High Performance Computing (HPC) at UL
Overview and Usage

UL HPC Management Team

University of Luxembourg, Luxembourg
Preliminaries

Overview of the Main HPC Components

Interlude

The UL HPC platform
  Overview
  Platform Management Tools
  Monitoring
  Statistics & Milestones

UL HPC in Practice: Toward an [Efficient] Win-Win Usage
  General considerations
  The OAR Batch Scheduler
  Reporting problems
Main HPC Performance Metrics

- **Computing Capacity/speed**: often measured in **flops** (or **flop/s**)
  - Floating point operations per seconds (often in DP)
  - $\text{GFlops} = 10^9$ Flops  $\text{TFlops} = 10^{12}$ Flops  $\text{PFlops} = 10^{15}$ Flops

- **Storage Capacity** measured in multiples of **bytes** = 8 **bits**
  - $\text{GB} = 10^9$ bytes  $\text{TB} = 10^{12}$ bytes  $\text{PB} = 10^{15}$ bytes
  - $\text{GiB} = 1024^3$ bytes  $\text{TiB} = 1024^4$ bytes  $\text{PiB} = 1024^5$ bytes

- **Transfer rate** on a medium measured in **Mb/s** or **MB/s**

- **Other metrics**: Sequential vs Random **R/W speed**, **IOPS**
## Preliminaries

### Evolution of Computing Systems

<table>
<thead>
<tr>
<th>1st Generation</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ENIAC</strong></td>
<td><strong>Transistors</strong></td>
<td><strong>Integrated Circuit</strong></td>
<td><strong>Micro-Processor</strong></td>
<td><strong>Beowulf Cluster</strong></td>
</tr>
<tr>
<td>180,000 tubes</td>
<td>Replace tubes</td>
<td>Thousands of transistors in one circuit</td>
<td>Millions of transistors in one circuit</td>
<td>Multi-core processor</td>
</tr>
<tr>
<td>150 Flops</td>
<td>33 KFlops</td>
<td>0.06 Mips</td>
<td>1 MFlops</td>
<td>74 MFlops</td>
</tr>
</tbody>
</table>

- **1946**: ENIAC
- **1956**: Transistors
- **1963**: Integrated Circuit
- **1974**: Micro-Processor
- **1980**: Beowulf Cluster
- **1994**: Multi-Core Processor
- **2005**: Pentium D
- **2010**: Cloud

**arpanet → internet**
Preliminaries

Why High Performance Computing?

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

- Accelerate research by accelerating **computations**
  
  - 14.4 GFlops (Dual-core i7 1.8GHz)
  - 49.872 TFlops (400 computing nodes, 4280 cores)

- Increase **storage** capacity
  
  - 2TB (1 disk)
  - 3364.4TB raw (642 disks)

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

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 Medicine Biology

Materials Spintronic Nano-Sciences
HPC in Worldwide Strategies

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

Segments System Share:
- Industry: 51%
- Research: 23.6%
- Academic: 17.6%
- Government
- Vendor
- Others
Preliminaries

Computing for Researchers

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

Diagram:

<table>
<thead>
<tr>
<th>IaaS</th>
<th>PaaS</th>
<th>SaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest OS (VM)</td>
<td>User's application</td>
<td>User's data</td>
</tr>
<tr>
<td></td>
<td>Libraries &amp; Softwares</td>
<td>Applications</td>
</tr>
<tr>
<td></td>
<td>Operating System</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware &amp; Network</td>
<td></td>
</tr>
</tbody>
</table>

User Control

Provider Control
Cloud Computing Platform

Platform as a Service (PaaS)
Preliminaries

Computing for Researchers

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

Diagram:

- IaaS
  - Guest OS (VM)
  - Operating System
  - Hardware & Network
  - User Control
  - Provider Control

EC2
Computing for Researchers

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

Yet...

HPC ≠ Formula 1

HPC can end badly, even after minor errors
Preliminaries

Computing for Researchers

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

**YET...**

PC $\neq$ HPC
Preliminaries

Computing for Researchers

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

**YET...**

\[ \text{PC} \neq \text{HPC} \]

- HPC \( \simeq \) Formula 1
  - can end badly, even after minor errors
Preliminaries

Jobs, Tasks & Local Execution

$> ./myprog

CPU 1
Core 2
Core 1

Time
Preliminaries

Jobs, Tasks & Local Execution

```
$> ./myprog
CPU 1
Core 2
Core 1
Time
```

UL HPC Management Team (UL)
Preliminaries

Jobs, Tasks & Local Execution

$> ./myprog
$> ./myprog
$> ./myprog -n 10
Jobs, Tasks & Local Execution

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

Jobs, Tasks & Local Execution

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

CPU 1
Core 2
Core 1

Time

UL HPC Management Team (UL)

High Performance Computing (HPC) at UL
Jobs, Tasks & Local Execution

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

![Diagram showing CPU and Core execution]

- Core 1
- Core 2

Time

CPU 1
Preliminaries

Jobs, Tasks & Local Execution

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

T₁(local) = 100s

CPU 1

Core 1

Core 2
Preliminaries

Jobs, Tasks & Local Execution

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

CPU 1
Core 1
Core 2
Time

Job(s)
3

Task(s)
3

T₁(local) = 100s

./myprog | ./myprog -n 10 | ./myprog -n 100
Preliminaries

Jobs, Tasks & Local Execution

# launcher
./myprog
./myprog -n 10
./myprog -n 100
Preliminaries

Jobs, Tasks & Local Execution

# launcher

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

CPU 1
Core 2

Time
Preliminaries

Jobs, Tasks & Local Execution

# launcher
./myprog
./myprog -n 10
./myprog -n 100

CPU 1
Core 2
Core 1

Time

./myprog ./myprog -n 10
# Preliminaries

Jobs, Tasks & Local Execution

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

```
CPU 1
Core 2
Core 1
Time
```

```
CPU 1
./myprog  ./myprog -n 10  ./myprog -n 100
```

---

UL HPC Management Team (UL)  High Performance Computing (HPC) at UL
# Jobs, Tasks & Local Execution

**Preliminaries**

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

```
CPU 1
Core 2
Core 1
```

```
T_1 (local) = 100s
```

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

```
Time
```

```
UL HPC Management Team  (UL)
High Performance Computing (HPC) at UL
```
# Preliminaries

## Jobs, Tasks & Local Execution

- **Jobs**
  - Job(s)
    - # launcher
    - `./myprog`
    - `./myprog -n 10`
    - `./myprog -n 100`

- **Task(s)**
  - Task(s)
  - `T_1 (local) = 100s`
Preliminaries

Jobs, Tasks & Local Execution

# launcher
./myprog
./myprog -n 10
./myprog -n 100

CPU 1
Core 1
Core 2

Time

11 / 91
Jobs, Tasks & Local Execution

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

CPU 1
Core 2
Core 1
Time
Jobs, Tasks & Local Execution

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

```
CPU 1
Core 2
Core 1
```

- T_2(local) = 70s
- ./myprog
- ./myprog -n 10
- ./myprog -n 100
Preliminaries

Jobs, Tasks & Local Execution

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

CPU 1
Core 1
Core 2

T_2 (local) = 70s
./myprog
./myprog -n 10
./myprog -n 100
```

1 Job(s)
3 Task(s)

Time
Preliminaries

Jobs, Tasks & HPC Execution

# launcher

./myprog

./myprog -n 10

./myprog -n 100

Node 1
CPU 1
Core 2
Core 1
CPU 2
Core 4
Core 3

Node 2
CPU 1
Core 2
CPU 2
Core 1
Core 2
Core 3
Core 4
**Preliminaries**

Jobs, Tasks & HPC Execution

---

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

---

```
T_1(hpc) = T_8(hpc) = 120s
```

---

Node 1
CPU 1
Core 1
Core 2
CPU 2
Core 3
Core 4
Node 2
CPU 1
Core 1
Core 2
CPU 2
Core 3
Core 4

---

Time
Preliminaries

Jobs, Tasks & HPC Execution

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

T_1(hpc) = T_8(hpc) = 120s
```

Node 1
- CPU 1
  - Core 1
  - Core 2

Node 2
- CPU 1
  - Core 1
  - Core 2

Node 2
- CPU 2
  - Core 3
  - Core 4

Node 2
- CPU 2
  - Core 3
  - Core 4

Time

High Performance Computing (HPC) at UL
Preliminaries

Jobs, Tasks & HPC Execution

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

Node 1
CPU 1
Core 2
Core 1
CPU 2
Core 4
Core 3

Node 2
CPU 1
Core 2
Core 1
CPU 2
Core 4
Core 3

Time
Preliminaries

Jobs, Tasks & HPC Execution

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

```
Node 1
CPU 1
Core 2
Core 1
CPU 2
Core 4
Core 3

Node 2
CPU 1
Core 2
Core 1
CPU 2
Core 4
Core 3
```

\[ T_2(\text{hpc}) = 80 \text{s} \]
Jobs, Tasks & HPC Execution

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

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
</tr>
</tbody>
</table>

Node 1
- CPU 1
  - Core 1
  - Core 2
- CPU 2
  - Core 3
  - Core 4

Node 2
- CPU 1
  - Core 1
  - Core 2
- CPU 2
  - Core 3
  - Core 4

CPU 1
- Core 1
- Core 2
- Core 3
- Core 4

HPC (hpc) = 80s

Task(s) 3

UL HPC Management Team (UL)
Jobs, Tasks & HPC Execution

T(hpc) = 60s

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

Task(s) 3

CPU 1
Core 2
Core 1
CPU 2
Core 3
Core 4

Node 1
CPU 1
Core 1
Core 2
CPU 2
Core 3
Core 4

Node 2
CPU 1
Core 1
Core 2
CPU 2
Core 3
Core 4
### Preliminaries

#### Local vs. HPC Executions

<table>
<thead>
<tr>
<th>Context</th>
<th>Local PC</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>$T_1(\text{local}) = 100s$</td>
<td>$T_1(\text{hpc}) = 120s$</td>
</tr>
<tr>
<td>Parallel/Distributed</td>
<td>$T_2(\text{local}) = 70s$</td>
<td>$T_2(\text{hpc}) = 80s$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_8(\text{hpc}) = 120s$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_8(\text{hpc}) = 60s$</td>
</tr>
</tbody>
</table>

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 $\geq 2$ cores/nodes

\[ \text{You have to explicitly adapt your jobs to benefit from the multi-cores/nodes} \]
Local vs. HPC Executions

<table>
<thead>
<tr>
<th>Context</th>
<th>Local PC</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>$T_1$ (local) = 100s</td>
<td>$T_1$ (hpc) = 120s</td>
</tr>
<tr>
<td>Parallel/Distributed</td>
<td>$T_2$ (local) = 70s</td>
<td>$T_2$ (hpc) = 80s, $T_8$ (hpc) = 120s, $T_8$ (hpc) = 60s</td>
</tr>
</tbody>
</table>

- **Sequential** runs **WON’T BE FASTER** on HPC
  - Reason: Processor Frequency (typically 3GHz vs 2.26GHz)
### Local vs. HPC Executions

<table>
<thead>
<tr>
<th>Context</th>
<th>Local PC</th>
<th>HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>$T_1$(local) = 100s</td>
<td>$T_1$(hpc) = 120s</td>
</tr>
<tr>
<td>Parallel/Distributed</td>
<td>$T_2$(local) = 70s</td>
<td>$T_2$(hpc) = 80s, $T_8$(hpc) = 120s, $T_8$(hpc) = 60s</td>
</tr>
</tbody>
</table>

- **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 $\geq$ 2 cores/nodes
  - You have to explicitly adapt your jobs to benefit from the multi-cores/nodes
Preliminaries

Identifying Potential Parallelism

**In your workflow**

```bash
>$> ./my_sequential_prog -n 1
>$> ./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

```c
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

\[ x = \text{initX}(A, B); \]
\[ y = \text{initY}(A, B); \]
\[ z = \text{initZ}(A, B); \]

\[
\text{for}(i = 0; i < \text{N_ENTRIES}; i++) \\
\quad x[i] = \text{compX}(y[i], z[i]);
\]

\[
\text{for}(i = 1; i < \text{N_ENTRIES}; i++) \\
\quad x[i] = \text{solveX}(x[i-1]);
\]

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

Functional 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

\begin{align*}
x &= \text{initX}(A, B); \\
y &= \text{initY}(A, B); \\
z &= \text{initZ}(A, B); \\
\text{Functional Parallelism} \\
\text{for} \ (i = 0; i < \text{N_ENTRIES}; i++) \\
& \quad x[i] = \text{compX}(y[i], z[i]); \\
\text{Data Parallelism} \\
\text{for} \ (i = 1; i < \text{N_ENTRIES}; i++) \\
& \quad x[i] = \text{solveX}(x[i-1]); \\
\text{Pipelining} \\
\text{finalize1} (&x, &y, &z); \\
\text{finalize2} (&x, &y, &z); \\
\text{finalize3} (&x, &y, &z); \\
\end{align*}
Preliminaries

Identifying Potential Parallelism

In your program

\[
x = \text{initX}(A, B);
y = \text{initY}(A, B);
z = \text{initZ}(A, B);
\]

Functional Parallelism

\[
\text{for } (i = 0; i < N\_ENTRIES; i++)
\quad x[i] = \text{compX}(y[i], z[i]);
\]

Data Parallelism

\[
\text{for } (i = 1; i < N\_ENTRIES; i++)
\quad x[i] = \text{solveX}(x[i-1]);
\]

Pipelining

\[
\text{finalize1 } (&x, &y, &z);
\text{finalize2 } (&x, &y, &z);
\text{finalize3 } (&x, &y, &z);
\]

No good?
Overview of the Main HPC Components

Summary

1 Preliminaries

2 Overview of the Main HPC Components

3 Interlude

4 The UL HPC platform
   Overview
   Platform Management Tools
   Monitoring
   Statistics & Milestones

5 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
   General considerations
   The OAR Batch Scheduler
   Reporting problems
Overview of the Main HPC Components

HPC Components: [GP]CPU

**CPU**

- Always multi-core
- **Ex:** Intel Core i7-970 (July 2010) \( R_{\text{peak}} \approx 100 \text{ GFlops (DP)} \)
  \[ \Rightarrow 6 \text{ cores } @ 3.2\text{GHz} \ (32\text{nm}, 130\text{W}, 1170 \text{ millions transistors}) \]

**GPU / GPGPU**

- Always multi-core, optimized for vector processing
- **Ex:** Nvidia Tesla C2050 (July 2010) \( R_{\text{peak}} \approx 515 \text{ GFlops (DP)} \)
  \[ \Rightarrow 448 \text{ cores } @ 1.15\text{GHz} \]

\[ \approx 10 \text{ Gflops for 50 } \€ \]
Overview of the Main HPC Components

HPC Components: Local Memory

- CPU
  - Registers
  - L1-cache (SRAM) reference
  - L2-cache (SRAM) reference
  - L3-cache (DRAM) reference

Memory

- Memory (DRAM) reference
- Disk memory reference

Larger, slower and cheaper

- Level: 1
  - Size: 500 bytes
  - Speed: sub ns
  - 1-2 cycles

- Level: 2
  - Size: 64 KB to 8 MB
  - Speed: 10 cycles

- Level: 3
  - Size: 1 GB
  - Speed: hundreds cycles

- Level: 4
  - Size: 1 TB
  - Speed: ten of thousands cycles

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

1500 €/TB
150 €/TB
Overview of the Main HPC Components

**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

<table>
<thead>
<tr>
<th>Technology</th>
<th>Effective Bandwidth</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gigabit Ethernet</td>
<td>1 Gb/s</td>
<td>125 MB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>40µs to 300µs</td>
</tr>
<tr>
<td>Myrinet (Myri-10G)</td>
<td>9.6 Gb/s</td>
<td>1.2 GB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.3µs</td>
</tr>
<tr>
<td>10 Gigabit Ethernet</td>
<td>10 Gb/s</td>
<td>1.25 GB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4µs to 5µs</td>
</tr>
<tr>
<td>Infiniband QDR</td>
<td>40 Gb/s</td>
<td>5 GB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.29µs to 2.6µs</td>
</tr>
<tr>
<td>SGI NUMAlink</td>
<td>60 Gb/s</td>
<td>7.5 GB/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1µs</td>
</tr>
</tbody>
</table>
Overview of the Main HPC Components

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

<table>
<thead>
<tr>
<th>Technology</th>
<th>Effective Bandwidth</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gigabit Ethernet</td>
<td>1 Gb/s</td>
<td>125 MB/s</td>
</tr>
<tr>
<td>Myrinet (Myri-10G)</td>
<td>9.6 Gb/s</td>
<td>1.2 GB/s</td>
</tr>
<tr>
<td>10 Gigabit Ethernet</td>
<td>10 Gb/s</td>
<td>1.25 GB/s</td>
</tr>
<tr>
<td>Infiniband QDR</td>
<td>40 Gb/s</td>
<td>5 GB/s</td>
</tr>
<tr>
<td>SGI NUMAlink</td>
<td>60 Gb/s</td>
<td>7.5 GB/s</td>
</tr>
</tbody>
</table>
Overview of the Main HPC Components

HPC Components: Operating System

- Mainly **Linux**-based OS (91.4%) (Top500, Nov 2011)
- ... or Unix based (6%)
- Reasons:
  - stability
  - prone to devals
Overview of the Main HPC Components

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.
Overview of the Main HPC Components

HPC Components: Data Management

Storage architectural classes & I/O layers

Application

[Distributed] File system

Network

NFS
CIFS
AFP...

Network

Ethernet/
Network

SAN Interface

File System

Ethernet/
Network

NAS Interface

SATA
SAS
Fiber Channel

DAS Interface

SATA
SAS
Fiber Channel

SAN

NAS

SATA
SAS
Fiber Channel

DASSATA
SAS
Fiber Channel

DAS Interface

NAS Interface
Overview of the Main HPC Components

HPC Components: Data Management

RAID standard levels

<table>
<thead>
<tr>
<th>RAID Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID 0</td>
<td>Basic (low cost): 300 MB/s; Advanced (expansive): 1,5 GB/s</td>
</tr>
<tr>
<td>RAID 1</td>
<td>Disk 0: A1, A2, A3, A4, A5, A6, A7; Disk 1: A8</td>
</tr>
<tr>
<td>RAID 5</td>
<td>Disk 0: A1, B1, C1, D1, E1; Disk 1: A2, B2, C2, D2, E2; Disk 2: A3, B3, C3, D3, E3; Disk 3: A4, B4, C4, D4, E4</td>
</tr>
<tr>
<td>RAID 6</td>
<td>Disk 0: A1, B1, C1, D1, E1; Disk 1: A2, B2, C2, D2, E2; Disk 2: A3, B3, C3, D3, E3; Disk 3: A4, B4, C4, D4, E4; Disk 4: A5, B5, C5, D5, E5</td>
</tr>
</tbody>
</table>

Software vs. Hardware

RAID management

Controller card performances differ!
Overview of the Main HPC Components

HPC Components: Data Management

RAID combined levels

RAID 10
RAID 0

RAID 1

RAID 1

A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8
Overview of the Main HPC Components

HPC Components: Data Management

RAID combined levels

- RAID 0
  - RAID 5
    - A1
    - B1
    - C1
    - D1
  - Disk 0
    - 120GB
- RAID 5
  - RAID 5
    - A2
    - B2
    - C2
    - D2
  - Disk 1
    - 120GB
- RAID 5
  - RAID 5
    - Ap
    - Bp
    - Cp
    - Dp
  - Disk 2
    - 120GB
- RAID 5
  - RAID 5
    - A3
    - B3
    - C3
    - D3
  - Disk 3
    - 120GB
- RAID 5
  - RAID 5
    - A4
    - B4
    - C4
    - D4
  - Disk 4
    - 120GB
- RAID 5
  - RAID 5
    - A5
    - B5
    - C5
    - D5
  - Disk 5
    - 120GB
- RAID 5
  - RAID 5
    - A6
    - B6
    - C6
    - D6
  - Disk 6
    - 120GB
- RAID 5
  - RAID 5
    - Ap
    - Bp
    - Cp
    - Dp
  - Disk 7
    - 120GB
- RAID 5
  - RAID 5
    - Ap
    - Bp
    - Cp
    - Dp
  - Disk 8
    - 120GB
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
Overview of the Main HPC Components

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
Overview of the Main HPC Components

HPC Components: Data Center

**Definition (Data Center)**

Facility to house computer systems and associated components

→ Basic storage component: **rack** (height: 42 RU)
Overview of the Main HPC Components

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

\[ PUE = \frac{\text{Total facility power}}{\text{IT equipment power}} \]
Overview of the Main HPC Components

HPC Components: Data Center
Overview of the Main HPC Components

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...
Overview of the Main HPC Components

Putting it all together...

- Site <sitename>
- Other Clusters network
  - Fast local interconnect (Infiniband, 10GbE)
- Local Institution Network
  - Site access server
  - Site router
  - 1 GbE
  - 10 GbE

- Site Computing Nodes
  - Cluster A
  - Cluster B
  - NFS and/or Lustre
  - Disk Enclosure
  - Site Shared Storage Area
  - Adminfront
    - OAR
    - Puppet
    - Kadeploy
    - supervision
    - etc...
  - Fast local interconnect (Infiniband, 10GbE)
Summary

1 Preliminaries

2 Overview of the Main HPC Components

3 Interlude

4 The UL HPC platform
   Overview
   Platform Management Tools
   Monitoring
   Statistics & Milestones

5 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
   General considerations
   The OAR Batch Scheduler
   Reporting problems
Remote Connection by SSH

- Establish **Secure** / Encrypted tunnel between $\geq 2$ hosts
  - 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)
  - Linux/Mac:
    $\texttt{ssh-keygen -t dsa}$
  - Windows: **PuttyGen** + Pageant
Interlude

UL HPC access

*Restricted* to SSH connection *with public key authentication*

on a non-standard port (8022) *limits kiddie script scans/dictionary’s attacks*

```
local homedir
~/.ssh/

id_dsa.pub
id_dsa
known_hosts

remote homedir
~/.ssh/

authorized_keys

SSH server config
/etc/ssh/

ssh_host_dsa_key.pub
ssh_host_rsa_key.pub

or

ssh_config
sshd_config

ssh_host_dsa_key
ssh_host_rsa_key
```
Interlude

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/
Interlude

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
Interlude

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

1 Preliminaries

2 Overview of the Main HPC Components

3 Interlude

4 The UL HPC platform
   Overview
   Platform Management Tools
   Monitoring
   Statistics & Milestones

5 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
   General considerations
   The OAR Batch Scheduler
   Reporting problems
The UL HPC platform

HPC @ UL  http://hpc.uni.lu
Meet the Team

UL HPC Newsletter

Issue 1 - March 2015

Meet the Team

Management

Pascal Bouvy is a full professor of the FSTC and the head of the IIS research unit and the DR-CSCE doctoral school. His team (POG) is composed of 25 researchers working on Parallel Computing and Optimization applied to cloud computing and HPC (scheduling, energy efficiency, security). His research interests include simulation and service optimization and biology (gene sequencing, regulatory networks, protein folding).

Sébastien Varrette, PhD, is a Research Associate in Prof. Bouvy’s team since 2007. Along with Prof. Bouvy, he defined and set up the global HPC initiative of the UL in 2007. In this context, he is managing the sysadmin team that maintains and extends 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 Cantaux 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 Grid. 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. Bouvy’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 Fillatre 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.

http://hpc.uni.lu
The UL HPC platform

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...
The UL HPC platform

HPC server rooms

- **2009** CS.43 (Kirchberg campus)  14 racks, 100 m², $\sim 800,000\text{€}$

- **2011** LCSB 6th floor (Belval)  14 racks, 112 m², $\sim 1,100,000\text{€}$
# UL HPC Computing Nodes

<table>
<thead>
<tr>
<th>Date</th>
<th>Vendor</th>
<th>Proc. Description</th>
<th>#N</th>
<th>#C</th>
<th>R&lt;sub&gt;peak&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>HP</td>
<td>Intel Xeon <a href="mailto:L5640@2.26GHz">L5640@2.26GHz</a>, 2 x 6C, 24GB</td>
<td>32</td>
<td>384</td>
<td>3.472 TFlops</td>
</tr>
<tr>
<td>2011</td>
<td>Dell</td>
<td>Intel Xeon <a href="mailto:L5640@2.26GHz">L5640@2.26GHz</a>, 2 x 6C, 24GB</td>
<td>16</td>
<td>192</td>
<td>1.736 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Dell</td>
<td>Intel Xeon <a href="mailto:X7560@2.26GHz">X7560@2.26GHz</a>, 4 x 6C, 1TB</td>
<td>1</td>
<td>32</td>
<td>0.289 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Dell</td>
<td>Intel Xeon <a href="mailto:E5-2660@2.2GHz">E5-2660@2.2GHz</a>, 2 x 8C, 32GB</td>
<td>16</td>
<td>256</td>
<td>4.506 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>HP</td>
<td>Intel Xeon <a href="mailto:E5-2660@2.2GHz">E5-2660@2.2GHz</a>, 2 x 8C, 32GB</td>
<td>16</td>
<td>256</td>
<td>4.506 TFlops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chaos TOTAL:</td>
<td>81</td>
<td>1120</td>
<td>14.49 TFlops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Granduc/petitprince TOTAL:</td>
<td>38</td>
<td>368</td>
<td>4.48 TFlops</td>
</tr>
<tr>
<td>2011</td>
<td>Bull</td>
<td>Intel Xeon <a href="mailto:L5640@2.26GHz">L5640@2.26GHz</a>, 2 x 6C, 48GB</td>
<td>72</td>
<td>864</td>
<td>7.811 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Dell</td>
<td>Intel Xeon <a href="mailto:E5-4640@2.4GHz">E5-4640@2.4GHz</a>, 4 x 8C, 1TB</td>
<td>1</td>
<td>32</td>
<td>0.307 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Bull</td>
<td>Intel Xeon <a href="mailto:E5-2660@2.2GHz">E5-2660@2.2GHz</a>, 2 x 8C, 64GB</td>
<td>5</td>
<td>80</td>
<td>1.408 TFlops</td>
</tr>
<tr>
<td>2013</td>
<td>Bull</td>
<td>Intel Xeon <a href="mailto:X5670@2.93GHz">X5670@2.93GHz</a>, 2 x 6C, 48GB</td>
<td>40</td>
<td>480</td>
<td>5.626 TFlops</td>
</tr>
<tr>
<td>2013</td>
<td>Bull</td>
<td>Intel Xeon <a href="mailto:X5675@3.07GHz">X5675@3.07GHz</a>, 2 x 6C, 48GB</td>
<td>32</td>
<td>384</td>
<td>4.746 TFlops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gaia TOTAL:</td>
<td>183</td>
<td>2400</td>
<td>30.382 TFlops</td>
</tr>
<tr>
<td>2008</td>
<td>Dell</td>
<td>Intel Xeon L5335@2GHz, 2 x 4C, 16GB</td>
<td>22</td>
<td>176</td>
<td>1.408 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Dell</td>
<td>Intel Xeon E5-2630L@2GHz, 2 x 6C, 24GB</td>
<td>16</td>
<td>192</td>
<td>1.536 TFlops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Granduc/petitprince TOTAL:</td>
<td>38</td>
<td>368</td>
<td>4.48 TFlops</td>
</tr>
<tr>
<td>2012</td>
<td>Dell</td>
<td>Intel Xeon <a href="mailto:E5-2420@1.9GHz">E5-2420@1.9GHz</a>, 1 x 6C, 32GB</td>
<td>2</td>
<td>12</td>
<td>0.091 TFlops</td>
</tr>
<tr>
<td>2013</td>
<td>Viridis</td>
<td>ARM A9 <a href="mailto:Cortex@1.1GHz">Cortex@1.1GHz</a>, 1 x 4C, 4GB</td>
<td>96</td>
<td>384</td>
<td>0.422 TFlops</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Viridis TOTAL:</td>
<td>98</td>
<td>392</td>
<td>0.52 TFlops</td>
</tr>
</tbody>
</table>

**Total:** 400 nodes, 4280 cores, 49.872 TFlops
The UL HPC platform

UL HPC: General cluster organization

Site <sitename>

Other Clusters network

Local Institution Network

Site access server

Site router

1 GbE

1 GbE

10 GbE

10 GbE

Fast local interconnect (Infiniband, 10GbE)

Site Computing Nodes

Cluster A

Cluster B

Site Shared Storage Area

Disk Enclosure

NFS and/or Lustre

Adminfront

OAR, Puppet, Kadeploy

supervision, etc...

Puppet
**The UL HPC platform**

### 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)

---

**Uni.lu** (Kirchberg)

**Chaos cluster**

**LCSB Belval**

**(gaia cluster)**

**Uni.lu**

**Network Diagram**

- **Chaos cluster**
  - **Access**:
    - Bull R423 (2U) (2x4c Intel Xeon E5530 @ 2.53GHz), RAM: 24GB
    - Dell PE R610 (2U) (2x4c Intel Xeon L5640 @ 2.26GHz), RAM: 24GB
  - **Storage**:
    - NetApp E5486 (180 TB)
    - RAID 6 LUNs (8+2 disks) = 144 TB (LVM + XFS)
  - **Computing Nodes**
    - 1x Dell M1000e enclosure (8U)
      - 16 blades SL230s Gen8 [256 cores]
      - (2x8c Intel Xeon E5-2660@2.2GHz), RAM: 32GB
    - 1x Dell M1000e enclosure (10U)
      - 16 blades Dell M620 [256 cores]
      - (2x8c Intel Xeon ES-2660@2.2GHz), RAM: 32GB
  - **IB**

---

**AS.28** (708 cores) Computing Nodes

- **AS.28** (708 cores) Computing Nodes
  - 2x HP SL6500 (8U)
    - 16 blades SL230s Gen8 [256 cores]
    - (2x8c Intel Xeon E5-2660@2.2GHz), RAM: 32GB

**CS.43** (416 cores) Computing Nodes

- 1x HP c7000 enclosure (10U)
  - 32 blades HP BL2x220c G6 [384 cores]
  - (4x6c Intel Xeon X7560@2.26GHz), RAM: 1TB

---

**NFS server**

- Dell R710 (2U) (2x4c Intel Xeon E5506 @ 2.13GHz), RAM: 24GB

---

**Adminfront**

- 10 GbE IB

---

**NetApp**

- E5486 (180 TB)

---

**Chaos cluster access**

- Cisco Nexus C5010 10GbE

---

**Infiniband**

- QDR 40 Gb/s (Min Hop)

---

**UL HPC Management Team (UL)**

---

High Performance Computing (HPC) at UL
The UL HPC platform

Ex: The gaia cluster

Gaia cluster characteristics
- Computing: 154 nodes, 2024 cores; \( R_{\text{peak}} \approx 21.7 \) TFlops
- Storage: 480 TB (NFS) + 240 TB (Lustre) + 884 TB (backup)
The UL HPC platform

Ex: Some racks of the gaia cluster
The UL HPC platform

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

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

The case of 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
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](https://hpc.uni.lu/g5k)

Grid’5000 website and documentation: [https://www.grid5000.fr](https://www.grid5000.fr)
**Node deployment by FAI**

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

![Diagram of FAI installation process](http://fai-project.org/)

### Examples of installation times

<table>
<thead>
<tr>
<th>CPU + RAM</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Athlon 800 MHz, 512MB</td>
<td>2200 MB</td>
</tr>
<tr>
<td>Pentium 4, 3 GHz, 1GB</td>
<td>1100 MB</td>
</tr>
<tr>
<td>Pentium 4, 3 GHz, 1GB</td>
<td>10 min</td>
</tr>
<tr>
<td>Core2duo, 2 GHz, 2GB</td>
<td>471 MB</td>
</tr>
<tr>
<td>Core i7, 3.2 GHz, 6GB</td>
<td>4.3 GB</td>
</tr>
<tr>
<td>Core i7, 3.2 GHz, 6GB</td>
<td>4.3 GB</td>
</tr>
</tbody>
</table>

### FAI features

- Reproducible installation
- Advanced class concept
- Useful for XEN and Vserver host installations
- Installs Debian GNU/Linux, Ubuntu, Mandriva, Suse, Solaris,...
- Remote control
- Perl
- cfengine
- Disaster recovery
- Script support for customization

### FAI installations

- ETH Zurich, systems group, 1000 hosts
- Electricité de France (EDF), 1500 hosts
- Mobile.de, 200 hosts
- Stanford University, 450 hosts
- Opera Software, 200 hosts
- Albert Einstein Institute, 1725 hosts
- StayFriends, 300+ hosts
- City of Munich, 6000+, 12.000 hosts planned
- LVM insurance, 10.000 hosts
- Anonymous, financial industry, 32.000 hosts

### FAI documentation

- Detailed documentation, mailing lists, IRC channel
- Commercial support available
- Open source under GPL license
- For more see [http://fai-project.org/reports/](http://fai-project.org/reports/)
Node deployment by FAI

- Boot via network card (PXE) → ensure a running diskless Linux OS
- Get configuration data (NFS)
Node deployment by FAI

- 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
Node deployment by FAI

- 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: \( \simeq 500 \text{s} \)
Server/Service configuration by Puppet

Configuration management made easy

Automates service deployment / sysadmin tasks

3 components:

- declarative language (manifests)
- client/server model
- Git-based workflow + Rakefile + Capistrano

Hierarchical PKI
Server/Service configuration by Puppet

- 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: ≃ 3-6 min
The UL HPC platform

Platform Management: Puppet

**TOTAL resources managed**
- 150 servers (101 VMs)
- 368 computing nodes (3880 cores)
- 1996 TB raw shared storage (NFS / Lustre)
- 2 system administrators / 2 sites

**Puppet Infrastructure**
- Managed hosts: 124 (+26 Grid'5000 Luxembourg)

**FAI Infrastructure**
- Managed computing nodes: 330 (3512 cores) (+38 (368 cores) Grid'5000 Luxembourg)

**UL HPC Platform**

**Puppet Master LCSB**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**Puppet Master CSC**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**Puppet Master gaia cluster**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**Puppet Master chaos cluster**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**LCSB site (Belval)**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**CSC site (Kirchberg)**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**gaia cluster (Belval)**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**chaos cluster (Kirchberg)**
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent
- puppet agent

**UL HPC Management Team (UL)**

High Performance Computing (HPC) at UL
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
RESIF: Revolutionary EasyBuild-based Software Installation Framework

**RESIF installation**
- `pip install resif` # Install RESIF binaries
- `resif init` # Get all eligible sources
- `resif bootstrap` # Install with default configuration

**Activate the installation**
- `source $rootinstall/LOADME-vx.y-YYYYMMDD.sh`

**RESIF software deployment workflow**
1. **Create software set**
   - Create software set
   - `$ resif build --swsets-config swset.yaml core`

2. **Build software set**
   - Ready to build

3. **Build software set**
   - Build software set

4. **Load software**
   - Load software
   - `$ module load software`

5. **Unload software**
   - Unload software
   - `$ module unload software`

**Available software usage workflow**
- Ready to use
- Ready to work
The UL HPC platform

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
The UL HPC platform

Platform Monitoring

Monika

http://hpc.uni.lu/{chaos,gaia,g5k}/monika
The UL HPC platform

Platform Monitoring

Drawgantt

http://hpc.uni.lu/{chaos,gaia,g5k}/drawgantt
The UL HPC platform

Platform Monitoring

Ganglia

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

CDash

http://cdash.uni.lu/
The UL HPC platform

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
  - **Backup**: NFS, GlusterFS, OneFS

- **6,340,316€ (Cumul. HW Investment)** since 2007

- 281 registered users
The UL HPC platform

Registered Users

Evolution of registered users within UL internal clusters

- LCSB (Bio−Medicine)
- URPM (Physics and Material Sciences)
- FDEF (Law, Economics and Finance)
- RUES (Engineering Science)
- SnT (Security and Trust)
- CSC (Computer Science and Communications)
- LSRU (Life Sciences)
- Bachelor and Master students
- Others
The UL HPC platform

CPU-year usage since 2008

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

![Platform Yearly CPU Used](image)
A Year on Gaia...
The UL HPC platform

Chronological Statistics

UL HPC Computing Capacity

- Nyx/Viridis
- G5K
- Gaia
- Chaos


Computing Capacity [TFlops]
The UL HPC platform

Chronological Statistics

UL HPC Storage Capacity (per type)

- Lustre
- GPFS
- NFS
- Tier-2 Backup

- Raw Storage Capacity [TB]

- 0, 4.2, 6.6, 6.6, 32.4, 512.4, 1052.4, 2000.4, 3368.4

UL HPC Yearly Investment, excl. Server rooms (per type)

- **Other / Support**
- **Software**
- **Interconnect**
- **Servers**
- **Storage**
- **Computing Nodes**

<table>
<thead>
<tr>
<th>Year</th>
<th>Other / Support</th>
<th>Software</th>
<th>Interconnect</th>
<th>Servers</th>
<th>Storage</th>
<th>Computing Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>21 850,00 €</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>111 168,20 €</td>
<td>74 773,78 €</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td></td>
<td>99 998,77 €</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td></td>
<td></td>
<td></td>
<td>323 478,47 €</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>926 777,35 €</td>
<td>917 647,50 €</td>
</tr>
<tr>
<td>2011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>707 338,04 €</td>
<td>865 458,42 €</td>
</tr>
</tbody>
</table>
# Research Domains and Usage

## Research Domains

<table>
<thead>
<tr>
<th>Research Areas</th>
<th>(Among the 288 registered users)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>([Ad-Hoc] Network, FT, Grid, Cloud etc.)</td>
</tr>
<tr>
<td>Mechanical Engineering</td>
<td></td>
</tr>
<tr>
<td>Physics, Geo-Physics</td>
<td></td>
</tr>
<tr>
<td>[Multi-Objective] Optimization</td>
<td>[Robust] Task Scheduling etc.</td>
</tr>
<tr>
<td>Cryptology</td>
<td></td>
</tr>
<tr>
<td>Economy</td>
<td></td>
</tr>
<tr>
<td>Life Sciences</td>
<td></td>
</tr>
</tbody>
</table>
The UL HPC platform

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: $\approx 4M\text{€}$ per year
The UL HPC platform

UL HPC Planning 2015-2020

UL HPC Computing Capacity Added in CDC S-02

<table>
<thead>
<tr>
<th>Year</th>
<th>Capacity [TFlops]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>65.09</td>
</tr>
<tr>
<td>2016</td>
<td>130.18</td>
</tr>
<tr>
<td>2017</td>
<td>442.60</td>
</tr>
<tr>
<td>2018</td>
<td>755.02</td>
</tr>
<tr>
<td>2019</td>
<td>1080.46</td>
</tr>
<tr>
<td>2020</td>
<td></td>
</tr>
</tbody>
</table>
UL HPC Planning 2015-2020

UL HPC Storage Capacity Added in CDC S-02

- 2015: 2,88 PB
- 2016: 6,48 PB
- 2017: 10,8 PB
- 2018: 15,12 PB
- 2019: 19,44 PB
- 2020: 0 PB

Raw Storage Capacity [PB]
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
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
The UL HPC platform

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€

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Credit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>Regular HPC resource</td>
<td>0.33 €</td>
</tr>
<tr>
<td>bigmem</td>
<td>Regular HPC resource with huge RAM ($\geq$ 1024 GB)</td>
<td>1.48 €</td>
</tr>
<tr>
<td>bigsmp</td>
<td>SMP node ($\geq$ 16 sockets) with a huge RAM ($\geq$ 1024 GB)</td>
<td>2.45 €</td>
</tr>
</tbody>
</table>
The UL HPC platform

Computing Credits Prices

- Price (Euro)
- EC2 Equivalent
- ULHPC Operating Costs

- d-cluster1
- e-cluster1
- h-cluster1
- r-cluster1
- s-cluster1
- gaia-[1−60]
- gaia-[123−154]
- gaia-[61−62]
- gaia-[63−72]
- gaia-[75−79]
- gaia-[83−122]
- gaia-73
- gaia-74

UL HPC Management Team (UL)
The UL HPC platform

Computing Credits Prices

<table>
<thead>
<tr>
<th>Price (Euro)</th>
<th>Normal</th>
<th>BigMem</th>
<th>BigSMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULHPC Operating Costs</td>
<td>0.33</td>
<td>1.1</td>
<td>1.48</td>
</tr>
<tr>
<td>EC2 Equivalent</td>
<td>1.1</td>
<td>7.49</td>
<td>8.54</td>
</tr>
</tbody>
</table>

UL HPC Management Team (UL)
Summary

1 Preliminaries

2 Overview of the Main HPC Components

3 Interlude

4 The UL HPC platform
   Overview
   Platform Management Tools
   Monitoring
   Statistics & Milestones

5 UL HPC in Practice: Toward an [Efficient] Win-Win Usage
   General considerations
   The OAR Batch Scheduler
   Reporting problems
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 weekends
  - try not to use more than 40 computing cores during working day
  - ... or use the 'besteffort' queue
UL HPC in Practice: Toward an [Efficient] Win-Win Usage

User Doc

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

... aka the rtfm paradigm
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  
   → (eventually) Prepare your experiments  
   → Test your experiment on small size problem  
   → 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

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

http://oar.imag.fr
The OAR Batch Scheduler

- 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

<table>
<thead>
<tr>
<th>Submission</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>interactive</td>
<td><code>oarsub [options] -I</code></td>
</tr>
<tr>
<td>passive</td>
<td><code>oarsub [options] scriptName</code></td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Submission</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>interactive</strong></td>
<td>oarsub [options] -I</td>
</tr>
<tr>
<td><strong>passive</strong></td>
<td>oarsub [options] <strong>scriptName</strong></td>
</tr>
</tbody>
</table>

- Each created job receives 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:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$OAR_NODEFILE</td>
<td>Filename which lists all reserved nodes for this job</td>
</tr>
<tr>
<td>$OAR_JOB_ID</td>
<td>OAR job identifier</td>
</tr>
<tr>
<td>$OAR_RESOURCE_PROPERTIES_FILE</td>
<td>Filename which lists all resources and their properties</td>
</tr>
<tr>
<td>$OAR_JOB_NAME</td>
<td>Name of the job given by the &quot;-n&quot; option of oarsub</td>
</tr>
<tr>
<td>$OAR_PROJECT_NAME</td>
<td>Job project name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Job Type</th>
<th>Max Walltime (hour)</th>
<th>Max #active_jobs</th>
<th>Max #active_jobs_per_user</th>
</tr>
</thead>
<tbody>
<tr>
<td>interactive</td>
<td>12:00:00</td>
<td>10000</td>
<td>5</td>
</tr>
<tr>
<td>default</td>
<td>120:00:00</td>
<td>30000</td>
<td>10</td>
</tr>
<tr>
<td>besteffort</td>
<td>9000:00:00</td>
<td>10000</td>
<td>1000</td>
</tr>
</tbody>
</table>

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`:
  - `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
  - Syntax: `-p "property='value'"`
OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15:
  
  (frontend) $> \text{oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15}$

- 4 cores on a GPU node for 8 hours
  
  (frontend) $> \text{oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"}$

- 2 nodes among the h-cluster1-* nodes (Chaos only)
  
  Total: 24 cores
  
  (frontend) $> \text{oarsub -I -l nodes=2 -p "nodeclass='h'"}$

- 4 cores on 2 GPU nodes + 20 cores on other nodes
  
  (frontend) $> \text{oarsub -I -l \{gpu='YES'\}/nodes=2/core=4 +\{gpu='NO'\}/core=20}$

- A full big SMP node
  
  Total: 160 cores on gaia-74
  
  (frontend) $> \text{oarsub -t bigsmp -I l node=1}$
OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15:
  
  (frontend)$> $ oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15

  Total: 6 cores

- 4 cores on a GPU node for 8 hours
  
  (frontend)$> $ oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"

  Total: 4 cores
OAR (interactive) job examples

- 2 cores on 3 nodes (same enclosure) for 3h15:
  
  (frontend)$> oarsub -I -l /enclosure=1/nodes=3/core=2,walltime=3:15

  Total: 6 cores

- 4 cores on a GPU node for 8 hours

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

  Total: 4 cores

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

  Total: 6 cores

- 4 cores on a GPU node for 8 hours
  
  (frontend)$> oarsub -I -l /core=4,walltime=8 -p "gpu='YES'"

  Total: 4 cores

- 2 nodes among the h-cluster1-* nodes
  
  (Chaos only)
  
  (frontend)$> oarsub -I -l nodes=2 -p "nodeclass='h'"

  Total: 24 cores

- 4 cores on 2 GPU nodes + 20 cores on other nodes
  
  $> oarsub -I -l "\{gpu='YES'\}/nodes=2/core=4+{\{gpu='NO'\}/core=20"

  Total: 28 cores

- A full big SMP node
  
  $> oarsub -t bigsmp -I 1 node=1

  Total: 160 cores on gaia-74
**Some other useful features of OAR**

- **Connect to a running job**
  
  (frontend)$> \text{oarsub -C} \text{ JobID}

- **Cancel a job**
  
  (frontend)$> \text{oardel} \text{ JobID}

- **Status of a jobs**
  
  (frontend)$> \text{oarstat -state -j} \text{ JobID}

- **View the job**
  
  (frontend)$> \text{oarstat}
  
  (frontend)$> \text{oarstat -f -j} \text{ JobID}

- **Get info on the nodes**
  
  (frontend)$> \text{oarnodes}
  
  (frontend)$> \text{oarnodes -l}
  
  (frontend)$> \text{oarnodes -s}

- **Run a best-effort job**
  
  (frontend)$> \text{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

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

# 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

Different runs may not take the same time: **load imbalance.**
Serial tasks with GNU Parallel

17 hours
42% utilization

10 hours
72% utilization
### 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

First checks

1. My issue is probably documented
2. An event is on-going
3. Check the state of your nodes

User doc

cf mail from hpc-platform@uni.lu

Ganglia (especially memory usage)

See next Talk

ONLY NOW consider the following depending on the severity

→ Open a 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},
  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

1 Preliminaries
2 Overview of the Main HPC Components
3 Interlude
4 The UL HPC platform
   Overview

Platform Management Tools
Monitoring
Statistics & Milestones

5 UL HPC in Practice: Toward an [Efficient]
Win-Win Usage
   General considerations
   The OAR Batch Scheduler
   Reporting problems

High Performance Computing (HPC) at UL