Grid'5000 as a Virtualization and Clouds Testbed

Lucas Nussbaum

with the Grid'5000 architects committee
and the Grid'5000 technical team

Distributed computing: a peculiar field in CS

- Most contributions are validated using experiments
  - Very little formal validation
  - Even for theoretical work $\rightsquigarrow$ simulation (SimGrid)
- 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 (load balancing, fault tolerance, middlewares, etc.)
- Experimenting is difficult and time-consuming
  - **How can one perform *good* experiments?**
  - Very similar to (not computational) biology or physics

The Grid'5000 testbed

- **World-leading testbed for distributed computing**
  - 9 sites, 30 clusters, 859 nodes, 8456 cores
  - Dedicated 10-Gbps backbone network
  - 550 users and 100 publications per year

The Grid'5000 testbed

- **World-leading testbed for distributed computing**
  - 9 sites, 30 clusters, 859 nodes, 8456 cores
  - Dedicated 10-Gbps backbone network
  - 550 users and 100 publications per year
- Not a typical grid / cluster / Cloud, more a meta-grid, meta-cloud:
  - Used by CS researchers in HPC / Clouds / Big Data / Networking to perform experiments
  - **Design goals:**
    - **Large-scale, shared infrastructure**
    - **Support high-quality, reproducible research**

Landscape – Cloud & experimentation

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

Outline

Description and verification of the environment

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

Description and verification of the environment

- Describing resources $\rightsquigarrow$ understand results
  - Covering nodes, network equipment, topology
  - Machine-parsable format (JSON) $\rightsquigarrow$ scripts
  - Archived (*State of testbed 6 months ago?*)
- Verifying the description
  - Avoid inaccuracies/errors $\rightsquigarrow$ 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"
  }
],
```

Description and verification of the environment

- Describing resources $\rightsquigarrow$ understand results
  - Covering nodes, network equipment, topology
  - Machine-parsable format (JSON) $\rightsquigarrow$ scripts
  - Archived (*State of testbed 6 months ago?*)
- Verifying the description
  - Avoid inaccuracies/errors $\rightsquigarrow$ 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:
  - 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?
  - Can I get a stable (over time) software environment for my experiment?

Reconfiguring the testbed

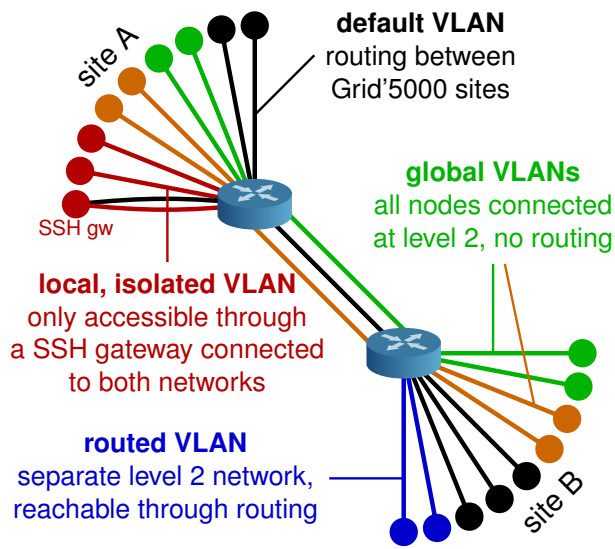
- Typical needs:
  - How can I install $SOFTWARE on my nodes?
  - How can I add $PATCH to the kernel running on my nodes?
  - Can I run a custom MPI to test my fault tolerance work?
  - How can I experiment with that Cloud/Grid middleware?
  - Can I get a stable (over time) software environment for my experiment?
- Likely answer on any production facility: **you can't**
- Or:
  - Use virtual machines $\rightsquigarrow$ experimental bias (performance), limitations
  - Install in $HOME, modules, etc. $\rightsquigarrow$ does not work for everything

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

KADEPLOY



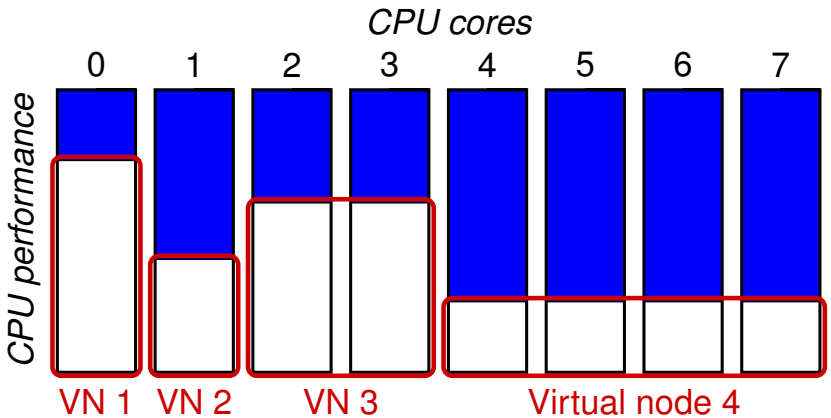
Creating and sharing Kadeploy images

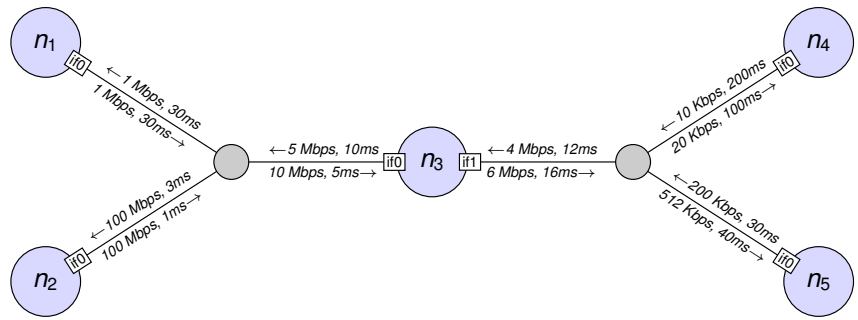
- Avoid manual customization:
  - Easy to forget some changes
  - Difficult to describe
  - The full image must be provided
  - Cannot really serve as a basis for future experiments (similar to binary vs source code)
- Kameleon: Reproducible generation of software appliances
  - Using *recipes* (high-level 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





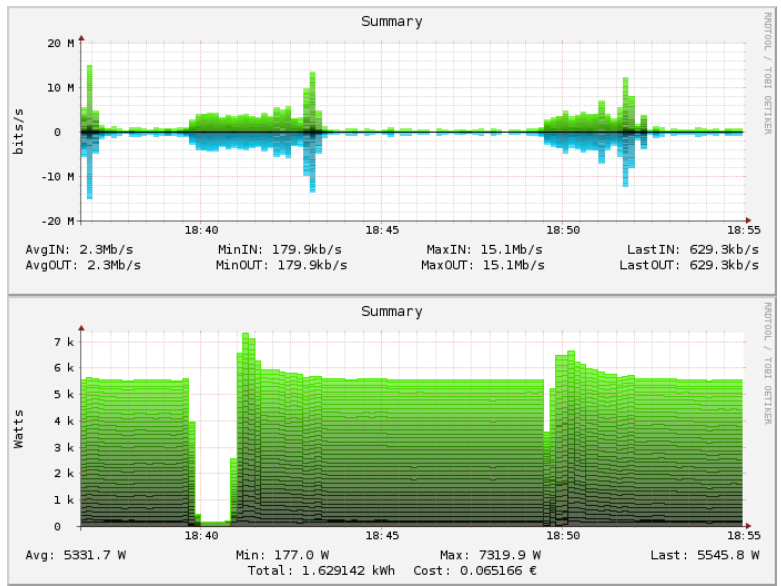
http://distem.gforge.inria.fr/

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

- 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):

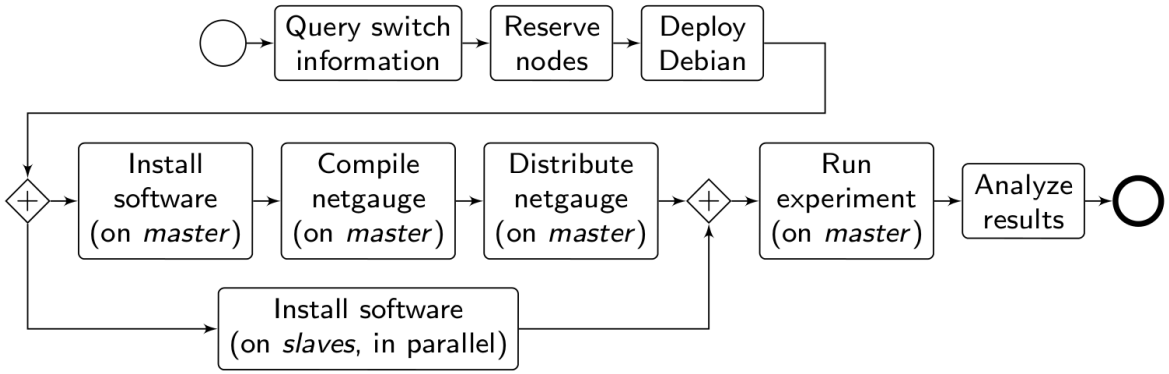
- g5k-campaign (Grid'5000 tech team)
- Expo (Cristian Ruiz)
- Execo (Mathieu Imbert)
- XPFlow (Tomasz Buchert)

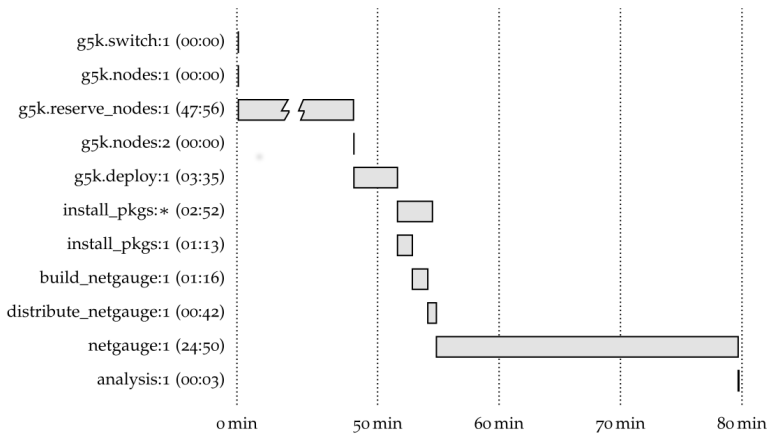
Features:

- Facilitate scripting of experiments in high-level languages (Ruby, Python)
- Provide useful and efficient abstractions :[1]
  - Testbed management
  - Local & remote execution of commands
  - Data management
- *Engines* for more complex processes

[1] 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. DOI: 10.1016/j.future.2014.10.007. URL: https://hal.inria.fr/hal-01087519.

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 and provenance collection, etc.

Outline

Some virtualization & Cloud experiments

- Virtual machines management
  - Study of the migration process $\rightsquigarrow$ SimGrid model[2]
  - Improving performance of VM migration[3]
  - Evaluation of VM placement strategies[4]
- Energy-aware clouds
- Design / Improvement of cloud middlewares
  - Autonomic IaaS Cloud: Snooze[5]
  - Fog computing, Distributed OpenStack (DISCOVERY project, Inria/Orange joint lab)[6]

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

[3] Pierre Riteau. "Dynamic Execution Platforms over Federated Clouds". Theses. Université Rennes 1, Dec. 2011.

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

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

[6] Frédéric Desprez et al. "Energy-Aware Massively Distributed Cloud Facilities: The DISCOVERY Initiative". In: *IEEE International Conference on Green Computing and Communications (GreenCom)*. Sydney, Australia, Dec. 2015, pages 476–477.

Virtualization & Cloud XP requirements

- Efficient provisioning of hypervisors
  - ✓ Kadeploy (support for Xen & KVM)
- Storage (VM images, etc.)
  - ✓ Storage5k (reserved NFS storage), Ceph clusters
- Networking support
- Easy Cloud stacks deployment

IP range reservation: G5K-subnets

- Grid'5000 enables different users to run experiments concurrently
  - Need a mechanism to provide IP ranges for virtual machines
- G5K-subnets adds IP ranges reservation to OAR

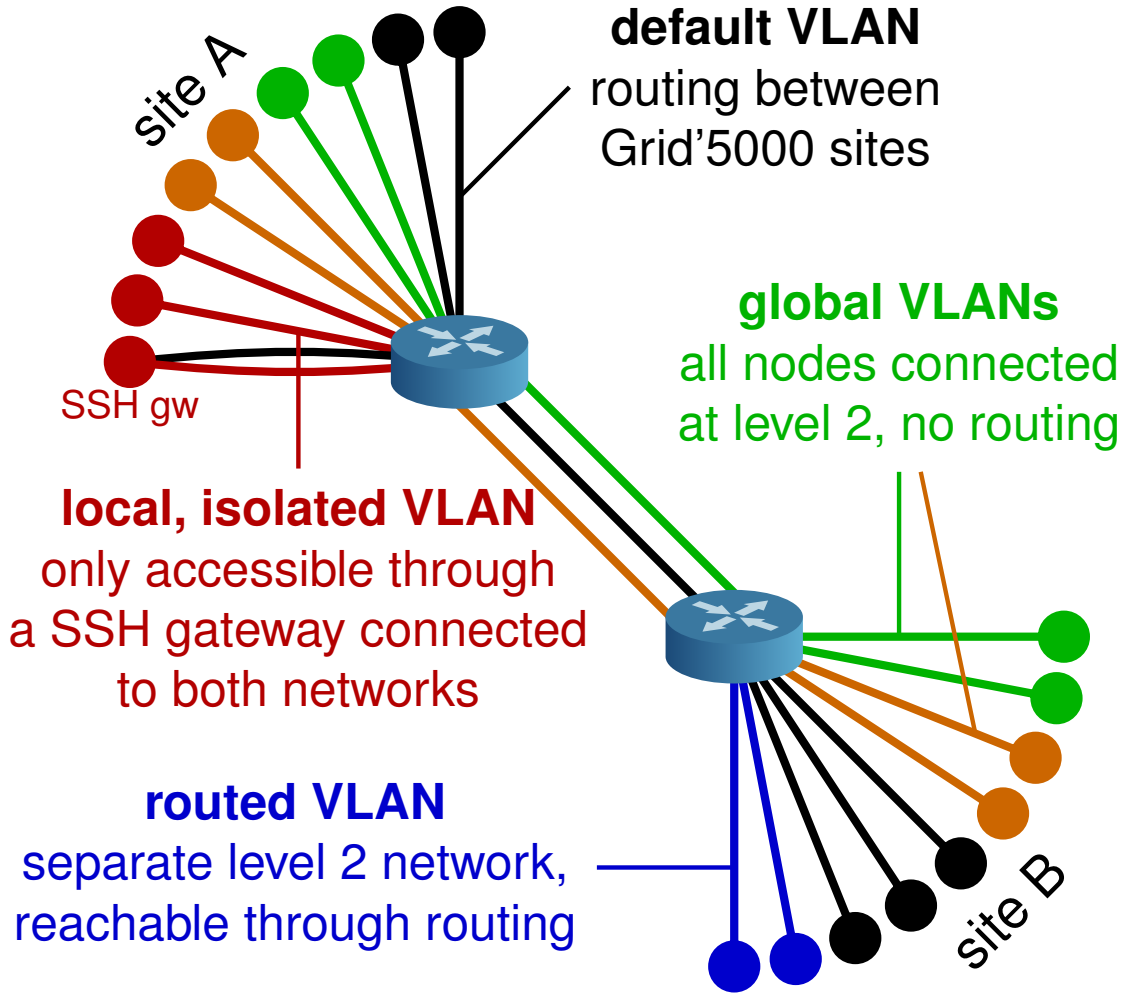
```
oarsub -l slash_22=2+nodes=8 -I
```

- Those IP ranges are routed inside Grid'5000
- But no isolation: one can *steal* IP addresses

Network isolation with KaVLAN

- Reconfigures switches for the duration of a user experiment to achieve **complete level 2 isolation**:
  - Avoid network pollution (broadcast, unsolicited connections)
  - Enable users to start their own DHCP servers
  - Experiment on ethernet-based protocols
  - Interconnect nodes with another testbed without compromising the security of Grid'5000
- Some nodes with several (up to 4) network interfaces
- Relies on **802.1q (VLANs)**
- Compatible with many network equipments
  - Can use SNMP, SSH or telnet to connect to switches
  - Supports Cisco, HP, 3Com, Extreme Networks and Brocade
- Controlled with a command-line client or a REST API

KaVLAN - different VLAN types



Deploying Cloud stacks: challenges

- Cloud stacks are complex beasts
- Short release cycles (6 months) vs staying up-to-date
- 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)

Current solution

- **2015: Users survey**: 10 different ways to deploy OpenStack on Grid'5000 (various versions, various tools)
  - 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 (≈ Ruby's make)
    - ☺ Liberty and Mitaka supported (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, Liberty supported, no customization instructions
  - **Chameleon**: DevStack-based single node deployment, Mitaka supported

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! $\rightsquigarrow$ Open Access program

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