High Performance Computing (HPC) at UL

Overview and Usage

UL HPC Management Team

University of Luxembourg, Luxembourg

Summary

Computing / Storage Performances

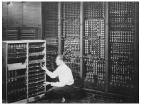
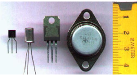
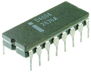
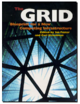
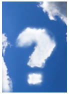
- **HPC**: **H**igh **P**erformance **C**omputing

Main HPC Performance Metrics

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

Evolution of Computing Systems

arpanet → internet

1st Generation	2nd	3rd	4th	5th			
					SETI@HOME 		
ENIAC	Transistors	Integrated Circuit	Micro-Processor	Beowulf Cluster		Multi-Core Processor	Cloud
180,000 tubes 30 t, 170 m^2	Replace tubes 1959: IBM 7090	Thousands of transistors in one circuit 1971: Intel 4004	Millions of transistors in one circuit 1989: Intel 80486			Multi-core processor 2005: Pentium D	 HW diversity
150 Flops	33 KFlops	0.06 Mips	1 MFlops	74 MFlops		2 GFlops	

1946 — 1956 — 1963 — 1974 — 1980 — 1994 — 1998 — 2005 — 2010

Why High Performance Computing ?

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

- Accelerate research by accelerating **computations**

14.4 **G**Flops

(Dual-core i7 1.8GHz)

49.872**T**Flops

(400computing nodes, 4280cores)

- Increase **storage** capacity

2TB (1 disk)

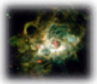
3364.4TB raw(642disks)

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

HPC at the Heart of our Daily Life

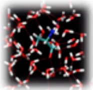
- **Today...** Research, Industry, Local Collectivities

Electro-
Magnetics

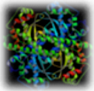


Computational Chemistry
Quantum Mechanics

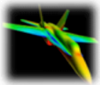
Computational Chemistry
Molecular Dynamics



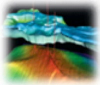
Computational
Biology



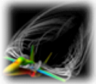
Structural Mechanics
Implicit



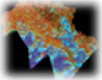
Seismic
Processing



Computational Fluid
Dynamics



Reservoir
Simulation



Rendering
Ray Tracing

Climate / Weather
Ocean Simulation

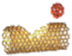
Data Analytics

Structural Mechanics
Explicit

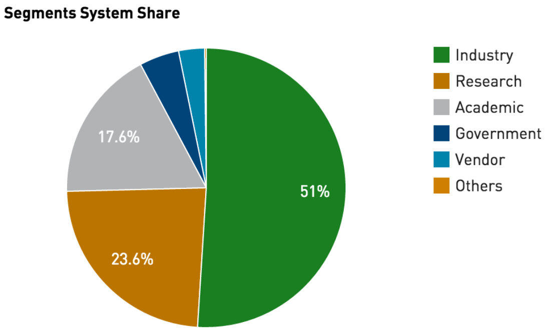
- ... **Tomorrow**: applied research, digital health, nano/bio techno

Ageing
Medecine
Biology

Materials
Spintronic
Nano-Sciences



HPC in Worldwide Strategies



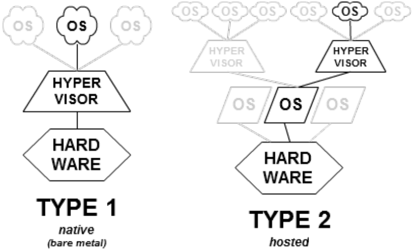
- EU: **77 B€** for **H2020** program 2014 → 2020

Computing for Researchers

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

Computing for Researchers

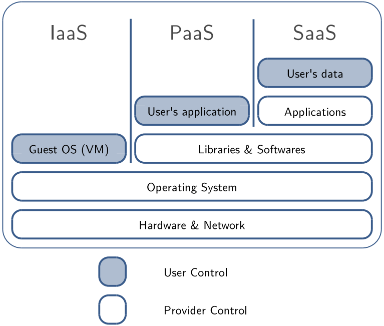
- Regular PC / Local Laptop / Workstation
  - ↪ Native OS (Windows, Linux, Mac etc.)
  - ↪ Virtualized OS through an **hypervisor**
    - ✓ Hypervisor: core virtualization engine / environment
    - ✓ **Performance loss:** $\geq$ 20%



Xen, VMWare ESXi, KVM VirtualBox

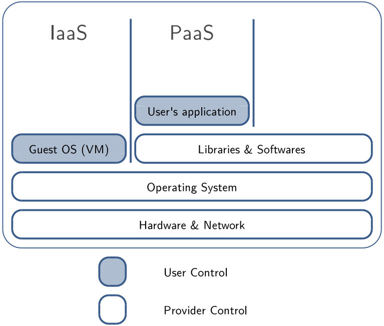
Computing for Researchers

- Cloud Computing Platform
  - ↪ **Infrastructure** as a Service (SaaS)



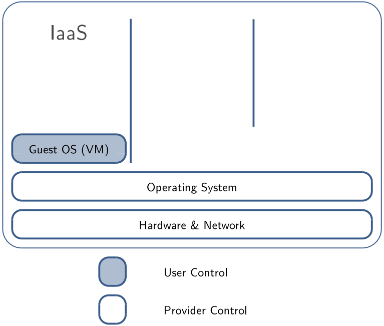
Computing for Researchers

- Cloud Computing Platform
  - ↪ **Platform** as a Service (PaaS)



Computing for Researchers

- Cloud Computing Platform
  ↪ **Software** as a Service (IaaS)



Computing for Researchers

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

Computing for Researchers

- High Performance Computing platforms
  - $\hookrightarrow$ For **Speedup**, **Scalability** and **Faster Time to Solution**

YET...

$$\mathbf{PC} \neq \mathbf{HPC}$$

Computing for Researchers

- High Performance Computing platforms
  - $\hookrightarrow$ For **Speedup**, **Scalability** and **Faster Time to Solution**

YET...

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

- HPC $\simeq$ Formula 1
  - $\hookrightarrow$ can end badly, even after minor errors

Jobs, Tasks & Local Execution

```
$> ./myprog
```

Jobs, Tasks & Local Execution

```
$> ./myprog
```

Jobs, Tasks & Local Execution

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

Jobs, Tasks & Local Execution

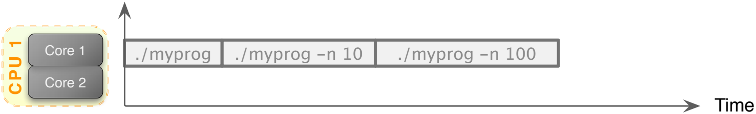
```
$> ./myprog
$> ./myprog -n 10
```

Jobs, Tasks & Local Execution

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

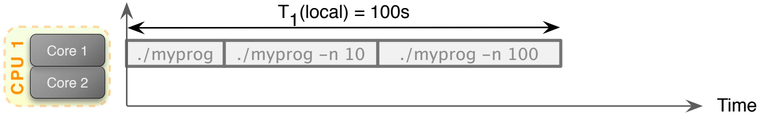
Jobs, Tasks & Local Execution

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

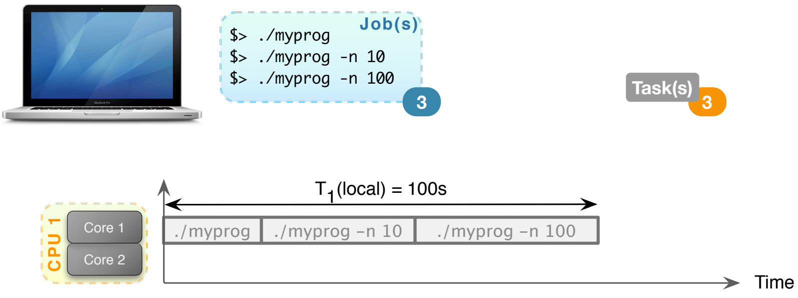


Jobs, Tasks & Local Execution

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



Jobs, Tasks & Local Execution



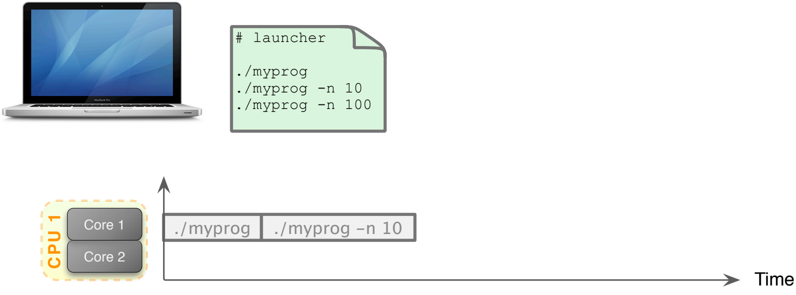
Jobs, Tasks & Local Execution

```
# launcher

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

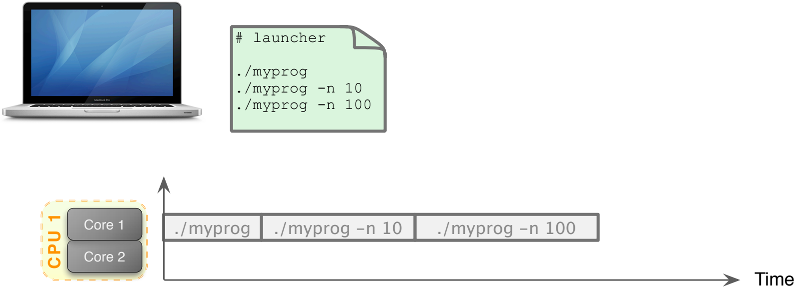
Jobs, Tasks & Local Execution

```
# launcher

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

Jobs, Tasks & Local Execution

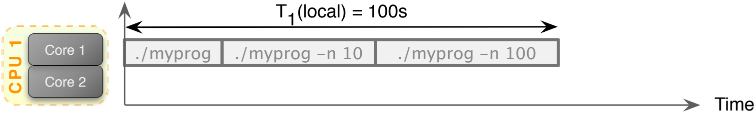


Jobs, Tasks & Local Execution

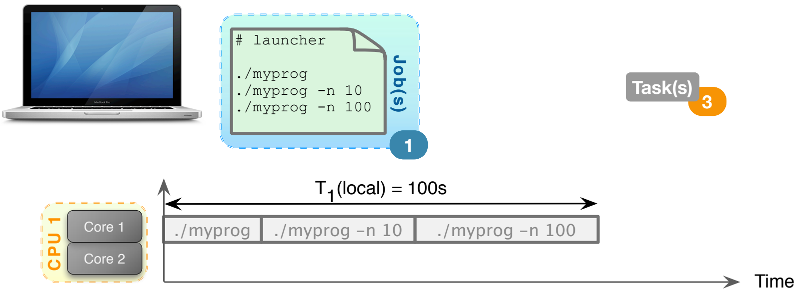


Jobs, Tasks & Local Execution

```
# launcher

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



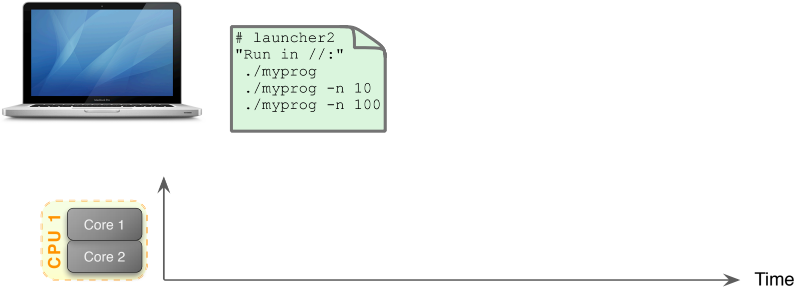
Jobs, Tasks & Local Execution



Jobs, Tasks & Local Execution

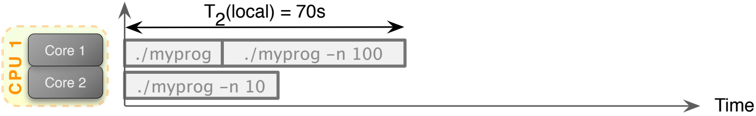
```
# launcher

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

Jobs, Tasks & Local Execution

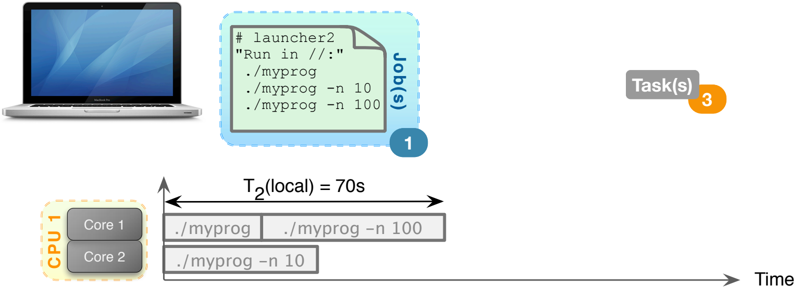


Jobs, Tasks & Local Execution

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

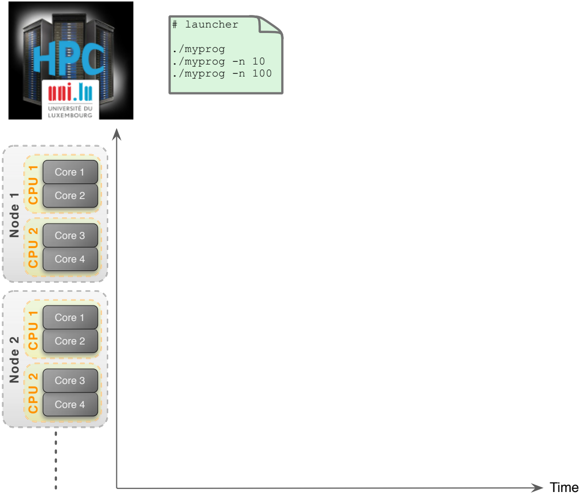


Jobs, Tasks & Local Execution



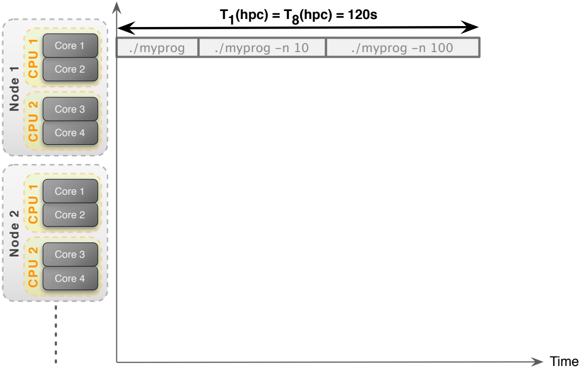
Jobs, Tasks & HPC Execution

```
# launcher

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

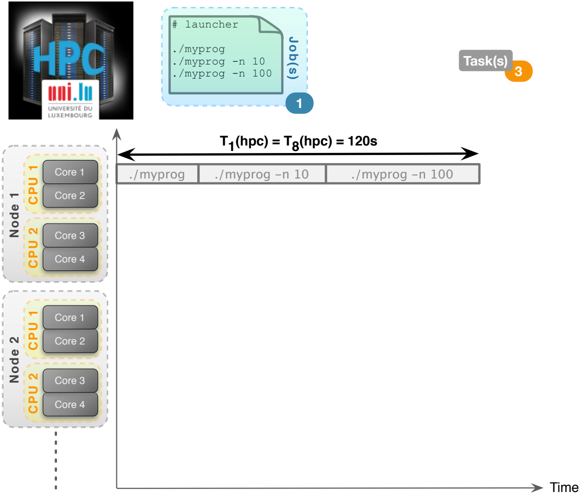


Jobs, Tasks & HPC Execution

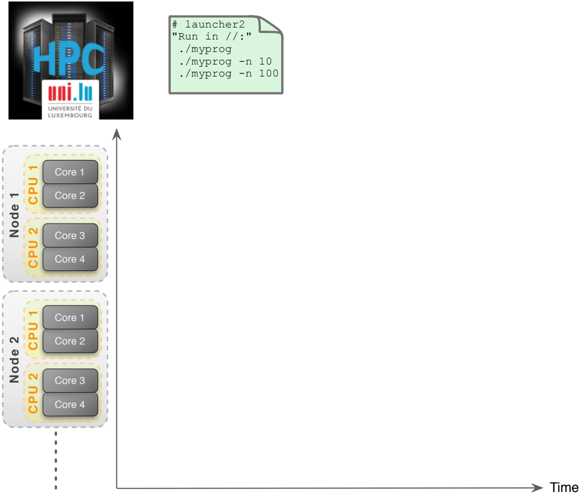
```
# launcher

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



Jobs, Tasks & HPC Execution

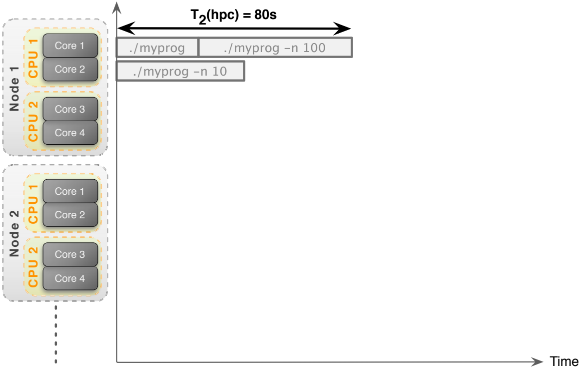


Jobs, Tasks & HPC Execution

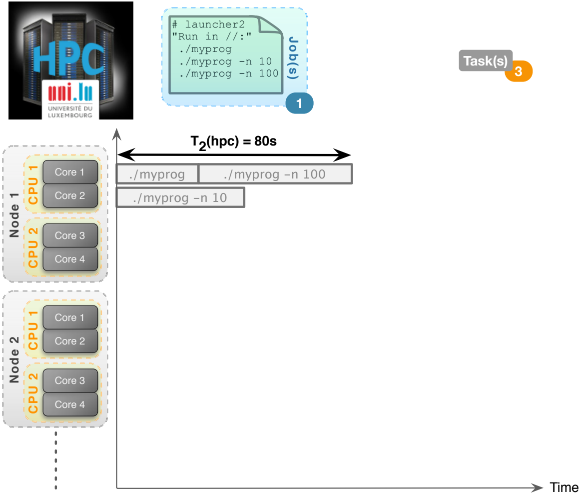


Jobs, Tasks & HPC Execution

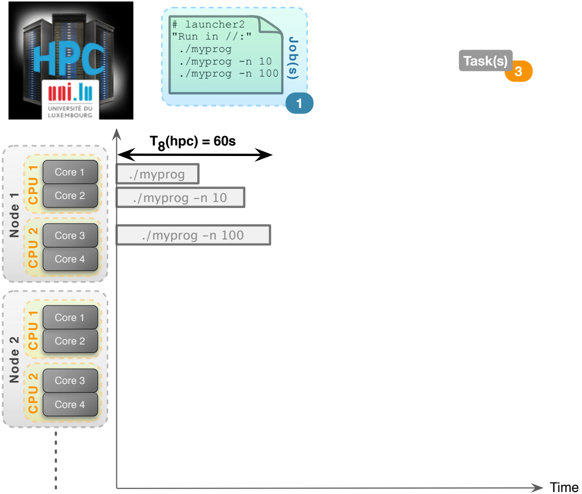
```
# launcher2
"Run in //:"
 ./myprog
 ./myprog -n 10
 ./myprog -n 100
```



Jobs, Tasks & HPC Execution



Jobs, Tasks & HPC Execution



Local vs. HPC Executions

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

Local vs. HPC Executions

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

- **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	$T_1(\text{local}) = 100s$	$T_1(\text{hpc}) = \mathbf{120s}$
Parallel/Distributed	$T_2(\text{local}) = 70s$	$T_2(\text{hpc}) = 80s$ $T_8(\text{hpc}) = 120s$ $T_8(\text{hpc}) = \mathbf{60s}$

- **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

In your program

```
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

In your program

```
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]);

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

In your program

```
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]);

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

Identifying Potential Parallelism

In your program

```
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);
```

Identifying Potential Parallelism

In your program

```
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

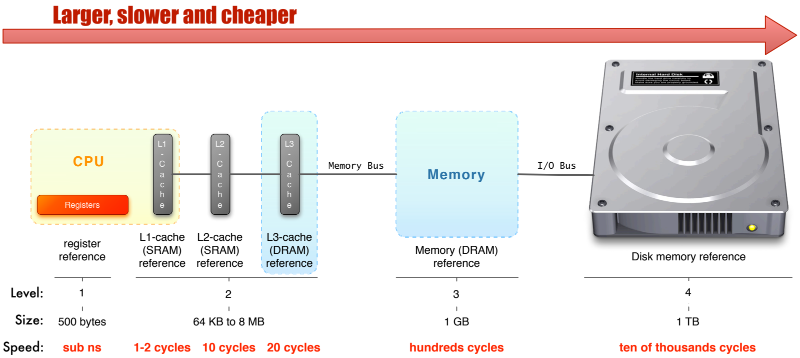
- Always multi-core
- Ex: Intel Core i7-970 (July 2010) $R_{peak} \simeq 100$ GFlops (DP)
  - ↪ 6 cores @ 3.2GHz (32nm, 130W, 1170 millions transistors)

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

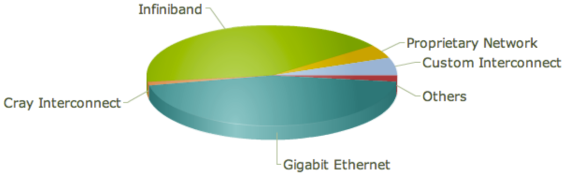


- SSD R/W: 560 MB/s; 85000 IOps **1500 €/TB**
- HDD (SATA @ 7,2 krpm) R/W: 100 MB/s; 190 IOps **150 €/TB**

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

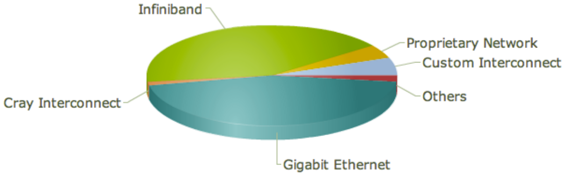
Technology	Effective Bandwidth		Latency
Gigabit Ethernet	1 Gb/s	125 MB/s	$40\mu s$ to $300\mu s$
Myrinet (Myri-10G)	9.6 Gb/s	1.2 GB/s	$2.3\mu s$
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	$4\mu s$ to $5\mu s$
Infiniband QDR	40 Gb/s	5 GB/s	$1.29\mu s$ to $2.6\mu s$
SGI NUMAlink	60 Gb/s	7.5 GB/s	$1\mu s$



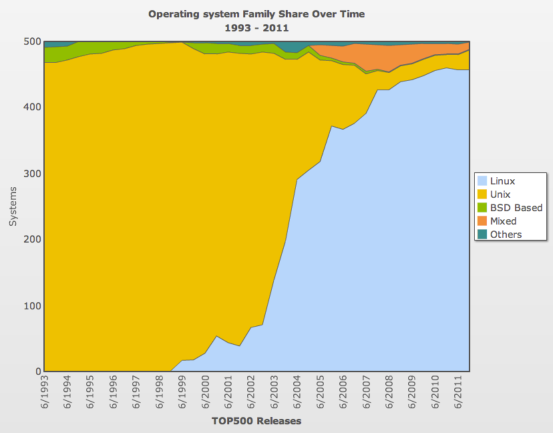
HPC Components: Interconnect

- **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	$40\mu s$ to $300\mu s$
Myrinet (Myri-10G)	9.6 Gb/s	1.2 GB/s	$2.3\mu s$
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	$4\mu s$ to $5\mu s$
Infiniband QDR	40 Gb/s	5 GB/s	$1.29\mu s$ to $2.6\mu s$
SGI NUMAlink	60 Gb/s	7.5 GB/s	$1\mu s$



HPC Components: Operating System



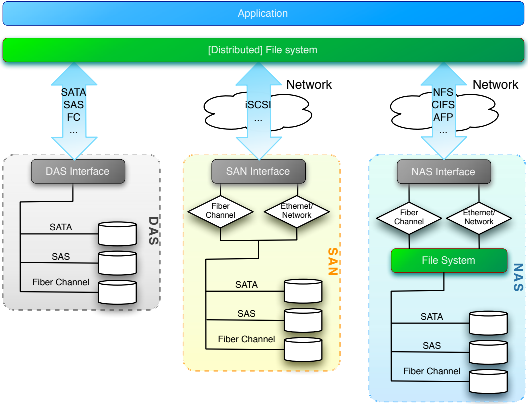
- Mainly Linux-based OS (91.4%) (Top500, Nov 2011)
- ... or Unix based (6%)
- Reasons:
  - ↪ stability
  - ↪ prone to devels

HPC Components: Software Stack

- **Remote connection to the platform**: SSH
- **User SSO**: NIS or OpenLDAP-based
- **Resource management**: job/batch scheduler
  ↪ OAR, PBS, Torque, MOAB Cluster Suite
- **(Automatic) Node Deployment**:
  ↪ FAI (Fully Automatic Installation), Kickstart, Puppet, Chef, Kadeploy etc.
- **Platform Monitoring**: Nagios, Ganglia, Cacti etc.
- (eventually) **Accounting**:
  ↪ `oarnodeaccounting`, Gold allocation manager etc.

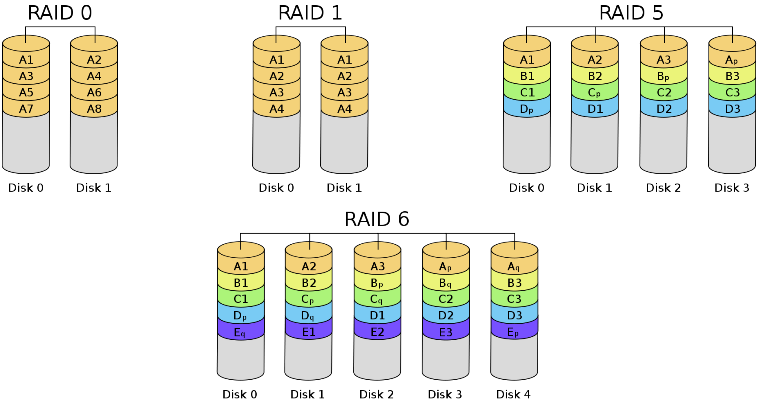
HPC Components: Data Management

Storage architectural classes & I/O layers



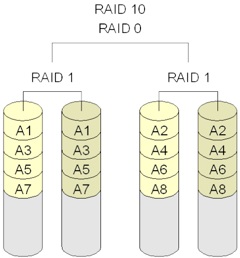
HPC Components: Data Management

RAID standard levels



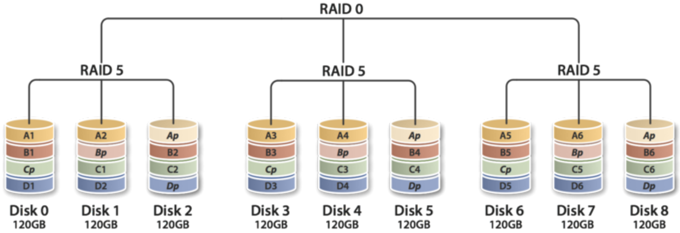
HPC Components: Data Management

RAID combined levels



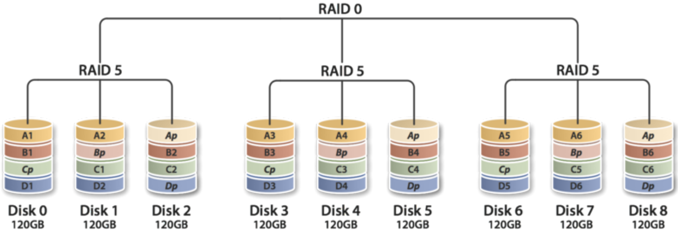
HPC Components: Data Management

RAID combined levels



HPC Components: Data Management

RAID combined levels



- Software vs. **Hardware** RAID management
- RAID Controller card performances differs!
  - ↪ Basic (low cost): 300 MB/s; Advanced (expansive): 1,5 GB/s

HPC Components: Data Management

File Systems

- Logical manner to store, organize, manipulate and access data.
- **Disk file systems**: `FAT32`, `NTFS`, `HFS`, `ext3`, `ext4`, `xfs`...
- **Network file systems**: `NFS`, `SMB`
- **Distributed parallel file systems**: HPC target
  - ↪ data are stripped over multiple servers for high performance.
  - ↪ generally add robust failover and recovery mechanisms
  - ↪ Ex: `Lustre`, `GPFS`, `FhGFS`, `GlusterFS`...

- HPC storage make use of high density **disk enclosures**
  - ↪ includes [redundant] RAID controllers

HPC Components: Data Center

Definition (Data Center)

Facility to house computer systems and associated components

$\hookrightarrow$ 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: 30-40 kW
  - ↪ 'Storage' racks: 15 kW
  - ↪ 'Interconnect' racks: 5 kW

Power Usage Effectiveness

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

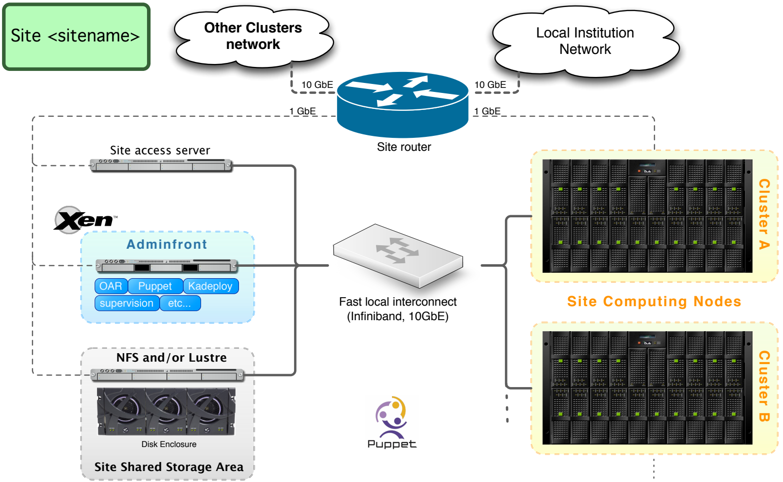
HPC Components: Data Center

HPC Components: Summary

HPC platforms involves:

- A data center / server room carefully designed
- Computing elements: CPU/GPGPU
- Interconnect elements
- Storage elements: HDD/SDD, disk enclosure,
  - ↪ disks are virtually aggregated by RAID/LUNs/FS
- A flexible software stack
- **Above all**: expert system administrators...

Putting it all together...



Summary

Remote Connection by SSH

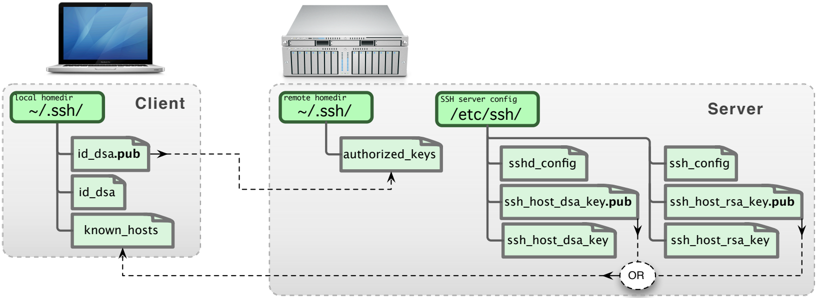
- Establish **Secure** / Encrypted tunnel between $\geq$ 2 hosts
  - $\hookrightarrow$ **Ex:** Your laptop to the access server of the UL HPC
- The tunnel can serve for data transfer
- Based on SSH Key Pairs (public vs private)
  - $\hookrightarrow$ Linux/Mac:

    ```
    $> ssh-keygen -t dsa
    ```

  - $\hookrightarrow$ Windows: PuttyGen + Pageant

UL HPC access

- ***Restricted*** to **SSH** connection **with public key authentication**
  - ↪ on a non-standard port (8022) *limits kiddie script scans/dictionary's attacks*



UL HPC SSH access ~/.ssh/config

```
Host chaos-cluster
    Hostname      access-chaos.uni.lu
Host gaia-cluster
    Hostname      access-gaia.uni.lu
Host *-cluster
    User          login
    Port          8022
    ForwardAgent no

Host myworkstation
    User          localadmin
    Hostname      myworkstation.uni.lux
Host *.ext_ul
    ProxyCommand ssh -q gaia-cluster "nc -q 0 %h %p"
```

```
$> ssh {chaos,gaia}-cluster
$> ssh myworkstation
```

When @ Home:

```
$> ssh myworkstation.ext_ul
```

More on this in **PS1**

- Getting Started

- **Transferring data...**

```
$> rsync -avzu  /devel/myproject chaos-cluster:
(gaia)$> gaia_sync_home *
(chaos)$> chaos_sync_home devel/
```

UL HPC SSH access (Windows)

- Download all the Putty tools
  - ↪ Extract them in an easy-to-find place, such as `C:\Putty`
- Run Pageant and load your SSH private key
- Run Putty (connection type: `SSH`)
  - ↪ Host Name: `access-{chaos,gaia}.uni.lu`
  - ↪ Port: 8022
  - ↪ Saved session: {Chaos,Gaia}
  - ↪ in Category:Connection:Data :
    - ✓ Auto-login username: your login
- **Transferring data...**
  - ↪ Download cwRsync

UL HPC SSH access (Windows)

- Access to an internal workstation (ProxyCommand via Gaia)
  - ↪ Host Name: Workstation IP or hostname
  - ↪ Port: 22
  - ↪ Saved session: MyWorkstation
  - ↪ in Category:Connection:Proxy :
    - ✓ Proxy type: Local
    - ✓ Proxy hostname: access-gaia.uni.lu
    - ✓ Port: 8022
    - ✓ Username: your login
    - ✓ Telnet command:

```
C:\Putty\plink -P %proxyport %user@%proxyhost nc -q 0 %host %port
```

  - ↪ in Category:Connection:Data :
    - ✓ Auto-login username: your login on your workstation

Summary

HPC @ UL http://hpc.uni.lu

Meet the Team

UL HPC Newsletter

Issue 1 – March 2015

Editors: Dr. S. Varrette and Prof. P. Bouvry
Authors: **The UL HPC Management Team** *aka*:
↪ S. Varrette (Manager)
↪ H. Cartiaux, V. Plugaru (FSTC)
↪ S. Diehl, J-F. Le Fillâtre (LCSB)
Date: 2015-03-12
Version: v0.2

Meet the team

Management

Pascal Bouvry is a full professor of the FSTC and the head of the IL-IAS research unit and the DS-CSCE doctoral school. His team (PCOG) is composed of 25 researchers working on Parallel computing and Optimization applied to Cloud Computing and HPC (scheduling, energy-efficiency, security), Ad-Hoc Networks (Vanets simulation and service optimization) and Biology (gene sequencing, regulatory networks, protein folding).

Sébastien Varrette, PhD, is a Research Associate in Prof. Bouvry's team since 2007. Along with Prof. Bouvry, he defined and set up the global HPC initiative of the UL in 2007. In this context, he is managing the sysadmin team that maintain and extend the platform. In parallel, his research work focuses on Distributed Computing Platforms (clusters, grids or clouds), with a particular interest on the security and performance evaluation of distributed or parallel executions.

FSTC

Hyacinthe Cartiaux joined the HPC team in 2011 to set up the Grid'5000 Luxembourg site and has since been involved with all the HPC infrastructure of the UL, and other external services such as the Gforge. His interests cover IT automation and devops techniques, HPC & Grid Computing.

Valentin Plugaru is an HPC engineer part of the HPC team since 2014. Beginning with 2012 he has collaborated with Prof. Bouvry's team on research in Energy Efficiency and Performance Evaluation of HPC/Cloud environments. His general interests span R&D in High Performance Computing, Grid and Cloud Computing.

LCSB

Jean-François Le Fillâtre is a Systems Administrator specializing in storage and HPC infrastructure. He has worked in HPC all over the world, including on Australia's #1 and Canada's #2 supercomputers. His main interests are pushing systems as fast as they can go and finding new ways of traveling to places he hasn't seen yet.

Sarah Diehl is a bioinformatician and joined the LCSB BioCore in 2015 as an HPC systems administrator. Her goal is to bridge the gap between researchers and IT specialists. She is experienced in data management, next-generation sequencing analysis and development of analysis pipelines.

UNIVERSITÉ DU LUXEMBOURG

8

http://hpc.uni.lu

UL HPC platforms at a glance (2015)

- **2 geographical sites, 3 server rooms**
- **4 clusters**: `chaos+gaia`, `granduc`, `nyx`.
  - ↪ **400 nodes, 4280 cores, 49.872 TFlops**
  - ↪ incl. **18** dual [GP]GPU nodes
  - ↪ **3364.4 TB (raw) shared storage**
    - ✓ incl. 1.7 PB for backup
    - ✓ **+1 PB** (EMC Isilon) SIU/LCSB/HPC
- **5 sysadmins** `hpc-sysadmins@uni.lu`
- **6,340,316€** **(Cumul. HW Investment)** since 2007
  - ↪ Hardware acquisition only
  - ↪ 4,077,913€ (excluding server rooms)
- **Open-Source** software stack
  - ↪ SSH, LDAP, OAR, Puppet, Modules...

HPC server rooms

- **2009** CS.43 (Kirchberg campus) 14 racks, 100 m^2, $\simeq$ 800,000€



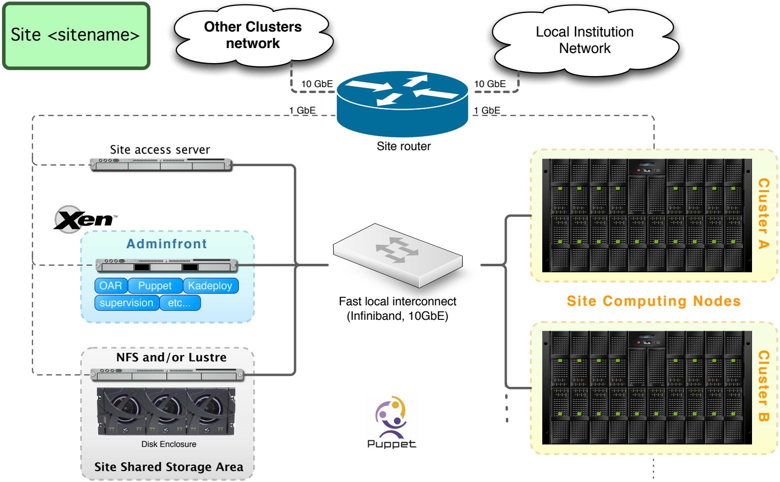
- **2011** LCSB 6th floor (Belval) 14 racks, 112 m^2, $\simeq$ 1,100,000€


UL HPC Computing Nodes

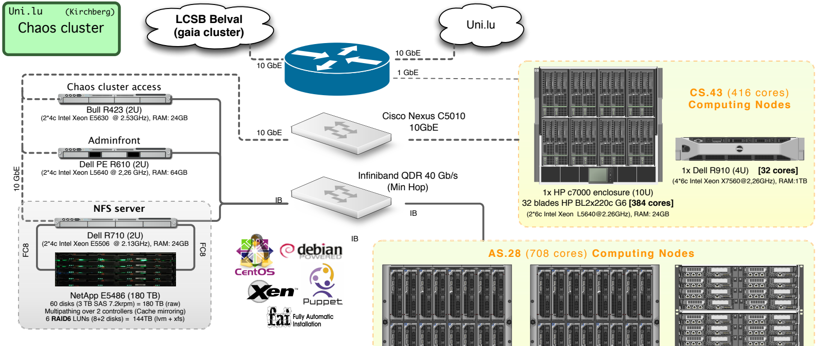
	Date	Vendor	Proc. Description		#N	#C	R_{peak}
chaos	2010	HP	Intel Xeon L5640@2.26GHz	$2 \times$ 6C,24GB	32	384	3.472 TFlops
	2011	Dell	Intel Xeon L5640@2.26GHz	$2 \times$ 6C,24GB	16	192	1.736 TFlops
	2012	Dell	Intel Xeon X7560@2,26GHz	$4 \times$ 6C, 1TB	1	32	0.289 TFlops
	2012	Dell	Intel Xeon E5-2660@2.2GHz	$2 \times$ 8C,32GB	16	256	4.506 TFlops
	2012	HP	Intel Xeon E5-2660@2.2GHz	$2 \times$ 8C,32GB	16	256	4.506 TFlops
				chaos **TOTAL:**	**81**	**1120**	**14.49 TFlops**
gaia	2011	Bull	Intel Xeon L5640@2.26GHz	$2 \times$ 6C,48GB	72	864	7.811 TFlops
	2012	Dell	Intel Xeon E5-4640@2.4GHz	$4 \times$ 8C, 1TB	1	32	0.307 TFlops
	2012	Bull	Intel Xeon E7-4850@2GHz	$16 \times$ 10C,1TB	1	160	1.280 TFLops
	2013	Dell	Intel Xeon E5-2660@2.2GHz	$2 \times$ 8C,64GB	5	80	1.408 TFlops
	2013	Bull	Intel Xeon X5670@2.93GHz	$2 \times 6C$,48GB	40	480	5.626 TFlops
	2013	Bull	Intel Xeon X5675@3.07GHz	$2 \times 6C$,48GB	32	384	4.746 TFlops
				gaia **TOTAL:**	**183**	**2400**	**30.382 TFlops**
g5k	2008	Dell	Intel Xeon L5335@2GHz	$2 \times$ 4C,16GB	22	176	1.408 TFlops
	2012	Dell	Intel Xeon E5-2630L@2GHz	$2 \times$ 6C,24GB	16	192	1.536 TFlops
				granduc/petitprince **TOTAL:**	**38**	**368**	**4.48 TFlops**
Testing cluster:							
nyx	2012	Dell	Intel Xeon E5-2420@1.9GHz	$1 \times$ 6C,32GB	2	12	0.091 TFlops
viridis	2013	Viridis	ARM A9 Cortex@1.1GHz	$1 \times$ 4C,4GB	96	384	0.422 TFlops
				nyx/viridis **TOTAL:**	**98**	**392**	**0.52 TFlops**

TOTAL: 400 nodes, 4280 cores, 49.872 TFlops

UL HPC: General cluster organization



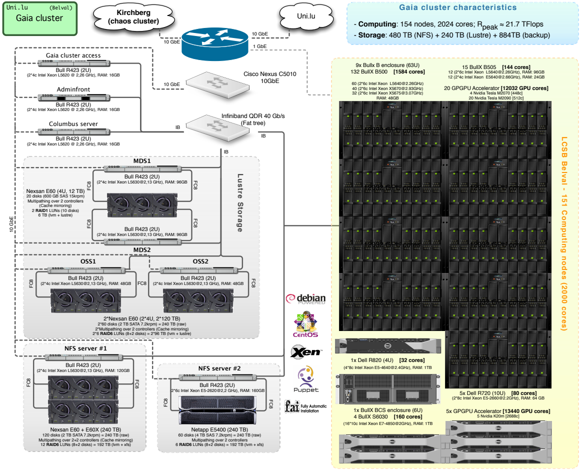
Ex: The chaos cluster



Chaos cluster characteristics

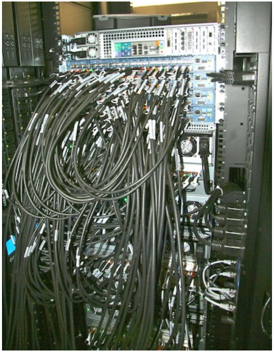
- **Computing**: 81 nodes, 1124 cores; $R_{peak} \approx$ 14.508 TFlops
- **Storage**: 180 TB (NFS) + 180TB (NFS, backup)

Ex: The gaia cluster



Ex: Some racks of the gaia cluster





UL HPC Software Stack

- **Operating System:** Linux Debian (CentOS on storage servers)
- **Remote connection to the platform**: SSH
- **User SSO**: OpenLDAP-based
- **Resource management**: job/batch scheduler: OAR
- **(Automatic) Computing Node Deployment**:
  - ↪ FAI (Fully Automatic Installation) (chaos,gaia,nyx only)
  - ↪ Puppet
  - ↪ Kadeploy (granduc,petitprince/Grid5000 only)
- **Platform Monitoring**: OAR Monika, OAR Drawgantt, Ganglia, Nagios, Puppet Dashboard etc.
- **Commercial Softwares**:
  - ↪ Intel Cluster Studio XE, TotalView, Allinea DDT, Stata etc.

HPC in the Grande region and Around

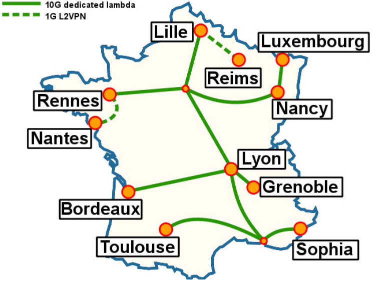
Country	Name/Institute	#Cores	TFlops R_{peak}	TB Storage	FTEs Manpower
Luxembourg	UL	4280	49.872	3364.4	2
	CRP GL	800	6.21	144	1.5
France	TGCC Curie, CEA	77184	1667.2	5000	n/a
	LORIA, Nancy	3724	29.79	82	5.05
	ROMEO, UCR, Reims	564	4.128	15	2
Germany	Juqueen, Juelich	393216	5033.2	448	n/a
	MPI, RZG	2556	14.1	n/a	5
	URZ, (bwGrid),Heidelberg	1140	10.125	32	9
Belgium	UGent, VCS	4320	54.541	82	n/a
	CECI, UMons/UCL	2576	25.108	156	> 4
UK	Darwin, Cambridge Univ	9728	202.3	20	n/a
	Legion, UCLondon	5632	45.056	192	6
Spain	MareNostrum, BCS	33664	700.2	1900	14

The case of Grid'5000

http://www.grid5000.fr

Grid'5000

- 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**

Grid'5000 Usage

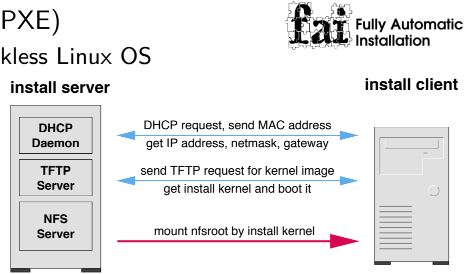
- Out of scope for this talk
- **Grid'5000 school 2015**
  - ↪ Organized in France, typically in June
  - ↪ See G5K School 2014 website
- General information: **https://hpc.uni.lu/g5k**
- Grid'5000 website and documentation: **https://www.grid5000.fr**

Computing nodes Management

Node deployment by FAI

http://fai-project.org/

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



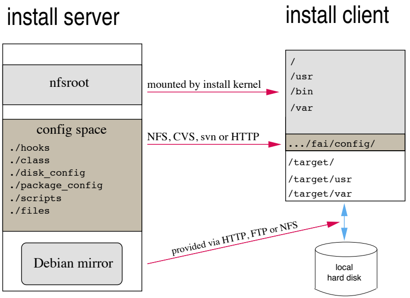
Computing nodes Management

Node deployment by FAI

http://fai-project.org/

Fully Automatic Installation

- 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/

Fully Automatic Installation

- 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/

Fully Automatic Installation

- 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

Platform Management: Puppet

Server/Service configuration by Puppet

http://puppetlabs.com

- *Configuration management made easy*
- Automates service deployment / sysadmin tasks
- 3 components:
  - ↪ declarative language (manifests)
  - ↪ client/server model
  - ↪ Git-based workflow + Rakefile + Capistrano
- Hierarchical PKI

Platform Management: Puppet

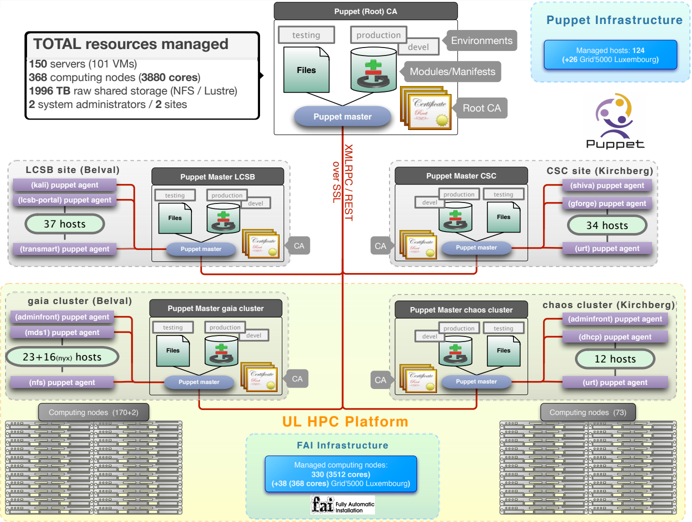
Server/Service configuration by Puppet

http://puppetlabs.com

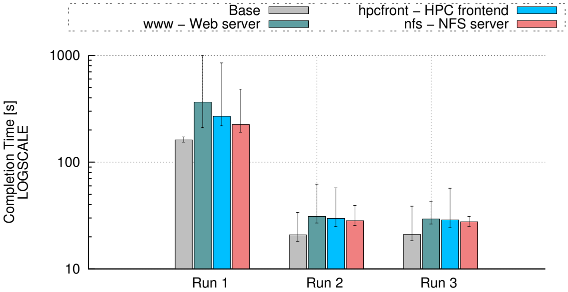
- *Configuration management made easy*
- Automates service deployment / sysadmin tasks
- 3 components:
  - ↪ declarative language (manifests)
  - ↪ client/server model
  - ↪ Git-based workflow + Rakefile + Capistrano
- Hierarchical PKI

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

Platform Management: Puppet



Puppet Performances



- between 161s and 364s to completely **bootstrap** a virgin node
  ↪ between 20s and 31s to later check/correct the config

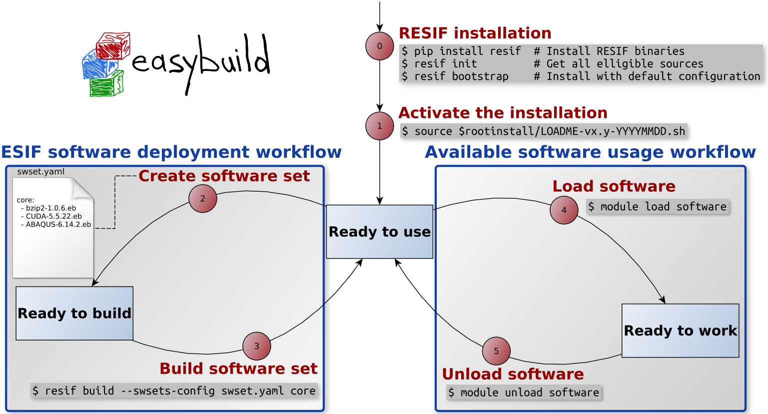
Now proposed as an IT service to external consumers

Software/Modules Management

- RESIF: Revolutionary EasyBuild-based Software Installation Framework
  - ↪ Automatic Management of Environment Modules deployment
  - ↪ Fully automates software builds and supports all available toolchains
  - ↪ Clean (hierarchical) modules layout to facilitate its usage
  - ↪ "Easy to use"
- Cf Practical Session

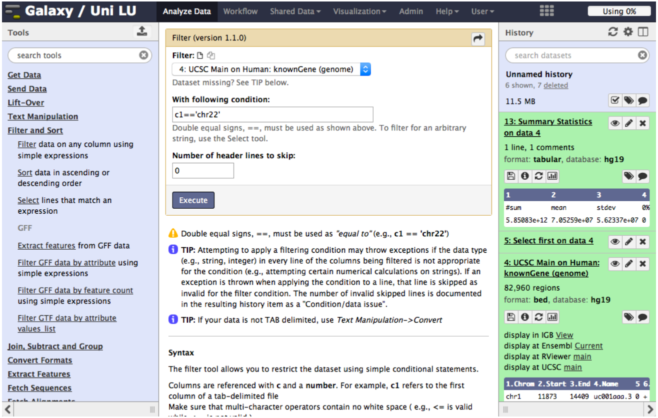
Software/Modules Management

RESIF: Revolutionary EasyBuild-based Software Installation Framework



BIO Workflow Management

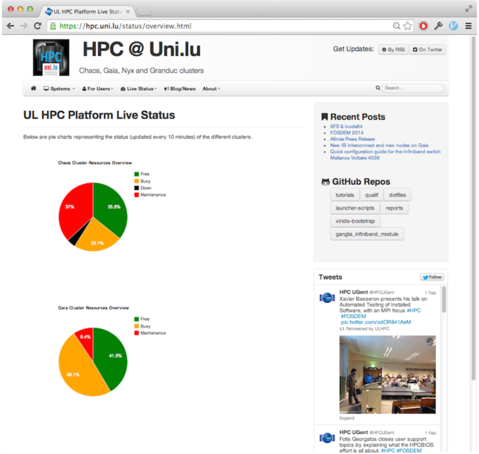
- Galaxy Portal galaxy-server.uni.lu
  ↪ web-based platform for data intensive biomedical research.



Platform Monitoring

General Live Status

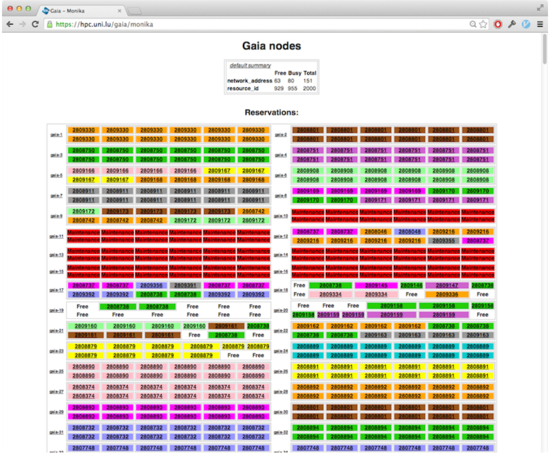
http://hpc.uni.lu/status/overview.html



Platform Monitoring

Monika

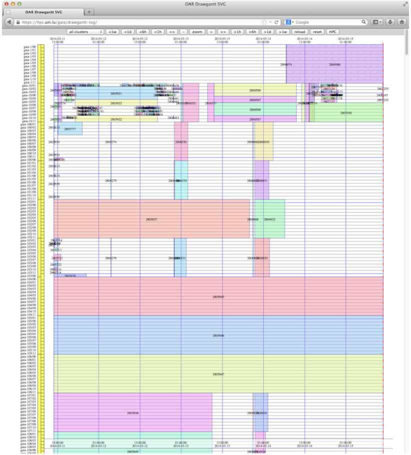
http://hpc.uni.lu/{chaos,gaia,g5k}/monika



Platform Monitoring

Drawgantt

http://hpc.uni.lu/{chaos,gaia,g5k}/drawgantt



Platform Monitoring

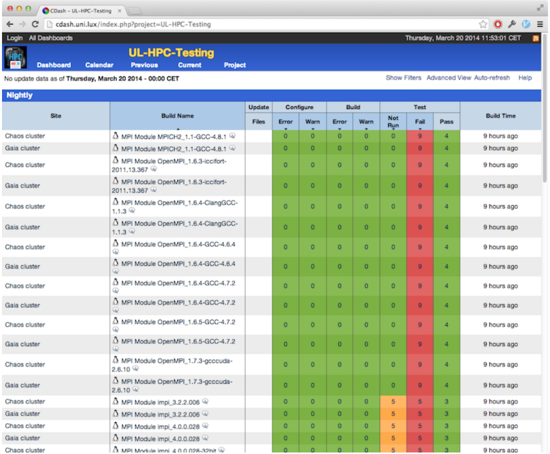
Ganglia

http://hpc.uni.lu/{chaos,gaia,g5k}/ganglia

Platform Monitoring

CDash

http://cdash.uni.lu/



CDash - UL-HPC-Testing

cdash.uni.lux/index.php?project=UL-HPC-Testing

Login All Dashboards Thursday, March 20 2014 11:53:01 CET

UL-HPC-Testing

Dashboard Calendar Previous Current Project

No update data as of **Thursday, March 20 2014 - 00:00 CET** Show Filters Advanced View Auto-refresh Help

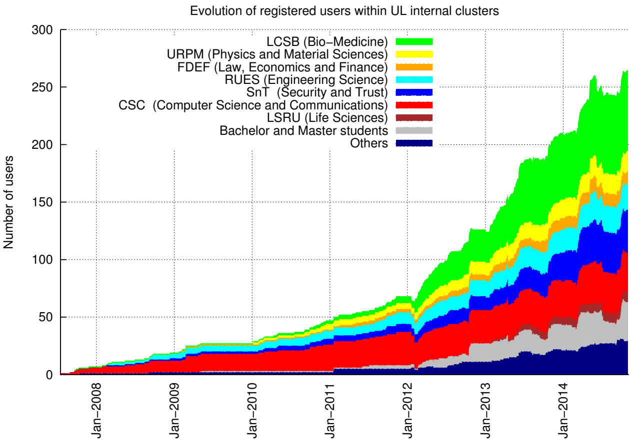
Nightly

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

Key numbers

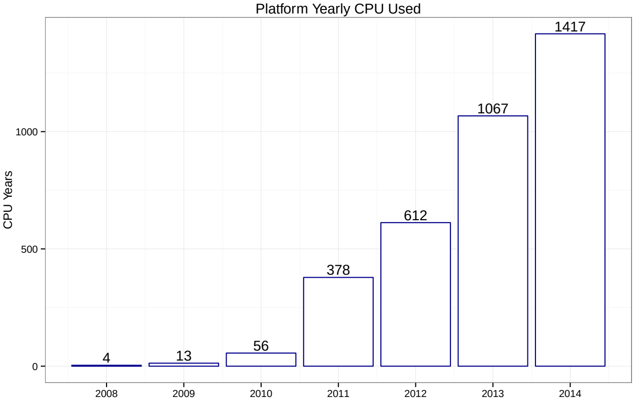
- **400 nodes, 4280 cores, 49.872 TFlops**
  - ↪ Mostly Intel-based architecture, multi vendors (Bull, HP, Dell, Delta, SGi)
- **3364.4 TB (raw) shared storage**
  - ↪ Based on NetApp / NexSAN / Certon disk enclosures
  - ↪ **Homedirs** / **Projects**: NFS → GPFS, OneFS
  - ↪ **Scratch** Lustre 720 TB
  - ↪ **Backup**: NFS, GlusterFS, OneFS 1.7 PB
- **6,340,316€** **(Cumul. HW Investment)** since 2007
- 281 registered users

Registered Users

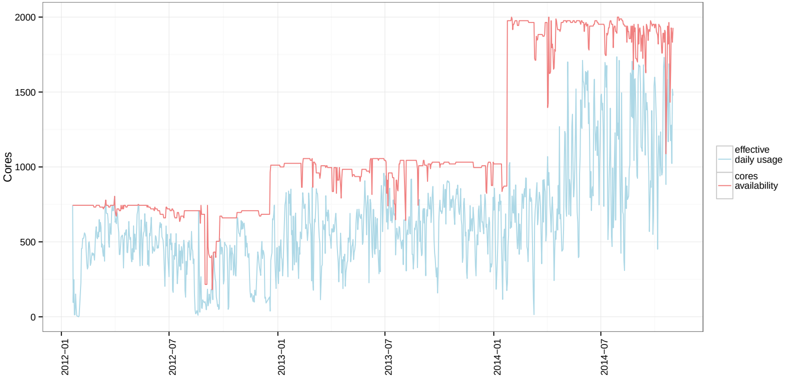


CPU-year usage since 2008

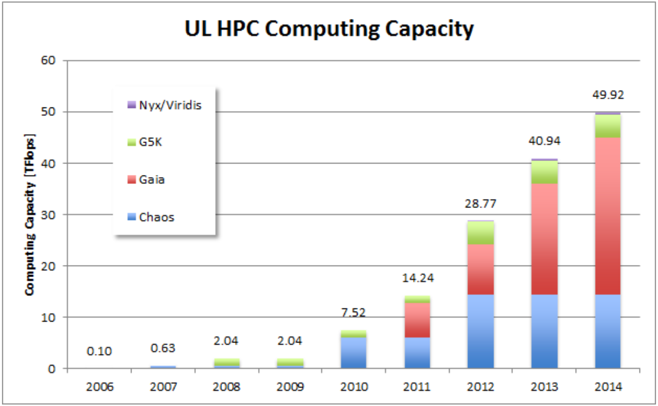
- **CPU-hour**: *work* done by a CPU in one hour of wall clock time



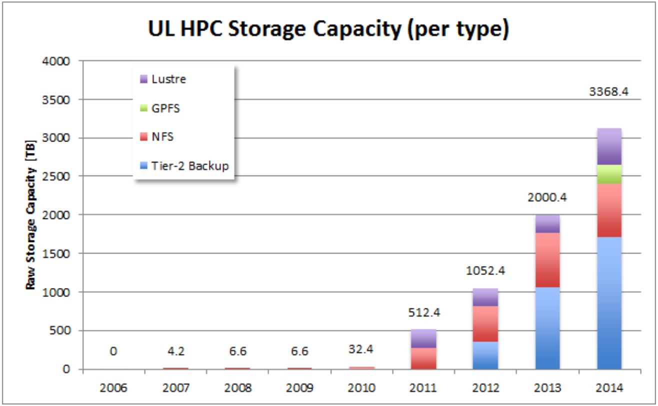
A Year on Gaia...



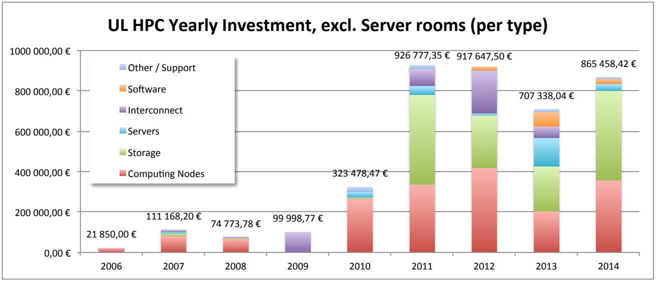
Chronological Statistics



Chronological Statistics



Chronological Statistics



Research Domains and Usage

Research Domains (Among the 288 registered users)

- **Research Areas** that currently benefit from UL HPC platforms:
  - ↪ Security ([Ad-Hoc] Network, FT, Grid, Cloud etc.)
  - ↪ Mechanical Engineering
  - ↪ Physics, Geo-Physics
  - ↪ [Multi-Objective] Optimization [Robust] Task Scheduling etc.
  - ↪ Cryptology
  - ↪ Economy
  - ↪ Life Sciences

2015 Milestones

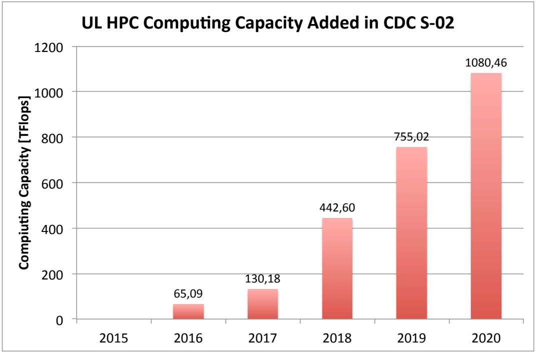
- cf Newsletter: OS/System upgrade
- Storage / Portal consolidation
  - ↪ No way to further extend the HW equipment
  - ↪ QoS, Establish UL as national HPC Center of Excellence

Coming Soon (~~2015~~ 2016)

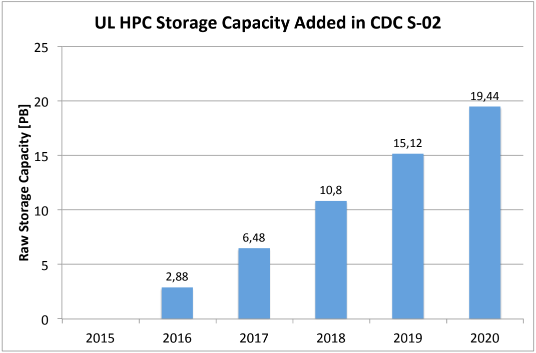
- Belval *Centre De Calcul* (CDC)
  - ↪ 5 new server rooms (3 storage, 2 HPC)
  - ↪ Pending discussions with Fond Belval to re-justify everything

- **Obj.**: Prepare 2 rooms (1 HPC, 1 Storage) by 2020
  - ↪ Budget: $\simeq$ 4M€ per year

UL HPC Planning 2015-2020



UL HPC Planning 2015-2020



UL HPC Planning 2015-2020

- Funding CDC is **mandatory**
  - ↪ Data Center providers (EBRC, LuxConnect etc.) not adapted
    - ✓ **do not have** HPC-ready cabinets (80 kW/rack)
    - ✓ thus proposed renting price is prohibitive
  - ↪ Cloud platforms (Amazon etc.) only able to absorb part of the needs
- **Computational Science** initiative
  - ↪ part of the Digital Strategy for the UL
  - ↪ pending PRIDE call / Doctoral School etc.
- **National HPC initiative**
  - ↪ discussion in progress (MECE, MESR)
  - ↪ **Obj**: national HPC Center of Excellence (CoE)

Cost Model

- UL HPC Platform funding **should** evolve
  - ↪ transition from a free service model to a mixed model
    - ✓ with paying and non-paying tiers
  - ↪ key for providing HPC services at the national level
- You shall also budget your usage upon new project proposal

Cost Model

- UL HPC Platform funding **should** evolve
  - ↪ transition from a free service model to a mixed model
    - ✓ with paying and non-paying tiers
  - ↪ key for providing HPC services at the national level
- You shall also budget your usage upon new project proposal

Cost policy

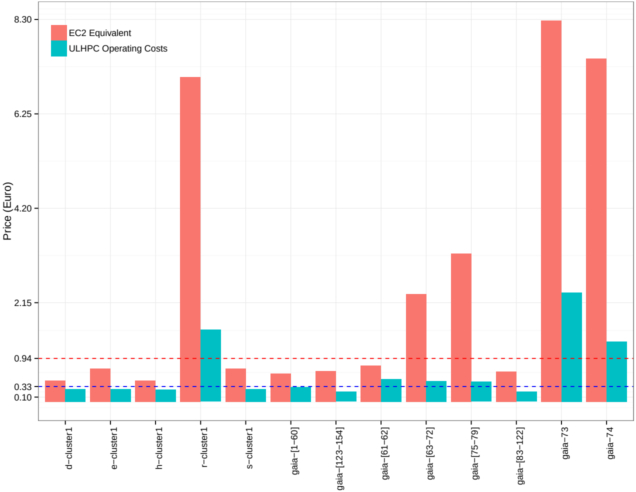
- no charge to the actors of the public research sector
  - ↪ **only** for internal research projects
  - ↪ UL Research Units & ICs, LIST, LISER, LIH
- the same actors **will** be charged for externally founded projects
  - ↪ FNR, European projects, projects with industry

Cost Model

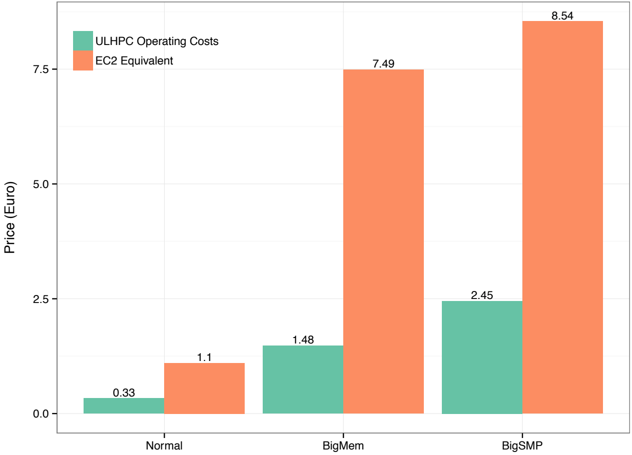
- Pricing units are in the form of usage credits
  - ↪ under a monthly accounting period.
- Two types of credits:
  1. 1 computing credit of class "X" = 1 CPU core for 1 hour
     - ✓ on a resource class "X"
  2. 1 storage credit = 1 TB of storage for 1 month.
     - ✓ 3 storage credits (thus maximum 3TB) for free (each month);
     - ✓ Additional credits: 1000€

Class	Description	Credit Price
normal	Regular HPC resource	0,33 €
bigmem	Regular HPC resource with huge RAM ($\geq$ 1024 GB)	1,48 €
bigsmp	SMP node ($\geq$ 16 sockets) with a huge RAM ($\geq$ 1024 GB)	2,45 €

Computing Credits Prices



Computing Credits Prices



Summary

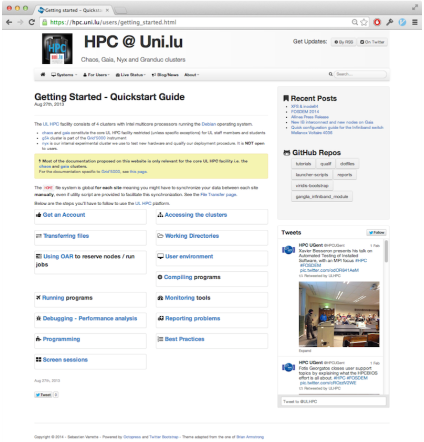
General Consideration

- The platform is ***restricted*** to UL members and is ***shared***
- Everyone should be civic-minded.
  - ↪ Just **avoid** the following behavior: (or you'll be banned)
    "My work is the most important: I use all the resources for 1 month"
  - ↪ regularly clean your homedir from useless files
- Plan large scale experiments during night-time or week-ends
  - ↪ try not to use more than 40 computing cores during working day
  - ↪ ... or use the '`besteffort`' queue

User Doc

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

... aka the **rtfm** paradigm



User Account

Get an account: https://hpc.uni.lu/get_an_account

With your account, you'll get:

- Access to the UL HPC wiki http://hpc.uni.lu/
- Access to the UL HPC bug tracker http://hpc-tracker.uni.lu/
- Subscribed to the mailing lists `hpc-{users,platform}@uni.lu`
  - ↪ raise questions and concerns. Help us to make it a community!
  - ↪ notification of platform maintenance on hpc-platform@uni.lu
- A nice way to reach workstation in the internal UL network (`ProxyCommand`)

Typical Workflow

1. Connect to the frontend of a site/cluster `ssh`
2. *(eventually)* synchronize you code `scp/rsync/svn/git`
3. *(eventually)* Reserve a few interactive resources `oarsub -I`
   - ↪ *(eventually)* Configure the resources `kadeploy`
   - ↪ *(eventually)* Prepare your experiments `gcc/icc/mpicc/javac/...`
   - ↪ Test your experiment on small size problem `mpirun/java/bash....`
   - ↪ Free the resources
4. Reserve some resources `oarsub`
5. Run your experiment via a launcher script `bash/python/perl/ruby...`
6. Grab the results `scp/rsync`
7. Free the resources

Typical Workflow

1. Connect to the frontend of a site/cluster `ssh`
2. *(eventually)* synchronize you code `scp/rsync/svn/git`
3. *(eventually)* Reserve a few interactive resources `oarsub -I`
   - ↪ *(eventually)* Configure the resources `kadeploy`
   - ↪ *(eventually)* Prepare your experiments `gcc/icc/mpicc/javac/...`
   - ↪ Test your experiment on small size problem `mpirun/java/bash....`
   - ↪ Free the resources
4. Reserve some resources `oarsub`
5. Run your experiment via a launcher script `bash/python/perl/ruby...`
6. Grab the results `scp/rsync`
7. Free the resources

See Next Talk

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

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

OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15: Total: 6 cores

```
(frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15
```

OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15: Total: 6 cores

```
(frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15
```

- 4 cores on a GPU node for 8 hours Total: 4 cores

```
(frontend)$> oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"
```

OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15: Total: 6 cores

```
(frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15
```

- 4 cores on a GPU node for 8 hours Total: 4 cores

```
(frontend)$> oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"
```

- 2 nodes among the `h-cluster1-*` nodes (Chaos only) Total: 24 cores

```
(frontend)$> oarsub -I -l nodes=2 -p "nodeclass='h'"
```

OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15: Total: 6 cores

```
(frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15
```

- 4 cores on a GPU node for 8 hours Total: 4 cores

```
(frontend)$> oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"
```

- 2 nodes among the `h-cluster1-*` nodes (Chaos only) Total: 24 cores

```
(frontend)$> oarsub -I -l nodes=2 -p "nodeclass='h'"
```

- 4 cores on 2 GPU nodes + 20 cores on other nodes Total: 28 cores

```
$> oarsub -I -l "{gpu='YES'}/nodes=2/core=4+{gpu='NO'}/core=20"
```

OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15: Total: 6 cores

```
(frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15
```

- 4 cores on a GPU node for 8 hours Total: 4 cores

```
(frontend)$> oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"
```

- 2 nodes among the `h-cluster1-*` nodes (Chaos only) Total: 24 cores

```
(frontend)$> oarsub -I -l nodes=2 -p "nodeclass='h'"
```

- 4 cores on 2 GPU nodes + 20 cores on other nodes Total: 28 cores

```
$> oarsub -I -l "{gpu='YES'}/nodes=2/core=4+{gpu='NO'}/core=20"
```

- A full big SMP node Total: 160 cores on gaia-74

```
$> oarsub -t bigsmp -I l node=1
```

Some other useful features of OAR

Connect to a running job

```
(frontend)$> oarsub -C JobID
```

Status of a jobs

```
(frontend)$> oarstat -state -j JobID
```

Get info on the nodes

```
(frontend)$> oarnodes
(frontend)$> oarnodes -l
(frontend)$> oarnodes -s
```

Cancel a job

```
(frontend)$> oardel JobID
```

View the job

```
(frontend)$> oarstat
(frontend)$> oarstat -f -j JobID
```

Run a best-effort job

```
(frontend)$> oarsub -t besteffort ...
```

Designing efficient OAR job launchers

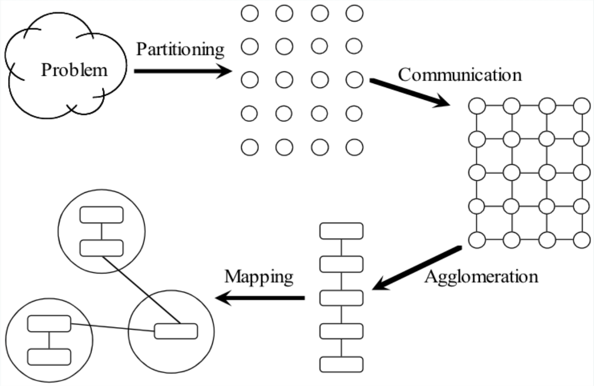
Resources/Example

https://github.com/ULHPC/launcher-scripts

- UL HPC grant access to **parallel computing** resources
  - ↪ ideally: OpenMP/MPI/CUDA/OpenCL jobs
  - ↪ if serial jobs/tasks: run them efficiently
- Avoid to submit purely serial jobs to the OAR queue a
  - ↪ waste the computational power (11 out of 12 cores on `gaia`).
  - ↪ use whole nodes by running at least 12 serial runs at once
- **Key**: understand difference between **Task** and **OAR job**

Designing efficient OAR job launchers

- Methodical Design of Parallel Programs



[Foster96] I. Foster, *Designing and Building Parallel Programs*. Addison Wesley, 1996.
Available at: `http://www.mcs.anl.gov/dbpp`

Serial tasks: BAD and NAIVE approach

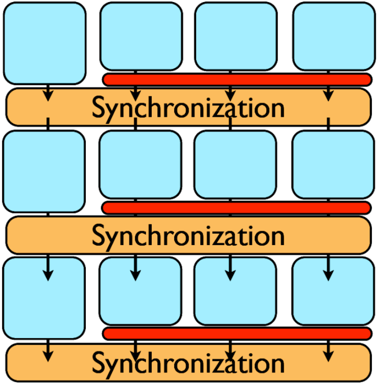
```
# Example 1: run in sequence $TASK 1...$TASK $NB_TASKS
for i in `seq 1 $NB_TASKS`; do
    $TASK $i
done
# Example 2: For each line of $ARG_TASK_FILE, run in sequence
#   $TASK <line1>... $TASK <lastline>
while read line; do
    $TASK $line
done < $ARG_TASK_FILE
```

Serial tasks: A better approach (fork & wait)

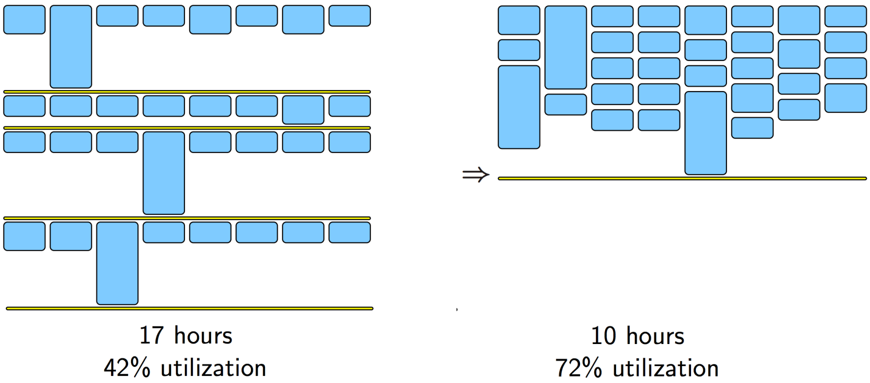
```
# Example 1: run in sequence $TASK 1...$TASK $NB_TASKS
for i in `seq 1 $NB_TASKS`; do
    $TASK $i &
done
wait

# Example 2: For each line of $ARG_TASK_FILE, run in sequence
#   $TASK <line1>... $TASK <lastline>
while read line; do
    $TASK $line &
done < $ARG_TASK_FILE
fi
wait
```

Serial tasks: A better approach (fork & wait)

Different runs may not take the same time: **load imbalance**.



Serial tasks with GNU Parallel



Serial tasks with GNU Parallel

```
### Example 1: run in sequence $TASK 1...$TASK $NB_TASKS
# On a single node
seq $NB_TASKS | parallel -u -j 12 $TASK {}

# on many nodes
seq $NB_TASKS | parallel -tag -u -j 4 \\
   -sshloginfile ${GP_SSHLOGINFILE}.task $TASK {}

### Example 2: For each line of $ARG_TASK_FILE, run in parallel
#   $TASK <line1>... $TASK <lastline>
# On a single node
cat $ARG_TASK_FILE | parallel -u -j 12 -colsep ' ' $TASK {}

# on many nodes
cat $ARG_TASK_FILE | parallel -tag -u -j 4 \\
   -sshloginfile ${GP_SSHLOGINFILE}.task -colsep ' ' $TASK {}
```

Designing efficient OAR job launcher

- More information:
  - ↪ **PS 2A**: HPC workflow with sequential jobs (test cases on GROMACS, Python and Java)
  - ↪ **PS 3A**: HPC workflow with Parallel/Distributed jobs: application on MPI software (test cases on OSU/HPL)
  - ↪ etc.

Reporting a problem

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

- **First checks**
  1. My issue is probably documented — User doc
  2. An event is on-going — cf mail from hpc-platform@uni.lu
  3. check the state of your nodes — Ganglia (especially memory usage)

See next Talk

- **ONLY NOW** consider the following depending on the severity
  - ↪ Open an new issue on http://hpc-tracker.uni.lu (prefered)
  - ↪ Mail (really all of) 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**

Reporting your usage of the platform

https://hpc.uni.lu/users/AUP.html

- In your scientific publications:
  - ↪ **acknowledge** your usage of the UL HPC platform
  - ↪ cf Acceptable Use Policy

```
Acknowledgment: Experiments presented in this paper were carried out using the
HPC facilities of University of Luxembourg~\cite{VBCG_HPCS14}
{\small -- see \url{http://hpc.uni.lu}}.
```

```
@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},
}
```

Bind your OrbiLu entry with UL HPC!!!

Thank you for your attention...

Questions?

Sébastien Varrette, PhD
mail: sebastien.varrette@uni.lu
Office E-007
Campus Kirchberg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg

UL HPC Management Team
mail: hpc-sysadmins@uni.lu