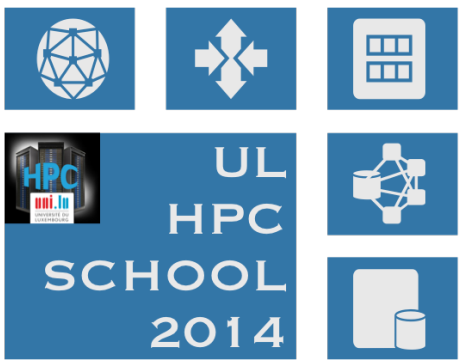


HPC platforms @ UL

Overview (as of 2014) and Usage

http://hpc.uni.lu

S. Varrette, H. Cartiaux

University of Luxembourg, Luxembourg

Summary

1. **Overview of the Main HPC Components**

2. **The UL HPC platform**
   - Overview
   - Platform Deployment with FAI and Puppet
   - Monitoring
   - Statistics & Milestones

3. **UL HPC in Practice: Toward an [Efficient] Win-Win Usage**
   - General considerations
   - SSH Access
   - The OAR Batch Scheduler
   - Available Software and Environment Modules

4. **Last Challenges**
   - Memory bottleneck
   - Effective Storage and Memory Management
   - Reporting problems and Promoting the Platform

Summary

1 Overview of the Main HPC Components

2 The UL HPC platform
Overview
Platform Deployment with FAI and Puppet
Monitoring
Statistics & Milestones

3 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
General considerations
SSH Access
The OAR Batch Scheduler
Available Software and Environment Modules

4 Last Challenges
Memory bottleneck
Effective Storage and Memory Management
Reporting problems and Promoting the Platform

HPC Components: [GP]CPU

CPU

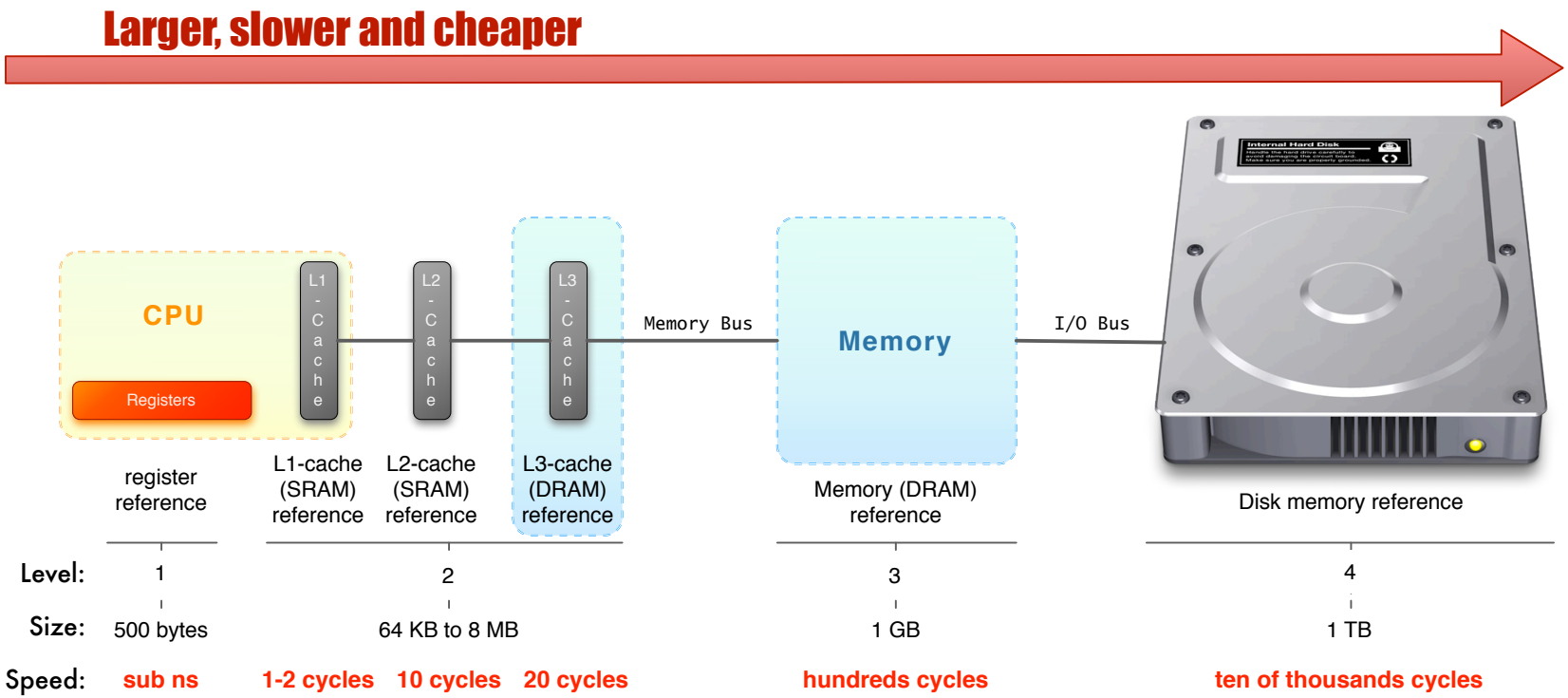
- Always multi-core
- Ex: Intel Core i7-970 (July 2010) $R_{peak} \simeq 100$ GFlops (DP)
  - ↪ 6 cores @ 3.2GHz (32nm, 130W, 1170 millions transistors)

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

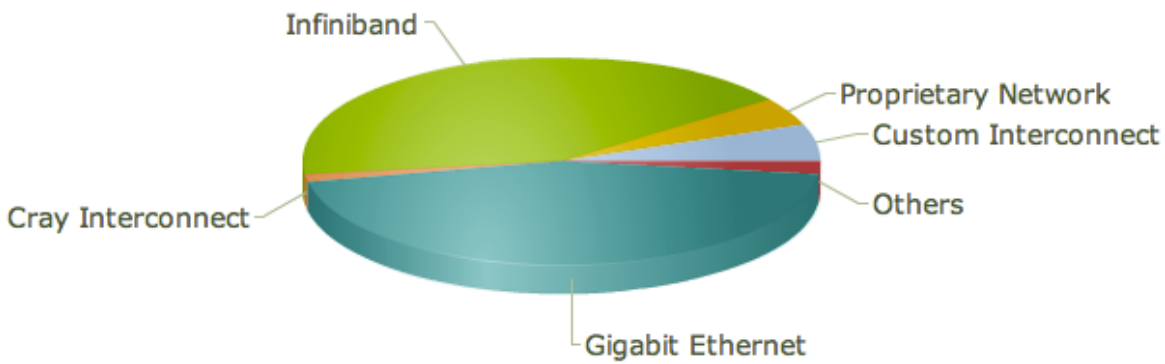


- SSD R/W: 560 MB/s; 85000 IOps **1500 €/TB**
- HDD (SATA @ 7,2 krpm) R/W: 100 MB/s; 190 IOps **150 €/TB**

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

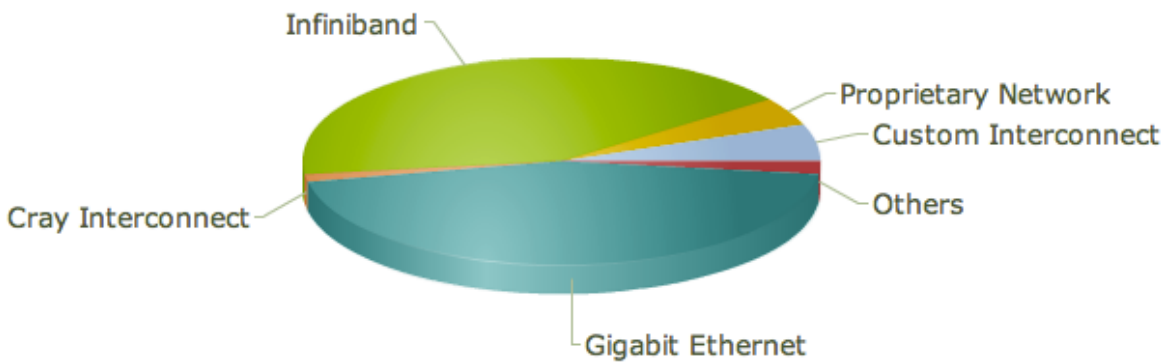
Technology	Effective Bandwidth		Latency
Gigabit Ethernet	1 Gb/s	125 MB/s	40μs to 300μs
Myrinet (Myri-10G)	9.6 Gb/s	1.2 GB/s	2.3μs
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	4μs to 5μs
Infiniband QDR	40 Gb/s	5 GB/s	1.29μs to 2.6μs
SGI NUMAlink	60 Gb/s	7.5 GB/s	1μs



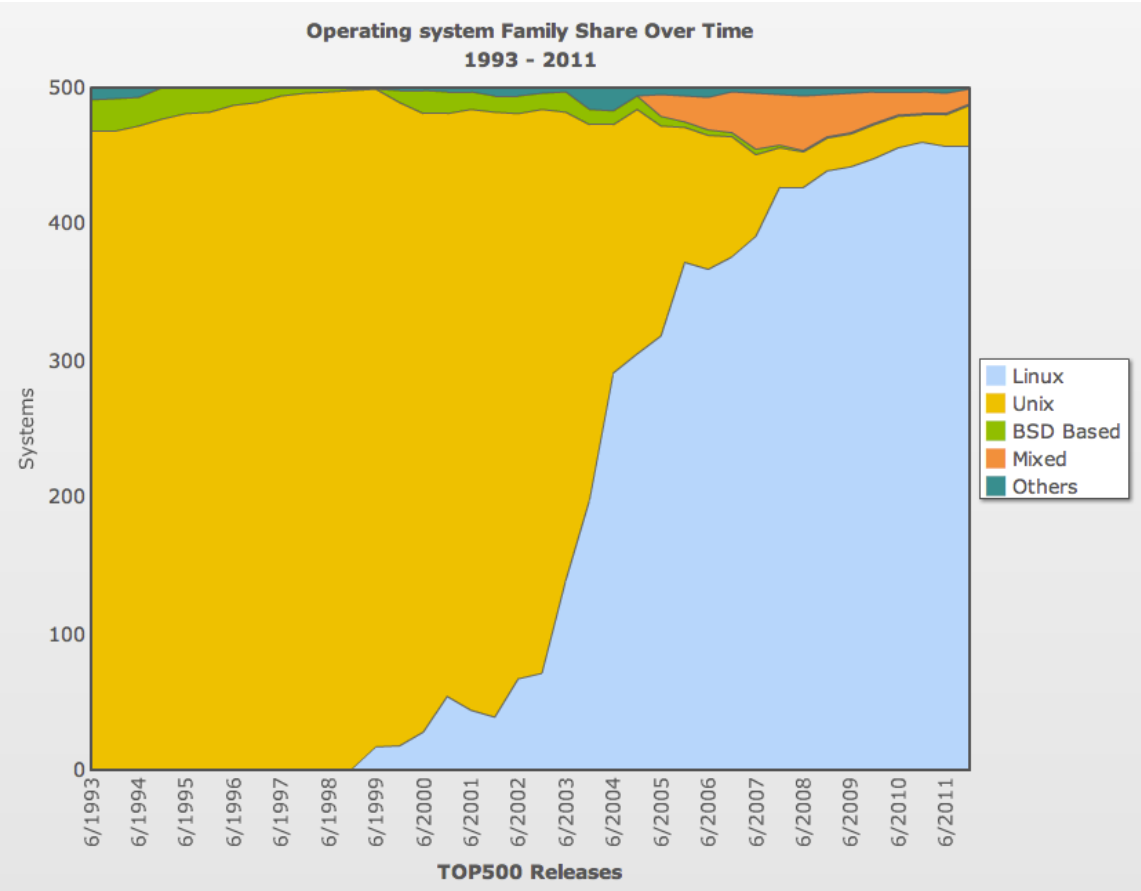
HPC Components: Interconnect

- **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
Myrinet (Myri-10G)	9.6 Gb/s	1.2 GB/s	2.3μs
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
SGI NUMAlink	60 Gb/s	7.5 GB/s	1μs



HPC Components: Operating System



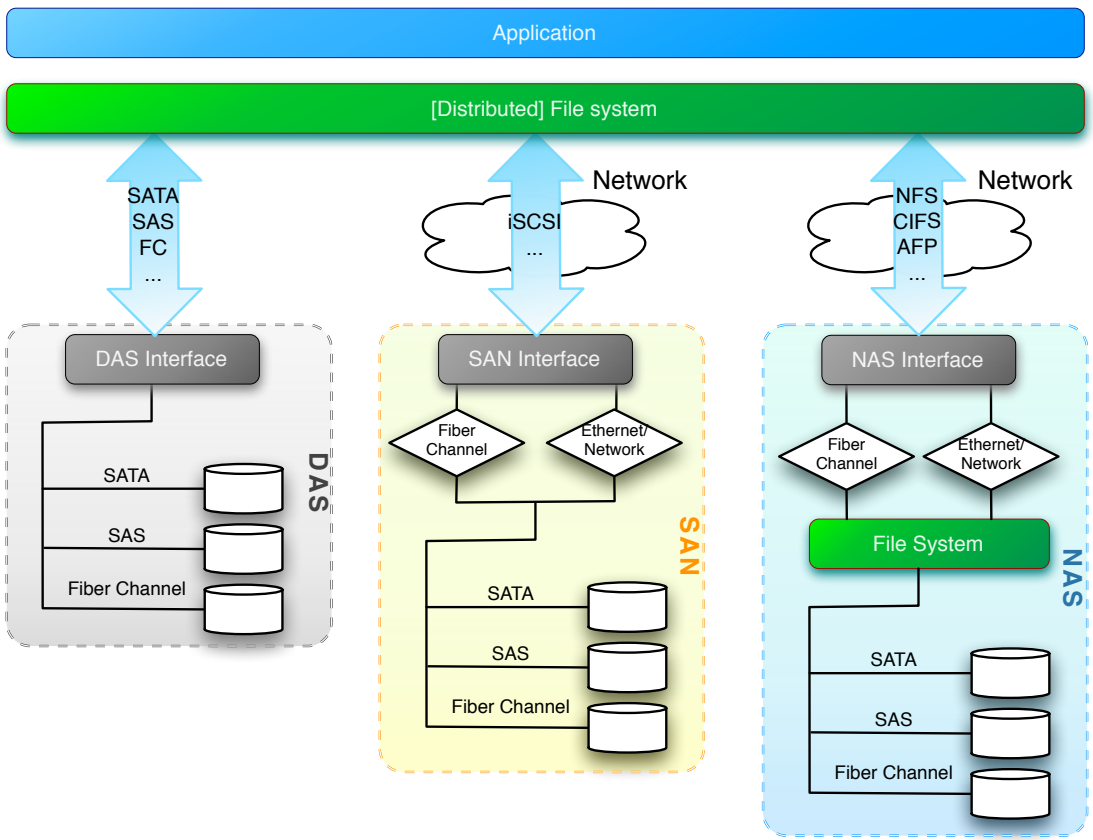
- Mainly Linux-based OS (91.4%) (Top500, Nov 2011)
- ... or Unix based (6%)
- Reasons:
  - ↪ stability
  - ↪ prone to devels

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.

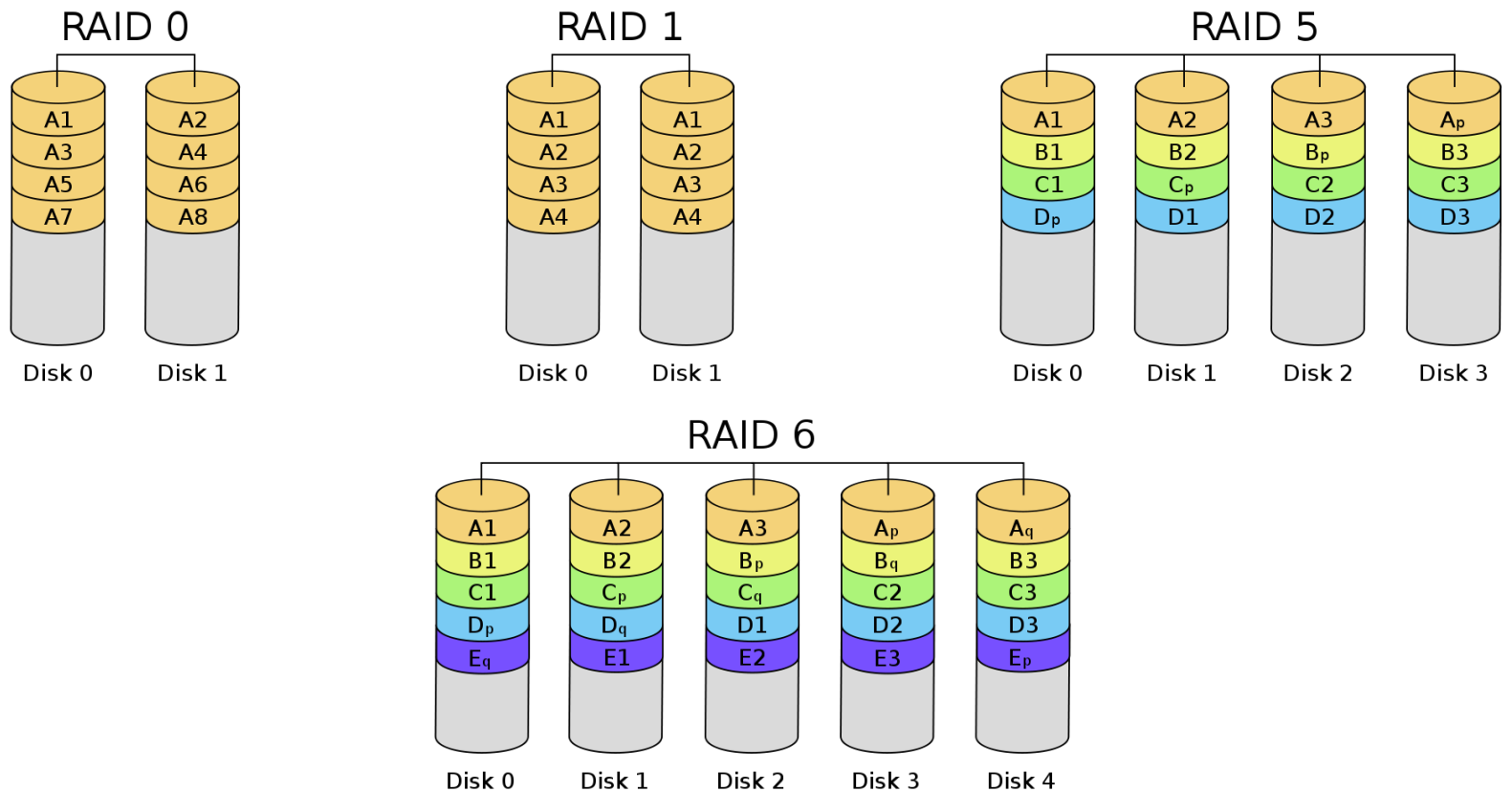
HPC Components: Data Management

Storage architectural classes & I/O layers



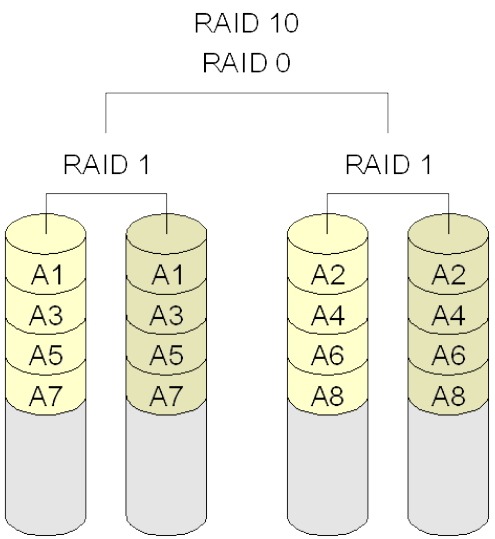
HPC Components: Data Management

RAID standard levels



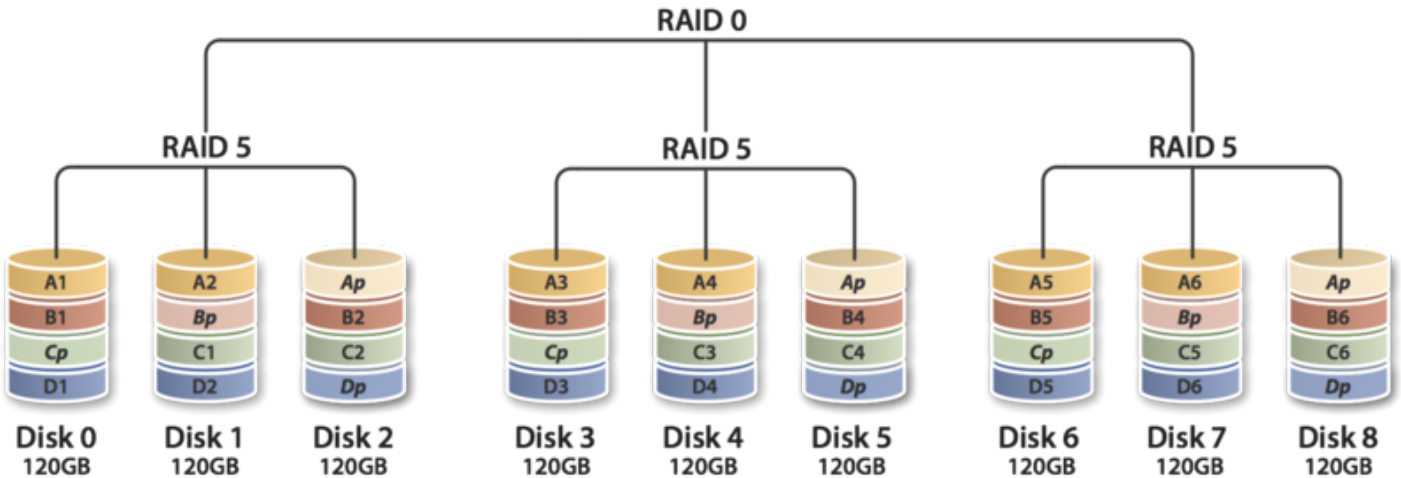
HPC Components: Data Management

RAID combined levels



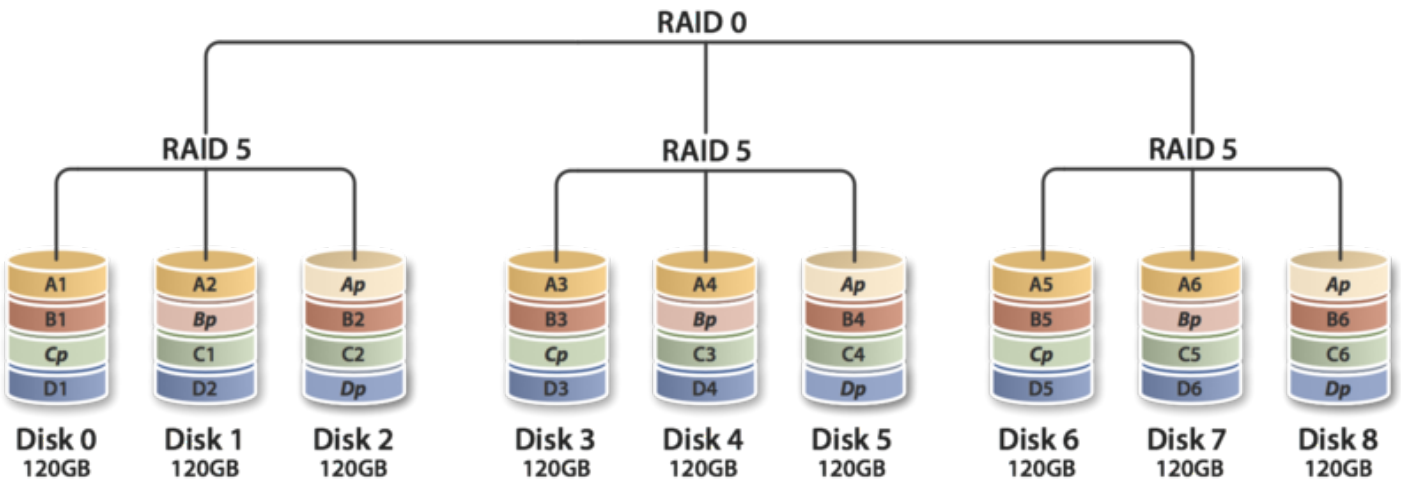
HPC Components: Data Management

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

↪ 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

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

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...

Summary

1 Overview of the Main HPC Components

2 **The UL HPC platform**
Overview
Platform Deployment with FAI and Puppet
Monitoring
Statistics & Milestones

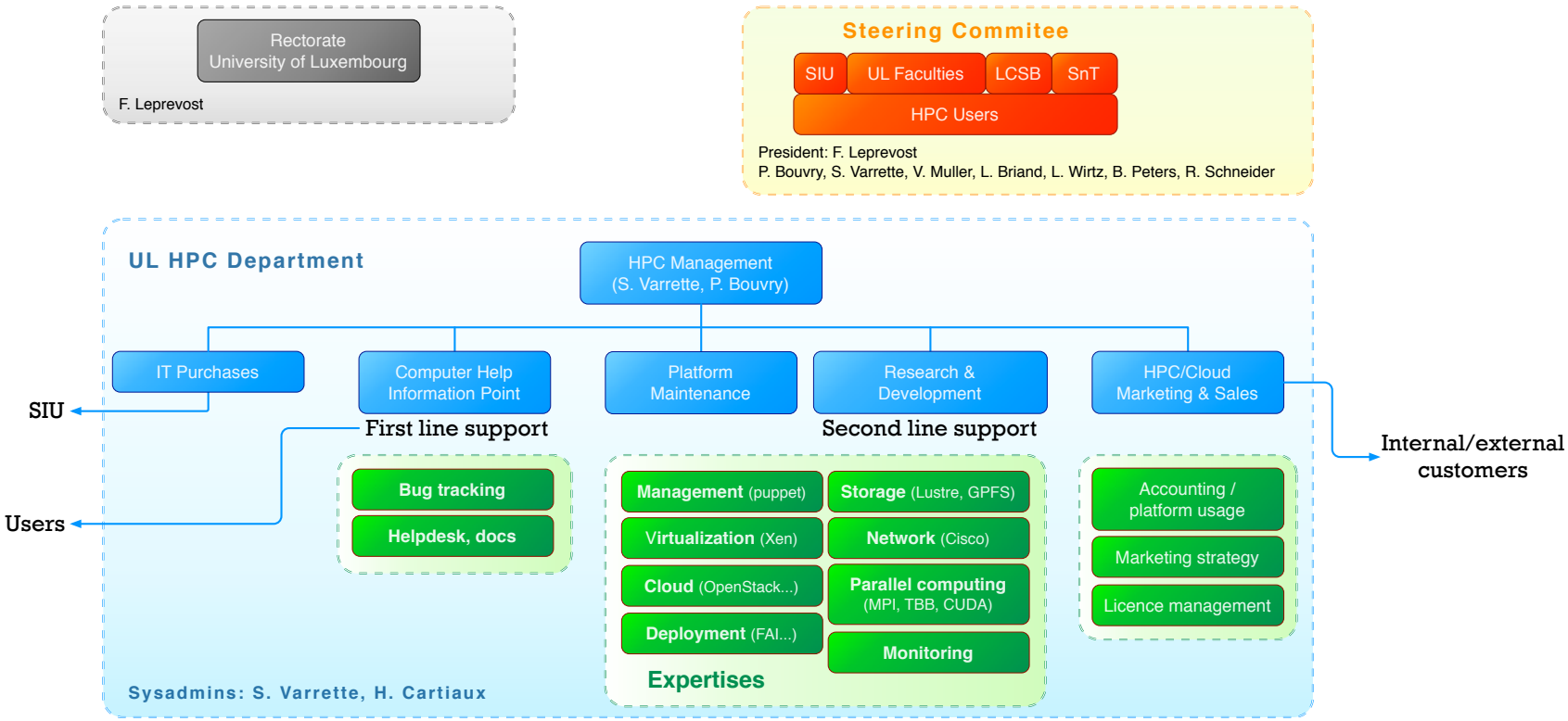
3 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
General considerations
SSH Access
The OAR Batch Scheduler
Available Software and Environment Modules

4 Last Challenges
Memory bottleneck
Effective Storage and Memory Management
Reporting problems and Promoting the Platform

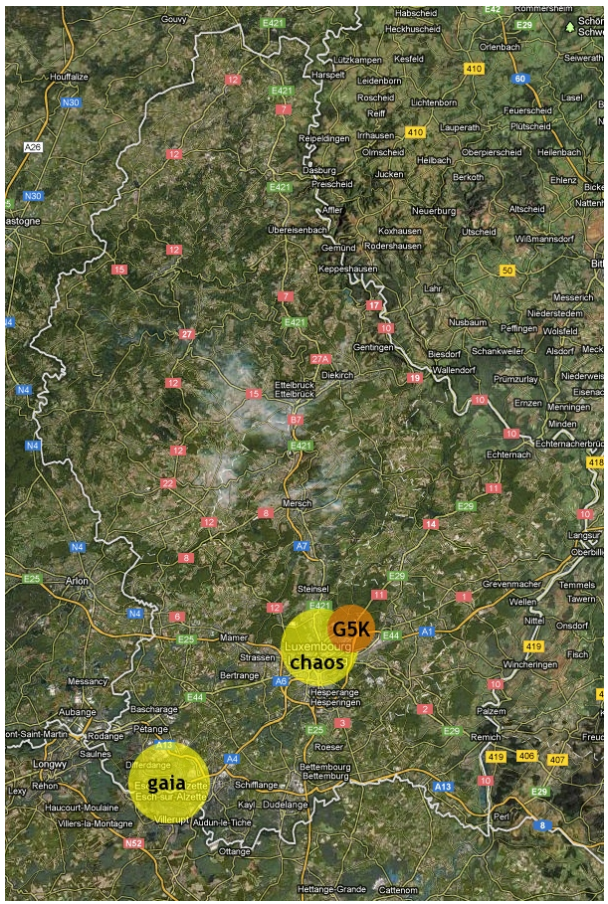
HPC @ UL

http://hpc.uni.lu

Organizational Chart



UL HPC platforms at a glance (2014)



- **2 geographical sites**, **3 server rooms**
- **4 clusters**: `chaos+gaia`, `granduc`, `nyx`.
  - ↪ **368 nodes, 3880 cores, 43.204 TFlops**
  - ↪ incl. **18** dual [GP]GPU nodes
  - ↪ **1996.4 TB (raw) shared storage**
    - incl. 1.3 PB on 7 backup enclosures
- **2 sysadmins** `hpc-sysadmins@uni.lu`
- **5,749,432€ (Cumul. HW Investment)** since 2007
  - ↪ Hardware acquisition only
  - ↪ 3,487,029€ (excluding server rooms)
- **Open-Source** software stack
  - ↪ SSH, LDAP, OAR, Puppet, Modules...

UNIVERSITÉ DU LUXEMBOURG

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

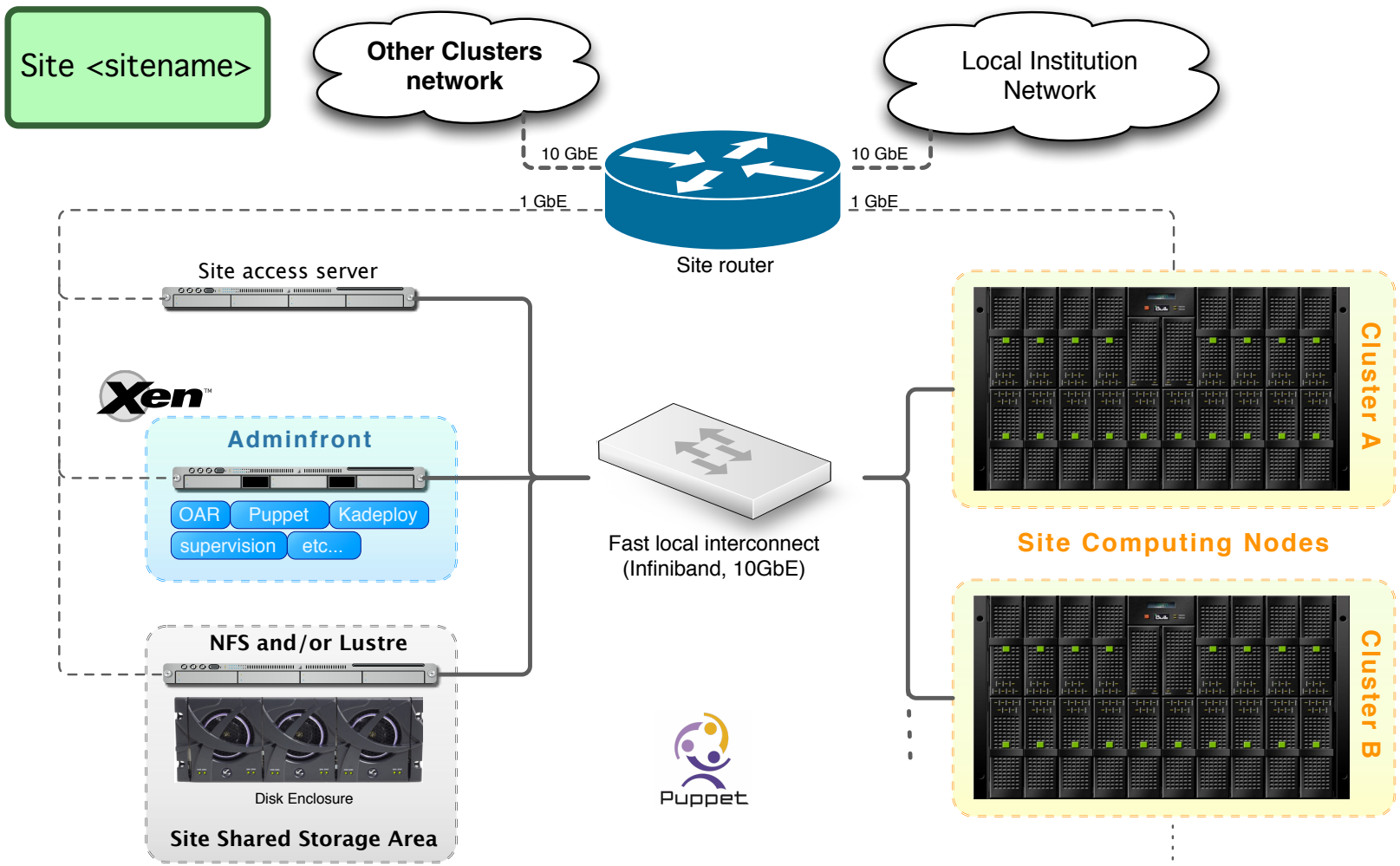
	Date	Vendor	Proc. Description		#N	#C	R_{peak}
chaos	2010	HP	Intel Xeon L5640@2.26GHz	2 × 6C,24GB	32	384	3.472 TFlops
	2011	Dell	Intel Xeon L5640@2.26GHz	2 × 6C,24GB	16	192	1.736 TFlops
	2012	Dell	Intel Xeon X7560@2,26GHz	4 × 6C, 1TB	1	32	0.289 TFlops
	2012	Dell	Intel Xeon E5-2660@2.2GHz	2 × 8C,32GB	16	256	4.506 TFlops
	2012	HP	Intel Xeon E5-2660@2.2GHz	2 × 8C,32GB	16	256	4.506 TFlops
				`chaos` **TOTAL:**	**81**	**1120**	**17.026 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	0.307 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
				`gaia` **TOTAL:**	**151**	**2000**	**21.178 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	1.536 TFlops
				`granduc/petitprince` **TOTAL:**	**38**	**368**	**4.48 TFlops**

Testing cluster:

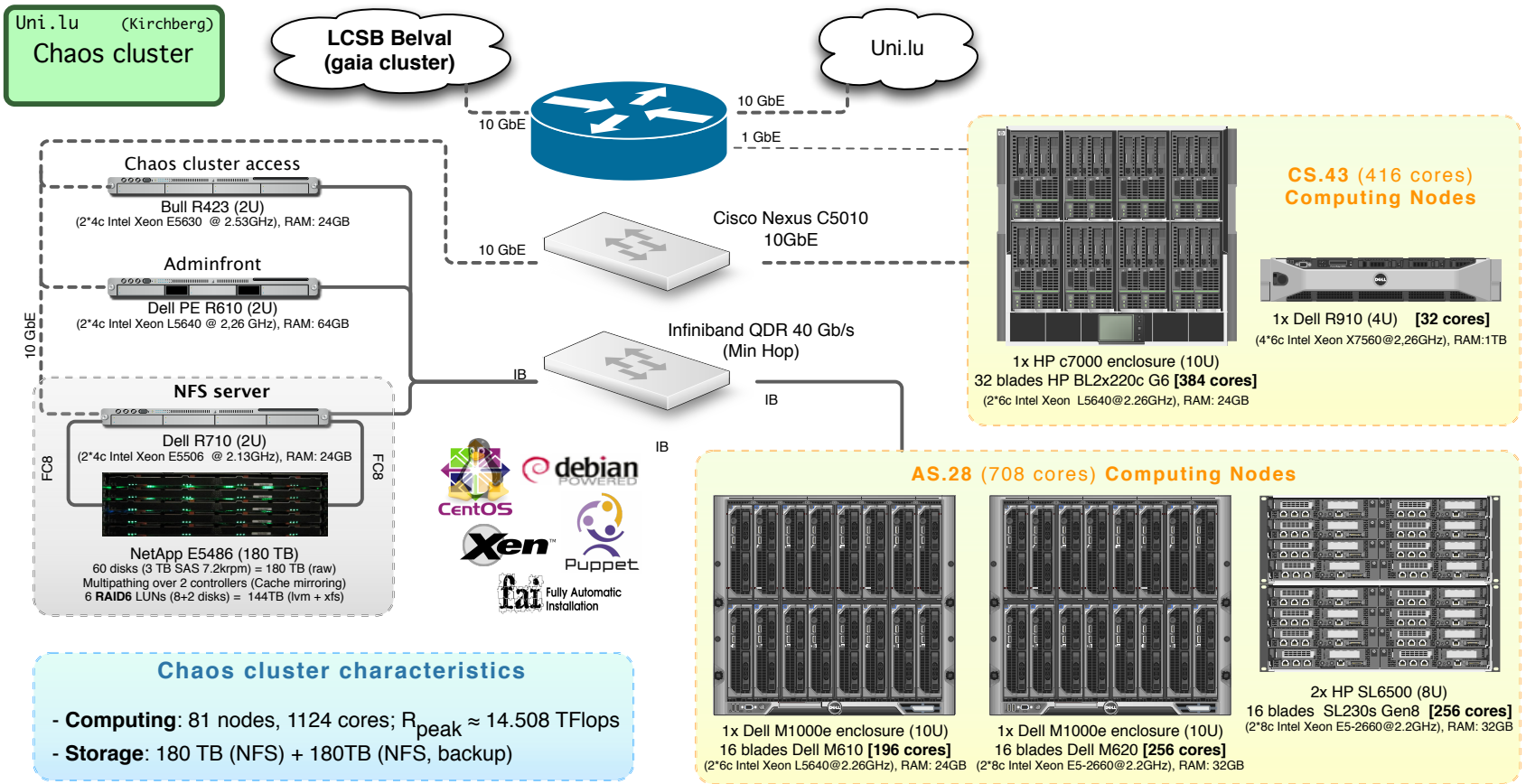
nyx	2012	Dell	Intel Xeon E5-2420@1.9GHz	1 × 6C,32GB	2	12	0.091 TFlops
viridis	2013	Viridis	ARM A9 Cortex@1.1GHz	1 × 4C,4GB	96	384	0.422 TFlops
				`nyx/viridis` **TOTAL:**	**98**	**392**	**0.52 TFlops**

TOTAL: 368 nodes, 3880 cores, 43.204 TFlops

UL HPC: General cluster organization



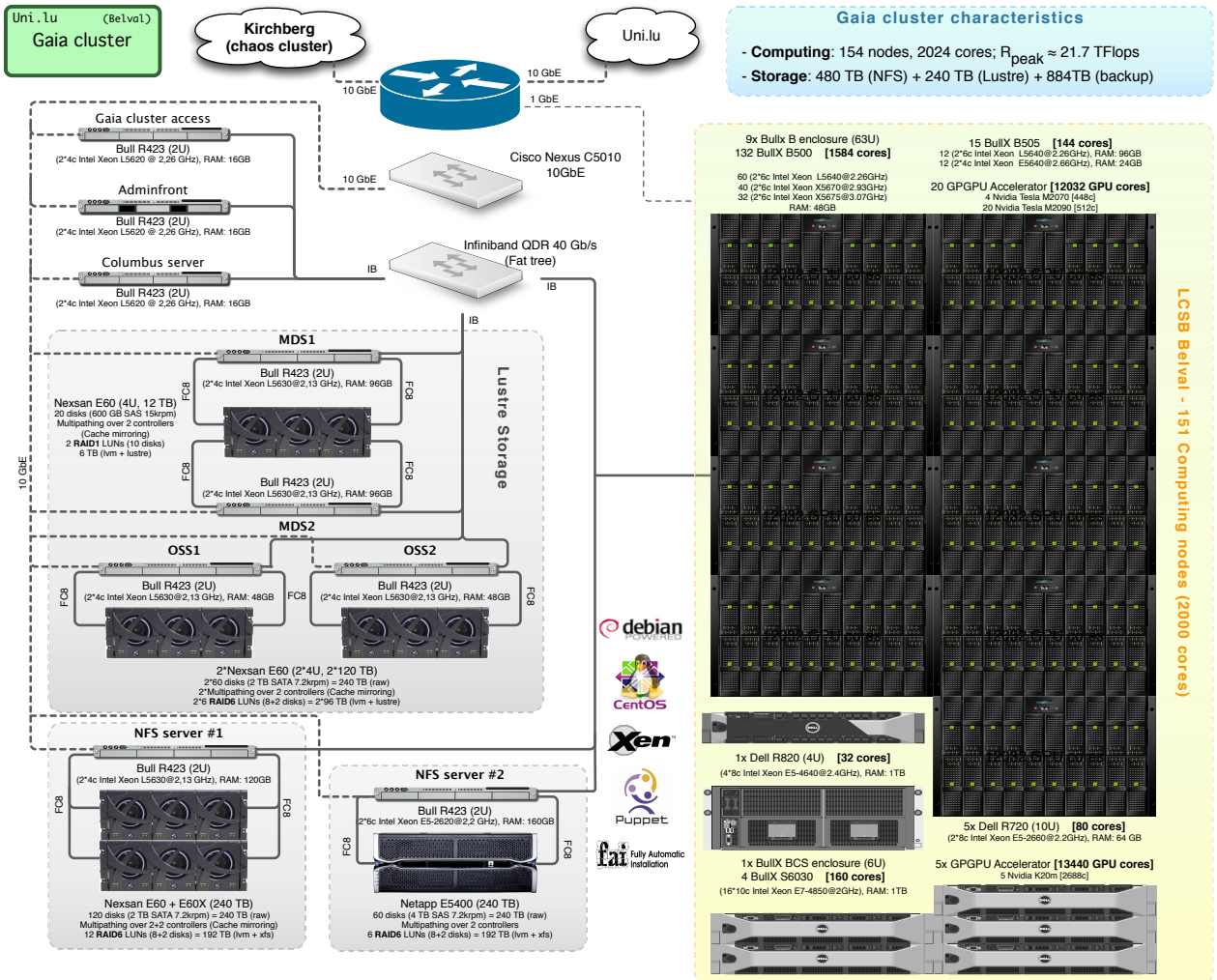
Ex: The chaos cluster



Chaos cluster characteristics

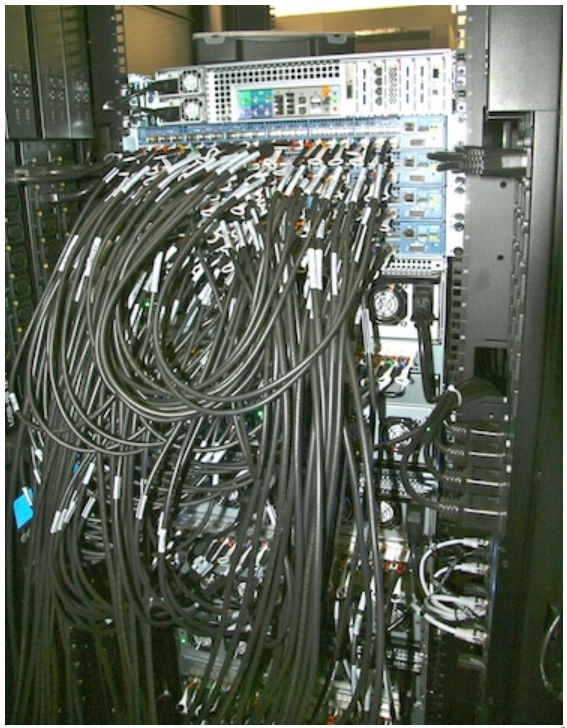
- **Computing**: 81 nodes, 1124 cores; $R_{peak} \approx 14.508$ TFlops
- **Storage**: 180 TB (NFS) + 180TB (NFS, backup)

Ex: The gaia cluster



Ex: Some racks of the gaia cluster





UL HPC Software Stack Characteristics

- **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.

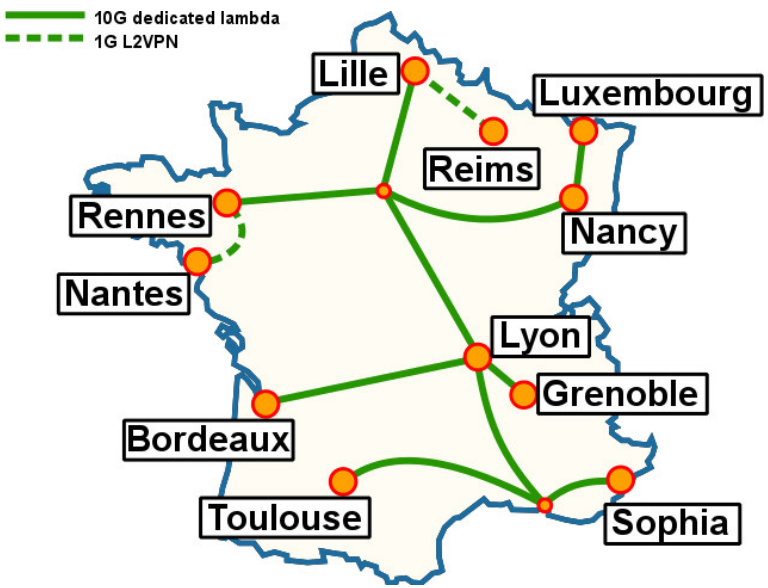
HPC in the Grande region and Around

			TFlops	TB	FTEs
Country	**Name/Institute**	**#Cores**	**R_{peak}**	**Storage**	**Manpower**
Luxembourg	UL	3880	43.204	1996.4	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

- 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

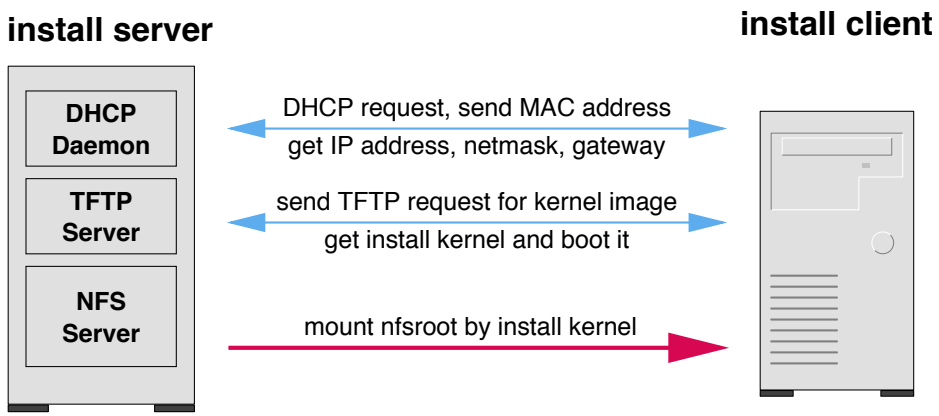
- Out of scope for this talk
- **Grid'5000 school 2014**
  - ↪ Organized in Lyon, from June 16th to June 19th 2014
  - ↪ More info: G5K School website
- General information: **https://hpc.uni.lu/g5k**
- Grid'5000 website and documentation: **https://www.grid5000.fr**

Computing nodes Management

Node deployment by FAI

http://fai-project.org/

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



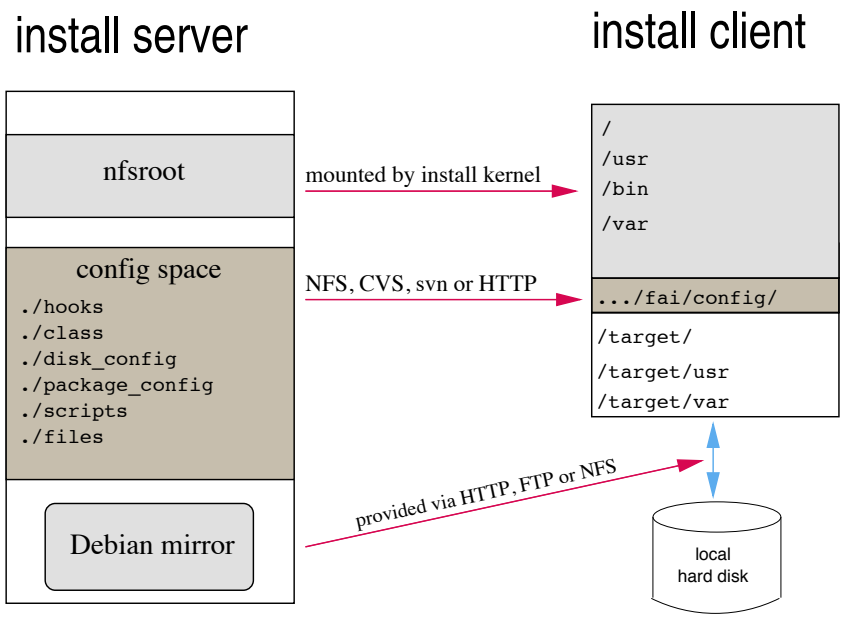
Computing nodes Management

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)



UNIVERSITÉ DU LUXEMBOURG

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/

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

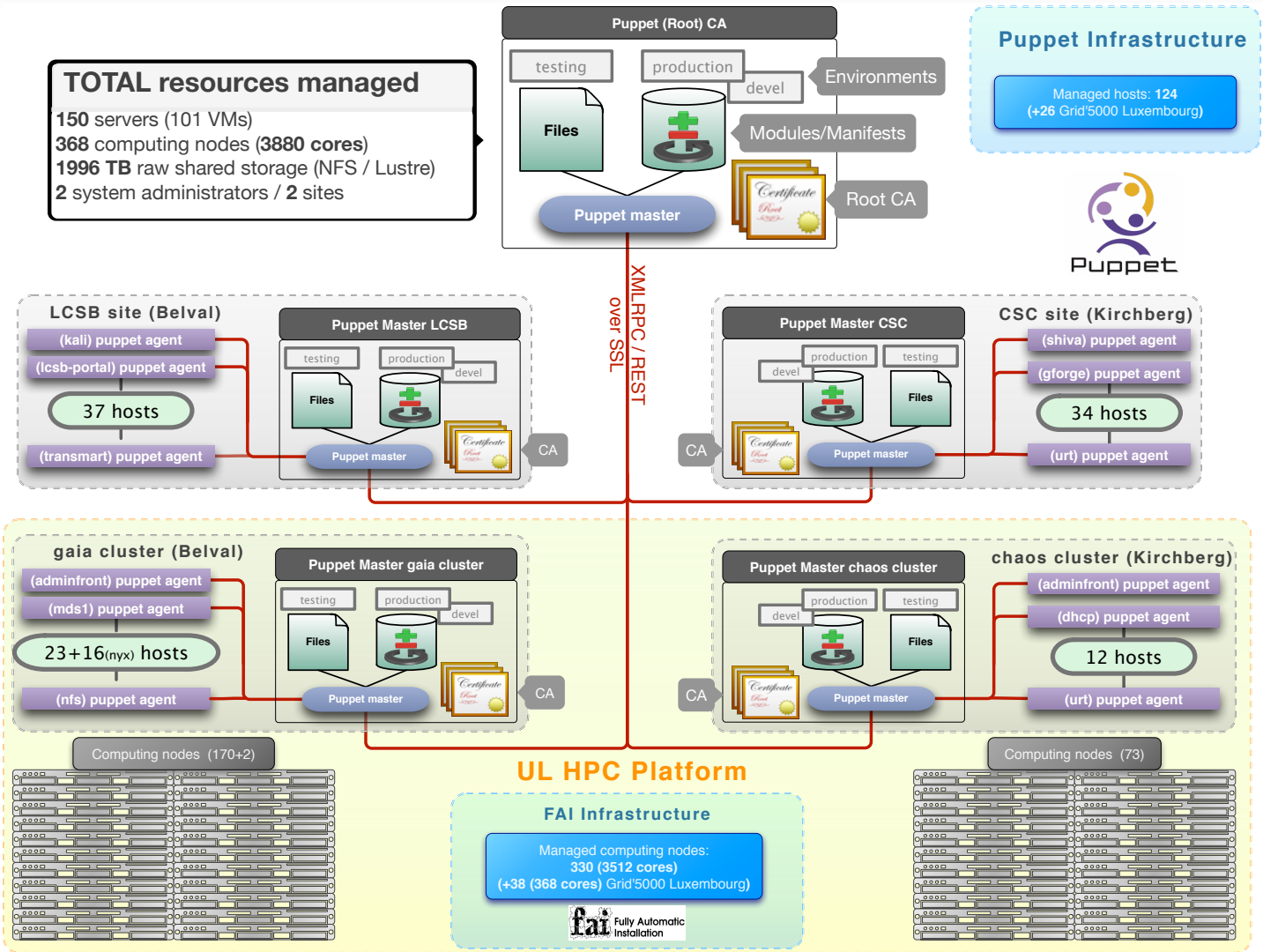
Server/Service configuration by Puppet

http://puppetlabs.com

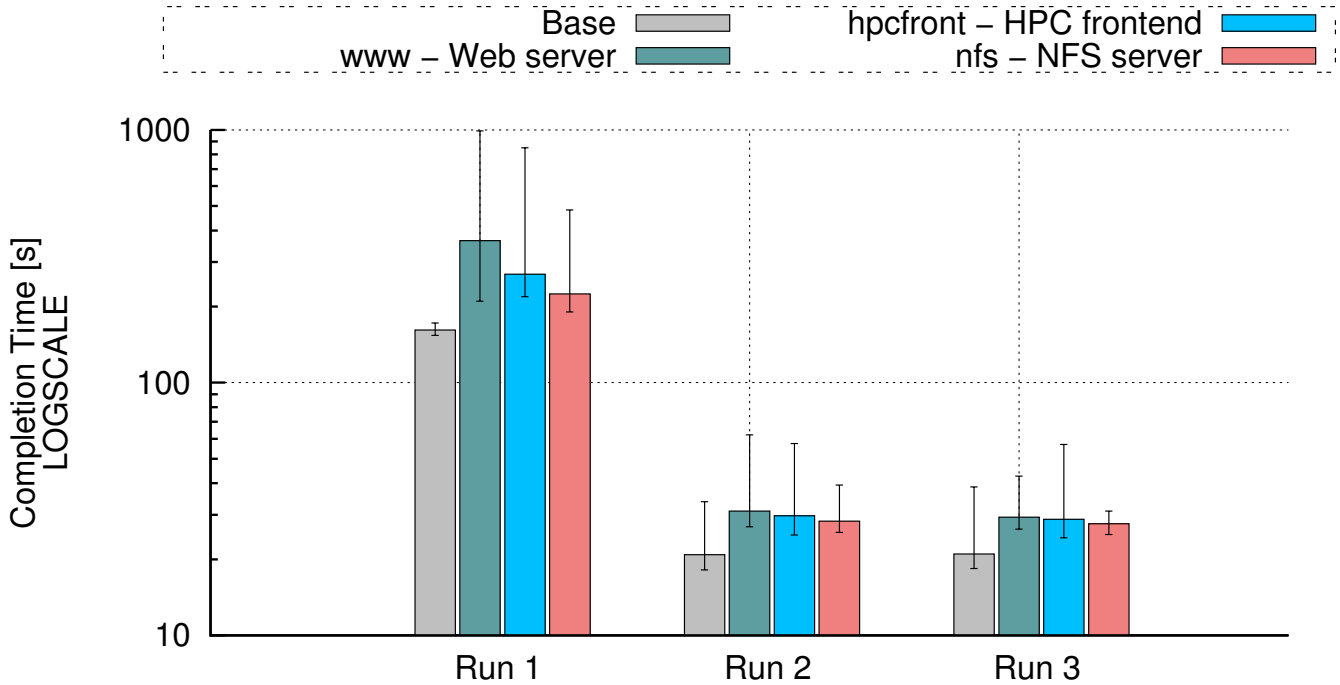
- *Configuration management made easy*
- Automates service deployment / sysadmin tasks
- 3 components:
  - ↪ declarative language (manifests)
  - ↪ client/server model
  - ↪ Git-based workflow + Rakefile + Capistrano
- Hierarchical PKI

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

Platform Management: Puppet



Puppet Performances



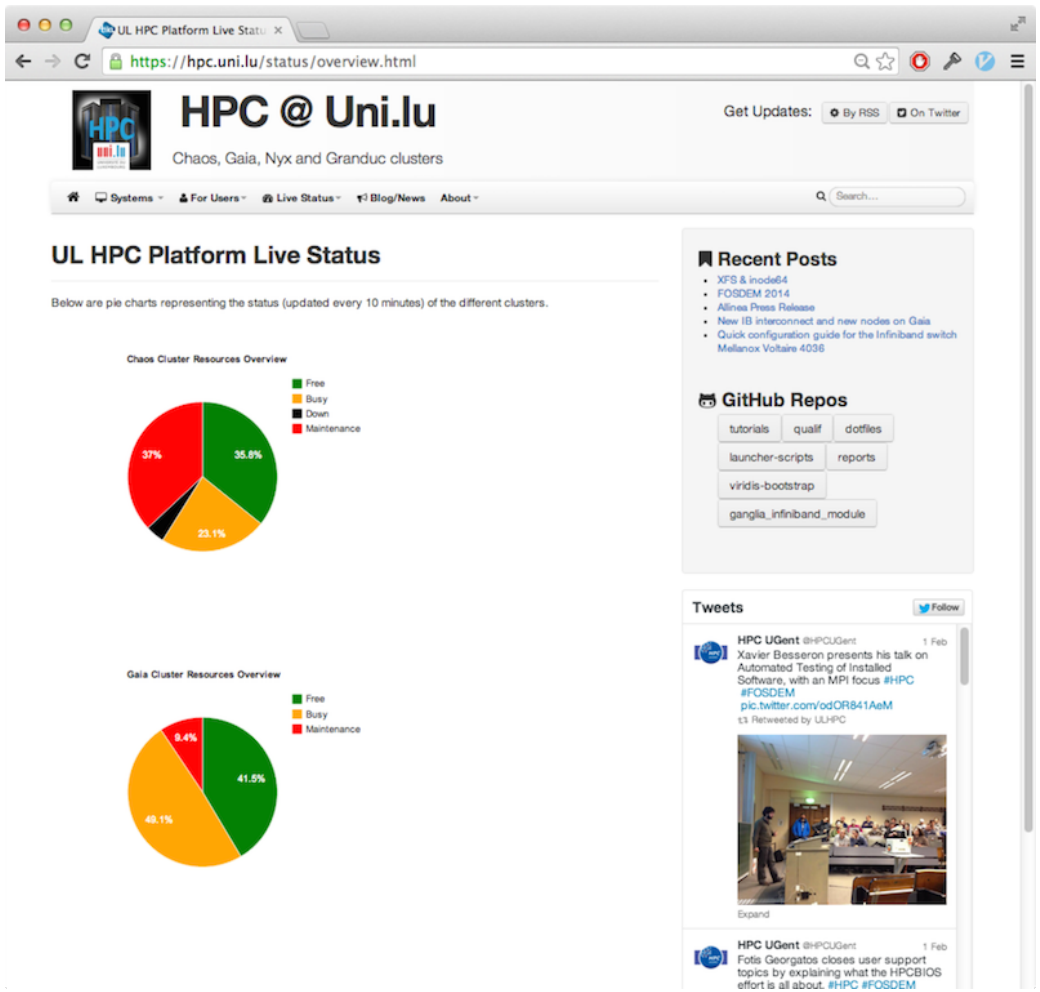
- between 161s and 364s to completely **bootstrap** a virgin node
  - ↪ between 20s and 31s to later check/correct the config

Now proposed as an IT service to external consumers

Platform Monitoring

General Live Status

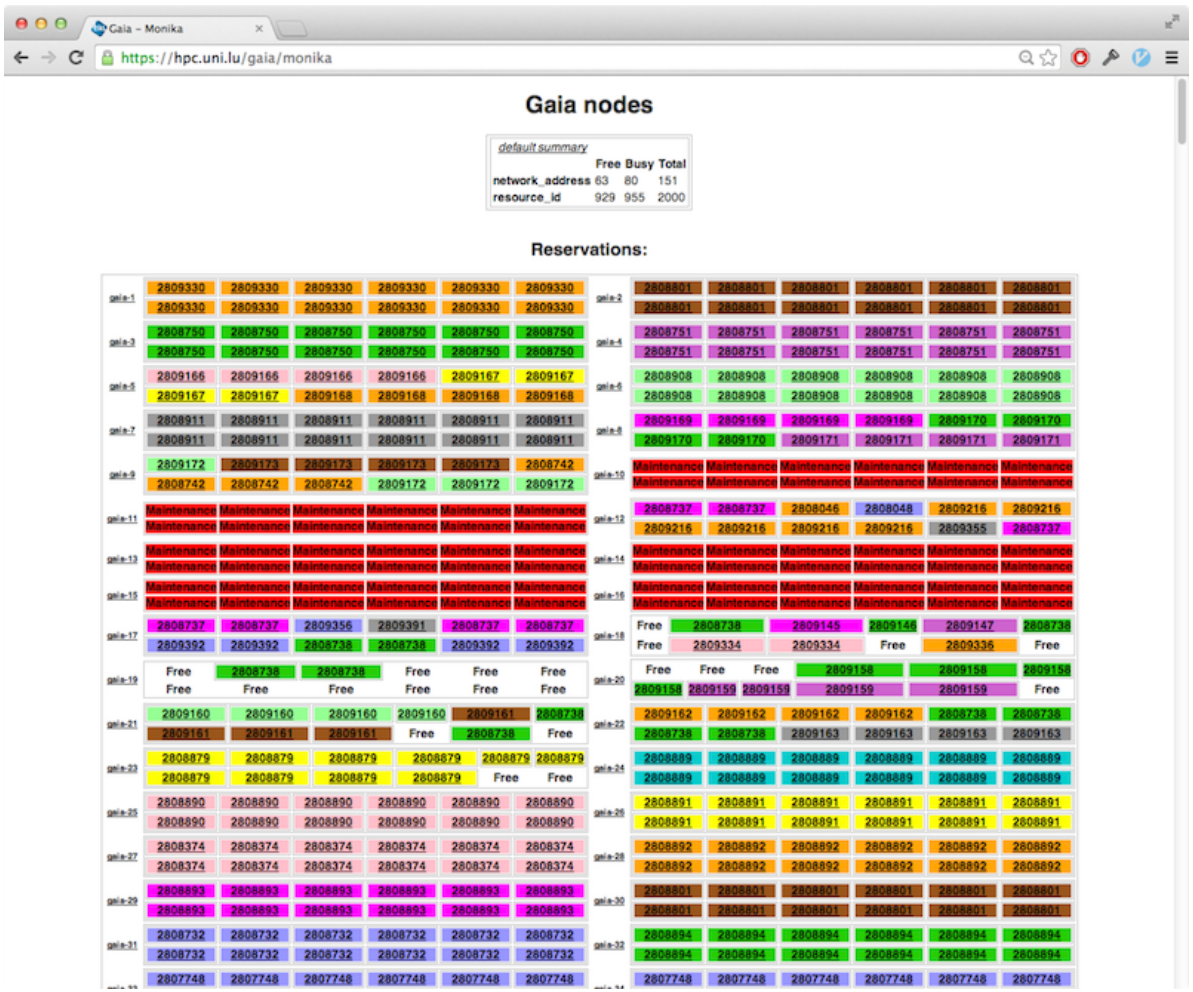
`http://hpc.uni.lu/status/overview.html`



Platform Monitoring

Monika

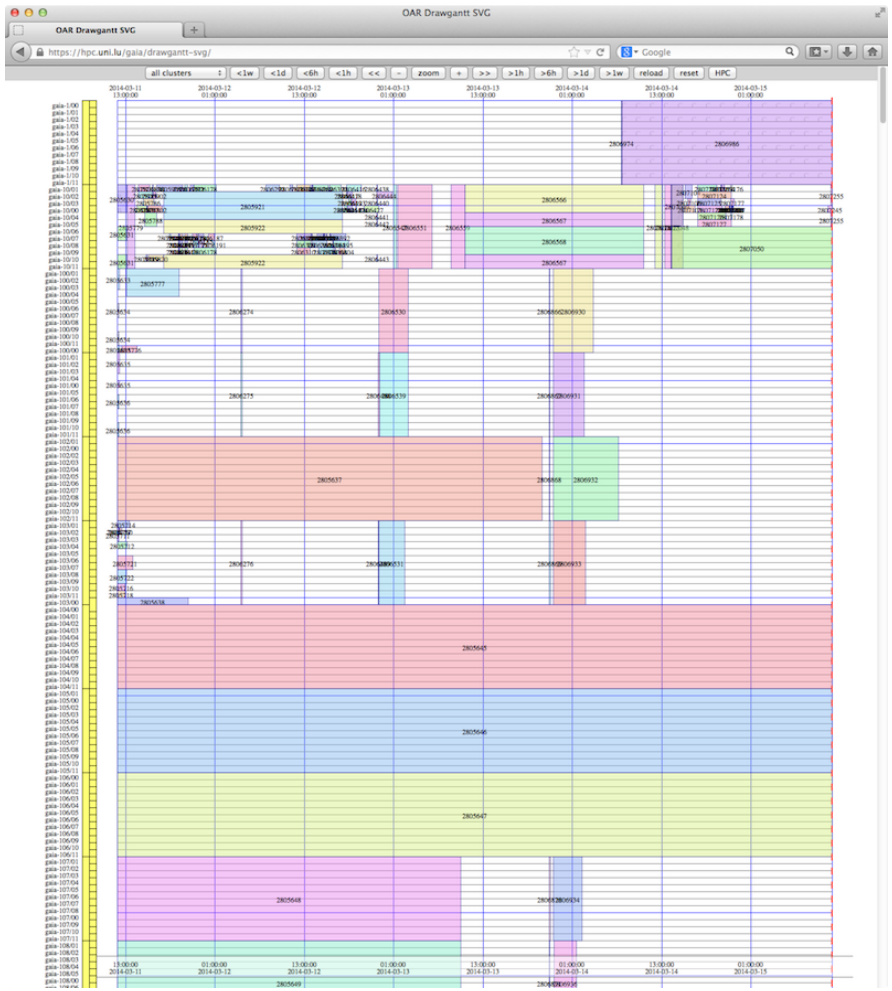
http://hpc.uni.lu/{chaos,gaia,g5k}/monika



Platform Monitoring

Drawgantt

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



Platform Monitoring

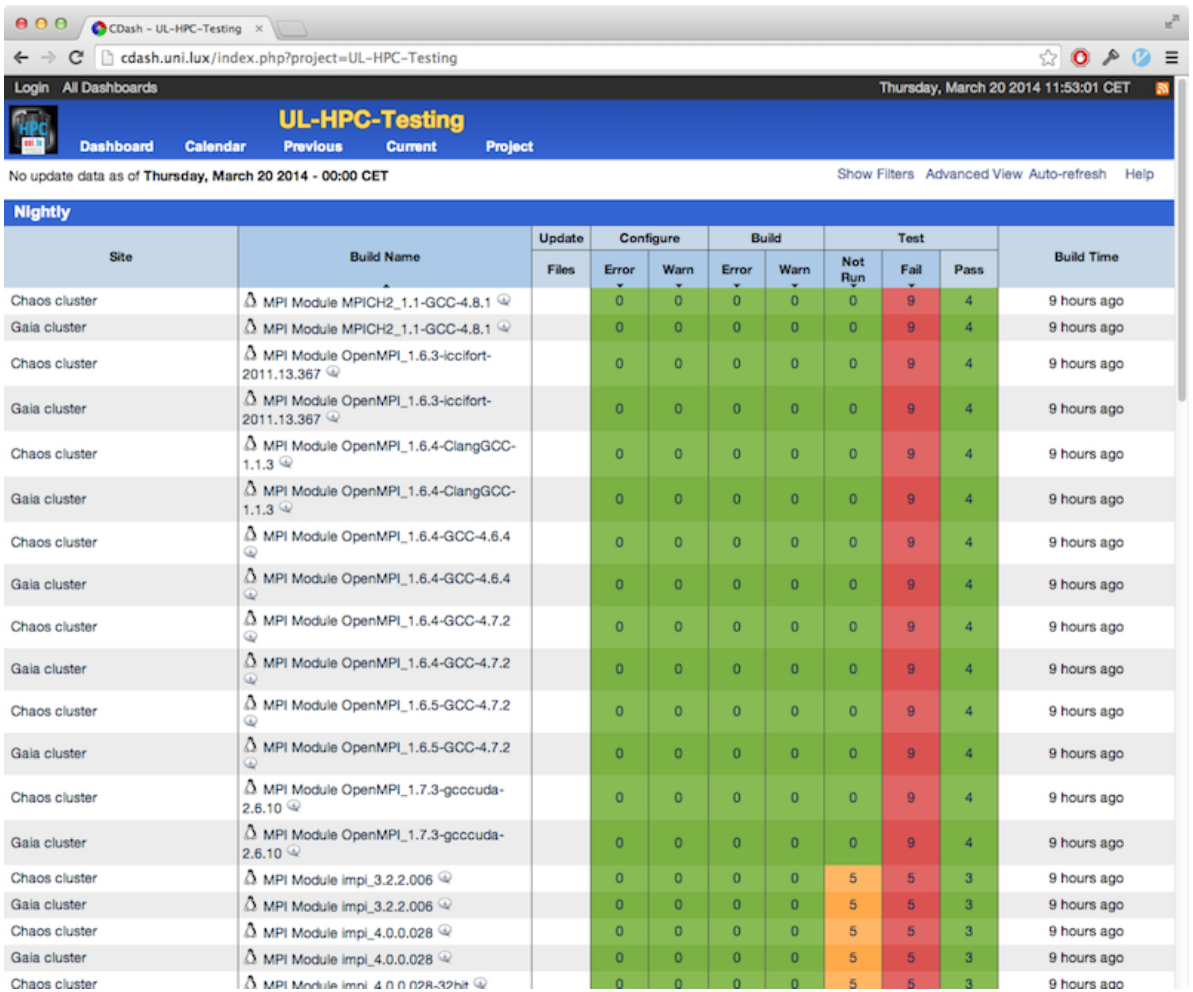
Ganglia

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

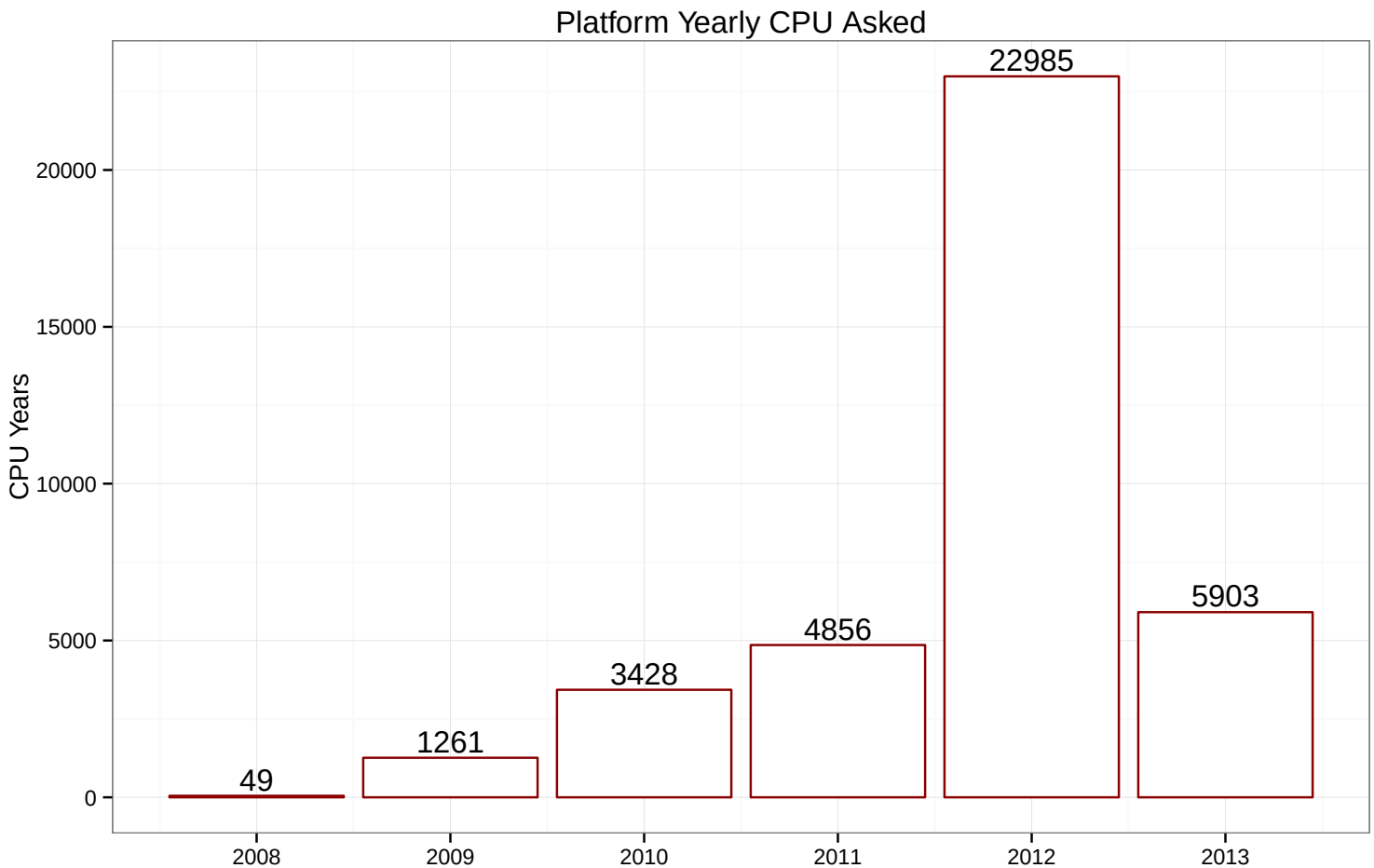
Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
Chaos cluster	MPI Module MPICH2_1.1-GCC-4.8.1		0	0	0	0	0	9	4	9 hours ago
Gaia cluster	MPI Module MPICH2_1.1-GCC-4.8.1		0	0	0	0	0	9	4	9 hours ago
Chaos cluster	MPI Module OpenMPI_1.6.3-iccifort-2011.13.367		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

- **368 nodes, 3880 cores, 43.204 TFlops**
- **1996.4 TB (raw) shared storage**
  - ↪ incl. 1.3 PB on 7 backup enclosures
- **5,749,432€ (Cumul. HW Investment)** since 2007
- 235 registered users
- Total **Asked** resources: **40537 years** since 2008
- Total **Used** resources: **2482 years** since 2008

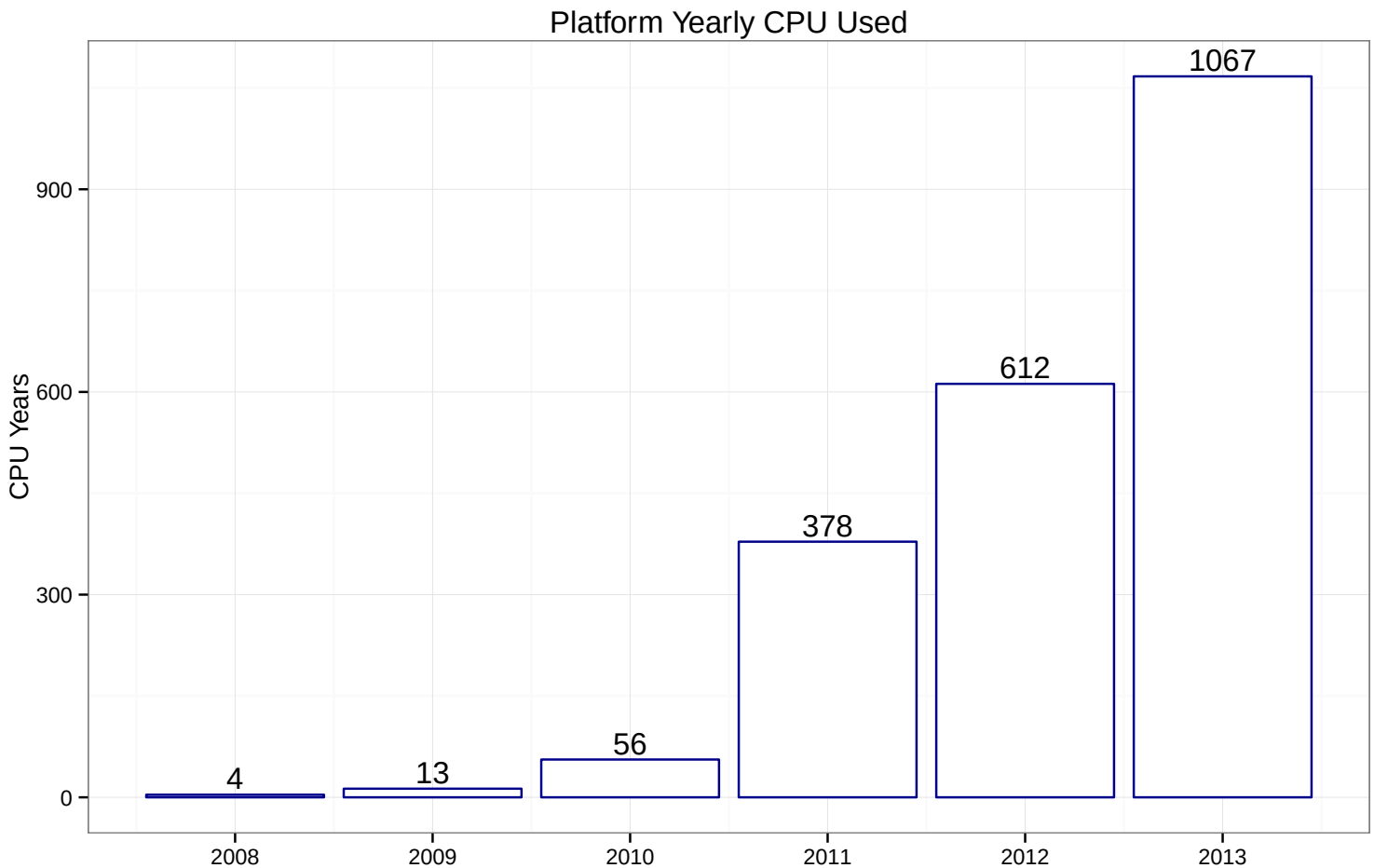
CPU-year usage since 2008

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

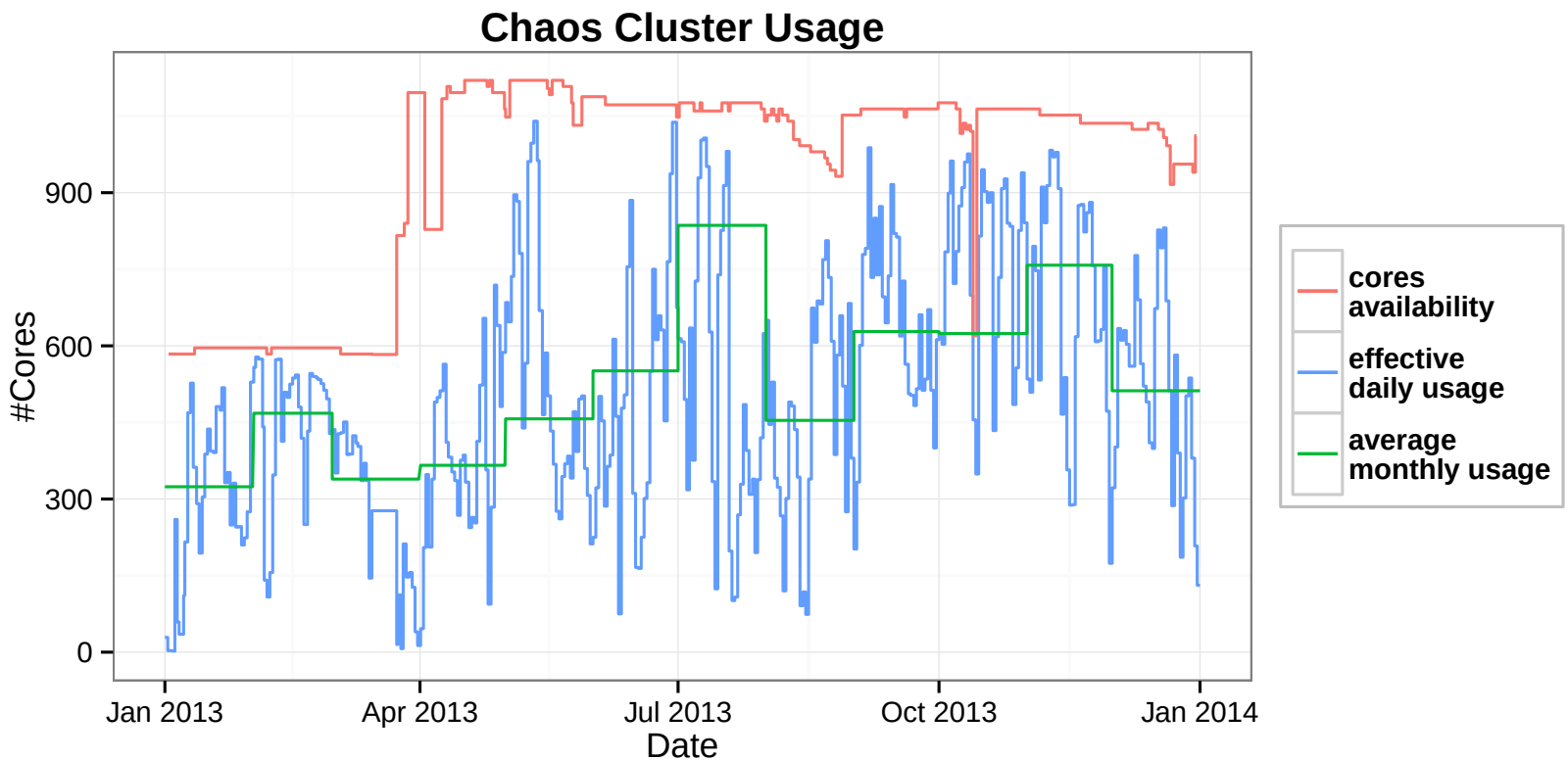


CPU-year usage since 2008

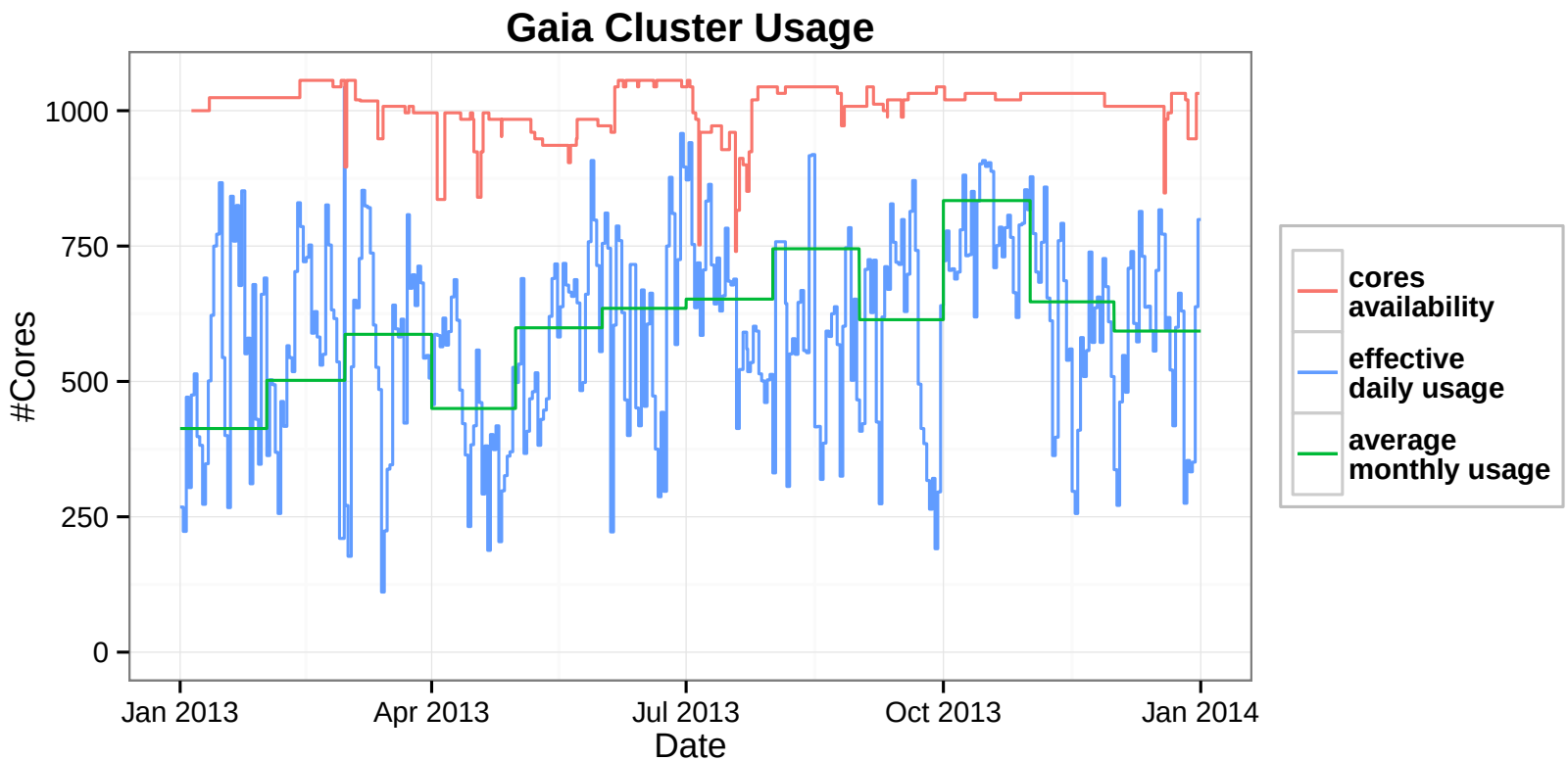
- **CPU-hour**: *work* done by a CPU in one hour of wall clock time



Chaos Cluster Usage in 2013



Gaia Cluster Usage in 2013



2013 Milestones

- Creation of the HPC Steering Committee
- **2013 Budget: 715 963,04 €**
  - ↪ HPC: 54,5% LCSB: 28,73% CSC:5,62% Other:11,15%
- Platform [continuous] evolution
  - ↪ Ex: +181 nodes (+1432 cores), +12,21 TFlops
  - ↪ Consolidation of shared storage stability (NFS, Lustre)
  - ↪ New website, Tutorials on Github etc.
  - ↪ **Job submission web-portal** (testing)
- Events / System Downtime

Cluster	Yearly Downtime	Yearly Uptime
chaos	70h	**99.201%**
gaia	401h	**95.422%**
g5k	320h	**96.347%**

2014 Milestones

- 30% budget drop
  - ↪ new budget repartition in progress
  - ↪ **unavoidable**: new cost-model (computing & storage)
- 2014 budget fully dedicated to storage component
  - ↪ BigNAS RFP (1 PB storage) – shared SIU / HPC / LCSB
  - ↪ Lustre consolidation
- PRACE observer status

Incoming milestones (2015/~~2016~~)

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

Summary

1 Overview of the Main HPC Components

2 The UL HPC platform
- Overview
- Platform Deployment with FAI and Puppet
- Monitoring
- Statistics & Milestones

3 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
- General considerations
- SSH Access
- The OAR Batch Scheduler
- Available Software and Environment Modules

4 Last Challenges
- Memory bottleneck
- Effective Storage and Memory Management
- Reporting problems and Promoting the Platform

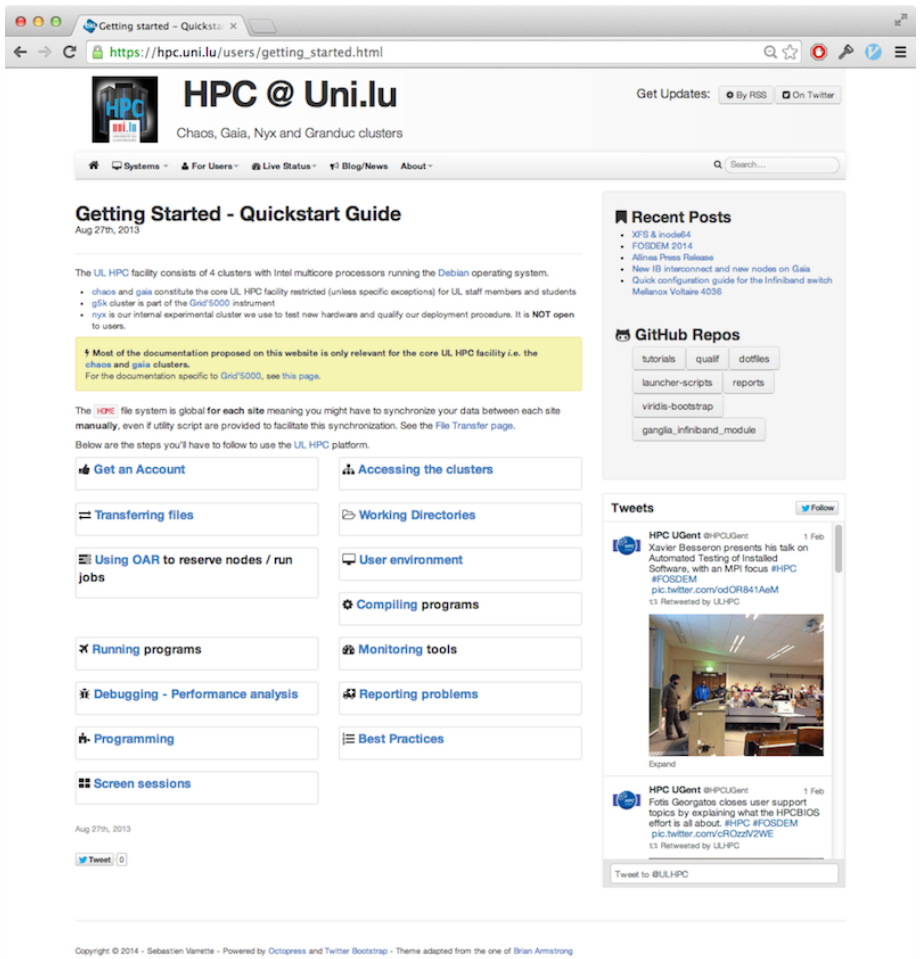
General Consideration

- 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:

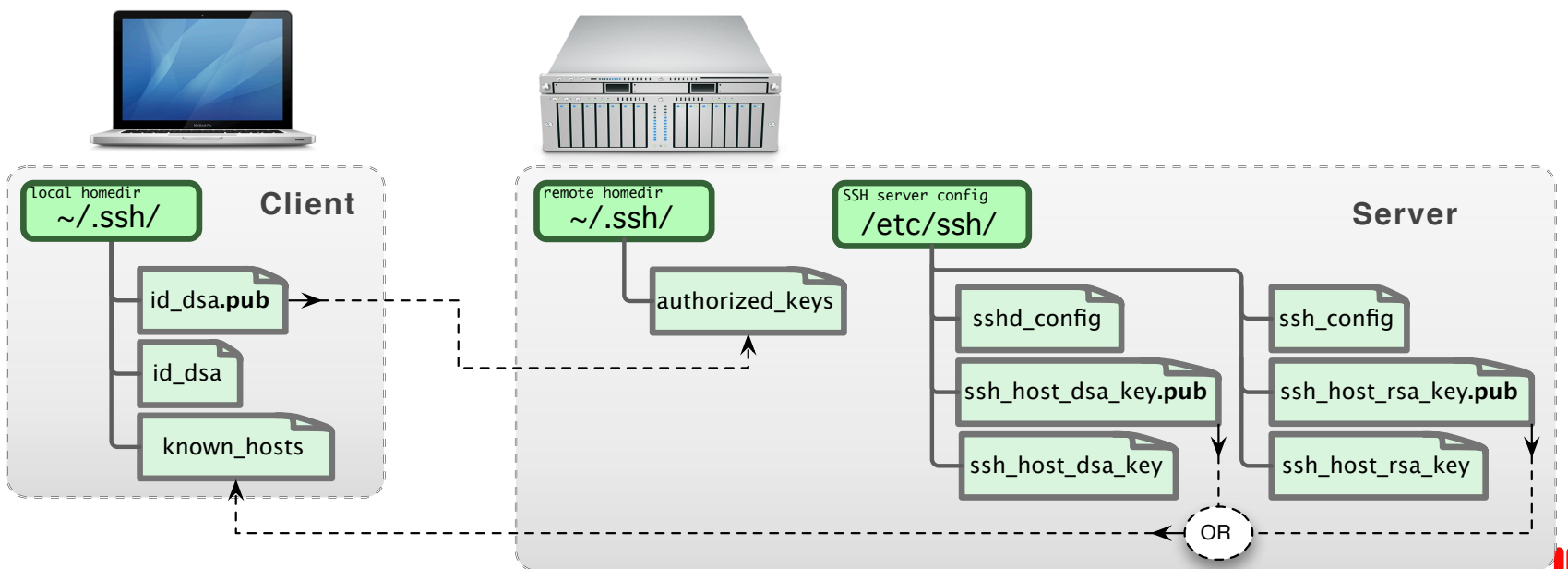
- Access to the UL HPC wiki http://hpc.uni.lu/
- Access to the UL HPC bug tracker http://hpc-tracker.uni.lu/
- Subscribed to the mailing lists hpc-{users,platform}@uni.lu
  - ↪ raise questions and concerns. Help us to make it a community!
  - ↪ notification of platform maintenance on hpc-platform@uni.lu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

Typical Workflow on UL HPC resources

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

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

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
```

Cancel a job

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

Status of a jobs

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

View the job

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

Get info on the nodes

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

Run a best-effort job

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

UNIVERSITÉ DU LUXEMBOURG

Designing efficient OAR job launchers

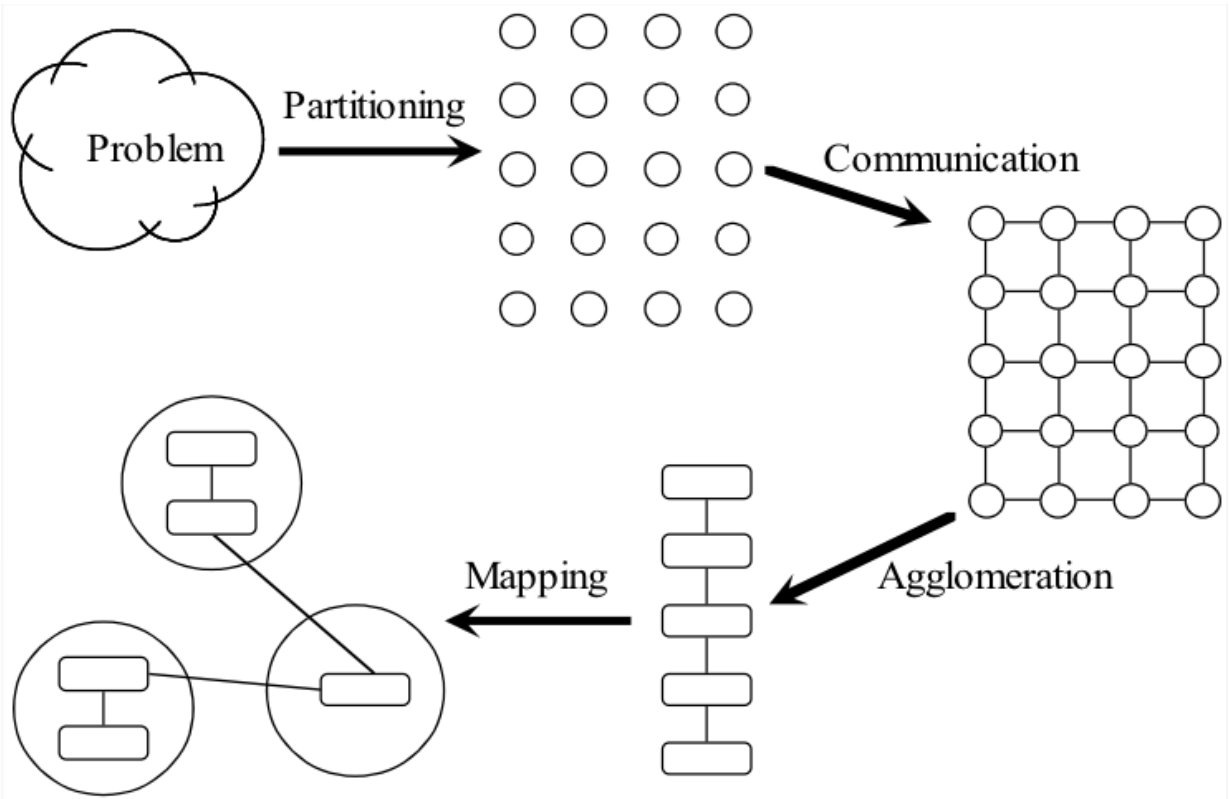
Resources/Example

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

```
#OAR -l nodes=1
#OAR -n BADSerial
#OAR -O BADSerial-%jobid%.log
#OAR -E BADSerial-%jobid%.log
if [ -f  /etc/profile ]; then
    .  /etc/profile
fi
# Now you can use: 'module load toto' or 'cd $WORK'
[...]
```

Serial tasks: BAD and NAIVE approach

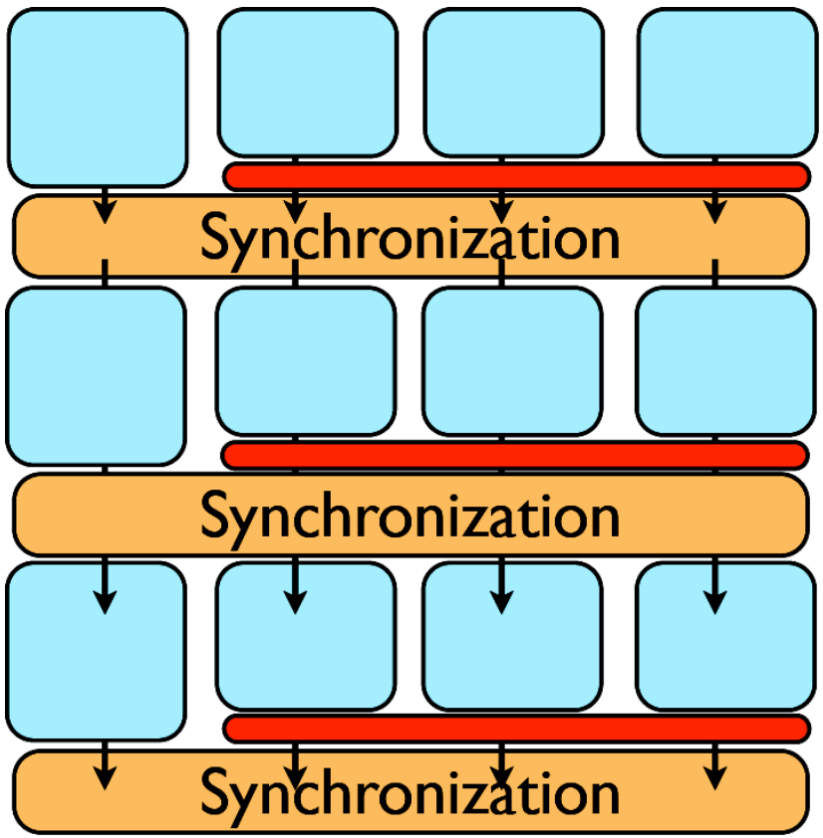
```
#OAR -l nodes=1
#OAR -n BADSerial
#OAR -O BADSerial-%jobid%.log
#OAR -E BADSerial-%jobid%.log
if [ -f  /etc/profile ]; then
    .  /etc/profile
fi
# Now you can use: 'module load toto' or 'cd $WORK'
[...]

# 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)

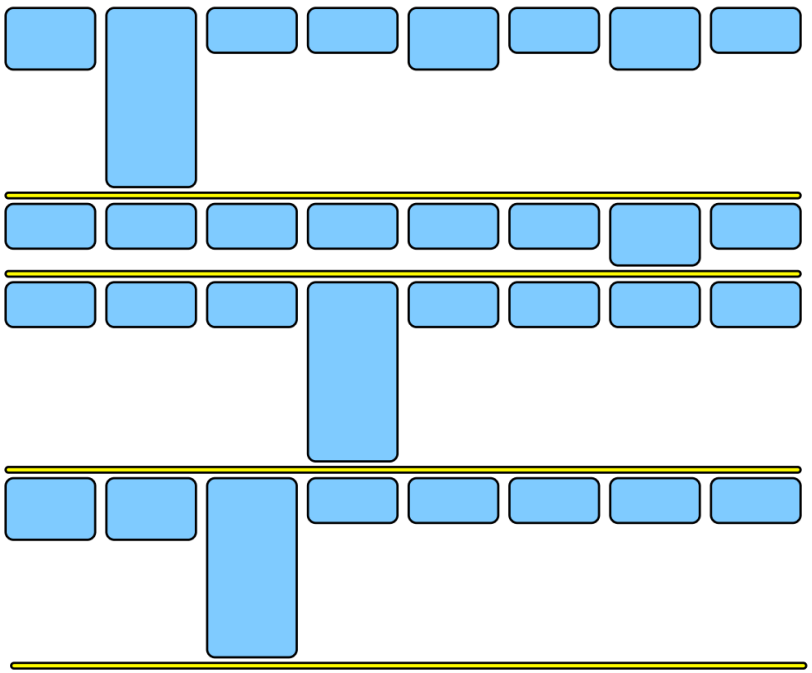
```
# Example 1: run in sequence $TASK 1...$TASK $NB_TASKS
for i in `seq 1 $NB_TASKS`; do
    $TASK $i &
done
wait

# 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)

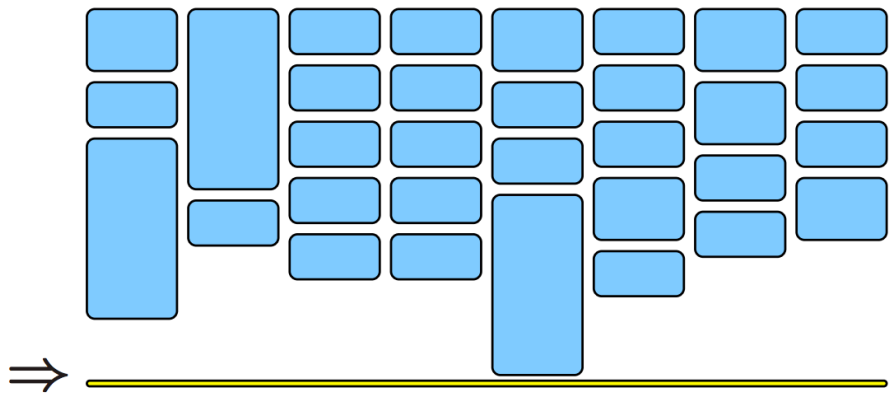
Different runs may not take the same time: **load imbalance**.



Serial tasks with GNU Parallel



17 hours
42% utilization



10 hours
72% utilization

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: **PS2**
  - ↪ HPC workflow with sequential jobs (C,python,java)

Software Management

- Most default software comes from the OS
- Specific / optimized software available under Environment Modules
  - ↪ https://hpc.uni.lu/users/docs/modules.html
  - ↪ **To be used on cluster nodes**. Main commands:

```
(node)$> module help
(node)$> module avail |& less
(node)$> module avail |& grep -i '^gcc'
(node)$> module list
(node)$> module load modulename
```

- Compilation suites available: Intel CC, GCC

Available software

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

- **377 software packages**, in multiple versions

MPI Suites available via `module`

1. OpenMPI — http://www.open-mpi.org/

```
(node)$> module load OpenMPI
(node)$> make
(node)$> mpirun -machinefile $OAR_NODEFILE /path/to/mpi_prog
```

2. MVAPICH2 — http://mvapich.cse.ohio-state.edu/overview/mvapich2

```
(node)$> module purge
(node)$> module load MVAPICH2
(node)$> make clean && make
(node)$> mpirun -machinefile $OAR_NODEFILE /path/to/mpi_prog
```

3. Intel Cluster Toolkit Compiler Edition (`ictce` for short):

```
(node)$> module purge
(node)$> module load ictce
(node)$> make clean && make
(node)$> mpirun -machinefile $OAR_NODEFILE /path/to/mpi_prog
```

Specific Software Guidelines

- **MPI**: see **PS3**
  - ↪ HPC workflow with MPI jobs. Ex: OSU MB/HPL
- **Parallel Debuggers**: see **PS4** and **PS8**
  - ↪ TotalView, Alinea
- **MATLAB**: see **PS5**
  - ↪ Using Matlab on the UL HPC Platform
- **R**: see **PS6**
  - ↪ Using R on the UL HPC Platform
- **ABySS, Gromacs, Bowtie2 / TopHat, mpiBLAST**: see **PS7**
  - ↪ Bio-informatic softwares on the UL HPC Platform

Software Management

My favorite software is not installed on the cluster!

- Check if it does not exists via `module`
- **If not**: compile it in your home/work directory
  - ↪ using GNU `stow` `http://www.gnu.org/software/stow/`
  - ↪ Share it to others: consider EasyBuild / `ModuleFile`
- General workflow for programs based on **Autotools**
  - ↪ Get the software sources (version `x.y.z`)
  - ↪ Compile and install it in your home/work directory

```
(node)$> ./configure [options] -prefix=$BASEDIR/stow/mysoft.x.y.z
(node)$> make && make install
(node)$> cd $BASEDIR/stow && stow mysoft.x.y.z
```

Summary

1 Overview of the Main HPC Components

2 The UL HPC platform
Overview
Platform Deployment with FAI and Puppet
Monitoring
Statistics & Milestones

3 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
General considerations
SSH Access
The OAR Batch Scheduler
Available Software and Environment Modules

4 Last Challenges
Memory bottleneck
Effective Storage and Memory Management
Reporting problems and Promoting the Platform

Memory bottleneck

- A regular computing node have at least 2GB/core RAM
  - ↪ Do 12-24 runs fit in the memory?
  - ↪ If your job runs out of memory, it simply crashes
- Use fewer simultaneous runs if **really** needed!
  - ↪ **OR** request a big memory machine (1TB RAM)

```
$> oarsub -t bigmem ...
```

  - ↪ **Or** explore parallization (MPI, OpenMP, pthreads)

Understanding Your Storage Options

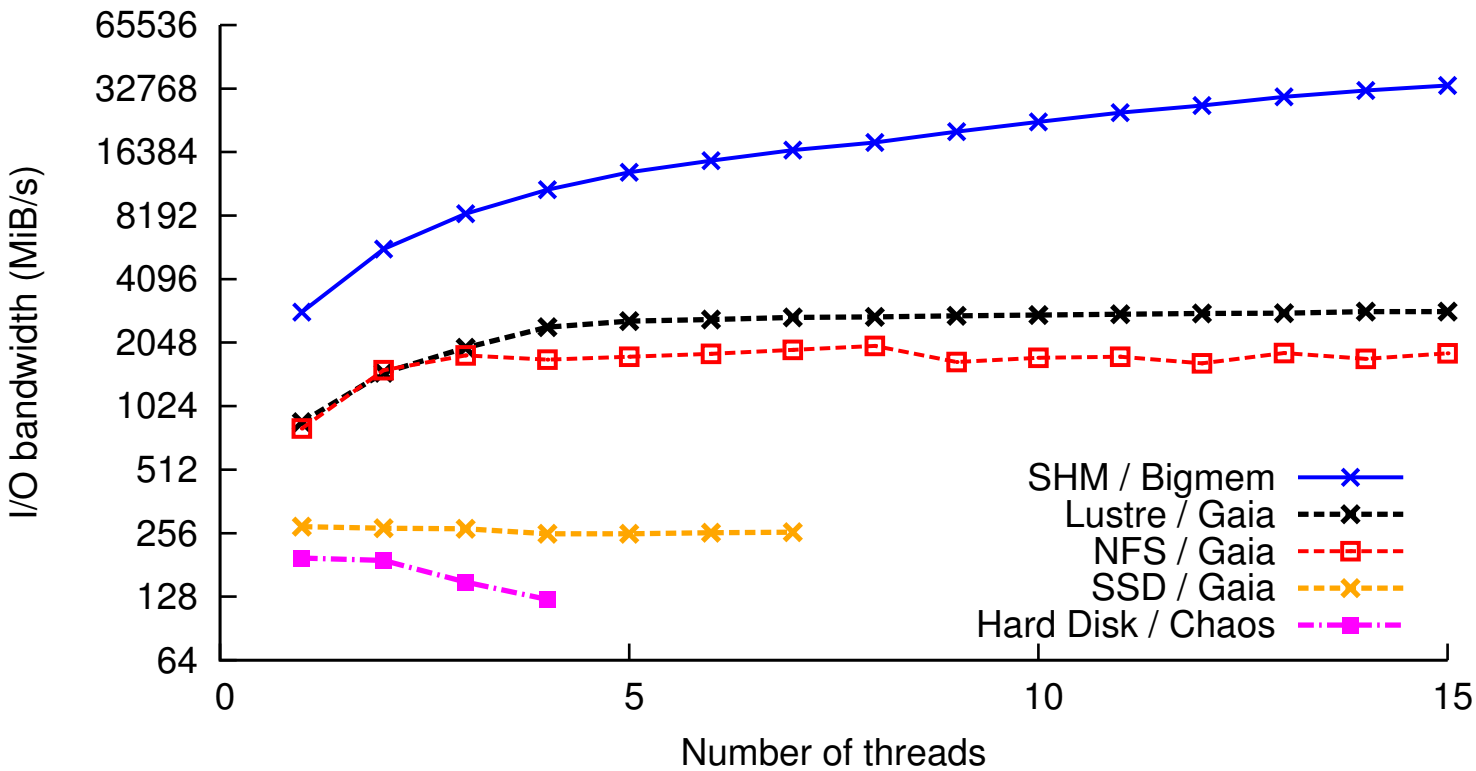
Where can I store and manipulate my data?

- **Shared** storage
  - ↪ `NFS` – **not scalable** $\simeq$ 1.5 GB/s (R) $\mathcal{O}$(200 TB)
  - ↪ `Lustre` – **scalable** $\simeq$ 3 GB/s (R) $\mathcal{O}$(200 TB)
- **Local** storage
  - ↪ local file system (`/tmp`) $\mathcal{O}$(200 GB)
    - over HDD $\simeq$ 100 MB/s
    - over SDD $\simeq$ 400 MB/s
  - ↪ RAM (`/dev/shm`) $\simeq$ 30 GB/s (R) $\mathcal{O}$(20 GB)

$\Rightarrow$ **In all cases:** small I/Os really **kill** storage performances

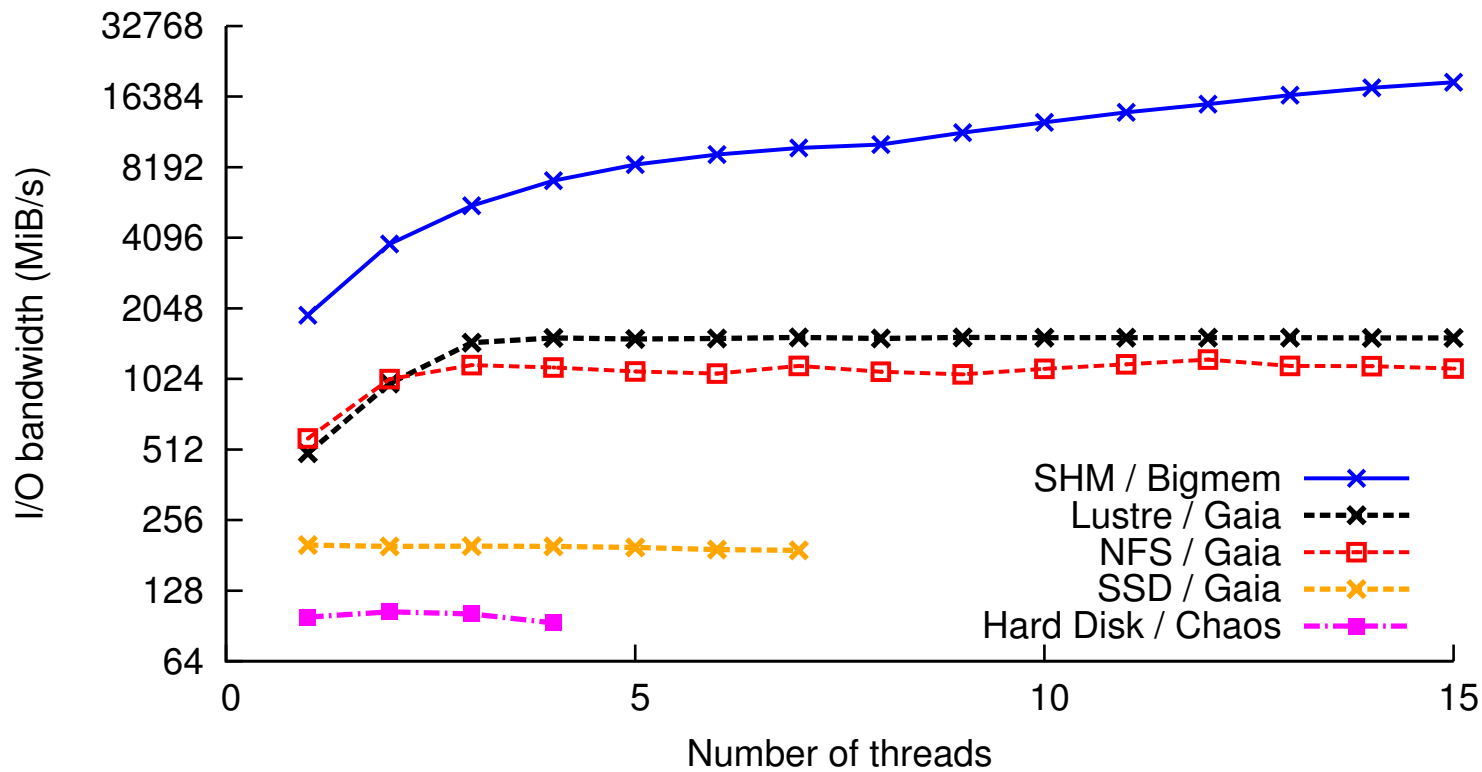
Storage performances

- Based on IOR or IOZone, reference I/O benchmarks **Read**



Storage performances

- Based on IOR or IOZone, reference I/O benchmarks **Write**



Speed Expectation on Data Transfer

http://fasterdata.es.net/

- How long to transfer **1 TB** of data across various speed networks?

Network	Time
10 Mbps	300 hrs (12.5 days)
100 Mbps	30 hrs
1 Gbps	3 hrs
10 Gbps	20 minutes

- **(Again)** small I/Os really **kill** performances
  - ↪ **Ex**: transferring 80 TB for the backup of `ecosystem_biology`
  - ↪ same rack, 10Gb/s. 4 weeks ⟶ 63TB transfer...

Speed Expectation on Data Transfer

http://fasterdata.es.net/

Data set size	1 Minute	5 Minutes	20 Minutes	1 Hour
	Time to transfer			
10PB	166.67 TB/sec	33.33 TB/sec	8.33 TB/sec	2.78 TB/sec
1PB	16.67 TB/sec	3.33 TB/sec	833.33 GB/sec	277.78 GB/sec
100TB	1.67 TB/sec	333.33 GB/sec	83.33 GB/sec	27.78 GB/sec
10TB	166.67 GB/sec	33.33 GB/sec	8.33 GB/sec	2.78 GB/sec
1TB	16.67 GB/sec	3.33 GB/sec	833.33 MB/sec	277.78 MB/sec
100GB	1.67 GB/sec	333.33 MB/sec	83.33 MB/sec	27.78 MB/sec
10GB	166.67 MB/sec	33.33 MB/sec	8.33 MB/sec	2.78 MB/sec
1GB	16.67 MB/sec	3.33 MB/sec	0.83 MB/sec	0.28 MB/sec
100MB	1.67 MB/sec	0.33 MB/sec	0.08 MB/sec	0.03 MB/sec

Legend:

- Requires less than 100Mbps throughput
- Requires between 100Mbps and 10Gbps throughput
- Requires between 10Gbps and 100Gbps throughput
- Requires more than 100Gbps throughput

Note: Kilo, Mega, etc. are in SI units. E.g. 1KB is 1000 bytes, not 1024 bytes

Speed Expectation on Data Transfer

http://fasterdata.es.net/

Data set size	8 Hours	24 Hours	7 Days	30 Days
1XB	34.72 TB/sec	11.57 TB/sec	1.65 TB/sec	385.80 GB/sec
100PB	3.47 TB/sec	1.16 TB/sec	165.34 GB/sec	38.58 GB/sec
10PB	347.22 GB/sec	115.74 GB/sec	16.53 GB/sec	3.86 GB/sec
1PB	34.72 GB/sec	11.57 GB/sec	1.65 GB/sec	385.80 MB/sec
100TB	3.47 GB/sec	1.16 GB/sec	165.34 MB/sec	38.58 MB/sec
10TB	347.22 MB/sec	115.74 MB/sec	16.53 MB/sec	3.86 MB/sec
1TB	34.72 MB/sec	11.57 MB/sec	1.65 MB/sec	0.39 MB/sec
100GB	3.47 MB/sec	1.16 MB/sec	0.17 MB/sec	0.04 MB/sec
10GB	0.35 MB/sec	0.12 MB/sec	0.02 MB/sec	0.00 MB/sec
Time to transfer				

Legend:

- Requires less than 100Mbps throughput
- Requires between 100Mbps and 10Gbps throughput
- Requires between 10Gbps and 100Gbps throughput
- Requires more than 100Gbps throughput

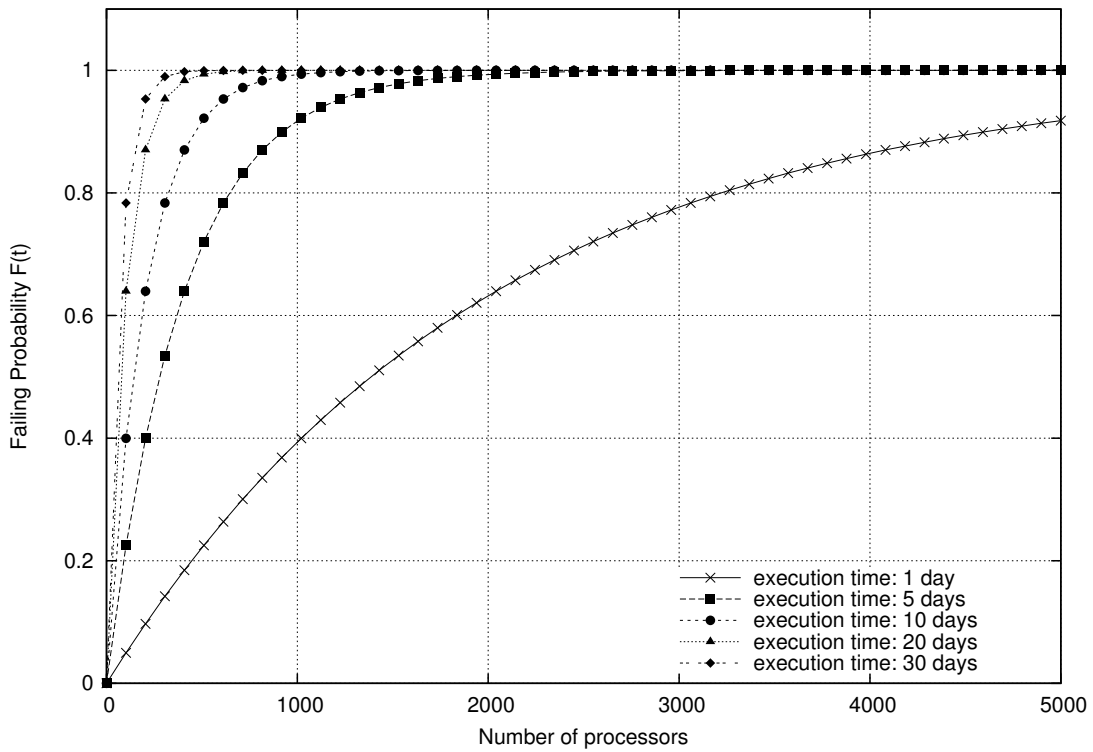
Note: Kilo, Mega, etc. are in SI units. E.g. 1KB is 1000 bytes, not 1024 bytes

Fault Tolerance

- Cluster maintenance from time to time

Fault Tolerance

- Cluster maintenance from time to time
- Reliability vs. Crash Faults in Distributed systems



Fault Tolerance

- Cluster maintenance from time to time
- Reliability vs. Crash Faults in Distributed systems
- Fault Tolerance general strategy: checkpoint/rollback
  - ↪ assumes a way to save the state of your program
  - ↪ hints: OAR `-signal -checkpoint -idempotent...`, BLCR
  - ↪ combine best-effort jobs with checkpointing (`http://git.io/c-dn1A`)

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)

http://tinyurl.com/ulhpc-ganglia-testcase

- **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**

Guidelines

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},
  OPTpages =     {},
  month =        {July},
  address =      {Bologna, Italy},
  publisher =    {IEEE},
}
```

Thank you for your attention....

http://hpc.uni.lu

1. **Overview of the Main HPC Components**
2. **The UL HPC platform**
   - Overview
   - Platform Deployment with FAI and Puppet
   - Monitoring
   - Statistics & Milestones
3. **UL HPC in Practice: Toward an [Efficient] Win-Win Usage**
   - General considerations
   - SSH Access
   - The OAR Batch Scheduler
   - Available Software and Environment Modules
4. **Last Challenges**
   - Memory bottleneck
   - Effective Storage and Memory Management
   - Reporting problems and Promoting the Platform

Contacts:

hpc-sysadmins@uni.lu

UNIVERSITÉ DU LUXEMBOURG