Contributions to Large Scale Distributed Systems
The infrastructure view point

Adrien Lebre
September 1, 2017

President and Examiner:
Claude Jard, Nantes Univ.
Frédéric Desprez, Inria

Reviewers:
Erik Elmroth, Umeå Univ.
Manish Parashar, Rutgers Univ
Pierre Sens, UPMC
If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility...

John Mc Carthy,
Speaking at the MIT centennial in 1961
**Computation**

- **1946**: ENIAC
- **1947**: Transistor
- **1950**: Batchmode
- **1960**: Interactive
- **1967**: First virtualisation attempt
- **1970**: Terminals (clients/server concepts)
- **1971**: micro processor

**Communication**

- **1838**: Telegraph
- **1876**: Telephone
- **1896**: Radio
- **1957**: Satellite
- **1969**: ARPANET
- **1973**: Ethernet
- **1985**: TCP/IP Adoption

**Computational Revolution**

- **1950-1990**: Mainframes
- **1950**: Batchmode
- **1960**: Interactive
- **1967**: First virtualisation attempt
- **1970**: Terminals (clients/server concepts)

**Virtualisation**

- **1967**: First virtualisation attempt
- **1970**: Terminals (clients/server concepts)

**1999**: The Grid

- **2002**: Amazon Initial Compute/Storage services
- **2006**: Amazon EC2 (IaaS)

**Cloud Computing**

- **2002**: Virtualised Infrastructure
- **2010**: Cloud democratisation
- **2015**: Network/Computers Convergence

**Post-Doctoral fellow**: 2006/2008

**(Storage/Grid)**

**MSc/PhD**: 2001/2006

**Cluster (Storage/HPC)**

**Ass. Prof. IMT**: (2008-20XX)


**Clouds and Beyond**
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of compute nodes](image)
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of Compute and Storage nodes](image)
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of frontend, compute nodes, and storage nodes connected to a resource management system.]
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of a network system with a frontend, compute nodes, and storage nodes.](image-url)
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990's)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of Distributed Infrastructures](image)
Utility Computing Infrastructures

• **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

• **Distributed Infrastructures** (since the 1990's)

• **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of a distributed infrastructure system](image)
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990’s)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of a distributed computing infrastructure](image)
Utility Computing Infrastructures

• **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

• **Distributed Infrastructures** (since the 1990's)

• **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

![Diagram of a distributed computing infrastructure](image)
Utility Computing Infrastructures

- **A common objective**: provide computing resources (both hardware and software) in a flexible, transparent, efficient, secured and reliable way

- **Distributed Infrastructures** (since the 1990’s)

- **A lot of challenges** (data-sharing, software/hardware heterogeneity, workload placements, isolation between applications, performance…)

---

![Diagram of a distributed computing infrastructure]

- **Frontend (Resource Management System)**
- **Storage nodes (Distributed File System)**
- **Compute nodes**
(Dynamic VM)

Placement Contributions

Research activities mainly supported by

- IMT Atlantique
- Inria

Cluster-Wide context switch

J.M. Menaud
F. Hermenier
PhD - F. Quesnel
Co supervised with M. Südholt
Distributed VM Scheduler

PhD - J. Pastor
Co supervised with F. Desprez
Locality Aware Placement

IMT Atlantique
Research activities mainly supported by
Inria

Cluster-Wide context switch

J.M. Menaud
F. Hermenier
PhD - F. Quesnel
Co supervised with M. Südholt
Distributed VM Scheduler

PhD - J. Pastor
Co supervised with F. Desprez
Locality Aware Placement
(Dynamic VM) Placement Contributions

Research activities mainly supported by
- IMT Atlantique
- Inria

- PhD - F. Quesnel
  Co supervised with M. Südholt
  Distributed VM Scheduler

- J.M. Menaud
  F. Hermenier
  Cluster-Wide context switch

- PhD - J. Pastor
  Co supervised with F. Desprez
  Locality Aware Placement

Placement Problem

- Jobs 1, 2, 3, and 4 arrive in the queue and have to be scheduled

- **FCFS + Easy back filling**
  Although Jobs 2 and 3 have been backed filled, some resources are unused (dark gray areas)

- **Easy back filling with preemption.**
  Job 4 can be started earlier without impacting Job 1’s performance.

Jobs cannot be easily preempted (OS’s internal states)

Even with preemption, some resources are still not wasted
Virtual Machine
The New Building Block

• **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

• **VM Capabilities**
  • Suspend/Resume
  • Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration

![Diagram showing virtual machines and hypervisor](image-url)
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

• **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

• **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration

[Diagram showing virtual machines and hypervisor]
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

• **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

• **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
Virtual Machine
The New Building Block

- **System virtualization**: One to multiple OSes on a physical machine thanks to a hypervisor (an operating system of OSes)

- **VM Capabilities**
  - Suspend/Resume
  - Live Migration
From Jobs to Virtualised Jobs

- A job is now encapsulated in one or several VMs
From Jobs to Virtualised Jobs

- A job is now encapsulated in one or several VMs
From Jobs to Virtualised Jobs

• A job is now encapsulated in one or several VMs
From Jobs to Virtualised Jobs

- A job is now encapsulated in one or several VMs

- **Challenge**: Maintain viable mappings between VMs and PMs

- Each VM consumes CPU, RAM...

credits: F. Hermenier. OSDI poster session 2008
From Jobs to Virtualised Jobs

- Maintain viable mappings between VMs and PMs.
- MAPE Control loop (leveraging the Entropy framework)
- Make the reconfiguration phase automatic

Cluster-Wide context switch

[VTDC2010]

Current Status

Correct Status

Non-viable manipulations

cost: 3

cost: 2

credits: F. Hermenier, OSDI poster session 2008
Cluster-Wide Context Switch - Evaluations

- **Scheduling policy**: A FIFO queue (priority between jobs to prevent starvation)

- **Testbed** (further details in the manuscript)
  - 11 Working nodes (22 CPUs)
  - A queue of 8 vjobs (NASGrid benchmarks)
  - Each job uses 9 VMs (9CPUs)

Cumulated completion times have been reduced by 40%
Cluster-Wide Context Switch - Evaluations

- **Scheduling policy**: A FIFO queue (priority between jobs to prevent starvation)

- **Testbed** (further details in the manuscript)
  - 11 Working nodes (22 CPUs)
  - A queue of 8 vjobs (NASGrid benchmarks)
  - Each job uses 9 VMs (9CPUs)

11 nodes/72 VMs…

What’s about scalability/reactivity?

credits: A. Simonet, Introduction to Cloud Computing Lecture - Inside a Google DC
Cluster-Wide Context Switch - Evaluations

- **Scheduling policy**: A FIFO queue (priority between jobs to prevent starvation)

- **Testbed** (further details in the manuscript)
  - 11 Working nodes (22 CPUs)
  - A queue of 8 vjobs (NASGrid benchmarks)
  - Each job uses 9 VMs (9CPUs)

11 nodes/72 VMs...

What’s about scalability/reactivity?

credits: A. Simonet, Introduction to Cloud Computing Lecture - Inside a Google DC
Scalability/Reactivity challenge

- Computing Phase: a NP hard problem in most cases
- Most works have been focusing on proposing heuristics to reduce the computing phase but... reconfiguring the infrastructure is time consuming too!

1. Monitoring
2. Computing
3. Reconfiguring

credits: F. Quesnel, PhD defense 2013
Scalability/Reactivity challenge

- Computing Phase: a NP hard problem in most cases

- Most works have been focusing on proposing heuristics to reduce the computing phase but… reconfiguring the infrastructure is time consuming too!

credits: F. Quesnel, PhD defense 2013
Scalability/Reactivity challenge

- Computing Phase: a NP hard problem in most cases

- Most works have been focusing on proposing heuristics to reduce the computing phase but... reconfiguring the infrastructure is time consuming too!

credits: F. Quesnel, PhD defense 2013
Scalability/Reactivity challenge

- Computing Phase: a NP hard problem in most cases
- Most works have been focusing on proposing heuristics to reduce the computing phase but... reconfiguring the infrastructure is time consuming too!

Can we reduce all phases?

credits: F. Quesnel, PhD defense 2013
Leverage P2P Algorithms

• Make dynamic partitioning of the system according to the effective usage of resources
• Make direct cooperations between hypervisors (no service node)

• **Distributed Virtual Machine Scheduler**
  • Event driven / P2P Like system
  • Local interactions between nodes
  • Scheduling performed on partitions of the system, created dynamically (nodes are reserved for an exclusive use by a scheduler, to prevent several schedulers from migrating the same VMs)

An Event occurs on a node

Can current node scheduler calculate valid schedule?

- no
- yes

Contact neighbor and ask it to solve the problem

Apply the schedule

[CCPE2012]
Understanding DVMS

Partition
Understanding DVMS
Understanding DVMS

Partition

Understanding DVMS

Partition

Understanding DVMS
Understanding DVMS
Understanding DVMS
Understanding DVMS

Partition

DVMS Evaluations

- Development of a PoC
- Evaluations (in-vivo) up to 5KVMs
- IEEE Scale challenge 2013
DVMS Evaluations

- Development of a PoC
- Evaluations (in-vivo) up to 5KVMs
- IEEE Scale challenge 2013

It looks promising...How can we test it at scale? How can we compare with others approaches?
VM Placement

(Hot Topic) Problem
VM Placement (Hot Topic) Problem
VM Placement
(Hot Topic) Problem
VM Placement (Hot Topic) Problem

Lots of articles (too many ?)
VM Placement (Hot Topic) Problem

Lots of articles (too many ?)

Evaluations are performed either at a low scale for in vivo experiments or with ad-hoc simulators. How can we evaluate/compare them?
VM Simulator Toolkits

Cluster-Wide context switch

Distributed VM Scheduler

Locality Aware Placement

French ANR SONGS Project

Hemera Inria Large Scale Initiative

Discovery Inria Project Lab

EU BigStorage Project

VMPlaceS

T. L. Nguyen
PhD

SimGridVM: VM abstractions

T. Hirofuchi
Postdoc/Invited researcher

VMPlaceS

A. Simonet
Postdoc

Energy dimension

T. L. Nguyen
PhD

Boot time model

IMT Atlantique

Inria

Research activities mainly supported by

EU BigStorage Project

Hemera Inria Large Scale Initiative

Discovery Inria Project Lab

Inria
VM Simulator Toolkits

Research activities mainly supported by:
- IMT Atlantique
- Inria
- French ANR SONGS Project
- Hemera Inria Large Scale Initiative
- Discovery Inria Project Lab
- EU BigStorage Project

Cluster-Wide context switch
Distributed VM Scheduler
VMPlaceS
SimGridVM: VM abstractions
Locality Aware Placement

T. Hirofuchi
Postdoc/Invited researcher

A. Simonet
Postdoc
Energy dimension

T. L. Nguyen
PhD
Boot time model

IMT Atlantique
Hemera Inria Large Scale Initiative
Discovery Inria Project Lab
EU BigStorage Project
French ANR SONGS Project
Inria
Toward a VM PLACEment Simulator

• A dedicated simulator to
  • Evaluate/compare VM placement policies at large-scale (and in reproducible manner)
  • Relieve researchers of the burden of dealing with VM creations and workloads generation/injection

• SimGrid as a base
  • A scientific instrument to study the behaviour of large-scale distributed systems
  • Design abstractions and models to enable researchers to control VMs in the same manner as in the real world (e.g., create/destroy, start/shutdown, suspend/resume and migrate)

Focus on the migration model
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM

- The more your VM is memory intensive, the longer the migration will be

Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)

1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
2. Transfer updated memory pages during the previous step
3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
4. Stop the VM. Transfer the rest of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM
- The more your VM is memory intensive, the longer the migration will be
- Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)
  1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
  2. Transfer updated memory pages during the previous step
  3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
  4. Stop the VM. Transfer the rest of of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM
- The more your VM is memory intensive, the longer the migration will be

Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)

1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
2. Transfer updated memory pages during the previous step
3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
4. Stop the VM. Transfer the rest of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM

- The more your VM is memory intensive, the longer the migration will be

- Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)
  1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
  2. Transfer updated memory pages during the previous step
  3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
  4. Stop the VM. Transfer the rest of of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM

- The more your VM is memory intensive, the longer the migration will be

- Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)
  1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
  2. Transfer updated memory pages during the previous step
  3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
  4. Stop the VM. Transfer the rest of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM
- The more your VM is memory intensive, the longer the migration will be

- Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)
  1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
  2. Transfer updated memory pages during the previous step
  3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
  4. Stop the VM. Transfer the rest of memory pages and states
Accurate Live Migration Model

- Migration time is not a linear function according to the size of the VM
- The more your VM is memory intensive, the longer the migration will be

- Transfer VM’s states to destination without perfectible shutdown (pre-copy algorithm)
  1. Transfer all memory pages of the VM (but, keep in mind the VM is still running at source)
  2. Transfer updated memory pages during the previous step
  3. Iterate this step until the rest of memory pages becomes sufficiently small to meet an acceptable downtime (30ms in KVM).
  4. Stop the VM. Transfer the rest of of memory pages and states
Accurate Live Migration Model

- Application memory footprints can be considered as linear functions

The time and the resulting traffic of a migration should be computed by taking into account competition arising in the presence of resource sharing and the memory refresh rate.

[CloudCom 2013]

- First accurate live migration model (implementing the pre-copy strategy)
SimGrid VM

- **SimGrid VM** allows users to launch hundreds of thousands of VMs on their simulation programs and control VMs in the same manner as in the real world.

- Users can execute computation and communication tasks on physical machines (PMs) and VMs through the same SimGrid API, which will provide a seamless migration path to IaaS simulations for hundreds of SimGrid users.

![Diagram showing the comparison between a physical machine and a virtual machine with tasks and equations](image)

1. Solve all the constraint problems at once.
   
   Eq1: \(X_1 + X_2 < C\)

2. Solve the constraint problems at the virtual machine layer.
   
   Eq2: \(X_{1,1} + X_{1,2} < X_1\)
   
   Eq3: \(X_{2,1} < X_2\)

All extensions have been integrated into SimGrid
VMPlaceS

- A three steps engine to evaluate VM Placement Strategies

**Initialization Phase**

Input: infrastructure topology, VM Nb, Workloads

**Injector/Scheduling Phase**

Injector

Scheduler

Injects events (CPU variations, node crashes…)

**Analysis Phase**

Researchers should (only) develop their scheduling algorithm in JAVA (or SCALA) using the SimGrid MSG API and a more abstract interface provided by VMPlaceS

Output: a JSON trace file which is then consumed by the statistics R system to deliver tables/graphs (VMPlaces records several metrics during the simulation execution)
**VMPlaceS: A First Use-Case**

- **To illustrate how different strategies can be evaluated/compared**

  ![Diagram](image.png)

  - Centralized Entropy [VEE’09]
  - Hierarchical Snooze [CCGRID’12]
  - Distributed DVMS [CCPE’12]

**Simulation Input parameters**

- PMs: 8 cores, 32GB, 1Gbps, 7 cores are considered.
- VMs: 1 core, 1GB, 1Gbps, memory footprint varies between 0 and 80%
- VM CPU load \((\mu=60, \sigma=20)\)
- 10 VMs per PM, Cluster infrastructure composed of 128/256/512/1024 PMs
- Duration: 1800 seconds
- Period of scheduling invocations: 30 seconds.
Entropy/Snooze/DVMS Analysis

The centralized strategy looks useless?

Cumulated violation time
Entropy/Snooze/DVMS Analysis

Another view focusing on Entropy and DVMS
## Entropy/Snooze/DVMS Analysis

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of violations (AVG ± STD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>256 nodes</td>
<td>40.09 ± 24.15</td>
<td>21.45 ± 12.10</td>
</tr>
<tr>
<td>512 nodes</td>
<td>55.63 ± 42.26</td>
<td>24.54 ± 16.95</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>81.57 ± 86.59</td>
<td>29.01 ± 38.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of computations (AVG ± STD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>128 nodes</td>
<td>3.76 ± 7.43</td>
<td>2.52 ± 4.63</td>
</tr>
<tr>
<td>256 nodes</td>
<td>7.97 ± 15.03</td>
<td>2.65 ± 4.69</td>
</tr>
<tr>
<td>512 nodes</td>
<td>15.71 ± 29.14</td>
<td>2.83 ± 4.98</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>26.41 ± 50.35</td>
<td>2.69 ± 4.92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of reconfigurations (AVG ± STD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>128 nodes</td>
<td>10.34 ± 1.70</td>
<td>10.02 ± 0.14</td>
</tr>
<tr>
<td>256 nodes</td>
<td>10.26 ± 1.45</td>
<td>10.11 ± 0.83</td>
</tr>
<tr>
<td>512 nodes</td>
<td>11.11 ± 3.23</td>
<td>10.28 ± 1.50</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>18.90 ± 7.57</td>
<td>10.30 ± 1.60</td>
</tr>
</tbody>
</table>
## Entropy/Snooze/DVMS Analysis

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of violations (AVG ± STD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>256 nodes</td>
<td>40.09 ± 24.15</td>
<td>21.45 ± 12.10</td>
</tr>
<tr>
<td>512 nodes</td>
<td>55.63 ± 42.26</td>
<td>24.54 ± 16.95</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>81.57 ± 86.59</td>
<td>29.01 ± 38.14</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of computations (AVG ± STD)</th>
<th>Directional</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
<td>Distributed</td>
</tr>
<tr>
<td>128 nodes</td>
<td>3.76 ± 7.43</td>
<td>2.52 ± 4.63</td>
<td>0.29 ± 0.03</td>
</tr>
<tr>
<td>256 nodes</td>
<td>7.97 ± 15.03</td>
<td>2.65 ± 4.69</td>
<td>0.25 ± 0.02</td>
</tr>
<tr>
<td>512 nodes</td>
<td>15.71 ± 29.14</td>
<td>2.83 ± 4.98</td>
<td>0.21 ± 0.01</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>26.41 ± 50.35</td>
<td>2.69 ± 4.92</td>
<td>0.14 ± 0.01</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Duration of reconfigurations (AVG ± STD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Centralized</td>
<td>Hierarchical</td>
</tr>
<tr>
<td>128 nodes</td>
<td>10.34 ± 1.70</td>
<td>10.02 ± 0.14</td>
</tr>
<tr>
<td>256 nodes</td>
<td>10.26 ± 1.45</td>
<td>10.11 ± 0.83</td>
</tr>
<tr>
<td>512 nodes</td>
<td>11.11 ± 3.23</td>
<td>10.28 ± 1.50</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>18.90 ± 7.57</td>
<td>10.30 ± 1.60</td>
</tr>
</tbody>
</table>

**DVMS outperforms the others!**
While the centralized approach does not scale, both phases are constant from the time viewpoint for the two other approaches.

### Duration of Violations (AVG | STD)

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Centralized</th>
<th>Hierarchical</th>
<th>DVMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 nodes</td>
<td>40.09 ± 24.15</td>
<td>21.45 ± 12.10</td>
<td>9.58 ± 2.51</td>
</tr>
<tr>
<td>512 nodes</td>
<td>55.63 ± 42.26</td>
<td>24.54 ± 16.95</td>
<td>9.57 ± 2.67</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>81.57 ± 86.59</td>
<td>29.01 ± 38.14</td>
<td>9.61 ± 2.54</td>
</tr>
</tbody>
</table>

### Duration of Computations (AVG | STD)

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Centralized</th>
<th>Hierarchical</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 nodes</td>
<td>3.76 ± 7.43</td>
<td>2.52 ± 4.63</td>
<td>0.29 ± 0.03</td>
</tr>
<tr>
<td>256 nodes</td>
<td>7.97 ± 15.03</td>
<td>2.65 ± 4.69</td>
<td>0.25 ± 0.02</td>
</tr>
<tr>
<td>512 nodes</td>
<td>15.71 ± 29.14</td>
<td>2.83 ± 4.98</td>
<td>0.21 ± 0.01</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>26.41 ± 50.35</td>
<td>2.69 ± 4.92</td>
<td>0.14 ± 0.01</td>
</tr>
</tbody>
</table>

### Duration of Reconfigurations (AVG | STD)

<table>
<thead>
<tr>
<th>Infrastructure size</th>
<th>Centralized</th>
<th>Hierarchical</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 nodes</td>
<td>10.34 ± 1.70</td>
<td>10.02 ± 0.14</td>
<td>10.01 ± 0.11</td>
</tr>
<tr>
<td>256 nodes</td>
<td>10.26 ± 1.45</td>
<td>10.11 ± 0.83</td>
<td>10.01 ± 0.08</td>
</tr>
<tr>
<td>512 nodes</td>
<td>11.11 ± 3.23</td>
<td>10.28 ± 1.50</td>
<td>10.08 ± 0.82</td>
</tr>
<tr>
<td>1024 nodes</td>
<td>18.90 ± 7.57</td>
<td>10.30 ± 1.60</td>
<td>10.04 ± 0.63</td>
</tr>
</tbody>
</table>

**DVMS outperforms the others!**
While the centralized approach does not scale, both phases are constant from the time viewpoint for the two other approaches.

1. Can we find a good partitioning size for Snooze?
2. What would be the benefit for Snooze of a reactive approach?
Investigate Variants

- Evaluate the impact of having smaller partitions in Snooze
- Same numbers of PMs but partitions grow from 2 LCs to 32 LCs per GM
Investigate Variants

- Evaluate the impact of having smaller partitions in Snooze
- Same numbers of PMs but partitions grow from 2 LCs to 32 LCs per GM

![Graph showing cumulated violation time vs infrastructure size]

<table>
<thead>
<tr>
<th>Infra. Size</th>
<th>No. of failed reconfigurations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 LCs</td>
</tr>
<tr>
<td>128</td>
<td>19</td>
</tr>
<tr>
<td>256</td>
<td>29</td>
</tr>
<tr>
<td>512</td>
<td>83</td>
</tr>
<tr>
<td>1024</td>
<td>173</td>
</tr>
</tbody>
</table>

![Table showing duration of computations]

<table>
<thead>
<tr>
<th>Infra. Size</th>
<th>Duration of the computations (AVG ± STD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 LCs</td>
</tr>
<tr>
<td>128</td>
<td>0.16 ± 1.23</td>
</tr>
<tr>
<td>256</td>
<td>0.18 ± 1.31</td>
</tr>
<tr>
<td>512</td>
<td>0.15 ± 1.20</td>
</tr>
<tr>
<td>1024</td>
<td>0.19 ± 1.37</td>
</tr>
</tbody>
</table>
Investigate Variants

- Evaluate the impact of having smaller partitions in Snooze
  - Same numbers of PMs but partitions grow from 2 LCs to 32 LCs per GM
Investigate Variants

- Evaluate the impact of having smaller partitions in Snooze.
  - Same numbers of PMs but partitions grow from 2 LCs to 32 LCs per GM.

The smaller is the size of the partition, the bigger the probability to do not find a viable solution.

<table>
<thead>
<tr>
<th>Infra. Size</th>
<th>No. of failed reconfigurations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 LCs</td>
</tr>
<tr>
<td>128</td>
<td>19</td>
</tr>
<tr>
<td>256</td>
<td>29</td>
</tr>
<tr>
<td>512</td>
<td>83</td>
</tr>
<tr>
<td>1024</td>
<td>173</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Infra. Size</th>
<th>Duration of the computations (AVG ± STD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 LCs</td>
</tr>
<tr>
<td>128</td>
<td>0.16 ± 1.23</td>
</tr>
<tr>
<td>256</td>
<td>0.18 ± 1.31</td>
</tr>
<tr>
<td>512</td>
<td>0.15 ± 1.20</td>
</tr>
<tr>
<td>1024</td>
<td>0.19 ± 1.37</td>
</tr>
</tbody>
</table>
Investigate Variants

- Evaluate the impact of having smaller partitions in Snooze
- Same numbers of PMs but partitions grow from 2 LCs to 32 LCs per GM

The smaller is the size of the partition, the bigger the probability to do not find a viable solution

Other variants and possible improvements (for instance contact neighbours two by two in DVMS)
VMPlaceS / VM Simulator toolkits

• Difficulties to conduct relevant evaluation of VM placement strategies (in vivo conditions, lot of metrics to monitor, scalability/reactivity, …)

• VMPlaceS, a framework providing
  • Programming support for the definition of new VM placement strategies
  • Execution support for their accurate simulation at large scale
  • Means to analyze the collected traces
  • Validated up to 10K PMs/100K VMs
  • Available online: http://beyondtheclouds.github.io/VMPlaceS/

• On-going and future work
  • Collect energy metrics
  • VM boot time
  • VM image migrations (storage challenge)
  • Workloads reproducing real traces (complex to get real traces)
  • Provide similar abstractions for container technologies (must have)
Beyond the Clouds

Research activities mainly supported by

- IMT Atlantique
- Inria
- French ANR SONGS Project
- Hemera Inria Large Scale Initiative
- Discovery Inria Project Lab
- EU BigStorage Project

Research Eng.
R-A Cherrueau
M. Simonin
EnOS

Discovery vision
OpenStack: From SQL to noSQL backends

Discovery Inria Project Lab

VMPlaceS
SimGridVM
Locality Aware Placement
Distributed VM Scheduler
Cluster-Wide context switch

UTILITY COMPUTING

From mainframes to ...
UTILITY COMPUTING

From mainframes to ...

...larger “mainframes”
Jurisdiction concerns
Reliability
CC distance (network overheads)

2012 - 2013
Major brakes for the adoption of the CC model
Discovery Vision

- **Bring Clouds back to the cloud**
  - Leverage the concept of μDC/nDC to extend any point of presence of network backbones (aka PoP) with servers
  - From network hubs up to major DSLAMs that are operated by telecom companies, network institutions…

[Image: Map of a network infrastructure, showing connections between RENATER, Geant, and Internet 2.]
Discovery Vision

• **Bring Clouds back to the cloud**
  
  • Leverage the concept of μDC/nDC to extend any point of presence of network backbones (aka PoP) with servers
  
  • From network hubs up to major DSLAMs that are operated by telecom companies, network institutions…

How operating/using such a massively distributed infrastructure from the software viewpoint?
What’s about Brokering Approaches?

• Sporadic (hybrid computing/cloud bursting) almost ready for production

• Brokers are rather limited to simple usages and not advanced administration operations
What’s about Brokering Approaches?

- Sporadic (hybrid computing/cloud bursting) almost ready for production
- Brokers are rather limited to simple usages and not advanced administration operations
What’s about Brokering Approaches?

- Sporadic (hybrid computing/cloud bursting) almost ready for production
- Brokers are rather limited to simple usages and not advanced administration operations

Advanced brokers must reimplement standard IaaS mechanisms while facing the API limitation.
Would OpenStack be the solution?

• Do not reinvent the wheel… it’s too late OpenStack (20Millions of LOC, 3M just for the core-services)

• **Discovery objectives** (overview)
  
  • Study to what extent the current OpenStack mechanisms can handle such massively distributed infrastructures
  
  • Propose revisions/extensions of internal mechanisms when appropriate

• **From SQL to NoSQL backend in OpenStack**
  (a research PoC, just the top of the iceberg, numerous challenges)

• **Toward a Holistic Framework for Conducting Scientific Evaluations of OpenStack**
  EnOS, A tool for diving into OpenStack and performing scientific investigations

[IC2E 2017]
[CCGRID 2017]
Conclusion / Future Work

• Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…
  • Complexity of the software stack
  • Difficulty to guarantee performance
Conclusion / Future Work

- Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…
  - Complexity of the software stack
  - Difficulty to guarantee performance

What you may expect!
Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…

- Complexity of the software stack
- Difficulty to guarantee performance

What you may have!
Conclusion / Future Work

• Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…
  • Complexity of the software stack
  • Difficulty to guarantee performance

What you may have!
Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…

- Complexity of the software stack
- Difficulty to guarantee performance

What you may have!
Conclusion / Future Work

• Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…
  • Complexity of the software stack
  • Difficulty to guarantee performance

• Placement challenges
  • How to express placement constraints? [plasma2013] a good starting point.
  • Can we consider network and storage dimensions?

• People expect containers technologies will help but..
  • Similar consolidation issues
  • Naive use (containers on top of VM on top of PM)

• Current trend: server densification (more cores per PMs, more RAMs….)


Conclusion / Future Work

- Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost...
  - Complexity of the software stack
  - Difficulty to guarantee performance

- Placement challenges
  - How to express placement constraints? [plasma2013] a good starting point.
  - Can we consider network and storage dimensions?

- People expect containers technologies will help but..
  - Similar consolidation issues
  - Naive use (containers on top of VM on top of PM)

- Current trend: server densification (more cores per PMs, more RAM)

Propose tools that help us to understand this complexity
Conclusion / Future Work

• Virtualization technologies: a key role in the Cloud Computing adoption (flexibility, portability) but with a cost…
  • Complexity of the software stack
  • Difficulty to guarantee performance

• Placement challenges
  • How to express placement constraints? [plasma2013] a good starting point.
  • Can we consider network and storage dimensions?

• People expect containers technologies will help but..
  • Similar consolidation issues
  • Naive use (containers on top of VM on top of PM)

• Current trend: server densification (more cores per PMs, more RAM)

Propose models to capture this complexity in advanced alg.
Propose tools that help to understand this complexity
Conclusion / Future Work

• Utility Computing: a constant switch between centralization and distribution (Mainframe/Cluster vs Grid vs Cloud)

• A new decentralization phase (no more debated) due to locality requirements of IoT and NFV Applications.

Industrial Internet / Internet of Skills
Conclusion / Future Work

• Utility Computing: a constant switch between centralization and distribution (Mainframe/Cluster vs Grid vs Cloud)

• A new decentralization phase (no more debated) due to locality requirements of IoT and NFV Applications.

Industrial Internet / Internet of Skills
Conclusion / Future Work

• Utility Computing: a constant switch between centralization and distribution (Mainframe/Cluster vs Grid vs Cloud)

• A new decentralization phase (no more debated) due to locality requirements of IoT and NFV Applications.
Conclusion / Future Work

- **Utility Computing**: a constant switch between centralization and distribution (Mainframe/Cluster vs Grid vs Cloud)

- A new decentralization phase (no more debated) due to locality requirements of IoT and NFV Applications.

- Are the challenges similar to the Grid ones?
  - Topology (static vs dynamic)
  - Federation of a few sites vs massively distributed
  - Heterogeneity in terms of ICT resources

- Some research groups work on federated clouds (in particular for science) but **edge/fog computing infrastructure significantly differ**

**Fog/Edge - What are the challenges?**
Weathermap métropole
(IPv4, IPv6, VPN)
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?
Where should I deploy micro DCs? On each PoP? What’s about control services? at what scale? What’s about the performance/reliability criteria?
Where should I deploy micro DCs? On each PoP?
What’s about control services?
at what scale?
What’s about the performance/reliability criteria?
How control services should be designed?
Centralised / Hierarchical / P2P based?
Where should I deploy micro DCs? On each PoP?
What's about control services?
at what scale?
What's about the performance/reliability criteria?
How control services should be designed?
Centralised / Hierarchical / P2P based?
Global vs partial views of the system?
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?
What’s about the performance/reliability criteria?
How control services should be designed? Centralised / Hierarchical / P2P based?
Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions (infinity of resources, data locality).
Here resources are bounded, applications have more constraints to deal with...
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?
What’s about the performance/reliability criteria?
How control services should be designed? Centralised / Hierarchical / P2P based?
Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions (infinity of resources, data locality).
Here resources are bounded, applications have more constraints to deal with…

How can developers express those constraints?
How can the system guarantee them during reconfiguration operations?
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?
What’s about the performance/reliability criteria?

How control services should be designed? Centralised / Hierarchical / P2P based?
Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions (infinity of resources, data locality).
Here resources are bounded, applications have more constraints to deal with…

How can developers express those constraints?
How can the system guarantee them during reconfiguration operations?

How should the system address big data applications (considering a significant number of geo-distributed data sources)?
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?
What’s about the performance/reliability criteria?

How control services should be designed? Centralised / Hierarchical / P2P based?
Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions (infinity of resources, data locality).
Here resources are bounded, applications have more constraints to deal with…

How can developers express those constraints?
How can the system guarantee them during reconfiguration operations?

How should the system address big data applications (considering a significant number of geo-distributed data sources)?

What’s about security aspects?
Where should I deploy micro DCs? On each PoP?
What’s about control services? at what scale?

What’s about the performance/reliability criteria?

How control services should be designed? Centralised / Hierarchical / P2P based?

Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions (infinity of resources, data locality).
Here resources are bounded, applications have more constraints to deal with…

How can developers express those constraints?
How can the system guarantee them during reconfiguration operations?

How should the system address big data applications (considering a significant number of geo-distributed data sources)?

Energy footprint of such infrastructures?

what’s about security aspects?
Where should I deploy micro DCs? On each PoP?

What’s about control services?

at what scale?

What’s about the performance/reliability criteria?

How control services should be designed?

Centralised / Hierarchical / P2P based?

Global vs partial views of the system?

Placement algorithms have been designed with strong assumptions

(infinity of resources, data locality).

Here resources are bounded, applications have more constraints to deal with…

How can developers express those constraints?

How can the system guarantee them during reconfiguration operations?

How should the system address big data applications

(considering a significant number of geo-distributed data sources)?

Energy footprint of such infrastructures?

Can μDCs benefit from renewable energy sources?
Where should I deploy micro DCs? On each PoP?

What’s about control services?

at what scale?

What’s about the performance/reliability criteria?

How control services should be designed?

Centralised / Hierarchical / P2P based?

Global vs partial views of the system?

How can developers express those constraints?

How can the system guarantee them during reconfiguration operations?

How should the system address big data applications (considering a significant number of geo-distributed data sources)?

STACK, a new research team to address Fog/Edge Infrastructures’ challenges

Energy footprint of such infrastructures?

Can μDCs benefit from renewable energy sources?
It does not matter how slowly you go as long as you do not stop.

— Confucius
Back up
A lot of challenges

• A huge gap between open source software stacks to run production systems and academic proposals

• Do not let Google/Amazon be the only actors to address infrastructure challenges just because they operate them.

• The scientific community (both network and distributed system ones) should take part to the evolution of major software stacks such as OpenStack like it has been once done for Linux.

• We need dedicated infrastructures to conduct such scientific studies.
A lot of challenges

• A huge gap between open source software stacks to run production systems and academic proposals

“Just good enough to publish a good paper”
Francesco Lo Presti, Ass. Prof. University Di Roma, Italy

“…Optimality is not needed, it should just run…”
Johan Ecker, Principal Researcher, Cloud Technology, Ericsson Research, Sweden

CloudControl WS June 2017

• Do not let Google/Amazon be the only actors to address infrastructure challenges just because they operate them.

• The scientific community (both network and distributed system ones) should take part to the evolution of major software stacks such as OpenStack like it has been once done for Linux.

• We need dedicated infrastructures to conduct such scientific studies
VM Placement
Still a Hot Topic Problem
Accuracy of VMPlaceS

- Implementation of a dedicated version of VMPlaceS on Grid’5000
- Implementation of the Entropy proposal [VEE’09] in both systems
- Comparison between in vivo and simulated executions

32PMs, 4 cores/16GB/1Gbps per PM
192 VMs (6 per node)
1 core,1GB, 1Gbps
Memory footprint (between 0 and 80% of 1Gbps)
Average CPU load 60%

The scheduling algorithm has been invoked every 60 seconds over a 3600 seconds execution

A dedicated tool to inject the load in each VM running on top of G5K according to the events consumed by the injector
Accuracy of VMPlaceS

- Implementation of a dedicated version of VMPlaceS on Grid’5000
- Implementation of the Entropy proposal [VEE’09] in both systems
- Comparison between in vivo and simulated executions

32PMs, 4 cores/16GB/1Gbps per PM

192 VMs (6 per node)

1 core, 1GB, 1Gbps

Memory footprint (between 0 and 80% of 1Gbps)

Average CPU load 60%

The scheduling algorithm has been invoked every 60 seconds over a 3600 seconds execution.

A dedicated tool to inject the load in each VM running on top of G5K according to the events consumed by the injector.

Difference between 6% and 18% (med: 12%)

Worst case when multiple migrations are performed simultaneously to the same node.
Understanding DVMS

Partition

Understanding DVMS
Understanding DVMS

Partition

Understanding DVMS

Partition

Understanding DVMS
Understanding DVMS

Partition

Understanding DVMS

Partition
Understanding DVMS

Partition

Locality-aware Cooperation

Partition

A Ring to Rule them All

From SQL to NoSQL backend in OpenStack
Looking back to the Future

- Austin Summit - May 2016 - Nova PoC

Technical considerations

**Technical considerations**

- Infrastructure segregation
- Host aggregates
- Availability zones
- Segregation example

Repurposing an existing OpenStack environment to be massively scalable is a formidable task. When building a massively scalable environment from the ground up, ensure you build the initial deployment with the same principles and choices that apply as the environment grows. For example, a good approach is to deploy the first site as a multi-site environment. This enables you to use the same deployment and segregation methods as the environment grows to separate locations across dedicated links or wide area networks. In a hyperscale cloud, scale trumps redundancy. Modify applications with this in mind, relying on the scale and homogeneity of the environment to provide reliability rather than redundant infrastructure provided by non-commodity hardware solutions.

**Infrastructure segregation**

OpenStack services support massive horizontal scale. Be aware that this is not the case for the entire supporting infrastructure. This is particularly a problem for the database management systems and message queues that OpenStack services use for data storage and remote procedure call communications.

Traditional clustering techniques typically provide high availability and some additional scale for these environments. In the quest for massive scale, however, you must take additional steps to relieve the performance pressure on these components in order to prevent them from negatively impacting the overall performance of the environment. Ensure that all the components are in balance so that if the massively scalable environment fails, all the components are near maximum capacity and a single component is not causing the failure.
Looking back to the Future

Technical considerations

Infrastructure segregation
Host aggregates
Availability zones
Segregation example

Repurposing an existing OpenStack environment to be massively scalable is a formidable task. When building a massively scalable environment from the ground up, ensure you build the initial deployment with the same principles and choices that apply as the environment grows. For example, a good approach is to deploy the first site as a multi-site environment. This enables you to use the same deployment and segregation methods as the environment grows to separate locations across dedicated links or wide area networks. In a hyperscale cloud, scale trumps redundancy. Modify applications with this in mind, relying on the scale and homogeneity of the environment to provide reliability rather than redundant infrastructure provided by non-commodity hardware solutions.

Infrastructure segregation

OpenStack services support massive horizontal scale. Be aware that this is not the case for the entire supporting infrastructure. This is particularly a problem for the database management systems and message queues that OpenStack services use for data storage and remote procedure call communications.

Traditional clustering techniques typically provide high availability and some additional scale for these environments. In the quest for massive scale, however, you must take additional steps to relieve the performance pressure on these components in order to prevent them from negatively impacting the overall performance of the environment. Ensure that all the components are in balance so that if the massively scalable environment fails, all the components are near maximum capacity and a single component is not causing the failure.
Leveraging a Key/Value Store DB

Nova (compute service) - software architecture
ROME

- Relational Object Mapping Extension for key/value stores
  Jonathan Pastor’s Phd
  https://github.com/BeyondTheClouds/rome

- Enables the query of Key/Value Store DB with the same interface as SQLAlchemy

- Enables Nova OpenStack to switch to a KVS without being too intrusive

- The KVS is distributed over (dedicated) nodes

- Nova services connect to the Key/value store cluster
Experiments

• Experiments have been conducted on Grid’5000
• Mono-site experiments ⟹ Evaluate the overhead of using ROME/Redis and the network impact.
• Multi-site experiments ⟹ Determine the impact of latency. ⟹ Validate compatibility with higher level mechanisms validation

www.grid5000.fr
1500 servers, spread across 10 sites
Full admin rights
Mono-Site Experiments

- Creation of 500 VMs
- Comparison MySQL/SQLAlchemy vs ROME/Redis (one dedicated node for the DB server/the REDIS server)

MySQL/SQLAlchemy

ROME/Redis
Mono-Site Experiments

- Evaluate the overhead of using ROME/Redis
- ROME stores objects in a JSON format: serialization/deserialization cost
- ROME reimplements some mechanisms: join, transaction/session, …

**ROME requests are faster for 80% of requests**

**SQLAlchemy is faster for 20% of requests**
Mono-Site Experiments

- Evaluate the overhead of using ROME/Redis
- ROME stores objects in a JSON format: serialization/deserialization cost
- ROME reimplements some mechanisms: join, transaction/session, …

Table 2: Time used to create 500 VMs on a single cluster configuration (in sec.)

<table>
<thead>
<tr>
<th>Backend configuration</th>
<th>REDIS</th>
<th>MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>322</td>
<td>298</td>
</tr>
<tr>
<td>4 nodes</td>
<td>327</td>
<td>-</td>
</tr>
<tr>
<td>4 nodes + repl</td>
<td>413</td>
<td>-</td>
</tr>
</tbody>
</table>
Multi-site Experiments

- Creation of 500 VMs, fairly distributed on each controller
- From 2 to 8 sites (emulation of virtual clusters by adding latency thanks to TC)
- Each cluster was containing 1 controller, 6 compute nodes (and 1 dedicated node in the case of REDIS).
- MySQL and Redis used in the default configuration
- To fairly compare with MySQL, data replication was not activated in Redis
- Galera experiments have been performed but due to reproducible issues with more than 4 sites, results are not satisfactory enough to be discussed (RR available on demand)
Multi-Site Experiments

Table 3: Time used to create 500 VMs with a 10ms inter-site latency (in sec.).

<table>
<thead>
<tr>
<th>Nb of locations</th>
<th>REDIS</th>
<th>MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 clusters</td>
<td>271</td>
<td>209</td>
</tr>
<tr>
<td>4 clusters</td>
<td>263</td>
<td>139</td>
</tr>
<tr>
<td>6 clusters</td>
<td>229</td>
<td>123</td>
</tr>
<tr>
<td>8 clusters</td>
<td>223</td>
<td>422</td>
</tr>
</tbody>
</table>

Table 4: Time used to create 500 VMs inter-site latency (in sec.).

<table>
<thead>
<tr>
<th>Nb of locations</th>
<th>REDIS</th>
<th>MySQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 clusters</td>
<td>723</td>
<td>268</td>
</tr>
<tr>
<td>4 clusters</td>
<td>427</td>
<td>203</td>
</tr>
<tr>
<td>6 clusters</td>
<td>341</td>
<td>184</td>
</tr>
<tr>
<td>8 clusters</td>
<td>302</td>
<td>759</td>
</tr>
</tbody>
</table>

Increasing the nb of nodes leads to better reactivity
From 8 clusters, MySQL becomes a bottleneck

SQL scalability bottleneck
(one SQL server for the whole infrastructure)
Compatibility with Higher Level Features

- Asses the usage of advanced OpenStack feature: *host-aggregates / availability zones*
- As we targeted a low-level component, ROME is compatible with most of the existing features.
- Performance is not impacted (same order of magnitude)
- VM Repartition is correctly achieved (without availability zones the distribution was respectively 26%, 20%, 22%, 32% of the created VMs for a 4 clusters experiments).

Can we go beyond a research POC?
Compatibility with Higher Level Features

- Asses the usage of advanced OpenStack feature: *host-aggregates / availability zones*

- As we targeted a low-level component, ROME is compatible with most of the existing features.

- Performance is not impacted (same order of magnitude)

- VM Repartition is correctly achieved (without availability zones: respectively 26%, 20%, 22%, 32% of the created VMs for a 4 clusters experiments)

---

Can we go beyond a research POC ?

Interesting but …just the top of the Iceberg !

- Glance, Neutron, Cinder…?
- Scalability of the AMQP bus?
- HA?

Reify locality aspects at every level of the stack?
STACK Proposal

ASCOLA Follow-up
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.

- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.

- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.

- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.
- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross-cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.

- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
Tomorrow STACK?

- Designing a tightly-coupled software stack to operate and use massively geo-distributed ICT infrastructures.

- Delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
STACK Proposal

App Mgmt - Programming Model/API, deployment and reconfiguration Engines (self*)

Resource Mgmt - Capacity Planning / Deployment and reconfiguration

Compute  Networking  Storage

Building block
STACK Proposal

Building blocks

security / energy

Compute

Storage

Networking

App Mgmt - Programming Model/API, deployment and reconfiguration Engines (self*)

Resource Mgmt - Capacity Planning / Deployment and reconfiguration

App Mgmt - Programming Model/API, deployment and reconfiguration Engines (self*)

Resource Mgmt - Capacity Planning / Deployment and reconfiguration
STACK Proposal

Scalability

Efficiency / Security / Reliability

Application management

Resource management

Building blocks

security / energy

Compute

Storage

Networking

App Mgmt - Programming Model/API, deployment and reconfiguration Engines (self*)

Resource Mgmt - Capacity Planning / Deployment and reconfiguration

Building block

Scalability

Efficiency / Security / Reliability

Application management

Resource management

Building blocks

security / energy

Compute

Storage

Networking

App Mgmt - Programming Model/API, deployment and reconfiguration Engines (self*)

Resource Mgmt - Capacity Planning / Deployment and reconfiguration

Building block
STACK Proposal

Scalability

Efficiency / Security / Reliability

Synergy

Application management

Resource management

Building blocks

“Just” a distributed resource management system?
Energy Dimension

- From sustainable data centers to a new source of energy
  A promising way to deliver highly efficient and sustainable UC services is to provide UC platforms as close as possible to the end-users and to...

- Leverage “green” energy (solar, wind turbines...)
  Transfer the green micro/nano DCs concept to the network PoP
  Take the advantage of the geographical distribution

- Leveraging the data furnaces concept
  Deploy UC servers in medium and large institutions and use them as sources of heat inside public buildings such as hospitals or universities
Challenges & Foundations

• Challenges

Identify and revise core mechanisms/algorithms to enable scalability, distribution… while taking account fog/edge specifics. Extend API and software programming abstractions (high level) and identify missing mechanisms (low-level) to benefit from geo-distribution opportunities.

Tightly coupled : synergy between all mechanisms composing the system.

• Foundations

Distributed systems
Software programming (Component-based model, DSL, composition)
Self-* mechanisms
Performance evaluations (experiment driven research)
STACK Proposal

- STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures

<table>
<thead>
<tr>
<th>Name</th>
<th>Title/Affiliation</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helene Coullon</td>
<td>Ass. Prof. IMT Atlantique (Inria Chair)</td>
<td>Domain Specific Languages, Component models, Distributed and High Performance Computing</td>
</tr>
<tr>
<td>Shadi Ibrahim</td>
<td>CR1 Inria</td>
<td>Big Data management, Virtualization, File and Storage Systems, High-Performance Computing</td>
</tr>
<tr>
<td>Adrien Lebre</td>
<td>Researcher Inria (on leave from IMT Atlantique)</td>
<td>Distributed Computing, Virtualization, Storage, Experiment-driven research</td>
</tr>
<tr>
<td>Thomas Ledoux</td>
<td>Ass. Prof. IMT Atlantique</td>
<td>Autonomic computing, Middleware, Software Engineering, Energy management</td>
</tr>
<tr>
<td>Jean Marc Menaud</td>
<td>Prof. IMT Atlantique</td>
<td>Virtualization, Energy, Data centers</td>
</tr>
<tr>
<td>Jacques Noyé</td>
<td>Ass. Prof. IMT Atlantique</td>
<td>Programming Languages, Software Composition, Concurrent and Distributed Computing</td>
</tr>
<tr>
<td>Mario Südholt</td>
<td>Prof. IMT Atlantique</td>
<td>Software Programming, Security, Distributed Algorithms.</td>
</tr>
</tbody>
</table>
STACK Proposal

- STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Role/Affiliation</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helene Coulon</td>
<td>Ass. Prof. IMT Atlantique (Inria Chair)</td>
<td>Domain Specific Languages, Component models, Distributed and High Performance Computing</td>
</tr>
<tr>
<td>Shadi Ibrahim</td>
<td>CR1 Inria</td>
<td>Big Data management, Virtualization, File and Storage Systems, High-Performance Computing</td>
</tr>
<tr>
<td>Adrien Lebre</td>
<td>Researcher Inria (on leave from IMT Atlantique)</td>
<td>Distributed Computing, Virtualization, Storage, Experiment-driven research</td>
</tr>
<tr>
<td>Thomas Ledoux</td>
<td>Ass. Prof. IMT Atlantique</td>
<td>Autonomic computing, Middleware, Software Engineering, Energy management</td>
</tr>
<tr>
<td>Jean Marc Menaud</td>
<td>Prof. IMT Atlantique</td>
<td>Virtualization, Energy, Data centers, Programming Languages, Software Composition, Concurrent and Distributed Computing</td>
</tr>
<tr>
<td>Mario Südholt</td>
<td>Prof. IMT Atlantique</td>
<td></td>
</tr>
</tbody>
</table>
STACK Proposal

• STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
STACK Proposal

- STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
STACK Proposal

STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
STACK Proposal

- STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helene Coullon</td>
<td>Domain Specific Languages, Component models, Distributed and High Performance Computing</td>
</tr>
<tr>
<td>Shadi Ibrahim</td>
<td>Big Data management, Virtualization, File and Storage Systems, High-Performance Computing</td>
</tr>
<tr>
<td>Adrien Lebre</td>
<td>Distributed Computing, Virtualization, Storage, Experiment-driven research</td>
</tr>
<tr>
<td>Thomas Ledoux</td>
<td>Autonomic computing, Middleware, Software Engineering, Energy management</td>
</tr>
<tr>
<td>Jean Marc Menaud</td>
<td>Virtualization, Energy, Data centers</td>
</tr>
<tr>
<td>Jacques Noyé</td>
<td>Programming Languages, Software Composition, Concurrent and Distributed Computing</td>
</tr>
<tr>
<td>Mario Südholt</td>
<td>Software Programming, Security, Distributed Algorithms.</td>
</tr>
</tbody>
</table>

- Deployment / Reconfiguration
- Storage / Big Data
- Large-scale platforms management
- Application life-cycle management
- Energy
STACK Proposal

• STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helene Coullon</td>
<td>Domain Specific Languages, Component models, Distributed</td>
</tr>
<tr>
<td></td>
<td>and High Performance Computing</td>
</tr>
<tr>
<td>Shadi Ibrahim</td>
<td>Big Data management, Virtualization, File and Storage Systems,</td>
</tr>
<tr>
<td></td>
<td>High-Performance Computing</td>
</tr>
<tr>
<td>Adrien Lebre</td>
<td>Distributed Computing, Virtualization, Storage, Experiment-driven research</td>
</tr>
<tr>
<td>Thomas Ledoux</td>
<td>Autonomic computing, Middleware, Software Engineering,</td>
</tr>
<tr>
<td></td>
<td>Energy management</td>
</tr>
<tr>
<td>Jean Marc Menaud</td>
<td>Virtualization, Energy, Data centers</td>
</tr>
<tr>
<td>Jacques Noyé</td>
<td>Programming Languages, Software Composition, Concurrent</td>
</tr>
<tr>
<td></td>
<td>and Distributed Computing</td>
</tr>
<tr>
<td>Mario Südholt</td>
<td>Software Programming models</td>
</tr>
</tbody>
</table>

- Deployment / Reconfiguration
- Storage / Big Data
- Large-scale platforms management
- Application life-cycle management
- Energy
- Software programming models
STACK Proposal

- STACK is a new research group focusing on challenges related to the management and advanced usages of Utility Computing infrastructures (i.e., Cloud, Fog, Edge, and beyond).

The team is interested by delivering appropriate system abstractions, from low (system) to high-levels (applications), and by addressing cross cutting dimensions such as energy or security, to operate massively geo-distributed infrastructures.
STACK - Positioning

- Inria

  - Several research groups address challenges related large-scale infrastructures (ASAP, AVALON, DATAMOVE, KERDATA…)

  - Stack members are used to collaborate/work with

    - AVALON, MYRIADS, KERDATA, MADYNES, REGAL, CTRL-A (IPL Discovery, SCALUS/BigStorage EU MCITN, EPOC CominLabs, ANR MyCloud, I/O Lab…)

- Nantes

  Historical links with TASC/AtlanModels
  CRE Orange Entreprise du future (Helene Coullon)
  Complementarities/Collaborations opportunities with GDD, STR, RESTO/RIO (co-supervision of a PhD on Fog/Edge storage backends)
STACK - Positioning

• France
  • ERODS-LIG (Distributed systems), SEPIA-IRIT (Energy)

• International
  • Pr. Erik Elmroth, Umea University, (Control, Autonomous mechanisms)
  • Prof. Mira Mezini, TU Darmstad, Germany (Software Defined XXX).
  • Ass. Pr. Paolo Bellavista, University of Bologna (Mobile Computing)
  • Pr. Manish Parashar, Rutgers University (Fog/Edge + Energy)
    Pr. Weisong Shi, Wayne State University. (Edge + Mobile Computing)
  • Pr. Hai Jin, Huazhong University (I/O + Virtualization)
Looking Back…

- Resource management of virtualised distributed systems
  - VM Placements
    - Entropy (Energy) [VEE09]
    - DVMS (Scalability) [FGCS12]
    - VMPlaceS (SimGrid) [TCC15, Europar16]
  - Synergy between applications and the resource manager
    - Autonomic models for Cloud Computing applications [IGI12, Closer17]
    - DSLs (Quality of Service, Elasticity) [FGCS16, SAC17]
    - Proportional Energy (use of renewable energy) [Computing17]
  - Security
    - Compose security/privacy mechanisms [CloudCom13, RATSP15]
  - Distributed Clouds
    - Inria Project Lab DISCOVERY (2015-2019)
Projects

Ongoing
- IPL Discovery (2015-2019)
- EU BigStorage (2015-2018)
- CPER SeDuCE (2016-2020)
- CominLabs PrivGen (2016-2019)
- ANR GRECO (2017-2020)
- FSN/PIA Hydda HPC/Cloud between distinct sites (2017-2020).

Proposal
- SILECS (TGIR/ESFRI, national/EU consortium)

Animation

- GDR RSD: co-chair of the Virtualisation action (CloudDays, ResCom17)
- IEEE ICFEC conference
- Chairs of Fog/Edge related tracks in EuroPar’16, CloudCom’16/’17
- Chair of the Massively Distributed WG - OpenStack.