Testbeds Support for Reproducible Research

Lucas Nussbaum
Inria, Villers-lès-Nancy, F-54600, France
Université de Lorraine, LORIA, F-54500, France
CNRS, LORIA - UMR 7503, F-54500, France
lucas.nussbaum@loria.fr

ABSTRACT

In the context of experimental research, testbeds play an important role in enabling reproducibility of experiments, by providing a set of services that help experiments with setting up the experimental environment, and collecting data about it. This paper explores the status of three different testbeds (Chameleon, CloudLab and Grid'5000) regarding features required for, or related to reproducible research, and discusses some open questions on that topic.

CCS CONCEPTS

• **General and reference** → **Experimentation**; • **Networks** → *Network performance evaluation*;

KEYWORDS

experimentation, testbeds, reproducible research

ACM Reference format:
Lucas Nussbaum. 2017. Testbeds Support for Reproducible Research. In *Proceedings of Reproducibility'17, Los Angeles, CA, USA, August 25, 2017,* 3 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Reproducibility has been the focus of a lot of interest over the recent years, in science in general, and in computer science specifically. But most of this focus has been targeted at the *reproducibility of data analysis*, which is usually handled by a pipeline [11] of several steps involving various tools, starting from *measured data* and going up to *figures* and *tables* included in an article. The various steps of that pipeline involve code for pre-processing the measured data, analyzing data, and presenting data. And the ultimate goal for improving this pipeline is usually considered to be *executable papers*, where the whole process is fully automated.

However, this focus on *reproducibility of data analysis* ignores the important question of how that *measured data* is produced. Experimental computer science generally involves two main methods to acquire data about the *systems under test*: simulation, and experimentation on testbeds. Experimenting on a testbed is a challenging process, and usually involves many different tools: the testbed itself, usually composed of numerous services, but also experiment orchestration solutions [3] ranging from shell scripts to complex frameworks, load or failure injectors, emulation solutions, measurement tools, etc. Each of those components has a huge impact on the experiment and the results that will be obtained from it.

Reproducibility'17, August 25, 2017, Los Angeles, CA, USA
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this paper, we provide an overview of computer science testbeds, and measure the status of three testbeds regarding reproducible research. We explore how design choices, services and features can contribute to reproducibility of experiments, and also highlight missing bits on testbeds and possible areas for improvement.

The paper is structured as follows. Section 2 provides an overview of testbeds, before Section 3 introduces in more details the three selected testbeds. Section 4 reviews how each testbed addressed various requirements for reproducible research. Some open questions are discussed in 5, before the paper is concluded in Section 6.

2 A SHORT PANORAMA OF TESTBEDS

Many testbeds have been designed for Computer Science, to the point that drawing a comprehensive global picture of this world is extremely challenging. Even when limiting only to the networking and distributed systems research communities, testbeds still differ in their focus (object of study): most testbeds focus on a specific field, such as IoT (e.g. IoT-Lab [1]), Internet-scale protocols and services (e.g. Planet-Lab [4] first, then GENI [9], or OFELIA [13]), wireless networking technologies such as WiFi (e.g. R2Lab, w-iLab.t), 4G, or Software Defined Radio (e.g. CorteXLab [8]), clouds, HPC, or Big Data. Some testbeds are able to address several of those fields, because the underlying requirements in terms of infrastructure are similar (for example, Chameleon, CloudLab and Grid'5000 all support cloud, HPC and Big Data experiments at least up to some level, using what could be described as reconfigurable data centers).

Different objects of study raise different requirements in terms of experimental support. Testbeds differ widely in terms of resources available (ranging from wireless sensors to physical servers), of levels of access and control (from the use of virtualization technologies to bare metal reconfiguration), of monitoring features, and of guarantees on the overall environment (level of multi-tenancy on servers and network links). As data points: Planet-Lab used a container technology to slice hosts between several concurrent users, resulting in varying load over time, and the network interconnecting hosts was the real Internet, with no guarantees on performance stability (available bandwidth). On the other hand, Emulab used a cluster of identical, physical hosts, allocated exclusively to users using bare metal deployment, and network conditions are emulated by the Emulab software stack, ensuring stability over time.

Additionally, testbeds are often organized in federations, which provide either identity-level federation (enabling users to access several testbeds using the same credentials), API-level federation (generally by supporting the GENI API), or data-plane federation (to aggregate various resources from various testbeds in a single experiment, usually to increase the scale of the experiment, or to combine different resources, e.g. IoT and clouds). For example, GENI can be seen as a tightly integrated federation of testbeds (*racks*)

throughout the United States. Fed4FIRE (and its follow-up project Fed4FIRE+) is a less tightly integrated federation (not supporting data-plane federation between all testbeds) of a more diverse set of testbeds in Europe.

3 EVALUATED TESTBEDS

In order to compare testbeds that are actually comparable, this paper focuses on three testbeds with a similar scope and a similar architecture: Chameleon[1], CloudLab[2], and Grid'5000 [2]. Those testbeds all welcome experiments in the fields of Cloud Computing (e.g. development and evaluation of custom cloud stacks), Big Data and High Performance Computing (mainly thanks to the availability of HPC networks like Infiniband and accelerators such as GPUs and Xeon Phi). CloudLab, and to a lesser level Grid'5000, are also suitable for experiments in wired networking, thanks to the availability of network topology emulation services. All three testbeds are similar in terms of hardware: each *site* is composed of racks of servers located in a data centers, and sites are interconnected by a dedicated network. All three testbeds are actively used by many users on a daily basis. Each testbed is described in the following paragraphs.

Chameleon is a recent project (started in 2014) composed of two sites, for a total of 424 nodes. Its code base is based on OpenStack, with the addition of Grid'5000 tools, and custom developments.

The CloudLab project also started in 2014 (funded by the same NSF program as Chameleon), but actually has its roots in Emulab [14][3]. Emulab is both a testbed (located in University of Utah) and a software system, developed since 2001 mainly by University of Utah. That code base is also used on numerous other Emulab instances throughout the world[4], such as *imec Virtual Wall* in Belgium. CloudLab is composed of three main sites, for a total of 1081 servers, and is also federated with other Emulab instances.

Grid'5000 is a testbed located mainly in France, composed of 8 sites, for a total of 830 nodes. Most of its software stack has been developed internally since the beginning of the project in 2003.

All three testbeds provide *in-vitro* experimentation: experiments performed on those testbeds receive very little or no influence from the outside world. This is very different from e.g. PlanetLab [4], where conditions in the real Internet have a direct impact on experiment results, and require the use of many repetitions and statistical methods to ensure sound results that do not depend on conditions at a particular time.

However, supporting reproducible research still raises a number of challenges, and requires specific features described in the next section.

4 SERVICES AND FEATURES

4.1 Support for Reconfiguration

The first requirement for such testbeds is to support reconfiguration by experimenters. This is required both to enable experimenters to set up an experimental environment that meets their needs, and later, to recreate the same experimental environment in order to support repeatability. All three testbeds support installing a custom software environment on reserved nodes, at the bare metal, through Frisbee [5] on CloudLab, OpenStack Ironic on Chameleon, and Kadeploy [6] on Grid'5000. Even if those solutions have technical differences, they provide similar functionality.

The approaches for generating the testbed-provided system images differ: while Chameleon and CloudLab don't provide information about the process, Grid'5000 uses Kameleon [12]. This tool supports creating images using a set of recipes similar to configuration management tools such as Puppet, Chef or Ansible, and also uses a cache to ensure that environments generated at different times, but from the same recipes, will not differ if data downloaded from the Internet (e.g. software packages) changed since the image was first generated.

At the networking level, testbeds differ significantly. Chameleon is limited to what is supported by OpenStack Neutron, which is suitable for ensuring network isolation, but not really for creating network topologies, e.g. splitting the network using many VLANs. Grid'5000 uses KaVLAN [2] to achieve that. CloudLab provides very advanced support for networking experiments, with the ability to describe topologies together with network emulation parameters (bandwidth, latency, etc.).

4.2 Support for Collecting Provenance

Another important requirement is the ability to collect data about the experimental environment, in order to provide context for the results obtained during the experiment: which nodes were used? What was their configuration and performance? Such information is required to repeat or replicate experiments[5] (in order to ensure that the type and configuration of resources are identical), and to reproduce experiments and confirm that results are consistent.

The most basic level of support for provenance collection is documentation of available resources. On CloudLab, this is done through a textual description on web pages[6][7]. Additional information about resources is also available through the AM API (see Section 4.4).

Grid'5000 and Chameleon share the same technical solution [7]: a detailed description of all resources (nodes, network equipment) in a set of JSON documents made available through a REST API. Those descriptions are automatically verified using hardware inventory tools and regression tests [10], to avoid errors that could be introduced due e.g. to broken hardware replacements. Those descriptions are also archived, to enable one to explore and reference the past state of the testbed. Finally, those descriptions can be explored through web interfaces[8] to enable experimenters to select adequate resources.

4.3 Support for Long-term Data Storage

It is also important to provide experimenters with a solution to store the large datasets used as part of the experiment, and the artifacts generated during the experiment. All three testbeds provide various services to that effect. ChameleonCloud provides a file-based object store based on OpenStack Swift. CloudLab provides file- and block-stores, with versioning and snapshotting thanks to

[1] https://www.chameleoncloud.org/
[2] https://cloudlab.us/
[3] http://www.emulab.net
[4] https://wiki.emulab.net/Emulab/wiki/OtherEmulabs

[5] terminology: https://www.acm.org/publications/policies/artifact-review-badging
[6] https://www.cloudlab.us/hardware.php
[7] http://docs.cloudlab.us/hardware.html
[8] https://www.chameleoncloud.org/user/discovery/

the use of ZFS as the underlying storage system. Grid'5000 provides a NFS-based service, and two managed Ceph clusters (file, block, or object storage). However, none of the testbeds provide support for exposing that data on the web, e.g. as companion data for articles. External data repositories would have to be used, which is probably a reasonable compromise.

4.4 Support for Automation

A common way to achieve repeatability and replicability is to rely on automation. All three testbeds provide APIs for experimenters to programmatically discover, reserve and setup resources, either using scripts or more advanced experiment orchestration tools [3].

CloudLab uses the SFA AM API, an XMLRPC API which is also used on many GENI and Fed4FIRE testbeds. ChameleonCloud exposes the API provided by the various OpenStack services it uses as building blocks. Grid'5000 provides a custom-made REST API for all its services, and is in the process of adding support for the SFA AM API as part of its integration in the Fed4FIRE+ federation.

Each testbed also provides a way to automate specific experiments, by providing a framework that will automate the steps to setup commonly needed configurations, like the instantiation of an OpenStack cloud. On Cloudlab, this is done through *profiles*; on Chameleon, through *Appliances*; and on Grid'5000, through specific tools (mainly for Ceph and OpenStack deployment).

5 OPEN QUESTIONS

There are open questions about testbeds and reproducible research.

5.1 Respective Responsibilities of Testbeds and Experimenters

A first question is the one of how roles should be split between testbeds and experimenters. How far should testbeds go with providing advanced services to experimenters? What should be left as a burden for experimenters? This is an open question for automation of experiments, for which some testbeds provide a framework, and for monitoring: while it is expected that testbeds help experiments retrieve metrics about the infrastructure (e.g. power usage and network traffic, on Grid'5000), how far should they go with helping experimenters manage their own metrics, for example through agents running on nodes and a monitoring service?

5.2 Load Generation

Another question that is of crucial importance on the testbeds surveyed in this paper is load generation. All three testbeds provide a pristine experimental environment, with no impact from the outside world. As a result, all background traffic, perturbations, variations or faults needed in order to build realistic experiments must be generated and controlled by experimenters, using generators based on traces or probability distributions. However, surprisingly none of the testbeds really provide services to that effect, probably because of the diversity of workloads that need to be injected (ranging from background network traffic to e.g. concurrent usage in a multi-tenant cloud), and of the difficulty to obtain traces.

5.3 Standardization and Federation of Efforts

An upcoming challenge in next years is to standardize interfaces and federate testbeds, to allow the portability of experiments, and to facilitate the reproduction or even the replication of experiments outside of their original testbed. This could be achieved thanks to works on testbeds federations in the GENI and Fed4FIRE+ projects.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we reviewed the state of three testbeds regarding support for reproducible research and discussed some open questions. In the future, this work should be extended to cover other types of testbeds, such as wireless testbeds, and more distributed testbeds. A specific goal could be to define a set of good practices for testbed operators in other to cover basic common functionality and avoid frequent mistakes. Additionally, this work mainly focused on the features advertised by testbeds: it would also be interesting to actually try to perform the same experiment on all three testbeds and draw lessons from that.

REFERENCES

[1] Cédric Adjih et al. 2015. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *IEEE World Forum on Internet of Things*.

[2] Daniel Balouek et al. 2013. Adding Virtualization Capabilities to the Grid'5000 Testbed. In *Cloud Computing and Services Science*. Springer International Publishing, 3–20.

[3] Tomasz Buchert, Cristian Ruiz, Lucas Nussbaum, and Olivier Richard. 2015. A survey of general-purpose experiment management tools for distributed systems. *Future Generation Computer Systems* 45 (2015), 1 – 12.

[4] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. 2003. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review* 33, 3 (2003).

[5] Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb. 2003. Fast, Scalable Disk Imaging with Frisbee.. In *USENIX Annual Technical Conference, General Track*. 283–296.

[6] Emmanuel Jeanvoine, Luc Sarzyniec, and Lucas Nussbaum. 2013. Kadeploy3: Efficient and Scalable Operating System Provisioning for Clusters. *USENIX; login:* 38, 1 (2013), 38–44.

[7] David Margery, Emile Morel, Lucas Nussbaum, Olivier Richard, and Cyril Rohr. 2014. Resources Description, Selection, Reservation and Verification on a Large-scale Testbed. In *TRIDENTCOM*.

[8] Albdelbassat Massouri, Leonardo Cardoso, Benjamin Guillon, Florin Hutu, Guillaume Villemaud, Tanguy Risset, and Jean-Marie Gorce. 2014. CorteXlab: An Open FPGA-based Facility for Testing SDR & Cognitive Radio Networks in a Reproducible Environment. In *INFOCOM'2014 Demo/Poster Session*.

[9] Rick McGeer, Mark Berman, Chip Elliott, and Robert Ricci. 2016. *The GENI Book* (1st ed.). Springer Publishing Company, Incorporated.

[10] Lucas Nussbaum. 2017. Towards Trustworthy Testbeds thanks to Throughout Testing. In *REPPAR - Fourth International Workshop on Reproducibility in Parallel Computing*.

[11] Roger D. Peng. 2015. Reproducible Research. (2015). https://www.coursera.org/learn/reproducible-research Coursera lecture, Week 1, Part 2.

[12] Cristian Ruiz, Salem Harrache, Michael Mercier, and Olivier Richard. 2015. Reconstructable software appliances with kameleon. *ACM SIGOPS Operating Systems Review* 49, 1 (2015), 80–89.

[13] Marc Suñé, Leonardo Bergesio, Hagen Woesner, Tom Rothe, Andreas Köpsel, Didier Colle, Bart Puype, Dimitra Simeonidou, Reza Nejabati, Mayur Channegowda, et al. 2014. Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed. *Computer Networks* 61 (2014), 132–150.

[14] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. 2002. An Integrated Experimental Environment for Distributed Systems and Networks. In *OSDI*.