Towards better experiments on Grid’5000

Lucas Nussbaum
lucas.nussbaum@loria.fr

ALGORILLE team
LORIA / Nancy-Université
Introduction

- Standard experiment on Grid’5000 today:
  - Quick and dirty shell scripts
    - Best case: quick and dirty Ruby scripts
  - Many manual steps

- We agree that we need to improve the quality of our experiments
  - To push back the limits of what is currently doable
    - Complex experiments at a large scale
  - To increase the quality of our experiments
    - Get more confidence in the obtained results
    - Be able to repeat experiments and reproduce results
Introduction

▶ Standard experiment on Grid’5000 today:
  ▶ Quick and dirty shell scripts
    Best case: quick and dirty Ruby scripts
  ▶ Many manual steps

▶ We agree that we need to improve the quality of our experiments
  ▶ To push back the limits of what is currently doable
    → Complex experiments at a large scale
  ▶ To increase the quality of our experiments
    ▶ Get more confidence in the obtained results
    ▶ Be able to repeat experiments and reproduce results

But how can we do that on Grid’5000?
Experimental testbed (Grid’5000): provides reconfigurable hardware and network, isolation, some instrumentation and monitoring

What’s needed?
What’s needed?

Layer 3

Experimental methodology: experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results

Layer 0

Experimental testbed (Grid’5000): provides reconfigurable hardware and network, isolation, some instrumentation and monitoring

Olivier Richard’s talk

David Margery’s talk
What’s needed?

Layer 3

**Experimental methodology:** experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results

Olivier Richard’s talk

Layer 1

**Basic services:** common tools required by most experiments

- **Interact w/ testbed** find/reserve/configure resources
- **Check resources before using them**
- **Control a large number of nodes**
- **Manage data**
- **Change experimental conditions**
- **Instrument and monitor; extract data**

Layer 0

**Experimental testbed (Grid’5000):** provides reconfigurable hardware and network, isolation, some instrumentation and monitoring

David Margery’s talk
### What’s needed?

**Experimental methodology:** experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results

**Orchestration of experiments:** organize the execution of complex and large-scale experiments; run experiments unattended and efficiently; handles failures; compose experiments

**Basic services:** common tools required by most experiments

<table>
<thead>
<tr>
<th>Interact w/ testbed</th>
<th>Control a large number of nodes</th>
<th>Change experimental conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>find/reserve/configure resources</td>
<td>Manage data</td>
<td>Instrument and monitor; extract data</td>
</tr>
<tr>
<td>Check resources before using them</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Experimental testbed (Grid’5000):** provides reconfigurable hardware and network, isolation, some instrumentation and monitoring

---

Olivier Richard’s talk

David Margery’s talk
What’s needed?

**Experimental methodology:** experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results

**Orchestration of experiments:** organize the execution of complex and large-scale experiments; run experiments unattended and efficiently; handles failures; compose experiments

**Basic services:** common tools required by most experiments

1. Interact w/ testbed find/reserve/configure resources
2. Check resources before using them
3. Control a large number of nodes
4. Manage data
5. Change experimental conditions
6. Instrument and monitor; extract data

**Experimental testbed (Grid’5000):** provides reconfigurable hardware and network, isolation, some instrumentation and monitoring
### What’s needed?

#### Layer 3

**Experimental methodology:** experiment design & planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results.

#### Layer 2

**Orchestration of experiments:** organize the execution of complex and large-scale experiments; run experiments unattended and efficiently; handles failures; compose experiments.

#### Layer 1

**Basic services:** common tools required by most experiments.

1. **Interact w/ testbed** find/reserve/configure resources
2. **Check resources before using them**
3. **Control a large number of nodes**
4. **Manage data**
5. **Change experimental conditions**
6. **Instrument and monitor; extract data**

#### Layer 0

**Experimental testbed (Grid’5000):** provides reconfigurable hardware and network, isolation, some instrumentation and monitoring.
Interacting with the testbed infrastructure

Past:

- Command-line tools: oarsub, kadeploy3, ...
- Hard to interact with in an automated way

Now:

- Grid'5000 API – https://api.grid5000.fr/
- Reference API: description of the testbed
- Jobs API: interact with OAR
- Deployment API: interact with Kadeploy
- Monitoring API: data about availability of resources
- Metrology API: interact with Ganglia

It's (quite) ready, just use it!

(Already 160 users, including 50 very active; 2 millions requests in April 2011)
Interacting with the testbed infrastructure

Past:
- Command-line tools: oarsub, kadeploy3, ...
- Hard to interact with in an automated way

Now:
- Grid’5000 API – https://api.grid5000.fr/
  - Reference API: description of the testbed
  - Jobs API: interact with OAR
  - Deployment API: interact with Kadeploy
  - Monitoring API: data about availability of resources
  - Metrology API: interact with Ganglia

It’s (quite\(^1\)) ready, just use it!
(Already 160 users, including 50 very active; 2 millions requests in April 2011)

\(^1\) description of resources not uniform across sites, and sometimes incorrect. Doesn’t contain everything you might need.
Problem: What if the resources you got are not perfect?

- Slow or badly partitioned disks
- Network cards with wrong auto-negotiation (100 Mb/s vs 1 Gb/s)
- Wrong BIOS settings (e.g., hardware prefetching; frequency scaling)
Checking resources before using them

Problem: What if the resources you got are not perfect?

- Slow or badly partitioned disks
- Network cards with wrong auto-negotiation (100 Mb/s vs 1 Gb/s)
- Wrong BIOS settings (e.g. hardware prefetching; frequency scaling)

Current status:

- Up to the user to detect problems
- In the production environment: G5K-checks
  - Will not catch all problems
  - Not usable from deployed environments
- The problem can be mitigated using statistics
  (e.g. detect and exclude outliers)

Possible future:

- G5K-checks runnable from deployed envs, including user-provided tests
Control of a large number of nodes

What most users (still) do:

```bash
for n in $(<nodes); do
  ssh $n command
done
```
Control of a large number of nodes

What most users (still) do:

```bash
for n in $(<nodes); do
    ssh $n command
done
```

Other (better) solutions:

- TakTuk + kanif or Taktuk::Pilot (Perl) – good performance 😊, but UI… 😞
- Ruby’s net-ssh-multi – UI 😊😊, but performance 😞
  - Work in progress: ruby libssh bindings:
    https://github.com/leehambley/libssh.rb
- Clustershell (Python library, CEA) – UI 😊😊, but performance 😞
- pdsh, dsh, dish, clusterssh
- GNU parallel, xargs
Managing data

Different use cases:

- **Broadcast**: push identical data to all nodes
  - Asynchronous or synchronous start
- **Scatter**: push different data to each node
- **Gather**: gather data from nodes
- **Share**: data between nodes

No silver bullet, but several solutions:

- standard tools: Rsync, SSHFS, NFS
- Kastafior (chain-based data broadcast)
Managing data

Different use cases:

- **Broadcast**: push identical data to all nodes
  - Asynchronous or synchronous start
- **Scatter**: push different data to each node
- **Gather**: gather data from nodes
- **Share**: data between nodes

No silver bullet, but several solutions:

- Standard tools: Rsync, SSHFS, NFS
- Kastafior (chain-based data broadcast)
Changing experimental conditions

Sometimes Grid’5000 is too perfect:

- Need to degrade experimental conditions
  - Emulate different CPU or network performance
    - Some standard tools (Linux TC, etc.)
    - **Modelnet**: used by Davide Frey in Rennes
    - **Wrekavoc**: emulate different CPU speeds & network topology
      *New release expected fall 2011!*
  - Inject load and faults
    - only ad-hoc tools?
Instrumentation and monitoring

Tools provided by the testbed:
- Ganglia, through the monitoring API
  - Can be customized to push your own data
- Metroflux
  - Monitor inter-site traffic with high accuracy
  - Only in Lyon and Lille

Standard tools:
- Systemtap, perf, PAPI, etc.
- ...
## What’s needed?

<table>
<thead>
<tr>
<th><strong>Layer 3</strong></th>
<th><strong>Experimental methodology:</strong></th>
<th>experiment design &amp; planning; description of scenarios, of experimental conditions; definition of metrics; laboratory journal; analysis and visualization of results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Layer 2</strong></td>
<td><strong>Orchestration of experiments:</strong></td>
<td>organize the execution of complex and large-scale experiments; run experiments unattended and efficiently; handles failures; compose experiments</td>
</tr>
<tr>
<td><strong>Layer 1</strong></td>
<td><strong>Basic services:</strong></td>
<td>common tools required by most experiments</td>
</tr>
<tr>
<td></td>
<td><strong>Interact w/ testbed</strong></td>
<td>find/reserve/configure resources</td>
</tr>
<tr>
<td></td>
<td><strong>Control a large number of nodes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Change experimental conditions</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Check resources before using them</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Manage data</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Instrument and monitor; extract data</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Layer 0</strong></td>
<td><strong>Experimental testbed (Grid’5000):</strong></td>
<td>provides reconfigurable hardware and network, isolation, some instrumentation and monitoring</td>
</tr>
</tbody>
</table>
Orchestrating experiments

Not a new problem:

- Emulab
- PlanetLab & GENI
- Computational sciences
- Other sciences
- some Grid’5000 attempts
Emulab

Experiment management integrated in the Emulab framework
- Eric Eide, Leigh Stoller, Tim Stack, Juliana Freire, and Jay Lepreau. Integrated scientific workflow management for the emulab network testbed. USENIX’06


http://www.cs.utah.edu/flux/workbench/
PlanetLab & GENI


- Inactive GENI WG: **GENI Experiment Workflow and Services**
  - Scope: What do experimenter-users need from GENI? Consider planning, scheduling, running, debugging, analyzing experiments; long running experiments & how they grow; archiving data.
Computational sciences

- Many scientific workflow management systems for computational sciences
- Kepler, Taverna, Triana, VisTrails, ...

https://kepler-project.org/
Computational sciences

- Many scientific workflow management systems for computational sciences
- Kepler, Taverna, Triana, VisTrails, …

http://www.taverna.org.uk/
Integrated with http://www.myexperiment.org/
The LabVIEW Environment

Engineers and scientists can rapidly and cost-effectively interface with measurement and control hardware, analyze data, share results, and distribute systems through intuitive graphical programming.

What is LabVIEW?

How Can I Use LabVIEW?
Applications are as varied as the engineers who create them. Fortunately, LabVIEW combines the flexibility of a programming language with the power of an advanced engineering tool so users can complete their projects regardless of their unique, custom requirements.

Browse all applications

LabVIEW Product Options
Compare LabVIEW development systems, explore add-ons, and see pricing.

Shop for LabVIEW Products
On Grid’5000

Several attempts already:

▶ GRUDU (Lyon – GRAAL)
▶ NXE (Romaric Guillier, Lyon – RESO)
▶ Expo (Brice Videau & Olivier Richard, Grenoble)
▶ Execo (Matthieu Imbert, Lyon)
  https://gforge.inria.fr/projects/execo/
▶ g5k-campaign (Cyril Rohr, Rennes)
  http://g5k-campaign.gforge.inria.fr/

But:

▶ Not clear where we should go
▶ More attempts are probably needed
Conclusions

- Grid’5000 is a great experimental platform
- But it requires software to support a better experimental process
- Many tools already exist
- More work is needed