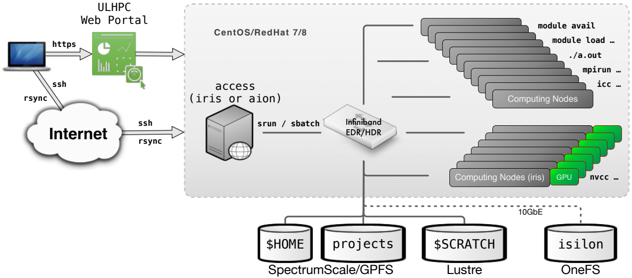
guidelines

- **In all cases**: contact UL research support / facilitators for help and guidance

Summary

1. High Performance Computing (HPC) @ UL
2. Batch Scheduling Configuration
3. **User [Software] Environment**
4. Usage Policy

Compute Nodes / Storage Environment

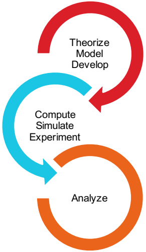


- **Storage usage**: `df-ulhpc [-i]`
  - ↪ `$HOME`: regular backup policy
  - ↪ `$SCRATCH` **NO** backup & purged
    - ✓ 60 days retention policy
  - ↪ Project quotas attached to group
    - ✓ **not** *(default)* `clusterusers` group
    - ✓ Commands writing in project dir: `sg <group> -c "<command>"`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

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>

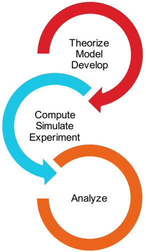


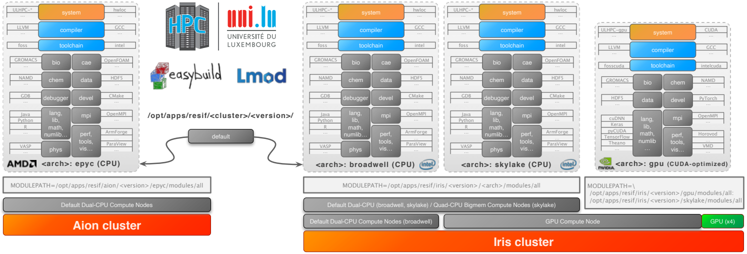
	Software set release <version>		
Component	**2019b** legacy	**2020b** prod	**2021b** devel
binutils	2.32	2.35	2.37
GCCCore	8.3.0	10.2.0	11.2.0
foss	2019b	2020b	2021b
- OpenMPI	3.1.4	4.0.5	4.1.2
intel	2019b	2020b	2021a
- Compilers/MKL	2019.5.281	2020.1.217	2021.4.0
- Intel MPI	2018.5.288	2019.7.217	2021.4.0
Python	3.7.4	3.8.6	3.9.6
RESIF version	3.0	3.0	3.1
#Software Modules	<arch>: 269 gpu: 135	<arch>: 274 gpu: 151	<arch>: 282 gpu: 157

[PEARC21] S. Varrette, E. Kieffer, F. Pinel, E. Krishnasamy, S. Peter, H. Cartiaux, and X. Besseron. "*RESIF 3.0: Toward a Flexible & Automated Management of User Software Environment on HPC facility*". In ACM Practice & Experience in Advanced Research Computing (PEARC'21) pdf – code

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>

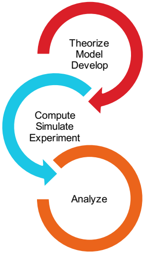


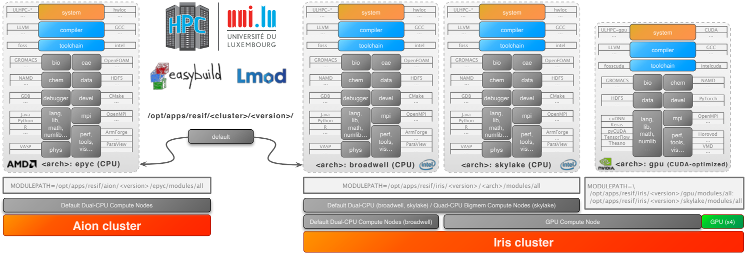


[PEARC21] S. Varrette, E. Kieffer, F. Pinel, E. Krishnasamy, S. Peter, H. Cartiaux, and X. Besseron. "*RESIF 3.0: Toward a Flexible & Automated Management of User Software Environment on HPC facility*". In ACM Practice & Experience in Advanced Research Computing (PEARC'21) pdf – code

Accelerating Research - User Software Sets

- **Over 280 software packages** available for researchers
  - ↪ software environment generated using **RESIF 3.0 framework** [PEARC21] over Easybuild
    - ✓ **optimized builds** organized by architecture, exposed through Environment Modules/Lmod
    - ✓ Categorized Naming Scheme <category>/<name>/<version>-<toolchain><versionsuffix>





  - ↪ containerized applications delivered with Singularity system
  - ↪ user web/application portal (ouside regular SSH access): Open OnDemand

`module` Command Usage

- **ONLY available on compute nodes**
  - ↪ restrict mistaken runs on access frontends
  - ↪ software module categories accessible with `eb --show-default-moduleclasses`

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

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

Software/Modules Management

- **Key `module` variable: `$MODULEPATH`**: where to look for modules!
  - ↪ **Format:** `/opt/apps/resif/<cluster>/<version>/<arch>/modules/all`
  - ↪ `<cluster>` depicts the name of the cluster (`iris` or `aion`). `$ULHPC_CLUSTER`
  - ↪ `<version>`: ULHPC Software set release `$RESIF_VERSION_{PROD,DEVEL,LEGACY}`
    - ✓ aligned with Easybuid toolchains release), **i.e.** 2019b, 2020b etc.
  - ↪ `<arch>`: identify CPU architecture or GPU node. `$RESIF_ARCH`
    - ✓ Intel nodes: `broadwell` (**default**), `skylake`
    - ✓ AMD nodes: `epyc` (**default**)
    - ✓ GPU nodes: `gpu`

Cluster	**Arch.** `$RESIF_ARCH`	`$MODULEPATH` **Environment variable**
Iris	`broadwell` (default)	`/opt/apps/resif/iris/<version>/broadwell/modules/all`
Iris	`skylake`	`/opt/apps/resif/iris/<version>/skylake/modules/all`
Iris	`gpu`	`/opt/apps/resif/iris/<version>/gpu/modules/all`
Aion	`epyc` (default)	`/opt/apps/resif/aion/<version>/{epyc}/modules/all`

ULHPC Toolchains and Software Set Versioning

- **Yearly** release based on Easybuild release of toolchains
  - ↪ see Component versions (**fixed per release**) in the foss and intel toolchains
    - ✓ count 6 months of validation/import *after* EB release before ULHPC release

Name	Type	**2019b** (legacy)	**2020a**	**2020b** (prod)	**2021a**	**2021b** (devel)
GCCCore	compiler	8.3.0	9.3.0	10.2.0	10.3.0	11.2.0
foss	toolchain	2019b	2020a	2020b	2021a	2021b
intel	toolchain	2019b	2020a	2020b	2021a	2021b
binutils		2.32	2.34	2.35	2.36	2.37
Python		3.7.4 (and 2.7.16)	3.8.2 (and 2.7.18)	3.8.6	3.9.2	3.9.6
LLVM	compiler	9.0.1	10.0.1	11.0.0	11.1.0	12.0.1
OpenMPI	MPI	3.1.4	4.0.3	4.0.5	4.1.1	4.1.2

```
# test (new) development software set
resif-load-swset-devel
# Restore production settings
resif-load-swset-prod
```

ULHPC Software Sets in RESIF 3

- **User Software Sets** now defined as native Easybuild Module `Bundle` easyblock
  - ↪ **ULHPC** bundles, associated to toolchain version – see `easyconfigs/u/ULHPC*`

Bundle Name	Description	Featured applications
`ULHPC-<version>`	Default global bundle for 'regular' nodes	`ULHPC-*-<version>` (root bundle)
`ULHPC-toolchains-<version>`	Toolchains, compilers, debuggers, programming languages, MPI suits, Development tools and libraries	GCCcore, foss, intel, LLVM, OpenMPI, CMake, Go, Java, Julia, Python, Spack...
`ULHPC-bd-<version>`	Big Data	Apache Spark, Flink, Hadoop...
`ULHPC-bio-<version>`	Bioinformatics, biology and biomedical	GROMACS, Bowtie2, TopHat, Trinity...
`ULHPC-cs-<version>`	Computational science, incl. CAE, CFD, Chemistry, Earth Sciences, Physics and Materials Science	ANSYS, OpenFOAM, ABAQUS, NAMD, GDAL, QuantumExpresso, VASP...
`ULHPC-dl-<version>`	AI / Deep Learning / Machine Learning	TensorFlow, PyTorch, Horovod...
`ULHPC-math-<version>`	High-level mathematical software and Optimizers	R, MATLAB, CPLEX, GEOS, GMP, Gurobi...
`ULHPC-perf-<version>`	Performance evaluation / Benchmarks	ArmForge, PAPI, HPL, IOR, Graph500...
`ULHPC-tools-<version>`	General purpose tools	DMTC, Singularity, gocryptfs...
`ULHPC-visu-<version>`	Visualization, plotting, documentation & typesetting	OpenCV, ParaView...
`ULHPC-gpu-<version>`	Specific GPU/CUDA-accelerated software	{foss,intel}cuda, cuDNN, TensorFlow, PyTorch, GROMACS...

[PEARC21] S. Varrette, E. Kieffer, F. Pinel, E. Krishnasamy, S. Peter, H. Cartiaux, and X. Besseron. "*RESIF 3.0: Toward a Flexible & Automated Management of User Software Environment on HPC facility*". In ACM Practice & Experience in Advanced Research Computing (PEARC'21) pdf – code

Extending ULHPC Software Set

easybuild.readthedocs.io

hpc-docs.uni.lu/software/build/

- *aka* **Compiling/Building your own software**
  - ↪ *If possible*: rely on Easybuild (`eb -S <pattern>`) - beware of Easybuild prefix path
    - ✓ affects **default** builds/modules installdir `${EASYBUILD_PREFIX}/{software,modules/all}`
    - ✓ **MUST follow ULHPC guidelines!** `resif-load-{home,project}-swset-{prod,devel}`

Extending ULHPC Software Set

easybuild.readthedocs.io

hpc-docs.uni.lu/software/build/

- *aka* **Compiling/Building your own software**
  - ↪ *If possible*: rely on Easybuild (`eb -S <pattern>`) - beware of Easybuild prefix path
    - ✓ affects **default** builds/modules installdir `${EASYBUILD_PREFIX}/{software,modules/all}`
    - ✓ **MUST follow ULHPC guidelines!** `resif-load-{home,project}-swset-{prod,devel}`
- Using Easybuild to Build software **in your Home**

```
$ si[-gpu] [-c <threads>]    # get an interactive job
$ module load tools/EasyBuild
# /!\ IMPORTANT: ensure EASYBUILD_PREFIX == [basedir]/<cluster>/<environment>/<arch>
#                and that MODULEPATH is prefixed accordingly
$ resif-load-home-swset-{prod | devel}  # adapt environment
$ eb -S <softwarename>
# collect <filename>.eb == <name>-<version>[-<toolchain>][-<suffix>].eb
$ eb -Dr <filename>.eb   # check dependencies, normally most MUST be satisfied
$ eb -r  <filename>.eb
```

Extending ULHPC Software Set

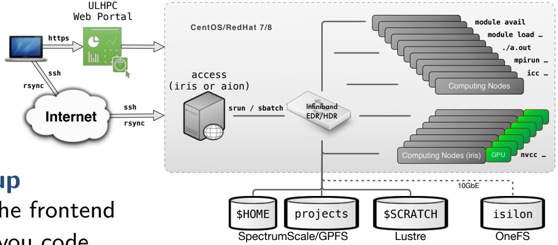
easybuild.readthedocs.io

hpc-docs.uni.lu/software/build/

- *aka* **Compiling/Building your own software**
  - ↪ *If possible*: rely on Easybuild (`eb -S <pattern>`) - beware of Easybuild prefix path
    - ✓ affects **default** builds/modules installdir `${EASYBUILD_PREFIX}/{software,modules/all}`
    - ✓ **MUST follow ULHPC guidelines!** `resif-load-{home,project}-swset-{prod,devel}`
- Using Easybuild to Build software **in the shared project `<project>`**

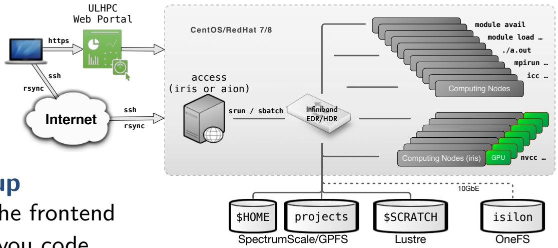
```
$ si[-gpu] [-c <threads>]   # get an interactive job
$ module load tools/EasyBuild
# /!\ IMPORTANT: ensure EASYBUILD_PREFIX == [basedir]/<cluster>/<environment>/<arch>
#                and that MODULEPATH is prefixed accordingly
$ resif-load-project-swset-{prod | devel} $PROJECTHOME/<project>
$ sg <project> -c "eb -S <softwarename>"
# collect <filename>.eb == <name>-<version>[-<toolchain>][-<suffix>].eb
$ sg <project> -c "eb -Dr <filename>.eb"   # check dependencies, normally most MUST be satisfied
$ sg <project> -c "eb -r  <filename>.eb"
```

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

1. High Performance Computing (HPC) @ UL
2. Batch Scheduling Configuration
3. User [Software] Environment
4. **Usage Policy**

General Guidelines

Acceptable Use Policy (AUP) 2.1

`Uni.lu-HPC-Facilities_Acceptable-Use-Policy_v2.1.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) 2.1

- **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

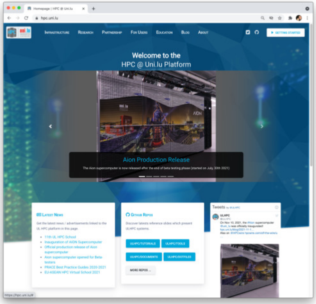
hpc.uni.lu

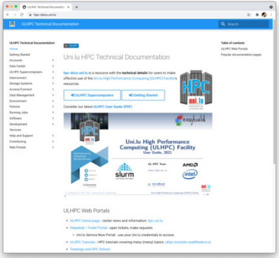
ULHPC Technical Docs

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

- **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`

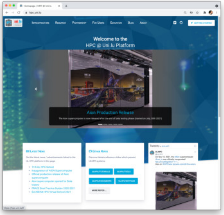
In all cases: **Carefully describe the problem and the context** Guidelines

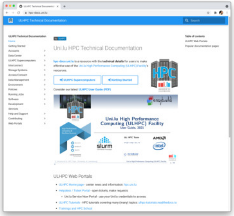
Thank you for your attention...

Questions?

High Performance Computing @ Uni.lu

hpc.uni.lu





ULHPC Technical Docs

hpc-docs.uni.lu

Research Computing and HPC Operations Team

~~Dr. Sebastien Varrette~~ **Hyacinthe Cartiaux** (Head)
Teddy Valette
Abatcha Olloh
Sarah Peter (BioCore Liaison)

High Level Support Team

Dr. Julien Schleich (Head)
Dr. Emmanuel Kieffer
Dr. Ezhilmathi Krishnasamy

Partnership Officer

Dr. Johnathan Pecero

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