UL HPC School 2017

Overview & Challenges of the UL HPC Facility at the Belval & EuroHPC Horizon

Prof. Pascal Bouvry, Dr. Sebastien Varrette

and the UL HPC Team

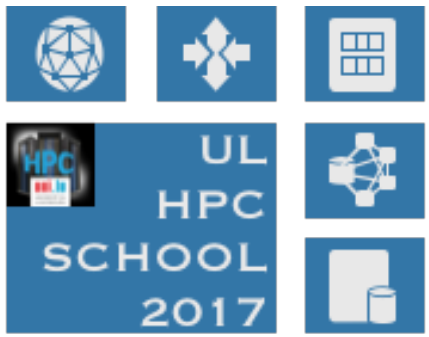
June. 12th, 2017, MSA 4.510

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

uni.lu
UNIVERSITÉ DU
LUXEMBOURG

Welcome to the UL HPC School 2017



https://hpc.uni.lu/hpc-school/

- **5th edition** of this training session...
  - ↪ previous editions in 2014, 2015 and 2016
  - ↪ This one is the "full" version
    - ✓ 2-days event
    - ✓ Parallel sessions, feat. basic & advanced tutorials
- **Requirement**:
  - ↪ your favorite laptop with your favorite OS
    - ✓ Linux / Mac OS preferred, but Windows accepted
  - ↪ basic knowledge in Linux command line
  - ↪ ability to take notes (Markdown etc.)

- Next edition planned for **Nov., 2017** in Belval
  - ↪ 1-days event, mainly targeting newcomers
  - ↪ focusing on the basic tutorials

Agenda Day 1: June 12th, 2017

Time	Main Track (MSA 4.510)
9h00 – 10h00	PS1: Getting Started on the UL HPC platform
10h00 – 10h30	**Coffee break**
10h30 – 12h30	**Overview and Challenges of the UL HPC Facility at the Belval and EuroHPC Horizon**
12h30 – 13h30	**LUNCH**
13h30 – 15h30	PS2: HPC workflow with sequential jobs (test cases on GROMACS, Java and Python)
15h30 – 16h00	**Coffee break**
16h00 – 17h00	PS4a: UL HPC Monitoring in practice: why, what, how, where to look
17h00 – 18h30	PS5: HPC workflow with Parallel/Distributed jobs

Time	Advanced Parallel Track (MSA 4.520)
9h00 – 10h00	
10h00 – 10h30	**Coffee break**
10h30 – 12h30	**Overview and Challenges of the UL HPC Facility at the Belval and EuroHPC Horizon**
12h30 – 13h30	**LUNCH**
13h30 – 15h30	PS3: Advanced Scheduling (Slurm, OAR) and Software Customization
15h30 – 16h00	**Coffee break**
16h00 – 17h00	PS4b: Debugging, profiling and performance analysis
17h00 – 18h30	PS6: Bioinformatics workflows and applications

PS = *Practical Session using your laptop*

Agenda Day 2: June 12th, 2017

Time	Main Track (MSA 4.510)
9h00 – 10h30	PS7: Big Data Applications
10h30 – 11h00	**Coffee break**
11h00 – 12h30	Users' session: UL HPC experiences
12h30 – 13h30	**LUNCH**
13h30 – 15h00	PS9: [Advanced] Prototyping with Python
15h30 – 16h00	**Coffee break**
16h00 – 17h30	PS10: R - statistical computing
17h30 – 18h30	Closing Keynote: Take Away Messages

Time	Advanced Parallel Track (MSA 4.520)
9h00 – 10h30	PS8: MATLAB (interactive, passive, sequential, checkpointing and parallel)
10h30 – 11h00	**Coffee break**
11h00 – 12h30	
12h30 – 13h30	**LUNCH**
13h30 – 15h00	PS11: Multi-Physics workdflows (CFD / MD / Chemistry Applications)
15h30 – 16h00	**Coffee break**
16h00 – 17h30	PS12: Virtualization
17h30 – 18h30	

PS = *Practical Session using your laptop*

Summary

Summary

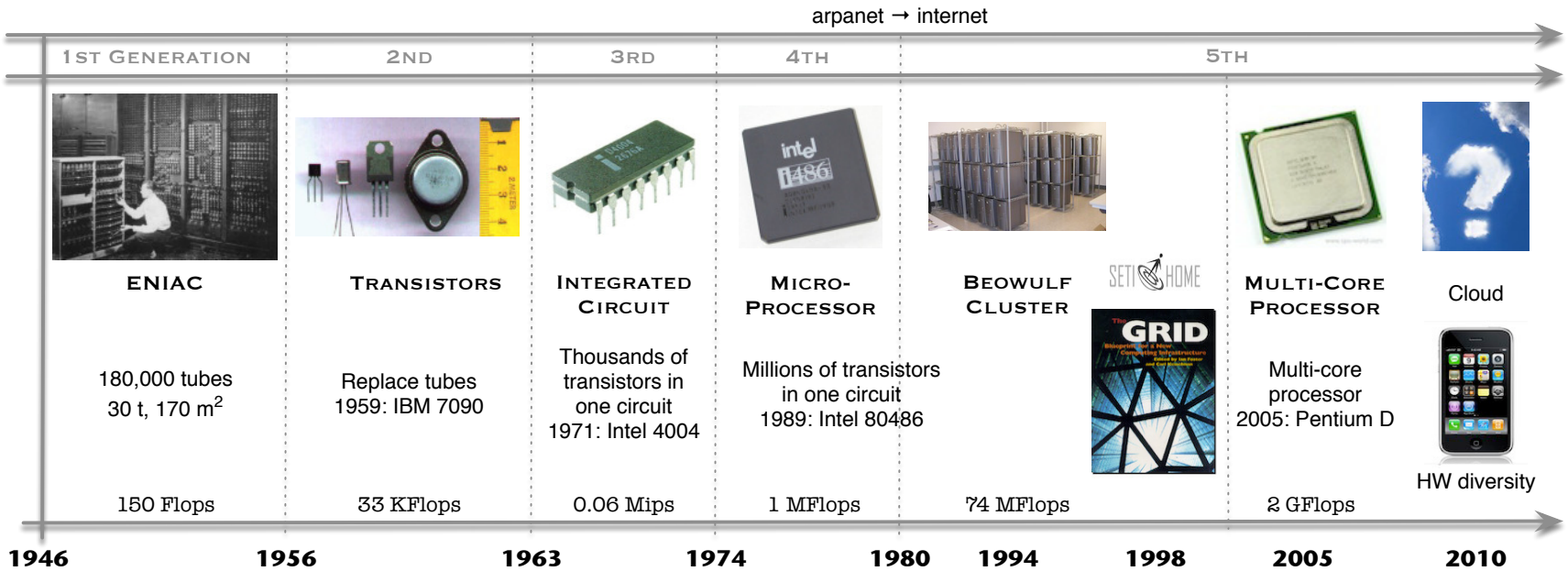
Prerequisites

- **HPC**: **H**igh **P**erformance **C**omputing

Main HPC Performance Metrics

- Computing Capacity/speed: often measured in **flops** (or **flop/s**)
  - ↪ **Fl**oating **p**oint **o**perations per **s**econds (often in DP)
  - ↪ **GFlops** $= 10^9$ Flops **TFlops** $= 10^{12}$ Flops **PFlops** $= 10^{15}$ Flops
- Storage Capacity: measured in multiples of **bytes** $= 8$ **bits**
  - ↪ **GB** $= 10^9$ bytes **TB** $= 10^{12}$ bytes **PB** $= 10^{15}$ bytes
  - ↪ **GiB** $= 1024^3$ bytes **TiB** $= 1024^4$ bytes **PiB** $= 1024^5$ bytes
- Transfert rate on a medium measured in **Mb/s** or **MB/s**
- Other metrics: Sequential vs Random **R/W speed**, **IOPS** ...

Evolution of Computing Systems



Why High Performance Computing ?

"The country that out-computes will be the one that out-competes". *Council on Competitiveness*

- **Accelerates** research by accelerating **computations**

≃ 20 **G**Flops (Dual-core i5 1.6GHz)

198.172 **T**Flops (594 computing nodes,8228 cores)

- Increases **storage** capacity and velocity for Big Data processing

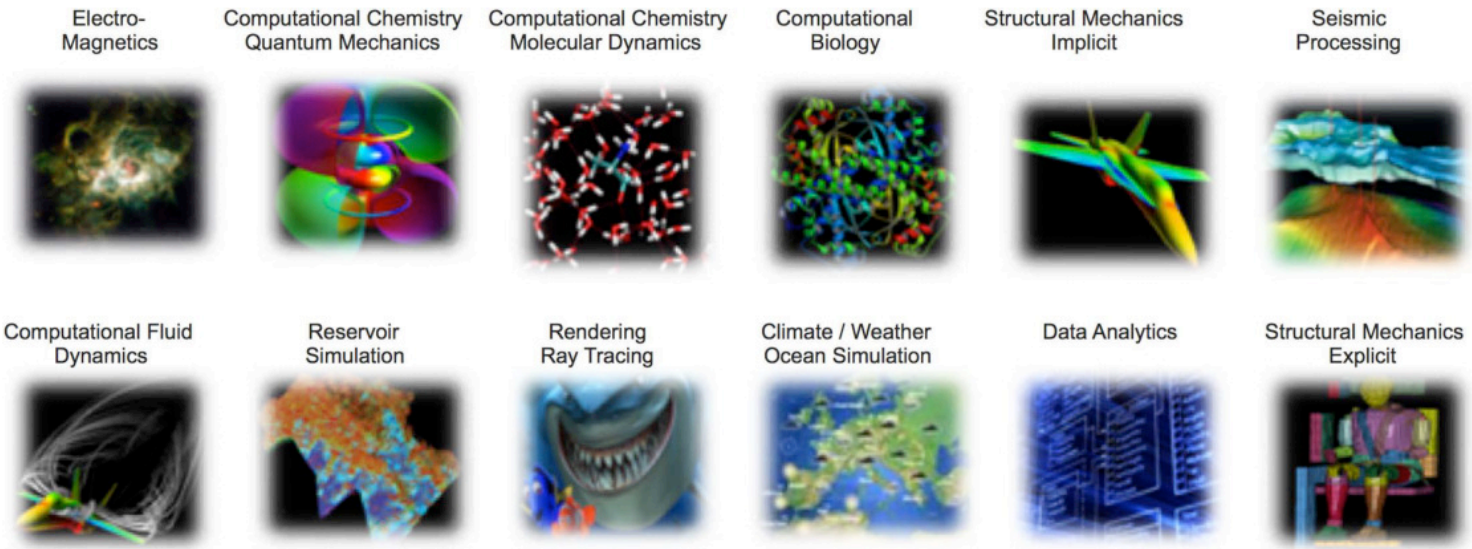
2TB (1 disk, 300 MB/s)

6856.4TB (1558 disks, 7 GB/s)

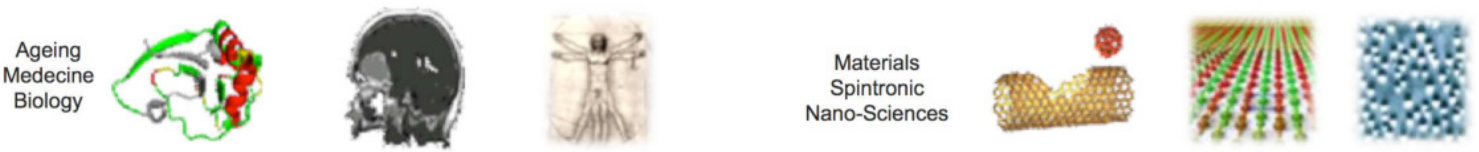
- **Communicates faster** 1 GbE (1 Gb/s) vs Infiniband QDR (40 Gb/s)

HPC at the Heart of our Daily Life

- **Today**: Research, Industry, Local Collectivities



- **. . . Tomorrow**: applied research, digital health, nano/bio techno

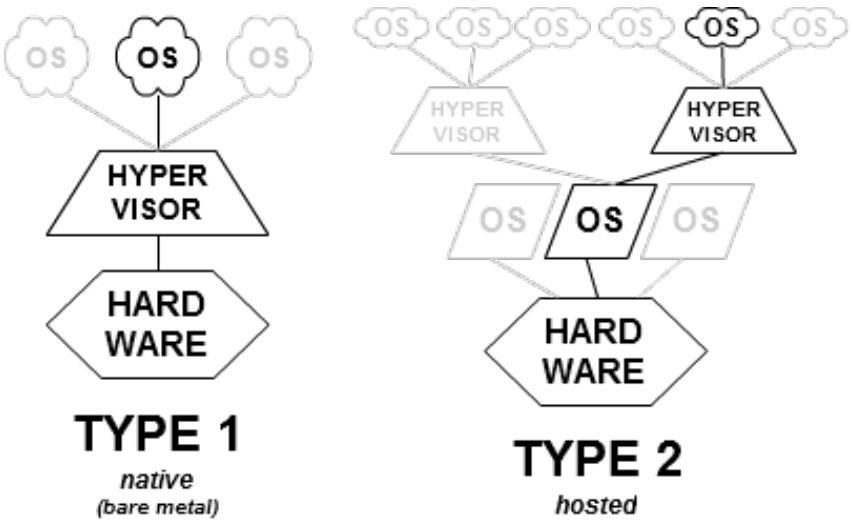


Computing for Researchers: Laptop

- Regular PC / Local Laptop / Workstation
  - ↪ Native OS (Windows, Linux, Mac etc.)

Computing for Researchers: Laptop

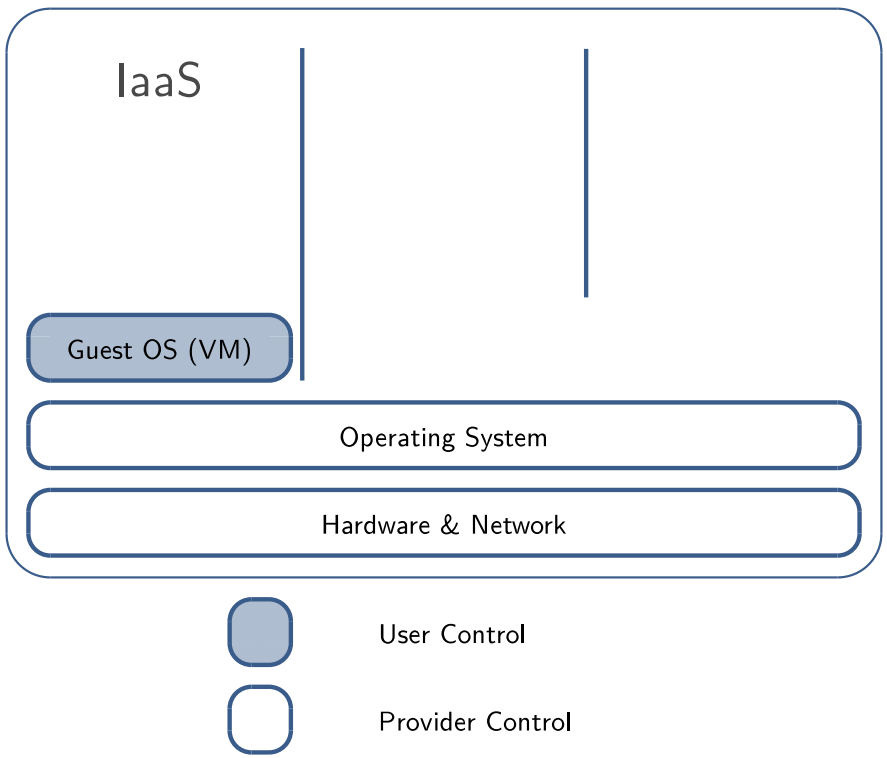
- Regular PC / Local Laptop / Workstation
  - ↪ Native OS (Windows, Linux, Mac etc.)
  - ↪ Virtualized OS through an **hypervisor**
    - ✓ Hypervisor: core virtualization engine / environment
    - ✓ **Performance loss:** $\geq 20\%$



Xen, VMWare ESXi, KVM VirtualBox

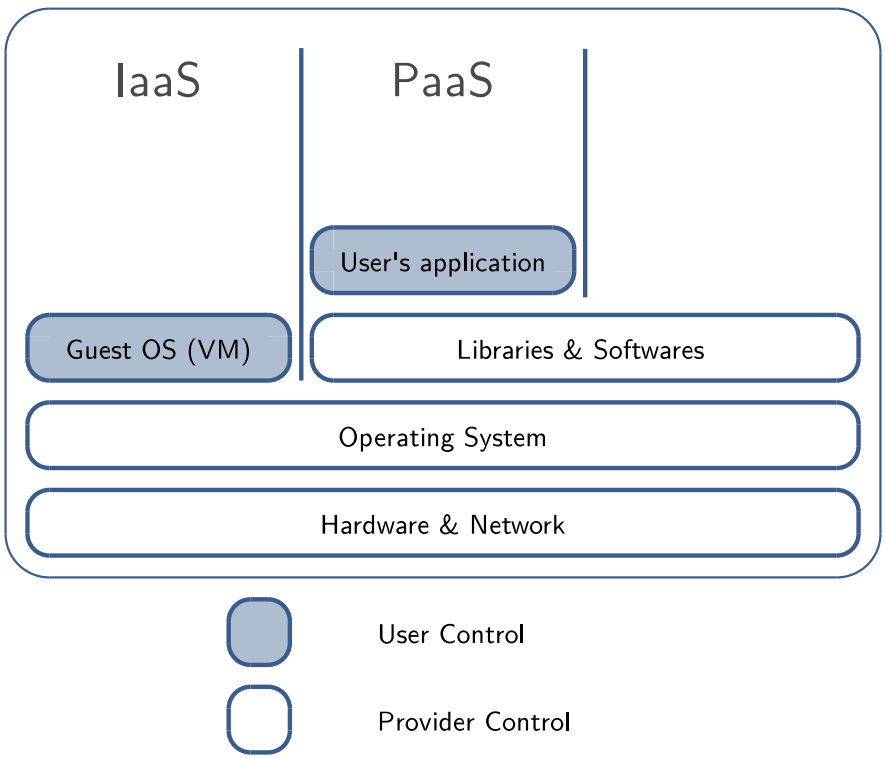
Computing for Researchers: Cloud

- Cloud Computing
  - ↪ access to shared *(generally virtualized)* resources in a pay-per-use manner
  - ↪ **Infrastructure** as a Service (**S**aaS)



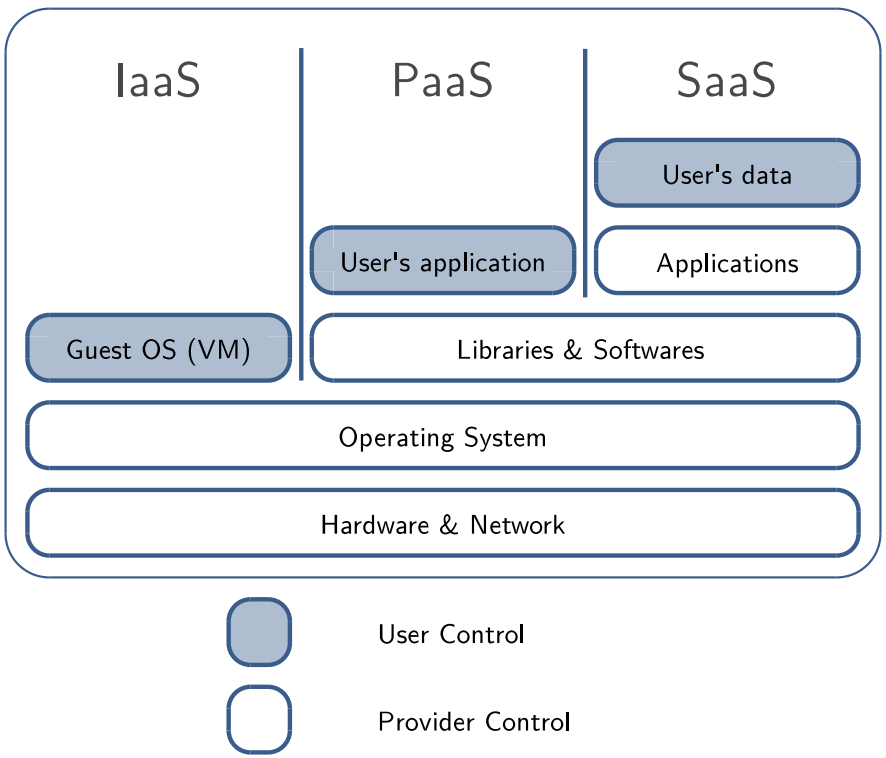
Computing for Researchers: Cloud

- Cloud Computing
  - ↪ access to shared *(generally virtualized)* resources in a pay-per-use manner
  - ↪ **Platform** as a Service (**P**aaS)



Computing for Researchers: Cloud

- Cloud Computing
  - ↪ access to shared *(generally virtualized)* resources in a pay-per-use manner
  - ↪ **Software** as a Service (**I**aaS)



Computing for Researchers: HPC

- High Performance Computing (HPC) platforms
  - ↪ For **Speedup**, **Scalability** and **Faster Time to Solution**

Computing for Researchers: HPC

- High Performance Computing (HPC) platforms
  - ↪ For **Speedup**, **Scalability** and **Faster Time to Solution**

YET...

$$\textbf{PC} \neq \textbf{Cloud} \neq \textbf{HPC}$$

Computing for Researchers: HPC

- High Performance Computing (HPC) platforms
  - ↪ For **Speedup**, **Scalability** and **Faster Time to Solution**

YET...

PC $\neq$ Cloud $\neq$ HPC

- HPC $\simeq$ Formula 1
  - ↪ relies on ultra efficient hardware / interconnect (IB EDR...)
  - ↪ ... when Cloud has to stay standard ([10] GbE etc...)

- **Does not mean the 3 approaches cannot work together**

Jobs, Tasks & Local Execution

```
$> ./myprog
```

Jobs, Tasks & Local Execution

```
$> ./myprog
```

Jobs, Tasks & Local Execution

```
$> ./myprog
$> ./myprog -n 10
```

Jobs, Tasks & Local Execution

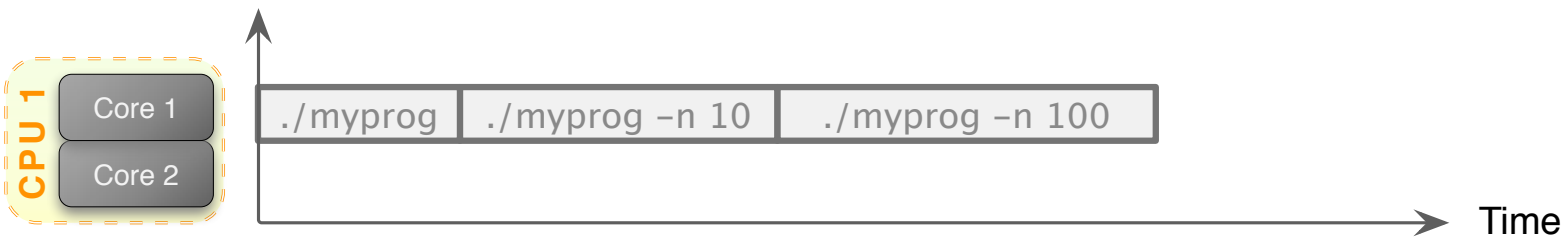
```
$> ./myprog
$> ./myprog -n 10
```

Jobs, Tasks & Local Execution

```
$> ./myprog
$> ./myprog -n 10
$> ./myprog -n 100
```

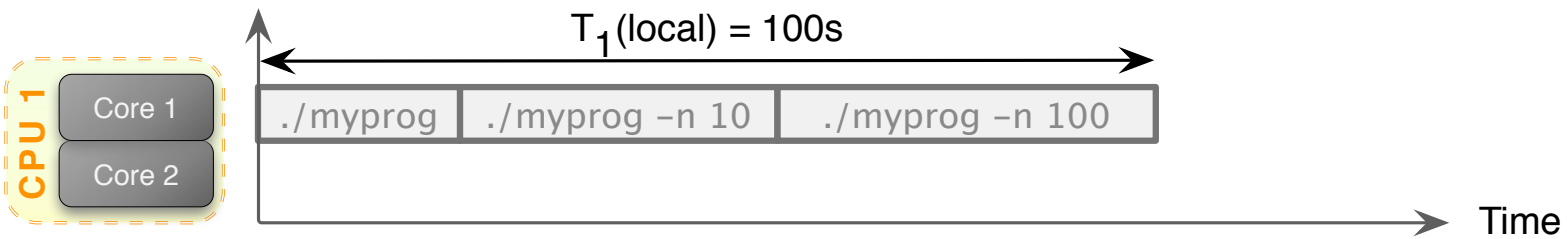
Jobs, Tasks & Local Execution

```
$> ./myprog
$> ./myprog -n 10
$> ./myprog -n 100
```



Jobs, Tasks & Local Execution

```
$> ./myprog
$> ./myprog -n 10
$> ./myprog -n 100
```

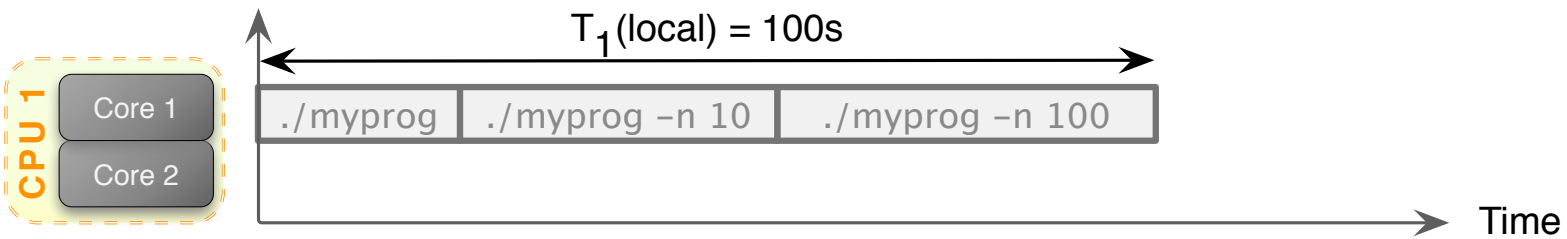


Jobs, Tasks & Local Execution

```
$> ./myprog
$> ./myprog -n 10
$> ./myprog -n 100
```

Job(s) 3

Task(s) 3



Jobs, Tasks & Local Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

Jobs, Tasks & Local Execution

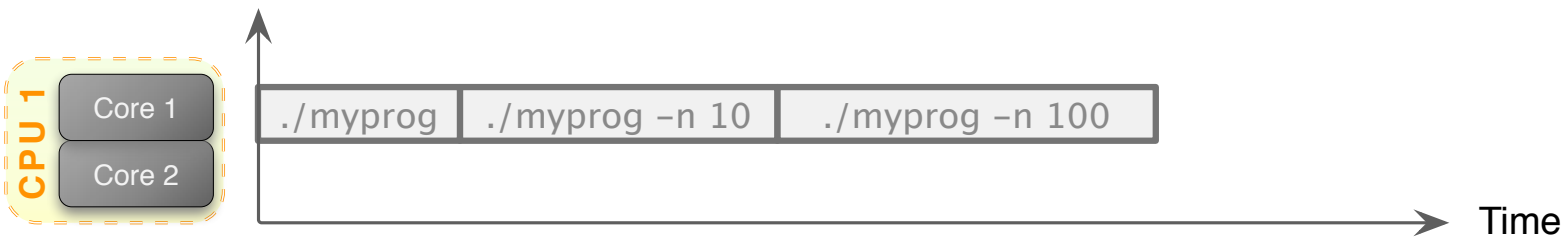
```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

Jobs, Tasks & Local Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

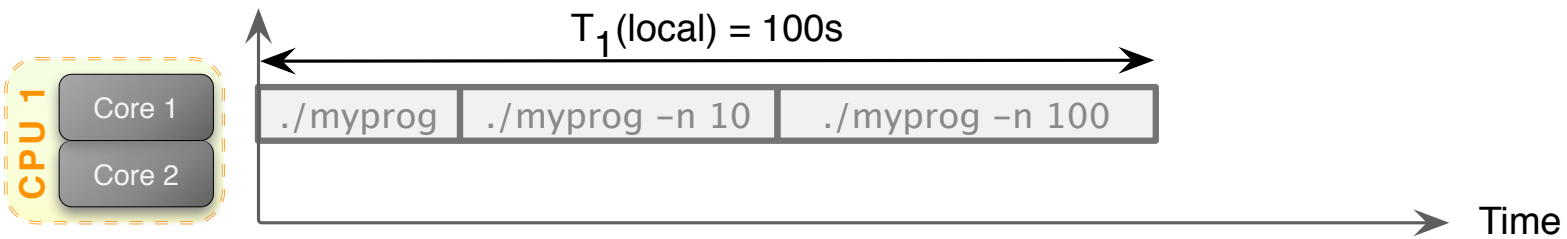
Jobs, Tasks & Local Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```



Jobs, Tasks & Local Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

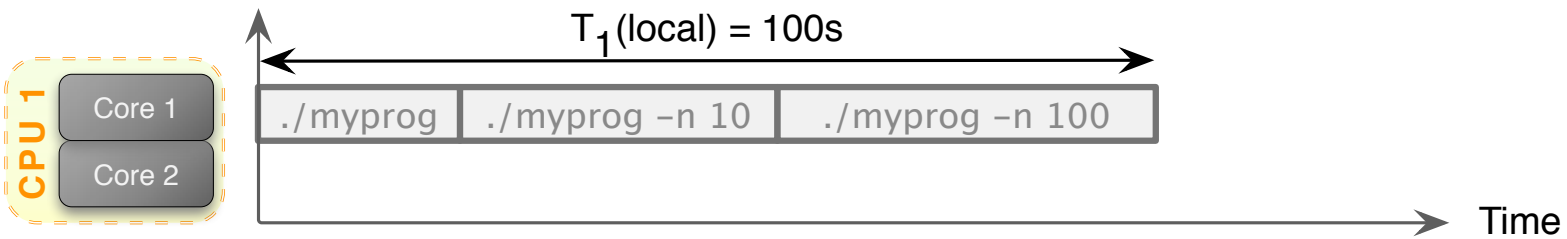


Jobs, Tasks & Local Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

job(s) 1

Task(s) 3



Jobs, Tasks & Local Execution

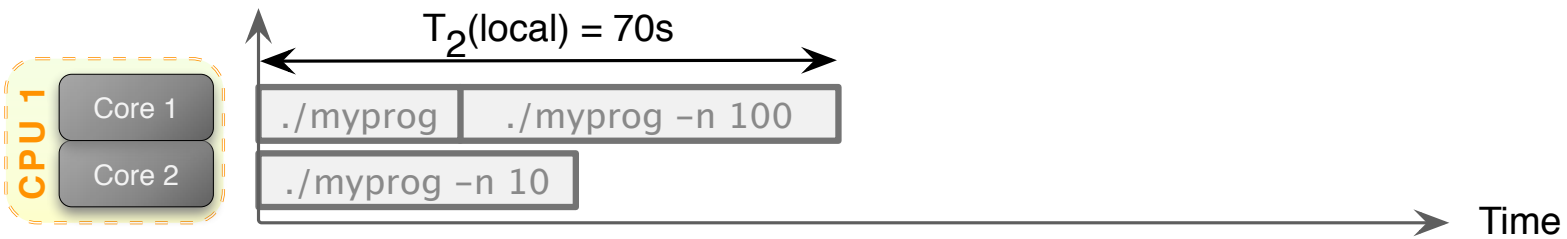
```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

Jobs, Tasks & Local Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

Jobs, Tasks & Local Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

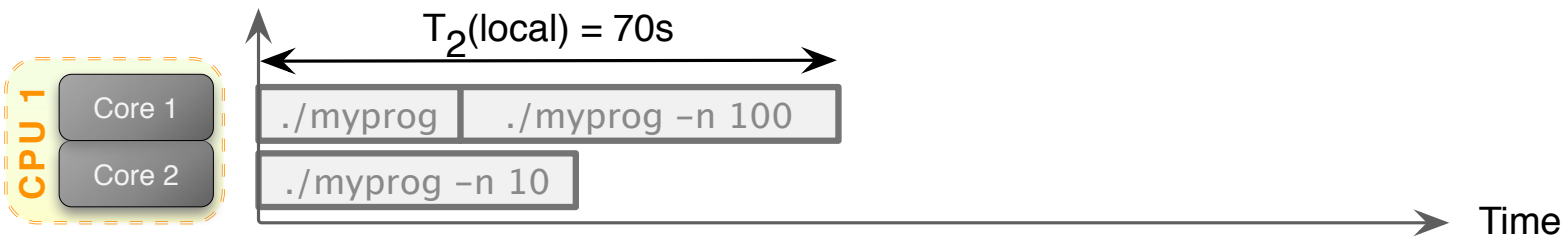


Jobs, Tasks & Local Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

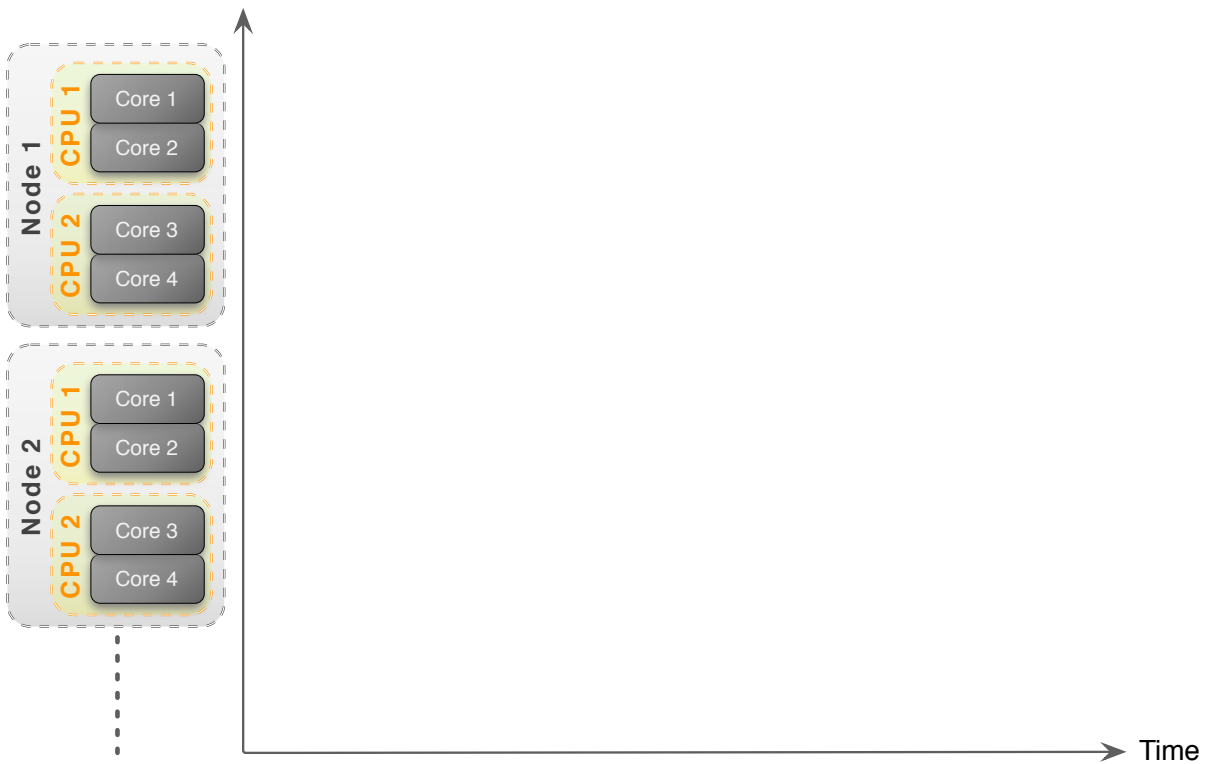
job(s) 1

Task(s) 3



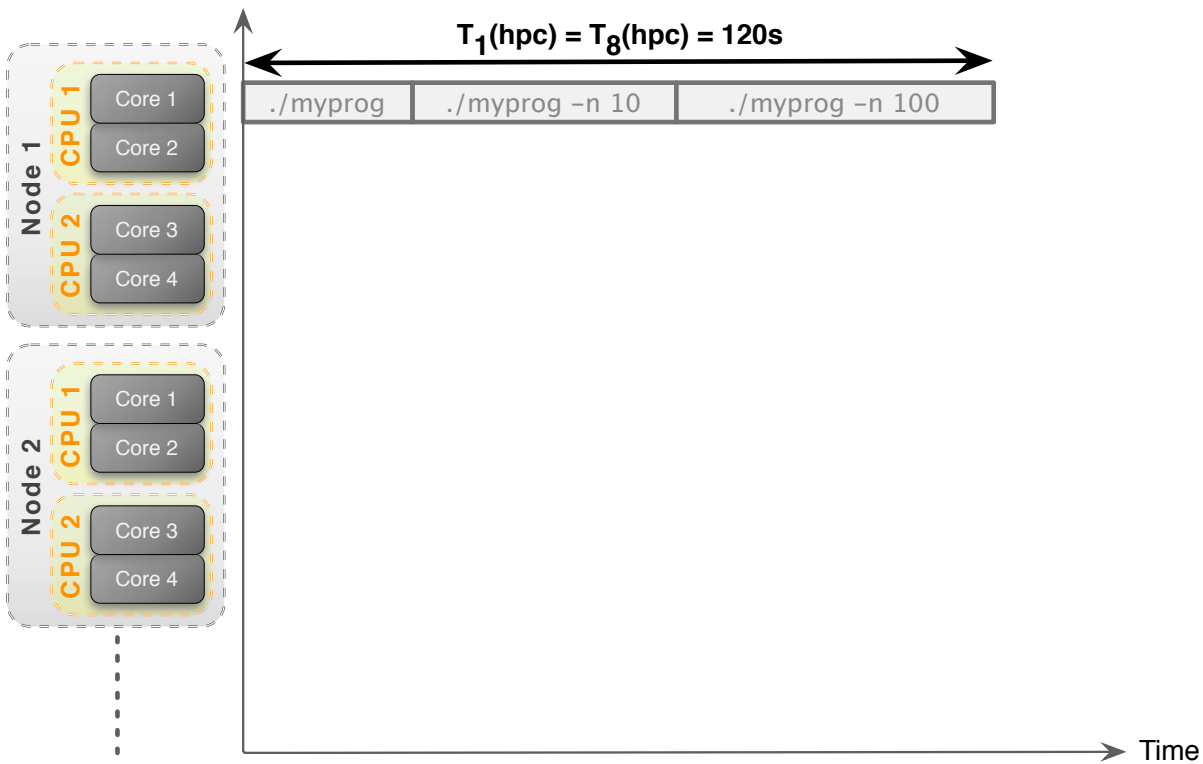
Jobs, Tasks & HPC Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```



Jobs, Tasks & HPC Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

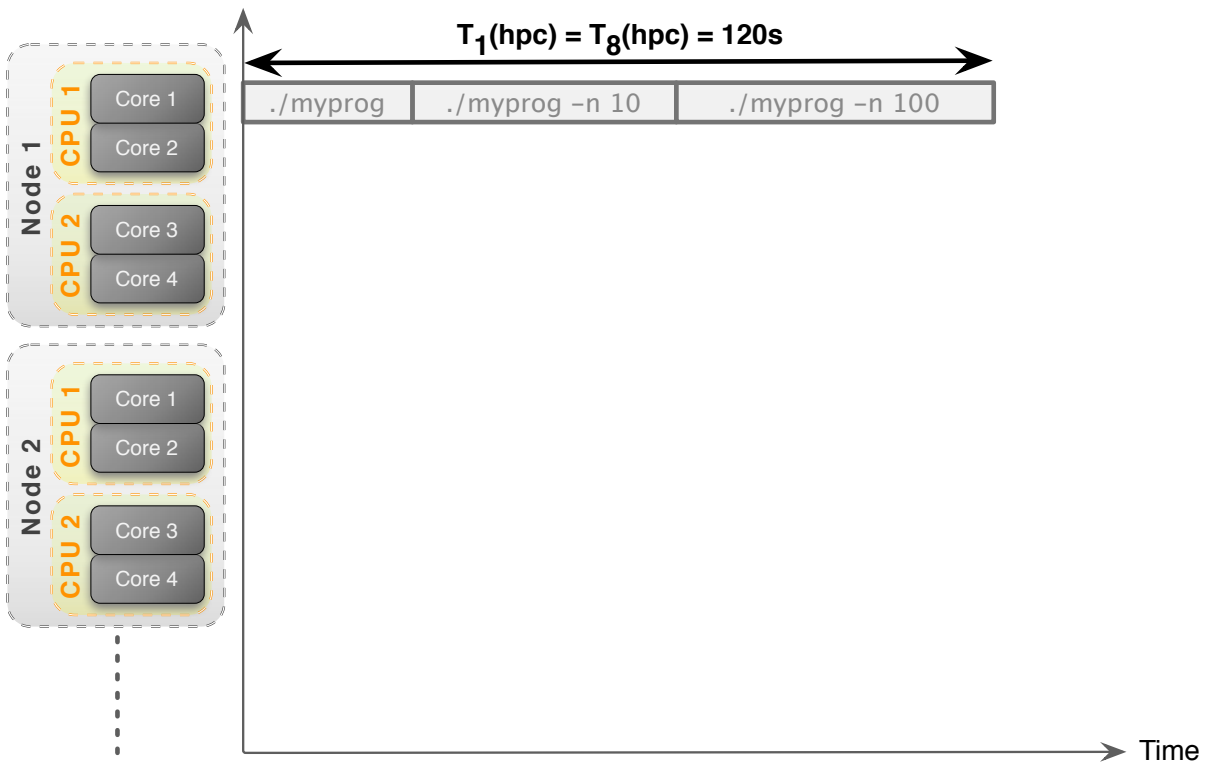


Jobs, Tasks & HPC Execution

```
# launcher

./myprog
./myprog -n 10
./myprog -n 100
```

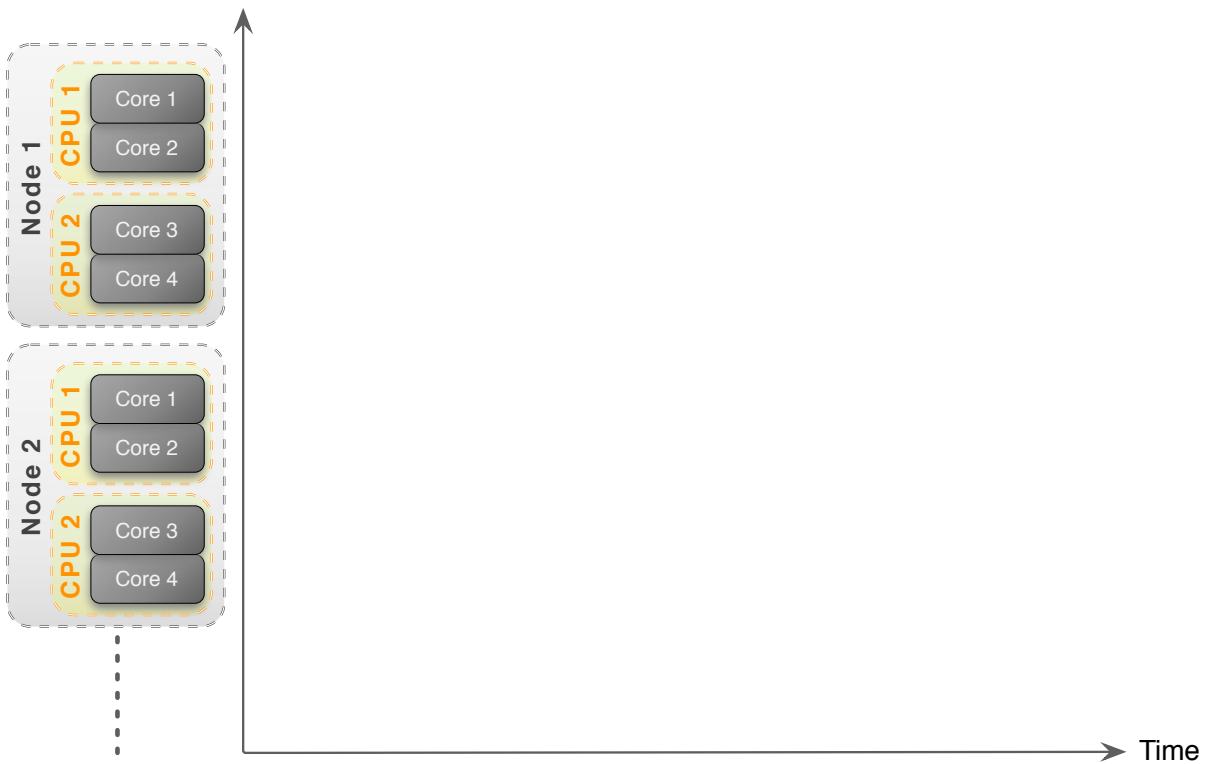
Job(s) 1

Task(s) 3



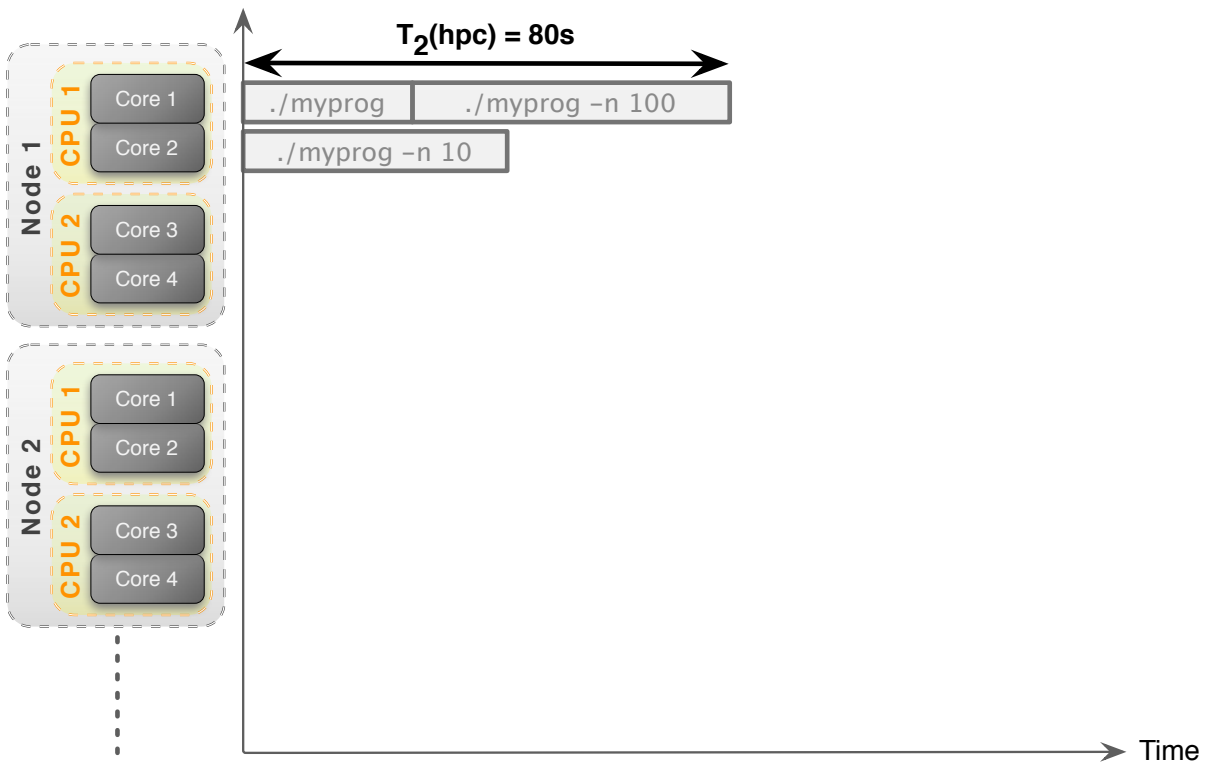
Jobs, Tasks & HPC Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```



Jobs, Tasks & HPC Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

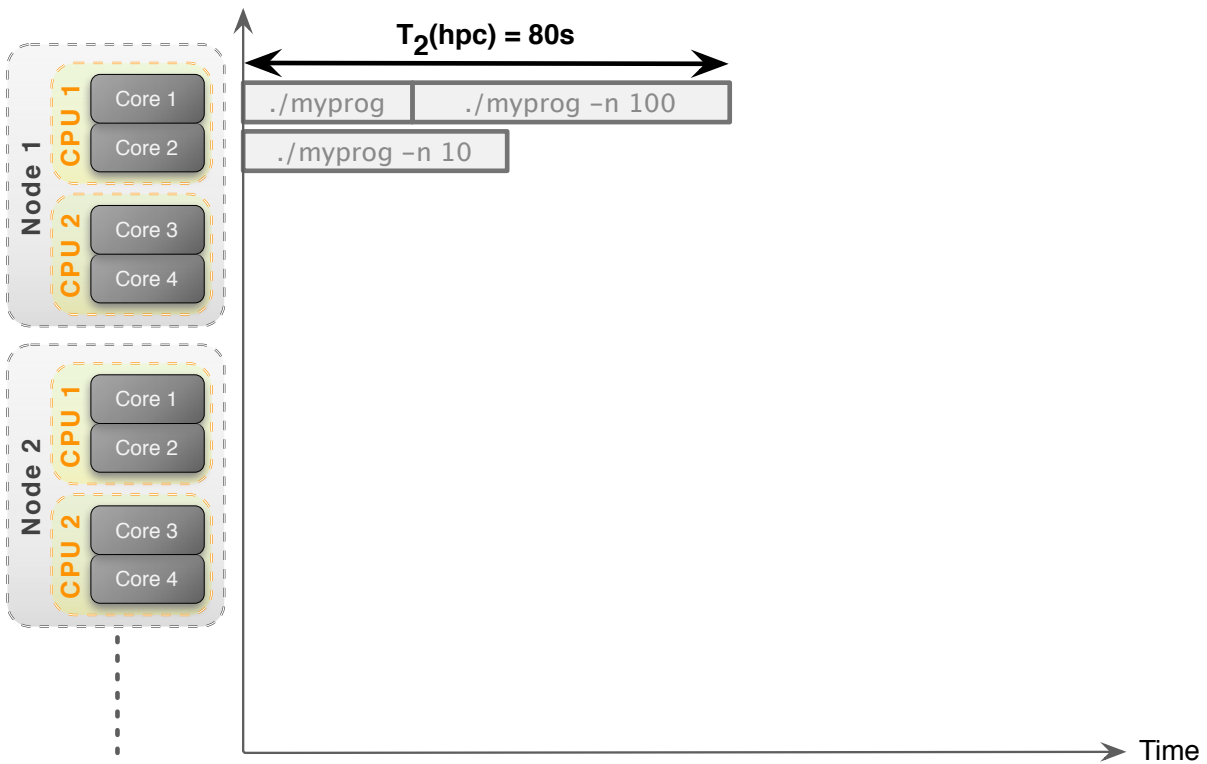


Jobs, Tasks & HPC Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

Job(s) 1

Task(s) 3

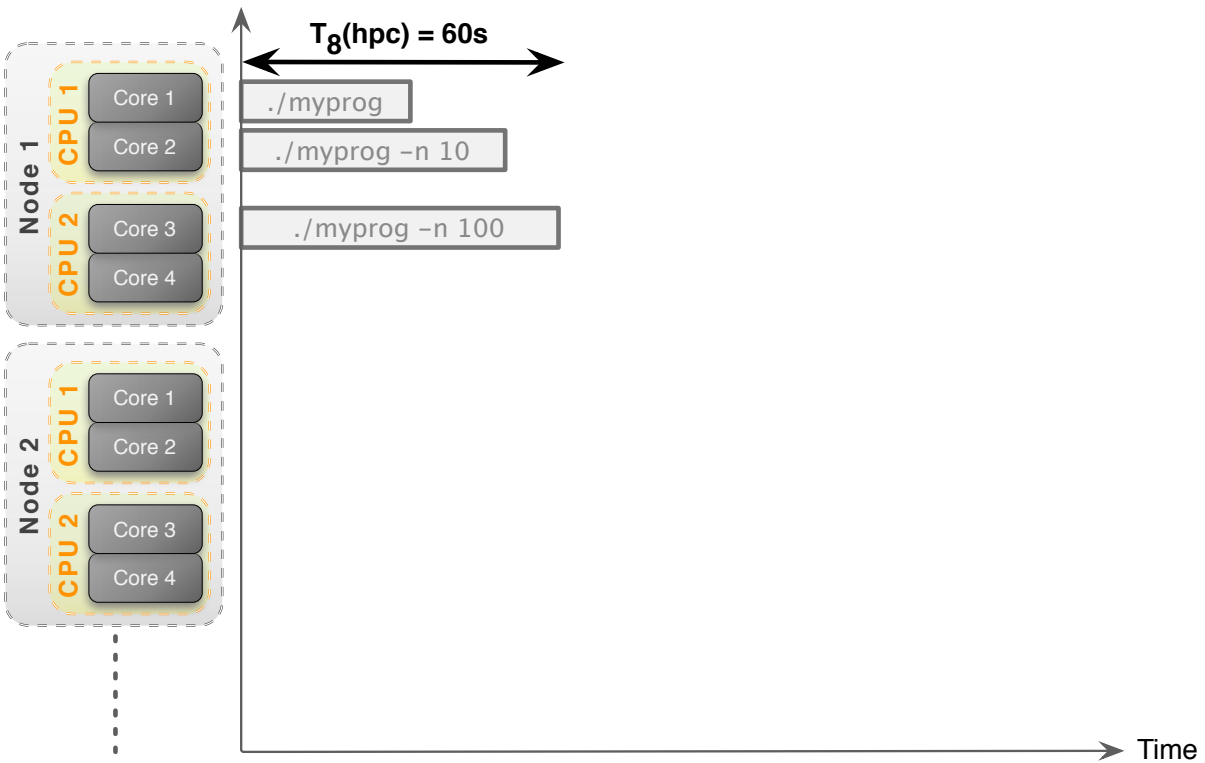


Jobs, Tasks & HPC Execution

```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```

Job(s) 1

Task(s) 3



Local vs. HPC Executions

Context	Local PC	HPC
Sequential	$T_1(\text{local}) = 100$	$T_1(\text{hpc}) = \mathbf{120s}$
Parallel/Distributed	$T_2(\text{local}) = 70\text{s}$	$T_2(\text{hpc}) = 80\text{s}$
		$T_8(\text{hpc}) = \mathbf{60s}$

Local vs. HPC Executions

Context	Local PC	HPC
Sequential	$T_1(\text{local}) = 100$	$T_1(\text{hpc}) = \mathbf{120s}$
Parallel/Distributed	$T_2(\text{local}) = 70\text{s}$	$T_2(\text{hpc}) = 80\text{s}$
		$T_8(\text{hpc}) = \mathbf{60s}$

- **Sequential** runs **WON'T BE FASTER** on HPC
  - ↪ Reason: Processor Frequency (typically 3GHz vs 2.26GHz)

Local vs. HPC Executions

Context	Local PC	HPC
Sequential	$T_1(\text{local}) = 100$	$T_1(\text{hpc}) = \mathbf{120s}$
Parallel/Distributed	$T_2(\text{local}) = 70\text{s}$	$T_2(\text{hpc}) = 80\text{s}$
		$T_8(\text{hpc}) = \mathbf{60s}$

- **Sequential** runs **WON'T BE FASTER** on HPC
  - ↪ Reason: Processor Frequency (typically 3GHz vs 2.26GHz)

- **Parallel/Distributed** runs **DO NOT COME FOR FREE**
  - ↪ runs **will be sequential** even if you reserve ≥ 2 cores/nodes
  - ↪ you have to **explicitly** adapt your jobs to benefit from the multi-cores/nodes

Identifying Potential Parallelism

In your workflow

```
$> ./my_sequential_prog -n 1
$> ./my_sequential_prog -n 2
$> ./my_sequential_prog -n 3
$> ./my_sequential_prog -n 4
$> ./my_sequential_prog -n 5
$> ./my_sequential_prog -n 6
$> ./my_sequential_prog -n 7
$> ...
```

Identifying Potential Parallelism

```
x = initX(A, B);
y = initY(A, B);
z = initZ(A, B);

for(i = 0; i < N_ENTRIES; i++)
        x[i] = compX(y[i], z[i]);

for(i = 1; i < N_ENTRIES; i++)
        x[i] = solveX(x[i-1]);

finalize1 (&x, &y, &z);
finalize2 (&x, &y, &z);
finalize3 (&x, &y, &z);
```

Identifying Potential Parallelism

```
x = initX(A, B);
y = initY(A, B);
z = initZ(A, B);
```

Functional Parallelism

Identifying Potential Parallelism

```
x = initX(A, B);
y = initY(A, B);
z = initZ(A, B);
```

Functional Parallelism

```
for(i = 0; i < N_ENTRIES; i++)
        x[i] = compX(y[i], z[i]);
```

Data Parallelism

Identifying Potential Parallelism

```
x = initX(A, B);
y = initY(A, B);
z = initZ(A, B);
```

Functional Parallelism

```
for(i = 0; i < N_ENTRIES; i++)
        x[i] = compX(y[i], z[i]);
```

Data Parallelism

```
for(i = 1; i < N_ENTRIES; i++)
        x[i] = solveX(x[i-1]);
```

Pipelining

Identifying Potential Parallelism

```
x = initX(A, B);
y = initY(A, B);
z = initZ(A, B);
```

Functional Parallelism

```
for(i = 0; i < N_ENTRIES; i++)
        x[i] = compX(y[i], z[i]);
```

Data Parallelism

```
for(i = 1; i < N_ENTRIES; i++)
        x[i] = solveX(x[i-1]);
```

Pipelining

```
finalize1 (&x, &y, &z);
finalize2 (&x, &y, &z);
finalize3 (&x, &y, &z);
```

No good?

Summary

HPC Components: [GP]CPU

CPU

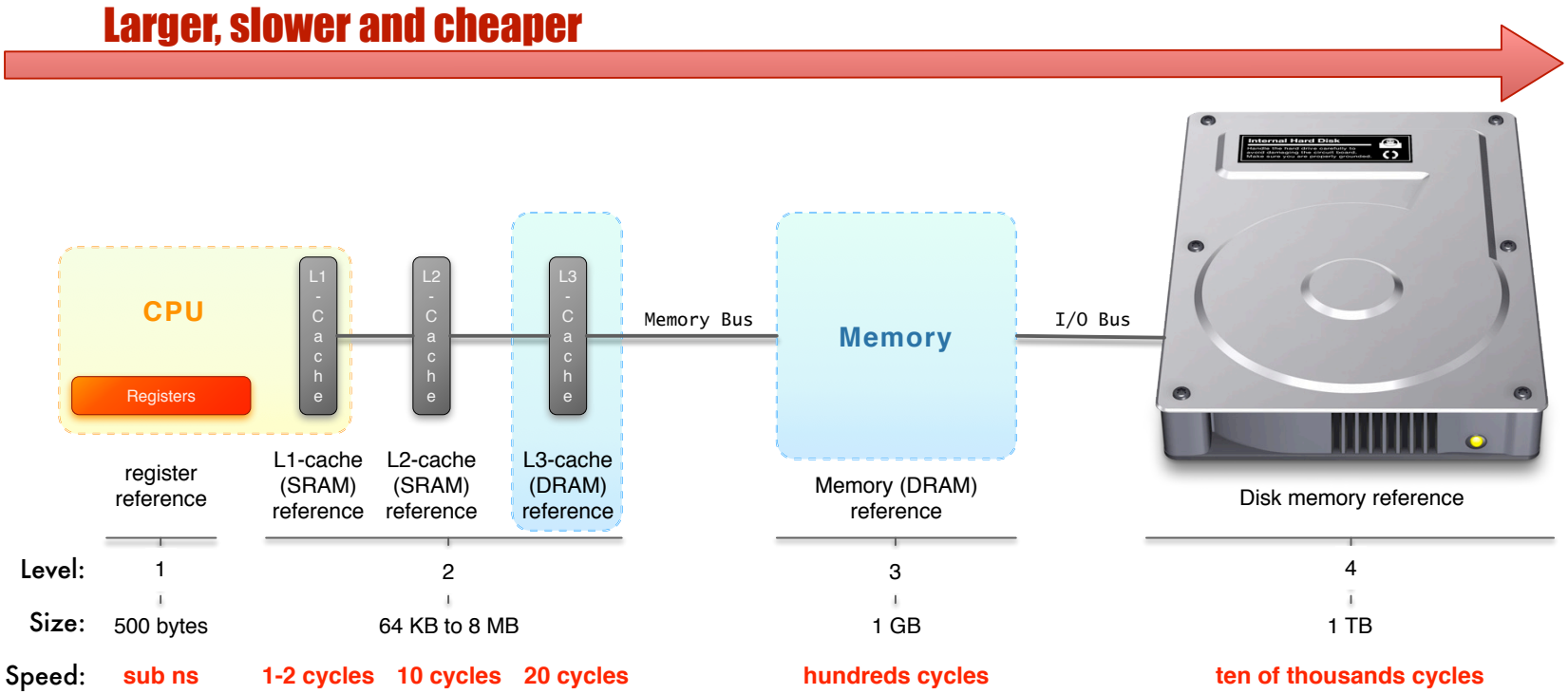
- Always multi-core
- Ex: Intel Core i7-970 (July 2010) $R_{peak} \simeq 100$ GFlops (DP)
  - ↪ 6 cores @ 3.2GHz (32nm, 130W, 1170 millions transistors)

GPU / GPGPU

- Always multi-core, optimized for vector processing
- Ex: Nvidia Tesla C2050 (July 2010) $R_{peak} \simeq 515$ GFlops (DP)
  - ↪ 448 cores @ 1.15GHz

$\simeq$ **10 Gflops** for 50 €

HPC Components: Local Memory

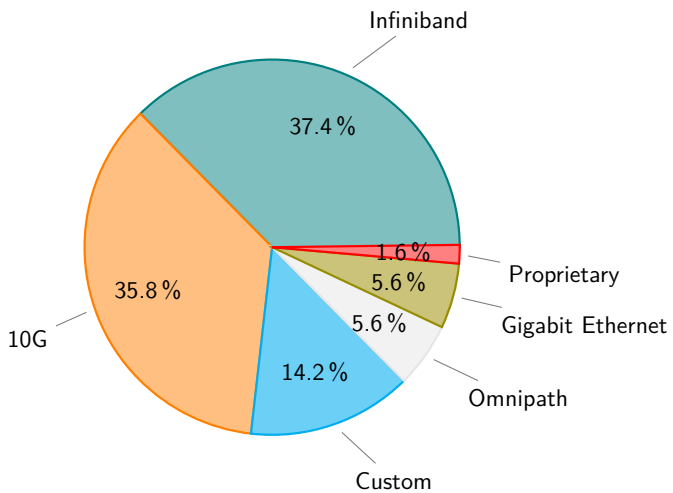


- SSD R/W: 560 MB/s; 85000 IOps **1000 €/TB**
- HDD (SATA @ 7,2 krpm) R/W: 100 MB/s; 190 IOps **100 €/TB**

HPC Components: Interconnect

- **latency**: time to send a minimal (0 byte) message from A to B
- **bandwidth**: max amount of data communicated per unit of time

Technology	Effective Bandwidth		Latency
Gigabit Ethernet	1 Gb/s	125 MB/s	40μs to 300μs
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	4μs to 5μs
Infiniband QDR	40 Gb/s	5 GB/s	1.29μs to 2.6μs
Infiniband EDR	100 Gb/s	12.5 GB/s	0.61μs to 1.3μs
100 Gigabit Ethernet	100 Gb/s	1.25 GB/s	30μs
Intel Omnipath	100 Gb/s	12.5 GB/s	0.9μs

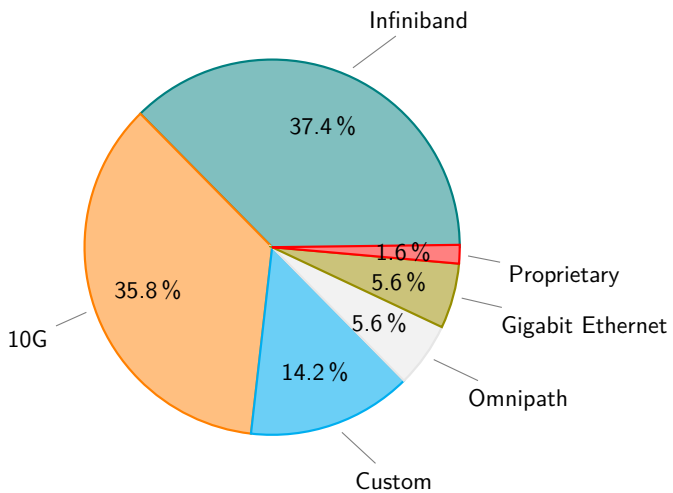


[Source : www.top500.org, Nov. 2016]

HPC Components: Interconnect

- **latency**: time to send a minimal (0 byte) message from A to B
- **bandwidth**: max amount of data communicated per unit of time

Technology	Effective Bandwidth		Latency
Gigabit Ethernet	1 Gb/s	125 MB/s	40μs to 300μs
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	4μs to 5μs
Infiniband QDR	40 Gb/s	5 GB/s	1.29μs to 2.6μs
Infiniband EDR	100 Gb/s	12.5 GB/s	0.61μs to 1.3μs
100 Gigabit Ethernet	100 Gb/s	1.25 GB/s	30μs
Intel Omnipath	100 Gb/s	12.5 GB/s	0.9μs



[Source : www.top500.org, Nov. 2016]

Network Topologies

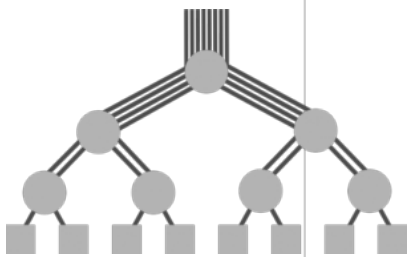
- **Direct** vs. **Indirect** interconnect
  - ↪ *direct*: each network node attaches to at least one compute node
  - ↪ *indirect*: compute nodes attached at the edge of the network only
    - ✓ many routers only connect to other routers.

Network Topologies

- **Direct** vs. **Indirect** interconnect
  - ↪ *direct*: each network node attaches to at least one compute node
  - ↪ *indirect*: compute nodes attached at the edge of the network only
    - ✓ many routers only connect to other routers.

Main HPC Topologies

- **CLOS Network / Fat-Trees** [Indirect]
  - ↪ can be fully non-blocking (1:1) or blocking (x:1)
  - ↪ typically enables **best performance**
    - ✓ Non blocking bandwidth, lowest network latency

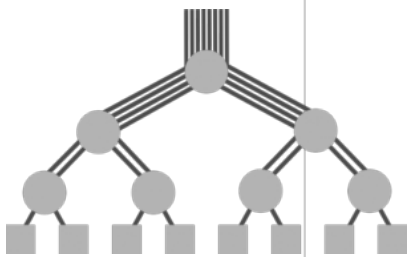


Network Topologies

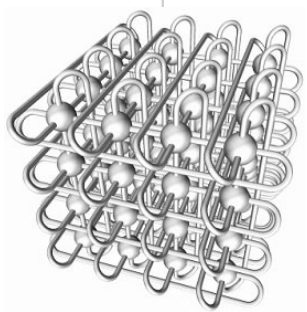
- **Direct** vs. **Indirect** interconnect
  - ↪ *direct*: each network node attaches to at least one compute node
  - ↪ *indirect*: compute nodes attached at the edge of the network only
    - ✓ many routers only connect to other routers.

Main HPC Topologies

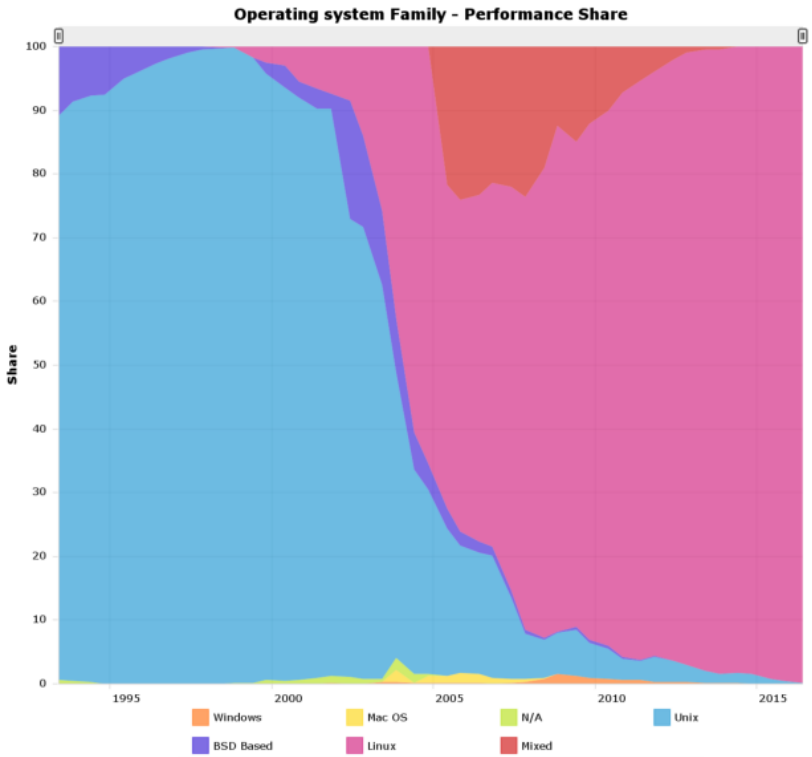
- **CLOS Network / Fat-Trees** [Indirect]
  - ↪ can be fully non-blocking (1:1) or blocking (x:1)
  - ↪ typically enables **best performance**
    - ✓ Non blocking bandwidth, lowest network latency



- **Mesh or 3D-torus** [Direct]
  - ↪ Blocking network, cost-effective for systems at scale
  - ↪ Great performance solutions for applications with locality
  - ↪ Simple expansion for future growth

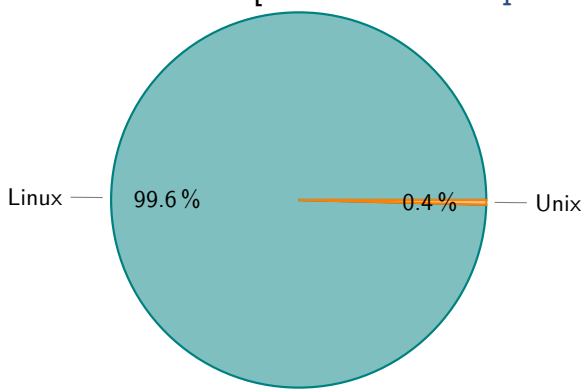


HPC Components: Operating System

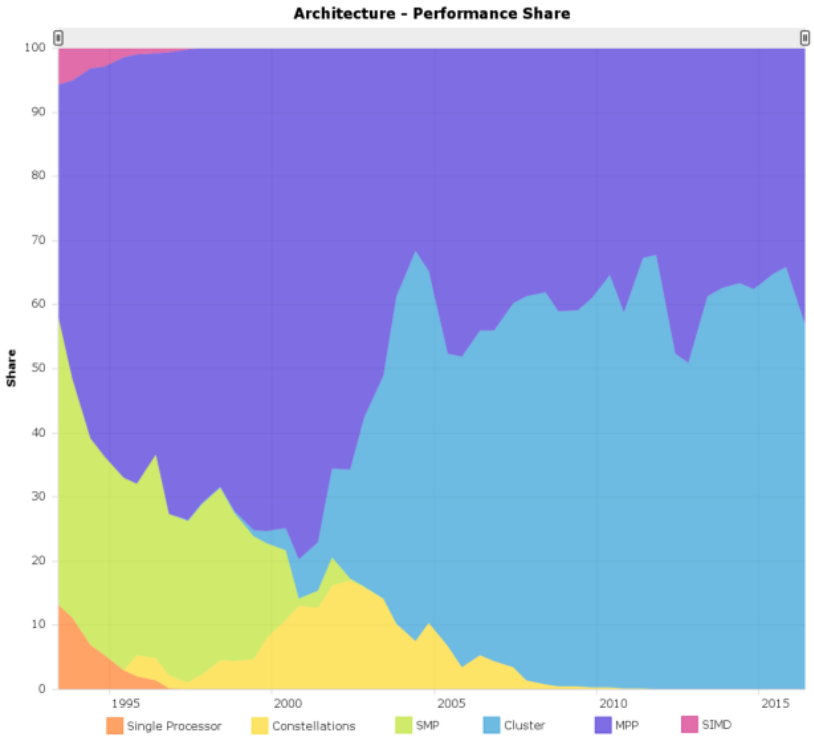


- Exclusively Linux-based (99.6%)
  - ↪ . . . or Unix (0.4%)
- Reasons:
  - ↪ stability
  - ↪ prone to devels

[Source : www.top500.org, Nov 2016]

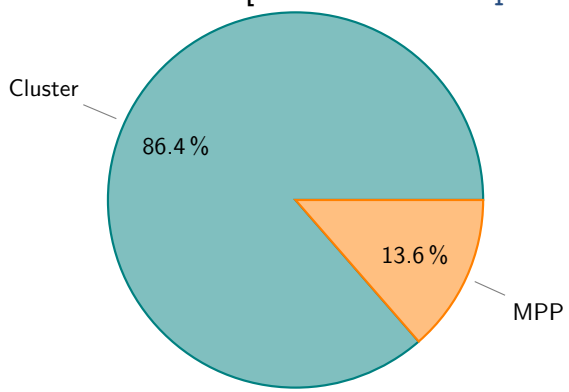


HPC Components: Architecture



- Mainly Cluster-based (86.4%)
  - ↪ . . . or MPP (13.6%)
- Reasons:
  - ↪ scalable
  - ↪ cost-effective

[Source : www.top500.org, Nov 2016]



HPC Components: Software Stack

- **Remote connection to the platform** SSH
- **Identity Management / SSO**: LDAP, Kerberos, IPA...
- **Resource management**: job/batch scheduler
  - ↪ SLURM, OAR, PBS, MOAB/Torque...
- (Automatic) **Node Deployment**:
  - ↪ FAI, Kickstart, Puppet, Chef, Ansible, Kadeploy...
- (Automatic) **User Software Management**:
  - ↪ Easybuild, Environment Modules, LMod
- **Platform Monitoring**:
  - ↪ Nagios, Icinga, Ganglia, Foreman, Cacti, Alerta...

[Big]Data Management: Disk Encl.

- $\simeq$ **120 K€** / enclosure – 48-60 disks (4U)
  - ↪ incl. redundant (**i.e.** 2) RAID controllers (master/slave)

[Big]Data Management: File Systems

File System (FS)

- Logical manner to **store**, **organize**, **manipulate** & **access** data

[Big]Data Management: File Systems

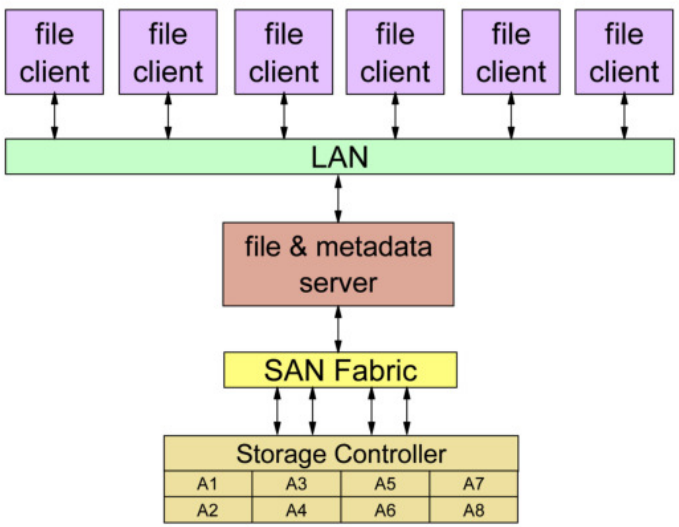
File System (FS)

- Logical manner to **store**, **organize**, **manipulate** & **access** data

- (local) **Disk FS** : `FAT32`, `NTFS`, `HFS+`, `ext{3,4}`, `{x,z,btr}fs`...
  - ↪ manage data on permanent storage devices
  - ↪ 'poor' perf. **read**: 100 → 400 MB/s | **write**: 10 → 200 MB/s

[Big]Data Management: File Systems

- **Networked FS**: `NFS, CIFS/SMB, AFP`
  - ↪ disk access from remote nodes via network access
  - ↪ poorer performance for HPC jobs especially parallel I/O
    - ✓ **read**: only 381 MB/s on a system capable of 740MB/s (16 tasks)
    - ✓ **write**: only 90MB/s on system capable of 400MB/s (4 tasks)

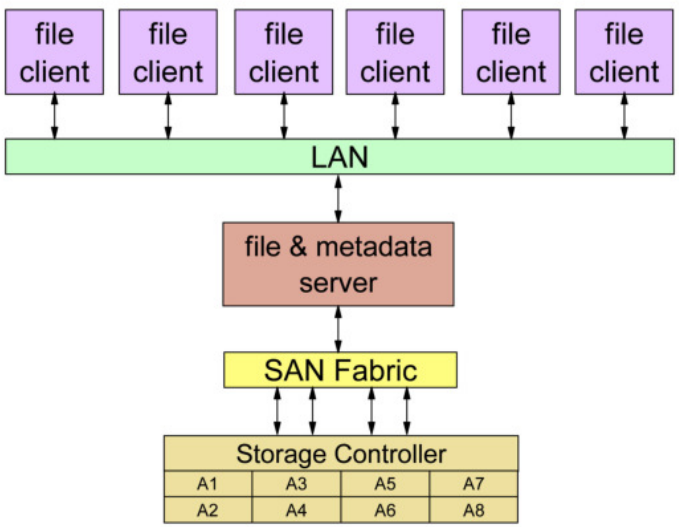


[Source : LISA'09] Ray Paden: *How to Build a Petabyte Sized Storage System*

COMMENT:
Traditionally, a single NFS/CIFS file server manages both user data and metadata operations which "gates" performance/scaling and presents a single point of failure risk. Products (e.g., CNFS) are available that provide multiple server designs to avoid this issue.

[Big]Data Management: File Systems

- **Networked FS**: `NFS, CIFS/SMB, AFP`
  - ↪ disk access from remote nodes via network access
  - ↪ poorer performance for HPC jobs especially parallel I/O
    - ✓ **read**: only 381 MB/s on a system capable of 740MB/s (16 tasks)
    - ✓ **write**: only 90MB/s on system capable of 400MB/s (4 tasks)



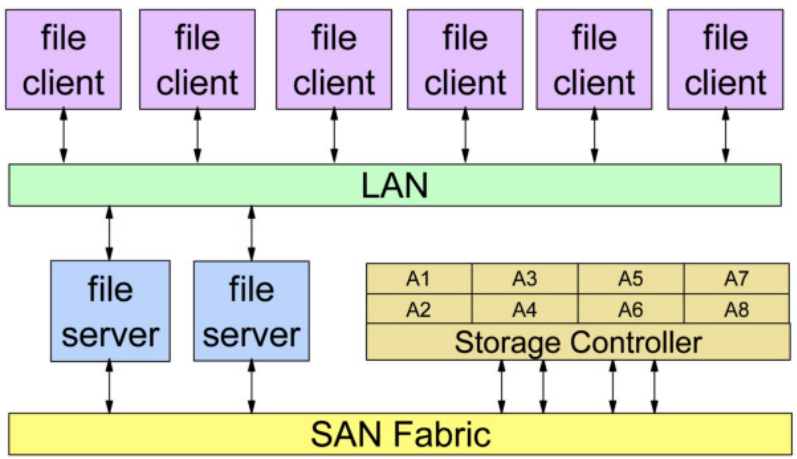
COMMENT:
Traditionally, a single NFS/CIFS file server manages both user data and metadata operations which "gates" performance/scaling and presents a single point of failure risk. Products (e.g., CNFS) are available that provide multiple server designs to avoid this issue.

[Source : LISA'09] Ray Paden: *How to Build a Petabyte Sized Storage System*

- [scale-out] **NAS**
  - ↪ aka Appliances `OneFS...`
  - ↪ Focus on CIFS, NFS
  - ↪ Integrated HW/SW
  - ↪ **Ex**: **EMC (Isilon)**, IBM (SONAS), **DDN**...

[Big]Data Management: File Systems

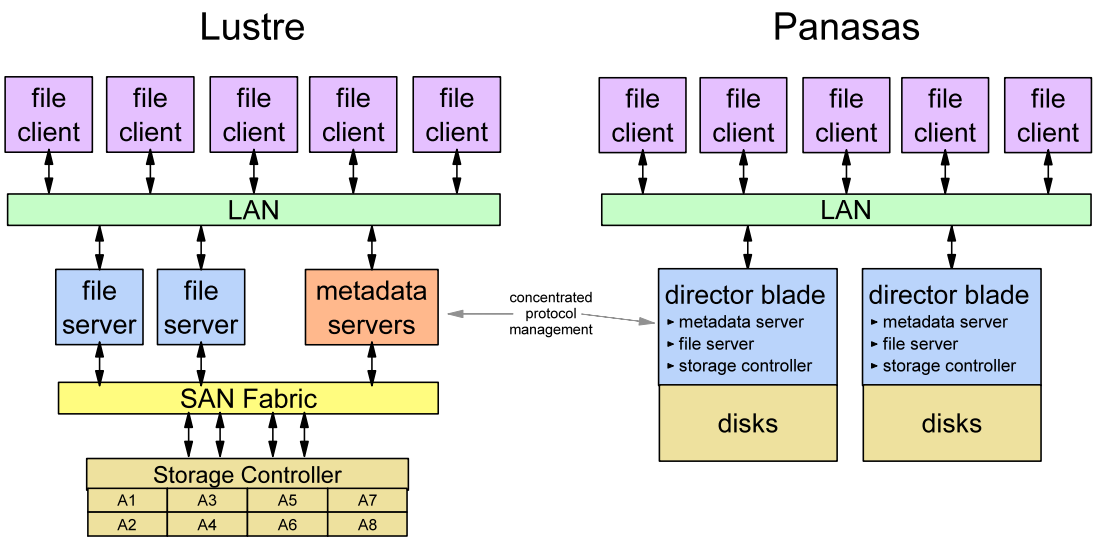
- **Basic Clustered FS** GPFS
  - ↪ File access is parallel
  - ↪ File System overhead operations is distributed and done in parallel
    - ✓ **no** metadata servers
  - ↪ File clients access file data through file servers via the LAN



File system overhead operations are *distributed* across the entire cluster and is done in parallel; it is not concentrated in any given place. There is no single server bottleneck. User data and metadata flows betweem all nodes and all disks via the file servers.

[Big]Data Management: File Systems

- **Multi-Component Clustered FS** Lustre, Panasas
  - ↪ File access is parallel
  - ↪ File System overhead operations on dedicated components
    - ✓ metadata server (Lustre) or director blades (Panasas)
  - ↪ Multi-component architecture
  - ↪ File clients access file data through file servers via the LAN



[Big]Data Management: FS Summary

- **{ Basic | Multi-Component } Clustered FS**
  ≃ Parallel/Distributed FS: `GPFS, Lustre`
  - ↪ for Input/Output (I/O)-intensive HPC systems
  - ↪ data are stripped over multiple servers for high performance
  - ↪ generally add robust failover and recovery mechanisms

Main Characteristic of Parallel/Distributed File Systems

capacity and performance increase with #servers

Name	Type	Read* [GB/s]	Write* [GB/s]
`ext4`	Disk FS	0.426	0.212
`nfs`	Networked FS	0.381	0.090
`gpfs (iris)`	Parallel/Distributed FS	**10.14**	**8,41**
`gpfs (gaia)`	Parallel/Distributed FS	7.74	6.524
`lustre`	Parallel/Distributed FS	4.5	2.956

* maximum **random** read/write, per IOZone or IOR measures, using 15 concurrent nodes for networked FS. Measured performed on the UL HPC facility in Jan. 2015

HPC Components: Data Center

Definition (Data Center)

- Facility to house computer systems and associated components
  - ↪ Basic storage component: **rack** (height: 42 RU)

HPC Components: Data Center

Definition (Data Center)

- Facility to house computer systems and associated components
  - ↪ Basic storage component: **rack** (height: 42 RU)

Challenges: Power (UPS, battery), Cooling, Fire protection, Security

- Power/Heat dissipation per rack:
  - ↪ HPC **computing** racks: **30-120 kW**
  - ↪ **Storage** racks: **15 kW**
  - ↪ **Interconnect** racks: **5 kW**
- Various **Cooling** Technology
  - ↪ Airflow
  - ↪ Direct-Liquid Cooling, Immersion...

Power Usage Effectiveness

$$PUE = \frac{Total\ facility\ power}{IT\ equipment\ power}$$

HPC Components: Summary

Running an HPC Facility involves...

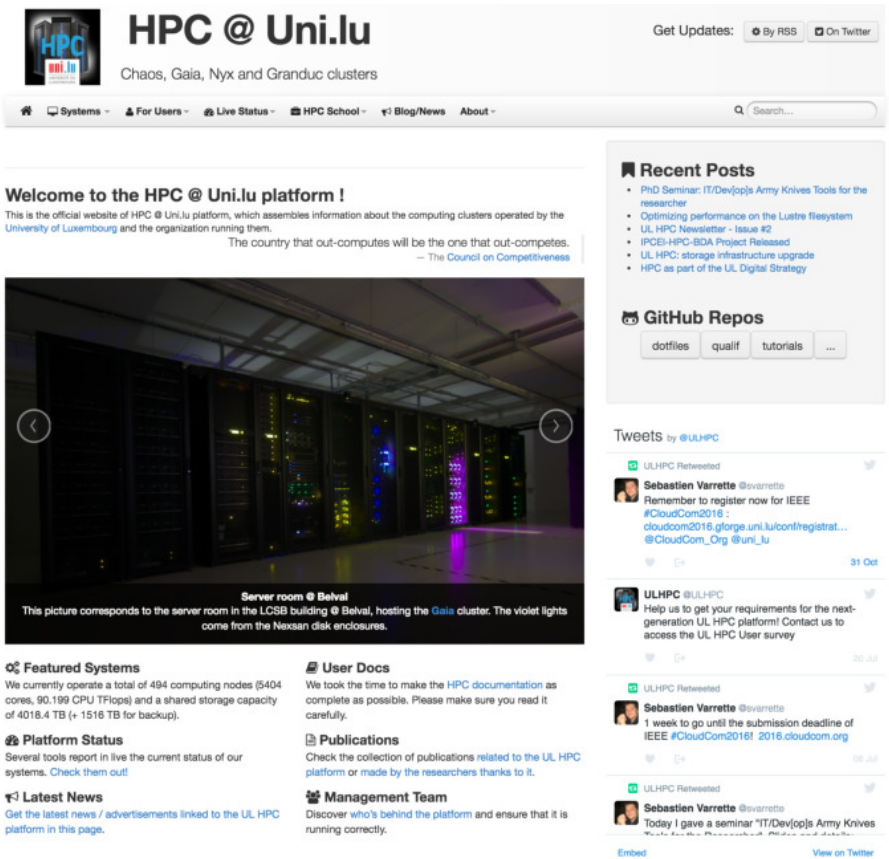
- A **data center** / server room carefully designed
- Many **computing** elements: CPU, GPGPU, Accelerators
- **Fast interconnect** elements
  - ↪ high *bandwidth* and low *latency*
- [Big]-Data **storage** elements: HDD/SDD, disk enclosure,
  - ↪ disks are virtually aggregated by RAID/LUNs/FS
  - ↪ parallel and distributed FS
- A flexible software stack
- Automated management everywhere

Above all: **expert** system administrators !

Summary

Summary

High Performance Computing @ UL



http://hpc.uni.lu

Key numbers

- 416 users
- 110 servers
- 594 nodes
  - ↪ 8228 cores
  - ↪ **198.172 TFlops**
  - ↪ 50 accelerators
    - ✓ + **76 TFlops**
- **6856.4 TB**
- 5 sysadmins
- 2 sites
  - ↪ Kirchberg
  - ↪ Belval

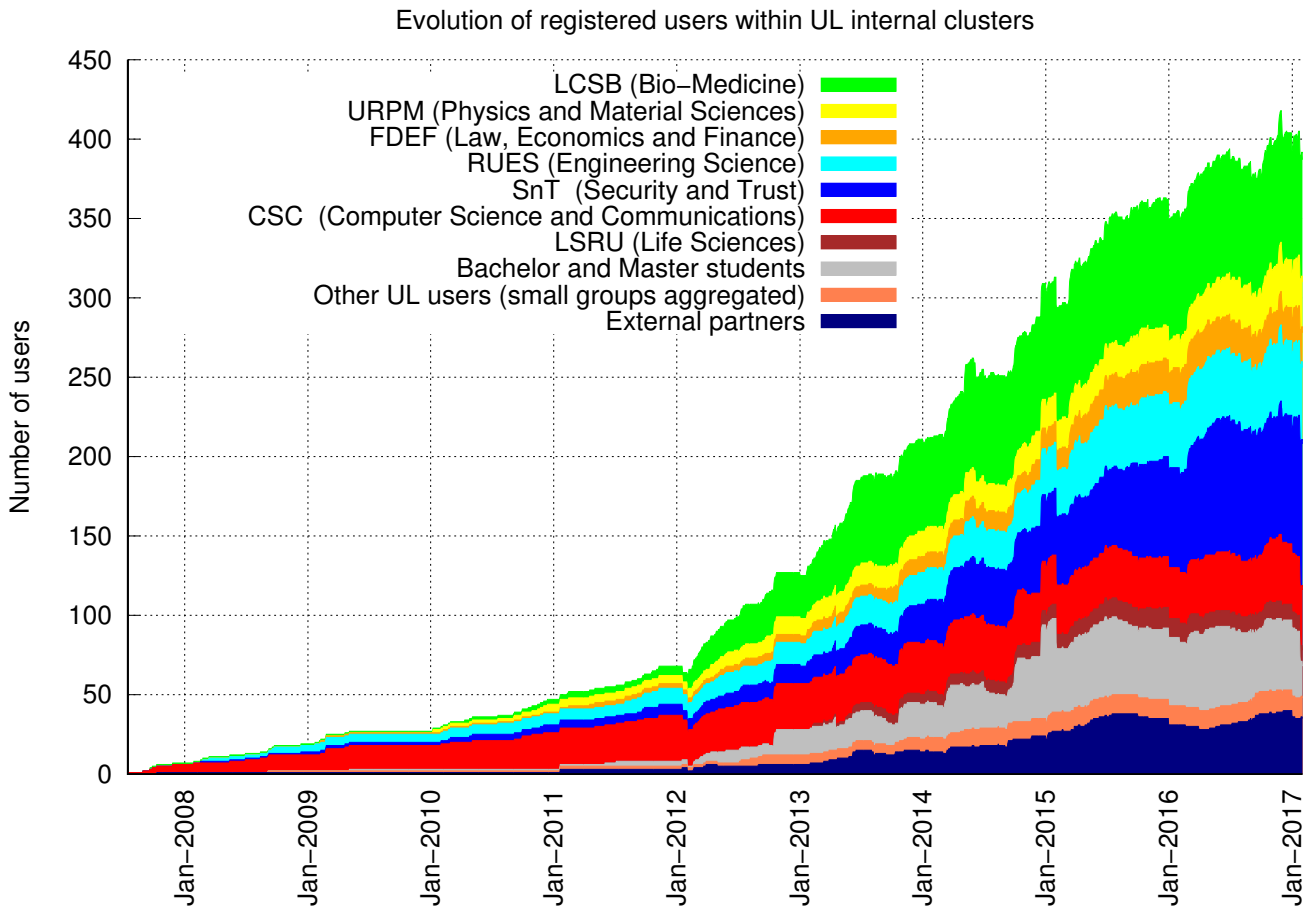
High Performance Computing @ UL

- **Enables & accelerates** scientific discovery and innovation
- **Largest facility** in Luxembourg (after GoodYear R&D Center)

			(CPU)	TFlops	TB (Shared)
Country	**Institute**	**#Nodes**	**#Cores**	**R_{peak}**	**Storage**
Luxembourg	**UL HPC** (Uni.lu)	**594**	**8228**	**198.172**	**6856.4**
	LIST	58	800	6.21	144
France	LORIA (G5K), Nancy	320	2520	26.98	82
	ROMEO, Reims	174	3136	49.26	245
Belgium	NIC4, University of Liège	128	2048	32.00	20
	Université Catholique de Louvain	112	1344	13.28	120
	UGent / VSC, Gent	440	8768	275.30	1122
Germany	bwGrid, Heidelberg	140	1120	12.38	32
	bwForCluster, Ulm	444	7104	266.40	400
	bwHPC MLS&WISO, Mannheim	604	9728	371.60	420

UL HPC User Base

- **416 Active** HPC Users



UL HPC Beneficiaries

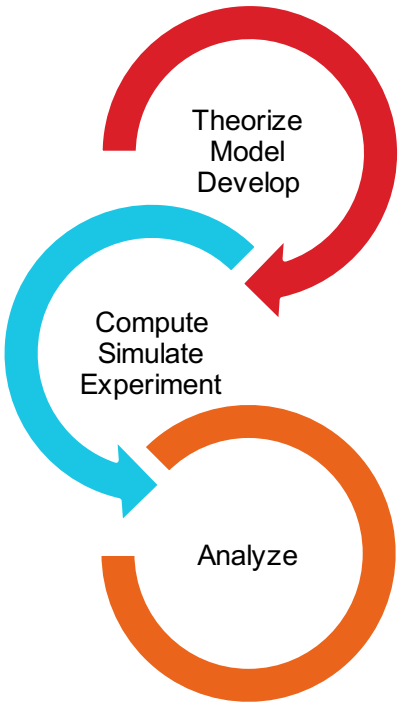
23 computational domains accelerated on UL HPC

- for the UL Faculties, Research Units and Interdisciplinary Centres
  - ↪ incl. LCSB, SnT... and now C2DH thematics
  - ↪ UL **strategic research priorities**
    - ✓ computational sciences, finance (fintech)
    - ✓ systems biomedicine, security, reliability and trust

- UL HPC feat. special systems targeting specific workloads:
  - ↪ **Machine Learning & AI**: GPU accelerators
    - ✓ 10 Tesla K40 + 16 Tesla K80 + 24 Tesla M20*: **76 GPU Tflops**
  - ↪ **BigData analytics & data driven science**: large memory systems
    - ✓ Large SMP systems with 1, 2, 3 & 4 TB RAM
  - ↪ **Scale-out workloads**: energy efficient systems
    - ✓ 90 HP Moonshot servers + 96 viridis ARM-based systems

Accelerating UL Research

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



- **133 software packages** available for researchers
  - ↪ **General purpose**, statistics, optimization:
    - ✓ Matlab, Mathematica, R, Stata, CPLEX, Gurobi Optimizer...
  - ↪ **Bioinformatics**
    - ✓ BioPython, STAR, TopHat, Bowtie, mpiHMMER...
  - ↪ **Computer aided engineering**:
    - ✓ ABAQUS, OpenFOAM...
  - ↪ **Molecular dynamics**:
    - ✓ ABINIT, QuantumESPRESSO, GROMACS...
  - ↪ **Visualisation**: ParaView, VisIt, XCS portal
  - ↪ Compilers, libraries, performance
  - ↪ [Parallel] debugging tools aiding development

UL HPC Team

Prof. Pascal Bouvry
Director of DS-CSCE, Leader of PCO Group
Senior advisor for the president as regards the HPC strategy

Sébastien Varrette, PhD
CDI, Research Scientist (CSC, FSTC)

Valentin Plugaru, MSc.
CDI, Research Collaborator (CSC, FSTC)

Sarah Diehl, MSc.
CDD, Research Associate (LCSB)

Hyacinthe Cartiaux
CDI, Support (SIU)

Clement Parisot
CDI, Support (FSTC)

UNIVERSITÉ DU LUXEMBOURG

Summary

Sites / Data centers

Kirchberg

CS.43, AS. 28

Belval

Biotech I, CDC/MSA

2 sites, $\geq$ 4 server rooms

Sites / Data centers

Kirchberg

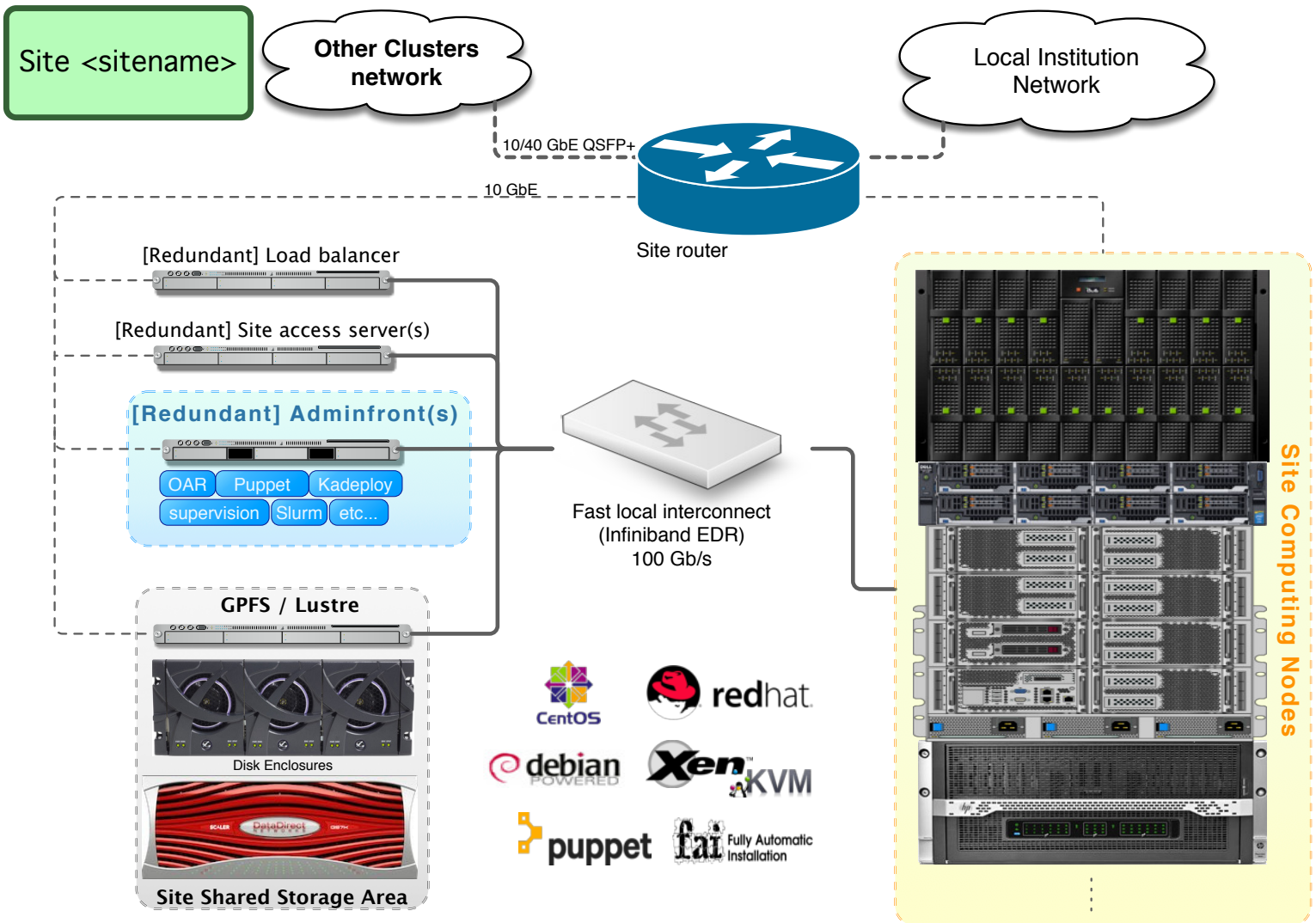
CS.43, AS. 28

Belval

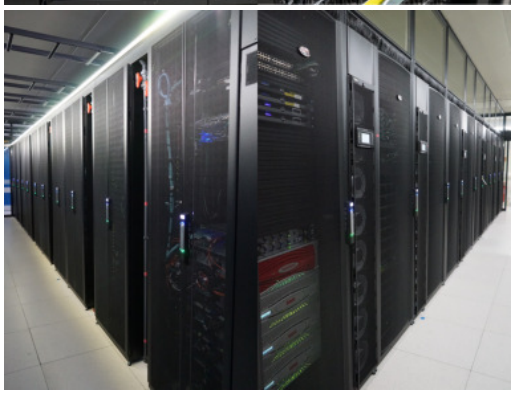
Biotech I, CDC/MSA

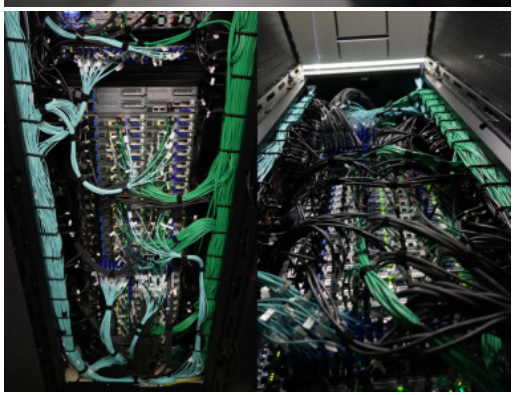
2 sites, $\geq$ 4 server rooms

UL HPC: General cluster organization



UL HPC Computing capacity





5 clusters

198.172 TFlops

594 nodes

8228 cores

34512GPU cores

INFINIBAND TRADE ASSOCIATION intel

UL HPC Computing Clusters

Cluster	Location	**#N**	**#C**	**Rpeak**	GPU Rpeak
`iris`	CDC S-01	100	2800	107.52	0
`gaia`	BT1	273	3440	69.296	76
`chaos`	Kirchberg	81	1120	14.495	0
`g5k`	Kirchberg	38	368	4.48	0
`nyx` (experimental)	BT1	102	500	2.381	0
	TOTAL:	**594**	**8228**	**198.172**	**+ 76 TFlops**

UL HPC – Detailed Computing Nodes

	Date	Vendor	Proc. Description	#N	#C	R_{peak}
iris	2017	Dell	Intel Xeon E5-2680 v4@2.4GHz 2 × 14C,128GB	100	2800	107,52 TFlops
			`iris` TOTAL:	**100**	**2800**	**107.52 TFlops**
gaia	2011	Bull	Intel Xeon L5640@2.26GHz 2 × 6C,48GB	72	864	7.811 TFlops
	2012	Dell	Intel Xeon E5-4640@2.4GHz 4 × 8C, 1TB	1	32	0.614 TFlops
	2012	Bull	Intel Xeon E7-4850@2GHz 16 × 10C,1TB	1	160	1.280 TFLops
	2013	Dell	Intel Xeon E5-2660@2.2GHz 2 × 8C,64GB	5	80	1.408 TFlops
	2013	Bull	Intel Xeon X5670@2.93GHz 2 × 6C,48GB	40	480	5.626 TFlops
	2013	Bull	Intel Xeon X5675@3.07GHz 2 × 6C,48GB	32	384	4.746 TFlops
	2014	Delta	Intel Xeon E7-8880@2.5 GHz 8 × 15C,1TB	1	120	2.4 TFlops
	2014	SGi	Intel Xeon E5-4650@2.4 GHz 16 × 10C,4TB	1	160	3.072 TFlops
	2015	Dell	Intel Xeon E5-2680@2.5 GHz 2 × 12C,128GB	28	672	26.88 TFlops
	2015	HP	Intel E3-1284Lv3, 1.8GHz 1 × 4C,32GB	90	360	10.368 TFlops
	2016	Dell	Intel Xeon E7-8867@2.5 GHz 4 × 16C,2TB	2	128	5.12 TFlops
			`gaia` TOTAL:	**273**	**3440**	**69.296 TFlops**
chaos	2010	HP	Intel Xeon L5640@2.26GHz 2 × 6C,24GB	32	384	3.472 TFlops
	2011	Dell	Intel Xeon L5640@2.26GHz 2 × 6C,24GB	16	192	1.736 TFlops
	2012	Dell	Intel Xeon X7560@2,26GHz 4 × 6C, 1TB	1	32	0.289 TFlops
	2012	Dell	Intel Xeon E5-2660@2.2GHz 2 × 8C,32GB	16	256	4.506 TFlops
	2012	HP	Intel Xeon E5-2660@2.2GHz 2 × 8C,32GB	16	256	4.506 TFlops
			`chaos` TOTAL:	**81**	**1120**	**14.495 TFlops**
g5k	2008	Dell	Intel Xeon L5335@2GHz 2 × 4C,16GB	22	176	1.408 TFlops
	2012	Dell	Intel Xeon E5-2630L@2GHz 2 × 6C,24GB	16	192	3.072 TFlops
			`granduc/petitprince` TOTAL:	**38**	**368**	**4.48 TFlops**
Testing cluster:						
nyx, viridis, pyro...	2012	Dell	Intel Xeon E5-2420@1.9GHz 1 × 6C,32GB	2	12	0.091 TFlops
	2013	Viridis	ARM A9 Cortex@1.1GHz 1 × 4C,4GB	96	384	0.422 TFlops
	2015	Dell	Intel Xeon E5-2630Lv2@2.4GHz 2 × 6C,32GB	2	24	0.460 TFlops
	2015	Dell	Intel Xeon E5-2660v2@2.2GHz 2 × 10C,32GB	4	80	1.408 TFlops
			`nyx/viridis` TOTAL:	**102**	**500**	**2.381 TFlops**

UL HPC Storage capacity

4 distributed/parallel FS

1558 disks

6856.4 TB

(incl. 1020TB for Backup)

UL HPC Shared Storage Capacities

Cluster	GPFS	Lustre	Other (NFS...)	Backup	TOTAL
`iris`	1440	0	6	600	**2046** TB
`gaia`	960	480	0	240	**1680** TB
`chaos`	0	0	180	180	**360** TB
`g5k`	0	0	32.4	0	**32.4** TB
`nyx` (experimental)	0	0	242	0	**242** TB
TOTAL:	**2400**	**480**	**2956.4**	**1020**	**6856.4 TB**

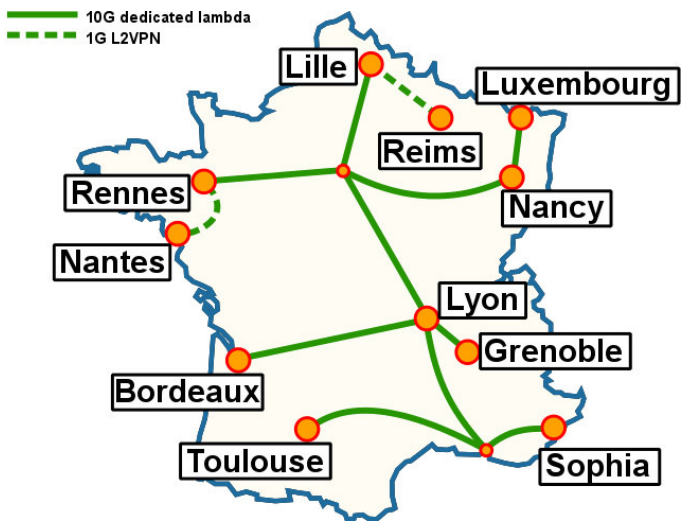
UL HPC Software Stack

- **Operating System**: **Linux** CentOS 7 (`iris`), Debian 7 (others)
- **Remote connection to the platform**: SSH
- **User SSO**: IPA, OpenLDAP
- **Resource management**: job/batch scheduler: **Slurm**(`iris`), OAR
- **(Automatic) Computing Node Deployment**:
  - ↪ FAI (Fully Automatic Installation)
  - ↪ Bright Cluster Manager (`iris`)
  - ↪ Puppet
  - ↪ Kadeploy
- **Platform Monitoring**:
  - ↪ OAR Monika/Drawgantt, Ganglia, Allinea Perf Report, Slurm
  - ↪ Icinga, NetXMS, PuppetBoard etc.
- **Commercial Softwares**:
  - ↪ Intel Cluster Studio XE, TotalView, Allinea DDT, Stata etc.

The case of Grid'5000

http://www.grid5000.fr

- Large scale nation wide infrastructure
  - ↪ for large scale parallel and distributed computing research.

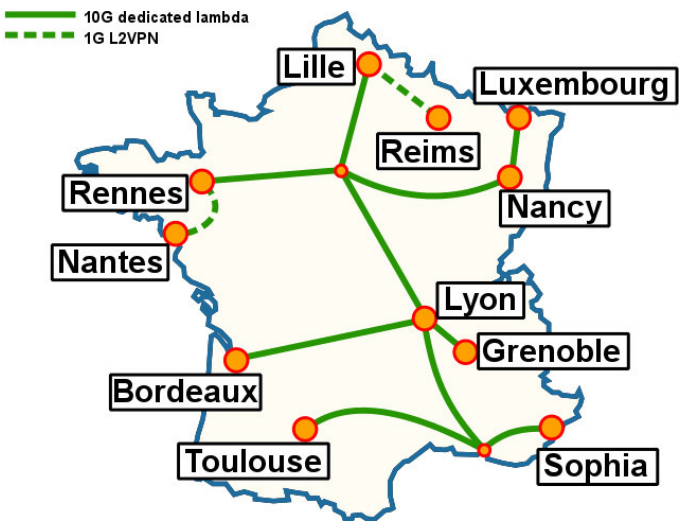


- 10 sites in France
  - ↪ **Abroad**: Luxembourg, Porto Allegre
  - ↪ Total: **7782** cores over **26** clusters
- 1-10GbE / Myrinet / Infiniband
  - ↪ **10Gb/s dedicated** between all sites
- Unique software stack
  - ↪ **kadeploy, kavlan, storage5k**

The case of Grid'5000

http://www.grid5000.fr

- Large scale nation wide infrastructure
  - ↪ for large scale parallel and distributed computing research.



- 10 sites in France
  - ↪ **Abroad**: Luxembourg, Porto Allegre
  - ↪ Total: **7782** cores over **26** clusters
- 1-10GbE / Myrinet / Infiniband
  - ↪ **10Gb/s dedicated** between all sites
- Unique software stack
  - ↪ **kadeploy, kavlan, storage5k**

- **Out of scope for this talk**
  - ↪ General information: https://hpc.uni.lu/g5k
  - ↪ Grid'5000 website and documentation: https://www.grid5000.fr

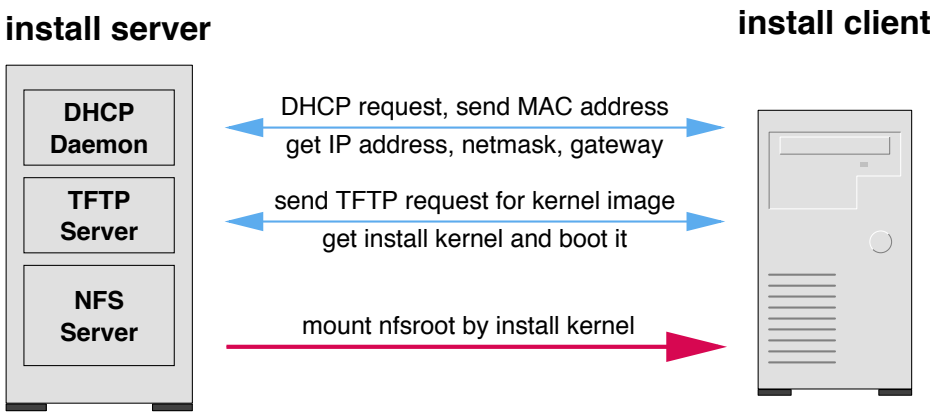
Summary

Computing nodes Management

Node deployment by FAI

http://fai-project.org/

- Boot via network card (PXE)
  - ↪ ensure a running diskless Linux OS

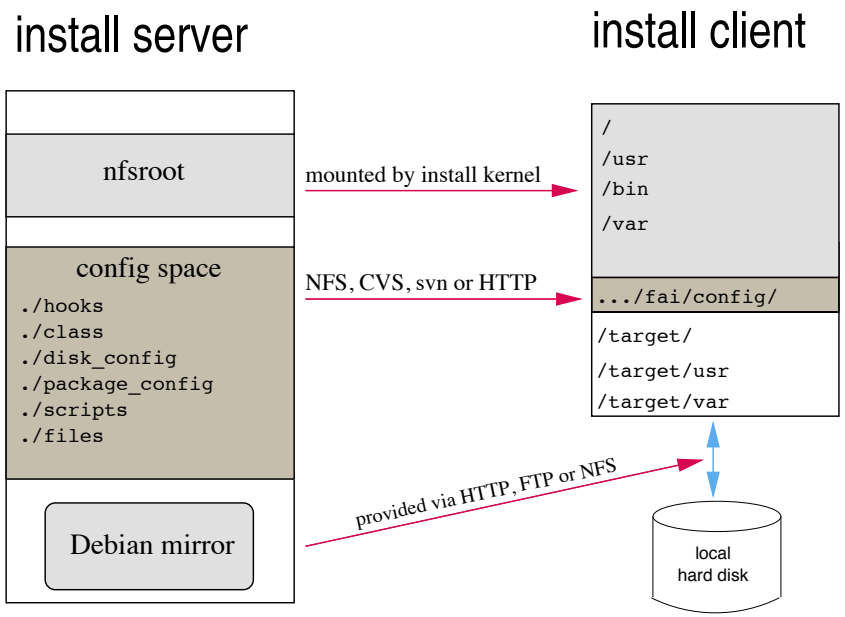


Computing nodes Management

Node deployment by FAI

http://fai-project.org/

- Boot via network card (PXE)
  - ↪ ensure a running diskless Linux OS
- Get configuration data (NFS)



Computing nodes Management

Node deployment by FAI

http://fai-project.org/

- Boot via network card (PXE)
  - ↪ ensure a running diskless Linux OS
- Get configuration data (NFS)
- Run the installation
  - ↪ partition local hard disks and create filesystems
  - ↪ install software using apt-get command
  - ↪ configure OS and additional software
  - ↪ save log files to install server, then reboot new system

Computing nodes Management

Node deployment by FAI

http://fai-project.org/

- Boot via network card (PXE)
  - ↪ ensure a running diskless Linux OS
- Get configuration data (NFS)
- Run the installation
  - ↪ partition local hard disks and create filesystems
  - ↪ install software using apt-get command
  - ↪ configure OS and additional software
  - ↪ save log files to install server, then reboot new system

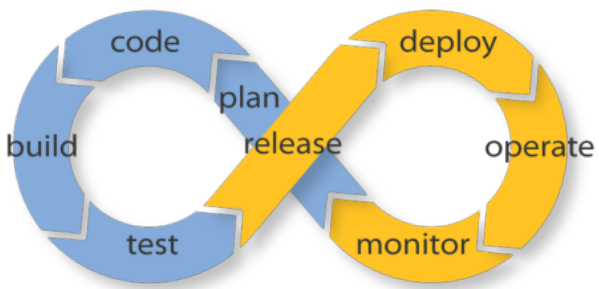
Average reinstallation time: $\simeq$ 500s

IT Serv[er|ice] Management: Puppet

Server/Service configuration by Puppet

http://puppetlabs.com

- **IT Automation** for configuration management
  - ↪ idempotent
  - ↪ agent/master OR stand-alone architecture
  - ↪ cross-platform through Puppet's Resource Abstraction Layer (RAL)
  - ↪ Git-based workflow
  - ↪ PKI-based security (X.509)
- DevOps tool of choice for configuration management
  - ↪ Declarative Domain Specific Language (DSL)



Endless Possibilities: DevOps can create an infinite loop of release and feedback for all your code and deployment targets.

IT Serv[er|ice] Management: Puppet

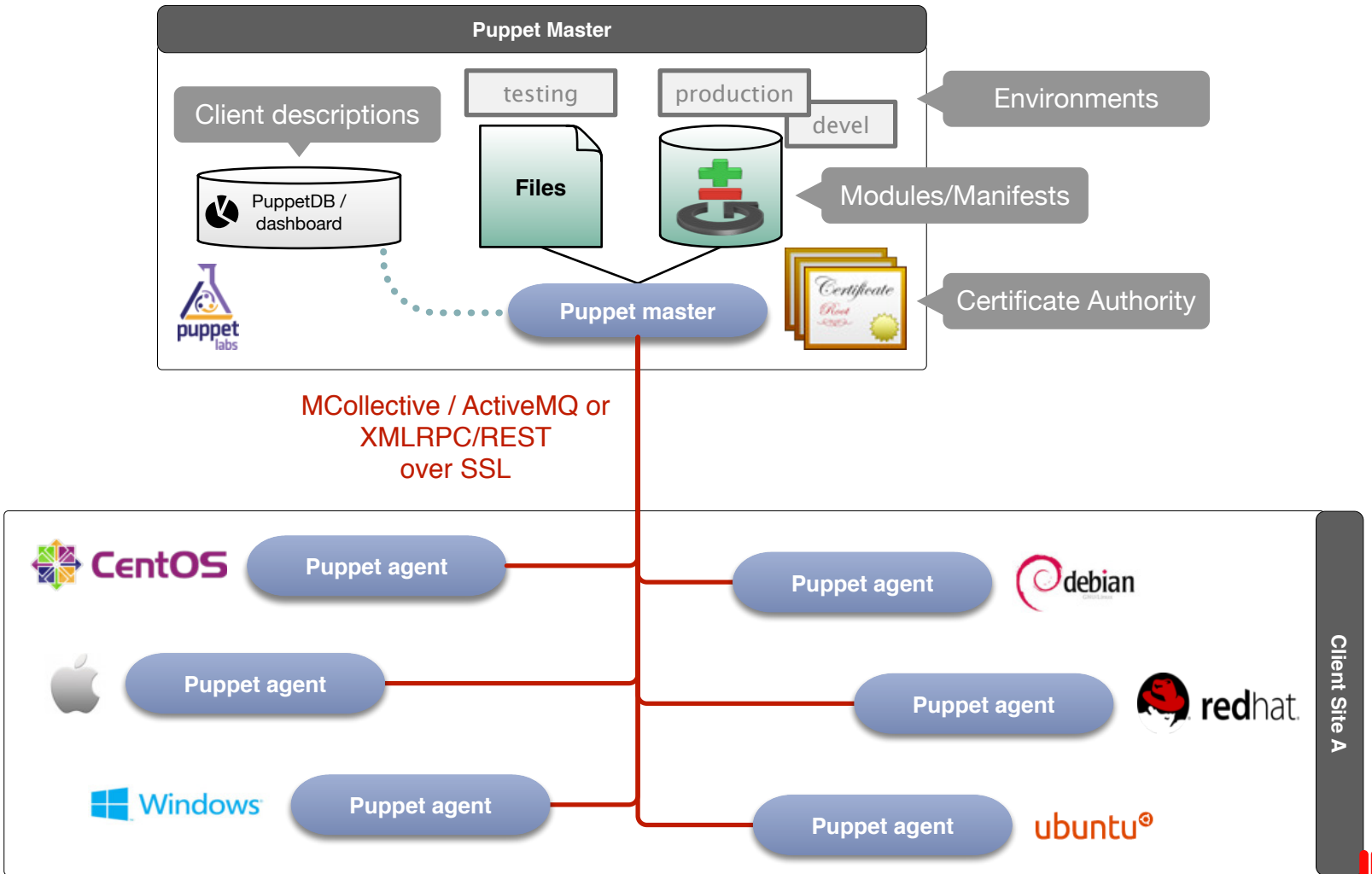
Server/Service configuration by Puppet

http://puppetlabs.com

- **IT Automation** for configuration management
  - ↪ idempotent
  - ↪ agent/master OR stand-alone architecture
  - ↪ cross-platform through Puppet's Resource Abstraction Layer (RAL)
  - ↪ Git-based workflow
  - ↪ PKI-based security (X.509)
- DevOps tool of choice for configuration management
  - ↪ Declarative Domain Specific Language (DSL)

Average server installation/configuration time: $\simeq$ 3-6 min

General Puppet Infrastructure



Software/Modules Management

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

- Based on Environment Modules / LMod
  - ↪ convenient way to dynamically change the users' environment `$PATH`
  - ↪ permits to easily load software through `module` command
- Currently on UL HPC:
  - ↪ **133 software packages**, in *multiple* versions, within **18 categories**
  - ↪ reworked software set for `iris` cluster and soon deployed everywhere
    - ✓ RESIF v2.0, allowing [real] semantic versionning of released builds
  - ↪ hierarchical organization **Ex**: `toolchain/{foss,intel}`

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

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

Software/Modules Management

http://hpcugent.github.io/easybuild/

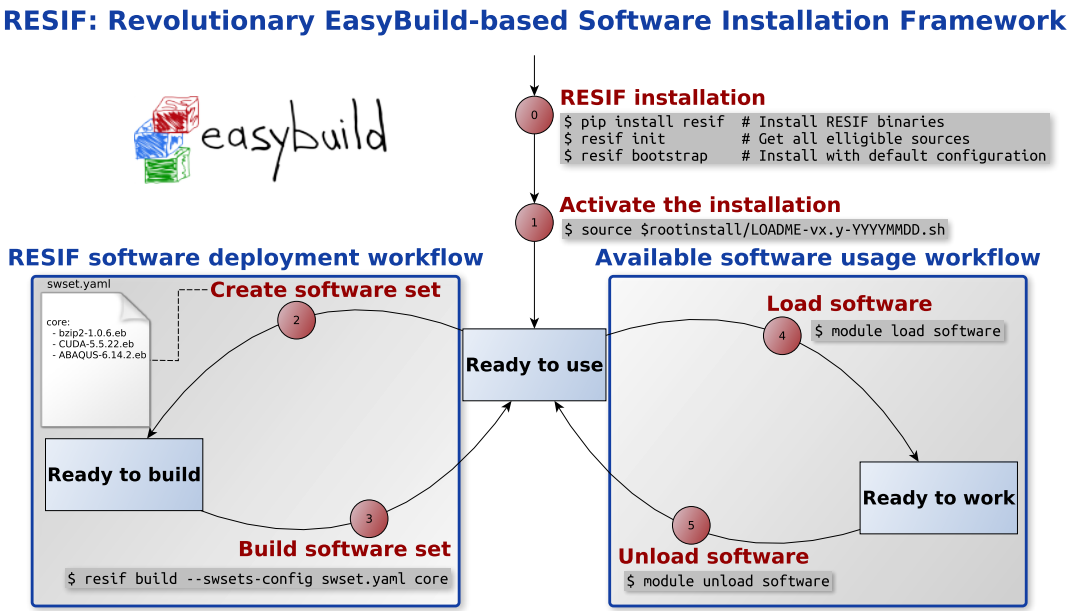
- Easybuild: open-source framework to (automatically) build scientific software
- **Why?**: *"Could you please install this software on the cluster?"*
  - ↪ Scientific software are often **painful** to build
    - ✓ non-standard build tools / incomplete build procedure
    - ✓ hardcoded parameters and/or poor/outdated documentation
  - ↪ EasyBuild helps to facilitate this task
    - ✓ consistent software build and installation framework
    - ✓ automatically generates LMod modulefiles

```
$> module use /path/to/easybuild
$> module load tools/EasyBuild  toolchain/intel
$> eb -S HPL    # Search for recipes for HPL software
$> eb HPL-2.2-intel-2017a.eb # Install HPC 2.2 w. Intel toolchain
```

Software/Modules Management

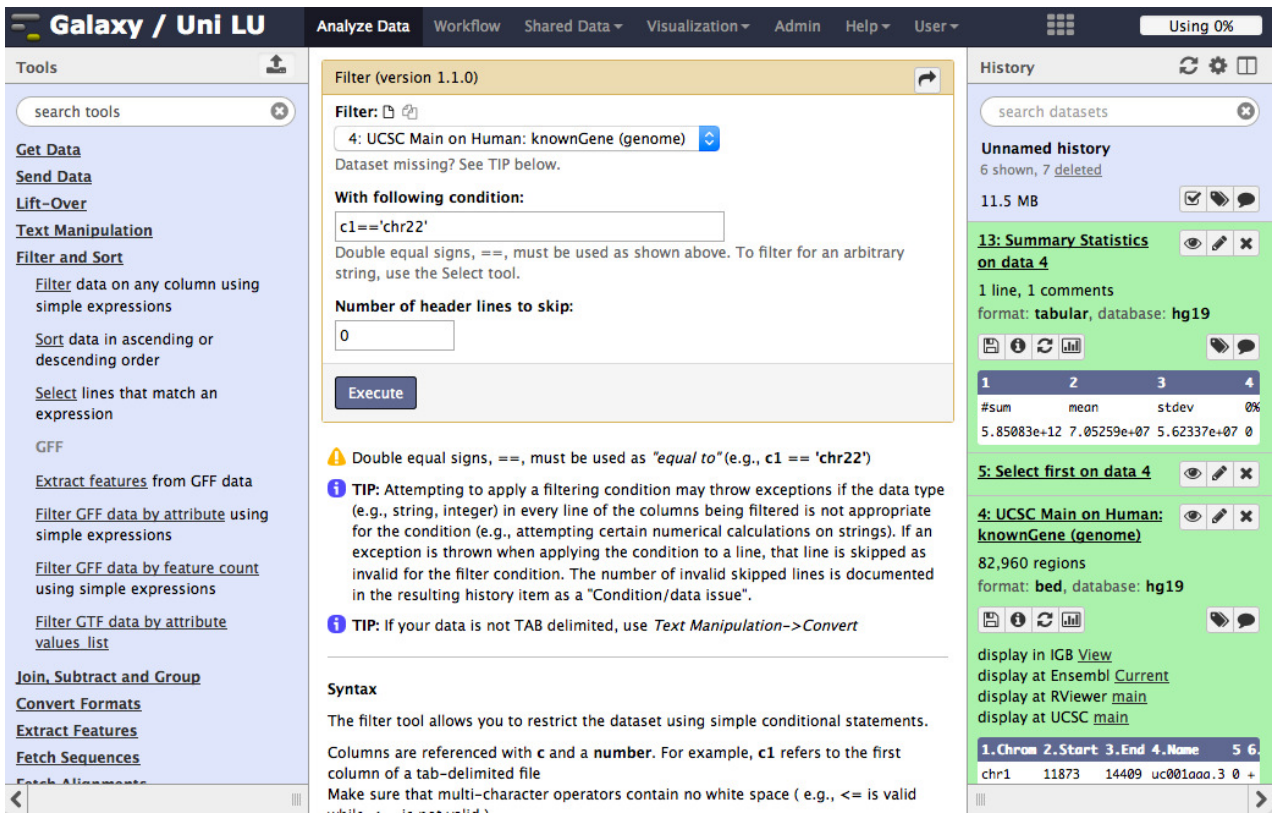
http://resif.readthedocs.io/en/latest/

- **RESIF**: Revolutionary EasyBuild-based Software Installation Framework
  - ↪ Automatic Management of **software sets**
  - ↪ Fully automates software builds and supports all available toolchains
  - ↪ Clean (hierarchical) modules layout to facilitate its usage
  - ↪ "Easy to use" yet **pending workflow rework**



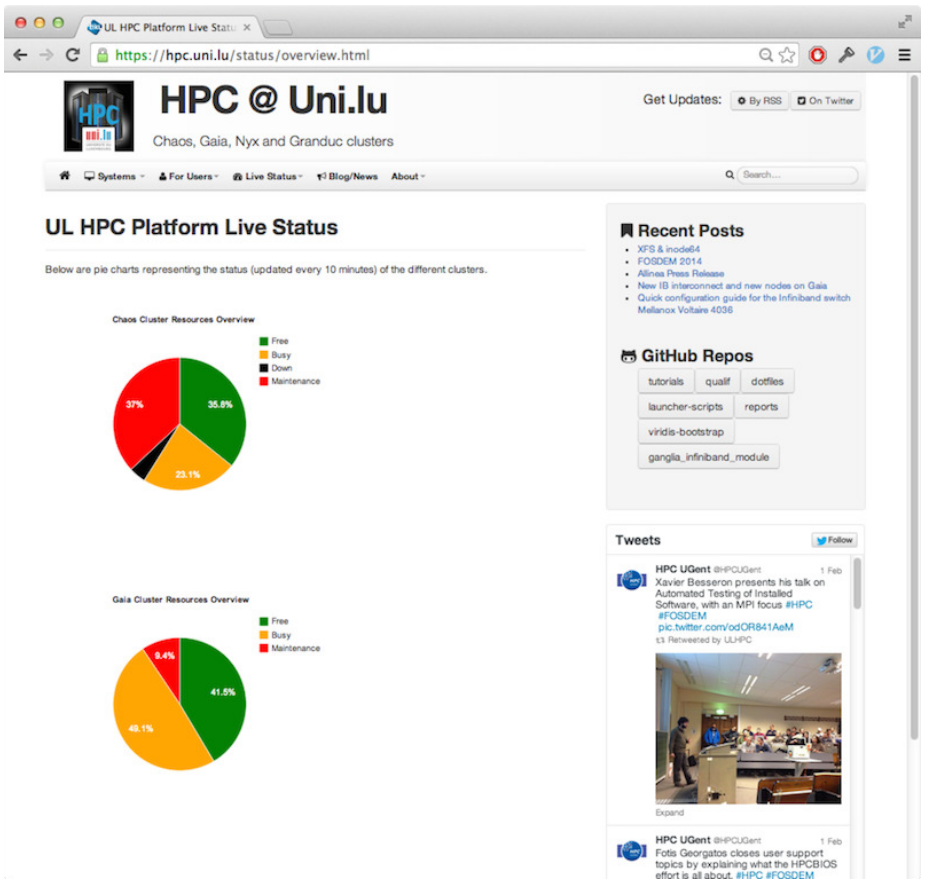
BIO Workflow Management

- Galaxy Portal `http://galaxy-server.uni.lu`
  - ↪ web-based platform for data intensive biomedical research



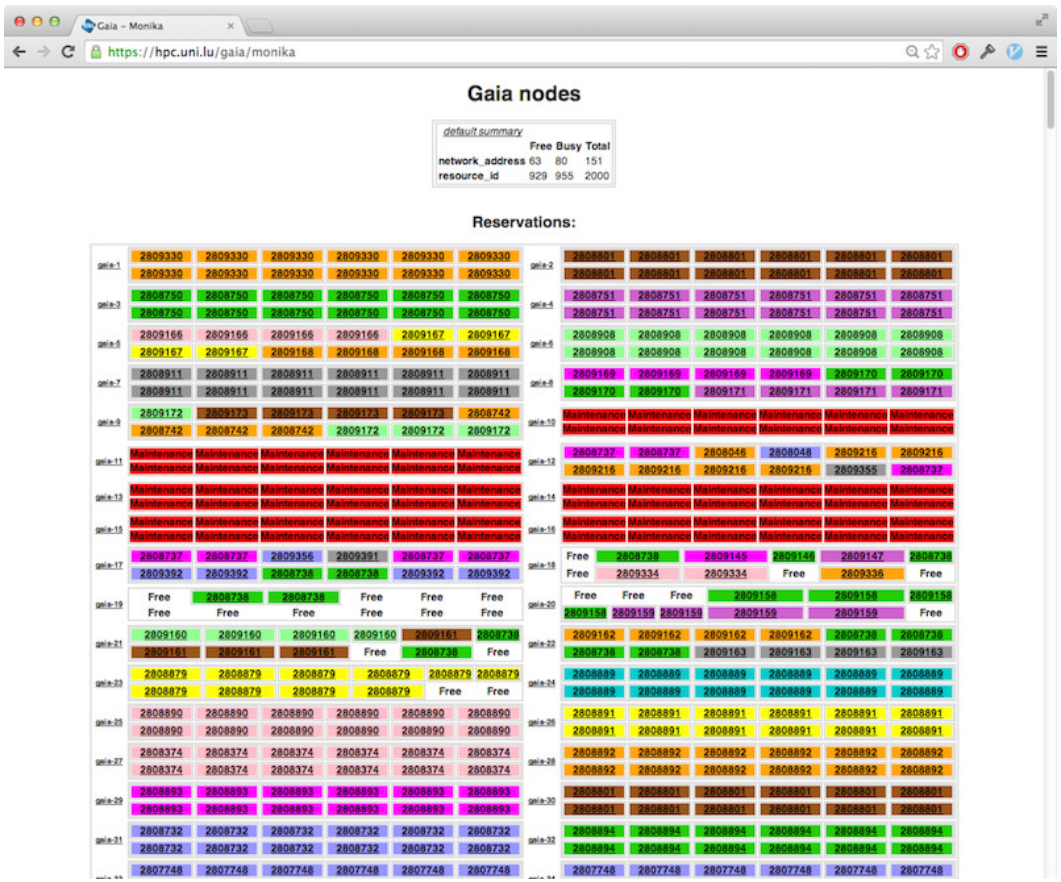
Platform Monitoring

- **General Live Status** `http://hpc.uni.lu/status/overview.html`



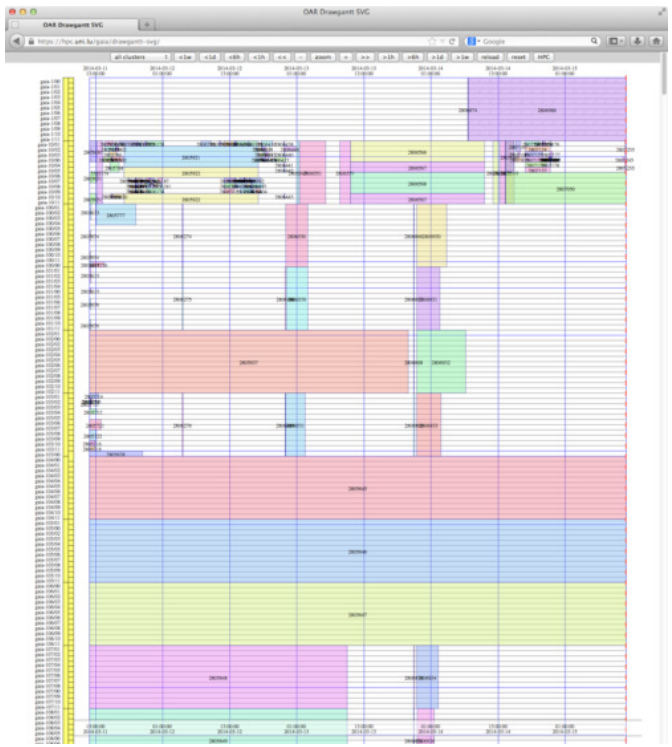
Platform Monitoring

- **Monika** `http://hpc.uni.lu/{iris,gaia,chaos,g5k}/monika`



Platform Monitoring

- **Drawgantt** `http://hpc.uni.lu/{iris,gaia,chaos,g5k}/drawgantt`

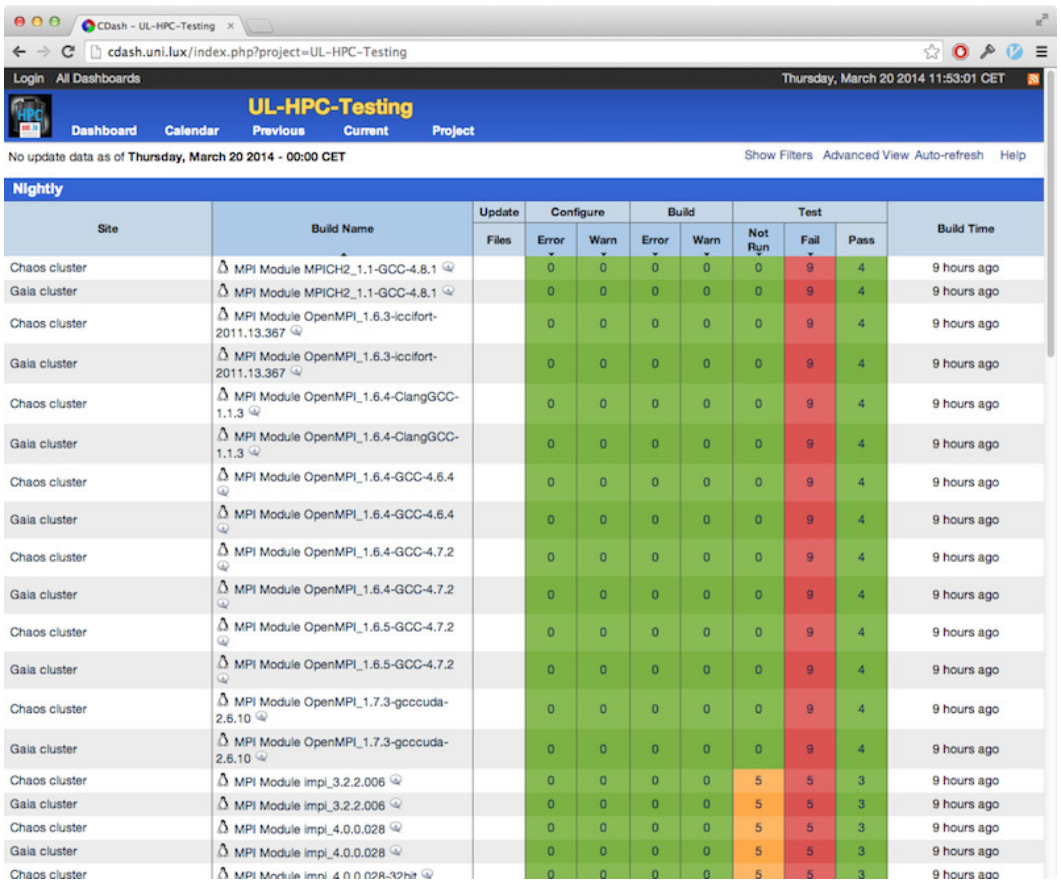


Platform Monitoring

- **Ganglia** `http://hpc.uni.lu/{iris,gaia,chaos,g5k}/ganglia`

Platform Monitoring

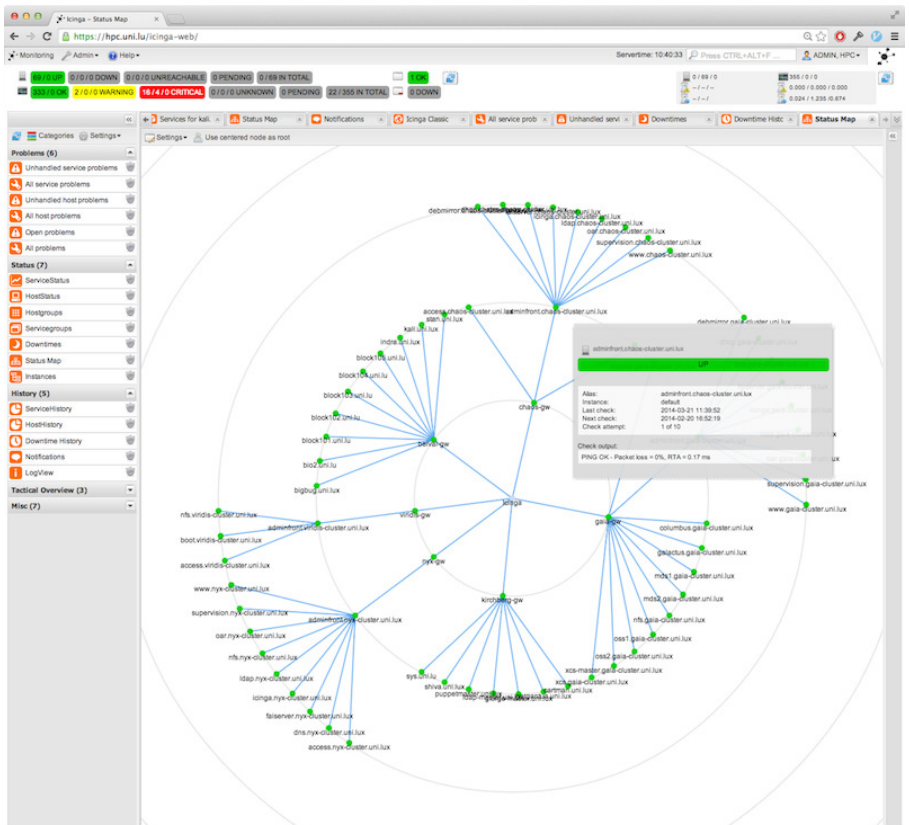
- **CDash** http://cdash.uni.lu/



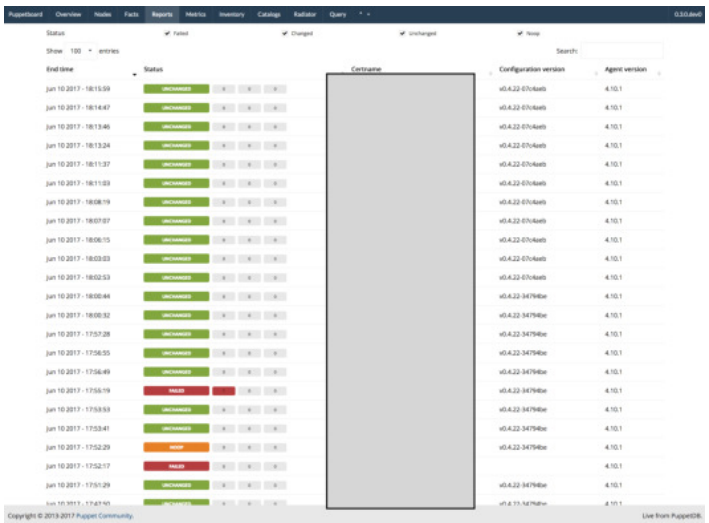
Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
Chaos cluster	MPI Module MPICH2_1.1-GCC-4.8.1		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module MPICH2_1.1-GCC-4.8.1		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.3-iccifort-2011.13.367		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.6.3-iccifort-2011.13.367		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.4-ClangGCC-1.1.3		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.6.4-ClangGCC-1.1.3		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.4-GCC-4.6.4		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.6.4-GCC-4.6.4		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.4-GCC-4.7.2		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.6.4-GCC-4.7.2		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.5-GCC-4.7.2		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.6.5-GCC-4.7.2		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.7.3-gcccuda-2.6.10		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module OpenMPI_1.7.3-gcccuda-2.6.10		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module impi_3.2.2.006		0	0	0	0	5	5	3	9 hours ago
Gaia cluster	MPI Module impi_3.2.2.006		0	0	0	0	5	5	3	9 hours ago
Chaos cluster	MPI Module impi_4.0.0.028		0	0	0	0	5	5	3	9 hours ago
Gaia cluster	MPI Module impi_4.0.0.028		0	0	0	0	5	5	3	9 hours ago
Chaos cluster	MPI Module impi_4.0.0.028-32bit		0	0	0	0	5	5	3	9 hours ago

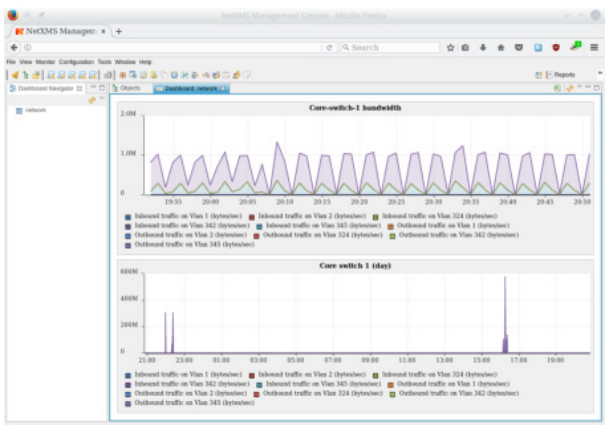
Platform Monitoring

- **Internal Monitoring**



Icinga / Puppet / NetXMS (networking)

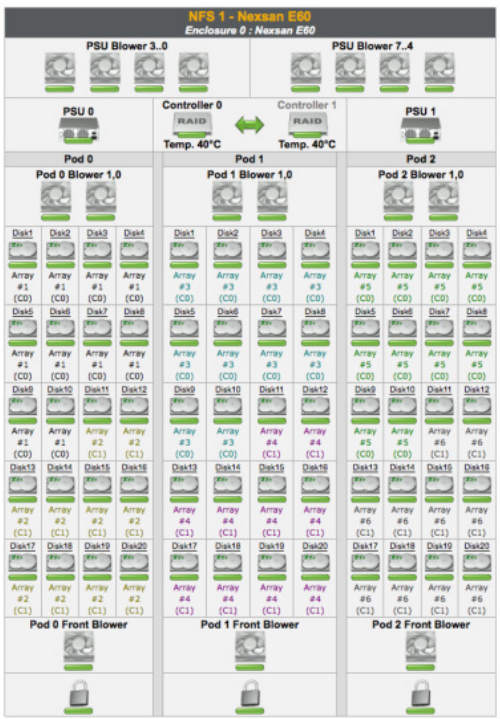


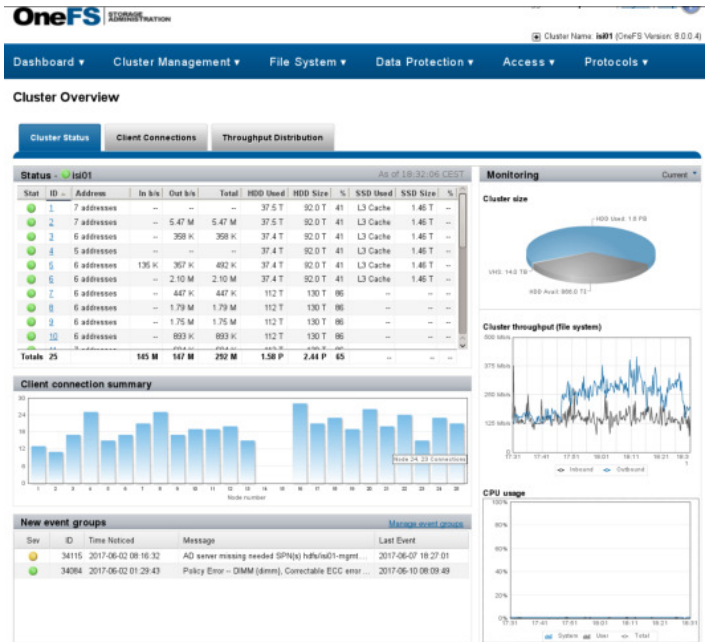


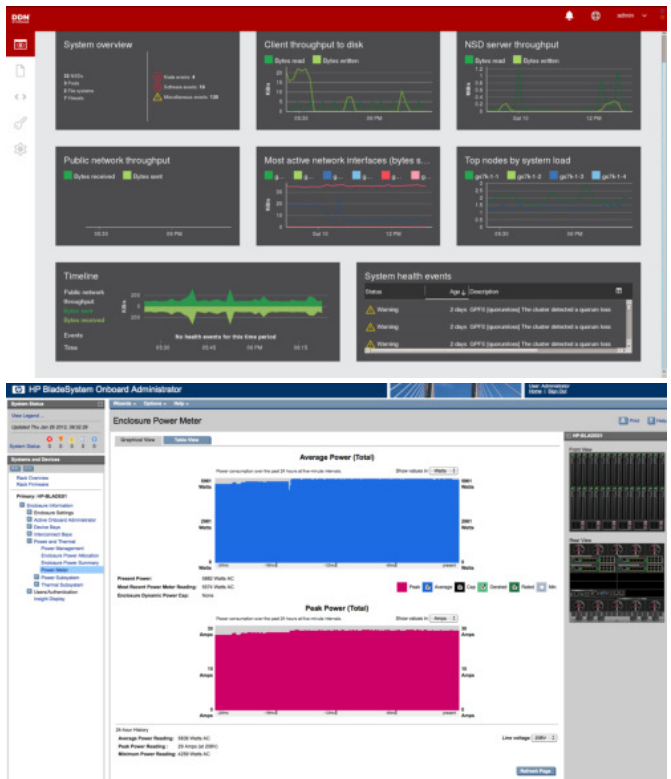
Platform Monitoring

- **Internal Monitoring**

[Disk] Enclosure status

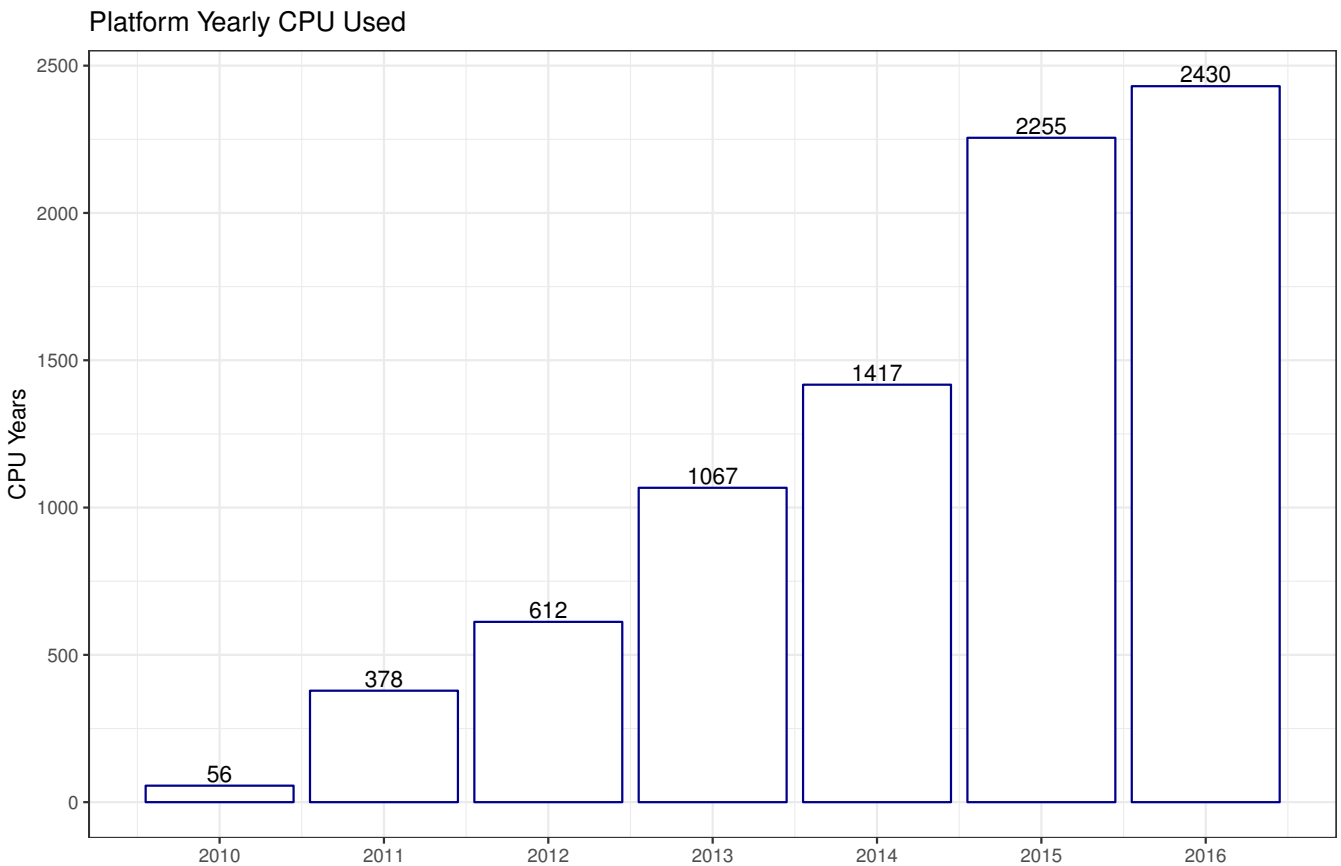






CPU-year usage since 2008

- **CPU-hour**: *work* done by a CPU in one hour of wall clock time



Summary

Chronology

- **Sept. 2016**: 2016 iris RFPs official release
  - ↪ RFP 160019: storage part
  - ↪ RFP 160020: computing/interconnect part
- **Oct. 12th, 2016**: responses from vendors
- **Nov. 16th 2016**: winner notifications — Total Budget: **1.6 M€**
  - ↪ RFP 160019 **Storage**: **Telindus/HPE/DDN**
  - ↪ RFP 160020 **Computing/interconnect**: **Post/DELL**
- **Dec. 12th 2016**: BDC confirmed to vendors
- **Mar. 6th 2016**: Dell racking + configuration starts
  - ↪ expected to last 3 weeks before we are given the hand on it
  - ↪ ... finally **racking** $\simeq$ end April 4th, 2017
    - ✓ fat-tree still incomplete, interconnect not properly configured
    - ✓ 2w for solving power balance not made according to our plan
    - ✓ continuous vendor failure to provide the requested SW config

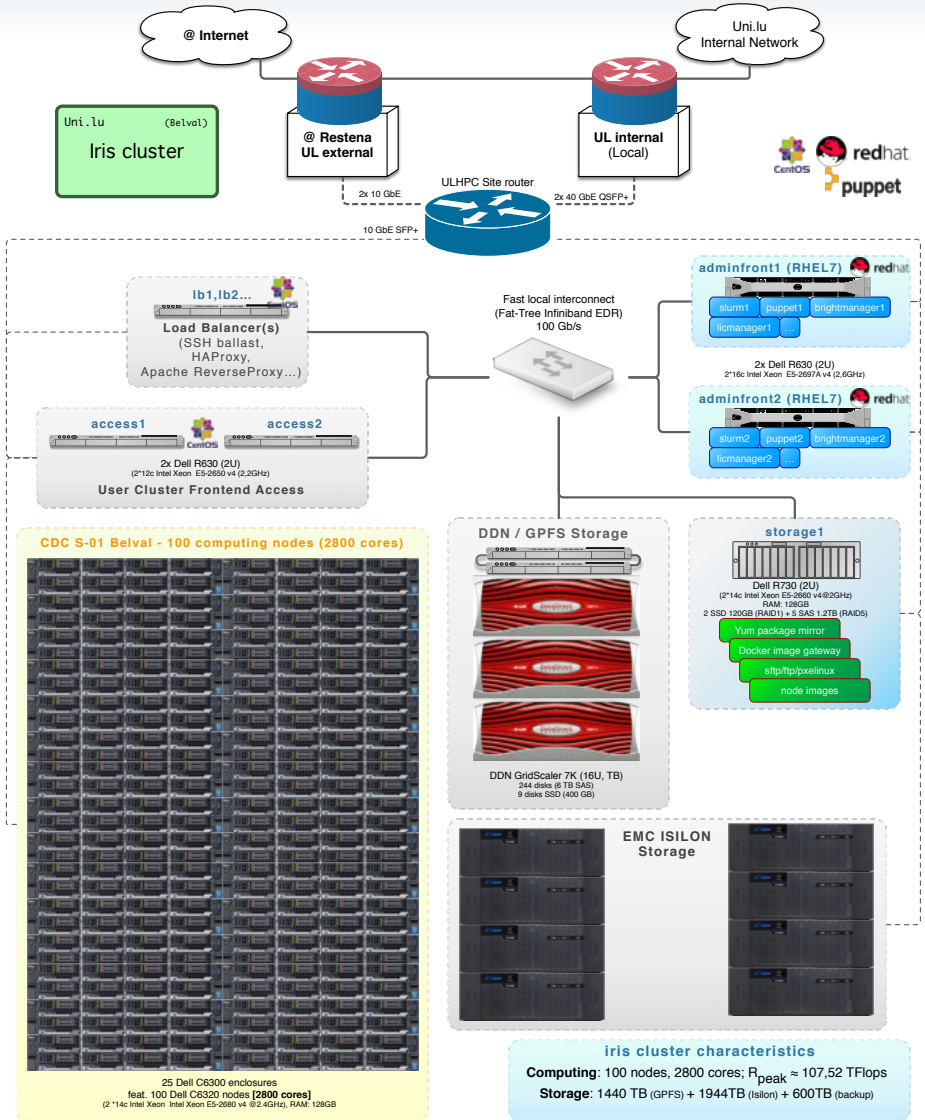
Chronology (cont.)

- **April 20th, 2016**: decision to take over the setup initiated by Dell
  - ↪ reverse-engineering on network stack configuration
  - ↪ alignement to plan proposed since Feb. 2017
  - ↪ Deployment of the administrative services
  - ↪ Deployment of the nodes
- **May 2th, 2017**: DDN team starts GPFS config. & validation
- **May 15th, 2017**: Fat-tree completed
  - ↪ Slurm configuration and QOS setup validated for production
  - ↪ Preliminary large-scale benchmarks completed
    - ✓ OSU/HPL/HPCG etc
    - ✓ IOR runs highlight GFPS stability issues
- **May 17th, 2017**: first completed RESIF-based software set build
- **May 29th, 2017**: DDN team still investigating statibility issues
  - ↪ Still pending as of June 2th. . .

Chronology (cont.)

- **June 2nd, 2017**: cluster opened for beta-test to users
- **June 7th, 2017**: UL HPC Team exclusive access to perform final qualifications
- **June 12th, 2017**: `iris` cluster released for production
  - ↪ during the UL HPC Scool 2017

The new iris cluster

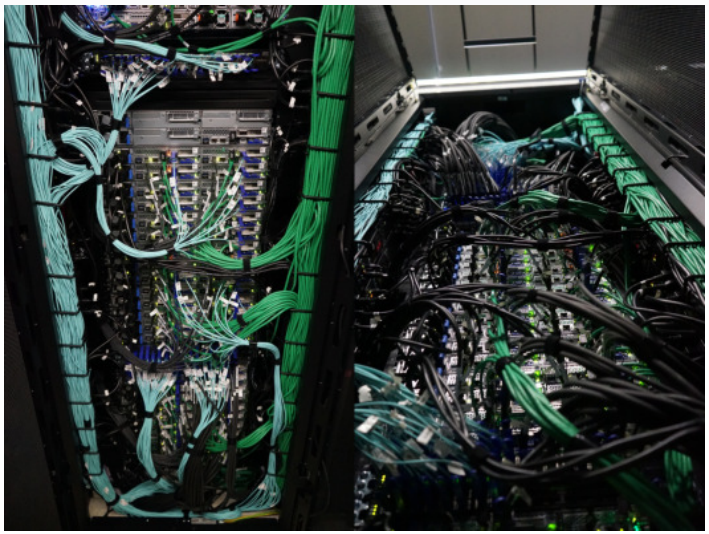


Iris Cluster Characteristics

- **100 nodes**, 2800 cores, **107.52 TFlops**
  - ↪ Dell C6320, Intel Xeon E5-2680v4@2.4 GHz [2x14c]
  - ↪ 128 GB RAM each
- **SpectrumsScale GPFS**: **1440 TB raw**
  - ↪ DDN GridScaler
  - ↪ GS7K base encl. + 3 SS8460 expansio
  - ↪ 248 disks (240x 6TB SED + 8 SSD)
- $\simeq$ 1500 cores reserved for Prof. Tkatchenko's group

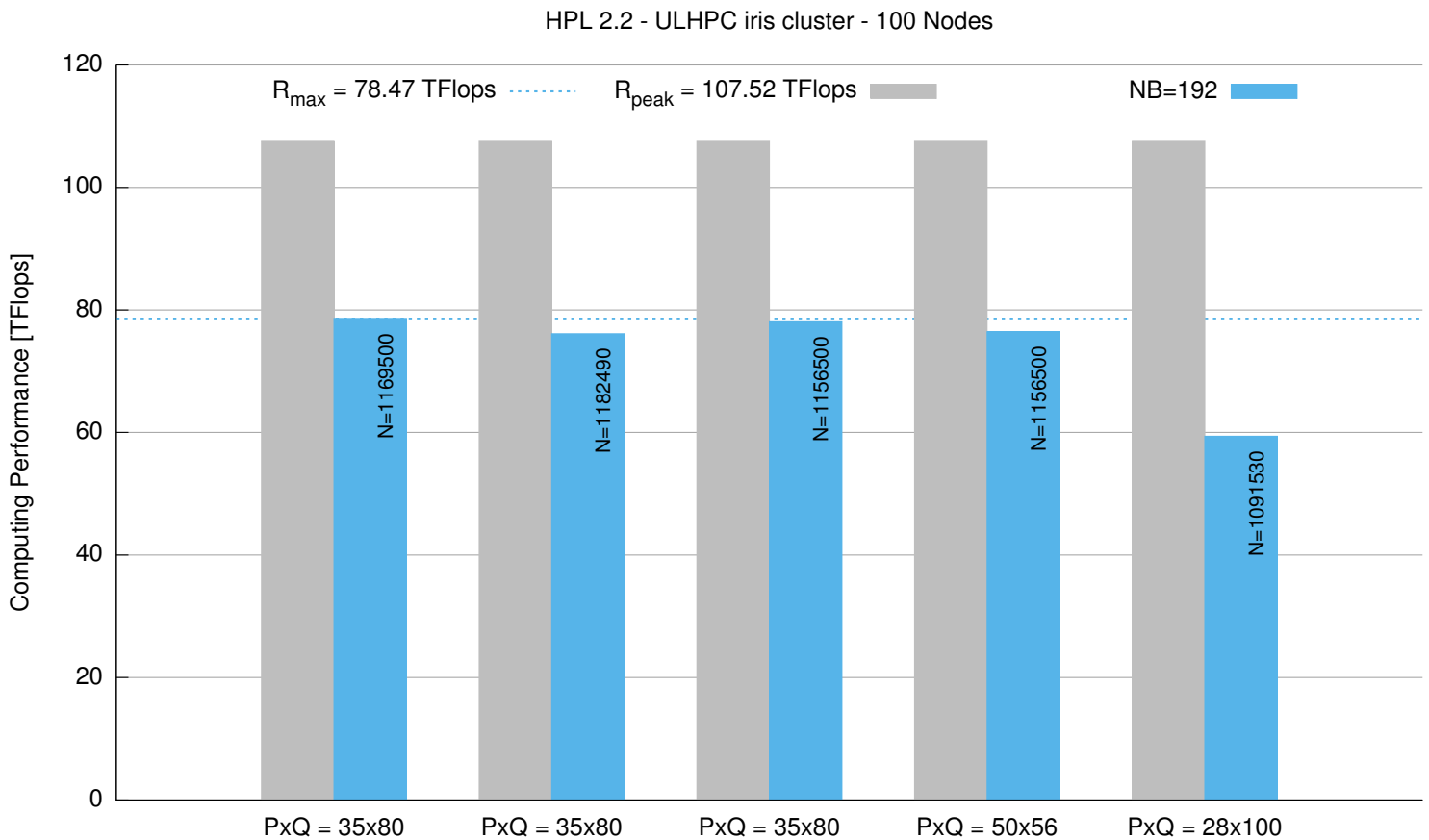
Software Stack Specifications

- **OS**: CentOS 7.3
- **Job scheduler**: SLURM 17.02
- **software**: updated Env. modules
  - ↪ RESIF/Easybuild refactored code
- Storage:
  - ↪ connected to Isilon
  - ↪ no scratch / Lustre for now
- **Interconnect**:
  - ↪ 10/40GB Ethernet network
  - ↪ **Infiniband EDR 100Gb/s** with non-blocking/Fat-Tree Topology
- Redundant / load-balanced services with:
  - ↪ 2x adminfront servers (cluster management)
  - ↪ 2x access servers (user frontend)
  - ↪ 2x storage servers



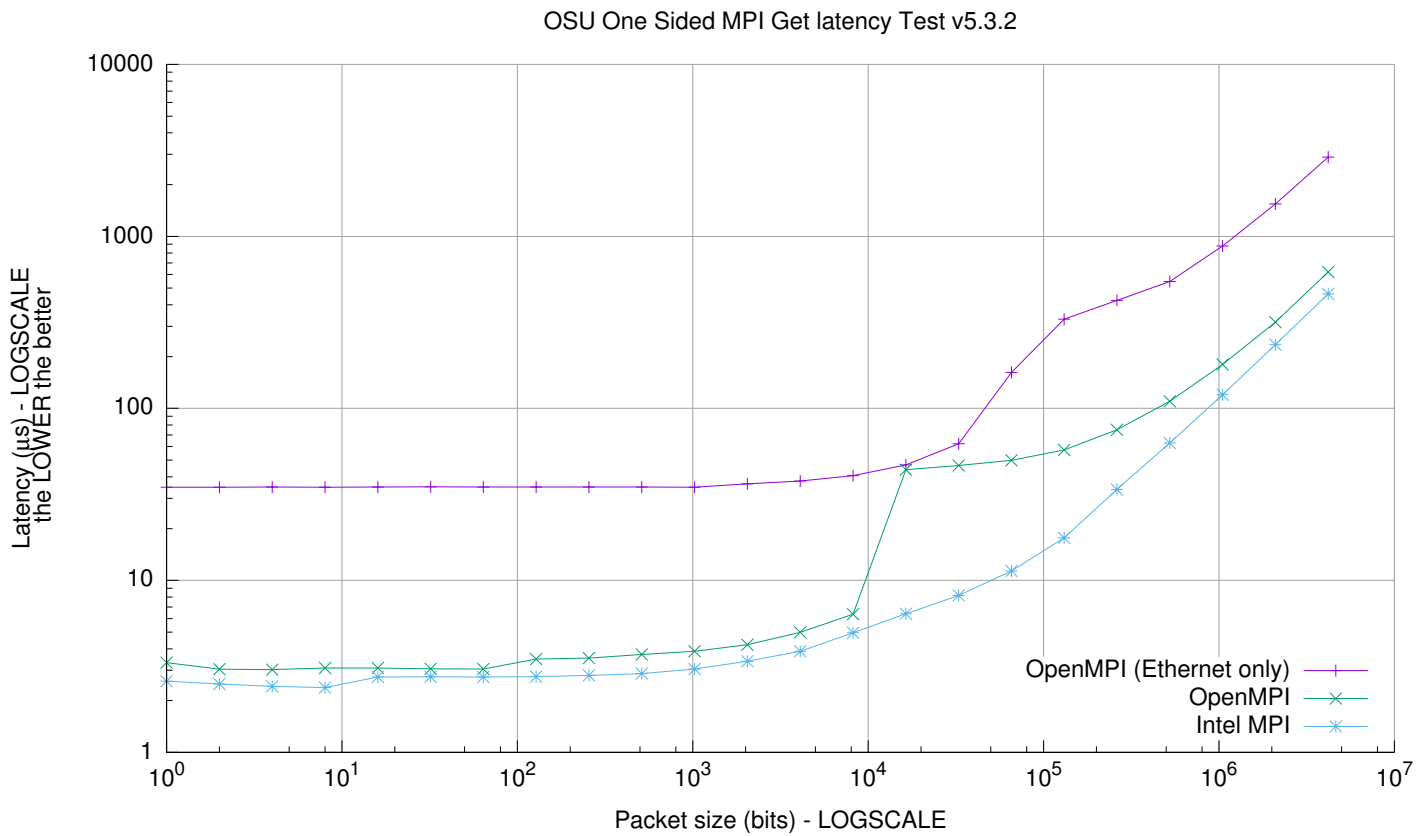
Computing Performances / HPL

- Based on High-Performance Linpack (HPL)
  - ↪ reference benchmark for Top 500



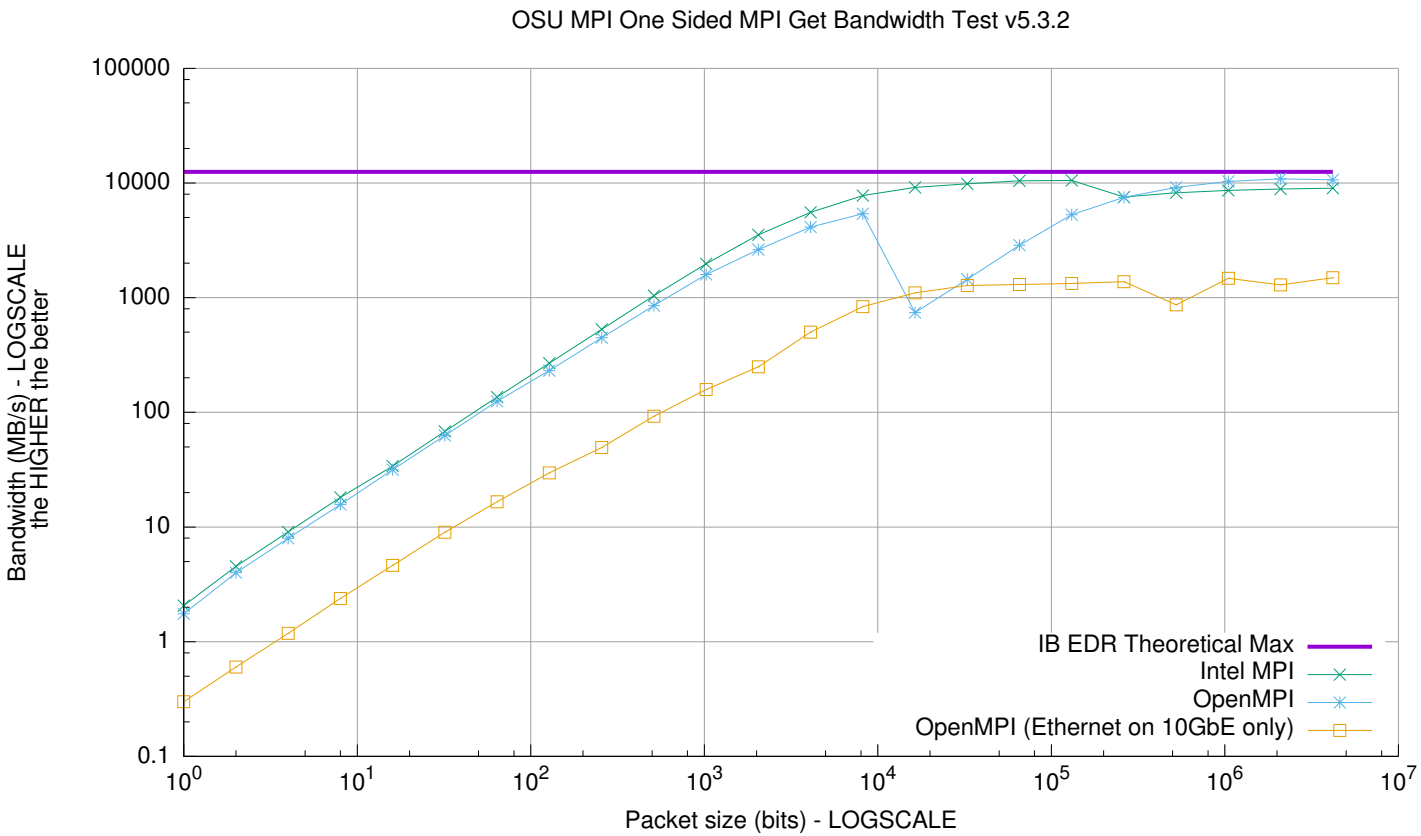
Interconnect Performances

- Based on OSU Micro-benchmarks



Interconnect Performances

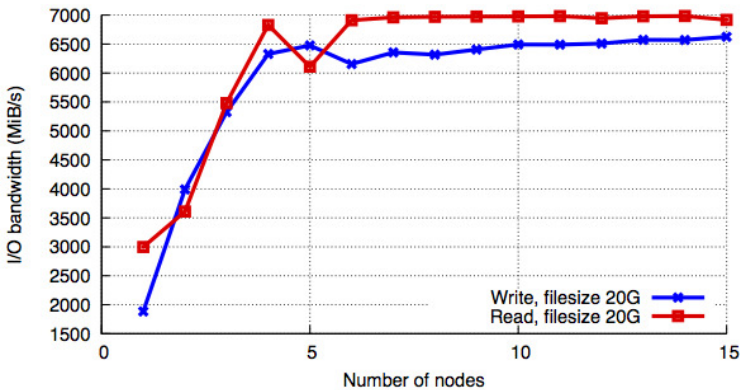
- Based on OSU Micro-benchmarks



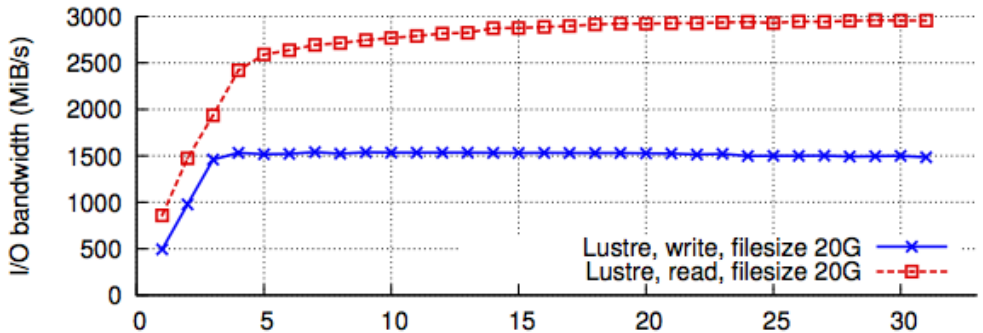
Storage Performances / IOR

- Based on Parallel filesystem I/O benchmark by LLNL

GPFS (gaia)



Lustre (gaia)



GPFS (iris)

#Nodes	Write [MiB/s]	Read [MiB/s]
1	3407,34	5162,10
10	8298,99	9329,27
20	8390,64	10047,88
30	8411,45	10139,65

Summary

Summary

General Guidelines

- The UL HPC is a ***shared*** resource
  - ↪ hundreds of users may be logged on at one time
  - ↪ hundreds of jobs may be running on all compute nodes,
- All users must practice ***good citizenship***
  - ↪ 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
  - ↪ **Don't run programs on the login nodes**
  - ↪ Plan large scale experiments during night-time or week-ends
    - ✓ **no more than 120 cores** during working day and working hours

General Guidelines

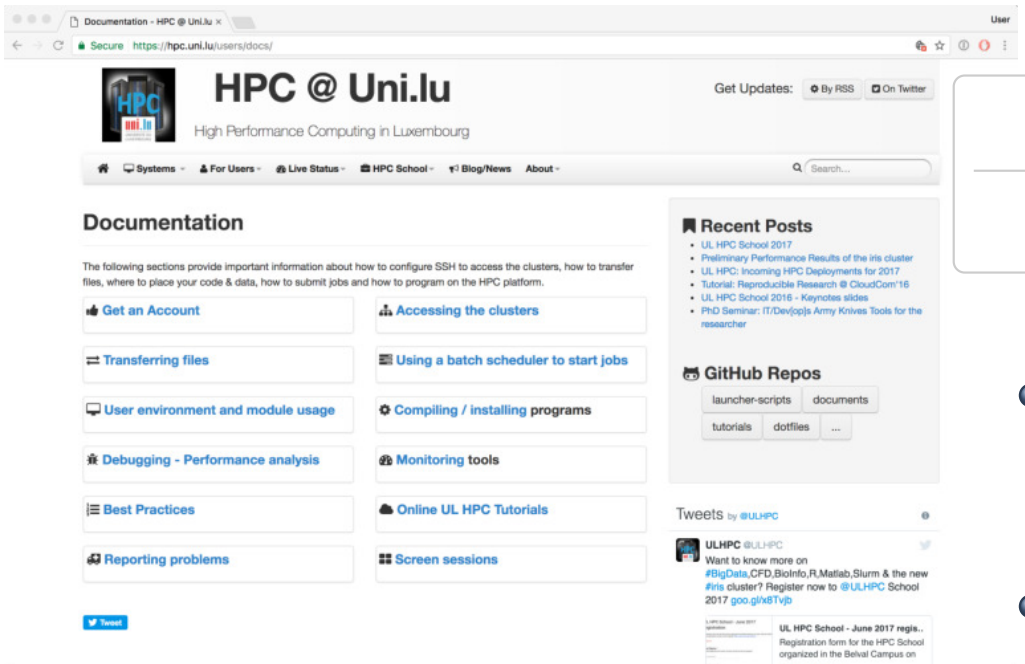
- The UL HPC is a ***shared*** resource
  - ↪ hundreds of users may be logged on at one time
  - ↪ hundreds of jobs may be running on all compute nodes,
- All users must practice ***good citizenship***
  - ↪ limit activities that may impact the system for other users.
  - ↪ **Do not abuse the shared filesystems**
    - ✓ Avoid too many simultaneous file transfers
    - ✓ regularly clean your directories from useless files
  - ↪ **Don't run programs on the login nodes**
  - ↪ Plan large scale experiments during night-time or week-ends
    - ✓ **no more than 120 cores** during working day and working hours

- For **ALL** publications having results produced using the UL HPC
  - ↪ Acknowledge / cite the UL HPC facility (using official banner)
  - ↪ Tag your publication upon registration on ORBiLu.

Summary

Documentation

http://hpc.uni.lu/users/getting_started.html



http://hpc.uni.lu

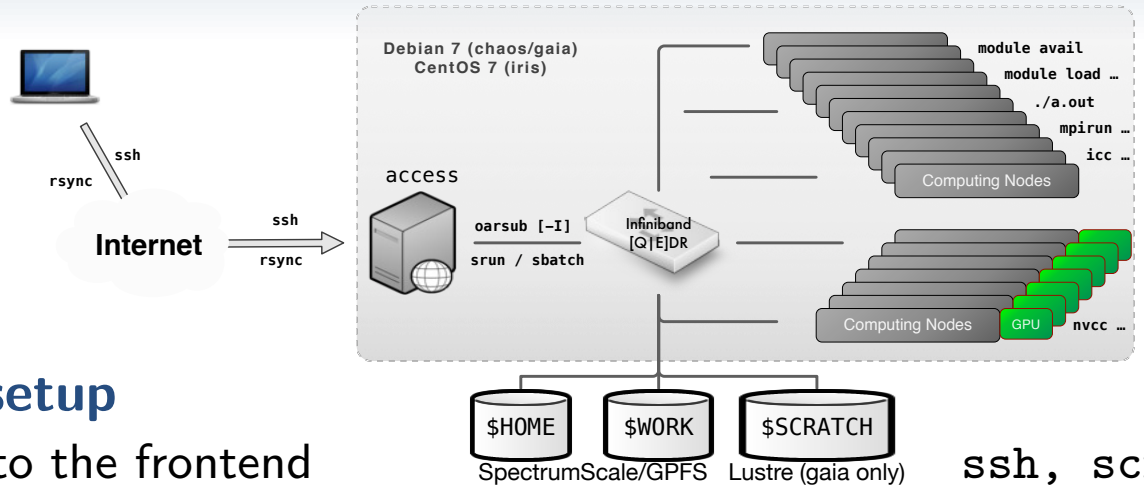
... aka the **rtfm** paradigm

Reference documentation

http://hpc.uni.lu/docs/

- Github Tutorials
  - ↪ http://ulhpc-tutorials.rtfd.io/
  - ↪ https://github.com/ULHPC/tutorials
- UL HPC Ticketing System
  - ↪ https://hpc-tracker.uni.lu/
- Ask other users hpc-users@uni.lu
  - ↪ . . . or us hpc-sysadmins@uni.lu

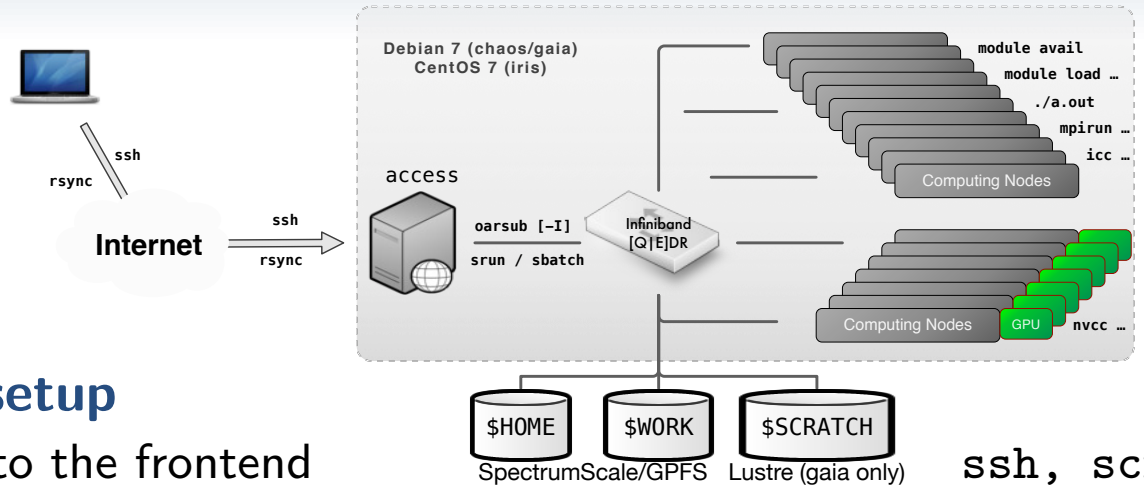
Typical Workflow on UL HPC resources



- **Preliminary setup**
  1. Connect to the frontend — `ssh, screen`
  2. Synchronize you code — `scp/rsync/svn/git`
  3. Reserve a few interactive resources — `oarsub -I [...]`

     or, on iris: `srun -p interactive [...]`
     - ✓ *(eventually)* build your program — `gcc/icc/mpicc/nvcc..`
     - ✓ Test on small size problem — `mpirun/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 — `oarsub -I [...]`

     or, on iris: `srun -p interactive [...]`
     - ✓ *(eventually)* build your program — `gcc/icc/mpicc/nvcc..`
     - ✓ Test on small size problem — `mpirun/srun/python/sh...`
     - ✓ Prepare a launcher script — `<launcher>.{sh|py}`
- **Real Experiment**
  1. Reserve passive resources — `oarsub [...] <launcher>`

     or, on iris: `sbatch -p {batch|long} [...] <launcher>`
  2. Grab the results — `scp/rsync/svn/git`

Summary

UL HPC resource manager: OAR

The OAR Batch Scheduler

http://oar.imag.fr

- Versatile resource and task manager
  - ↪ schedule **jobs** for users on the cluster **resource**
  - ↪ OAR resource = a node or part of it (CPU/core)
  - ↪ OAR job = execution time (**walltime**) on a set of resources

UL HPC resource manager: OAR

The OAR Batch Scheduler

http://oar.imag.fr

- Versatile resource and task manager
  - ↪ schedule **jobs** for users on the cluster **resource**
  - ↪ OAR resource = a node or part of it (CPU/core)
  - ↪ OAR job = execution time (**walltime**) on a set of resources

OAR main features includes:

- **interactive vs. passive (aka. batch) jobs**
- **best effort jobs**: use more resource, accept their release any time
- **deploy jobs** (Grid5000 only): deploy a customized OS environment
  - ↪ ... and have full (root) access to the resources
- **powerful resource filtering/matching**

Main OAR commands

oarsub	submit/reserve a job	(by default: **1 core for 2 hours**)
oardel	delete a submitted job	
oarnodes	shows the resources states	
oarstat	shows information about running or planned jobs	

	Submission
interactive	`oarsub [options] -I`
passive	`oarsub [options] scriptName`

- Each created job receive an identifier `JobID`
  - ↪ Default passive job log files: `OAR.JobID.std{out,err}`
- You can make a reservation with `-r "YYYY-MM-DD HH:MM:SS"`

Main OAR commands

oarsub	submit/reserve a job	(by default: **1 core for 2 hours**)
oardel	delete a submitted job	
oarnodes	shows the resources states	
oarstat	shows information about running or planned jobs	

	Submission
interactive	`oarsub [options] -I`
passive	`oarsub [options] scriptName`

- Each created job receive an identifier `JobID`
  - ↪ Default passive job log files: `OAR.JobID.std{out,err}`
- You can make a reservation with `-r "YYYY-MM-DD HH:MM:SS"`

Direct access to nodes by `ssh` is forbidden: use `oarsh` instead

OAR job environment variables

Once a job is created, some environments variables are defined:

Variable	Description
`$OAR_NODEFILE`	Filename which lists all reserved nodes for this job
`$OAR_JOB_ID`	OAR job identifier
`$OAR_RESOURCE_PROPERTIES_FILE`	Filename which lists all resources and their properties
`$OAR_JOB_NAME`	Name of the job given by the "-n" option of oarsub
`$OAR_PROJECT_NAME`	Job project name

Useful for MPI jobs for instance:

```
$> mpirun -machinefile $OAR_NODEFILE /path/to/myprog
```

... Or to collect how many cores are reserved per node:

```
$> cat $OAR_NODEFILE | uniq -c
```

OAR job types (gaia, chaos)

Job Type	Max Walltime (hour)	Max #active_jobs	Max #active_jobs_per_user
interactive	12:00:00	10000	5
default	120:00:00	30000	10
besteffort	9000:00:00	10000	1000

cf `/etc/oar/admission_rules/*.conf`

- **interactive**: useful to test / prepare an experiment
  - ↪ you get a shell on the first reserved resource
- **best-effort vs. default**: nearly unlimited constraints **YET**
  - ↪ a `besteffort` job can be killed as soon as a default job as no other place to go
  - ↪ enforce checkpointing (and/or idempotent) strategy

Characterizing OAR resources

Specifying wanted resources in a hierarchical manner

- Use the `-l` option of `oarsub`. Main constraints:

`enclosure=N`	number of enclosure
`nodes=N`	number of nodes
`core=N`	number of cores
`walltime=hh:mm:ss`	job's max duration

Specifying OAR resource properties

- Use the `-p` option of `oarsub`:

Syntax: `-p "property='value'"`

`gpu='{YES,NO}'`	has (or not) a GPU card
`host='fqdn'`	full hostname of the resource
`network_address='hostname'`	Short hostname of the resource
(Chaos only) `nodeclass='{k,b,h,d,r}'`	Class of node

Summary

Slurm Workload Manager

Documentation & comparison to OAR

`https://hpc.uni.lu/users/docs/scheduler.html`

Main change compared to the other clusters!!! (`gaia` etc.)

Slurm Workload Manager

Documentation & comparison to OAR

`https://hpc.uni.lu/users/docs/scheduler.html`

Main change compared to the other clusters!!! (`gaia` etc.)

- Predefined **Queues/Partitions**:
  - ↪ `batch` (Default) — **Max**: 30 nodes, 5 days walltime
  - ↪ `interactive` — **Max**: 2 nodes, 4h walltime, 10 jobs
  - ↪ `long` — **Max**: 2 nodes, 30 days walltime, 10 jobs
- Corresponding Quality of Service (QOS)
- Possibility to run `besteffort` jobs via the `qos-besteffort` QOS
- Accounts associated to supervisor (multiple associations possible)
- Proper group/user accounting

Slurm Job Management

- 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 user's tasks to complete their work
  - ↪ a requested node **partition** (job queue)
  - ↪ a requested **quality of service** (QoS) level which grants users specific accesses
  - ↪ a requested **account** for accounting purposes

By default...

- users submit jobs to a particular partition, **and** under a particular account (pre-set per user).

Slurm vs. OAR Main Commands

Action	SLURM command	OAR Command
Submit passive/batch job	`sbatch [...] $script`	`oarsub [...] $script`
Start interactive job	`srun [...] --pty bash`	`oarsub -I [...]`
Queue status	`squeue`	`oarstat`
User job status	`squeue -u $user`	`oarstat -u $user`
Specific job status (detailed)	`scontrol show job $jobid`	`oarstat -f -j $jobid`
Job accounting status (detailed)	`sacct --job $jobid -l`	
Delete (running/waiting) job	`scancel $jobid`	`oardel $jobid`
Hold job	`scontrol hold $jobid`	`oarhold $jobid`
Resume held job	`scontrol release $jobid`	`oarresume $jobid`
Node list and their properties	`scontrol show nodes`	`oarnodes`

Job Specifications

Specification	SLURM	OAR
Script directive	`#SBATCH`	`#OAR`
<n> **Nodes** request	`-N <n>`	`-l nodes=<n>`
<n> **Cores/Tasks** request	`-n <n>`	`-l core=<n>`
<c> Cores-per-node request	`--ntasks-per-node=<c>`	`-l nodes=<n>/core=<c>`
<c> Cores-per-task request (multithreading)	`-c=<c>`	
<m>**GB memory per node** request	`--mem=<m>GB`	
Walltime request	`-t <mm>/<days-hh[:mm:ss]>`	`-l walltime=hh[:mm:ss]`
Job array	`--array <specification>`	`--array <count>`
Job name	`-J <name>`	`-n <name>`
Job dependency	`-d <specification>`	`-a <jobid>`
Property request	`-C <specification>`	`-p "<property>=<value>"`
Specify job partition/queue	`-p <partition>`	`-t <queue>`
Specify job qos	`--qos <qos>`	
Specify account	`-A <account>`	
Specify email address	`--mail-user=<email>`	`--notify "mail:<email>"`

Available Node partitions

- Slurm Command Option `-p, --partition=<partition>`
  - ↪ **Ex**: `{srun,sbatch} -p batch [...]`
- Date format: `-t <minutes>` or `-t <D>-<H>:<M>:<S>`

Partition	#Nodes	Default time	Max time	Max nodes/user
`batch`	80%	0-2:0:0 [**2h**]	5-0:0:0 [**5d**]	unlimited
`interactive`	10%	0-1:0:0 [**1h**]	0-4:0:0 [**4h**]	2
`long`	10%	0-2:0:0 [**2h**]	30-0:0:0 [**30d**]	2

Quality of Service (QOS)

- Slurm Command Option `--qos=<qos>`
- There is **no default** QOS (due to the selected scheduling model)
  - ↪ you **MUST** provide upon **any job submission**

QoS	User group	Max nodes	Max jobs/user	Description
qos-besteffort	**ALL**	**no limit**		Preemptible jobs, requeued on preemption
qos-batch	**ALL**	**30**	100	Normal usage of the `batch` partition
qos-interactive	**ALL**	**8**	10	Normal usage of the `interactive` partition
qos-long	**ALL**	**8**	10	Normal usage of the `long` partiton
qos-batch-###	TBD	TBD	100	Special usage of the `batch` partition
qos-interactive-###	TBD	TBD	10	Special usage of the `interactive` partition
qos-long-###	TBD	TBD	10	Special usage of the `long` partiton

- Special A.TKATCHENKO group settings:
  - ↪ Use partitions `{interactive,batch,long}-001`
  - ↪ Use qos `qos-{interactive,batch,long}-001`

Accounts

- Every user job runs under a group account
  - ↪ granting access to specific QOS levels.

Account	Parent Account
UL	
FSTC	UL
FDEF	UL
FLSHASE	UL
LCSB	UL
SNT	UL
Professor $X	*FACULTY* /IC
Group head $G	*FACULTY* /IC
Researcher $R	Professor $X
Researcher $R	Group head $G
Student $S	Professor $X
Student $S	Group head $G
External collaborator $E	Professor $X
External collaborator $E	Group head $G

```
$> sacctmgr list associations where users=$USER \
          format=Account%30s,User,Partition,QOS
```

Typical Workflow

```
# Run an interactive job     -- make an alias 'si [...]'
$> srun -p interactive --qos qos-interactive --pty bash
# Ex: interactive job for 30 minutes, with 2 nodes/4 tasks per node
$> si --time=0:30:0 -N 2 --ntasks-per-node=4
# Run a [passive] batch job -- make an alias 'sb [...]'
$> sbatch -p batch     --qos qos-batch        /path/to/launcher.sh
# Will create (by default) slurm-<jobid>.out file
```

Environment variable	SLURM	OAR
Job ID	$SLURM_JOB_ID	$OAR_JOB_ID
Resource list	$SLURM_NODELIST **#List not file!**	$OAR_NODEFILE
Job name	$SLURM_JOB_NAME	$OAR_JOB_NAME
Submitting user name	$SLURM_JOB_USER	$OAR_USER
Task ID within job array	$SLURM_ARRAY_TASK_ID	$OAR_ARRAY_INDEX
Working directory at submission	$SLURM_SUBMIT_DIR	$OAR_WORKING_DIRECTORY
Number of nodes assigned to the job	$SLURM_NNODES	
Number of tasks of the job	$SLURM_NTASKS	$(wc -l ${OAR_NODEFILE})

- **Note**: create the equivalent of $OAR_NODEFILE in Slurm:
  - ↪ `srun hostname | sort -n > hostfile`

Other Features

- **Checkpoint / Restart**
  - ↪ Based on DMTCP: Distributed MultiThreaded CheckPointing
  - ↪ see the official DMTCP launchers
  - ↪ ULHPC example
- Binding with Allinea Performance Report: see ULHPC School
- More advanced admission rules
  - ↪ to simplify CLI
- Container Shifter
  - ↪ Work in progress, not yet available on the

Basic Slurm Launcher Examples

Documentation

https://hpc.uni.lu/users/docs/slurm_launchers.html

See also **PS1**, **PS2** and **PS3**

```
#!/bin/bash -l
# Request one core for 5 minutes in the batch queue

#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=0-00:05:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

[...]
```

Basic Slurm Launcher Examples (cont.)

```
#!/bin/bash -l
# Request two cores on each of two nodes for 3 hours

#SBATCH -N 2
#SBATCH --ntasks-per-node=2
#SBATCH --time=0-03:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"
echo "== Node list: ${SLURM_NODELIST}"
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"

[...]
```

Basic Slurm Launcher Examples (cont.)

```
#!/bin/bash -l
# Request one core and half the memory available on an iris cluster
# node for one day
#
#SBATCH -J MyLargeMemorySequentialJob
#SBATCH --mail-type=end,fail
#SBATCH --mail-user=Your.Email@Address.lu
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=64GB
#SBATCH --time=1-00:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

echo "== Starting run at $(date)"
echo "== Job ID: ${SLURM_JOBID}"
echo "== Node list: ${SLURM_NODELIST}"
echo "== Submit dir. : ${SLURM_SUBMIT_DIR}"
```

pthreads/OpenMP Slurm Launcher

```
#!/bin/bash -l
# Single node, threaded (pthreads/OpenMP) application launcher,
# using all 28 cores of an iris cluster node:

#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 28
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
/path/to/your/threaded.app
```

MATLAB Slurm Launcher

```
#!/bin/bash -l
# Single node, multi-core parallel application (MATLAB, Python, R...)
# launcher, using all 28 cores of an iris cluster node:

#SBATCH -N 1
#SBATCH --ntasks-per-node=28
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load base/MATLAB
matlab -nodisplay -nosplash < /path/to/inputfile > /path/to/outputfile
```

Intel MPI Slurm Launchers

- Official SLURM guide for Intel MPI

```
#!/bin/bash -l
# Multi-node parallel application IntelMPI launcher,
# using 128 distributed cores:

#SBATCH -n 128
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/intel
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun -n $SLURM_NTASKS /path/to/your/intel-toolchain-compiled-application
```

OpenMPI Slurm Launchers

- Official SLURM guide for Open MPI

```
#!/bin/bash -l
# Multi-node parallel application openMPI launcher,
# using 128 distributed cores:

#SBATCH -n 128
#SBATCH -c 1
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/foss
mpirun -n $SLURM_NTASKS /path/to/your/foss-toolchain-compiled-application
```

Hybrid IntelMPI+OpenMP Launcher

```
#!/bin/bash -l
# Multi-node hybrid application IntelMPI+OpenMP launcher,
# using 28 threads per node on 10 nodes (280 cores):

#SBATCH -N 10
#SBATCH --ntasks-per-node=1
#SBATCH -c 28
#SBATCH --time=0-01:00:00
#SBATCH -p batch
#SBATCH --qos=qos-batch

module load toolchain/intel
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
srun -n $SLURM_NTASKS /path/to/your/parallel-hybrid-app
```

Summary

Reporting Problems

https://hpc.uni.lu/users/docs/report_pbs.html

- **First checks**
  1. My issue is probably documented — see User Doc
  2. An event is on-going — cf mail from `hpc-platform@uni.lu`
  3. check the state of your nodes
     - ✓ `{ oarsub -C <jobid> | ssh <node>}; htop` *on active jobs*
     - ✓ `{ oarsub -f -j <jobid> }` *post-mortem (check the events field)*
       iris: `scontrol show job <jobid>` OR `sacct --job <jobid> -l`
     - ✓ Ganglia on your node(s) https://hpc.uni.lu/status/ganglia.html

Reporting Problems

https://hpc.uni.lu/users/docs/report_pbs.html

- **First checks**
  1. My issue is probably documented — see User Doc
  2. An event is on-going — cf mail from `hpc-platform@uni.lu`
  3. check the state of your nodes
     - ✓ `{ oarsub -C <jobid> | ssh <node>}; htop` *on active jobs*
     - ✓ `{ oarsub -f -j <jobid> }` *post-mortem (check the events field)*
       iris: `scontrol show job <jobid>` OR `sacct --job <jobid> -l`
     - ✓ Ganglia on your node(s) — https://hpc.uni.lu/status/ganglia.html

- **ONLY NOW**, consider the following depending on the severity:
  - ↪ Open an new issue on http://hpc-tracker.uni.lu (**preferred**)
  - ↪ Mail (only now) us `hpc-sysadmins@uni.lu`
  - ↪ **Ask the help of other users** `hpc-users@uni.lu`

Reporting Problems

https://hpc.uni.lu/users/docs/report_pbs.html

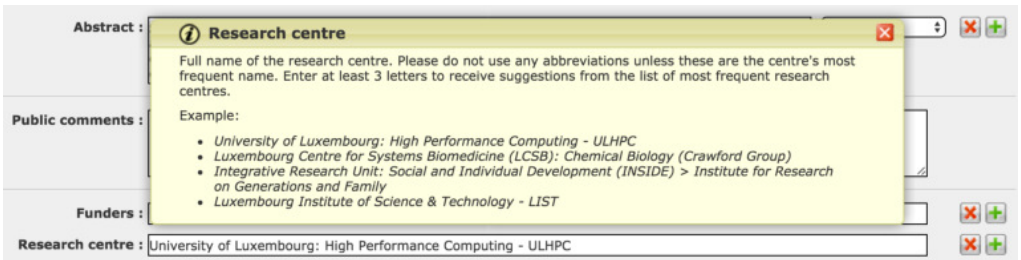
- **First checks**
  1. My issue is probably documented — see User Doc
  2. An event is on-going — cf mail from `hpc-platform@uni.lu`
  3. check the state of your nodes
     - ✓ `{ oarsub -C <jobid> | ssh <node>}; htop` *on active jobs*
     - ✓ `{ oarsub -f -j <jobid> }` *post-mortem (check the events field)*
       iris: `scontrol show job <jobid>` OR `sacct --job <jobid> -l`
     - ✓ Ganglia on your node(s) — https://hpc.uni.lu/status/ganglia.html

- **ONLY NOW**, consider the following depending on the severity:
  - ↪ Open an new issue on http://hpc-tracker.uni.lu (**preferred**)
  - ↪ Mail (only now) us — `hpc-sysadmins@uni.lu`
  - ↪ **Ask the help of other users** — `hpc-users@uni.lu`

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

Reporting Obtained Results

- In your **scientific publications**: *as per Acceptable Use Policy (AUP)*
  - ↪ **acknowledge** your usage of the UL HPC platform
  - ↪ *(if possible)* **cite** the UL HPC paper `\cite{VBCG_HPCS14}`
- **More importantly**: add ULHPC Tag on your ORBi[lu] publication



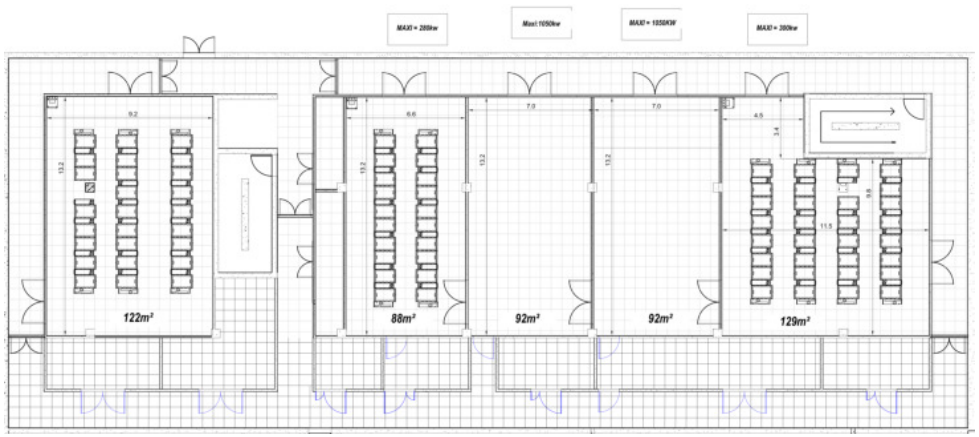

```
@InProceedings{VBCG_HPCS14,
   author =       {S. Varrette and P. Bouvry and H. Cartiaux and F. Georgatos},
   title =        {Management of an Academic HPC Cluster: The UL Experience},
   booktitle =    {Proc. of the 2014 Intl. Conf. on High Performance Computing \& Simulation (HPCS 2014)},
   year =         {2014},
   pages =        {959--967},
   month =        {July},
   address =      {Bologna, Italy},
   publisher =    {IEEE},
}
```

Summary

Infrastructure Plans starting 2017

MSA CDC S-02 as the new UL HPC Data Center (DC)

- $\simeq$500m^2 for max. 5 server rooms sustaining HPC requirements
- DC preparation will result in **2 rooms** being ready early 2017
  - ↪ **RFP 1** (DC infrastructure): **Oct. 2016** (SIU)
  - ↪ **RFP 2 & 3** (HPC + storage equipment): **Sept. 2016** (HPC)
  - ↪ **RFP 4** (DLC HPC): **2018** (HPC)

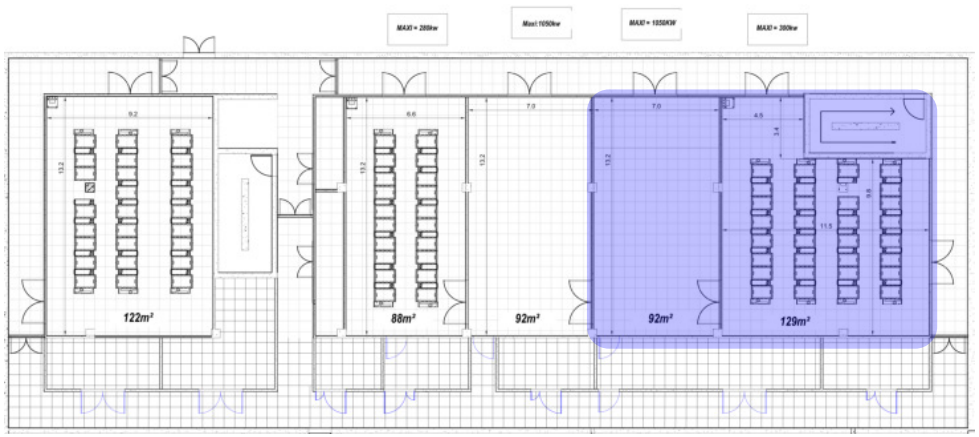


- $\simeq$ **1050kW** per **HPC** room
  - ↪ Direct Liquid Cooling (DLC)
- $\simeq$ **300kW** per **storage** room
  - ↪ rooms 1, 2 & 5

Infrastructure Plans starting 2017

MSA CDC S-02 as the new UL HPC Data Center (DC)

- $\simeq$500m^2 for max. 5 server rooms sustaining HPC requirements
- DC preparation will result in **2 rooms** being ready early 2017
  - ↪ **RFP 1** (DC infrastructure): **Oct. 2016** (SIU)
  - ↪ **RFP 2 & 3** (HPC + storage equipment): **Sept. 2016** (HPC)
  - ↪ **RFP 4** (DLC HPC): **2018** (HPC)



- $\simeq$ **1050kW** per **HPC** room
  - ↪ Direct Liquid Cooling (DLC)
- $\simeq$ **300kW** per **storage** room
  - ↪ rooms 1, 2 & 5

ETP4HPC

http://www.etp4hpc.eu

- **European Technology Platform (ETP) for HPC**
  - ↪ Industry-led forum founded by stakeholders of HPC technology
  - ↪ Providing the framework to define research priorities and actions
  - ↪ **Objective**: EU growth, competitiveness, sustainability by HPC
  - ↪ **Strategic Research Agenda**
    - ✓ **Creation** of new technologies within the entire HPC stack
    - ✓ **Improvement** of system characteristics (Extreme Scale Reqs.)
    - ✓ **New deployment** fields and **expansion** of HPC utilization

ETP4HPC

http://www.etp4hpc.eu

- **European Technology Platform (ETP) for HPC**
  - ↪ Industry-led forum founded by stakeholders of HPC technology
  - ↪ Providing the framework to define research priorities and actions
  - ↪ **Objective**: EU growth, competitiveness, sustainability by HPC
  - ↪ **Strategic Research Agenda**
    - ✓ **Creation** of new technologies within the entire HPC stack
    - ✓ **Improvement** of system characteristics (Extreme Scale Reqs.)
    - ✓ **New deployment** fields and **expansion** of HPC utilization

Since July 2016...

- **UL is an official member of ETP4HPC!**
  - ↪ participation of key UL HPC experts in various WG

EU HPC Initiatives In progress

PRACE

- Partnership for Advanced Computing in Europe
- Non-profit association with 25 member countries
- Providing access to EU Tier-0 compute & data resources
  - ↪ for large-scale scientific and engineering applications
  - ↪ **Objective**:
    - ✓ enable high impact scientific discovery and engineering R&D
    - ✓ enhance European competitiveness

EU HPC Initiatives In progress

PRACE

- Partnership for Advanced Computing in Europe
- Non-profit association with 25 member countries
- Providing access to EU Tier-0 compute & data resources
  - ↪ for large-scale scientific and engineering applications
  - ↪ **Objective**:
    - ✓ enable high impact scientific discovery and engineering R&D
    - ✓ enhance European competitiveness

- UL to apply as official national representative for PRACE
  - ↪ nomination pending approval by ministry

EU HPC Initiatives In progress

- Important Project of Common European Interest
- IPCEI on *HPC and Big Data Application*
  - ↪ part of Junker plan
  - ↪ launched on Nov. 17th 2015 (at European Data Forum)
  - ↪ $\simeq$ 3 B€ european investment
- Lead by Luxembourg through Ministry of Economy
  - ↪ Jean-Marie Spauss appointed as advisor to MECO
  - ↪ UL, LIST & Luxinnovation to support MECO

- **Toward a National HPC Center of Excellence**
  - ↪ **Euro-HPC** project
  - ↪ effective deployment and implementation planned for **2018**

Thank you for your attention...

Questions?

http://hpc.uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette & The UL HPC Team
University of Luxembourg, Belval Campus:
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu