SimGrid: a Sustained Effort for the Versatile Simulation of Large Scale Distributed Systems

Henri Casanova
ICS Dept., U. of Hawai'i
Manoa, U.S.A

Arnaud Giersch
FEMTO-ST, U. of Franche-Comté
Belfort, France

Arnaud Legrand
LIG, CNRS
Grenoble, France

Martin Quinson
Loria, U. de Lorraine
Nancy, France

Frédéric Suter
IN2P3 Computing Center, CNRS
Lyon-Villeurbanne, France

arXiv:1309.1630v1 [cs.DC] 6 Sep 2013

1 Introduction

Computers have revolutionized science by fully automating the mathematical computations involved in modeling activities. This move toward computational science has proved decisive throughout modern science and engineering. The computations underlying these scientific advances are carried out on very large computing facilities the scale of which is ever increasing. For instance, current dedicated High Performance Computing (HPC) systems can comprise hundreds of thousands of cores. Furthermore, Grid systems are deployed as transnational federations of large-scale computing facilities that can aggregate millions of heterogeneous computing elements in deep hierarchies. Such systems are certainly among the most complex artifacts ever built by mankind. Then, cloud computing and virtualization technologies allow to outsource a computing infrastructure to dedicated data centers whose physical location is transparent to the users. Finally, among the most complex large-scale platforms are the decentralized infrastructures enabled by P2P technologies, which federate widely heterogeneous compute and storage resources located at the edge of the network.

While used routinely, these platforms remain for the most part poorly understood. Simulation is an appealing approach to study large-scale distributed systems. Simulation consists in predicting the evolution of a behavior of the system (both the application and the platform) through numerical and algorithmic models. Simulation experiments are *fully repeatable and configurable*, making it possible to explore "what if" scenarios without an actual platform deployment. Furthermore, simulation is often less labor intensive, costly, and/or time consuming than conducting experiments on a real platform. Simulation raises its own issues, among which the question of the induced simulation bias is critical. Nevertheless, simulation has proved useful in most scientific and engineering disciplines, and holds the same promise for the study of large-scale distributed computing platforms.

Many simulators have been developed over the last decade for the simulation of almost every kind of large scale distributed systems. However, the field is highly partitioned and each tool is usually limited to a specific domain, and even to a specific study within that domain. Moreover the simulators, most of which do not survive beyond the lifetime of a given funded project, are rarely made available or even detailed in publications. For instance, [6] points out that out of 141 surveyed papers that use simulation for the study of P2P systems, 30% use a custom simulator, and 50% do not even report which simulator was used. The consequence is that most published simulation results are impossible to reproduce by researchers other than the authors. And yet, simulation results should be easily repeatable by design!

A noticeable exception is the SimGrid framework [1]. This versatile scientific instrument has been used successfully for simulation studies in Grid Computing, Cloud Computing, HPC, Volunteer

Computing and P2P Systems. It was shown both more realistic and more scalable than its major competitors, thus blurring the boundaries between research domains. Over the past five years only, this framework was used by more than 100 distinct authors from 4 continents in over 60 research articles and Ph.D. dissertations. Its development has been sustained for nearly 15 years. In Section 2 we briefly retrace the history and evolution of SimGrid. In Section 3 we explain how SimGrid has achieved sustained development. Finally, Section 4 provides a brief summary of the lesson learned while developing SimGrid.

2 SimGrid: A Versatile Scientific Instrument

SimGrid is a 15-year old open source project whose domain of application has kept growing since its inception. It was initiated in 1998 by H. Casanova at the University of California, San Diego as a way to factor and consolidate different simulators developed by different graduate students for studying scheduling algorithms for heterogeneous platforms. The first version of SimGrid [2] made it easy to prototype scheduling heuristics and to test them on a variety of abstract applications (expressed as task graphs) and platforms.

In 2003, the work of A. Legrand at the École Normale Supérieure de Lyon led to the second major release of SimGrid [4] that extended the capabilities of its predecessor in two major ways. First, the accuracy of the simulation models was improved by transitioning the network model from a wormhole model to an analytical fluid model. Second, an API, called MSG, was added to simulate generic Communicating Sequential Processes (CSP) scenarios.

The third major release of SimGrid was distributed in 2005, but major new features appeared in version 3.3 in April 2009 [5], after contributions from M. Quinson and F. Suter. These features include a complete rewrite of the simulation core for better modularity, speed, and scalability, the possibility to attach traces to resources to simulate time-dependent performance characteristics as well as failure events. Two new user interface were added: SimDag, which revives some of the original concepts from the first release, and GRAS, which allows users to develop/test code in simulation and then run it unmodified on a real platform.

From early 2009 up until today, and thanks to securing large amounts of funding, the domain of application of SimGrid has been extended to P2P, HPC and Cloud infrastructures and applications. New models and concepts have been added to the code base and a new API for the simulation of unmodified MPI applications was proposed. In the same period, the internals became more mature thanks to the work of A. Giersch and the heavy use of a multi-platform/multi-OS testing infrastructure. Finally the simulation framework itself now comes with an ecosystem of tools ranging from the generation of simulation scenarios to the visualization of simulation results.

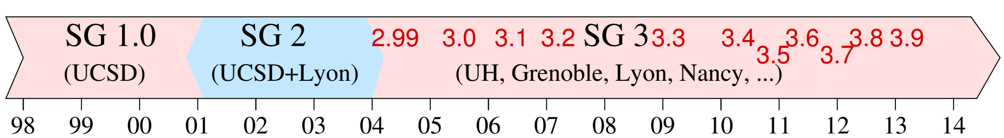

Figure 1: The SimGrid development timeline.

Fig. 1 summarizes the release timeline of SimGrid since its very first version and indicates the main locations of the members of the development team. In 15 years, SimGrid evolved from a project developed in-house at a single laboratory into an international collaboration that involves more than 20 active contributors. It also evolved from a very domain-specific simulator into a *versatile scientific instrument*, whose performance and accuracy are continuously and thoroughly assessed, for the study of large scale distributed systems [3].

3 Sustaining SimGrid as a Research Software

The sustained evolution of SimGrid was made possible by different factors: obtaining *recurring funding*, which provides manpower, *improving the code base*, which can be accomplished via both custom and standard tools, and retaining and growing the *user base*.

Securing Recurrent Funding and Support – As mentioned earlier, most simulators do not survive beyond the lifetime of the grant that funded its development. However, the SimGrid project was developed without specific financial support for about eight years. The development effort was then supported by A. Legrand and M. Quinson alone. They used SimGrid early on for their Ph.D. research, and then continued to use and develop it as part of their subsequent research activities. But using and developing a tool both for one's own research and for reaching out to a larger user community requires additional manpower. Since 2006, the SimGrid project was funded by four major grants. We coupled funding requests on scientific aspects, which allowed us to extend the domain of application SimGrid and enlarge the community of research that use it and contributed to it, with funding requests targeted to the technical support and development necessary to strengthen the code base. The USS SimGrid[1] project (2009-2012) aimed at simulating P2P and volunteer computing systems, while the SONGS[2] project (2013-2015) add two more application domains: HPC and Cloud computing. This funding provided manpower, i.e., Ph.D students, postdocs, and engineers to tackle the challenging issues raised by the new application domains. Two more grants, solely dedicated to software engineering, were also secured over that time period. They led to improved implementation, portability, testing capabilities, packaging, and documentation.

The key to sustainability, at least in our case, was to request funding from different sources, i.e., national funding agency or academic institutes and universities, both for scientific and for technical aspects of the project. The successes achieved in the scope of a given grant generally helps to secure a subsequent one. However, the ability and willingness to sustain the research and software development efforts in the (temporary) absence of funding have also been essential.

Increasing Code Stability and Quality – Going from an in-house project at a single laboratory to a widely distributed and used Open Source simulation software requires to make the code base as robust, stable, available, usable, and trustworthy, as possible. While the research activities improve the models and simulations quality and make the obtained results trustworthy, a large body of engineering work has been done about the other requirements. Since March 2005, the project is hosted on a forge [1] operated by Inria, the French Institute for Computer Science, which ensures a sustained maintenance. It offers classical services, such as a git repository, bug tracking systems, mailing lists, or web hosting. SimGrid is distributed along with more than 250 unit tests and examples that are managed by a custom tool called *tesh* (testing environment shell). Tesh runs the complete test suite and checks for discrepancies in output that indicate the apparition of bugs in the code. Thanks to dashboard[3] and continuous integration[4] platforms, provided by Inria, the code base is regularly tested on several architectures and operating systems combinations. Code

[1] http://uss-simgrid.gforge.inria.fr
[2] http://infra-songs.gforge.inria.fr
[3] http://cdash.inria.fr/CDash/index.php?project=SimGrid
[4] http://ci.inria.fr

coverage and memory leaks are also monitored thanks to these platforms. All these tools helped to increase the release frequency seen in Fig. 1.

The key point is that while research usually is the main focus of the developers of a scientific software, sustainability requires an important amount of engineering work that cannot be neglected. Leveraging available tools and platforms that can ease this task is crucial.

Fostering a User Base – The main reason for a scientific software to perdure is its user community, which should be growing and encouraging user-driven contributions. The most standard way to interact with users is via a mailing list. More than a hundred people subscribe to `simgrid-user@lists.gforge.inria.fr` and *every* message is answered in a reasonable delay. This is a good way to retain existing users and newcomers, but getting more users requires more proactive communication actions. Over the last five years we have developed a comprehensive set of tutorials, which are available online, and have greatly improved the user documentation. SimGrid is also presented on the Inria booth at the SuperComputing conference since 2009. We also initiated a series of events, the SimGrid Users Days, that are an opportunity for advanced users to meet developers and discuss the latest features of SimGrid and currently explored research areas as well as to influence future developments. The last edition in June 2013 was a bit different. Users and developers of SimGrid participated to a 3-day "coding sprint." This was an opportunity to solve existing issues and to address new challenges jointly with users on very concrete use cases.

We believe that communicating about the tool outside of regular scientific publications and making efforts to interact with users on a regular basis are key actions to make SimGrid perdure and evolve beyond the limits of its original developers' domain of research.

4 Conclusion

In this paper we presented SimGrid, a toolkit for the versatile simulation of large scale distributed systems, whose development effort has been sustained for the last fifteen years. Over this time period SimGrid has evolved from a one-laboratory project in the U.S. into a scientific instrument developed by an international collaboration. The keys to making this evolution possible have been *securing of funding*, *improving the quality of the software*, and *increasing the user base.* In this paper we have described how we have been able to make advances on all three fronts, on which we plan to intensify our efforts over the upcoming years.

References

[1] Simgrid. http://simgrid.gforge.inria.fr.

[2] H. Casanova. Simgrid: A Toolkit for the Simulation of Application Scheduling. In *Proc. of CCGrid'01*, pages 430–437, 2001.

[3] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Systems. *JPDC (under review)*, 2013.

[4] H. Casanova, A. Legrand, and L. Marchal. Scheduling Distributed Applications: the SimGrid Simulation Framework. In *Proc. of CCGrid'03*, pages 138–145, 2003.

[5] H. Casanova, A. Legrand, and M. Quinson. SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *Proc. of the 10th Intl. Conf. on Computer Modeling and Simulation (UKSim)*, 2008.

[6] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. Towards Yet Another Peer-to-Peer Simulator. In *Proc. of the 4th Intl. Conf. on Performance Modelling and Evaluation of Heterogeneous Networks*, 2006.