HPC platforms @ UL

Overview (as of 2013) and Usage

http://hpc.uni.lu

S. Varrette, PhD.

University of Luxembourg, Luxembourg

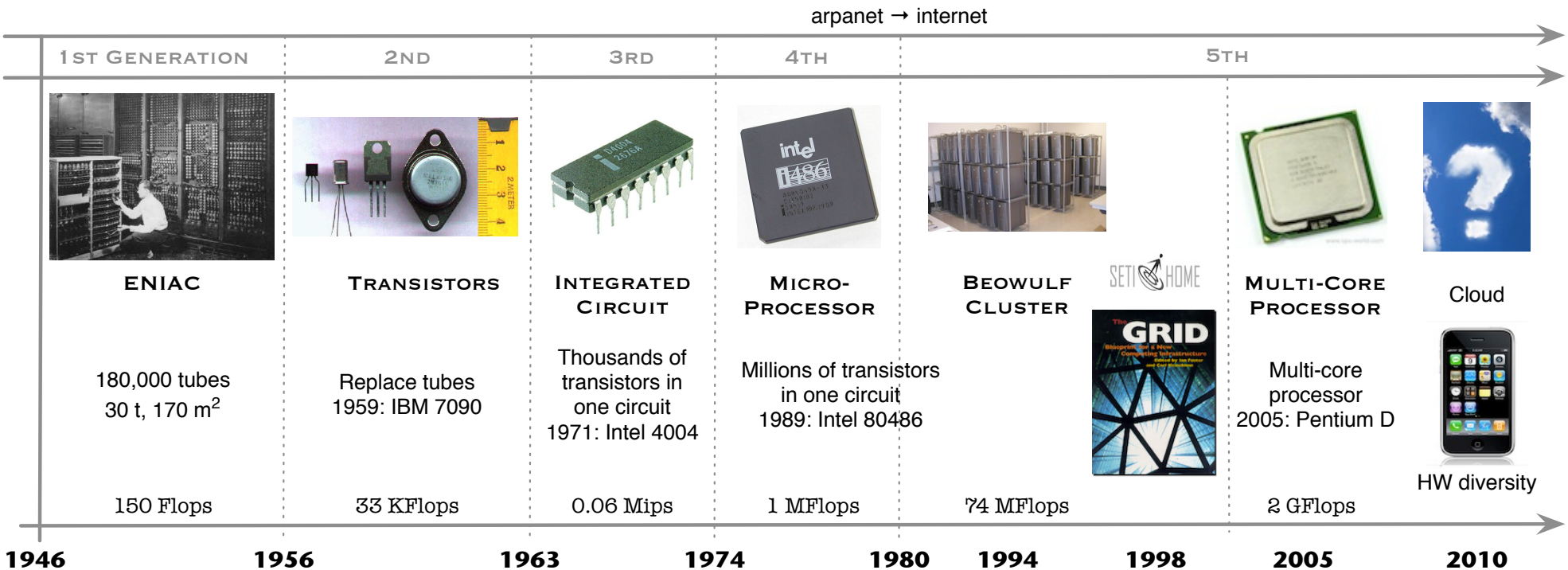
uni.lu
UNIVERSITÉ DU
LUXEMBOURG

Summary

uni.lu
UNIVERSITÉ DU
LUXEMBOURG

Summary

Evolution of Computing Systems



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)

27.363**T**Flops

(291computing nodes, 2944cores)

- Increase **storage** capacity

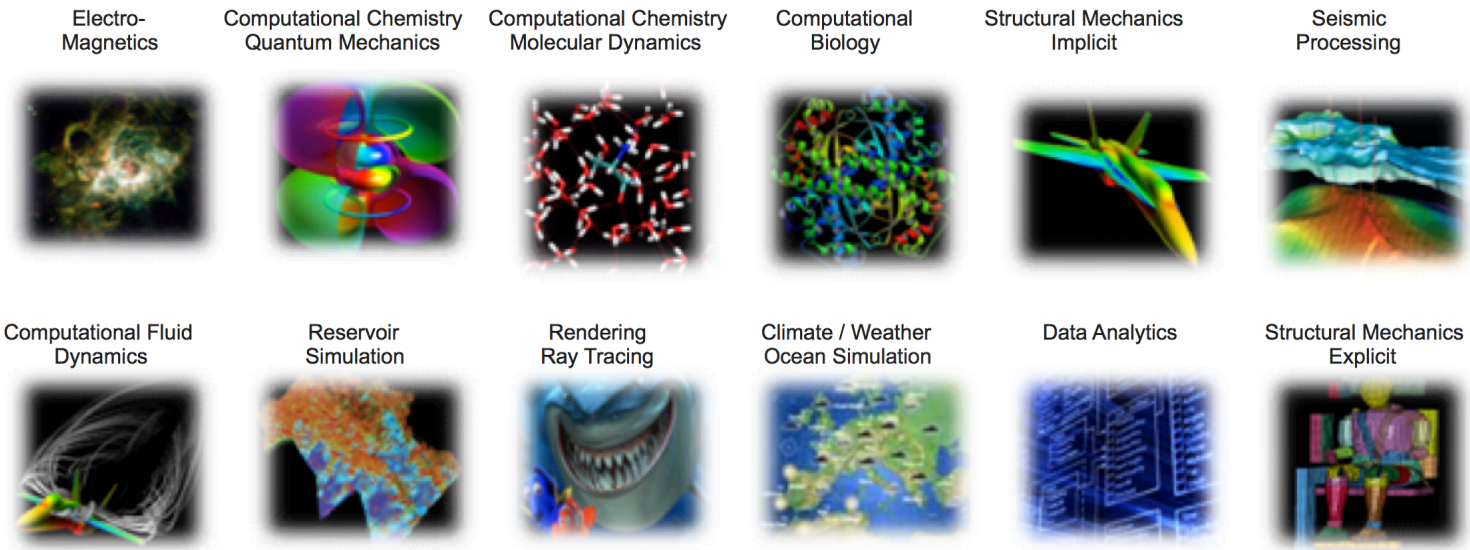
2TB (1 disk)

1042TB raw(444disks)

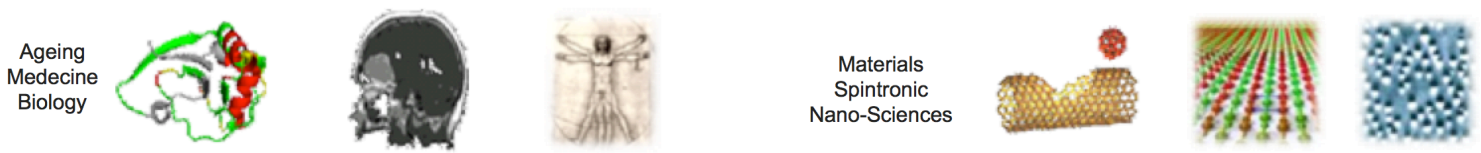
- Communicate **faster** 1 GbE (1 Gb/s) vs Infiniband QDR (40 Gb/s)

HPC at the Heart of our Daily Life

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



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



HPC at the Heart of National Strategies

USA	R&D program: **1G$/y** for HPC	2005 → 2011
	↪ 2014 DOE R&D budget: 12.7G$/y	
Japan	**800 M€** (Next Generation Supercomputer Program)	2008 → 2011
	↪ K supercomputer, first to break the 10 Pflops mark	
China	massive investments (exascale program) since 2006	
Russia	**1.5G$** for the exascale program (T-Platform)	
India	**1G$** program for exascale Indian machine	2012
EU	**1.58G$** program for exascale	2012

HPC at the Heart of National Strategies

USA	R&D program: **1G$/y** for HPC	2005 → 2011
	↪ 2014 DOE R&D budget: 12.7G$/y	
Japan	**800 M€** (Next Generation Supercomputer Program)	2008 → 2011
	↪ K supercomputer, first to break the 10 Pflops mark	
China	massive investments (exascale program) since 2006	
Russia	**1.5G$** for the exascale program (T-Platform)	
India	**1G$** program for exascale Indian machine	2012
EU	**1.58G$** program for exascale	2012

2012: **11.1G$** profits in the HPC technical server industry

↪ Record Revenues (10.3G$ in 2011) +7.7% [Source: IDC]

Summary

HPC Components: [GP]CPU

CPU

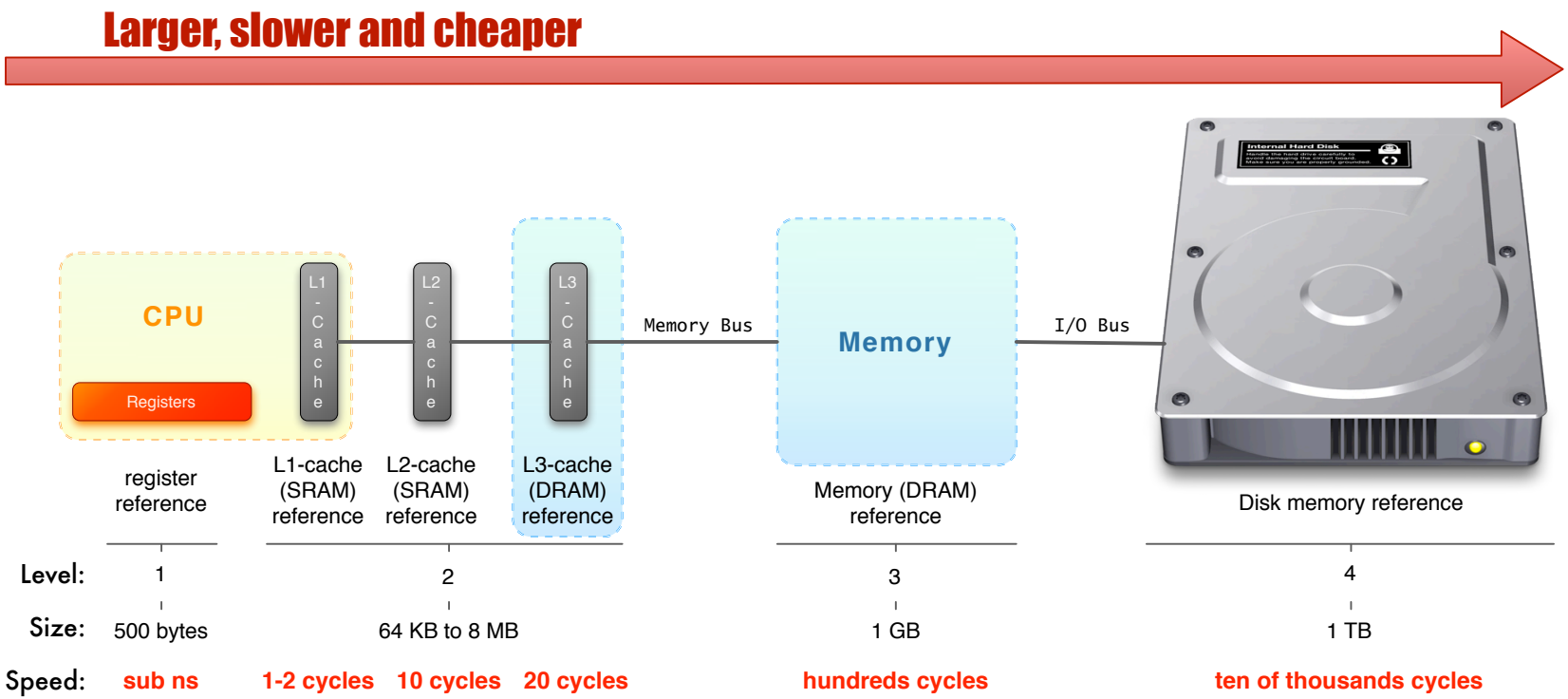
- Always multi-core
- Ex: Intel Core i7-970 (July 2010) $R_{peak} \simeq 100$ GFlops (DP)
  - ↪ 6 cores @ 3.2GHz (32nm, 130W, 1170 millions transistors)

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

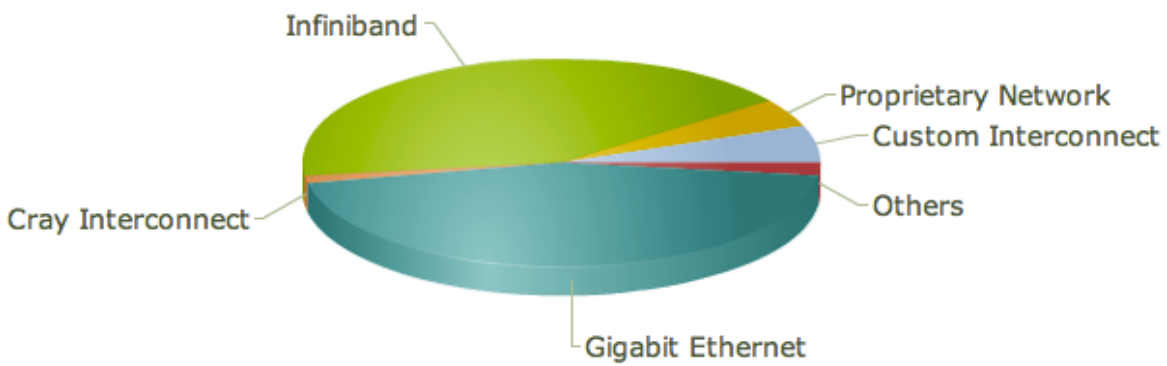


- SSD R/W: 560 MB/s; 85000 IOps **1500 €/TB**
- HDD (SATA @ 7,2 krpm) R/W: 100 MB/s; 190 IOps **150 €/TB**

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

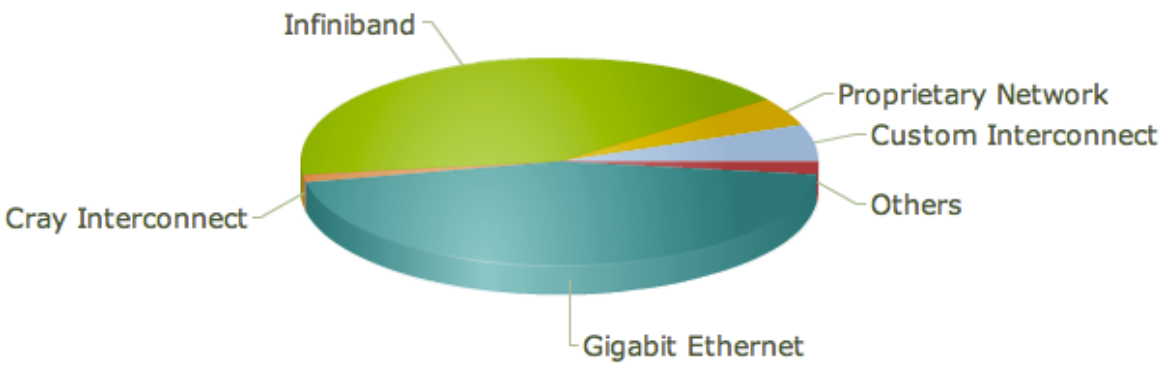
Technology	Effective Bandwidth		Latency
Gigabit Ethernet	1 Gb/s	125 MB/s	40μs to 300μs
Myrinet (Myri-10G)	9.6 Gb/s	1.2 GB/s	2.3μs
10 Gigabit Ethernet	10 Gb/s	1.25 GB/s	4μs to 5μs
Infiniband QDR	40 Gb/s	5 GB/s	1.29μs to 2.6μs
SGI NUMAlink	60 Gb/s	7.5 GB/s	1μs



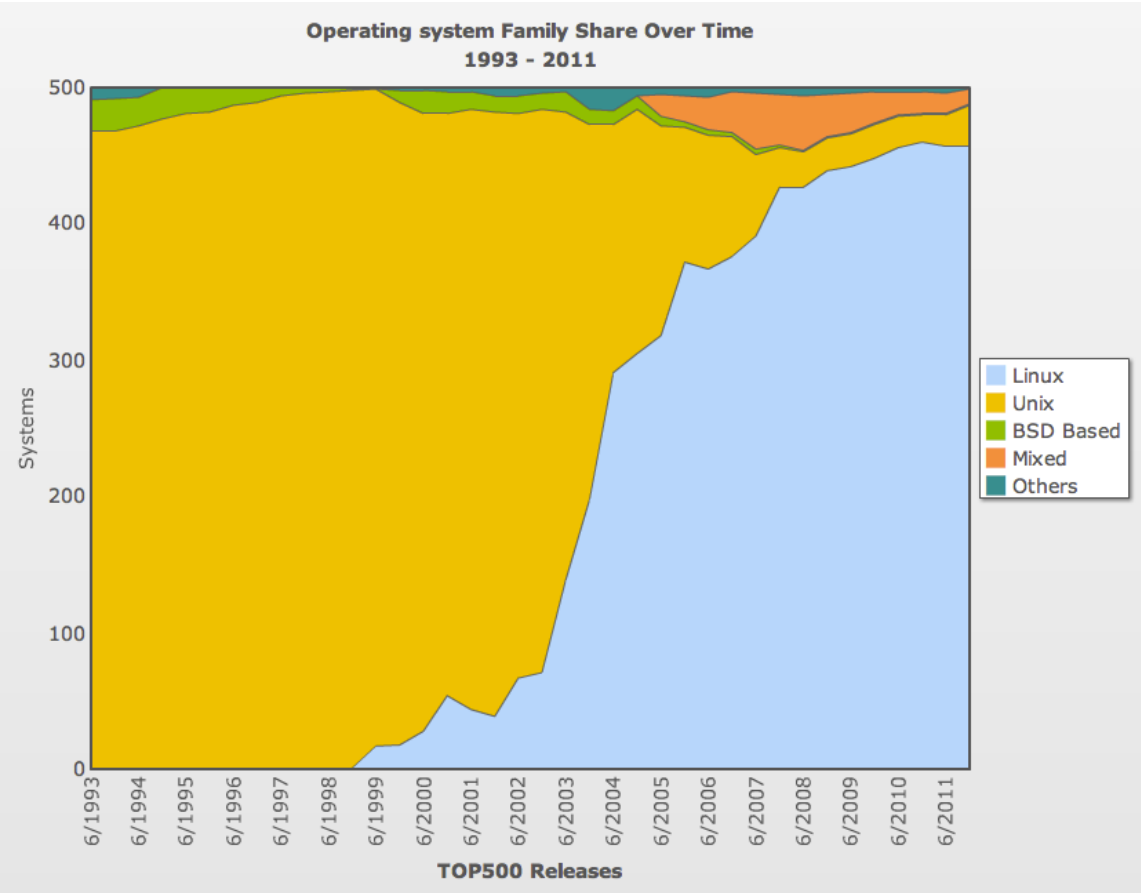
HPC Components: Interconnect

- **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



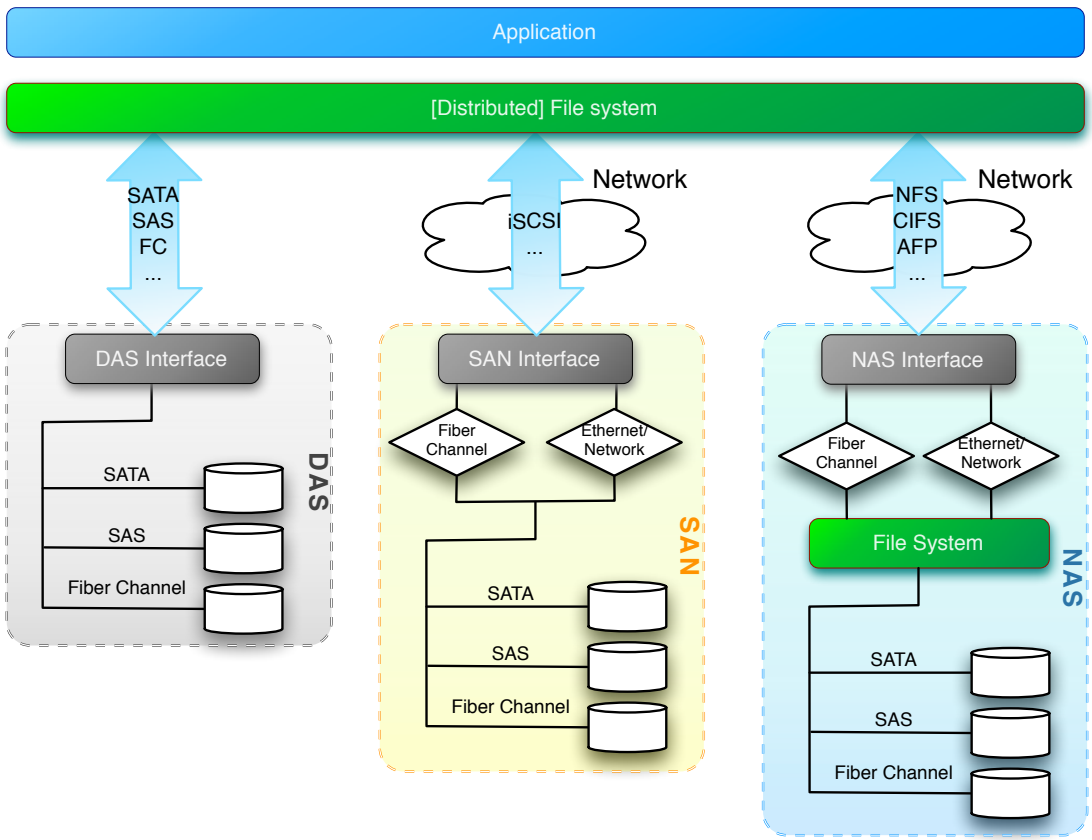
- Mainly Linux-based OS (91.4%) (Top500, Nov 2011)
- ... or Unix based (6%)
- Reasons:
  - ↪ stability
  - ↪ prone to devels

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.

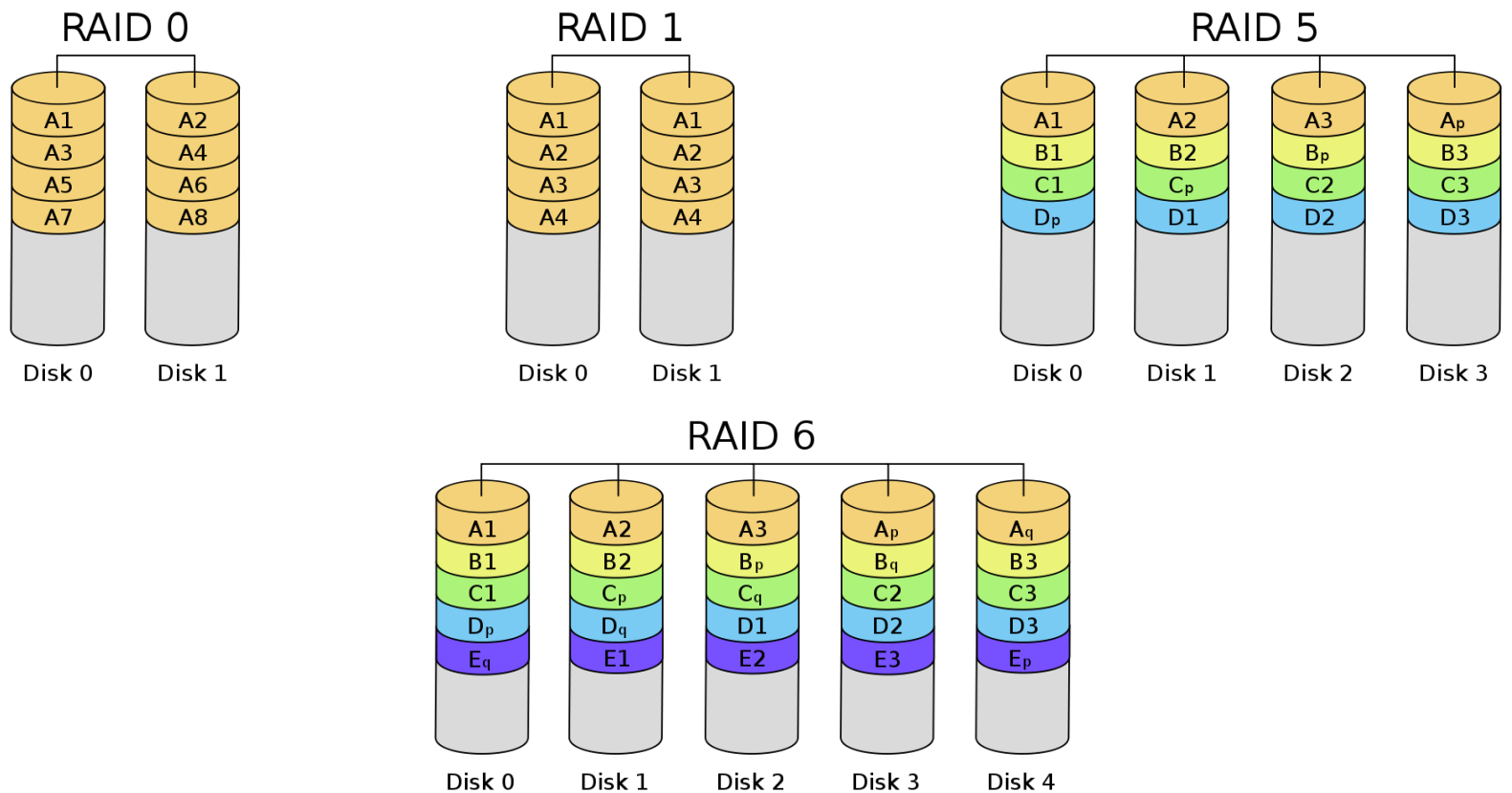
HPC Components: Data Management

Storage architectural classes & I/O layers



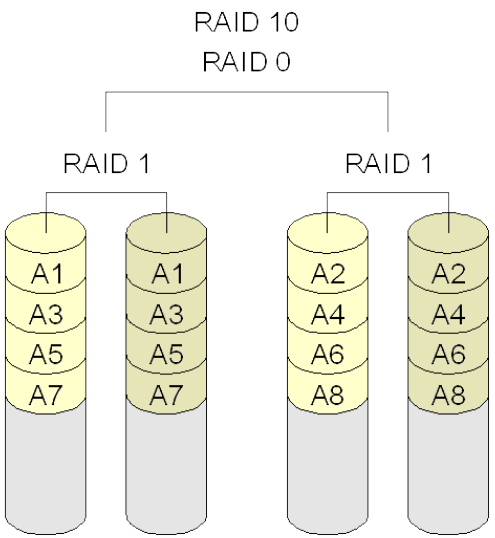
HPC Components: Data Management

RAID standard levels



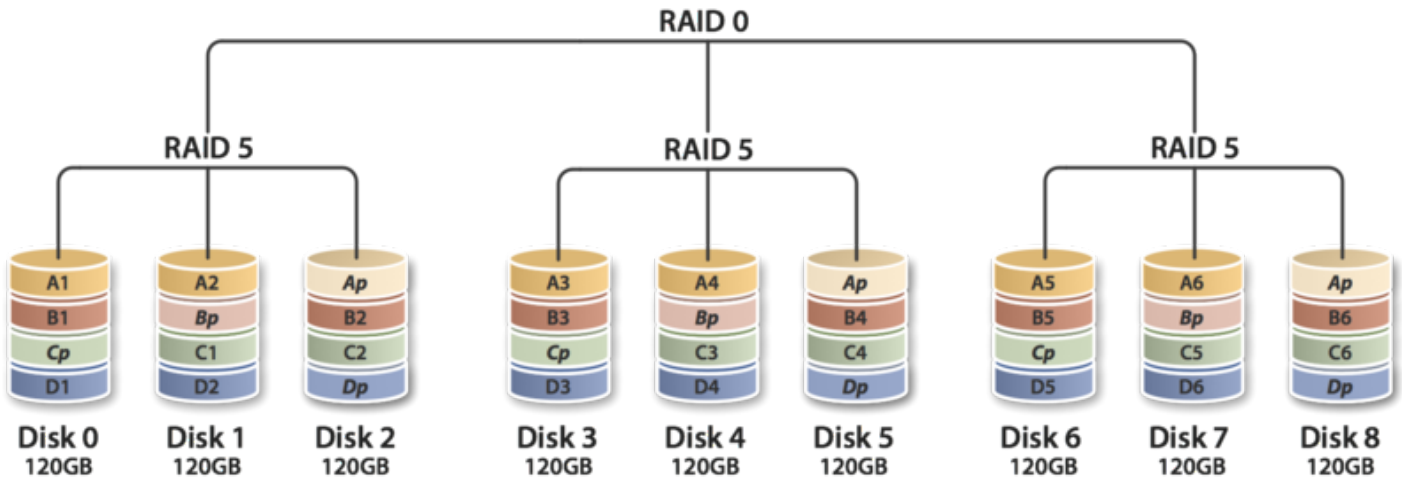
HPC Components: Data Management

RAID combined levels



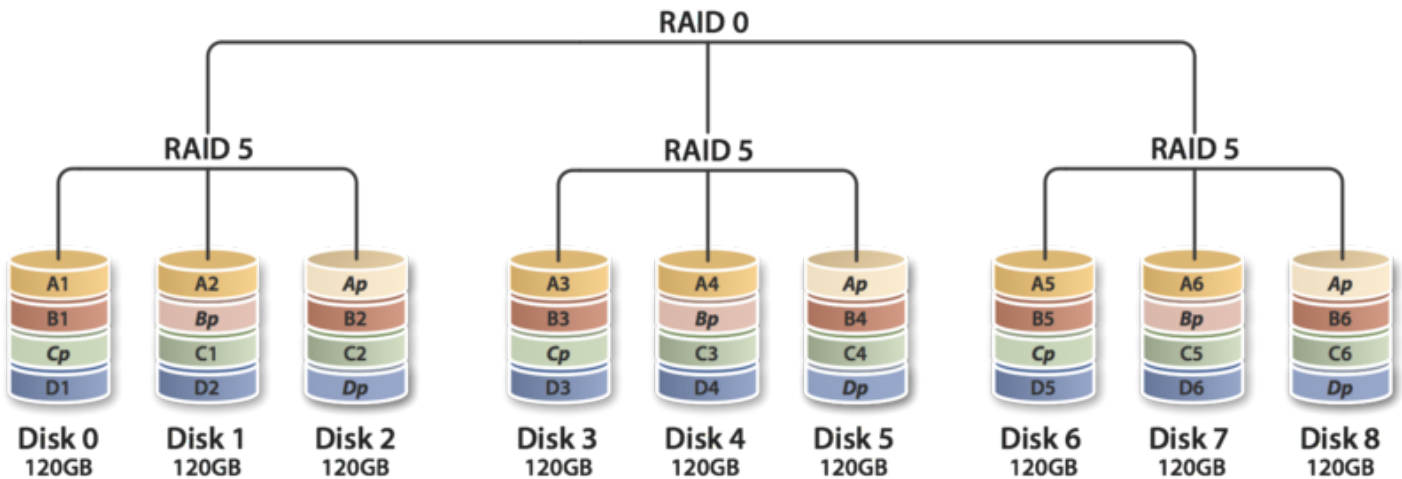
HPC Components: Data Management

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...

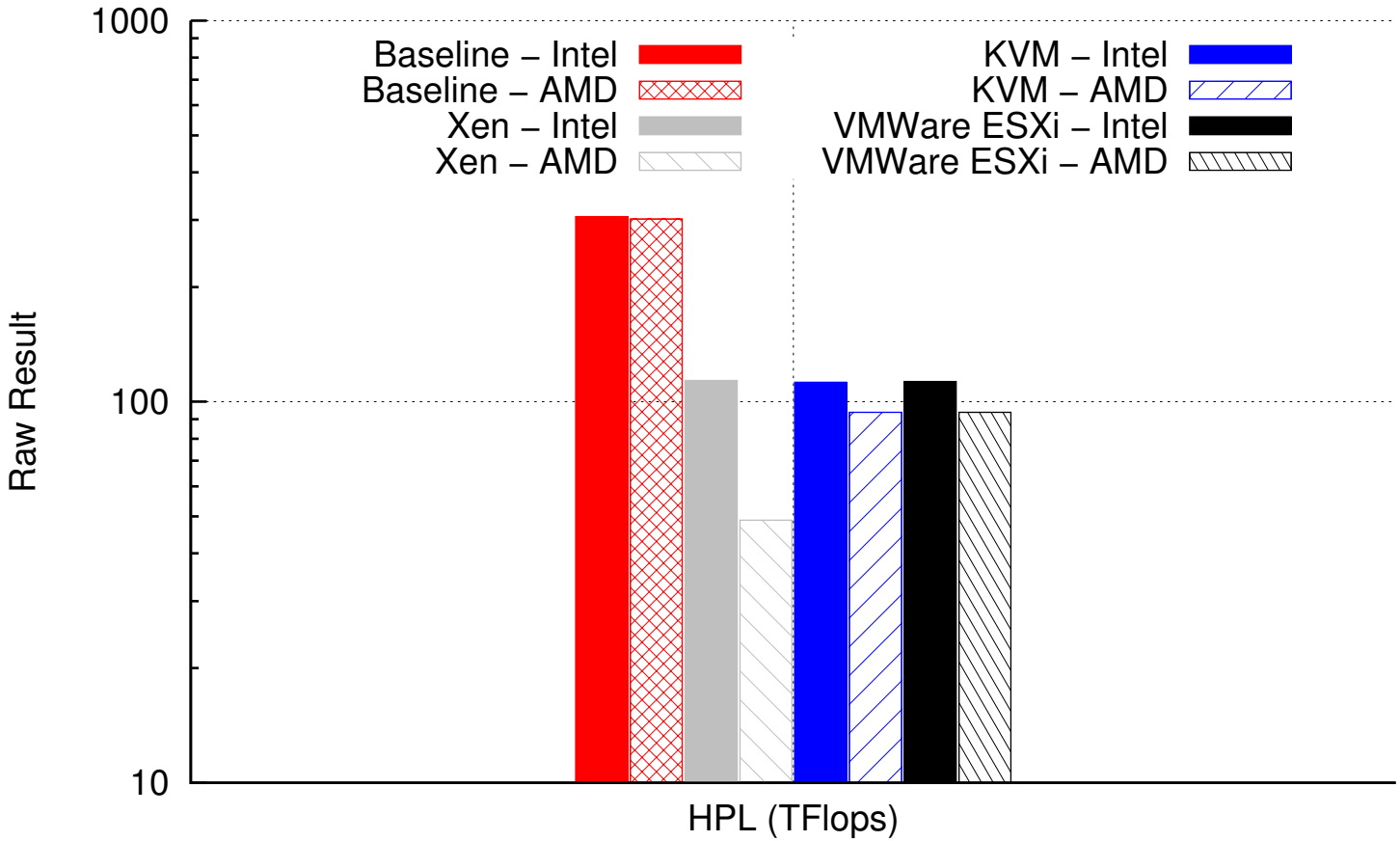
uni.lu
UNIVERSITÉ DU LUXEMBOURG

Summary

CC characteristics in HPC context

- **Horizontal scalability**: perfect for replication/ HA (High Availability)
  - ↪ best suited for runs with minimal communication and I/O
  - ↪ nearly useless for true parallel/distributed HPC runs
- **Cloud Data storage**
  - ↪ Data locality enforced for performance
  - ↪ Data outsourcing vs. legal obligation to keep data local
  - ↪ Accessibility, security challenges
- Virtualization layer lead to **decreased performances**
  - ↪ huge overhead induced on I/O + no support of IB [Q|E|F]DR
- **Cost effectiveness?**

Virtualization layer overhead



[1] M. Guzek, S. Varrette, V. Plugaru, J. E. Sanchez, and P. Bouvry. *A Holistic Model of the Performance and the Energy-Efficiency of Hypervisors in an HPC Environment*. LNCS EE-LSDS'13, Apr 2013.

EC2 [Poor] performances

- Illustration on UnixBench index (the higher the better) Gist
  - ↪ dedicated server 10× faster than `m1.small` instance (both 65$/m)
  - ↪ IO throughput 29× higher for just $13 more a month.

	m1.small	m1.large	m1.xlarge	c1.xlarge	Joe's Dedicated	Hetzner EX 4S
Monthly Price	$65	$250	$500	$500	$65	$78
Architecture (Ubuntu)	32-bit	64-bit	64-bit	64-bit	64-bit	64-bit
CPUs/Cores	1	2	4	8	2	8
RAM (GB)	1.7	7.5	15	7	4	32
Index (1 Thread)	116	357.5	438.1	494.4	777.6	1803.3
Index (2 Threads)		571.5			1210.5	
Index (4 Threads)			1070.5			
Index (8 Threads)				1746.3		6696.7

EC2 [Poor] performances

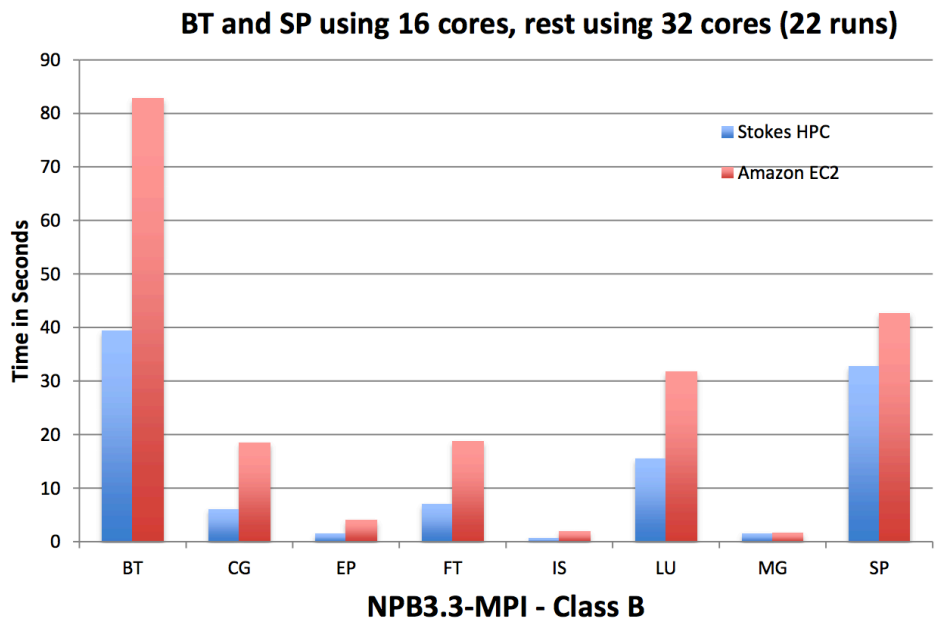
- NAS Parallel Benchmark (NPB) compared to ICHEC [1]

	Amazon EC2	Stokes HPC
Compute Node	23 GB of memory, 2 x Intel Xeon X5570, quad-core "Nehalem"	24 GB memory, 2 x Intel Xeon E5650, hex-core "Westmere"
Connectivity	10 Gigabit Ethernet	ConnectX Infiniband (DDR)
OS	Ubuntu, 64-bit platform	Open-SUSE, 64-bit platform
Resource manager	Sun Grid Engine	Torque
Compilers & libraries	Intel C, Intel Fortran, Intel MKL, Intel MVAPICH2	Intel C, Intel Fortran, Intel MKL, Intel MVAPICH2

[1] K Iqbal and E. Brazil. *HPC vs. Cloud Benchmarking. An empirical evaluation of the performance and cost metrics*. eFISCAL Workshop, 2012

EC2 [Poor] performances

- NAS Parallel Benchmark (NPB) compared to ICHEC [1]

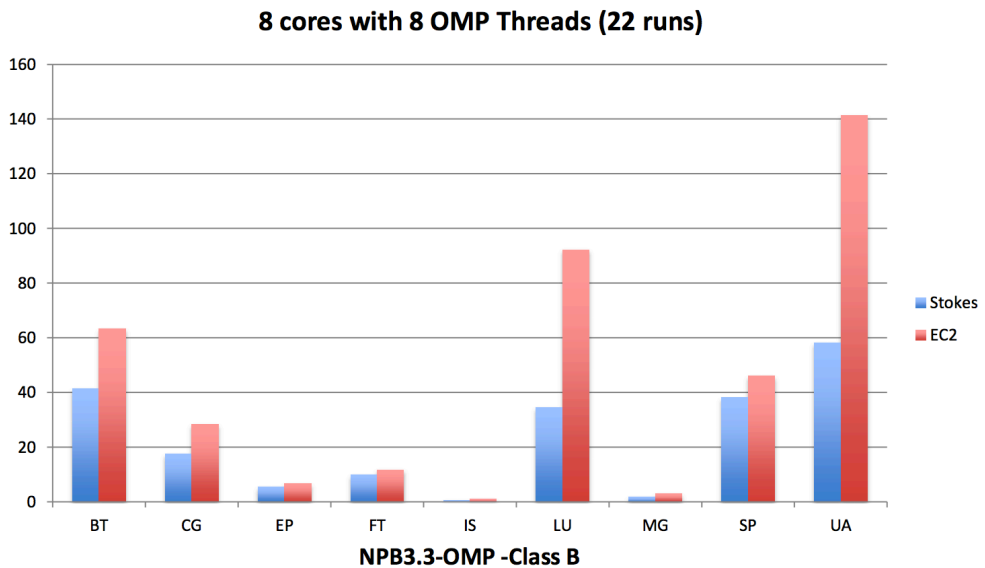


The average performance loss was **48.42%**

[1] K Iqbal and E. Brazil. *HPC vs. Cloud Benchmarking. An empirical evaluation of the performance and cost metrics*. eFISCAL Workshop, 2012

EC2 [Poor] performances

- NAS Parallel Benchmark (NPB) compared to ICHEC [1]



The average performance loss was **37.26%**

[1] K Iqbal and E. Brazil. *HPC vs. Cloud Benchmarking. An empirical evaluation of the performance and cost metrics*. eFISCAL Workshop, 2012

CC "Cost effectiveness"...

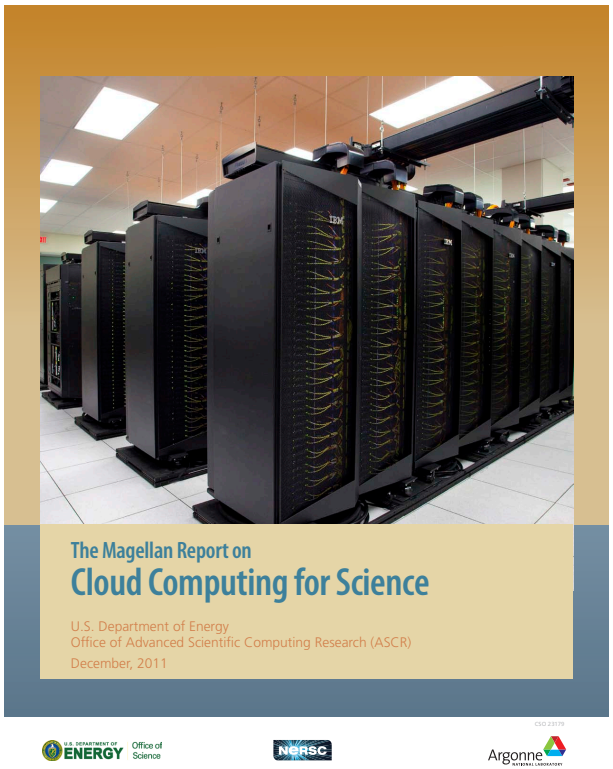
- Amazon HPC instances (as `cc2.8xlarge`) inherent restrictions
  - ↪ Cluster Compute Eight Extra Large Instance
  - ↪ 10 GbE interconnect only
  - ↪ max 240 IPs
- "Hardware cost coverage"
  - ↪ `chaos+gaia` usage: 11,154,125 CPUhour (**1273 years**) since 2007
  - ↪ **15,06M$** on EC2* **vs. 4 M€** cumul. HW investment

  *`cc2.8xlarge` Cluster Compute Eight Extra Large Instance EU
- Data transfer cost (up to 0.12$ per GB) and performances
  - ↪ 614,4$ to retrieve a single 5TB microscopic image
  - ↪ Internet (GEANT network) to reach Ireland Amazon data center

UNIVERSITÉ DU LUXEMBOURG

Magellan Report: CC for Science

Download



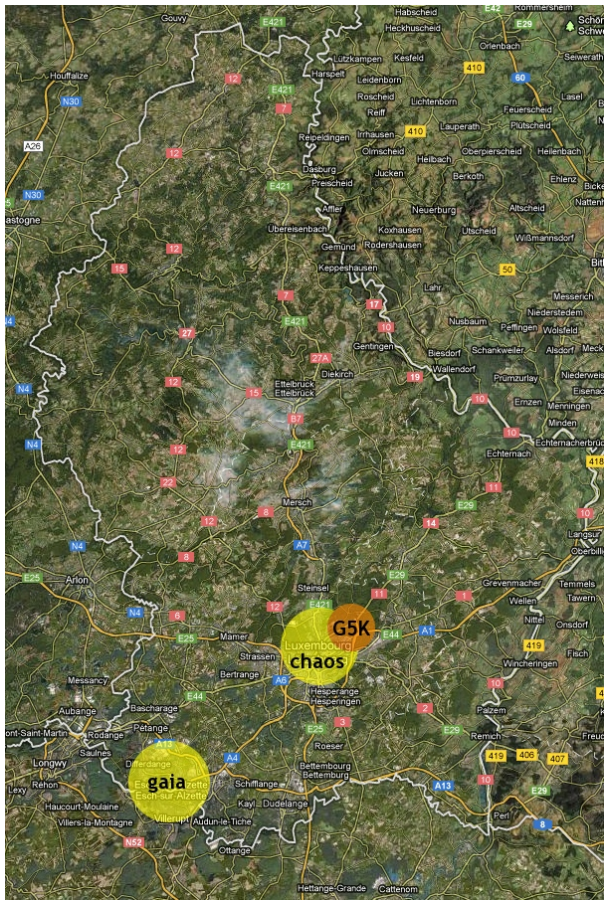
- Virtualization power **of value for scientists**
  - ↪ enable fully customized environments
  - ↪ flexible resource management
- Requires **significant** prog./sysadmin support
- **Significant gaps and challenges** exist
  - ↪ managing VMs, workflows, data, security...
- Public clouds can be expansive
  - ↪ **more expansive than in-house** large systems
- Ideal for resource consolidation

- HPC centers should continuously benchmark their computing cost
  - ↪ DOE centers are cost competitive, (3-7× less expensive) when compared to commercial cloud providers

UNIVERSITÉ DU LUXEMBOURG

Summary

UL HPC platforms at a glance (2013)



- **2 geographic sites**
  - ↪ Kirchberg campus (AS.28, CS.43)
  - ↪ LCSB building (Belval)
- **4 clusters**: `chaos+gaia`, `granduc`, `nyx`.
  - ↪ **291 nodes, 2944 cores, 27.363 TFlops**
  - ↪ **1042TB shared storage** (raw capa.)
- **3 system administrators**
- **4,091,010€** **(Cumul. HW Investment)** since 2007
  - ↪ Hardware acquisition only
  - ↪ 2,122,860€ (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€



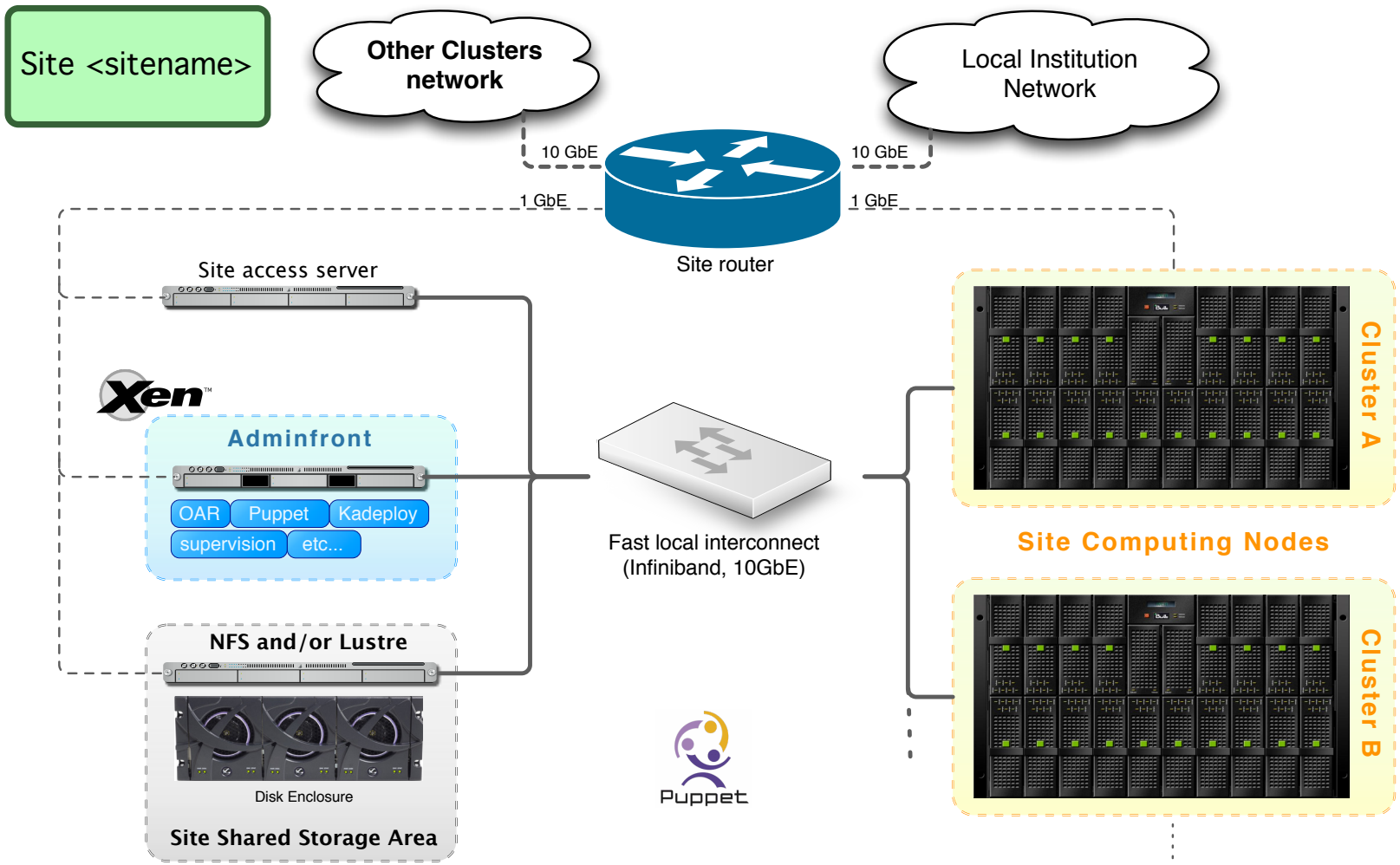
- **2011** LCSB 6^{th} floor (Belval) 14 racks, 112 m^2, $\simeq$ 1,100,000€


UL HPC Computing Nodes

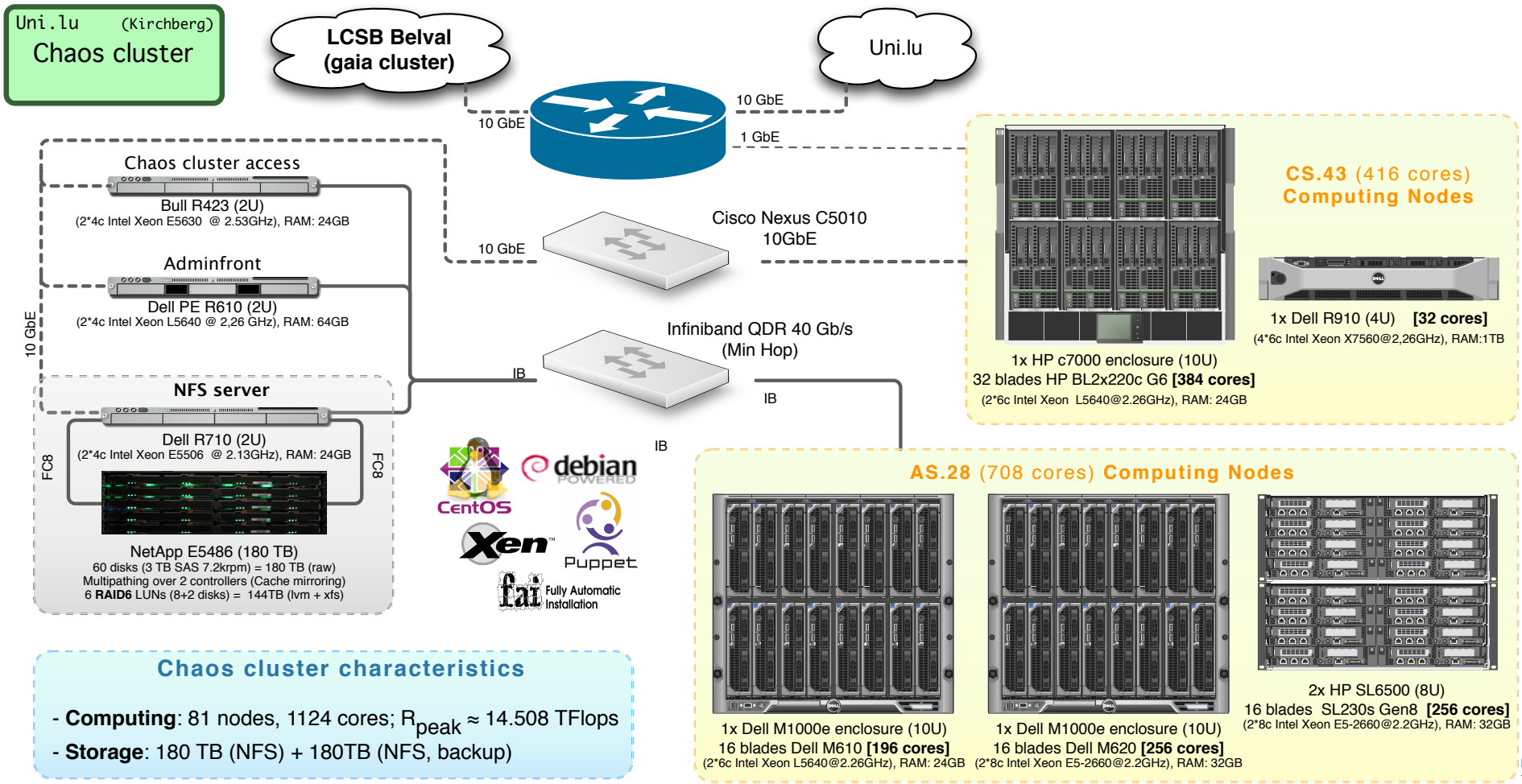
	Date	Vendor	Proc. Description		#N	#C	R_{peak}
chaos	2010	HP	Intel Xeon L5640@2.26GHz	2 × 6C,24GB	32	384	3.472 TFlops
	2011	Dell	Intel Xeon L5640@2.26GHz	2 × 6C,24GB	16	192	1.736 TFlops
	2012	Dell	Intel Xeon X7560@2,26GHz	4 × 6C, 1TB	1	32	0.289 TFlops
	2012	Dell	Intel Xeon E5-2660@2.2GHz	2 × 8C,32GB	16	256	4.506 TFlops
	2012	HP	Intel Xeon E5-2660@2.2GHz	2 × 8C,32GB	16	256	4.506 TFlops
				`chaos` **TOTAL:**	**81**	**1124**	**14.508 TFlops**
gaia	2011	Bull	Intel Xeon L5640@2.26GHz	2 × 6C,24GB	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	Viridis	ARM A9 Cortex@1.1GHz	1 × 4C,4GB	96	384	0.422 TFlops
				`gaia` **TOTAL:**	**170**	**1440**	**9.82 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**	**2.944 TFlops**
Testing cluster:							
nyx	2012	Dell	Intel Xeon E5-2420@1.90GHz	1 × 6C,32GB	2	12	0.091 TFlops

TOTAL: 291 nodes, 2944 cores, 27.363 TFlops

UL HPC: General cluster organization



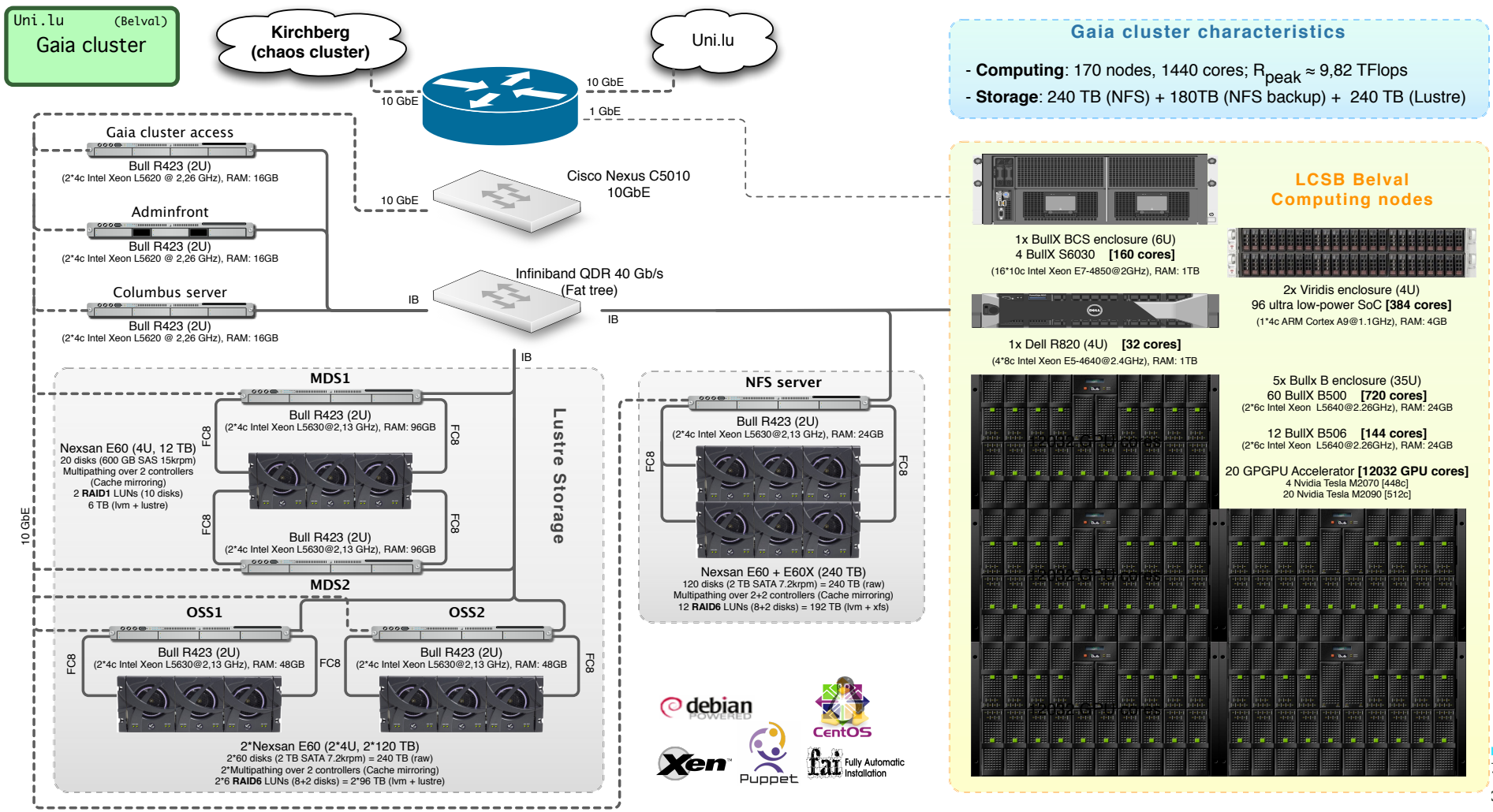
Ex: The chaos cluster



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



Gaia cluster characteristics

- **Computing**: 170 nodes, 1440 cores; $R_{peak} \approx 9{,}82$ TFlops
- **Storage**: 240 TB (NFS) + 180TB (NFS backup) + 240 TB (Lustre)

LCSB Belval Computing nodes

1x BullX BCS enclosure (6U)
4 BullX S6030 **[160 cores]**
(16*10c Intel Xeon E7-4850@2GHz), RAM: 1TB

2x Viridis enclosure (4U)
96 ultra low-power SoC **[384 cores]**
(1*4c ARM Cortex A9@1.1GHz), RAM: 4GB

1x Dell R820 (4U) **[32 cores]**
(4*8c Intel Xeon E5-4640@2.4GHz), RAM: 1TB

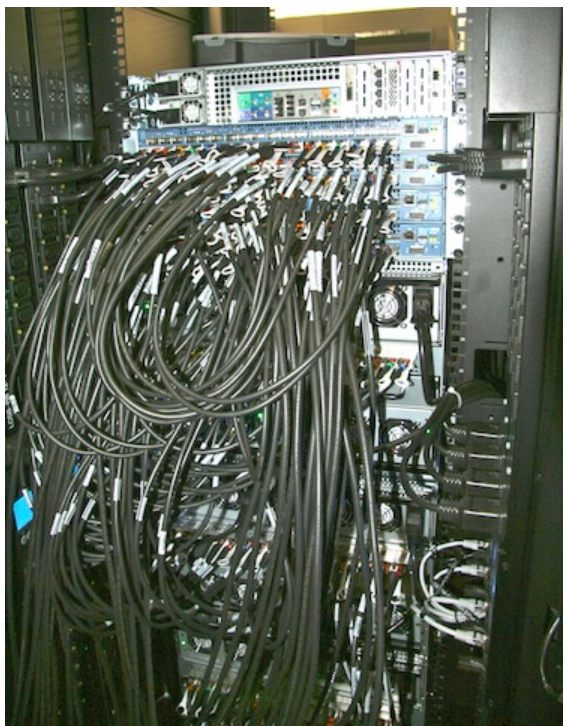
5x BullX B enclosure (35U)
60 BullX B500 **[720 cores]**
(2*6c Intel Xeon L5640@2.26GHz), RAM: 24GB

12 BullX B506 **[144 cores]**
(2*6c Intel Xeon L5640@2.26GHz), RAM: 24GB

20 GPGPU Accelerator **[12032 GPU cores]**
4 Nvidia Tesla M2070 [448c]
20 Nvidia Tesla M2090 [512c]

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.

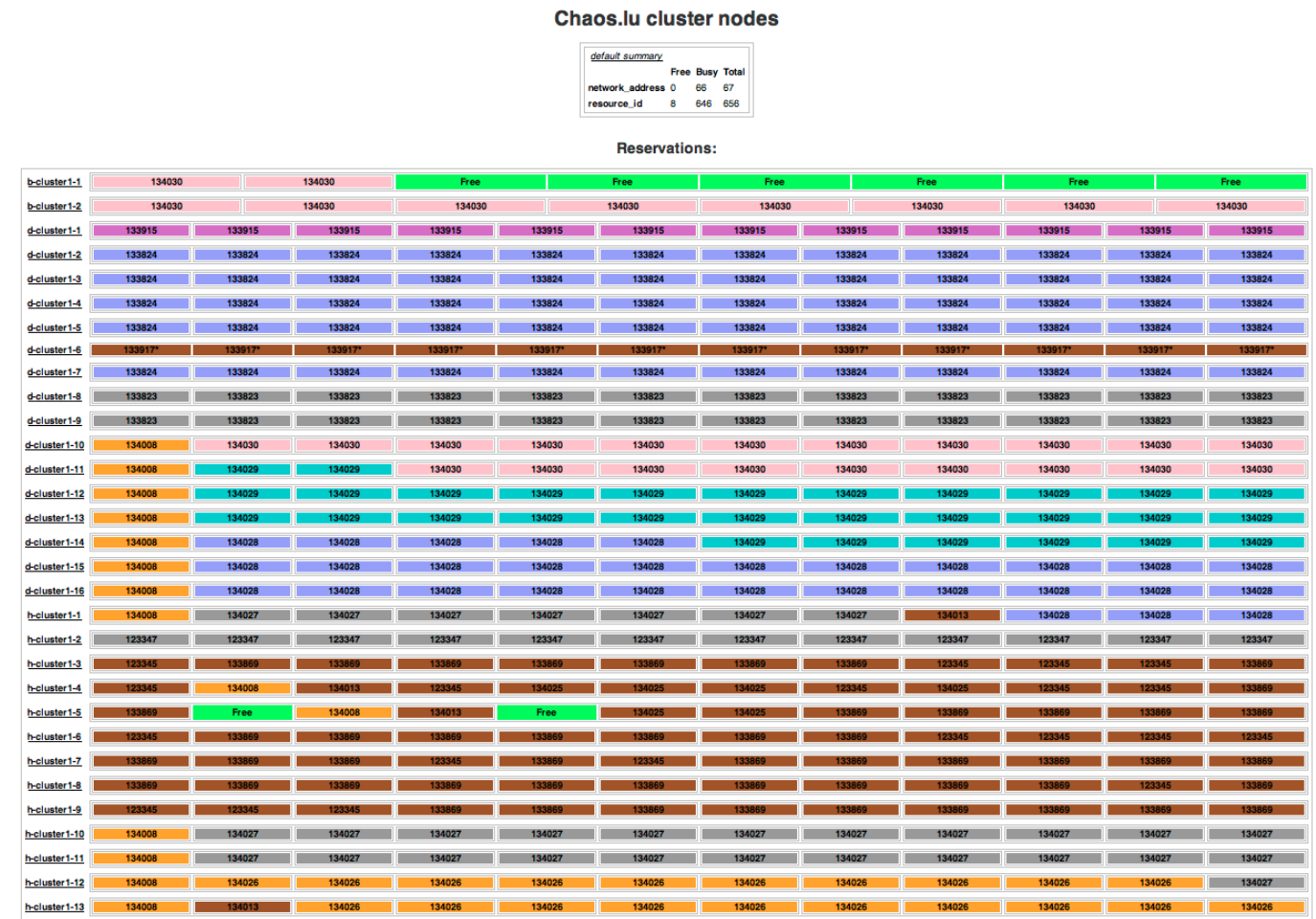
HPC in the Grande region and Around

			TFlops	TB	FTEs
Country	**Name/Institute**	**#Cores**	**R_{peak}**	**Storage**	**Manpower**
Luxembourg	UL	2944	27.363	1042	3
	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

Platform Monitoring

Monika

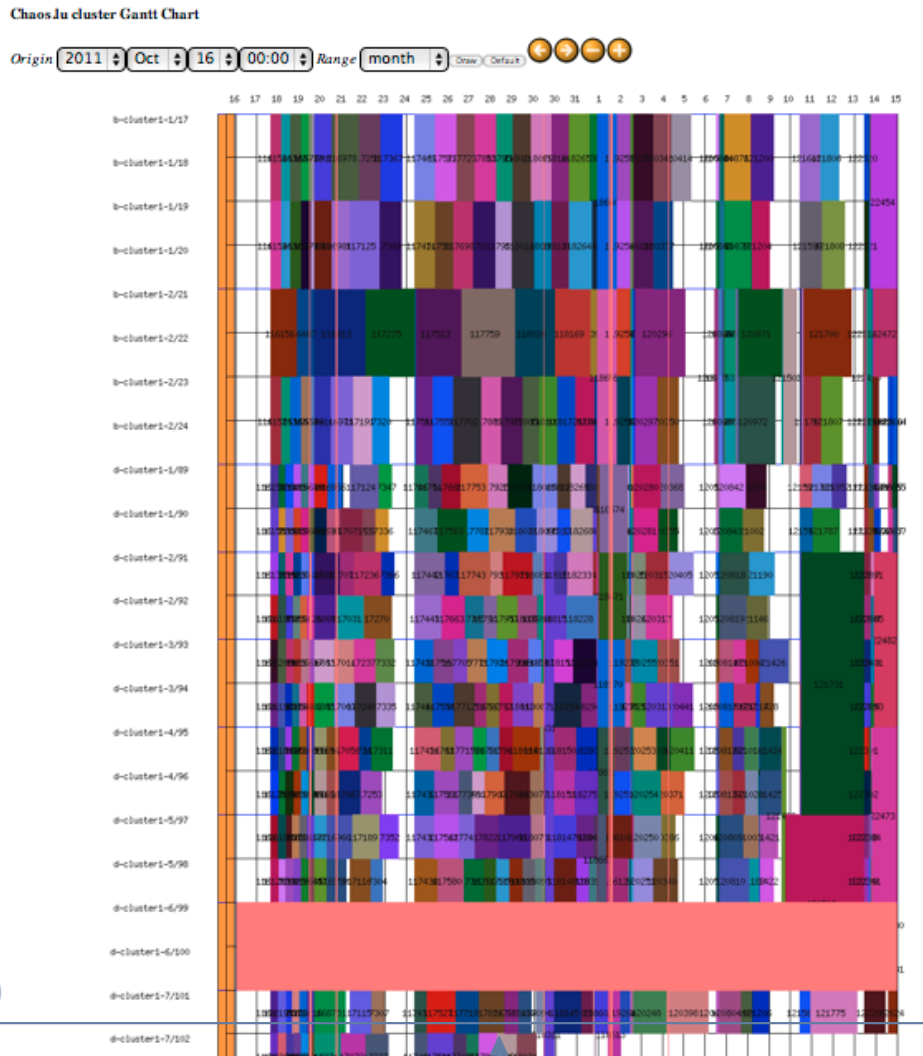
http://hpc.uni.lu/{chaos,gaia,granduc}/monika



Platform Monitoring

Drawgantt

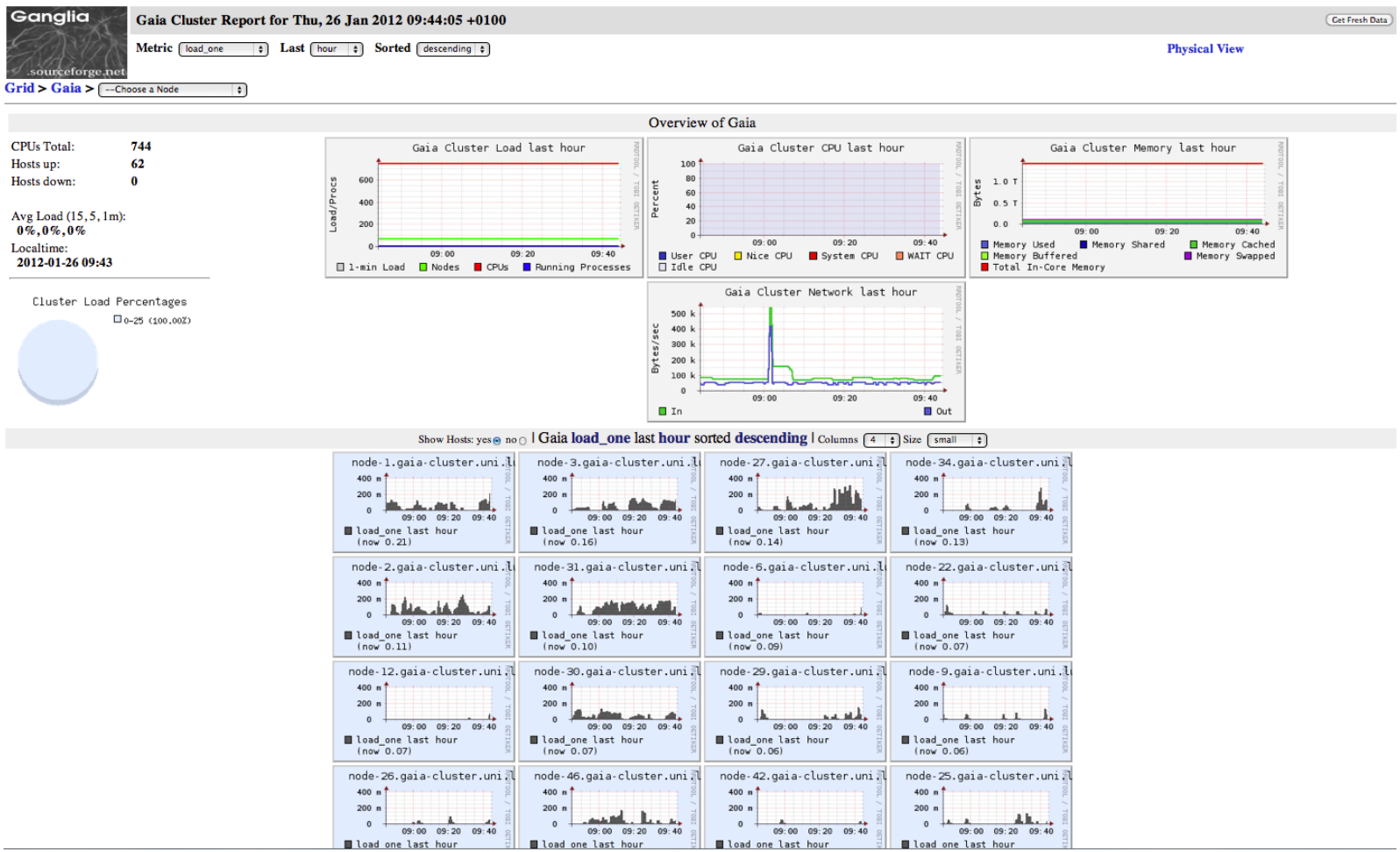
http://hpc.uni.lu/{chaos,gaia,granduc}/drawgantt



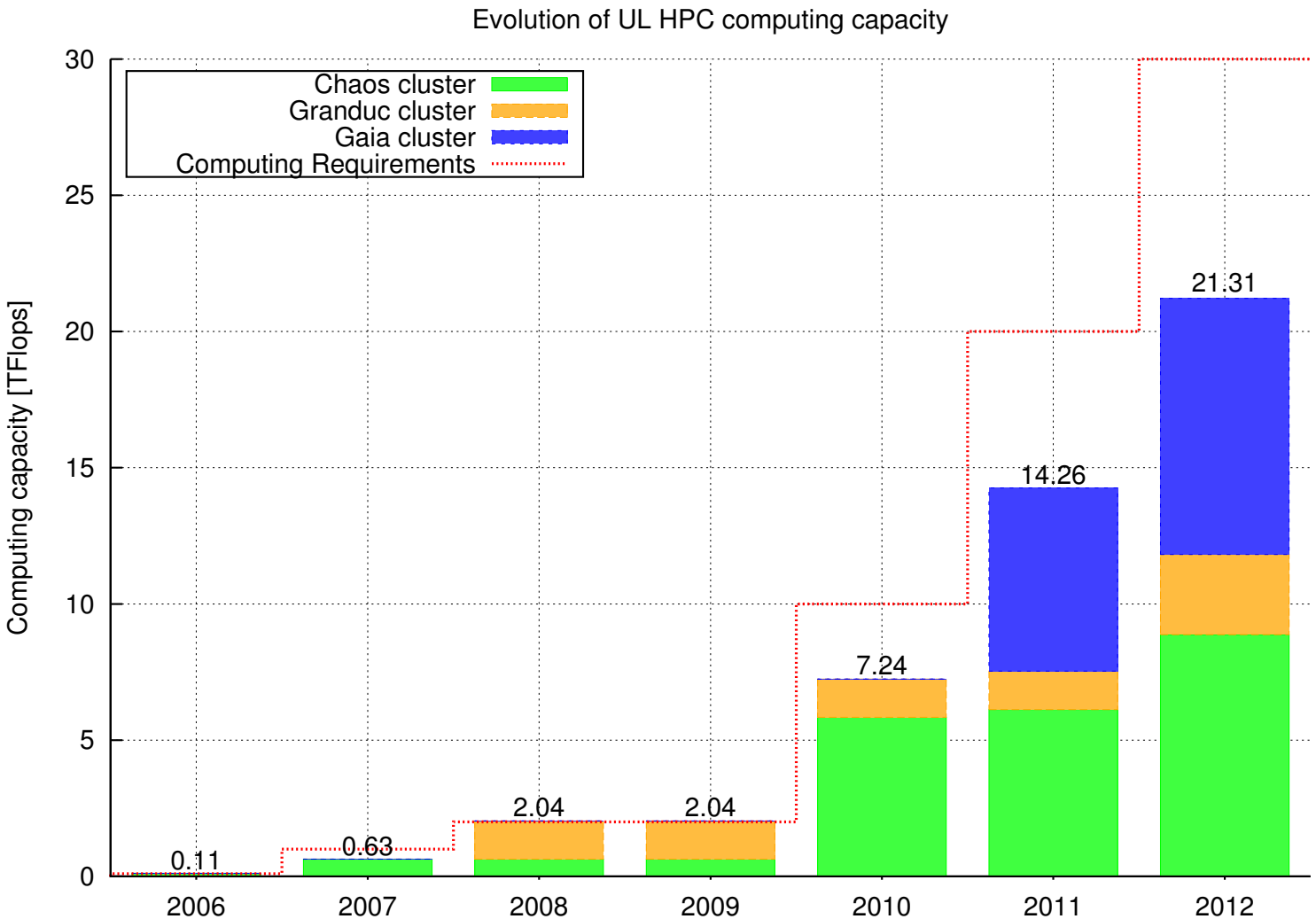
Platform Monitoring

Ganglia

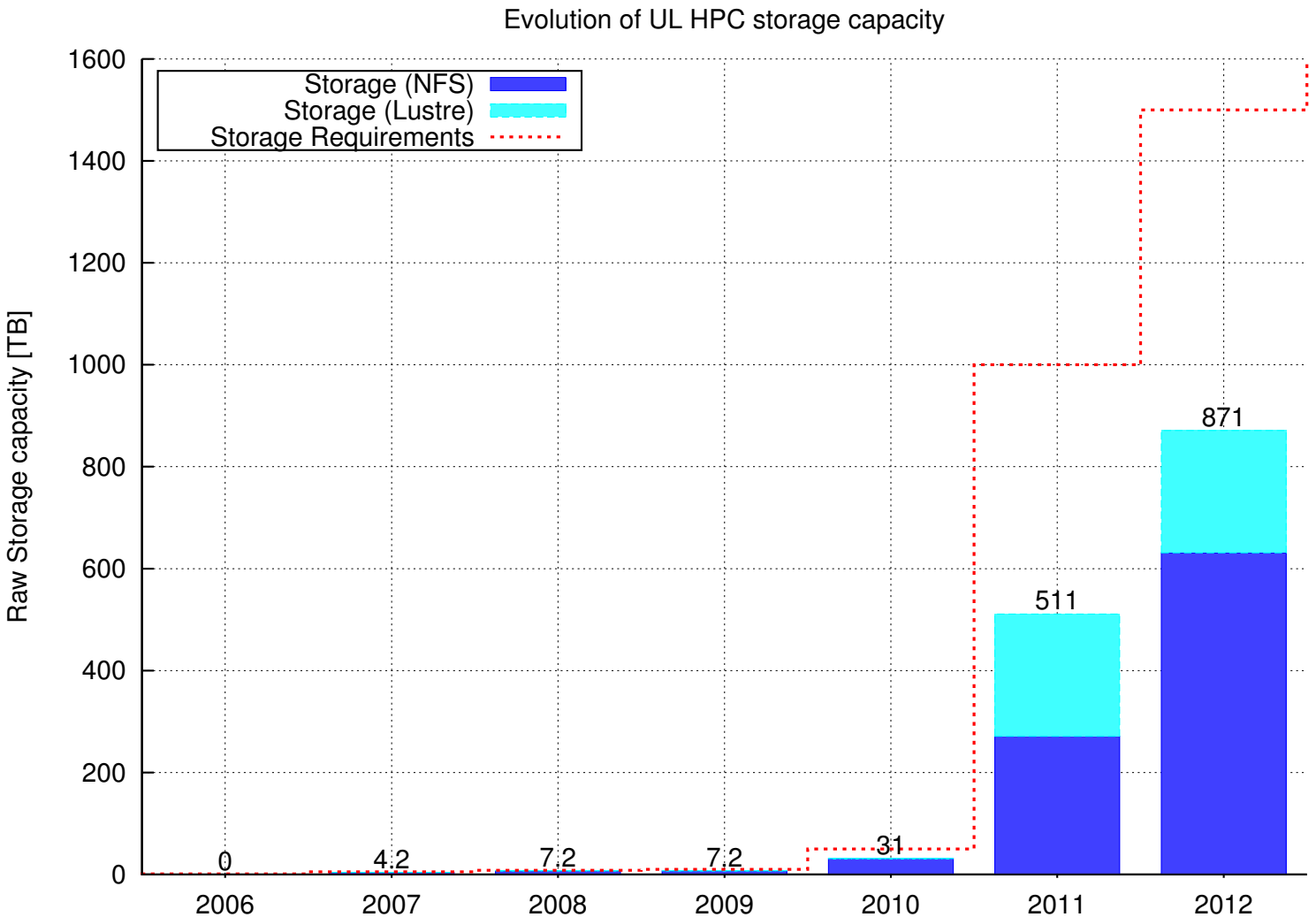
http://hpc.uni.lu/{chaos,gaia,granduc}/ganglia



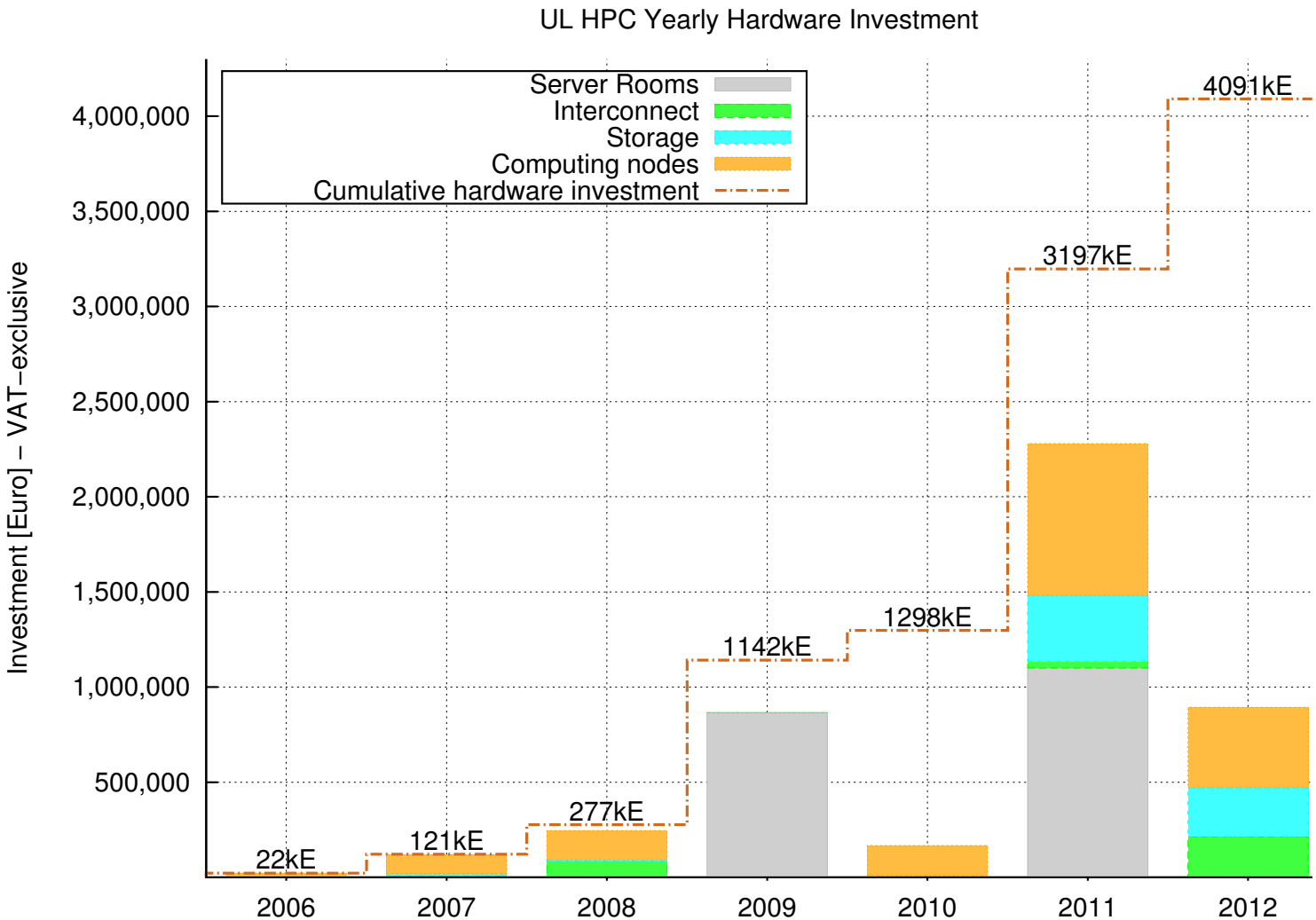
Chronological Statistics



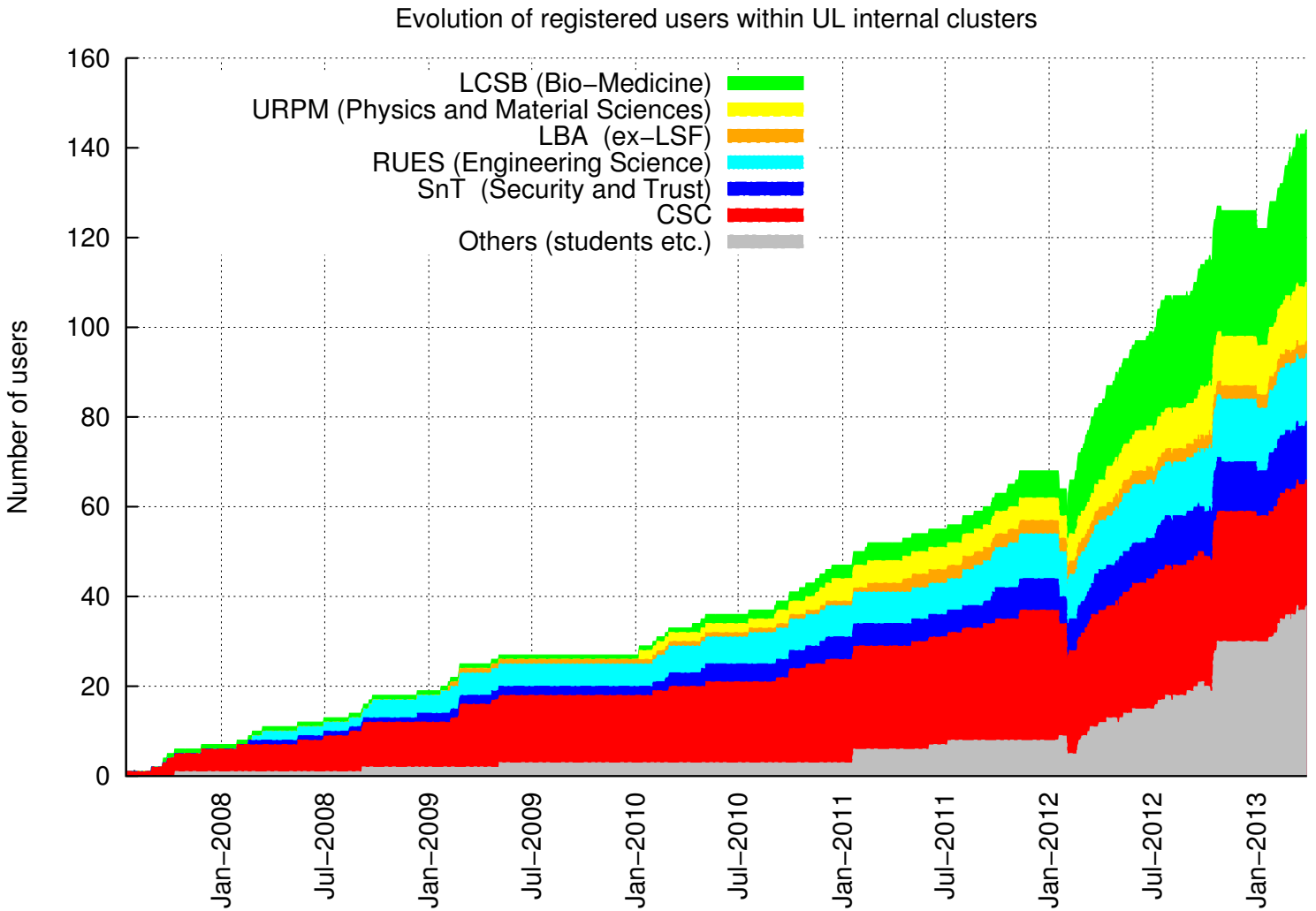
Chronological Statistics



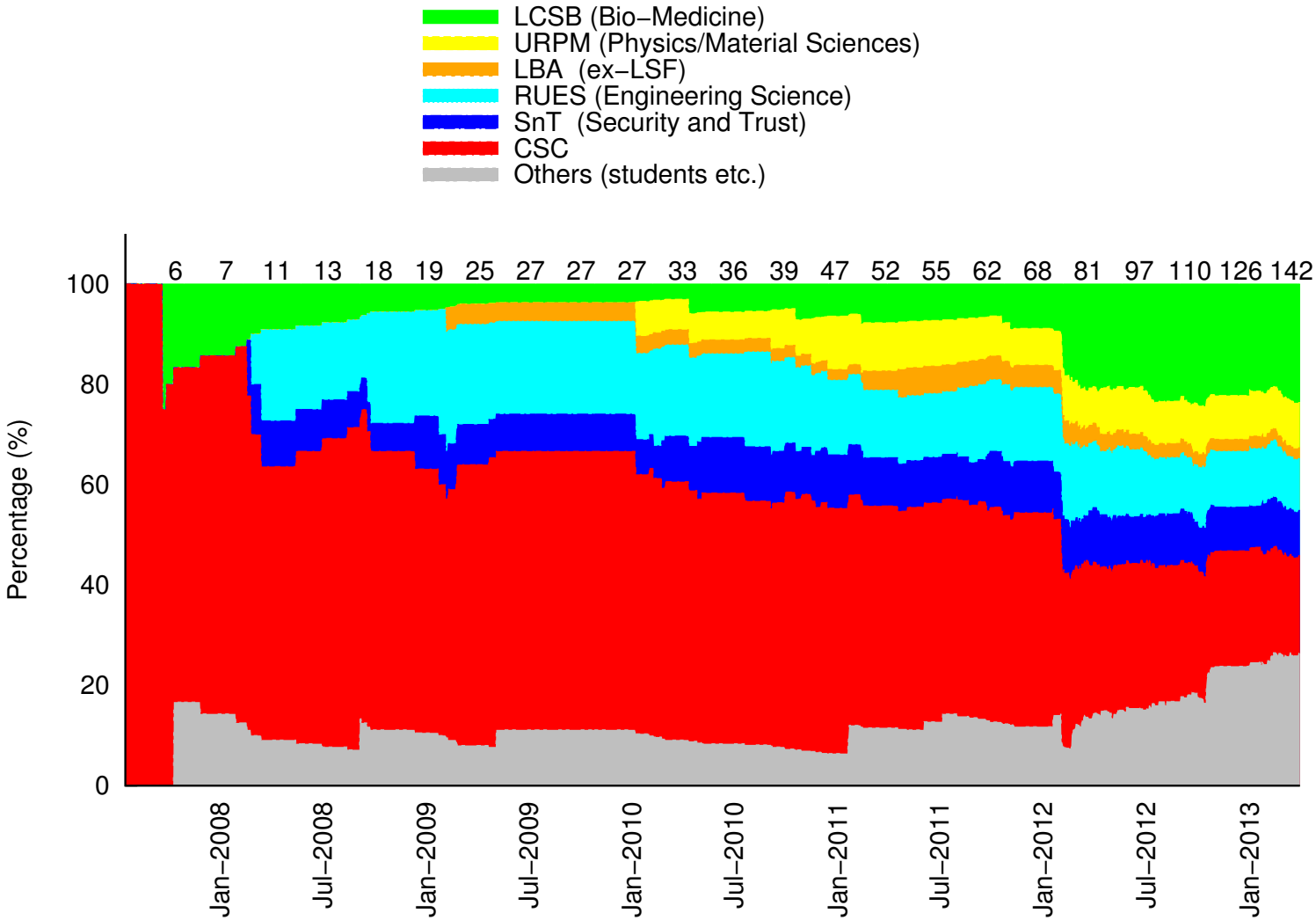
Chronological Statistics



142 Registered Users (chaos+gaia only)



142 Registered Users (chaos+gaia only)



What's new since last year?

Current capacity (2013)

291 nodes, 2944 cores, **27.363 TFlops** **1042 TB** (incl. backup)

- **New Computing nodes for `chaos` and `gaia`**
  - ↪ +10 GPU nodes `gaia-{63-72}`
  - ↪ +1 SMP node (16 procs/160 cores/1TB RAM) `gaia-73`
  - ↪ +1 big RAM node (4 procs/32 cores/1TB RAM) `gaia-74`
  - ↪ +16 HP SL nodes `chaos: s-cluster1-{1-16}`
  - ↪ +16 Dell M620 `chaos: e-cluster1-{1-16}`
- **Other computing nodes**
  - ↪ + 96 Viridis ARM nodes `viridis-{1-48}, viridis-{101-148}`
  - ↪ +16 Dell M620 nodes Grid5000 `petitprince-{1-16}`
- **Interconnect consolidation**: 10 GbE switches + IB QDR (`chaos`)

What's new since last year?

Current capacity (2013)

291 nodes, 2944 cores, **27.363 TFlops** **1042 TB** (incl. backup)

- **Storage**: 3 encl. feat. 60 disks (3TB), 6 x RAID 6 (8+2), `xfs`+LVM
  - ↪ NFS `chaos` + ~~cross~~-backup `cartman` (Kirchberg) / `stan` (Belval)
- **Commercial software** Intel Studio, // debuggers, **Mathlab**...
- **New OAR policy** for a more efficient usage of the platform
  - ↪ restrict **default** jobs to promote container approach GitHUB
    [before] 10 jobs of 1 core – [now] 1 job of 10 cores + GNU parallel
  - ↪ better incentives to `best-effort` jobs for more resilient workflow
  - ↪ project/long run management, `big{mem,smp}`
- **Directory structure** (`$HOME`,`$WORK`,`$SCRATCH`), **Modules**

2013: Incoming Milestones

- Full website reformatting with **improved doc/tutorials**
- Training/Advert: **UL HPC school** (may-june 2013)
- **OAR RESTful API**
  - ↪ cluster actions by standard HTTP operations (POST, GET, PUT, DELETE)
  - ↪ better job monitoring (cost, power consumption etc.)
- Scalable primary **backup** ($>$ 1 PB) solution
- Complement [on demand] cluster capacity
  - ↪ investigate virtualization (**Cloud / [K]VMs on the nodes**)
  - ↪ desktop grid on university TP rooms
- **Job submission web-portal** (Extreme Factory) ?

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 Charter

- Everyone must read and accept the user charter!

`https://hpc.uni.lu/documentation/user_charter`

User Account

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

With your account, you'll get:

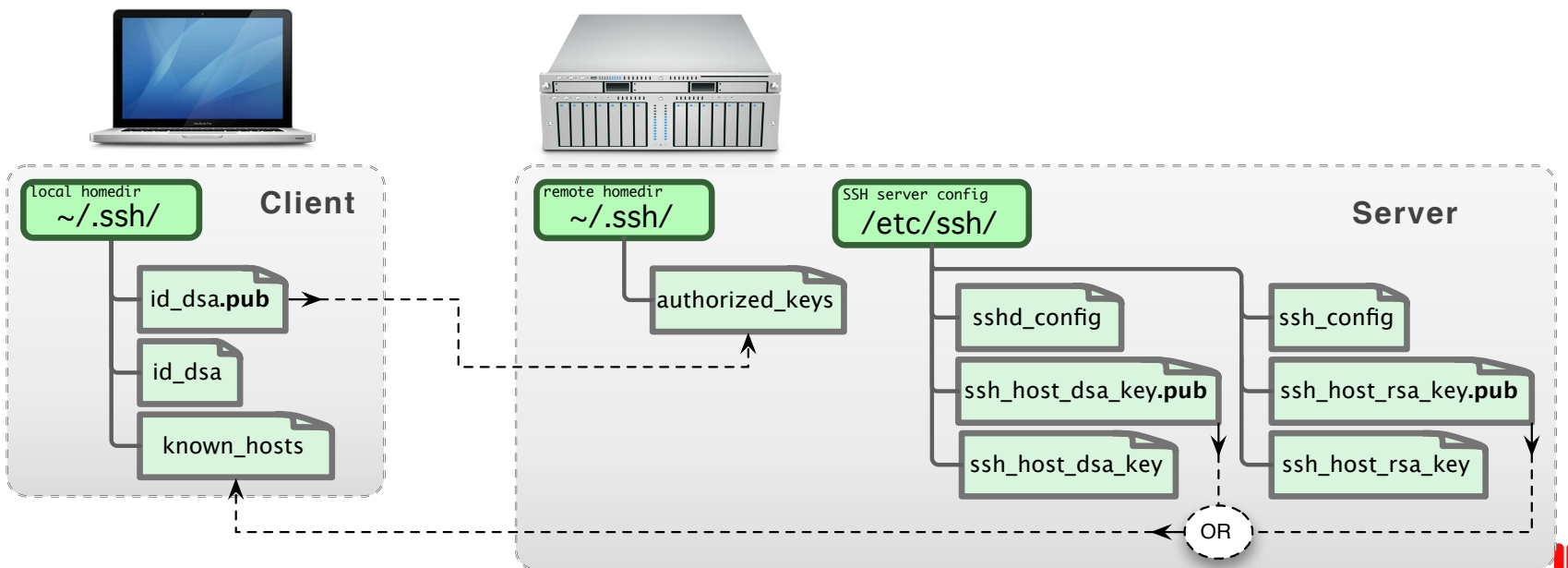
- Access to the UL HPC wiki `http://hpc.uni.lu/`
- Access to the UL HPC bug tracker `http://hpc-tracker.uni.lu/`
- Subscribed to the mailing lists `hpc-{users,platform}@uni.lu`
  - ↪ raise questions and concerns. Help us to make it a community!
  - ↪ notification of platform maintenance on `hpc-platform@uni.lu`
- 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
```

- **Transferring data...**

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

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

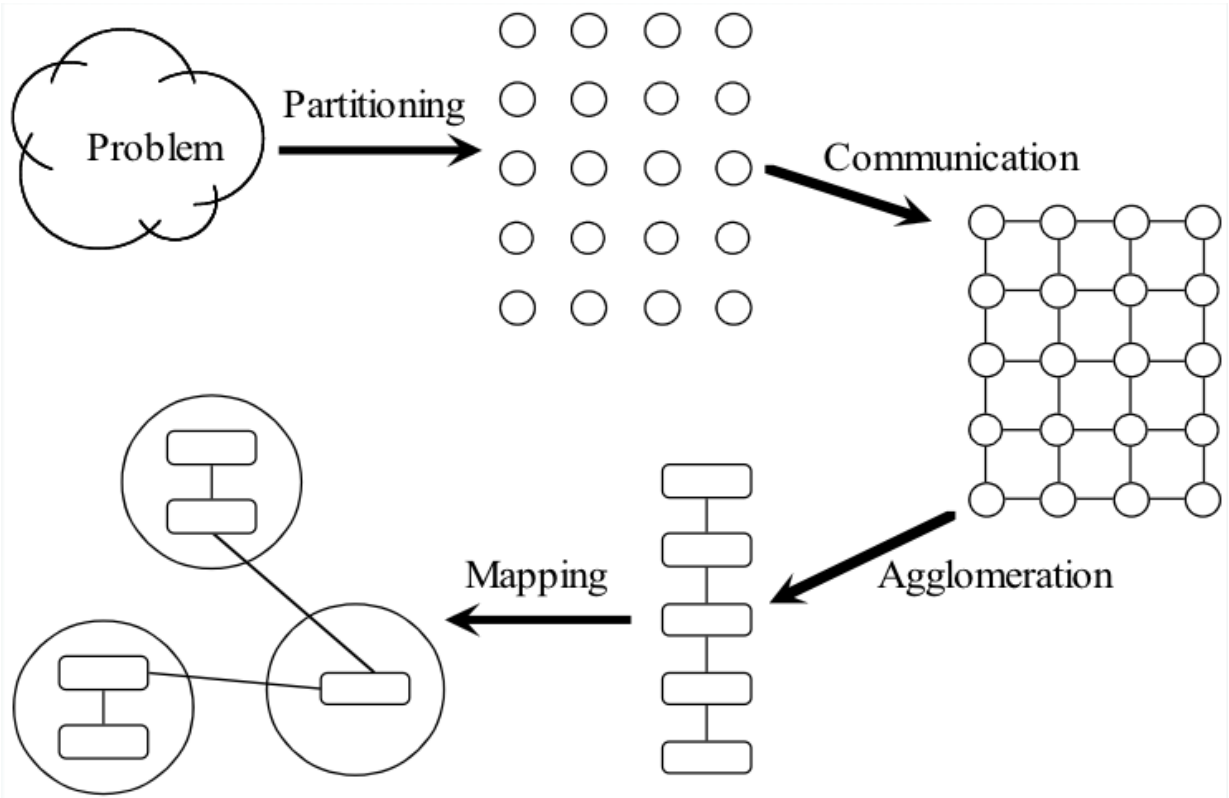
Resources/Example

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

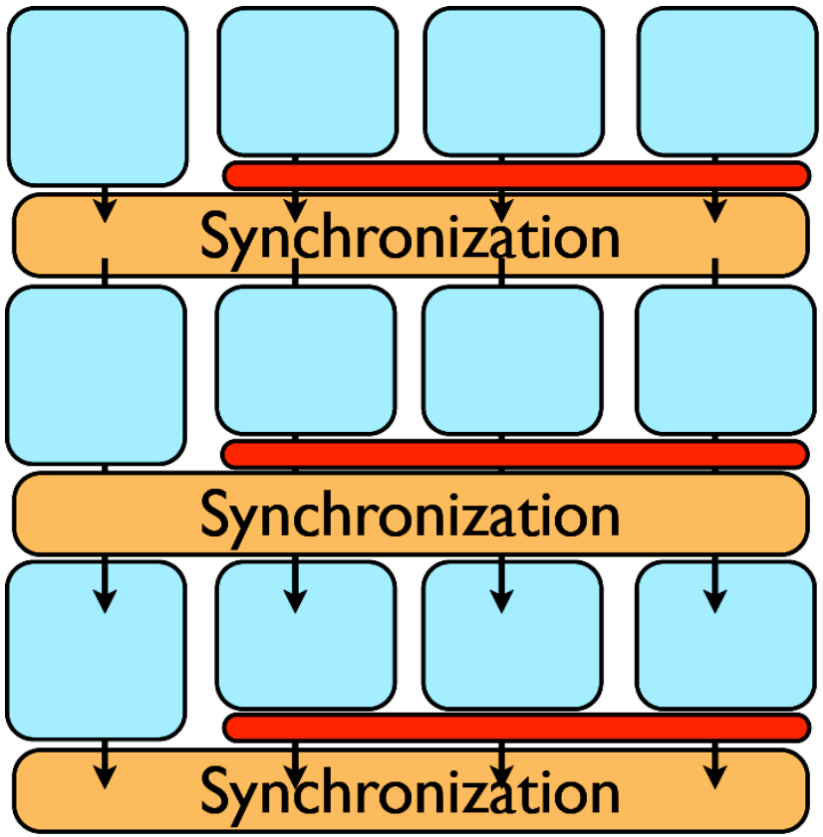
```
#OAR -l nodes=1
#OAR -n BADSerial
#OAR -O BADSerial-%jobid%.log
#OAR -E BADSerial-%jobid%.log
if [ -f  /etc/profile ]; then
    .  /etc/profile
fi
# Now you can use: 'module load toto' or 'cd $WORK'
[...]

# 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)

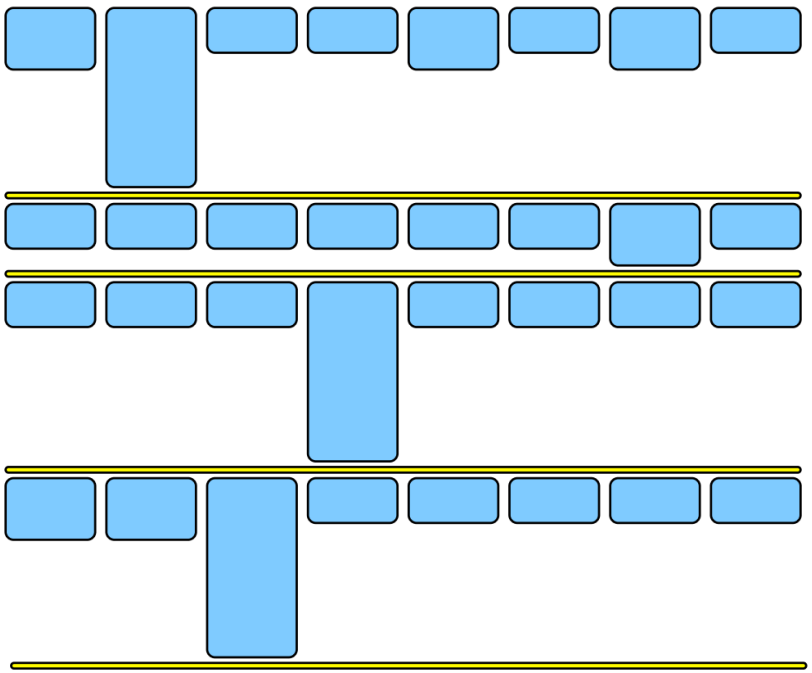
```
# Example 1: run in sequence $TASK 1...$TASK $NB_TASKS
for i in `seq 1 $NB_TASKS`; do
    $TASK $i &
done
wait

# 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)

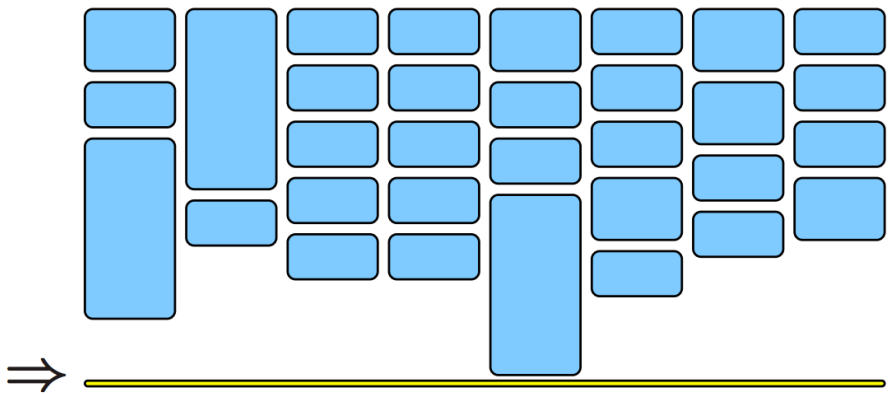
Different runs may not take the same time: **load imbalance**.



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 {}
```

MPI tasks: 3 Suites via `module`

1. OpenMPI — http://www.open-mpi.org/

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

Last Challenges for a better efficiency

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)

    ```
    $> oarsub -t bigmem ...
    ```
  - ↪ **Or** explore parallization (MPI, OpenMP, pthreads)
- Use `$SCRATCH` directory whenever you can
  - ↪ `gaia`: **shared** between nodes (Lustre FS)
  - ↪ `chaos`: **NOT shared** (`/tmp`) and clean at job end.

Last Challenges for a better efficiency

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
```

Last Challenges for a better efficiency

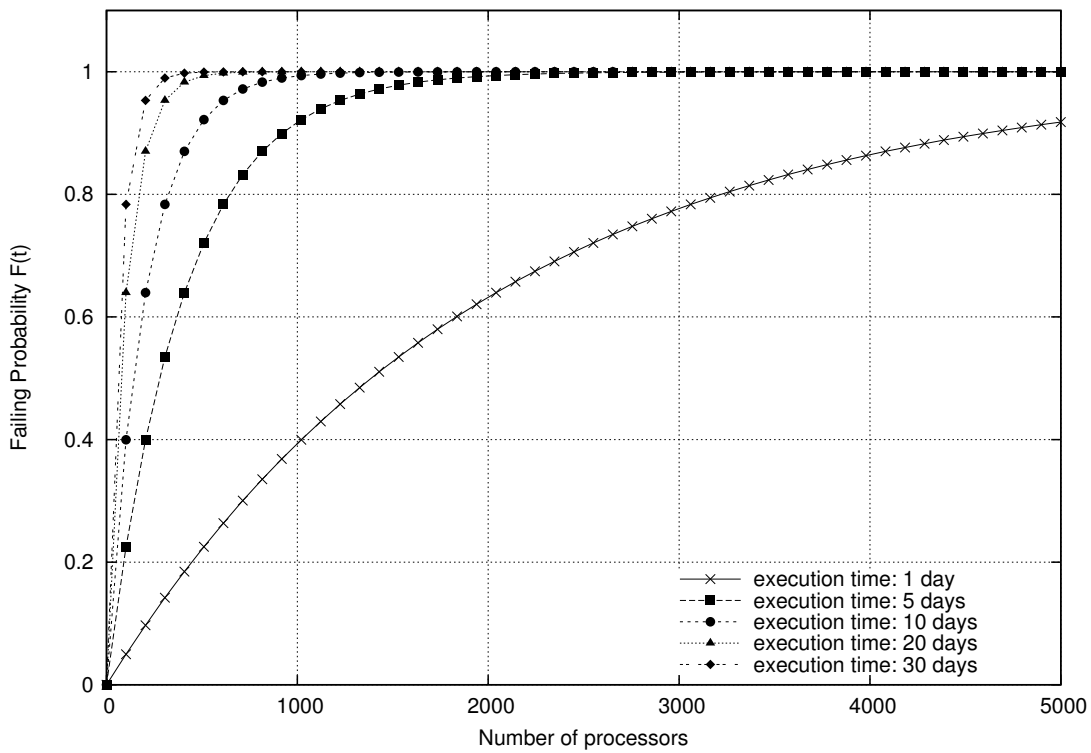
Fault Tolerance

- Cluster maintenance from time to time

Last Challenges for a better efficiency

Fault Tolerance

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



Last Challenges for a better efficiency

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

Thank you for your attention....

http://hpc.uni.lu

1. Introduction
2. Overview of the Main HPC Components
3. HPC and Cloud Computing (CC)
4. The UL HPC platform
5. UL HPC in Practice: Toward an [Efficient] Win-Win Usage

Contacts: hpc-sysadmins@uni.lu

UNIVERSITÉ DU LUXEMBOURG