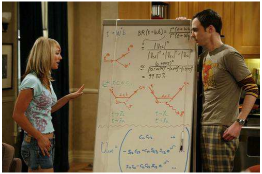
Grid'5000 for high-quality reproducible research

Lucas Nussbaum

Grid'5000 for high-quality reproducible research

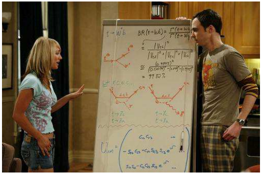
Lucas Nussbaum
lucas.nussbaum@loria.fr

Validation in (Computer) Science



- Two classical approaches for validation:
  - Formal: equations, proofs, etc.
  - Experimental, on a scientific instrument
- Often a mix of both:
  - In Physics
  - In Computer Science

Validation in (Computer) Science



- Two classical approaches for validation:
  - Formal: equations, proofs, etc.
  - Experimental, on a scientific instrument
- Often a mix of both:
  - In Physics
  - In Computer Science
- Very little formal validation in distributed systems research
  - Counter-examples:
    - Worst-case analysis of allocation/scheduling heuristics
    - Properties of algorithms (e.g. deadlock-free)
  - Our scientific objects are often intractable theoretically: too complex, dynamic, heterogeneous, large

(Poor) state of experimentation in CS

- 1994: survey of 400 papers[1]
  - *among published CS articles in ACM journals, 40%-50% of those that require an experimental validation had none*
- 1998: survey of 612 papers[2]
  - *too many papers have no experimental validation at all*
  - *too many papers use an informal (assertion) form of validation*
- 2009 update: *situation is improving*[3]

[1] Paul Lukowicz et al. “Experimental Evaluation in Computer Science: A Quantitative Study”. In: *Journal of Systems and Software* 28 (1994), pages 9–18.

[2] M.V. Zelkowitz and D.R. Wallace. “Experimental models for validating technology”. In: *Computer* 31.5 (1998), pages 23 –31.

[3] Marvin V. Zelkowitz. “An update to experimental models for validating computer technology”. In: *J. Syst. Softw.* 82.3 (Mar. 2009), pages 373–376.

(Poor) state of experimentation in CS (2)

- Most papers do not use even basic statistical tools

Papers published at the Europar conference[4]

Year	Tot. papers	With error bars	Percentage
2007	89	5	5.6
2008	89	3	3.4
2009	86	2	2.4
2010	90	6	6.7
2011	81	7	8.6
2007-2011	435	23	5.3

- 2007: Survey of simulators used in P2P research[5]
  - Most papers use an unspecified or custom simulator

[4] Study carried out by E. Jeannot.

[5] S. Naicken et al. "The state of peer-to-peer simulators and simulations". In: *SIGCOMM Comput. Commun. Rev.* 37.2 (Mar. 2007), pages 95–98.

State of experimentation in other sciences

- 2008: Study shows lower fertility for mices exposed to transgenic maize
  - AFSSA report[6]:
    - *Several calculation errors have been identified*
    - *led to a false statistical analysis and interpretation*

[6] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al. entitled "*Biological effects of transgenic maize NK603xMON810 fed in long-term reproduction studies in mice*"

State of experimentation in other sciences

- 2008: Study shows lower fertility for mices exposed to transgenic maize
  - AFSSA report[6]:
    - *Several calculation errors have been identified*
    - *led to a false statistical analysis and interpretation*
- 2011: CERN Neutrinos to Gran Sasso project: faster-than-light neutrinos
  - 2012: caused by timing system failure

[6] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al. entitled "*Biological effects of transgenic maize NK603xMON810 fed in long-term reproduction studies in mice*"

State of experimentation in other sciences

- 2008: Study shows lower fertility for mices exposed to transgenic maize
  - AFSSA report[6]:
    - *Several calculation errors have been identified*
    - *led to a false statistical analysis and interpretation*
- 2011: CERN Neutrinos to Gran Sasso project: faster-than-light neutrinos
  - 2012: caused by timing system failure
- ☹ Not everything is perfect
- ☺ But some errors are properly identified

[6] Opinion of the French Food Safety Agency (Afssa) on the study by Velimirov et al. entitled "*Biological effects of transgenic maize NK603xMON810 fed in long-term reproduction studies in mice*"

Related to the Reproducible Research movement

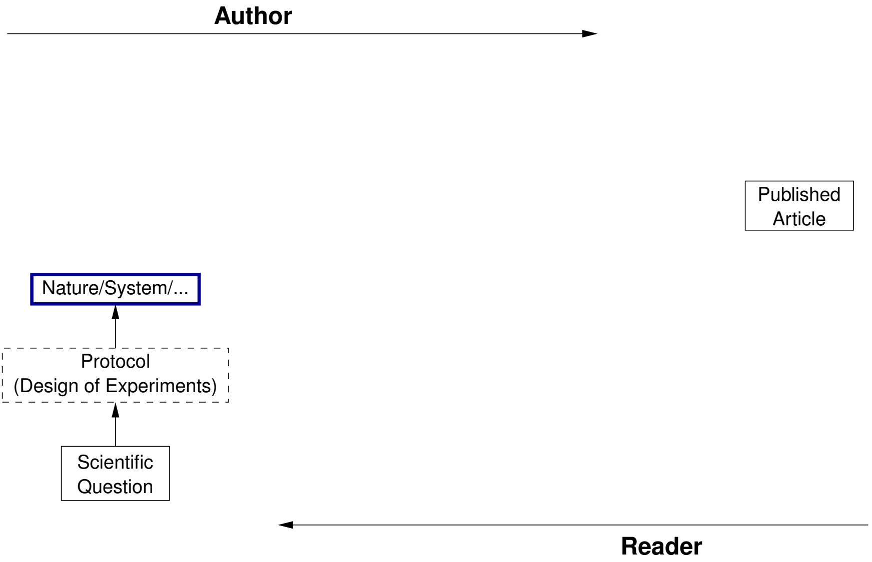
- Mostly in computational sciences
- Explores tools and methods (provenance, executable papers, etc.)
- Different types of experimental reproducibility[7]:
  - *Replications that vary little or not at all with respect to the reference experiment*

    same method, environment, parameters → same result
  - *Replications that do vary but still follow the same method as the reference experiment*

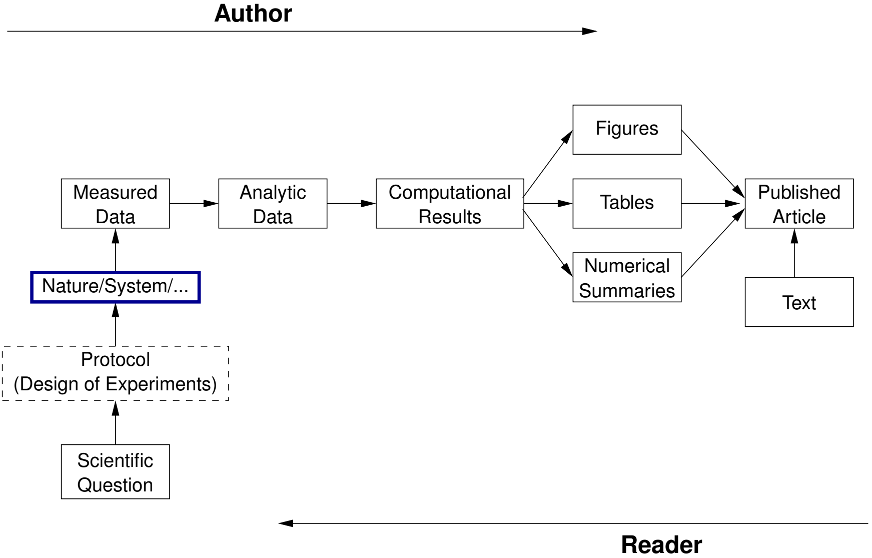
    same method, but different {env., params} → same conclusion
  - *Replications that use different methods to verify the reference experiment results*

    different method → same conclusion

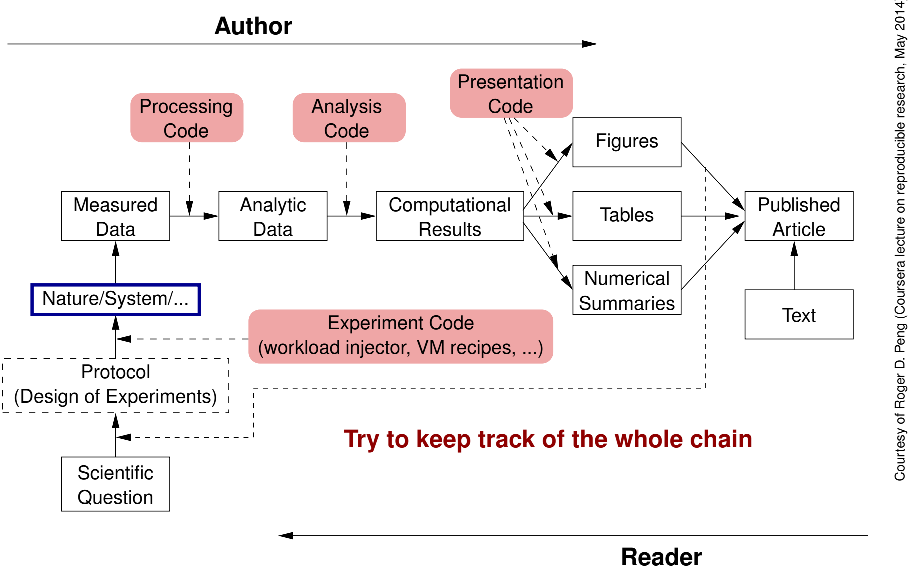
[7] Omar S. Gómez et al. "Replications types in experimental disciplines". In: *ESEM'10*. 2010.

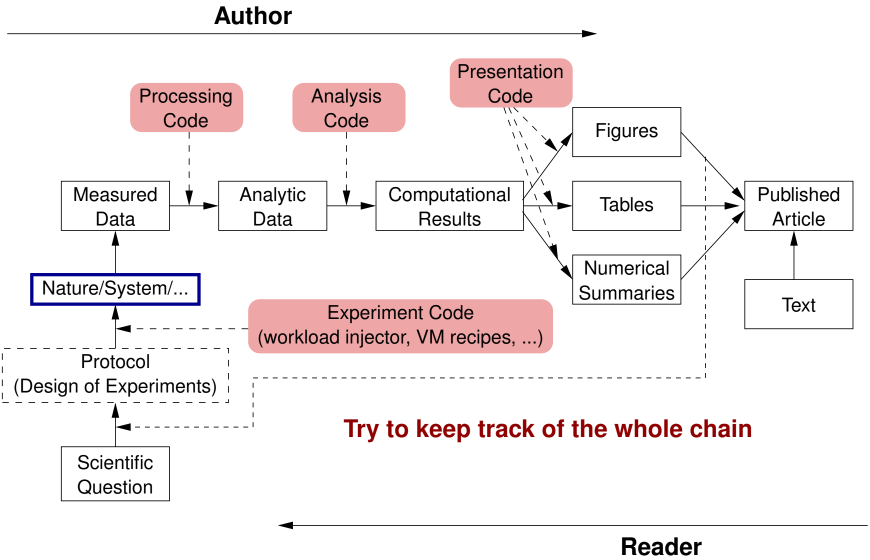


Courtesy of Roger D. Peng (Coursera lecture on reproducible research, May 2014)



Courtesy of Roger D. Peng (Coursera lecture on reproducible research, May 2014)


Author
Processing Code
Analysis Code
Presentation Code
Figures
Measured Data
Analytic Data
Computational Results
Tables
Published Article
Numerical Summaries
Nature/System/...
Text
Experiment Code (workload injector, VM recipes, ...)
Protocol (Design of Experiments)
Try to keep track of the whole chain
Scientific Question
Reader
Courtesy of Roger D. Peng (Coursera lecture on reproducible research, May 2014)



Courtesy of Roger D. Peng (Coursera lecture on reproducible research, May 2014)

- Grid'5000 mission: support high-quality, reproducible experiments on a distributed systems testbed

Two axes of work

- Improve trustworthiness
  - Testbed description
  - Experiment description
  - Control of XP conditions
  - Automate experiments
  - Monitoring & measurement
- Improve scope & scale
  - Handle large number of nodes
  - Automate experiments
  - Handle failures
  - Monitoring & measurement

Both goals raise similar challenges

Outline

Description and verification of the environment

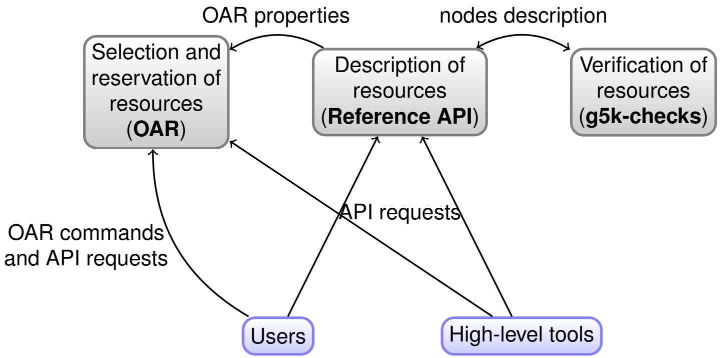
Typical needs:

- How can I find suitable resources for my experiment?
- How sure can I be that the actual resources will match their description?
- What was the hard drive on the nodes I used six months ago?

Description and verification of the environment

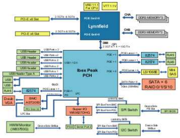
Typical needs:

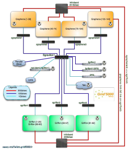
- How can I find suitable resources for my experiment?
- How sure can I be that the actual resources will match their description?
- What was the hard drive on the nodes I used six months ago?



Description and selection of resources

- Describing resources ⇝ understand results
  - Detailed description on the Grid'5000 wiki
  - Machine-parsable format (JSON)
  - Archived (*State of testbed 6 months ago?*)

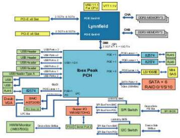


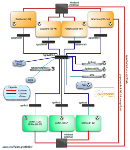



```
"processor": {
  "cache_l2": 8388608,
  "cache_l1": null,
  "model": "Intel Xeon",
  "instruction_set": "",
  "other_description": "",
  "version": "X3440",
  "vendor": "Intel",
  "cache_l1i": null,
  "cache_l1d": null,
  "clock_speed": 2530000000.0
},
"uid": "graphene-1",
"type": "node",
"architecture": {
  "platform_type": "x86_64",
  "smt_size": 4,
  "smp_size": 1
},
"main_memory": {
  "ram_size": 17179869184,
  "virtual_size": null
},
"storage_devices": [
  {
    "model": "Hitachi HDS72103",
    "size": 298023223876.953,
    "driver": "ahci",
    "interface": "SATA II",
    "rev": "JPFO",
    "device": "sda"
  }
],
```

Description and selection of resources

- Describing resources ⇝ understand results
  - Detailed description on the Grid'5000 wiki
  - Machine-parsable format (JSON)
  - Archived (*State of testbed 6 months ago?*)






```
"processor": {
  "cache_l2": 8388608,
  "cache_l1": null,
  "model": "Intel Xeon",
  "instruction_set": "",
  "other_description": "",
  "version": "X3440",
  "vendor": "Intel",
  "cache_l1i": null,
  "cache_l1d": null,
  "clock_speed": 2530000000.0
},
"uid": "graphene-1",
"type": "node",
"architecture": {
  "platform_type": "x86_64",
  "smt_size": 4,
  "smp_size": 1
},
"main_memory": {
  "ram_size": 17179869184,
  "virtual_size": null
},
"storage_devices": [
  {
    "model": "Hitachi HDS72103",
    "size": 298023223876.953,
    "driver": "ahci",
    "interface": "SATA II",
    "rev": "JPFO",
    "device": "sda"
  }
],
```


- Selecting resources
  - OAR database filled from JSON

```
oarsub -p "wattmeter='YES' and gpu='YES'"
oarsub -l "cluster='a'/nodes=1+cluster='b' and
      eth10g='Y'/nodes=2,walltime=2"
```

Verification of resources

- Inaccuracies in resources descriptions $\rightsquigarrow$ dramatic consequences:
  - Mislead researchers into making false assumptions
  - Generate wrong results $\rightsquigarrow$ retracted publications!
- Happen frequently: maintenance, broken hardware (e.g. RAM)

Verification of resources

- Inaccuracies in resources descriptions $\rightsquigarrow$ dramatic consequences:
  - Mislead researchers into making false assumptions
  - Generate wrong results $\rightsquigarrow$ retracted publications!
- Happen frequently: maintenance, broken hardware (e.g. RAM)
- Our solution: g5k-checks
  - Runs at node boot (can also be run manually by users)
  - Retrieves current description of node in Reference API
  - Acquire information on node using OHAI, ethtool, etc.
  - Compare with Reference API

Verification of resources

- Inaccuracies in resources descriptions $\rightsquigarrow$ dramatic consequences:
  - Mislead researchers into making false assumptions
  - Generate wrong results $\rightsquigarrow$ retracted publications!
- Happen frequently: maintenance, broken hardware (e.g. RAM)
- Our solution: g5k-checks
  - Runs at node boot (can also be run manually by users)
  - Retrieves current description of node in Reference API
  - Acquire information on node using OHAI, ethtool, etc.
  - Compare with Reference API
- Future work (maybe?)
  - Verification of performance, not just availability and configuration of hardware (hard drives, network, etc.)
  - Provide tools to capture the state of the testbed $\rightsquigarrow$ archival with the rest of the experiment's data

Outline

Reconfiguring the testbed

- Typical needs:
  - How can I install $SOFTWARE on my nodes?
  - How can I add $PATCH to the kernel running on my nodes?
  - Can I run a custom MPI to test my fault tolerance work?
  - How can I experiment with that Cloud/Grid middleware?

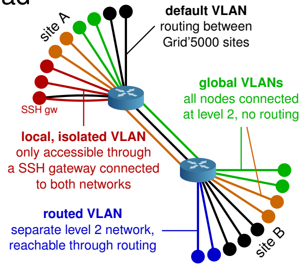
Reconfiguring the testbed

- Typical needs:
  - How can I install $SOFTWARE on my nodes?
  - How can I add $PATCH to the kernel running on my nodes?
  - Can I run a custom MPI to test my fault tolerance work?
  - How can I experiment with that Cloud/Grid middleware?
- Likely answer on any production facility: **you can't**
- Or: use virtual machines $\leadsto$ experimental bias

Reconfiguring the testbed

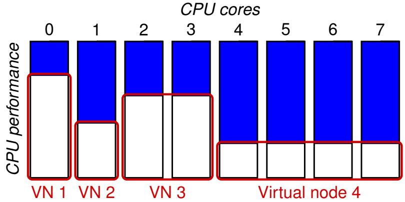
- Operating System reconfiguration with Kadeploy:
  - Provides a *Hardware-as-a-Service* Cloud infrastructure
  - Enable users to deploy their own software stack & get *root* access
  - **Scalable, efficient, reliable and flexible**:
    200 nodes deployed in ~5 minutes (120s with Kexec)
- Customize networking environment with KaVLAN
  - Deploy intrusive middlewares (Grid, Cloud)
  - Protect the testbed from experiments
  - Avoid network pollution
  - By reconfiguring VLANS $\rightsquigarrow$ almost no overhead
  - Recent work: support several interfaces

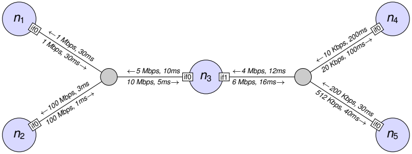
KADEPLOY



Changing experimental conditions

- Reconfigure experimental conditions with Distem
  - Introduce heterogeneity in an homogeneous cluster
  - Emulate complex network topologies





http://distem.gforge.inria.fr/

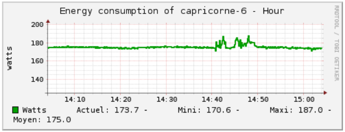
What else can we enable users to change?

- BIOS settings
  - Power management settings
  - CPU features (Hyperthreading, Turbo mode, etc.)
- We need more crazy ideas:
  - Cooling system $\rightsquigarrow$ temperature in the machine room?

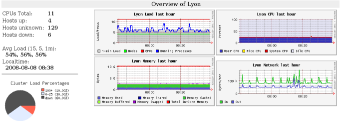
Outline

Monitoring experiments

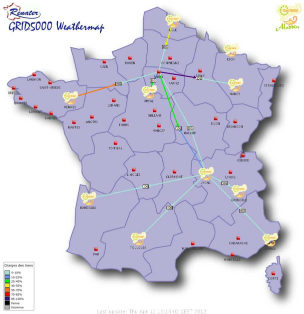
Goal: enable users to understand what happens during their experiment



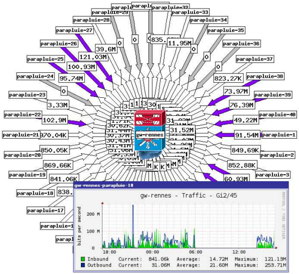
Power consumption



CPU – memory – disk



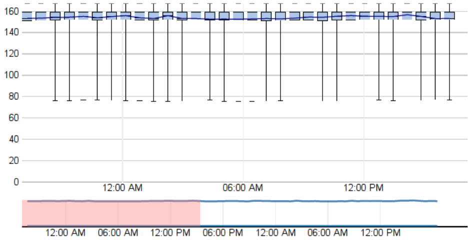
Network backbone



Internal networks

Exporting and analyzing data

- Unified access to monitoring tools through the Grid'5000 API
- Automatically export data during/after an experiment
- Current work: high resolution monitoring for energy & network



Outline

Improving control and description of experiments

- Legacy way of performing experiments: shell commands
  - ☹ time-consuming
  - ☹ error-prone
  - ☹ details tend to be forgotten over time

- Promising solution: automation of experiments
  $\rightsquigarrow$ Executable description of experiments

- Support from the testbed: Grid'5000 RESTful API
  (Resource selection, reservation, deployment)

Tools for automation of experiments

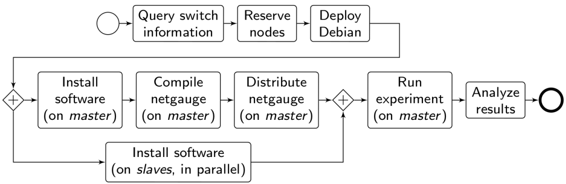
Several projects around Grid'5000 (but not specific to Grid'5000):

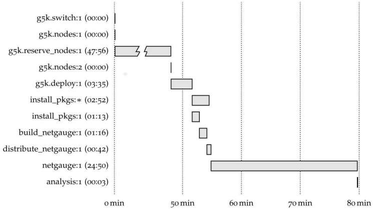
- g5k-campaign (G5K tech team)
- Expo (Cristian Ruiz)
- Execo (Mathieu Imbert)
- XPFlow (Tomasz Buchert)

Features:

- Facilitate scripting of experiments in high-level languages (Ruby, Python)
- Provide useful and efficient abstractions :
  - Testbed management
  - Local & remote execution of commands
  - Data management
- *Engines* for more complex processes

XPFlow






```
engine.process :exp do |site, switch|
    s = run g5k.switch, site, switch
    ns = run g5k.nodes, s
    r = run g5k.reserve_nodes,
        :nodes => ns, :time => '2h',
        :site => site, :type => :deploy
    master = (first_of ns)
    rest = (tail_of ns)
    run g5k.deploy,
        r, :env => 'squeeze-x64-nfs'
    checkpoint :deployed
    parallel :retry => true do
        forall rest do |slave|
            run :install_pkgs, slave
        end
        sequence do
            run :install_pkgs, master
            run :build_netgauge, master
            run :dist_netgauge,
                master, rest
        end
    end
    checkpoint :prepared
    output = run :netgauge, master, ns
    checkpoint :finished
    run :analysis, output, switch
end
```

Experiment description and execution as a Business Process Workflow

Supports parallel execution of activities, error handling, snapshotting, built-in logging, etc.

Outline

A multi-tier challenge

Layer 3

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

Layer 2

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

Layer 1

Basic services: common tools required by most experiments

Interact w/ testbed find, reserve and configure resources	**Manage the environment**	**Manage data**	**Instrument the application & the environment**
Test resources before using them	**Control a large number of nodes**	**Change experimental conditions**	**Monitoring and data collection**

Layer 0

Experimental testbed (e.g Grid'5000, FutureGrid):
reconfigurable hardware and network; isolation; some instrumentation and monitoring

Conclusions

- Grid'5000: a testbed for high-quality, reproducible research on HPC, Clouds and Big Data
- With a unique combination of features
  - Description and verification of testbed
  - Reconfiguration (hardware, network)
  - Monitoring
  - Support for automation of experiments
- Paving the way to Open Science of HPC and Cloud – long term goals:
  - Fully automated execution of experiments
  - Automated tracking + archiving of experiments and associated data

One could determine the age of a science by looking at the state of its measurement tools.

Gaston Bachelard – *La formation de l'esprit scientifique*, 1938

Bibliography

- Resources management:
  - Resources Description, Selection, Reservation and Verification on a Large-scale Testbed. `http://hal.inria.fr/hal-00965708`
- Kadeploy:
  - Kadeploy3: Efficient and Scalable Operating System Provisioning for Clusters. `http://hal.inria.fr/hal-00909111`
- KaVLAN, Virtualization, Clouds deployment:
  - Adding Virtualization Capabilities to the Grid'5000 testbed. `http://hal.inria.fr/hal-00946971`
  - Enabling Large-Scale Testing of IaaS Cloud Platforms on the Grid'5000 Testbed. `http://hal.inria.fr/hal-00907888`