Grid'5000: a Large-Scale Instrument for Parallel and Distributed Computing Experiments

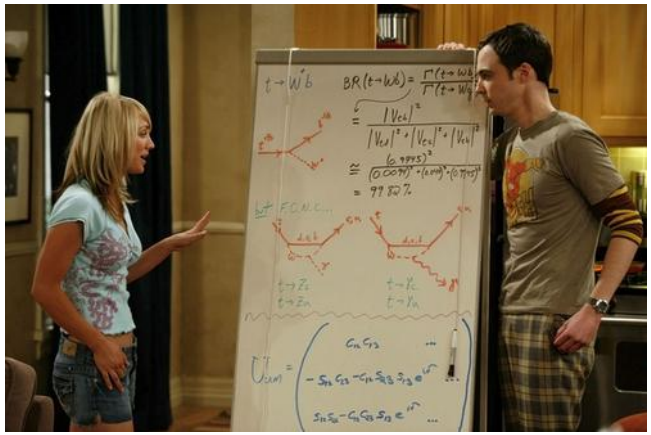
Lucas Nussbaum

Joint work with S. Delamare, F. Desprez, E. Jeanvoine, A. Lebre, L. Lefevre, D. Margery, P. Morillon, P. Neyron, C. Perez, O. Richard and many others

CloudCom'2016

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, Chemistry, Biology, etc.
  - In Computer Science



Distributed computing: a peculiar field in CS

- Performance and scalability are central to results
  - But depend greatly on the environment (hardware, network, software stack, etc.)
  - Many contributions are about *fighting* the environment
    - Making the most out of limited resources
    - Handling performance imbalance $\rightsquigarrow$ load balancing
    - Handling faults $\rightsquigarrow$ fault tolerance
    - Hiding complexity $\rightsquigarrow$ abstractions: middlewares, runtimes

Distributed computing: a peculiar field in CS

- Performance and scalability are central to results
  - But depend greatly on the environment (hardware, network, software stack, etc.)
  - Many contributions are about *fighting* the environment
    - Making the most out of limited resources
    - Handling performance imbalance $\rightsquigarrow$ load balancing
    - Handling faults $\rightsquigarrow$ fault tolerance
    - Hiding complexity $\rightsquigarrow$ abstractions: middlewares, runtimes
- Validation of most contributions require experiments
  - Very little formal validation
  - Even for more theoretical work $\rightsquigarrow$ simulation (SimGrid, CloudSim)

Distributed computing: a peculiar field in CS

- Performance and scalability are central to results
  - But depend greatly on the environment (hardware, network, software stack, etc.)
  - Many contributions are about *fighting* the environment
    - Making the most out of limited resources
    - Handling performance imbalance $\rightsquigarrow$ load balancing
    - Handling faults $\rightsquigarrow$ fault tolerance
    - Hiding complexity $\rightsquigarrow$ abstractions: middlewares, runtimes
- Validation of most contributions require experiments
  - Very little formal validation
  - Even for more theoretical work $\rightsquigarrow$ simulation (SimGrid, CloudSim)
- But experimenting is difficult and time-consuming, but often neglected
  - **How could we perform *better* experiments ?**
  - Very similar to (not computational) biology or physics

Related to the Reproducible Research movement

- Originated in computational sciences
- Explores tools and methods (provenance, executable papers, etc.)
- Different types of experimental reproducibility[1]:
  - *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*

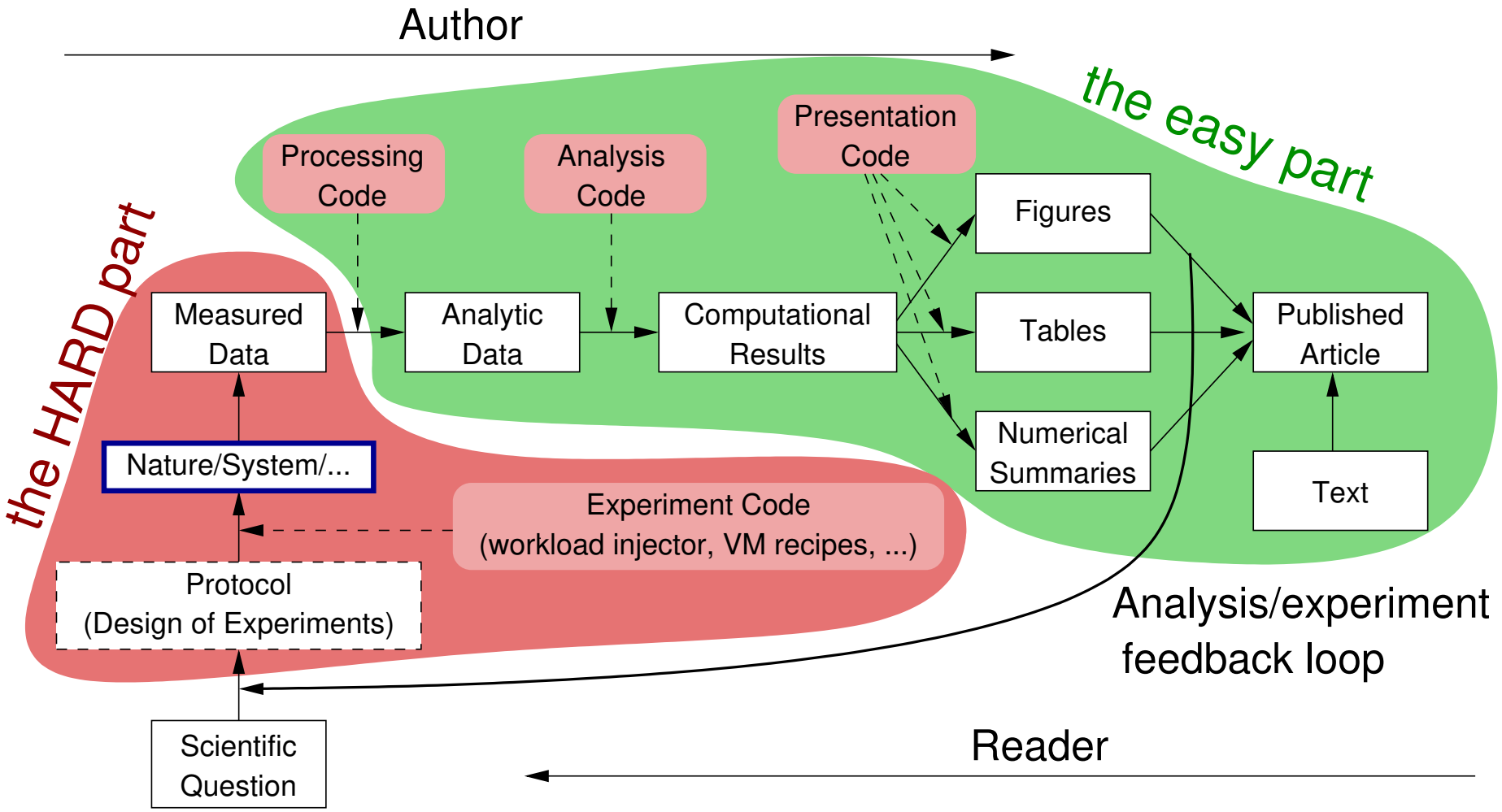
    same method, but different {env., params} → same conclusion

  - *Replications that use different methods to verify the reference experiment results*

    different method → same conclusion

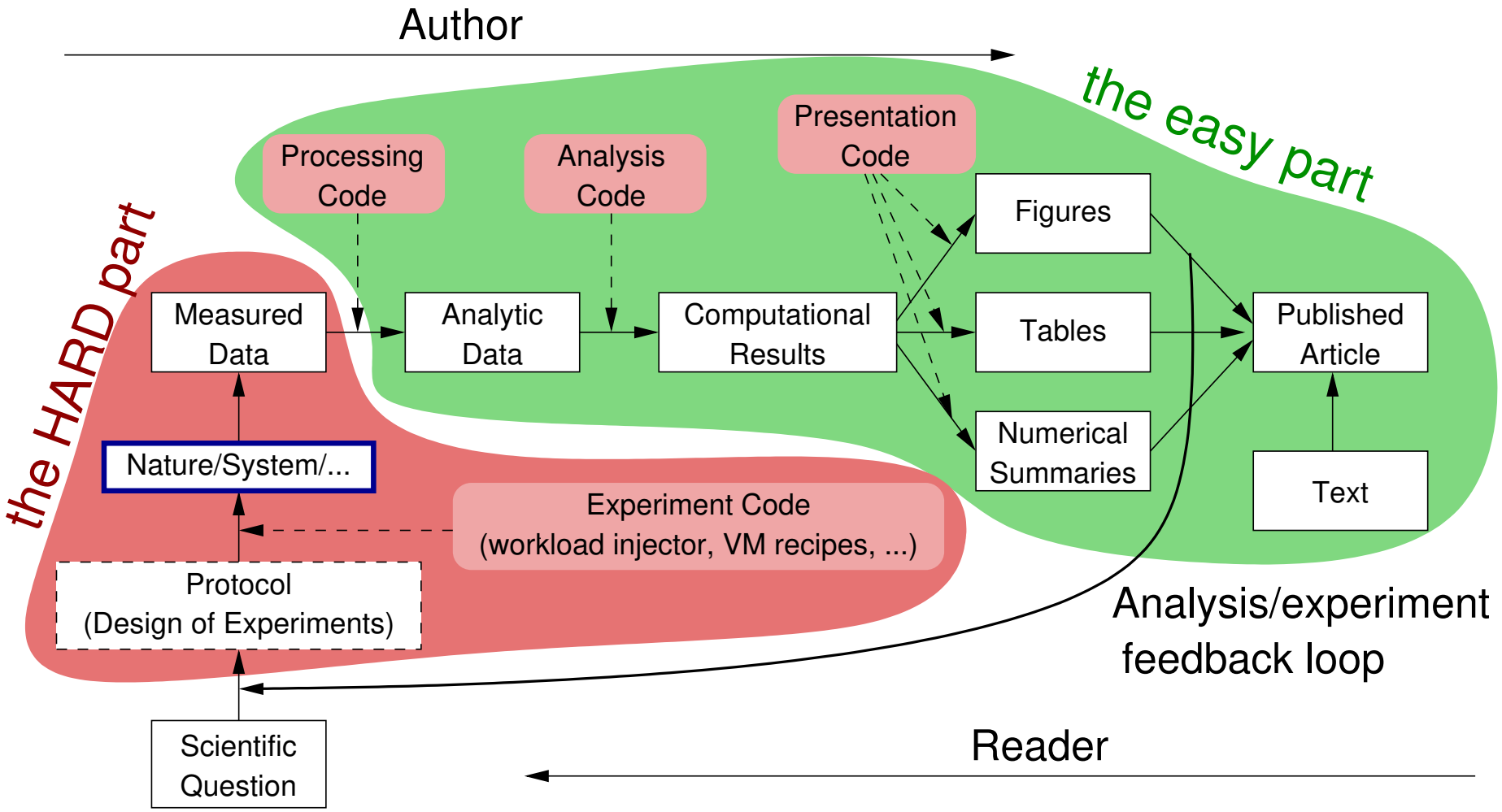
[1] Omar S. Gómez et al. "Replications types in experimental disciplines". In: *ESEM'10*. 2010.

What's an experiment: the research pipeline



Based on figure from Roger D. Peng (Coursera lecture on reproducible research)

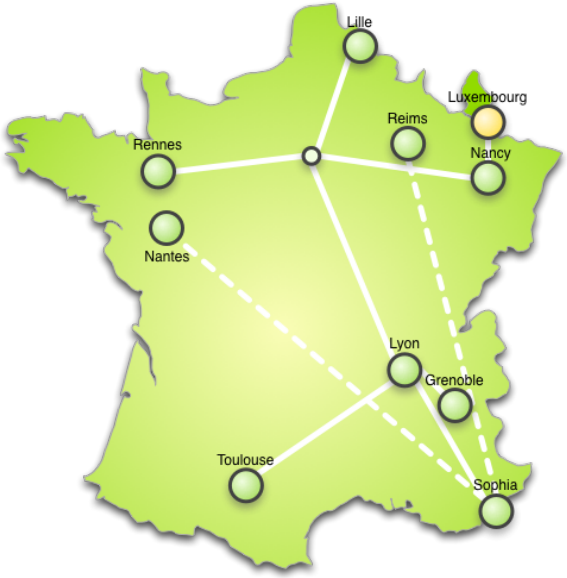
What's an experiment: the research pipeline



Based on figure from Roger D. Peng (Coursera lecture on reproducible research)

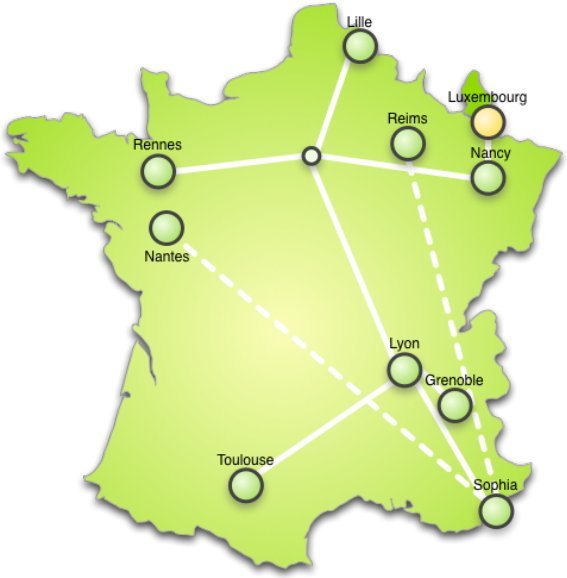
Grid'5000 mission: a large-scale, shared testbed to support high-quality, reproducible experiments

The Grid'5000 testbed



- **One of the world-leading testbeds for distributed computing**
  - 8 sites, 30 clusters, 840 nodes, 8490 cores
  - Dedicated 10-Gbps backbone network
  - 550 users and 100 publications per year

The Grid'5000 testbed



- **One of the world-leading testbeds for distributed computing**
  - 8 sites, 30 clusters, 840 nodes, 8490 cores
  - Dedicated 10-Gbps backbone network
  - 550 users and 100 publications per year
- A meta-grid, meta-cloud, meta-cluster, meta-data-center:
  - Used by CS researchers in HPC / Clouds / Big Data / Networking
  - To experiment in a fully controllable and observable environment
  - **Design goals:**
    - ★ **Support high-quality, reproducible experiments**
    - ★ **On a large-scale, shared infrastructure**

Landscape – cloud & experimentation

- Public cloud infrastructures (AWS, Azure, Google, etc.)
  - ☹ No information/guarantees on placement, multi-tenancy, real performance
- Private clouds: Shared observable infrastructures
  - ☺ Monitoring & measurement
  - ☹ No control over infrastructure settings
  - ⇝ Ability to understand experiment results
- On-demand clouds – dedicated observable infrastructures (BonFIRE)
  - ☺ Limited ability to alter infrastructure
- Bare-metal as a service, fully reconfigurable infrastructure (Grid'5000)
  - ☺ Control/alter all layers, including virtualization technology, operating system, networking

Outline

Discovering resources from their description

- Describing resources ⇝ understand results
  - Covering nodes, network equipment, topology
  - Machine-parsable format (JSON) ⇝ scripts
  - 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"
  }
],
```

Discovering resources from their description

- Describing resources ⇝ understand results
  - Covering nodes, network equipment, topology
  - Machine-parsable format (JSON) ⇝ scripts
  - Archived (*State of testbed 6 months ago?*)
- Verifying the description
  - Avoid inaccuracies/errors ⇝ wrong results
  - Could happen frequently: maintenance, broken hardware (e.g. RAM)
  - Our solution: g5k-checks
    - ★ Runs at node boot (or manually by users)
    - ★ Acquires info using OHAI, ethtool, etc.
    - ★ Compares with Reference API


```
"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"
  }
],
```

Discovering resources from their description

- Describing resources ⇝ understand results
  - Covering nodes, network equipment, topology
  - Machine-parsable format (JSON) ⇝ scripts
  - Archived (*State of testbed 6 months ago?*)
- Verifying the description
  - Avoid inaccuracies/errors ⇝ wrong results
  - Could happen frequently: maintenance, broken hardware (e.g. RAM)
  - Our solution: g5k-checks
    - ★ Runs at node boot (or manually by users)
    - ★ Acquires info using OHAI, ethtool, etc.
    - ★ Compares with Reference API
- Selecting resources
  - OAR database filled from Reference API

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


```
"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"
  }
],
```

Outline

Reconfiguring the testbed

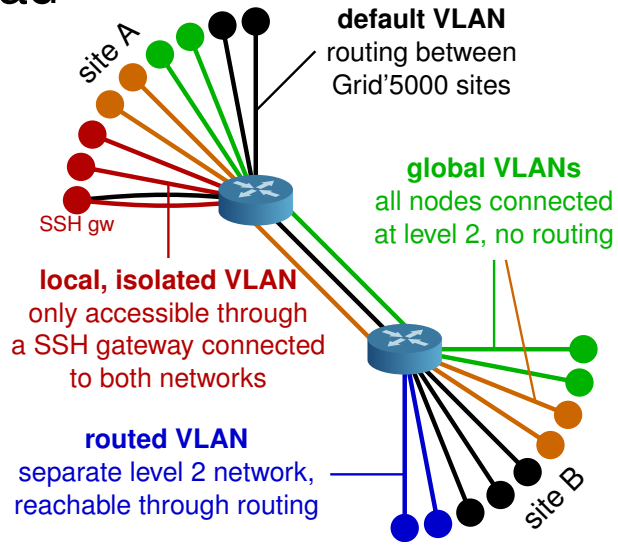
- Typical needs:
  - Install specific software
  - Modify the kernel
  - Run custom distributed middlewares (Cloud, HPC, Grid)
  - Keep a stable (over time) software environment

Reconfiguring the testbed

- Typical needs:
  - Install specific software
  - Modify the kernel
  - Run custom distributed middlewares (Cloud, HPC, Grid)
  - Keep a stable (over time) software environment
- Likely answer on any production facility: **you can't**
- Or:
  - Install in $HOME, modules $\rightsquigarrow$ no root access, handle custom paths
  - Use virtual machines $\rightsquigarrow$ experimental bias (performance), limitations
  - Containers: kernel is shared $\rightsquigarrow$ various limitations

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
- Customize networking environment with KaVLAN
  - Protect the testbed from experiments (Grid/Cloud middlewares)
  - Avoid network pollution
  - By reconfiguring VLANS $\rightsquigarrow$ almost no overhead



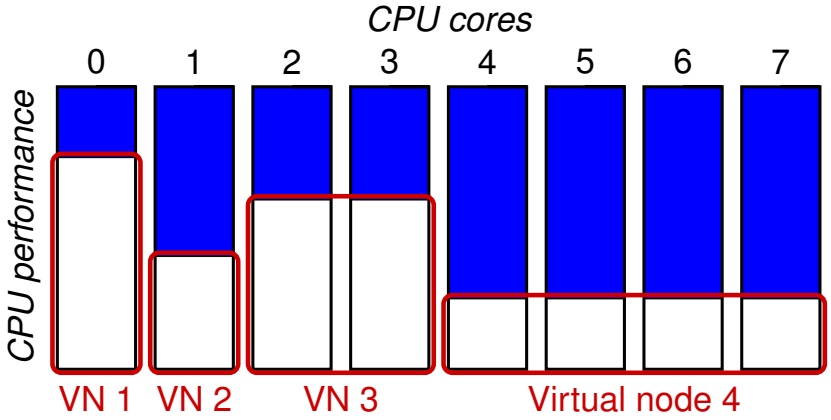
Creating and sharing Kadeploy images

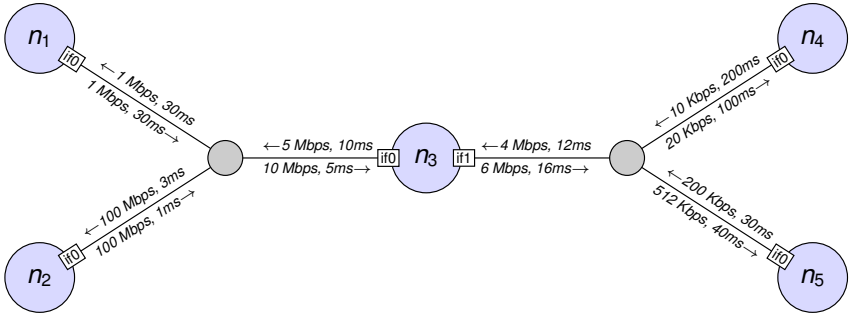
- When doing manual customization:
  - Easy to forget some changes
  - Difficult to describe
  - The full image must be provided
  - Cannot really serve as a basis for future experiments (similar to binary vs source code)
- Kameleon: Reproducible generation of software appliances
  - Using *recipes* (high-level description)
  - Persistent cache to allow re-generation without external resources (Linux distribution mirror) $\rightsquigarrow$ self-contained archive
  - Supports Kadeploy images, LXC, Docker, VirtualBox, qemu, etc.

http://kameleon.imag.fr/

Changing experimental conditions

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

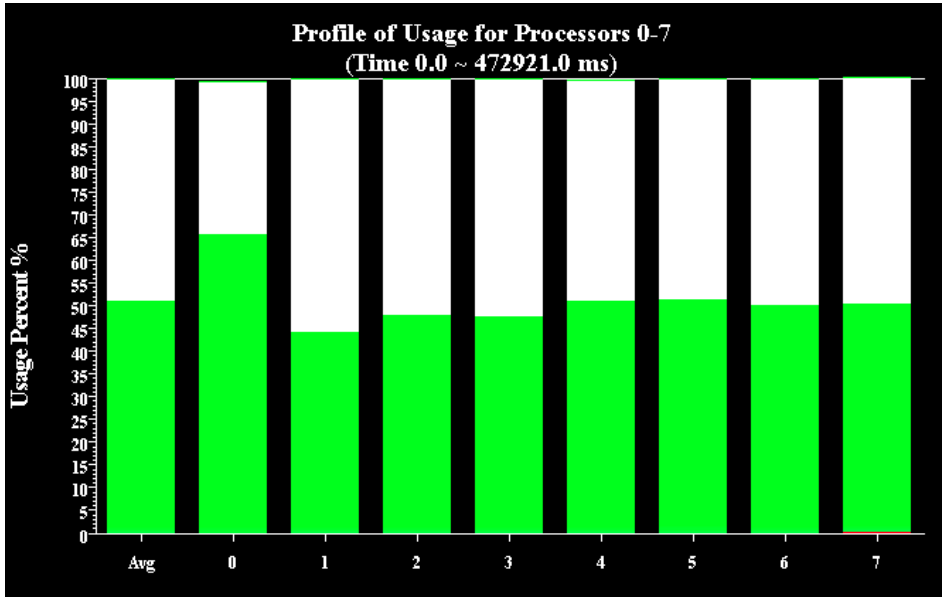




http://distem.gforge.inria.fr/

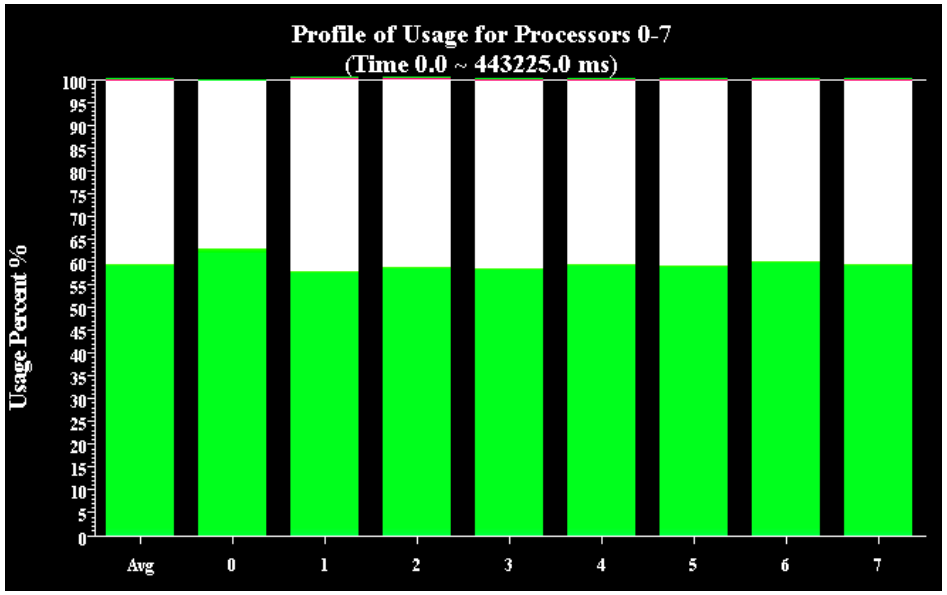
Testing Charm++ load balancing with Distem

No load balancing



total run time: 473s
Average CPU usage: 51%

RefineLB



total run time: 443s
Average CPU usage: 59%

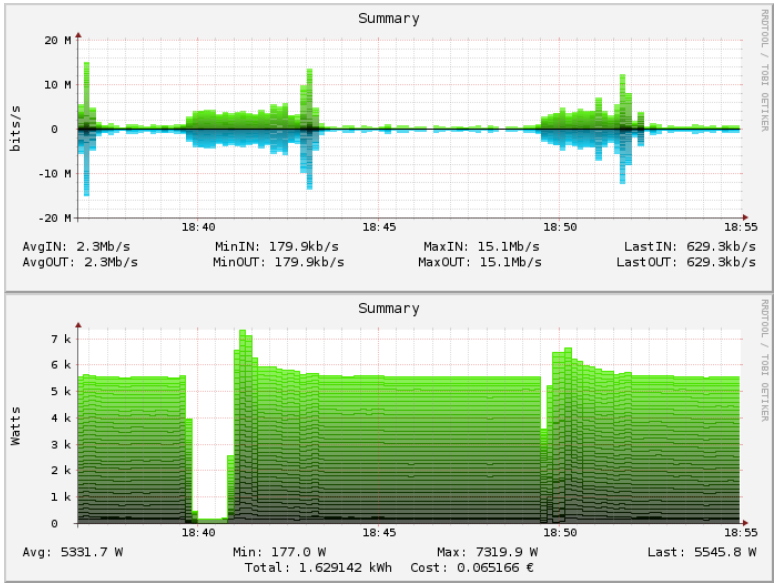
- Every 2 minutes, 1/8 of the nodes are downclocked for 2 minutes
- On the figure, node 0 has been downclocked
- Visible improvement thanks to load balancing

Outline

Monitoring experiments

Goal: enable users to understand what happens during their experiment

- System-level probes (usage of CPU, memory, disk, with Ganglia)
- Infrastructure-level probes
  - Network, power consumption
  - Captured at high frequency (≈1 Hz)
  - Live visualization
  - REST API
  - Long-term storage



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
- Similar problem-space as *configuration mgmt*, *infrastructure as code*
  - ♦ But not just the initial setup
- Support from the testbed: Grid'5000 RESTful API
  (Resource selection, reservation, deployment, monitoring)

Tools for automation of experiments

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

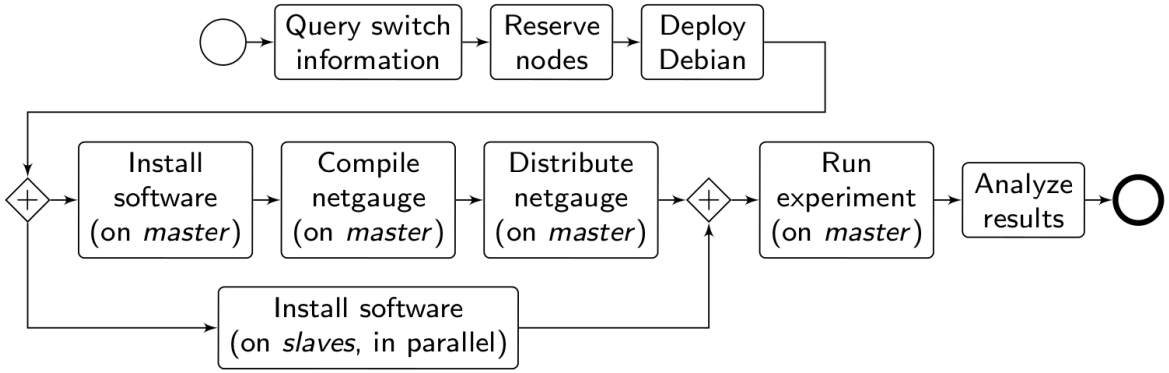
- g5k-campaign (Grid'5000 tech team)
- Expo (Cristian Ruiz)
- Execo (Mathieu Imbert)
- XPFlow (Tomasz Buchert)

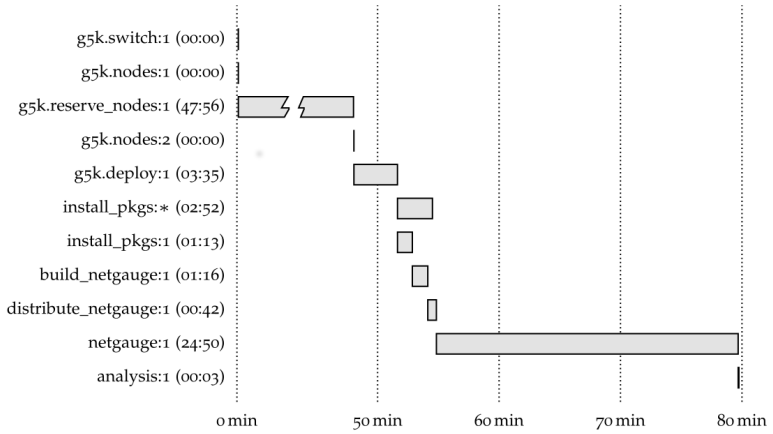
Features:

- Facilitate scripting of experiments in high-level languages (Ruby, Python)
- Provide useful and efficient abstractions :[2]
  - Testbed management
  - Local & remote execution of commands
  - Data management
- *Engines* or *workflows* for more complex processes

[2] Tomasz Buchert et al. "A survey of general-purpose experiment management tools for distributed systems". In: *Future Generation Computer Systems* 45 (2015), pages 1–12.

XPFlow[3]






```
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 and provenance collection, etc.

[3] Tomasz Buchert. "Managing large-scale, distributed systems research experiments with control-flows". PhD Thesis. Université de Lorraine, Jan. 2016.

Outline

Some virtualization & cloud experiments (1/2)

- Virtual machines management
  - Study of the migration process $\rightsquigarrow$ SimGrid model[4]
  - Improving performance of VM migration[5]
  - Evaluation of VM placement strategies[6]

[4] Laurent Pouilloux et al. "SimGrid VM: Virtual Machine Support for a Simulation Framework of Distributed Systems". In: *IEEE Transactions on Cloud Computing* (Sept. 2015).

[5] Pierre Riteau. "Dynamic Execution Platforms over Federated Clouds". PhD Thesis. Université Rennes 1, Dec. 2011.

[6] Adrien Lebre et al. "VMPlaceS: A Generic Tool to Investigate and Compare VM Placement Algorithms". In: *Europar 2015*. Vienne, Austria, Aug. 2015.

Some virtualization & cloud experiments (2/2)

- Energy efficiency of cloud infrastructures[7][8][9]
- Design & improvement of cloud middlewares
  - Autonomic IaaS Cloud: Snooze[10]
  - Fog computing, Distributed OpenStack (DISCOVERY project, Inria/Orange joint lab)[11][12]

[7] Mascha Kurpicz et al. “How much does a VM cost? Energy-proportional Accounting in VM-based Environments”. In: *PDP*. 2016.

[8] Violaine Villebonnet et al. “Towards Generalizing" Big Little" for Energy Proportional HPC and Cloud Infrastructures”. In: *BdCloud*. 2014.

[9] Md Sabbir Hasan et al. “Enabling Green Energy awareness in Interactive Cloud Application”. In: *CloudCom*. 2016.

[10] Eugen Feller. “Autonomic and Energy-Efficient Management of Large-Scale Virtualized Data Centers”. Theses. Université Rennes 1, Dec. 2012.

[11] Frédéric Desprez et al. “Energy-Aware Massively Distributed Cloud Facilities: The DISCOVERY Initiative”. In: *GreenCom*. Dec. 2015.

[12] Bastien Confais et al. “Performance Analysis of Object Store Systems in a Fog/Edge Computing Infrastructures”. In: *CloudCom*. 2016.

Virtualization & Cloud XP requirements

- Efficient provisioning of hypervisors
  - ✓ Kadeploy (support for Xen & KVM)
- Storage (VM images, large datasets)
  - ✓ Storage5k (reserved NFS storage), Ceph clusters
- Networking support
  - ✓ Reconfiguration using KaVLAN
  - ✓ Reservable and routable IP addresses for VMs
- Easy cloud stacks provisioning

Deploying Cloud stacks: challenges

- Cloud stacks are complex beasts
- Short release cycles (6 months) but need to stay up-to-date
- Provisioning tools:
  - Need a low entry barrier (for tutorials etc.)
  - Need support for customization
  - Need to scale (many-nodes experiments)

Deploying Cloud stacks: historical efforts

- Grid'5000 school, **June 2011**: tutorial about Nimbus and OpenNebula (custom-made scripts)
- **April 2012**: workshop about *IaaS on Grid'5000*
  - One solution for OpenStack (custom-made script)
  - Three solutions for OpenNebula (two using Ruby+Chef, one unspecified)
- Grid'5000 school, **December 2012**, tutorials:
  - Nimbus, OpenNebula and Cloudstack (*engines* for an orchestration tool, g5k-campaign)
  - OpenStack (using PuppetLabs' OpenStack modules + script)
    - Maintained until Grizzly (2013.1)
    - **2014**: Attempts to port it to IceHouse (2014.1) by the technical team, additional problems with Neutron (required 3 NICs)
- **2015: Users survey**: 10 different ways to deploy OpenStack on Grid'5000 (various versions, various tools)

Current solution

- Most promising user solution made *official* (work by Matthieu Simonin and Pascal Morillon)
  - Core: **OpenStack's official Puppet modules**
  - Instantiated on an basic Ubuntu 14.04 image
  - Orchestration using Rake ($\approx$ Ruby's make)
  - ☺ Easy to support new releases (complexity in Puppet modules)
  - ☺ Easy to customize (already received users contributions)
  - ☹ Quite slow to deploy (18.5 mins, inc. resources reservation)
- Related work:
  - **CloudLab**: One image per node type, Python + bash scripts for setup, no customization instructions
  - **Chameleon**: DevStack-based single node deployment

Conclusions

- Grid'5000: a testbed for high-quality, reproducible research on HPC, Clouds, Big Data and Networking
- With a unique combination of features
  - Description and verification of testbed
  - Reconfiguration (hardware, network)
  - Monitoring
  - Support for automation of experiments
- Good support for virtualization and cloud experiments
  - Main missing item: real cloud traces
- Try it yourself!
  - Today: tutorial now in room Diekrich
  - Later: free account through the Open Access program `http://www.grid5000.fr/open-access`

More: `https://www.grid5000.fr`

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
- Kameleon: Reproducible Software Appliances for Experimentation. https://hal.inria.fr/hal-01064825
- Distem: Design and Evaluation of a Virtual Experimental Environment for Distributed Systems. https://hal.inria.fr/hal-00724308
- XP management tools:
  - A survey of general-purpose experiment management tools for distributed systems. https://hal.inria.fr/hal-01087519
  - XPFlow: A workflow-inspired, modular and robust approach to experiments in distributed systems. https://hal.inria.fr/hal-00909347
  - Using the EXECO toolbox to perform automatic and reproducible cloud experiments. https://hal.inria.fr/hal-00861886
  - Expo: Managing Large Scale Experiments in Distributed Testbeds. https://hal.inria.fr/hal-00953123
- Kwapi: A Unified Monitoring Framework for Energy Consumption and Network Traffic. https://hal.inria.fr/hal-01167915
- Realis'2014: Reproductibilité expérimentale pour l'informatique en parallélisme, architecture et système. https://hal.inria.fr/hal-01011401