Versatile Software Framework for the Monitoring and Control of Distributed Computing Systems

Francesco Di Gregorio, Etienne Dupuis, Arnaud Castelltort, Gilles Sassatelli, Abdoulaye Gamatié

Versatile Software Framework for the Monitoring and Control of Distributed Computing Systems

Francesco Di Gregorio, Etienne Dupuis, Arnaud Castelltort, Gilles Sassatelli, Abdoulaye Gamatié
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)
CNRS and University of Montpellier
Montpellier, France
firstname.lastname@lirmm.fr

***Abstract*—This paper presents the development of a versatile software framework enabling the multiform and seamless evaluation of distributed systems: from fast and flexible simulation to actual system execution. The OpenStack datacenter resource management infrastructure is considered as the main entry point of the proposed framework. Since the technological bridge between OpenStack and systems in production already exists, we propose a similar bridge to the popular SimGrid distributed system simulator. This way, actual system resources, and abstract SimGrid system models can be managed uniformly from the OpenStack environment. This paves the way to the multiform design exploration of distributed systems.**

***Index Terms*—Simulation, distributed systems, datacenter, OpenStack, SimGrid**

I. Introduction

A. Context

Distributed computing has been widely present in a number of computing infrastructures, encountered in high-performance and cloud computing domains. Among the major issues related to these systems, we mention the resource allocation problem, which aims at optimizing important metrics such as computation performance, energy consumption, quality-of-service, etc. To address these challenges, some high-level approaches have been proposed in the literature to accelerate and facilitate the design space exploration [1]. These approaches often rely on simulators capable of capturing the system features up to some degree of precision corresponding to the tolerated abstraction assumptions. To validate the conclusions obtained from these abstract reasoning supports, a full-fledged implementation is usually required which can reveal tedious, at least from the engineering cost point of view.

B. Addressed problem

In the present work, we focus on this specific engineering question, by experimenting the connection of a cloud management technology used in production, together with a simulator. We then create a framework continuum that seamlessly enables us to manage either real distributed computing infrastructures or the corresponding abstract simulators. The aim is to automate multi-target experiments depending on the design phase and system scalability. Typically, in early design phases, one would like to evaluate different options pertaining to the choice of adequate system components (servers, memory, interconnect, etc.) to meet the user requirements. On the other hand, for systems with hundreds or thousands of computing nodes, it is often costly to set them up for experiments. Therefore, having fast simulators capable of modeling such large-scale systems is highly desirable.

C. Contribution

The main contribution of this paper is a versatile software framework that enables the multiform and seamless evaluation of distributed systems: from fast and flexible simulation to actual system execution. To make it possible, the OpenStack standard software infrastructure is considered as the entry point of the proposed framework. It is usually deployed for datacenter resources (compute, storage and networking) management in production, through a dashboard or a dedicated API. Since the technological bridge between OpenStack and systems in production already exists, we focus here on the definition of a similar bridge to the popular SimGrid distributed system simulator. This way, actual system resources, and abstract SimGrid system models can be managed uniformly from OpenStack perspective. This paves the way for the multiform design exploration of distributed systems.

D. Outline

The remainder of this paper is organized as follows: Section II first discusses a few related studies; then, Section III describes the overall design approach of the proposed multiform software framework; Section IV focuses on the implementation of the bridge between OpenStack and SimGrid; Section V presents very preliminary evaluation results validating the implemented bridge; finally, Section VI gives some concluding remarks and opens future research directions.

II. Related work

The design space exploration of distributed systems has been addressed following different approaches. For instance, very active research has been devoted to cloud infrastructure simulators [1] [2]. The GreenCloud simulator [3] has been proposed for the evaluation of energy consumption in cloud communications. A fine-grained energy characterization of datacenter IT equipment (servers, network switches, and links) is leveraged by the tool. The GridSim simulator [4] enables the modeling and simulation of heterogeneous grid resources (both time- and space-shared), users and application

models. The CloudSim simulator [5] builds on the same simulation internals as GridSim, but exposes specific interfaces for simulating systems that support cloud services such as virtual machines (VMs), service brokers, and provisioning policies. In addition, it provides visualization support for the management of these concepts. The iCanCloud simulator [6] is another alternative devoted to the definition and integration of brokering policies enabling to minimize the user cost for public cloud infrastructures. Beyond all these simulators, SimGrid [7] certainly provides the most advanced solution by covering the simulation of a variety of large-scale distributed systems, e.g. grid, cloud, high-performance computing, and peer-to-peer systems. Thanks to its rich design and simulation features, we decided to select it for our proposed framework.

While the software engineering problem addressed in the current work could be solved with a mainstream model-driven engineering approach such as in [8], we rather consider a pragmatic approach as in co-simulation tools [9]. The main challenge in our case is the definition of a sound bridge between the connected frameworks, i.e. OpenStack and SimGrid.

III. Overall Approach

The proposed approach consists of a versatile framework relying on the OpenStack[1] platform, an open-source cloud computing platform (see Figure 1).

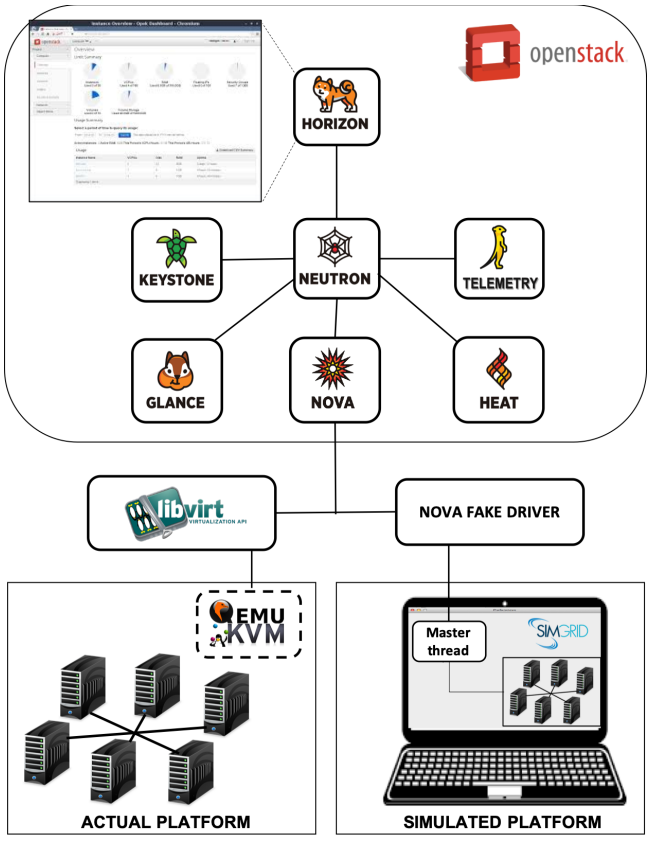

Fig. 1. Overview of the proposed multiform framework.

[1]https://www.openstack.org

It manages both real or simulated cloud infrastructures as depicted in Figure 1. It aims at providing a flexible implementation enabling design scalability upon a rich set of features. This favored by OpenStack, which supports cloud infrastructures via an automatic deployment facility (allowing to scale infrastructure if needed). In particular, it provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complementary services that run on any hardware platform. Each service offers an Application Programming Interface (API) that facilitates its integration. A comprehensive documentation on OpenStack is made available on its dedicated website. A very active community of users frequently provides useful insights on the platform.

The challenge in creating such a framework is to unify two tools of different nature. OpenStack works on actual systems in real-time, while SimGrid is simulation tool assuming a simulated time. Then, a time synchronization is needed between both tools in order to obtain a consistent co-execution. Moreover, time scaling approaches could be used to reduce the simulation time, while keeping precision.

A. OpenStack

From a technical point of view, OpenStack is divided into a number of projects. The most important projects represent the core set while other projects are optional but might be useful in several cases of deployment. The principle of such a deployment is the separation of two concerns: on the one hand, the controller nodes in charge of management services and, on the other hand, the computing nodes responsible for orchestrating the hypervisor and the executive parts of services. Note that virtual machines (VMs) are hosted on compute nodes.

Beyond these two fundamental node types, specific nodes for storage and networking could be deployed as well. Thanks to all these available features, one can deploy OpenStack by configuring a large panel of facilities, going from the operating system to the database system. Nevertheless, this deployment process may be sometimes tedious due to the rich setting parameter space.

Considered OpenStack services. We use a subset of OpenStack services. This subset is briefly described as follows:

- Nova: it is part of compute services and provides massively scalable, on-demand, self-service access to compute resources, including bare metal, virtual machines, and containers. Nova straightforwardly supports the following hypervisors: Hyper-V, VMware, Xen. It can also support KVM, LXC and UML via the LibVirt library, which enables virtualization management.
- Neutron: it is part of networking services and is responsible for provisioning the virtual or physical networks that the compute instances connect to upon booting. It delivers a networking-as-a-service (NaaS) in virtual compute environments.
- Keystone: it is part of shared services and provides identity and authentication for all OpenStack services

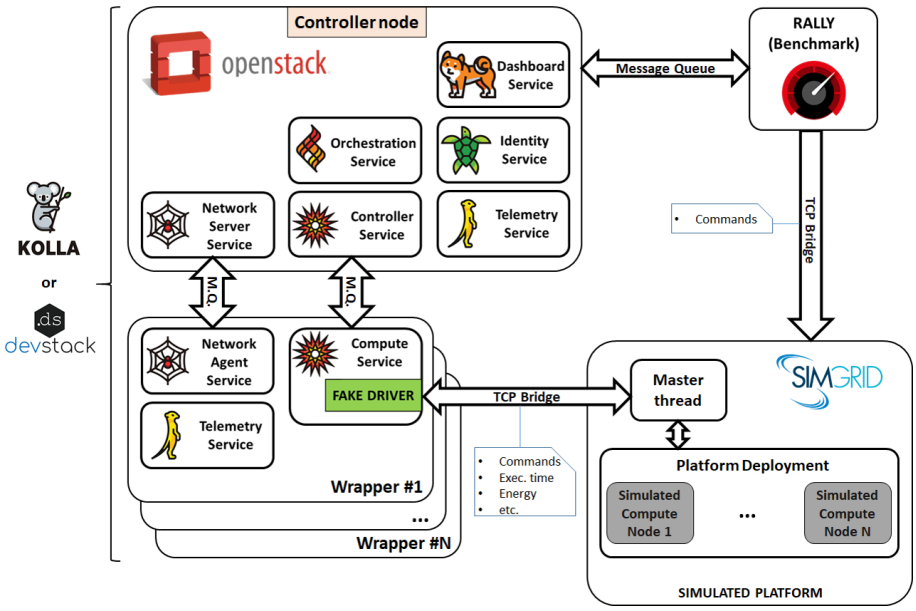

Fig. 2. Implementation of the bridge between OpenStack and SimGrid.

through API client authentication, service discovery, and distributed multi-tenant authorization.

- Glance: it is part of shared services and provides the compute image repository. All compute instances launch from glance images. This is achieved by leveraging its provided functionality for discovering, registering, and retrieving virtual machine images.
- Horizon: it is part of the web front-end services and provides the official user interface for OpenStack, including an extensible dashboard.
- Heat: it is part of orchestration services and enables to launch multiple composite cloud applications based on templates in the form of text files that can be manipulated as code files.
- Telemetry: it offers data collection services and enables us to reliably collect data on the utilization of the physical and virtual resources comprising deployed clouds. The data is saved for subsequent retrieval and analysis.

B. Connecting OpenStack to target systems

The connection of OpenStack to hardware execution platforms is usually straightforward. This is achieved via the Libvirt driver, which allows the Nova Compute Service to communicate with the hypervisor installed on the target platform. This covers the left-hand-side branch of the approach depicted in Figure 1.

Now, the remaining challenge concerns the connection to the chosen simulator, named SimGrid [7] (see the right-hand-side branch of the flow in Figure 1). Here, we replace the Libvirt driver by a ”fake” driver that mimics its functionality. OpenStack developers created this Fake Driver to test cloud platforms when scaling up a given infrastructure. This requires many compute nodes which sometimes are not available or in any case very costly. More precisely, the Fake Driver enables Nova Compute Nodes to answer all requests from the other OpenStack projects without errors while generating fake information. In this way, OpenStack has the illusion that there are real physical compute nodes. Such a mechanism allows one to test the OpenStack platform without having actual machines. Then, some problems, such as network congestion or high latency, can be easily addressed by using the Fake Driver when scaling up the system.

C. The SimGrid simulator

SimGrid [7] is a framework to simulate distributed computer systems, providing information such as execution time and energy consumption. It enables studies in the domains of data-Grids, IaaS Clouds, Clusters, High-Performance Computing, Volunteer Computing, and Peer-to-Peer systems. It can be used to either assess abstract algorithms or to profile and debug real distributed applications. It is free software that aims at assisting developers to explore the best platform configurations given an application.

More concretely, a SimGrid simulation usually employs a Master thread which is in charge of the deployment of the infrastructure described on an XML file and sends all commands to them. The master thread of SimGrid, therefore, acts as a hypervisor. It includes some commands such as createVM, DeleteVM, assign a task to a VM, etc. The accuracy issue of the SimGrid has already been addressed in [7].

IV. Implementation of the Framework

Figure 2 shows the architecture of the proposed framework. We exploit the Fake Driver and the potential of SimGrid to simulate the behavior of very large cloud infrastructure on a single physical machine. While the Fake Driver gives information about the control plane (for example, database,

messaging queue, nova-scheduler) when the system scales up, our implementation produces information about the execution time and energy consumption of the simulated tasks.

A. Deployment of OpenStack

The implemented system consists of a physical machine that runs a controller node and a compute node onto two different VMs using either Kolla or Devstack deployment. The controller node includes all the core projects allowing to deploy and manage the cloud infrastructure. Kolla provides tools to build container images for OpenStack services. For instance, it provides Docker containers for deploying OpenStack on a bare-metal or virtual machine. On the other hand, DevStack consists of extensible scripts used to quickly deploy a complete OpenStack environment. It is quite practical for the functional testing of OpenStack.

B. Focus on the Fake Driver

On the other hand, the compute node deploys the Neutron agent service, Telemetry service, and Nova compute services with the Fake Driver in place of the Libvirt driver. Here, the Fake Driver is a python API, with useful functions as in LibVirt, but each function just returns a success without executing any further instructions. For instance via the Fake Driver, the system can be notified access to 1000 CPUs, 781,3GB of RAM memory and 585.9TB of storage. These values can be changed in the python file if needed.

C. From Fake Driver to SimGrid

A TCP/IP bridge is considered as a mean for sending commands to a thread in charge of running the SimGrid simulation. TCP/IP is a popular protocol of the Internet providing a reliable, ordered and error-checked delivery of a stream between different applications. Concretely, the master thread in SimGrid handles the TCP/IP connection with the Fake Driver and distributes workloads over the worker threads for execution. Afterward, it retrieves the output results from the simulation and sends them back to the driver for monitoring appliance. For this purpose, a TCP/IP connection is put in each function of the Fake Driver. It sends the information to SimGrid and waits for an acknowledgment of OpenStack services. By doing so, we replace the physical cluster by the simulated platform. The resulting implemented mechanism, therefore, synchronizes the simulated time in SimGrid with the execution time in OpenStack.

V. Preliminary Experiments

A. Setup description

Our experimental setup is based on two hosts on which the components of the OpenStack execute. We use VMs instead of bare metal servers because they can be easily created and managed. These two ”virtual” hosts are connected to the Internet by NAT13. They are connected to two Host-only network interfaces allowing them to communicate with each other and with the actual host machine.

The first VM plays the role of a controller node. It is the most important and includes most of OpenStack components. The second VM represents the network node and hosts the Neutron service. Then, with the Kolla containerization, we can create as many fake compute nodes as needed. Each of them run in a container on the controller node.

B. Benchmarking approach

The considered benchmarking model consists of testing the behavior of a cloud platform by evaluating typical scenarios like VM creation and deletion. To achieve the experiments, we use the Rally benchmarking and performance analysis tool for OpenStack. Among the supports provided by Rally, we can mention the automatic measuring and profiling of OpenStack performance. In particular, Rally allows the detection of any scaling and performance issue, and an analysis of how different deployment architectures and hardware affect the performance of OpenStack. These facilities of Rally can be exploited by a user through proper scripts describing some operational scenarios of cloud infrastructure.

Typical scenarios specifying commands such as: booting VMs, deleting VMs, listing VMs, migrating VMs, etc. Rally proposes several such micro-benchmarks useful to test the platform. In the sequel, we use the VM booting and deleting micro-benchmarks for simplicity reasons. But, advanced scenarios involving VM migration and workload execution could be considered into the framework, as well. The results of these micro-benchmarks are collected and analyzed from diverse perspectives: either with SimGrid visualization tools or via Rally by exploiting the statistics generated from the system simulation via OpenStack.

C. Results

A typical working scenario has been tested which consists in alternating the boot and deletion of virtual machines, in a sequence of 10 iterations. For this purpose, we defined a Rally script that delivers the corresponding commands to OpenStack. Next, the commands are forwarded to the SimGrid simulation through the Fake Driver and the TCP/IP bridge.

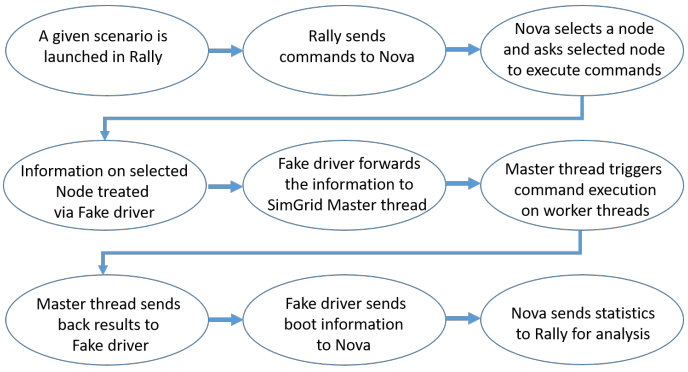

Fig. 3. Steps in a single iteration of commands execution.

The commands are executed afterward on the simulated platform within SimGrid by the Master thread. The statistics

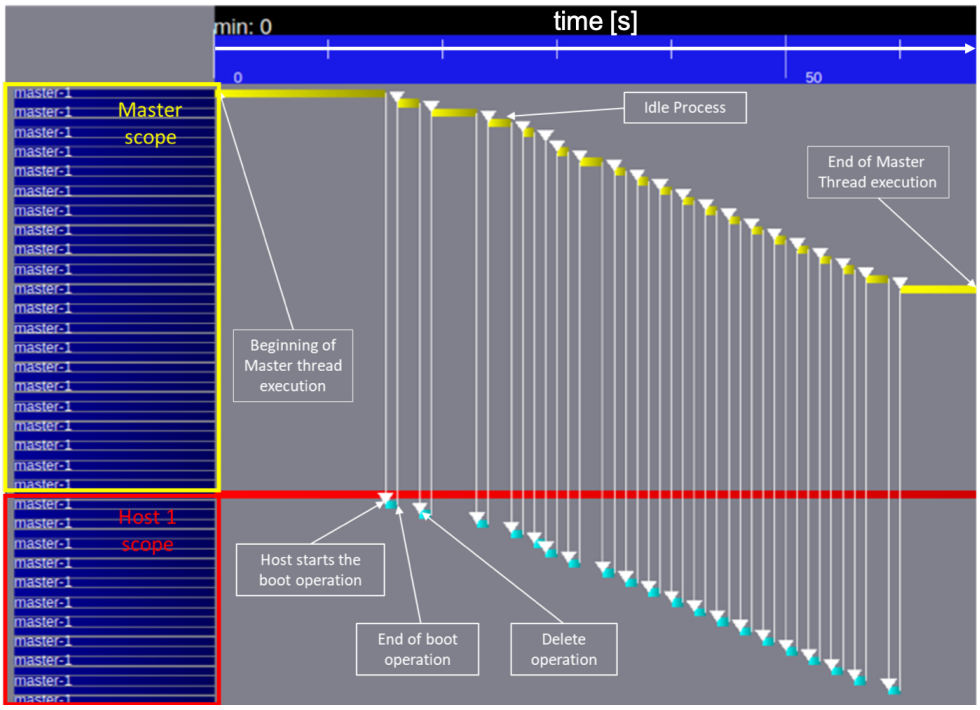

Fig. 4. Execution trace generated by the SimGrid Simulator.

TABLE I
STATISTICS ON DURATION OF EACH STEP.

Action	Minimun delay (sec)	Median delay (sec)	Maximum delay (sec)	Average delay (sec)	Successful completion (%)	Iteration count
nova.boot_server	3.753	3.992	4.133	3.96	100.0	10
nova.delete_server	2.32	2.386	2.511	2.399	100.0	10
total	**6.116**	**6.358**	**6.571**	**6.36**	**100.0**	**10**
duration	*5.116*	*5.358*	*5.571*	*5.36*	*100.0*	*10*
idle_duration	*1*	*1*	*1*	*1*	*100.0*	*10*

of the simulated scenario, e.g. execution time, is ultimately returned to Rally, through OpenStack. Figure 3 represents the different steps within any single iteration where some commands (e.g. VM booting, deletion) are executed in our devised software flow.

Temporal evolution of the simulated scenario in SimGrid. The visual execution trace generated by the SimGrid simulation is shown in Figure 4. The left-hand side part corresponds to threads instances. The yellow-colored scope represents the activity on the Master thread side, while the red-colored scope describes the action on the host thread side. Note that the host thread receives commands to execute from the master thread.

In Figure 4, a Master thread instance starts execution at instant 0 (i.e. the yellow segments above the red horizontal line). After a few seconds, it issues a VM boot command, sent to a Host thread instance. On the completion of the boot operation (i.e. the light-blue segments below the red horizontal line), the Master thread takes back the control of the execution. After a few idle seconds, the Master thread issues a deletion command for the VM booted previously by the Host thread instance. Upon deletion, the control is returned to the Master thread instance. The same scenarios are iterated nine times, until the completion of the Master thread execution. The duration between the very first command issued by the Master thread instance and the very last command it receives from the Host thread instance is about 45 sec.

Another perspective of the simulation statistics acquired from SimGrid is shown in Table I and Figure 5. The former provides global timing statistics while the latter focuses on the detailed breakdown between idle time and active times for each of the 10 boot-delete iterations. These statistics result from Rally, which retrieves the data from OpenStack itself connected to SimGrid. The aim of this perspective is to validate the correct synchronization between the OpenStack command management and the SimGrid simulation.

According to the statistics summarized in Table I, the average time for booting and deleting a VM is about 5.36 sec. The execution of the boot command takes on average 3.96 sec while that of delete command takes 2.39 sec. The average idle period is 1 sec. Further useful information is also available, such as the percentage of successful command execution over the whole iterations; the minimum, median and

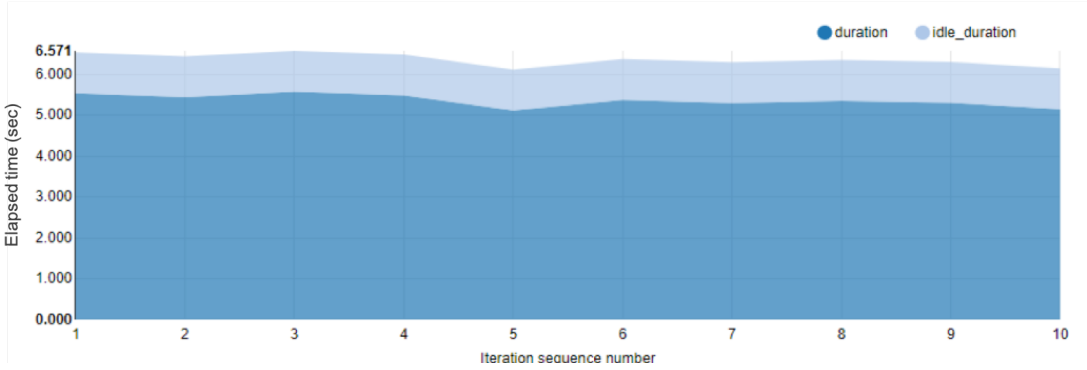

Fig. 5. Duration report per iteration.

maximum delays of each command.

Energy consumption of the simulated scenario in SimGrid. Finally, the evolution of the energy consumption corresponding to Host 1 is illustrated in Figure 6. It has been obtained from SimGrid based on the simulation of two different tasks representing the boot and deletion commands. The figure mainly shows the increase in energy during the processing of the twenty commands.

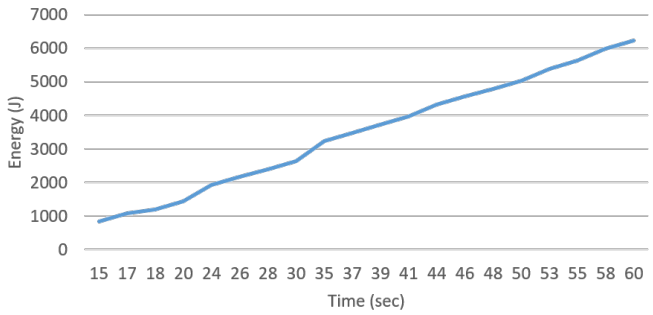

Fig. 6. Energy consumption of Host 1 from the SimGrid simulation.

VI. Conclusion and Perspectives

In this paper, we presented the implementation of a versatile software framework that enables the multiform and seamless evaluation of distributed systems: from fast and flexible simulation to actual system execution. We build a bridge between the OpenStack standard software infrastructure with the popular SimGrid distributed system simulator. This paves the way for the multiform design exploration of distributed systems. As a preliminary validation of the proposed framework, we illustrated a simple execution scenario consisting of virtual machine booting and deletion on a cloud computing infrastructure model. The source code defining the implementation of the proposed bridge between OpenStack and SimGrid is freely available[2]. It currently covers the commands illustrated in the previous section only.

Our future work aims at integrating further design evaluation metrics (e.g. communication bandwidth, energy/power consumption) beyond the timing information currently reported in the output of Rally. We also plan to design full test scenarios that cover representative cloud system operations so as to assess the scalability of the defined framework.

[2]https://gite.lirmm.fr/openstack_SimGrid/bridge

References

[1] J. Byrne, S. Svorobej, K. M. Giannoutakis, D. Tzovaras, P. J. Byrne, P.-O. Östberg, A. Gourinovitch, and T. Lynn, "A review of cloud computing simulation platforms and related environments," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, Portugal, 2017, pp. 679–691.

[2] R. Ursu, K. Latif, D. Novo, M. Selva, A. Gamatié, G. Sassatelli, D. Khabi, and A. Cheptsov, "A workflow for fast evaluation of mapping heuristics targeting cloud infrastructures," *CoRR*, vol. abs/1601.07420, 2016. [Online]. Available: http://arxiv.org/abs/1601.07420

[3] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "Greencloud: A packet-level simulator of energy-aware cloud computing data centers," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec 2010, pp. 1–5.

[4] R. Buyya and M. M. Murshed, "Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurr. Comput. Pract. Exp.*, vol. 14, pp. 1175–1220, 2002.

[5] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *2009 International Conference on High Performance Computing & Simulation, HPCS 2009, Leipzig, Germany, June 21-24, 2009*, W. W. Smari and J. P. McIntire, Eds. IEEE, 2009, pp. 1–11. [Online]. Available: https://doi.org/10.1109/HPCSIM.2009.5192685

[6] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *J. Grid Comput.*, vol. 10, no. 1, pp. 185–209, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1007/s10723-012-9208-5

[7] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014. [Online]. Available: http://hal.inria.fr/hal-01017319

[8] A. Gamatié, S. L. Beux, É. Piel, R. B. Atitallah, A. Etien, P. Marquet, and J. Dekeyser, "A model-driven design framework for massively parallel embedded systems," *ACM Trans. Embedded Comput. Syst.*, vol. 10, no. 4, pp. 39:1–39:36, 2011. [Online]. Available: https://doi.org/10.1145/2043662.2043663

[9] A. T. Al-Hammouri, M. S. Branicky, and V. Liberatore, "Co-simulation tools for networked control systems," in *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, ser. HSCC '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 16–29. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78929-1_2