# Emulating Geo-Replication on Grid5000

Dastagiri Reddy Malikireddy, Masoud Saeida Ardekani, Marc Shapiro


# Emulating Geo-Replication on Grid'5000

**Dastagiri Reddy Malikireddy** INRIA
**Masoud Saeida Ardekani** INRIA & UPMC-LIP6
**Marc Shapiro** INRIA & UPMC-LIP6

**TECHNICAL REPORT**
**N° 455**
August 2014
Project-Team Regal

ISSN 0249-0803 ISRN INRIA/RT--455--FR+ENG


# Emulating Geo-Replication on Grid'5000*

Dastagiri Reddy Malikireddy INRIA
Masoud Saeida Ardekani INRIA & UPMC-LIP6
Marc Shapiro INRIA & UPMC-LIP6

Project-Team Regal

Technical Report n° 455 — August 2014 — 12 pages

**Abstract:** In the field of Distributed systems, many experiments require cloud infrastructure. However, having a datacenter like architecture is not always possible. Therefore we use Distem to emulate geo-replication on Grid'5000 that allows experimentation similar to that on the cloud. This further allows us to simulate latencies of 100-200 milliseconds as opposed to latencies of 10-20 milliseconds observed on Grid'5000. We discuss the challenges faced in emulating Geo-Replication and thereby study the latency-throughput curves on the emulated platform for different transactional protocols.

**Key-words:** Grid'5000, large-scale experimentation,, cloud computing

* The work presented in this report has been funded in part by ANR projects Prose (ANR-09-VERS-007-02) and Concordant (ANR-10-BLAN 0208).

RESEARCH CENTRE
PARIS – ROCQUENCOURT

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

# Emulating Geo-Replication on Grid'5000

**Résumé :** Une infrastructure de type « Cloud » est souvent requise pour l'expérimentation dans le domaine des systèmes répartis. Cependant, on n'a pas toujours à disposition une architecture de type « centre de calcul ». C'est pourquoi, nous utilisons Distem pour émuler un environnement géo-répliqué sur Grid'5000, recréant ainsi des conditions expérimentales similaires au nuage. Cela permet aussi d'émuler des latences de l'ordre de 100 ou 200 ms, et non les 10–20 ms observées sur Grid'5000. Ainsi nous avons pu rendre GDUR, un intergiciel développé sur Grid'5000, compatible avec un environnement de type nuage. Ce papier discute des défis de l'émulation de la géo-réplication, et étudie les courbes latence-débit sur la plate-forme d'émulation pour les différents protocoles de GDUR.

**Mots-clés :** Grid'5000, expériences à grande échelle, informatique en nuage

# 1 Introduction

Grid'5000 is a scientific instrument for the study of large scale parallel and distributed systems. It aims at providing a highly reconfigurable, controllable and monitorable experimental platform to its users. The infrastructure of Grid'5000 is geographically distributed on 11 different sites in France hosting the instrument. The average latencies between these sites is 10-20 milliseconds. Any experimental framework is thus limited by the order of latencies observed on Grid'5000. There is thus a need to simulate latencies of a higher order which are observed in the cloud in order to provide for more realtime experimentation on Grid'5000.

Although experimentation on the cloud is often desirable, it has some drawbacks:

1. Cost: performing scientifically sound experiments on cloud infrastructures require multiple executions with different workloads and machines. Therefore, experimenting over the cloud may impose substantial cost to an evaluation campaign.

2. Reproducibility: due to certain volatile aspects of cloud computing (e.g., communication latency), it is very hard to reproduce the experiment results performed on the cloud.

In addition to the above issues, current cloud infrastructures are geographically distributed. In this environment, the latency among inter-datacenter nodes are up to two orders of magnitudes larger than the latency among intra-datacenter nodes. Therefore, performing reproducible experiments in geo-distributed environment is even harder since inter-datacenter latency can vary form 50 ms to more than 200 ms.

To side step these problems, we aim at emulating a geo-distributed cloud environment on France Grid'5000 testbed [5] and thereby performing experiments on an emulated infrastructure.

We also chose G-DUR [2] which perfectly matches our needs as an application for our emulated infrastructure. G-DUR is generic framework for performing apples-to-apples comparisons among transactional protocols. By running G-DUR in our emulated geo-distributed cloud, we are able to compare transactional protocols in different execution environments.

The rest of the report is structured as follows. Section 2 presents the overview of G-DUR - a generic DUR middleware. Then, Section 3 describes Distem, its features and the set of interfaces provided to the users. In Section 4, we present our design of emulating Geo-replication using Distem, the design choices and the challenges faced in doing so. Section 5 talks about the experiments performed on the emulated Geo-replicated setup. Finally, in Section 6, we conclude our findings and describe our plans for future work.

# 2 G-DUR Overview

Internet applications have conflicting requirements. On one hand, they should be highly parallel and distributed in order to be fast, responsive and available; on the other, application servers must remain synchronised, in order to maintain consistency. There, thus, exists a large number of distributed transaction models and protocols with different trade-offs: from very strongly synchronised to ones with lots of parallelism.

A large number of distributed transaction protocols have a common structure, called Deferred Update Replication(DUR) . DUR provides dependability by replicating data and performance by not re-executing transactions but only applying their updates. Protocols of the DUR family differ only in behaviors of few generic functions. G-DUR is generic DUR middleware along with a library of finely-optimized plug-in implementations of the required behaviors.

By mixing-and-matching the different plugins offered by G-DUR, it is relatively easy to obtain a high-performance implementation of a protocol. This capability is also leveraged in an extensive experimentation that we conduct in a geo-replicated environment.

# 3 Distem Overview

Distem is a distributed systems emulator. It is used to transform a homogeneous cluster into a platform for launching large scale experimentation with virtual machines linked through a complex network topology. Using Distem offloads users from tedious and repeatable tasks. It even allows greater control of the configurations pertaining to the experimental environment.

Different features provided by Distem made us to base our emulation setup on it. We discuss these features below.

## 3.1 Virtualized environment

Distem offers an easy way to launch multiple virtualized nodes on a physical node. It abstracts away from the location where the virtual nodes are actually deployed. It even provides for transferring images of virtual nodes on the fly. At a fine grained level, the mapping between the cores on virtual nodes and the nodes on physical machines can also be specified depending on the requirements. This can be used to better study the experimental results and fine tune experiments to suit the need. Launching and managing a virtual environment manually is tedious and an error-prone task. There are many

details that need to be taken care of and if overlooked, they may lead to wrong conclusions about the experimental results. Thus Distem eases this task by providing a set of interfaces to the users.

## 3.2 Reproducibility and User-friendliness

Distem is a user-friendly tool by ensuring the following properties: (i) it aids in defining the topology of the network by automatically configuring routing tables in the nodes to route packets properly; and (ii) it offers three different user interfaces with increasing level of complexity and number of features. Depending on the situation and experience, one interface may prove more appropriate than the others. These programmable interfaces also allows users to write experiments as Ruby programs instead of shell scripts.

In addition, being able to configure a network topology, and setting up an execution environment guarantees reproducibility of our experiments.

## 3.3 High level architecture

Distem uses non-virtualized physical nodes denoted as *Pnode* as a base for creating an experimental platform with virtualized nodes called *Vnode*. Each *Pnode* may contain multiple *Vnodes*.They are unaware of each other's presence. All *Vnodes* can also be linked through a complex network topology as per the experimental requirements.

Figure 1 shows the architecture of Distem. Every *Pnode* hosts its instance of Distem daemon -*distemd*-that is responsible for controlling the *Vnodes* hosted on it and other physical resources assigned to them. One of the nodes, called the *coordinator*, is special and acts as a frontend for controlling the whole experimental infrastructure. It acts as a communication bridge with all the other nodes.

The user provides a system image to boot up the virtualized nodes. The *Vnodes* can also share a system image.

## 3.4 Interfaces

Distem uses REST communication paradigm to provide users with a well defined stack of interfaces, each built on top of the previous one:

(i) *REST interface:* a well defined and structured schema to control the resources inside Distem.

(ii) *Ruby interface:* a programmatic way to work with Distem, by directly accessing the REST interface.

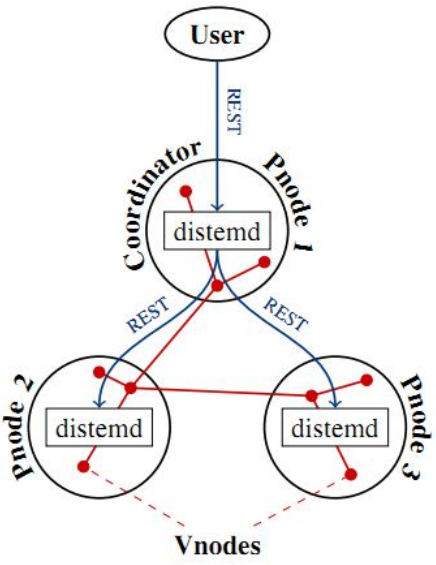


*Figure 1:* The communication architecture of Distem [1].

(iii) *Command line interface:* it leverages the Ruby library above to access Distem from the command line.

# 4 Emulating Geo-replication using Distem

Scaling services over the Internet to meet the needs of ever increasing user base is quite challenging. In order to achieve this, services replicate system state across geographically diverse sites and direct users to the closest site. Understanding the trade-offs on a Geo-replicated setup is critical to development of any system. We, thus, look at emulation of Geo-replication on Grid'5000. In doing so, we discuss below the challenges faced

## 4.1 Latency simulation

Distem allows users to configure latencies for the network interfaces defined on the virtualized nodes. Moreover the latencies can be set to be different for both incoming and outgoing packets. As opposed to link latencies, Distem makes latencies specific to a network interface. Since the network interface is configured on a *VNode*, this implies that the latency is specific to a *VNode*. In a broader sense, to emulate a network with four sites say UW, BR, IE, and SG, we need to define latencies for each site as opposed to the inter-site

*Table 1:* Latencies between the sites [4]

| Latency (in ms) | UW | IE | BR | SG |
|---|---|---|---|---|
| UW | 0.3 | 155 | 207 | 181 |
| IE | | 0.4 | 235 | 350 |
| BR | | | 0.3 | 380 |
| SG | | | | 0.3 |

latencies observed on the cloud. For example, Table 1 shows an example of inter-site latencies. This presents a challenge in determining these site-specific latencies.

### 4.1.1 Formulating the system of equations

Given an N*N inter-site latency matrix and a completely connected topology, there are $\binom{N}{2}$ numbers of links between sites. The above system of equations may be inconsistent and a solution may not exist. Given the edge latency matrix (if inconsistent), we first approximate it so as to obtain a consistent system of equations. This approximation is done using Hit-and-trial that makes some equations redundant and finally leaves the system consistent. Table 2 shows the approximation matrix of the matrix with real latencies shown in table 1.

*Table 2:* Approximated Latency Matrix

| Latency (in ms) | UW | IE | BR | SG |
|---|---|---|---|---|
| UW | 0 | 150 | 150 | 250 |
| IE | | 0 | 250 | 350 |
| BR | | | 0 | 350 |
| SG | | | | 0 |

Given the above inter-site edge latency matrix, we now need to derive a site latency vector from it. As a further simplification, we assume *inbound* latency and *outband* latency on any network interface configured on a virtual node to be equal. This is based on the intuition that a packet sent from one site to another site in one direction observes the same latency as in the other direction. Table 3 summarizes the latency vector.

*Table 3:* Site Latency Vector

| Site | Latency (In = Out) |
|---|---|
| UW | 25 |
| IE | 125 |
| BR | 125 |
| SG | 225 |

## 4.2 Network topology

### 4.2.1 Requirements

The emulated experimental setup must resemble the geographical datacenters. This means that the intra-datacenter latency should be less than 1 millisecond whereas the inter-datacenter latency is as specified in the approximated edge latency matrix. For instance the latency between servers in US and SG should be an order higher than the latencies observed between two servers in US.

### 4.2.2 Limitations and Transformation

Distem limits the latencies to be specified for *Vnode* that represents a site. So latencies are specified for network interfaces defined on each *Vnode*. As such two servers in a datacenter in US (represented as two Vnodes with network interfaces configured) cannot observe a net latency of less than 1 millisecond. Thus there is a need for a complex network topology.

Our network consists of multiple sites each represented as a sub-network. We introduce additional nodes, called *router nodes*, that route a packet between sub-networks. Each router node has two network interfaces: one that connects it to the nodes of its sub-network(representing a site/datacenter) and the other connects it to all the other *router nodes*. Latencies between the two sites (obtained from the approximated edge latency matrix) are imposed on the network interfaces of the *router nodes* forming a global network. Packets between two nodes in different sub-networks have to be routed through the *router nodes* of the sub-networks. Likewise, Latencies on the network interface connecting the *router node* to nodes within a site/datacenter is set to 0 millisecond. This clearly resembles the datacenter architecture while ensuring a completely connected topology.

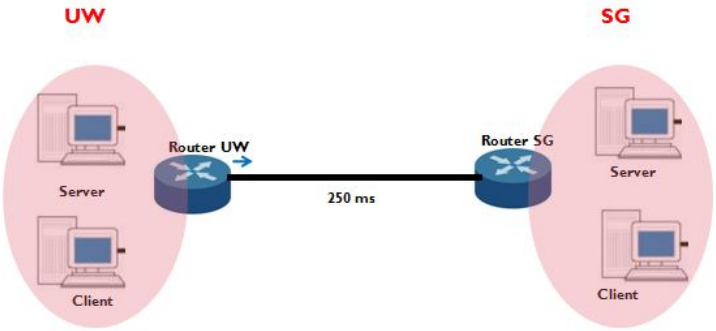


*Figure 2:* Transformed Setup

### 4.2.3 Bug fix in Distem

The network topology defined to resemble the datacenter architecture allowed us to find a bug in Distem. The routes between the sub-networks were not complete. As a workaround, the routing tables on the server nodes, client nodes and *router nodes* were manually configured until the bug was later fixed by Lucas Nussbaum (LORIA) *(distem-bootstrap -G refs/changes/62/2062/1)*. The fix allowed automatic configuration of routing tables in the *VNodes* to route packets properly.

# 5 Experiments

## 5.1 Experimental Setup

The experimental setup includes reserving nodes on Grid'5000, launching the platform and finally experimentation on the same.

(i) *Reservation on Grid'5000:* Physical machines *PNodes* are reserved on Grid'5000 using the *oarsub* utility. Network reservation is done using G5K-subnets. This basically reserves and adds IP ranges to the *oarsub* utility. These IP ranges are routable inside Grid'5000.

(ii) *Launch platform:* Experimental platform is launched from the cluster's frontend on Grid'5000

(1) *kadeploy3* is used to setup the virtual environment in deploy mode. Kadeploy is a scalable, efficient and reliable deployment system (cluster provisioning solution) for cluster and grid computing. It provides a set of tools for cloning, configuring (post installation) and managing cluster nodes.

(2) *distem-bootstrap* launches the distem daemon *distemd* on all the *PNodes*. This automatically sets up a distem environment. When run with-

out options, it creates a distem environment with the latest released version of distem, using nodes reserved from *oarsub* previously.

(3) The virtualized nodes, *VNodes* are created, configured and started. The network interfaces and the router nodes too are configured as defined by the topology.

(4) G-DUR files are transferred to the experimental frontend (on the *Coordinator*).The coordinator is used to perform all the operations on Distem, whatever the node targeted by the operation.

(iii) *Experimentation:* G-DUR is launched from the *Coordinator*

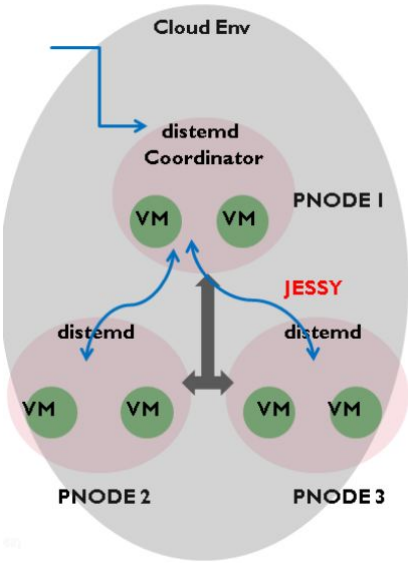


*Figure 3:* Emulated Geo-Replicated Setup

It is important to note that the user provides a system image to boot up the virtualized nodes. The *Vnodes* can also share a system image.

## 5.2 Challenges in Experimentation

Once the experimental environment is set up, one of the major tasks is to make the scripts running G-DUR compatible with the Distem framework. This involves configuring experiment specific parameters, using virtual IPs for the network of *Vnodes* and creating custom images for booting the *Vnodes*.

Unlike Grid'5000 where the files get synchronised automatically by virtue of NFS, experiments on the emulated platform require explicit synchronisation between the virtualized nodes. Distem provides an interface to transfer and synchronise files between the *VNodes* and the experimental frontend. A

considerable amount of time was spent on defining the mapping function between the *VNodes* and the *Pnodes*. The current architecture considers four sites: UW, IR, BR,and SG where different mapping functions were studied:

(i) Map each site (server *VNodes* and client *VNodes*) to one *PNode*. This leads to severe congestion.

(ii) Map each *VNode* to exactly one *PNode*. This requires a lot of physical resources even for experiments with 4 sites (with each site having 2 servers and 2 clients)

(iii) Other mapping functions can also be used.

One of the major challenges that we faced was allocating the physical resources to the *VNodes*. Different transactional protocols hit a saturation point beyond which throughput does not increase much for a given increase in latency. Reaching this saturation point is very critical to compare different protocols. A thorough investigation was done to fine tune the G-DUR experiments in regard to this. This required experimenting with different resource allocation and different mapping strategies to understand saturation points of servers *VNodes*.

Finally we arrived at the current setup which includes 2 *PNodes* (each with 8 cores) for each site. The first *PNode* hosts 2 servers (each with 3 cores) and *router node* (with 2 cores). The second *PNode* hosts 4 clients (each with 2 cores).YCSB benchmark was used to perform the experiments.

At the experimental level, we also changed some configuration settings. Previously on Grid'5000, we never did see the pronounced effect of the initial *warmup period* when the experiment starts. For experiments on the emulated setup we introduced the notion of constant number of operations per client thread for the same. Dampening of *warmup period* is critical to observing the saturation region for different protocols.

# 6 Conclusion and Future Work

Grid'5000, as a scientific instrument, is central to development of many systems. Emulation of Geo-replication is important to understanding the performance and behaviour of these systems at a more realistic level.

For instance, in G-DUR, some phenomena like dampening of *warmup period* become noticeable only when the setup is close to the realtime environment. This further allows us to fine tune the experimental framework making it more realistic. Emulation challenges us to address issues like physical resource allocation and mapping between *PNodes* and *VNodes* thereby understanding the immediate effect on the experimental results which otherwise would not be possible on the cloud. Unlike the cloud, Grid'5000

does not have a pay-as-you-go utility computing model and the experimental setup is constrained by the number of physical nodes and the users running experiments on the clusters. Thus we have to make the best of what we have.

Emulation allows us to study the systems from a more realistic perspective. As future work, we can create and integrate a generic version of emulated framework into Grid'5000 using utilities like *Distem*, *Oarsub*, *Kadeploy3* that are already present. This would allow users to perform two-fold experimentation: one on Grid'5000 and the other on the virtual emulated framework that Grid'5000 would provide.

# References

[1] Luc Sarzyniec, Tomasz Buchert, Emmanuel Jeanvoine and Lucas Nussbaum. "Design and Evaluation of a Virtual Experimental Environment for Distributed Systems." In *PDP 2013*.

[2] Masoud Saeida Ardekani, Pierre Sutra, Marc Shapiro. "G-DUR: A Middleware for Assembling, Analyzing, and Improving Transactional Protocols." In *Middleware 2014*.

[3] Masoud Saeida Ardekani, Pierre Sutra, Marc Shapiro. "Non-Monotonic Snapshot Isolation: scalable and strong consistency for geo-replicated transactional systems." In *SRDS 2013*.

[4] Li *et al.* "Making Geo-Replicated Systems Fast as Possible, Consistent Where Necessary." In *OSDI 2012*.

[5] Grid'5000. "Grid'5000, a scientific instrument designed to support experiment-driven research in all areas of computer science related to parralel, large-scale or distributed computing and networking."

[6] Masoud Saeida Ardekani. Jessy. https://github.com/msaeida/jessy. 2013.


**RESEARCH CENTRE**
**PARIS – ROCQUENCOURT**

Domaine de Voluceau, - Rocquencourt

B.P. 105 - 78153 Le Chesnay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-0803