Uni.lu HPC School 2019

PS4a: Monitoring & Profiling I

why, what, how, where to look

Uni.lu High Performance Computing (HPC) Team

S. Peter & 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:	http://hpc.uni.lu/hpc-school/
PS4a tutorial sources:	ulhpc-tutorials.rtfd.io/en/latest/

Summary

1. Objectives
2. Experiment planning & workflow
3. Hardware knowledge
4. Live status
5. Common mistakes and pitfalls
6. Getting help

Objectives

- Understanding a standard HPC workflow (development and production run)
- Monitoring your job at the system level
- Monitoring your job at the job scheduler level
- Finding your bottlenecks

Summary

1. Objectives
2. **Experiment planning & workflow**
3. Hardware knowledge
4. Live status
5. Common mistakes and pitfalls
6. Getting help

Workflow for Experiment Campaigns

1. Before the campaign
   - Send data to the clusters
   - Check/Install required software
2. Preparation
   - Test and debug
   - Prepare launcher script
3. Execution
   - Run the campaign
   - Monitor the execution
4. After a campaign
   - Retrieve output data
   - Archive and cleanup your data

Experiment campaign: Preparation

Goals

- Make sure everything will run OK
- Prepare submission script / launcher

Interactive approach

- Use the alias `si` to start an interactive job
- Allows to try commands one by one
- Work on a small case with a small number of cores
- Debug and check the results

Why prepare a submission script?

- Contains all commands and parameters
  ⇒ Easy re-execution
- No need to stay in front your computer

Experiment campaign: Execution

Submit the jobs

- Use the submission script / launcher
- Iris: Submit to Slurm with `sbatch your_script.sh` (Slurm batch Script)
- Actual experiment execution with possibly many nodes
- Non interactive execution, it might not start immediately

Get resource usage of your program with time

- `time` tool is installed on the nodes which give execution time of a command or script
- You can use time command to get more info about CPU usage, memory usage, socket messages sent, etc...
- `/usr/bin/time -v ./stress-script.sh`

Experiment campaign: Monitor the execution

- Output/Logfile of your application
- Slurm scheduler information:
  - ↪ List your jobs and statuses: `squeue -u <yourlogin>`
  - ↪ Information of a running job: `sstat -j <jobid>`
  - ↪ Used resources information: `sacct`

Summary

1 Objectives

2 Experiment planning & workflow

3 **Hardware knowledge**

4 Live status

5 Common mistakes and pitfalls

6 Getting help

Heterogeneity

- Heterogeneous clusters
- Special nodes (‘bigmem‘, ‘gpu‘)
- Nodes must be targeted explicitely if reproducibility is important (benchmarks, performance evaluation, algorithm comparison, etc)

Heterogeneity (Iris)

Vendor	N	Proc. Description	Cores	Mem
Dell	108	Xeon E5-2680 v4 @ 2.4GHz	2 × 14C	128GB
Dell	60	Xeon Gold 6132 @ 2.6GHz	2 × 14C	128GB
Dell	24	Xeon Gold 6132 @ 2.6GHz	2 × 14C	768GB
		+ 4x NVidia Tesla V100		
Dell	4	Xeon Platinum 8180M @ 2.5 GHz	4 × 28C	3072GB

- sbatch -p batch –qos qos-batch -C broadwell [...]
- sbatch -p batch –qos qos-batch -C skylake [...]
- sbatch -p gpu –qos qos-gpu [...]
- sbatch -p bigmem –qos qos-bigmem [...]

Summary

1 Objectives

2 Experiment planning & workflow

3 Hardware knowledge

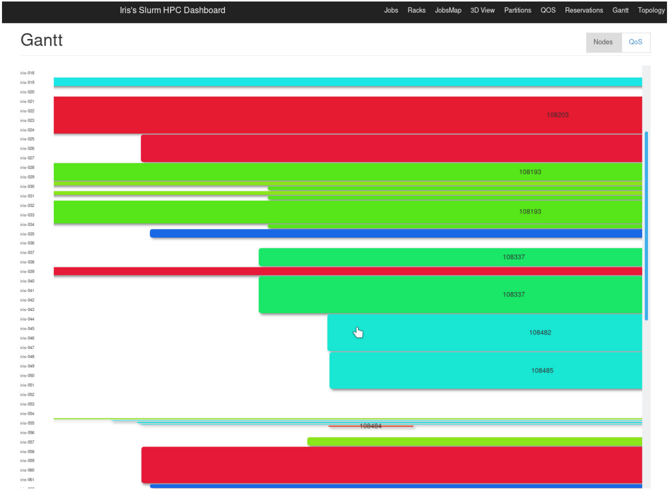
4 **Live status**

5 Common mistakes and pitfalls

6 Getting help

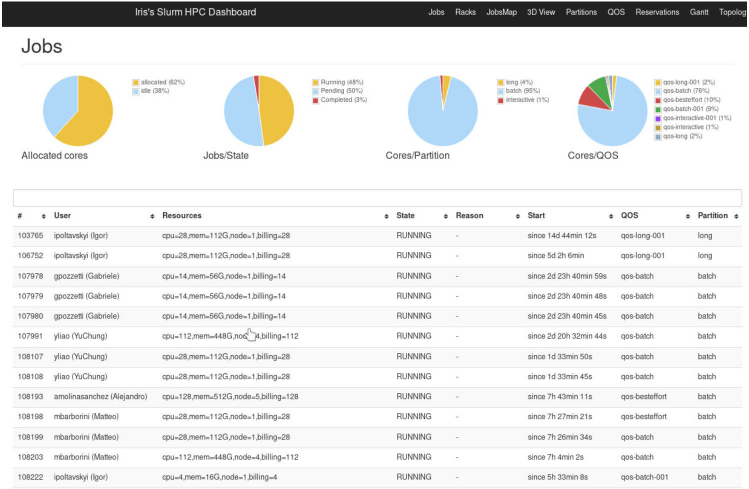
Slurm Web

`https://access-iris.uni.lu/slurm/`



Slurm Web

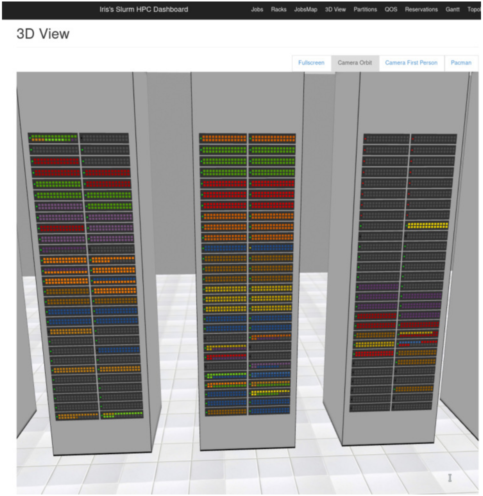
https://access-iris.uni.lu/slurm/



#	User	Resources	State	Reason	Start	QOS	Partition
103765	ipoltavskyi (Igor)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 14d 44min 12s	qos-long-001	long
106752	ipoltavskyi (Igor)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 5d 2h 6min	qos-long-001	long
107978	gpozzetti (Gabriele)	cpu=14,mem=56G,node=1,billing=14	RUNNING	-	since 2d 23h 40min 59s	qos-batch	batch
107979	gpozzetti (Gabriele)	cpu=14,mem=56G,node=1,billing=14	RUNNING	-	since 2d 23h 40min 48s	qos-batch	batch
107980	gpozzetti (Gabriele)	cpu=14,mem=56G,node=1,billing=14	RUNNING	-	since 2d 23h 40min 45s	qos-batch	batch
107991	yliao (YuChung)	cpu=112,mem=448G,node=4,billing=112	RUNNING	-	since 2d 20h 32min 44s	qos-batch	batch
108107	yliao (YuChung)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 1d 33min 50s	qos-batch	batch
108108	yliao (YuChung)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 1d 33min 45s	qos-batch	batch
108193	amolinasanchez (Alejandro)	cpu=128,mem=512G,node=5,billing=128	RUNNING	-	since 7h 43min 11s	qos-besteffort	batch
108198	mbarborini (Matteo)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 7h 27min 21s	qos-besteffort	batch
108199	mbarborini (Matteo)	cpu=28,mem=112G,node=1,billing=28	RUNNING	-	since 7h 26min 34s	qos-batch	batch
108203	mbarborini (Matteo)	cpu=112,mem=448G,node=4,billing=112	RUNNING	-	since 7h 4min 2s	qos-batch	batch
108222	ipoltavskyi (Igor)	cpu=4,mem=16G,node=1,billing=4	RUNNING	-	since 5h 33min 8s	qos-batch-001	batch

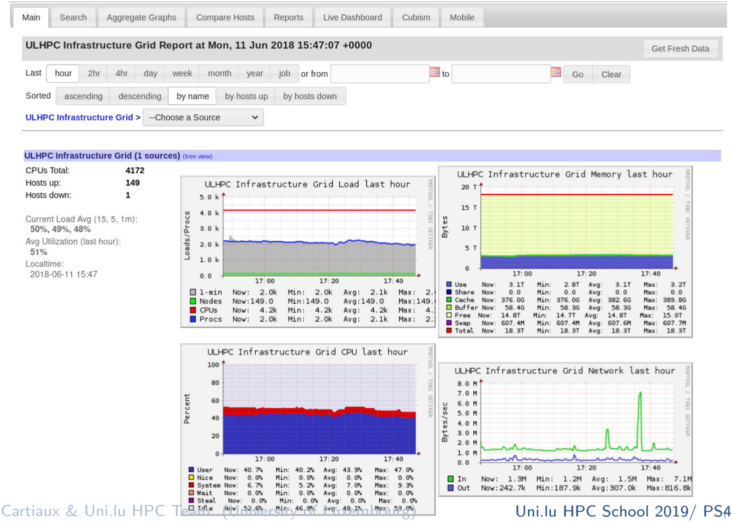
Slurm Web

https://access-iris.uni.lu/slurm/



Resource usage

`http://hpc.uni.lu/iris/ganglia`



ULHPC Infrastructure Grid Report at Mon, 11 Jun 2018 15:47:07 +0000

ULHPC Infrastructure Grid > --Choose a Source

ULHPC Infrastructure Grid (1 sources) (tree view)

CPUs Total:	**4172**
Hosts up:	**149**
Hosts down:	**1**

Current Load Avg (15, 5, 1m):
50%, 49%, 48%

Avg Utilization (last hour):
51%

Localtime:
2018-06-11 15:47

ULHPC Infrastructure Grid Load last hour

```
□ 1-min   Now:  2.0k   Min:  2.0k   Avg:  2.1k   Max:  2.
■ Nodes   Now:149.0    Min:149.0    Avg:149.0    Max:149.
■ CPUs    Now:  4.2k   Min:  4.2k   Avg:  4.2k   Max:  4.
■ Procs   Now:  2.0k   Min:  2.0k   Avg:  2.1k   Max:  2.
```

ULHPC Infrastructure Grid Memory last hour

```
■ Use    Now:   3.1T   Min:   2.8T  Avg:   3.1T   Max:   3.2T
■ Share  Now:   0.0    Min:   0.0   Avg:   0.0    Max:   0.0
■ Cache  Now: 376.0G   Min: 376.0G  Avg: 382.6G   Max: 389.8G
■ Buffer Now:  58.4G   Min:  58.3G  Avg:  58.3G   Max:  58.4G
□ Free  Now:  14.8T   Min:  14.7T  Avg:  14.8T   Max:  15.0T
■ Swap   Now: 607.4M   Min: 607.4M  Avg: 607.6M   Max: 607.7M
■ Total  Now:  18.3T   Min:  18.3T  Avg:  18.3T   Max:  18.3T
```

ULHPC Infrastructure Grid CPU last hour

```
■ User   Now:  40.7%   Min:  40.2%  Avg:  43.9%   Max:  47.0%
■ Nice   Now:   0.0%   Min:   0.0%  Avg:   0.0%   Max:   0.0%
■ System Now:   6.7%   Min:   5.2%  Avg:   7.0%   Max:   9.3%
■ Wait   Now:   0.0%   Min:   0.0%  Avg:   0.0%   Max:   0.0%
■ Steal   Now:   0.0%   Min:   0.0%  Avg:   0.0%   Max:   0.0%
□ Idle   Now:  52.6%   Min:  46.8%  Avg:  49.1%   Max:  53.0%
```

ULHPC Infrastructure Grid Network last hour

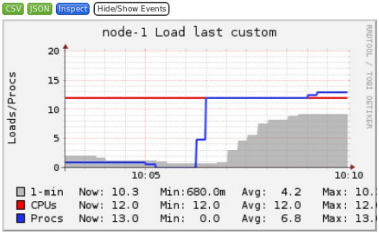
```
■ In     Now:   1.3M   Min:   1.2M  Avg:   1.5M   Max:   7.1M
■ Out    Now:242.7k    Min:187.9k   Avg:307.0k    Max:816.8k
```

Getting faster: Identify performance bottlenecks

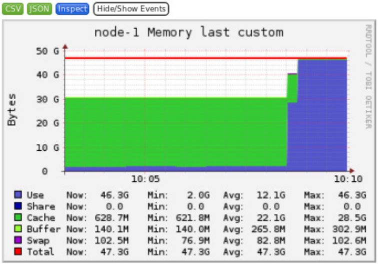
Note for code developers: The first bottleneck is your algorithm!
Know the hardware

- Computer nodes are connected using a fast interconnect
- Different types of resources:
  Processors, GPU, Memory, Storage, Network

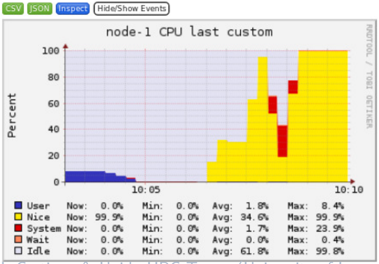
Identify your bottleneck (memory)



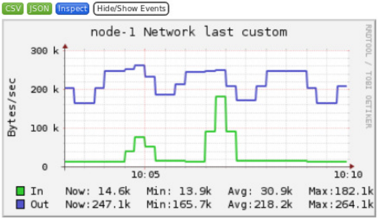
```
□ 1-min   Now: 10.3   Min:680.0m   Avg:  4.2   Max: 10.
■ CPUs    Now: 12.0   Min: 12.0    Avg: 12.0   Max: 12.
■ Procs   Now: 13.0   Min:  0.0    Avg:  6.8   Max: 13.
```



```
■ Use     Now:  46.3G   Min:   2.0G   Avg:  12.1G   Max:  46.3G
■ Share   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
■ Cache   Now: 628.7M   Min: 621.8M   Avg:  22.1G   Max:  28.5G
■ Buffer  Now: 140.1M   Min: 140.0M   Avg: 265.8M   Max: 302.9M
■ Swap    Now: 102.5M   Min:  76.9M   Avg:  82.8M   Max: 102.6M
■ Total   Now:  47.3G   Min:  47.3G   Avg:  47.3G   Max:  47.3G
```



```
■ User    Now:  0.0%   Min:  0.0%   Avg:  1.8%   Max:  8.4%
■ Nice    Now: 99.9%   Min:  0.0%   Avg: 34.6%   Max: 99.9%
■ System  Now:  0.0%   Min:  0.0%   Avg:  1.7%   Max: 23.9%
■ Wait    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.4%
□ Idle    Now:  0.0%   Min:  0.0%   Avg: 61.8%   Max: 99.8%
```



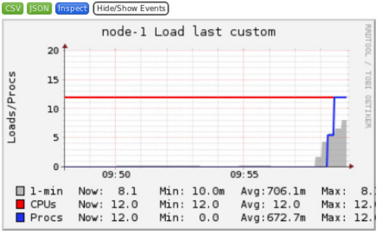
```
■ In    Now: 14.6k   Min: 13.9k   Avg: 30.9k   Max:182.1k
■ Out   Now:247.1k   Min:165.7k   Avg:218.2k   Max:264.1k
```

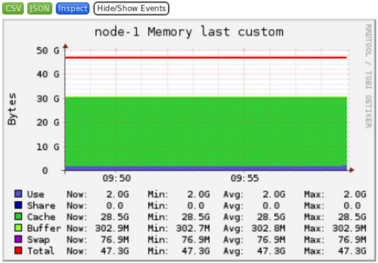
Identify your bottleneck (memory)

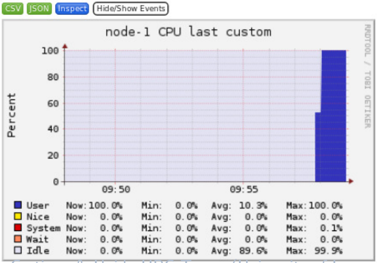
Application is limited by the size of the memory

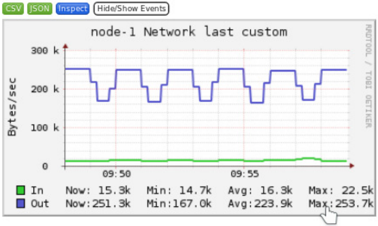
- Reserve all CPUs on a single node, to get access to all memory banks
- Use a node with a bigger memory (bigmem)
- Distributed execution on multiple nodes (MPI)

Identify your bottleneck (CPU)







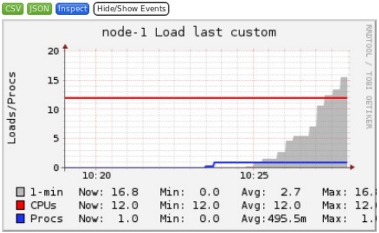


Identify your bottleneck (CPU)

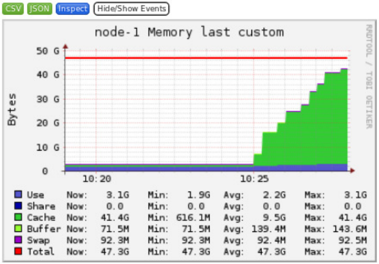
Application is limited by the speed of the processor

- Optimize your code
- Use GPU accelerator (CUDA)
- Parallel execution on multiple nodes (MPI)
- Parallel execution on multiple nodes with GPUs (MPI+CUDA)

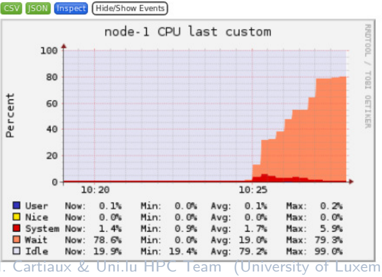
Identify your bottleneck (I/O)



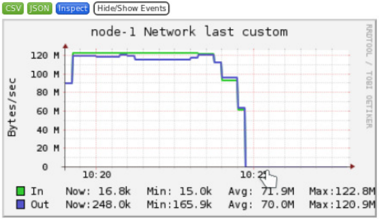
```
■ 1-min   Now: 16.8    Min:  0.0    Avg:   2.7    Max: 16.
■ CPUs    Now: 12.0    Min: 12.0    Avg:  12.0    Max: 12.
■ Procs   Now:  1.0    Min:  0.0    Avg:495.5m    Max:  1.
```



```
■ Use    Now:  3.1G   Min:   1.9G   Avg:   2.2G   Max:   3.1G
■ Share  Now:  0.0    Min:   0.0    Avg:   0.0    Max:   0.0
■ Cache  Now: 41.4G   Min: 616.1M   Avg:   9.5G   Max:  41.4G
■ Buffer Now: 71.5M   Min:  71.5M   Avg: 139.4M   Max: 143.6M
■ Swap   Now: 92.3M   Min:  92.3M   Avg:  92.4M   Max:  92.5M
■ Total  Now: 47.3G   Min:  47.3G   Avg:  47.3G   Max:  47.3G
```



```
■ User   Now:  0.1%   Min:  0.0%   Avg:  0.1%   Max:  0.2%
■ Nice   Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
■ System Now:  1.4%   Min:  0.9%   Avg:  1.7%   Max:  5.9%
■ Wait   Now: 78.6%   Min:  0.0%   Avg: 19.0%   Max: 79.3%
■ Idle   Now: 19.9%   Min: 19.4%   Avg: 79.2%   Max: 99.0%
```



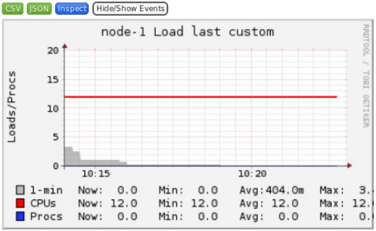
```
■ In    Now: 16.8k    Min: 15.0k    Avg: 71.9M    Max:122.8M
■ Out   Now:248.0k    Min:165.9k    Avg: 70.0M    Max:120.9M
```

Identify your bottleneck (I/O)

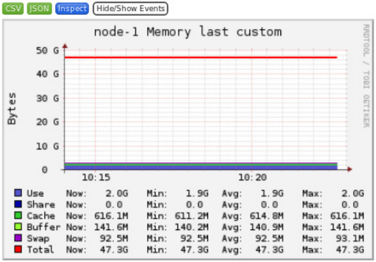
Application is limited by the speed of the storage

- Use local storage, eg `/tmp`, instead of network storage
- Use local memory, eg `/dev/shm`
- Use $SCRATCH (no backup)

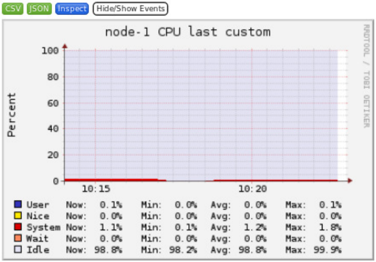
Identify your bottleneck (Network)



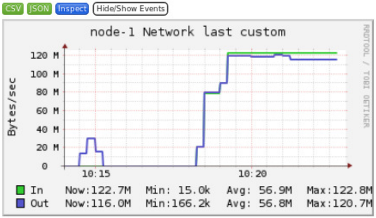
```
□ 1-min   Now:  0.0   Min:  0.0   Avg:404.0m   Max:  3.
■ CPUs    Now: 12.0   Min: 12.0   Avg: 12.0    Max: 12.
■ Procs   Now:  0.0   Min:  0.0   Avg:  0.0    Max:  0.
```



```
■ Use     Now:   2.0G   Min:   1.9G   Avg:   1.9G   Max:   2.0G
■ Share   Now:   0.0    Min:   0.0    Avg:   0.0    Max:   0.0
■ Cache   Now: 616.1M   Min: 611.2M   Avg: 614.8M   Max: 616.1M
■ Buffer  Now: 141.6M   Min: 140.2M   Avg: 140.9M   Max: 141.6M
■ Swap    Now:  92.5M   Min:  92.5M   Avg:  92.5M   Max:  93.1M
■ Total   Now:  47.3G   Min:  47.3G   Avg:  47.3G   Max:  47.3G
```



```
■ User    Now:  0.1%   Min:  0.0%   Avg:  0.0%   Max:  0.1%
■ Nice    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
■ System  Now:  1.1%   Min:  0.1%   Avg:  1.2%   Max:  1.8%
■ Wait    Now:  0.0%   Min:  0.0%   Avg:  0.0%   Max:  0.0%
□ Idle    Now: 98.8%   Min: 98.2%   Avg: 98.8%   Max: 99.9%
```



```
■ In    Now:122.7M   Min: 15.0k   Avg: 56.9M   Max:122.8M
■ Out   Now:116.0M   Min:166.2k   Avg: 56.8M   Max:120.7M
```

Identify your bottleneck (Network)

Application is limited by the speed of the network (too many communications)

- Use Infiniband instead of Ethernet
- Reduce the number of nodes

Command line tools

Connect to the node during your passive job

- SLURM: `srun -jobid <jobid> -pty bash`

- memory usage: `free -m`
- list current processes and statistics: `ps aux`
- processes ordered by CPU usage: `top`
- like top, but interactive: `htop`
- filesystem usage: `df-ulhpc`
- live system statistics (including I/O and network): `dstat`

Summary

1 Objectives

2 Experiment planning & workflow

3 Hardware knowledge

4 Live status

5 **Common mistakes and pitfalls**

6 Getting help

Common pitfalls

My job has been terminated, why ?

1. Maximum memory usage exceeded
   - ↪ The Linux *Out Of Memory Killer* (OOMK) mechanism killed your processes silently.
   - ↪ The available memory depends on the number of cores/CPUs reserved
   - ↪ Use the parameter `-mem-per-cpu` with srun
2. Requested walltime exceeded
   - ↪ The walltime specified with your submission command (sbatch, srun) was too short
   - ↪ Your job walltime cannot be extended after its submission
   - ↪ The job duration must be estimated before its submission
   - ↪ Use the parameter `-time` with srun

Summary

1 Objectives

2 Experiment planning & workflow

3 Hardware knowledge

4 Live status

5 Common mistakes and pitfalls

6 **Getting help**

Know the basics!

Access the clusters, access and reserve nodes

- Use SSH and public key authentication
  `https://hpc.uni.lu/users/docs/access.html`
- Learn how to use the SLURM batch scheduler (Iris cluster only)
  `https://hpc.uni.lu/users/docs/slurm.html`

Transfer files between your computer and the clusters

- Learn how to use tools like `scp`, `rsync`, etc.
  `https://hpc.uni.lu/users/docs/filetransfer.html`

Use pre-installed software

- Search and use software with the `module` command
  `https://hpc.uni.lu/users/docs/modules.html`

In this order

1. Check the UL HPC quick reference
   `https://hpc.uni.lu/download/documents/ulhpc-quickref.pdf`
2. Read The Fine Manual at `https://hpc.uni.lu/docs`
3. `$ man man`
4. Google is your friend!
5. Open a ticket on
   `hpc-tracker.uni.lu`
6. Ask the HPC sysadmins
   `hpc-sysadmins@uni.lu`
7. Bonus: ask the users community mailing list
   `hpc-users@uni.lu`

Thank you for your attention...

Questions?

http://hpc.uni.lu

High Performance Computing @ uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette
Valentin Plugaru
Sarah Peter
Hyacinthe Cartiaux
Clement Parisot
Dr. Fréderic Pinel
Dr. Emmanuel Kieffer

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. Objectives
2. Experiment planning & workflow
3. Hardware knowledge
4. Live status
5. Common mistakes and pitfalls
6. Getting help