J. Parallel Distrib. Comput. 74 (2014) 2899–2917

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Versatile, scalable, and accurate simulation of distributed applications and platforms

Henri Casanova [a], Arnaud Giersch [b], Arnaud Legrand [c], Martin Quinson [d], Frédéric Suter [e,f,*]

[a] Department of Information and Computer Sciences, University of Hawai'i at Manoa, USA
[b] FEMTO-ST, University of Franche-Comté, Belfort, France
[c] LIG, CNRS, Grenoble University, France
[d] LORIA, Université de Lorraine, France
[e] IN2P3 Computing Center, CNRS/IN2P3, Lyon-Villeurbanne, France
[f] LIP, INRIA, ENS Lyon, Lyon, France

HIGHLIGHTS

- We provide a presentation of the improvements done in SimGrid in the last 10 years.
- We rebut popular wisdom that specialization allows for "better" simulation.
- We claim that versatility leads to better accuracy and better scalability.
- We back up this claim with multiple use cases and (in)validation studies.

ARTICLE INFO

Article history:
Received 6 September 2013
Received in revised form
10 June 2014
Accepted 19 June 2014
Available online 10 July 2014

Keywords:
Simulation
Validation
Scalability
Versatility
SimGrid

ABSTRACT

The study of parallel and distributed applications and platforms, whether in the cluster, grid, peer-to-peer, volunteer, or cloud computing domain, often mandates empirical evaluation of proposed algorithmic and system solutions via *simulation*. Unlike direct experimentation via an application deployment on a real-world testbed, simulation enables fully repeatable and configurable experiments for arbitrary hypothetical scenarios. Two key concerns are accuracy (so that simulation results are scientifically sound) and scalability (so that simulation experiments can be fast and memory-efficient). While the scalability of a simulator is easily measured, the accuracy of many state-of-the-art simulators is largely unknown because they have not been sufficiently validated. In this work we describe recent accuracy and scalability advances made in the context of the SimGrid simulation framework. A design goal of SimGrid is that it should be versatile, i.e., applicable across all aforementioned domains. We present quantitative results that show that SimGrid compares favorably with state-of-the-art domain-specific simulators in terms of scalability, accuracy, or the trade-off between the two. An important implication is that, contrary to popular wisdom, striving for versatility in a simulator is not an impediment but instead is conducive to improving both accuracy and scalability.

© 2014 Elsevier Inc. All rights reserved.

The use of parallel and distributed computing platforms is pervasive in a wide range of contexts and for a wide range of applications. *High Performance Computing* (HPC) has been a consumer of and driver for these platforms. In particular, commodity clusters built from off-the-shelf computers interconnected with switches have been used for applications in virtually all fields of science and engineering, and exascale systems with millions of cores are already envisioned. Platforms that aggregate multiple clusters over wide-area networks, or *grids*, have received a lot of attention over the last decade with both specific software infrastructures and application deployments. Distributed applications and platforms have also come to prominence in the *peer-to-peer* and *volunteer computing* domains (e.g., for content sharing, scientific computing, data storage and retrieval, media streaming), enabled by the impressive capabilities of personal computers and high-speed

* Corresponding author at: IN2P3 Computing Center, CNRS/IN2P3, Lyon-Villeurbanne, France.
E-mail address: frederic.suter@cc.in2p3.fr (F. Suter).

http://dx.doi.org/10.1016/j.jpdc.2014.06.008
0743-7315/© 2014 Elsevier Inc. All rights reserved.

personal Internet connections. Finally, *cloud computing* relies on the use of large-scale distributed platforms that host virtualized resources leased to consumers of compute cycles and storage space.

While large-scale production platforms have been deployed and used successfully in all these domains, many open questions remain. Relevant challenges include resource management, resource discovery and monitoring, application scheduling, data management, decentralized algorithms, electrical power management, resource economics, fault-tolerance, scalability, and performance. Regardless of the specific context and of the research question at hand, studying and understanding the behavior of applications on distributed platforms is difficult. The goal is to assess the quality of competing algorithmic and system designs with respect to precise objective metrics. Theoretical analysis is typically tractable only when using stringent and ultimately unrealistic assumptions. As a result, relevant research is mostly empirical and proceeds as follows. An *experiment* consists in executing a *software application* on a target *hardware platform*. We use the term "application" in a broad sense here, encompassing a parallel scientific simulation, a peer-to-peer file sharing system, a cloud computing brokering system, etc. The application execution on the platform generates a time-stamped trace of events, from which relevant metrics can be computed (e.g., execution time, throughput, power consumption). Finally, research questions are answered by comparing these metrics across multiple experiments.

One can distinguish three classes of experiments. In *in vivo* experiments an actual implementation of the application is executed on a real-world platform. Unfortunately, real-world production platforms may not be available for the purpose of experiments. Even if a testbed platform is available, experiments can only be conducted for (subsets of) the platform configuration at hand, limiting the range of experimental scenarios. Finally, conducting reproducible in vivo experiments often proves difficult due to changing workload and resource conditions. An alternative that obviates these concerns is *in vitro* experiments, i.e., using emulation (e.g., virtual machines, network emulation). A problem with both in vivo and in vitro experiments is that experiments may be prohibitively time consuming. This problem is exacerbated not only by the need to study long-running applications but also by the fact that large numbers of experiments are typically needed to obtain results with reasonable statistical significance. Furthermore, when studying large-scale applications and platforms, commensurate amounts of hardware resources are required. Even if the necessary resources are available, power consumption considerations must be taken into account: using large-scale platforms merely for performance evaluation experiments may be an unacceptable expense and a waste of natural resources. The third approach consists in running (an abstraction of) the application *in silico*, i.e., using simulation. This approach is typically less labor-intensive, and often less costly in terms of hardware resources, when compared to in vivo or in vitro experiments. Consequently, it should be no surprise that many published results in the field are obtained in silico.

Two key concerns for simulation are *accuracy* (the ability to run in silico experiments with no or little result bias when compared to their in vivo counterparts) and *scalability* (the ability to run large and/or fast in silico experiments). A simulator relies on one or more simulation models to describe the interaction between the simulated application and the simulated platform. There is a widely acknowledged trade-off between model accuracy and model scalability (e.g., an analytical model based on equations may be less accurate than a complex event-driven procedure but its evaluation would also be less memory- and CPU-intensive). Simulation has been used in some areas of Computer Science for decades, e.g., for microprocessor and network protocol design, but its use in the field of parallel and distributed computing is less developed. While the scalability of a simulator can be easily quantified, evaluating its accuracy is painstaking and time consuming. As a result, published validation results often focus on a few scenarios, which may be relevant to a particular scope, instead of engaging in a systematic and critical evaluation methodology. Consequently, countless published research results are obtained with simulation methods whose accuracy is more or less unknown.

An important observation is that simulators used by parallel and distributed computing researchers are domain-specific (e.g., peer-to-peer simulators, grid simulators, HPC simulators). In some cases, domain-specificity is justified. For instance, wireless networks are markedly different from wired networks and in this work, for instance, we only consider wired networks. But, in general, many simulators are developed by researchers for their own research projects and these researchers are domain experts, not simulation experts. The popular wisdom seems to be that developing a versatile simulator that applies across domains is not a worthwhile endeavor because specialization allows for "better" simulation, i.e., simulations that achieve a desirable trade-off between accuracy and scalability. In this work, we *rebut popular wisdom* and claim that, when developing a simulation framework, *aiming for versatility is the way to achieve better accuracy and better scalability*. Our main contribution is that we confirm this claim by synthesizing the experience gained during the 10-year development of the SimGrid discrete-event simulation framework, presenting results relating to both simulation design and simulation implementation. Some of these results have been previously published in conference proceedings, as referenced hereafter, while others are novel contributions.

The rest of this article is organized as follows. Section 1 presents related work. Section 2 discusses the current design and design goals of SimGrid. Sections 3 and 4 explain how striving for versatility has led to advances in accuracy and scalability, respectively. While these sections include several short case studies, Section 5 presents a full-fledged case study in the HPC domain. Finally, Section 6 concludes the paper with a brief summary of findings and with perspectives on future work.

1. Related work

In this section we discuss popular simulators that have been used in the last decade and whose goal is to enable "fast" simulation of grid, cloud, peer-to-peer, volunteer, or HPC applications and platforms, meaning that the simulation time (i.e., the runtime of the in silico experiment) should be orders of magnitude faster than the simulated time (i.e., the simulated runtime of the application). Most of these simulators share the same design with three components: (i) simulation models; (ii) platform specification; and (iii) application specification. *Simulation models* are used to implement the evolution of simulated application activities (computations, data transfers) that use simulated resources (compute devices, network elements, storage devices) throughout simulated time. More specifically, given all the application activities that use a set of resources, resource models are used to compute the completion date of the activity that completes the earliest and the progress made by all other activities by that date. *Platform specification* mechanisms allow users to instantiate platform scenarios without having to modify the simulation models or the simulation's implementation. Each resource must be described using an instantiated simulation model, and resources can be connected together (e.g., a particular set of network links and routers is used for end-to-end communication between two compute resources). *Application specification* refers to the set of mechanisms and abstractions for users to describe the nature and sequence of activities that must be simulated. Existing simulators provide many options for

Download English Version:

https://daneshyari.com/en/article/433020

Download Persian Version:

https://daneshyari.com/article/433020

Daneshyari.com