HPC
UNIVERSITÉ DU LUXEMBOURG
High Performance Computing & Big Data Services
hpc.uni.lu
hpc@uni.lu
@ULHPC
LUXEMBOURG LET'S MAKE IT HAPPEN

Uni.lu HPC School 2021

PS3: Getting Started 2.0: SLURM, performance engineering and basic launchers

---

Uni.lu High Performance Computing (HPC) Team

H. Cartiaux

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

Latest versions available on Github:

UL HPC tutorials:	`https://github.com/ULHPC/tutorials`
UL HPC School:	`hpc.uni.lu/education/hpcschool`
PS3 tutorial sources:	`ulhpc-tutorials.rtfd.io/en/latest/beginners`

Summary

1 **Introduction**

2 **Getting Started on ULHPC**

3 **Batch Scheduling Configuration (Slurm)**
Slurm commands
ULHPC Slurm Configuration
Usage Example
Basic Slurm Launchers

4 **User [Software] Environment**

Main Objectives of this Session

- Review ULHPC environment
- **Design and usage of SLURM**
  - ↪ global configuration and Aion specificities
  - ↪ **common** and **advanced** SLURM tools and commands
    - ✓ `srun`, `sbatch`, `squeue` etc.
    - ✓ job specification
- Understand User Software environment
  - ↪ LMod modules, generated with Easybuild
- SLURM generic **launchers** you can use for your own jobs

Summary

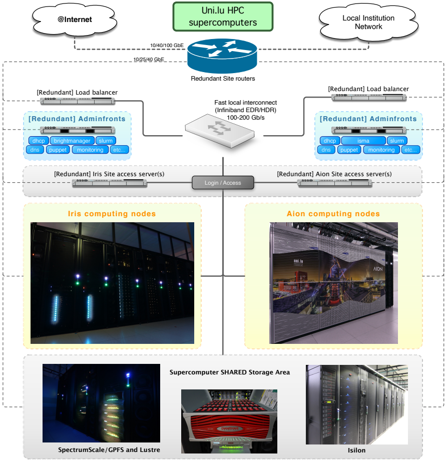
1 Introduction

2 Getting Started on ULHPC

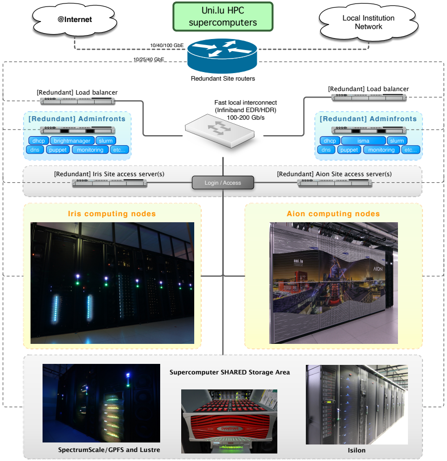
3 Batch Scheduling Configuration (Slurm)
   Slurm commands
   ULHPC Slurm Configuration
   Usage Example
   Basic Slurm Launchers

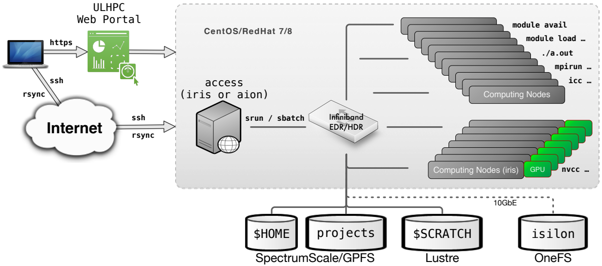
4 User [Software] Environment

UL HPC Supercomputers: General Architecture

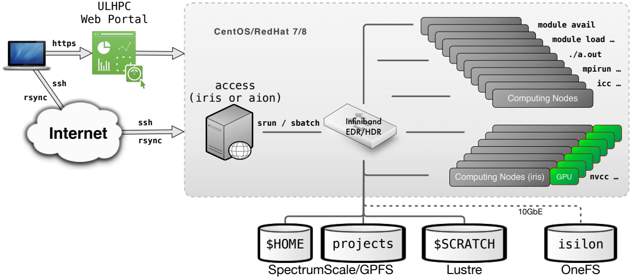

@Internet
Uni.lu HPC supercomputers
Local Institution Network
10/40/100 GbE
10/25/40 GbE
Redundant Site routers
[Redundant] Load balancer
[Redundant] Load balancer
Fast local interconnect (Infiniband EDR/HDR) 100-200 Gb/s
[Redundant] Adminfronts
dhcp
brightmanager
slurm
dns
puppet
monitoring
etc...
[Redundant] Adminfronts
dhcp
isma
slurm
dns
puppet
monitoring
etc...
[Redundant] Iris Site access server(s)
Login / Access
[Redundant] Aion Site access server(s)
Iris computing nodes
Aion computing nodes
Supercomputer SHARED Storage Area
SpectrumScale/GPFS and Lustre
Isilon

UL HPC Supercomputers: General Architecture


@Internet
Uni.lu HPC supercomputers
Local Institution Network
10/40/100 GbE
10/25/40 GbE
Redundant Site routers
[Redundant] Load balancer
[Redundant] Load balancer
Fast local interconnect (Infiniband EDR/HDR) 100-200 Gb/s
[Redundant] Adminfronts
dhcp
brightmanager
slurm
dns
puppet
monitoring
etc...
[Redundant] Adminfronts
dhcp
isma
slurm
dns
puppet
monitoring
etc...
[Redundant] Iris Site access server(s)
Login / Access
[Redundant] Aion Site access server(s)
Iris computing nodes
Aion computing nodes
Supercomputer SHARED Storage Area
SpectrumScale/GPFS and Lustre
Isilon



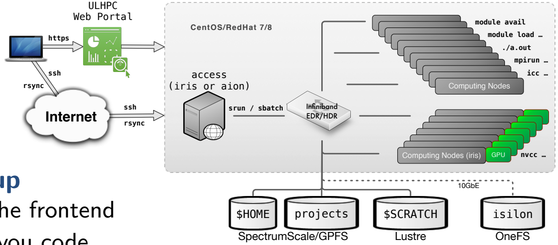
Compute Nodes / Storage Environment



- **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

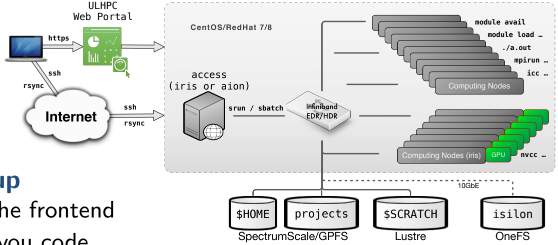
Directory	FileSystem	Max size	Max #files	Backup
`$HOME` (`iris`)	GPFS	500 GB	1.000.000	YES
`$SCRATCH`	Lustre	10 TB	1.000.000	**NO**
Project	GPFS	*per request*		PARTIALLY (`/backup` subdir)
Project	OneFS	*per request*		PARTIALLY

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}`

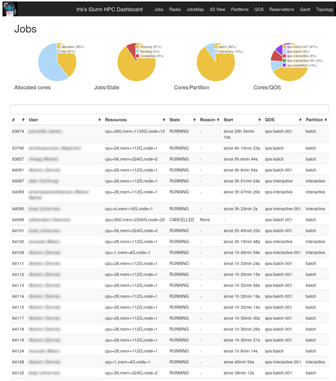
Typical Workflow on UL HPC resources



- **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 ...`

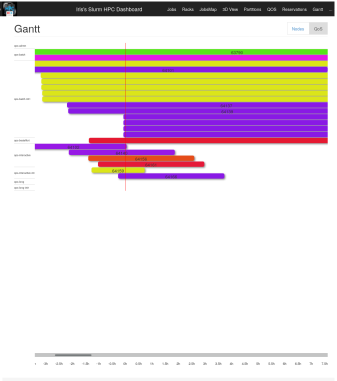
Slurm Web

(internal network only)

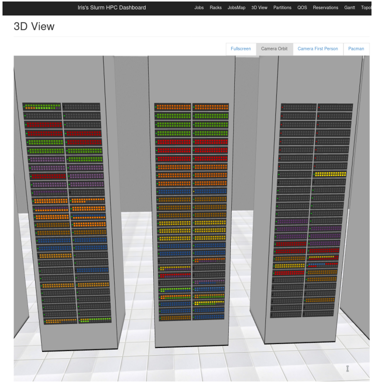


Jobs

#	User	Resources	State	Reason	Start	QOS	Partition
63674	[redacted]	cpu=280,mem=1120G,node=10	RUNNING	-	since 20h 34min 10s	qos-batch-001	batch
63790	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 6h 10min 23s	qos-batch	batch
63837	[redacted]	cpu=56,mem=224G,node=2	RUNNING	-	since 5h 6min 44s	qos-batch	batch
64081	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 4h 6min 54s	qos-batch-001	batch
64087	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 3h 51min 24s	qos-interactive	interactive
64089	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 3h 47min 20s	qos-interactive	interactive
64095	[redacted]	cpu=4,mem=16G,node=1	RUNNING	-	since 3h 33min 2s	qos-interactive-001	interactive
64099	[redacted]	cpu=560,mem=2240G,node=20	CANCELLED	None	-	qos-batch-001	batch
64101	[redacted]	cpu=56,mem=224G,node=2	RUNNING	-	since 2h 45min 22s	qos-batch-001	batch
64102	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 2h 19min 48s	qos-interactive	interactive
64109	[redacted]	cpu=1,mem=4G,node=1	RUNNING	-	since 1h 45min 59s	qos-interactive-001	interactive
64111	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 33min 24s	qos-batch-001	batch
64112	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 33min 15s	qos-batch-001	batch
64113	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 32min 56s	qos-batch-001	batch
64114	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 32min 19s	qos-batch-001	batch
64115	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 32min 14s	qos-batch-001	batch
64117	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 30min 30s	qos-batch-001	batch
64118	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 30min 26s	qos-batch-001	batch
64119	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 30min 21s	qos-batch-001	batch
64124	[redacted]	cpu=28,mem=112G,node=1	RUNNING	-	since 1h 8min 14s	qos-batch	batch
64128	[redacted]	cpu=1,mem=4G,node=1	RUNNING	-	since 45min 54s	qos-interactive-001	interactive
64133	[redacted]	cpu=56,mem=224G,node=2	RUNNING	-	since 38min 12s	qos-batch-001	batch



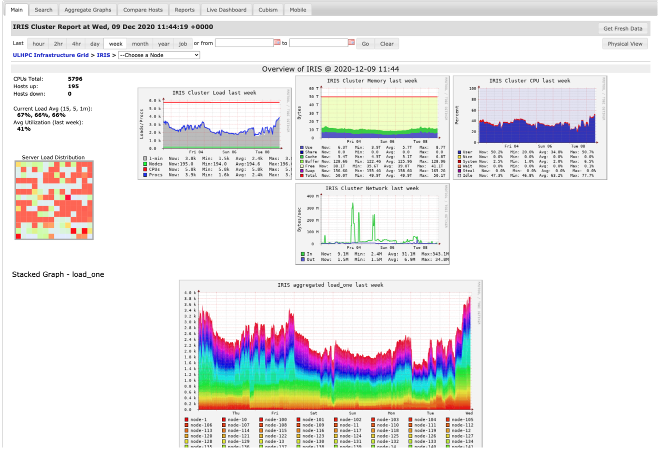
Gantt



3D View

Ganglia

https://hpc.uni.lu/iris/ganglia/


Main
Search
Aggregate Graphs
Compare Hosts
Reports
Live Dashboard
Cubism
Mobile
Get Fresh Data
Last hour 2hr 4hr day week month year job or from to Go Clear
Physical View
ULHPC Infrastructure Grid > IRIS > --Choose a Node
IRIS Cluster Load last week
Loads/Procs
Fri 04
Sun 06
Tue 08
IRIS Cluster Memory last week
Bytes
IRIS Cluster CPU last week
Percent
IRIS Cluster Network last week
Bytes/sec
Server Load Distribution
IRIS aggregated load_one last week
Thu Fri Sat Sun Mon Tue Wed
node-1 node-10 node-100 node-101 node-102 node-103 node-104 node-105
node-106 node-107 node-108 node-109 node-11 node-110 node-111 node-112
node-113 node-114 node-115 node-116 node-117 node-118 node-119 node-12
node-120 node-121 node-122 node-123 node-124 node-125 node-126 node-127
node-128 node-129 node-13 node-130 node-131 node-132 node-133 node-134

IRIS Cluster Report at Wed, 09 Dec 2020 11:44:19 +0000

Overview of IRIS @ 2020-12-09 11:44

CPUs Total:	**5796**
Hosts up:	**195**
Hosts down:	**0**

Current Load Avg (15, 5, 1m):
67%, 66%, 66%
Avg Utilization (last week):
41%

```
1-min   Now:  3.8k   Min:  1.5k   Avg:  2.4k   Max:  3.
Nodes   Now:195.0    Min:194.0    Avg:194.6    Max:196.
CPUs    Now:  5.8k   Min:  5.8k   Avg:  5.8k   Max:  5.
Procs   Now:  3.9k   Min:  1.6k   Avg:  2.4k   Max:  3.
```

```
Use     Now:   6.3T   Min:   3.9T   Avg:   5.7T   Max:   8.7T
Share   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
Cache   Now:   5.4T   Min:   4.5T   Avg:   5.1T   Max:   6.8T
Buffer  Now: 128.6G   Min: 122.4G   Avg: 125.9G   Max: 128.9G
Free   Now:  38.1T   Min:  35.6T   Avg:  39.0T   Max:  41.1T
Swap    Now: 156.6G   Min: 155.4G   Avg: 158.6G   Max: 165.2G
Total   Now:  50.0T   Min:  49.9T   Avg:  49.9T   Max:  50.1T
```

```
User    Now: 50.2%   Min: 20.0%   Avg: 34.8%   Max: 50.7%
Nice    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
System  Now:  2.5%   Min:  1.0%   Avg:  2.0%   Max:  6.5%
Wait    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.1%
Steal    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
Idle    Now: 47.3%   Min: 46.8%   Avg: 63.2%   Max: 77.7%
```

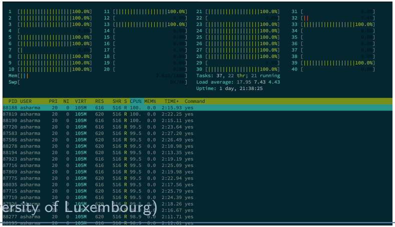
```
In    Now:  9.1M   Min:  2.4M   Avg: 31.1M   Max:343.1M
Out   Now:  1.5M   Min:  1.5M   Avg:  6.9M   Max: 34.8M
```

Stacked Graph - load_one

- **WIP**: Prometheus / Graphana
  ↪ together with aion

Interactive Monitoring/debugging

```
$> sq
# Connect to your job (first allocated node)
$> sjoin <JOBID>
$> htop    # view of all processes
           # F5: tree view
           # u <name>: filter by process of <name>
           # q: quit
```



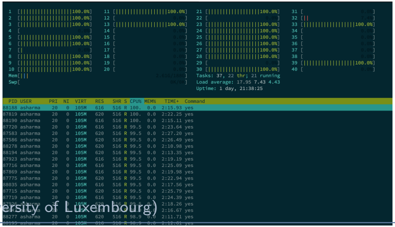
```
  1  [|||||||||||||||||100.0%]   11 [|||||||||||||||||100.0%]   21 [|||||||||||||||||100.0%]   31 [                    0.0%]
  2  [|||||||||||||||||100.0%]   12 [                    0.0%]   22 [                    0.0%]   32 [||                  7.4%]
  3  [|||||||||||||||||100.0%]   13 [|||||||||||||||||100.0%]   23 [|||||||||||||||||100.0%]   33 [|||||||||||||||||100.0%]
  4  [                    0.0%]   14 [                    0.0%]   24 [|||||||||||||||||100.0%]   34 [                    0.0%]
  5  [|||||||||||||||||100.0%]   15 [                    0.0%]   25 [|||||||||||||||||100.0%]   35 [                    0.0%]
  6  [|||||||||||||||||100.0%]   16 [                    0.0%]   26 [|||||||||||||||||100.0%]   36 [                    0.0%]
  7  [|                   0.6%]   17 [                    0.0%]   27 [|||||||||||||||||100.0%]   37 [                    0.0%]
  8  [|||||||||||||||||100.0%]   18 [                    0.0%]   28 [|||||||||||||||||100.0%]   38 [                    0.0%]
  9  [|||||||||||||||||100.0%]   19 [                    0.0%]   29 [                    0.0%]   39 [|||||||||||||||||100.0%]
  10 [|||||||||||||||||100.0%]   20 [                    0.0%]   30 [|||||||||||||||||100.0%]   40 [                    0.0%]
  Mem[|||                                            2.61G/188G]   Tasks: 37, 22 thr; 21 running
  Swp[                                                  0K/0K]   Load average: 17.95 7.43 4.43
                                                                  Uptime: 1 day, 21:38:25

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
88188 asharma    20   0  105M   616   516 R 100.  0.0  2:15.93 yes
87819 asharma    20   0  105M   620   516 R 100.  0.0  2:22.25 yes
88190 asharma    20   0  105M   616   516 R 100.  0.0  2:15.11 yes
87720 asharma    20   0  105M   616   516 R 99.5  0.0  2:23.64 yes
87583 asharma    20   0  105M   620   516 R 99.5  0.0  2:27.20 yes
87586 asharma    20   0  105M   620   516 R 99.5  0.0  2:26.49 yes
88278 asharma    20   0  105M   620   516 R 99.5  0.0  2:10.98 yes
88194 asharma    20   0  105M   620   516 R 99.5  0.0  2:13.35 yes
87923 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.19 yes
87716 asharma    20   0  105M   616   516 R 99.5  0.0  2:25.09 yes
87869 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.98 yes
87775 asharma    20   0  105M   620   516 R 99.5  0.0  2:22.94 yes
88035 asharma    20   0  105M   616   516 R 99.5  0.0  2:17.56 yes
87715 asharma    20   0  105M   620   516 R 99.5  0.0  2:25.79 yes
87719 asharma    20   0  105M   616   516 R 99.5  0.0  2:24.39 yes
87926 asharma    20   0  105M   620   516 R 99.5  0.0  2:18.26 yes
88042 asharma    20   0  105M   620   516 R 98.9  0.0  2:16.67 yes
88277 asharma    20   0  105M   620   516 R 98.9  0.0  2:11.71 yes
88195 asharma    20   0  105M   616   516 R 98.9  0.0  2:12.01 yes
```

Interactive Monitoring/debugging

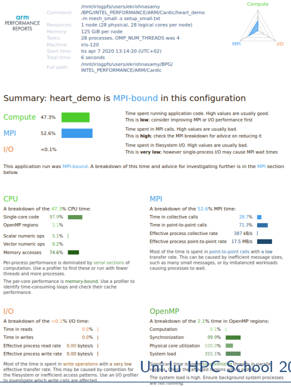
```
$> sq
# Connect to your job (first allocated node)
$> sjoin <JOBID>
$> htop    # view of all processes
           # F5: tree view
           # u <name>: filter by process of <name>
           # q: quit
```

- Arm Forge suit
  ↪ `ddt, map, perf-report`



```
  1  [|||||||||||||||||100.0%]   11 [|||||||||||||||||100.0%]   21 [|||||||||||||||||100.0%]   31 [                   0.0%]
  2  [|||||||||||||||||100.0%]   12 [                   0.0%]   22 [                   0.0%]   32 [||                 7.4%]
  3  [|||||||||||||||||100.0%]   13 [|||||||||||||||||100.0%]   23 [|||||||||||||||||100.0%]   33 [|||||||||||||||||100.0%]
  4  [                   0.0%]   14 [                   0.0%]   24 [|||||||||||||||||100.0%]   34 [                   0.0%]
  5  [|||||||||||||||||100.0%]   15 [                   0.0%]   25 [|||||||||||||||||100.0%]   35 [                   0.0%]
  6  [|||||||||||||||||100.0%]   16 [                   0.0%]   26 [|||||||||||||||||100.0%]   36 [                   0.0%]
  7  [|                  0.6%]   17 [                   0.0%]   27 [|||||||||||||||||100.0%]   37 [                   0.0%]
  8  [|||||||||||||||||100.0%]   18 [                   0.0%]   28 [|||||||||||||||||100.0%]   38 [                   0.0%]
  9  [|||||||||||||||||100.0%]   19 [                   0.0%]   29 [                   0.0%]   39 [|||||||||||||||||100.0%]
  10 [|||||||||||||||||100.0%]   20 [                   0.0%]   30 [|||||||||||||||||100.0%]   40 [                   0.0%]
  Mem[|||                                       2.61G/188G]   Tasks: 37, 22 thr; 21 running
  Swp[                                             0K/0K]   Load average: 17.95 7.43 4.43
                                                            Uptime: 1 day, 21:38:25

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
88188 asharma    20   0  105M   616   516 R 100.  0.0  2:15.93 yes
87819 asharma    20   0  105M   620   516 R 100.  0.0  2:22.25 yes
88190 asharma    20   0  105M   616   516 R 100.  0.0  2:15.11 yes
87720 asharma    20   0  105M   616   516 R 99.5  0.0  2:23.64 yes
87583 asharma    20   0  105M   620   516 R 99.5  0.0  2:27.20 yes
87586 asharma    20   0  105M   620   516 R 99.5  0.0  2:26.49 yes
88278 asharma    20   0  105M   620   516 R 99.5  0.0  2:10.98 yes
88194 asharma    20   0  105M   620   516 R 99.5  0.0  2:13.35 yes
87923 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.19 yes
87716 asharma    20   0  105M   616   516 R 99.5  0.0  2:25.09 yes
87869 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.98 yes
87775 asharma    20   0  105M   620   516 R 99.5  0.0  2:22.94 yes
88035 asharma    20   0  105M   616   516 R 99.5  0.0  2:17.56 yes
87715 asharma    20   0  105M   620   516 R 99.5  0.0  2:25.79 yes
87719 asharma    20   0  105M   616   516 R 99.5  0.0  2:24.39 yes
87926 asharma    20   0  105M   620   516 R 99.5  0.0  2:18.26 yes
88042 asharma    20   0  105M   620   516 R 99.5  0.0  2:16.67 yes
88277 asharma    20   0  105M   620   516 R 98.9  0.0  2:11.71 yes
88195 asharma    20   0  105M   616   516 R 98.9  0.0  2:12.01 yes
```



arm PERFORMANCE REPORTS

Command:	/mnt/irisgpfs/users/ekrishnasamy /BPG/INTEL_PERFORMANCE/ARM/Cardic/heart_demo -m mesh_small -s setup_small.txt
Resources:	1 node (28 physical, 28 logical cores per node)
Memory:	125 GiB per node
Tasks:	28 processes, OMP_NUM_THREADS was 4
Machine:	iris-120
Start time:	tis apr 7 2020 13:14:20 (UTC+02)
Total time:	6 seconds
Full path:	/mnt/irisgpfs/users/ekrishnasamy/BPG/ INTEL_PERFORMANCE/ARM/Cardic

Summary: heart_demo is MPI-bound in this configuration

Compute	47.3%	Time spent running application code. High values are usually good. This is **low**; consider improving MPI or I/O performance first
MPI	52.6%	Time spent in MPI calls. High values are usually bad. This is **high**; check the MPI breakdown for advice on reducing it
I/O	<0.1%	Time spent in filesystem I/O. High values are usually bad. This is **very low**; however single-process I/O may cause MPI wait times

This application run was MPI-bound. A breakdown of this time and advice for investigating further is in the MPI section below.

CPU

A breakdown of the 47.3% CPU time:

Single-core code	97.9%
OpenMP regions	2.1%
Scalar numeric ops	5.1%
Vector numeric ops	0.2%
Memory accesses	74.6%

Per-process performance is dominated by serial sections of computation. Use a profiler to find these or run with fewer threads and more processes.

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

MPI

A breakdown of the 52.6% MPI time:

Time in collective calls	28.7%
Time in point-to-point calls	71.3%
Effective process collective rate	387 kB/s
Effective process point-to-point rate	17.5 MB/s

Most of the time is spent in point-to-point calls with a low transfer rate. This can be caused by inefficient message sizes, such as many small messages, or by imbalanced workloads causing processes to wait.

I/O

A breakdown of the <0.1% I/O time:

Time in reads	0.0%
Time in writes	0.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	0.00 bytes/s

Most of the time is spent in write operations with a very low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

OpenMP

A breakdown of the 2.1% time in OpenMP regions:

Computation	0.1%
Synchronization	99.9%
Physical core utilization	100.0%
System load	355.1%

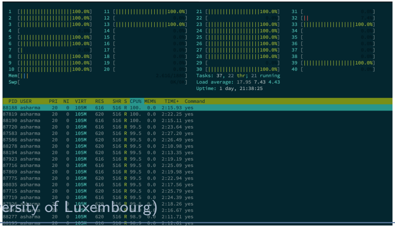
Significant time is spent synchronizing threads in parallel regions. Check the affected regions with a profiler.

The system load is high. Ensure background system processes are not running.

Interactive Monitoring/debugging

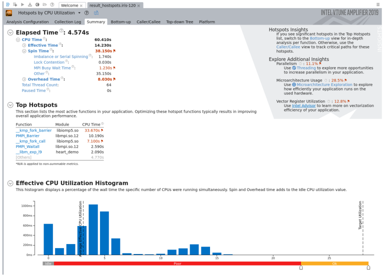
```
$> sq
# Connect to your job (first allocated node)
$> sjoin <JOBID>
$> htop   # view of all processes
          # F5: tree view
          # u <name>: filter by process of <name>
          # q: quit
```

- Arm Forge suit
  ↪ ddt, map, perf-report
- Intel VTune/Advisor/Inspector



```
  1  [||||||||||||||||||100.0%]   11 [||||||||||||||||||100.0%]   21 [||||||||||||||||||100.0%]   31 [                    0.0%]
  2  [||||||||||||||||||100.0%]   12 [                    0.0%]   22 [                    0.8%]   32 [||                  7.4%]
  3  [||||||||||||||||||100.0%]   13 [||||||||||||||||||100.0%]   23 [||||||||||||||||||100.0%]   33 [||||||||||||||||||100.0%]
  4  [                    0.0%]   14 [                    0.0%]   24 [||||||||||||||||||100.0%]   34 [                    0.0%]
  5  [||||||||||||||||||100.0%]   15 [                    0.0%]   25 [||||||||||||||||||100.0%]   35 [                    0.0%]
  6  [||||||||||||||||||100.0%]   16 [                    0.0%]   26 [||||||||||||||||||100.0%]   36 [                    0.0%]
  7  [|                   0.6%]   17 [                    0.0%]   27 [||||||||||||||||||100.0%]   37 [                    0.0%]
  8  [||||||||||||||||||100.0%]   18 [                    0.0%]   28 [||||||||||||||||||100.0%]   38 [                    0.0%]
  9  [||||||||||||||||||100.0%]   19 [                    0.0%]   29 [                    0.0%]   39 [||||||||||||||||||100.0%]
  10 [||||||||||||||||||100.0%]   20 [                    0.0%]   30 [||||||||||||||||||100.0%]   40 [                    0.0%]
  Mem[|||                                        2.61G/188G]   Tasks: 37, 22 thr; 21 running
  Swp[                                              0K/0K]   Load average: 17.95 7.43 4.43
                                                              Uptime: 1 day, 21:38:25

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
88188 asharma    20   0  105M   616   516 R 100.  0.0  2:15.93 yes
87819 asharma    20   0  105M   620   516 R 100.  0.0  2:22.25 yes
88190 asharma    20   0  105M   616   516 R 100.  0.0  2:15.11 yes
87720 asharma    20   0  105M   616   516 R 99.5  0.0  2:23.64 yes
87583 asharma    20   0  105M   620   516 R 99.5  0.0  2:27.20 yes
87586 asharma    20   0  105M   620   516 R 99.5  0.0  2:26.49 yes
88278 asharma    20   0  105M   620   516 R 99.5  0.0  2:10.98 yes
88194 asharma    20   0  105M   620   516 R 99.5  0.0  2:13.35 yes
87923 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.19 yes
87716 asharma    20   0  105M   616   516 R 99.5  0.0  2:25.09 yes
87869 asharma    20   0  105M   616   516 R 99.5  0.0  2:19.98 yes
87775 asharma    20   0  105M   620   516 R 99.5  0.0  2:22.94 yes
88035 asharma    20   0  105M   616   516 R 99.5  0.0  2:17.56 yes
87715 asharma    20   0  105M   620   516 R 99.5  0.0  2:25.79 yes
87719 asharma    20   0  105M   616   516 R 99.5  0.0  2:24.39 yes
87926 asharma    20   0  105M   620   516 R 99.5  0.0  2:18.26 yes
[illegible] asharma 20   0  105M   620   516 R [illegible] 0.0  2:16.67 yes
88277 asharma    20   0  105M   620   516 R 98.9  0.0  2:11.71 yes
88195 asharma    20   0  105M   616   516 R 98.9  0.0  2:12.81 yes
```



Elapsed Time: 4.574s

CPU Time:	60.410s
Effective Time:	14.230s
Spin Time:	38.150s
Imbalance or Serial Spinning:	1.740s
Lock Contention:	0.030s
MPI Busy Wait Time:	1.230s
Other:	35.150s
Overhead Time:	8.030s
Total Thread Count:	64
Paused Time:	0s

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
__kmp_fork_barrier	libiomp5.so	33.670s
PMPI_Barrier	libmpi.so.12	10.190s
__kmp_fork_call	libiomp5.so	7.100s
PMPI_Waitall	libmpi.so.12	2.590s
__libm_exp_l9	heart_demo	2.090s
[Others]		4.770s

*N/A is applied to non-summable metrics.

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the Bottom-up view for in-depth analysis per function. Otherwise, use the Caller/Callee view to track critical paths for these hotspots.

Explore Additional Insights

Parallelism : 11.1%
Use Threading to explore more opportunities to increase parallelism in your application.

Microarchitecture Usage : 28.5%
Use Microarchitecture Exploration to explore how efficiently your application runs on the used hardware.

Vector Register Utilization : 12.8%
Use Intel Advisor to learn more on vectorization efficiency of your application.

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.

Discovering and visualizing UL HPC environment

Your Turn!

Hands-on Discovering your HPC environment

▶ url ◀ | github | src

- **Connect to the cluster frontend**
- Navigate between `$HOME`, `$SCRATCH`
- Discovering storage usage
- Live status tools (Slurm Web, Ganglia. . .)

Summary

1 Introduction

2 Getting Started on ULHPC

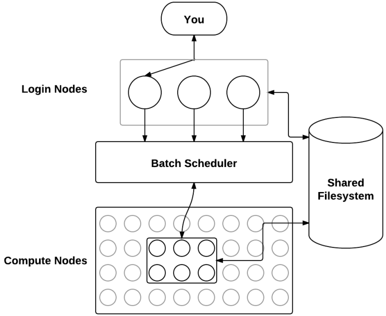
3 Batch Scheduling Configuration (Slurm)

Slurm commands
ULHPC Slurm Configuration
Usage Example
Basic Slurm Launchers

4 User [Software] Environment

Resource and Job Management Systems

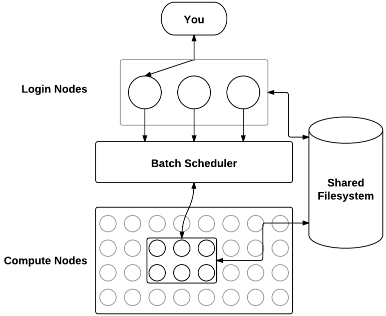
- **Resource and Job Management System (RJMS)**
  - ↪ *Glue* for a parallel computer to execute parallel jobs
  - ↪ **Goal**: satisfy users demands for computation
    - ✓ assign resources to user jobs with an efficient manner



Resource and Job Management Systems

- **Resource and Job Management System (RJMS)**
  - ↪ *Glue* for a parallel computer to execute parallel jobs
  - ↪ **Goal**: satisfy users demands for computation
    - ✓ assign resources to user jobs with an efficient manner

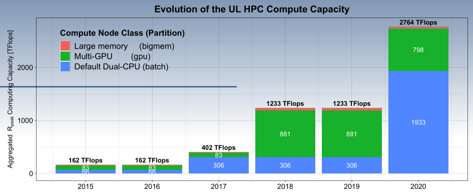
- **HPC Resources**:
  - ↪ Nodes (typically a unique IP address)
    - ✓ Sockets / Cores / Hyperthreads
    - ✓ Memory
    - ✓ Interconnect/switch resources
  - ↪ Generic resources (e.g. GPUs)
  - ↪ Licenses
- **Strategic Position**
  - ↪ Direct/constant knowledge of resources
  - ↪ Launch and otherwise manage jobs



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

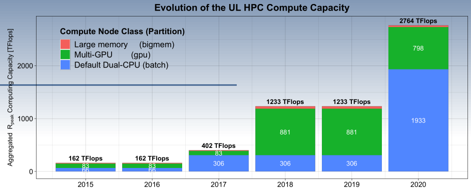
- **User jobs** have the following key characteristics:
  - ↪ set of requested resources:
    - ✓ number of computing resources: **nodes** (including all their CPUs and cores) or **CPUs** (including all their cores) or **cores**
    - ✓ amount of **memory**: either per node or per CPU
    - ✓ **(wall)time**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

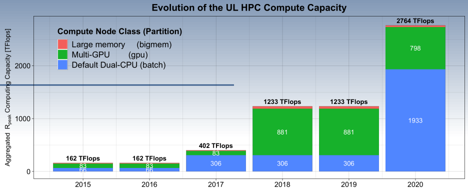
- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for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

Slurm on ULHPC clusters



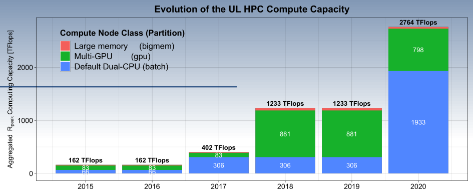
- Predefined **Queues/Partitions** depending on node type
  - ↪ `batch` (Default Dual-CPU nodes) **Max**: 64 nodes, 2 days walltime
  - ↪ `gpu` (GPU nodes nodes) **Max**: 4 nodes, 2 days walltime
  - ↪ `bigmem` (Large-Memory nodes) **Max**: 1 node, 2 days walltime
  - ↪ In addition: `interactive` (for 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 Multi-cluster configuration between `iris` and `aion`
  - ↪ easily **query** state on remote cluster `{squeue...} -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>
```

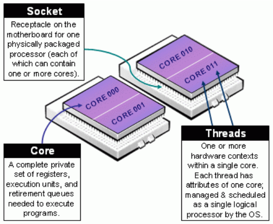
Submitting Jobs

- `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.

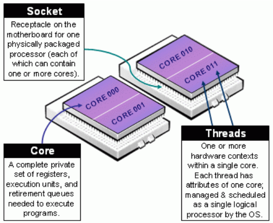
Specific Resource Allocation

- **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>`


Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 000
CORE 001
CORE 010
CORE 011
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

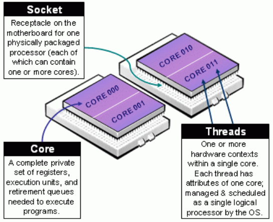
Specific Resource Allocation

- **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>`


Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 010
CORE 011
CORE 000
CORE 001
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

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} [...]`



Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).

Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.

Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

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`

Usage Example: Interactive Jobs

`si,si-[gpu|bigmem]`

```
# Simple interactive job - match feature name with target partition
srun -p interactive --qos debug -C {batch,gpu,bigmem} [...] --pty bash
# OR, **better**
si[-gpu|-bigmem] [...]
```

- *Floating* `interactive` partition allow to access all nodes
  - ↪ no more dedicated resources: optimize the cluster for passive jobs
  - ↪ 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 [...]`

Usage Example: Interactive Jobs

si,si-[gpu|bigmem]

```
# Simple interactive job - match feature name with target partition
srun -p interactive --qos debug -C {batch,gpu,bigmem} [...] --pty bash
# OR, **better**
si[-gpu|-bigmem] [...]
```

```
# Simple interactive job on batch partition
(access)$> si
# srun -p interactive --qos debug -C batch  --mem-per-cpu 4096 --pty bash
(node)$> echo $SLURM_NTASKS
1
(node)$> echo $SLURM_JOBID
2166371
$> sq
# squeue -u <login>
```

Usage Example: Regular Jobs

`sbatch [...]`

- Restricted to **Max 2 days** / Max 64 nodes   default QOS induced by the `job_submit.lua` plugin
  - ↪ enforce precision of project/training account (`-A <account>`) **when appropriate** *i.e.* for
    - ✓ project name `<project>`
    - ✓ lecture/course name: `<lecture>`
  - ↪ **do NOT reserve** `gpu` **nodes (**`-p gpu`**) WITHOUT** `-G [1-4]`
    - ✓ Iris compute nodes in the `gpu` partition have 4xVolta V100 GPUs and 768GB RAM

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

Usage Example: Long Jobs

--qos long

```
# Select target partition to bypass default walltime restrictions
sbatch -p {batch | gpu | bigmem} --qos long [...]
```

- Extended Max walltime (MaxWall) set to **14 days** EuroHPC/PRACE Recommendations
  ↪ **yet** restricted usage: Max 6 nodes, Max 2 nodes per Job, Max **4** Jobs 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 [...]`

Usage Example: Best-effort Jobs

`--qos besteffort`

- Special **preemptible QOS** for **best-effort jobs**
  - ↪ *lowest* priority
  - ↪ **deleted if another non-besteffort job wants resources** where they are running.
  - ↪ useful to maximize use of the cluster for short-term jobs
    - ✓ several limits inherent to *regular* QOS are **bypassed** in besteffort
  - ↪ many scientific applications support internal state saving and restart **Checkpoint-restart**
    - ✓ System-level CR possible with DMTCP: Distributed MultiThreaded CheckPointing
    - ✓ see the official DMTCP launchers

```
sbatch -p {batch | gpu | bigmem} --qos besteffort [...]
```

Live Job Statistics

```
$> scontrol show job 2166371
JobId=2166371 JobName=bash
   UserId=<login>(<uid>) GroupId=clusterusers(666) MCS_label=N/A
   Priority=12741 Nice=0 Account=ulhpc QOS=debug JobState=RUNNING Reason=None
   [...]
   SubmitTime=2020-12-07T22:08:25 EligibleTime=2020-12-07T22:08:25
   StartTime=2020-12-07T22:08:25 EndTime=2020-12-07T22:38:25
   [...]
   WorkDir=/mnt/irisgpfs/users/<login>
```

Node/Job Statistics

```
$> sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
interactive    up    4:00:00    196   idle iris-[001-196]
batch*         up 2-00:00:00      5    mix [...]
batch*         up 2-00:00:00    127  alloc [...]
batch*         up 2-00:00:00     36   idle [...]
gpu            up 2-00:00:00      4   resv iris-[186,191-193]
gpu            up 2-00:00:00     19  alloc [...]
gpu            up 2-00:00:00      1   idle iris-185
bigmem         up 2-00:00:00      4    mix iris-[187-190]
```

Node/Job Statistics

```
$> sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
interactive    up    4:00:00    196   idle iris-[001-196]
batch*         up 2-00:00:00      5    mix [...]
batch*         up 2-00:00:00    127  alloc [...]
batch*         up 2-00:00:00     36   idle [...]
gpu            up 2-00:00:00      4   resv iris-[186,191-193]
gpu            up 2-00:00:00     19  alloc [...]
gpu            up 2-00:00:00      1   idle iris-185
bigmem         up 2-00:00:00      4    mix iris-[187-190]
```

```
$> slist <JOBID>
# sacct -j <JOBID> --format User,JobID,Jobname%30,partition,state,time,elapsed,\
#              MaxRss,MaxVMSize,nnodes,ncpus,nodelist,AveCPU,ConsumedEnergyRaw
# seff <JOBID>
```

Basic Slurm Launcher Examples

Prefer `--ntasks-per-node` over `-n` !

```
#!/bin/bash -l                # <--- DO NOT FORGET '-l'
### Request a single task using one core on one node for 5 minutes in the batch queue
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 1
#SBATCH --time=0-00:05:00
#SBATCH -p batch

print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
# Safeguard for NOT running this launcher on access/login nodes
module purge || print_error_and_exit "No 'module' command"
# List modules required for execution of the task
module load <...>
# [...]
```

Basic Slurm Launcher Examples

Prefer `--ntasks-per-node` over `-n` !

```
#!/bin/bash -l
### Request 6 single core tasks equally spread across two nodes for 3 hours
#SBATCH -N 2
#SBATCH --ntasks-per-node=3
#SBATCH -c 1
#SBATCH --time=0-03:00:00
#SBATCH -p batch
print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
module purge || print_error_and_exit "No 'module' command"
module load <...>
# [...]
```

```
# Total: 6 tasks within 2 nodes
sbatch <launcher>
```

```
# Total: 9 tasks within 3 nodes
sbatch -N 3 <launcher>
```

Basic Slurm Launcher Examples

Prefer `--ntasks-per-node` over `-n` !

```
#!/bin/bash -l
### Request as many tasks as cores available on a single node for 3 hours
#SBATCH -N 1
#SBATCH --ntasks-per-node=28  # On iris; for aion, use --ntasks-per-node=128
#SBATCH -c 1
#SBATCH --time=0-03:00:00
#SBATCH -p batch
print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
module purge || print_error_and_exit "No 'module' command"
module load <...>
# [...]
```

```
# Total (iris): 28 tasks within 1 node
sbatch <launcher>
```

```
# Total (iris): 56 tasks within 2 nodes
sbatch -N 2 <launcher>
```

Basic Slurm Launcher Examples (Large memory)

```
#!/bin/bash -l
### Request one sequential task requiring half the memory of a regular iris node for 1 day
#SBATCH -J MyLargeMemorySequentialJob       # Job name
#SBATCH --mail-user=Your.Email@Address.lu   # mail me ...
#SBATCH --mail-type=end,fail                # ... upon end or failure
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 1
#SBATCH --mem=64GB         # if above 112GB: consider bigmem partition (USE WITH CAUTION)
#SBATCH --time=1-00:00:00
#SBATCH -p batch           # if above 112GB: consider bigmem partition (USE WITH CAUTION)
print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
module purge || print_error_and_exit "No 'module' command"
module load <...>
# [...]
```

Basic Slurm Launcher Examples (GPU)

```
#!/bin/bash -l
### Request one GPU tasks for 4 hours - dedicate 1/4 of available cores for its management
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 7
#SBATCH -G 1
#SBATCH --time=00:04:00
#SBATCH -p gpu
print_error_and_exit() { echo "***ERROR*** $*"; exit 1; }
module purge || print_error_and_exit "No 'module' command"
# You probably want to use more recent gpu/CUDA-optimized software builds
module use /opt/apps/resif/iris/2019b/gpu/modules
module load <...>
# This should report a single GPU (over 4 available per gpu node)
nvidia-smi
# [...]
```

Slurm [Generic] Launchers 2.0

```
#!/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`

Summary

1 Introduction

2 Getting Started on ULHPC

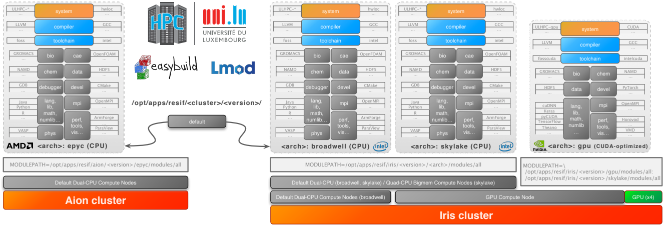
3 Batch Scheduling Configuration (Slurm)
   Slurm commands
   ULHPC Slurm Configuration
   Usage Example
   Basic Slurm Launchers

4 User [Software] Environment

Software/Modules Management

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

- Based on Environment Modules / LMod
  - ↪ convenient way to dynamically change the users environment with `module` command
- Currently on UL HPC: > **270 software packages**, in *multiple* versions
  - ↪ software environment generated using RESIF 3.0 framework over Easybuild
  - ↪ hierarchical organization (Categorized Naming Scheme) **Ex**: `toolchain/{foss,intel}`



MODULEPATH=/opt/apps/resif/aion/<version>/epyc/modules/all

MODULEPATH=/opt/apps/resif/iris/<version>/<arch>/modules/all

```
MODULEPATH=\
 /opt/apps/resif/iris/<version>/gpu/modules/all:
 /opt/apps/resif/iris/<version>/skylake/modules/all
```

```
$> 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`

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

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

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

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

Interaction with Slurm and the software environment

Your Turn!

Hands-on Slurm

▶ url ◀ | github | src

- Run interactive job si
- Run passive jobs sbatch
- Write basics launcher scripts

Hands-on Modules

▶ url ◀ | github | src

- play with `module` command
- Use the `MODULEPATH` variable in order to change the software set
- Compile and run examples in at least two different languages

Thank you for your attention...

Questions?

ulhpc-tutorials.rtfd.io/en/latest/beginners

High Performance Computing @ Uni.lu

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

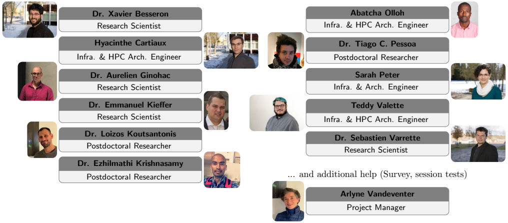
1 **Introduction**

2 **Getting Started on ULHPC**

3 **Batch Scheduling Configuration (Slurm)**
- Slurm commands
- ULHPC Slurm Configuration
- Usage Example
- Basic Slurm Launchers

4 **User [Software] Environment**

Uni.lu HPC School 2021 Contributors



Dr. Xavier Besseron
Research Scientist

Hyacinthe Cartiaux
Infra. & HPC Arch. Engineer

Dr. Aurelien Ginohac
Research Scientist

Dr. Emmanuel Kieffer
Research Scientist

Dr. Loizos Koutsantonis
Postdoctoral Researcher

Dr. Ezhilmathi Krishnasamy
Postdoctoral Researcher

Abatcha Olloh
Infra. & HPC Arch. Engineer

Dr. Tiago C. Pessoa
Postdoctoral Researcher

Sarah Peter
Infra. & Arch. Engineer

Teddy Valette
Infra. & HPC Arch. Engineer

Dr. Sebastien Varrette
Research Scientist

... and additional help (Survey, session tests)

Arlyne Vandeventer
Project Manager

hpc.uni.lu