WRENCH 2.0: Cyberinfrastructure simulation workbench

Henri Casanova[1]¶, Jesse McDonald[1], Loïc Pottier[2], and Rafael Ferreira da Silva[*][3]

1 Information and Computer Sciences Dept., University of Hawaiʻi at Manoa **2** Information Sciences Institute, University of Southern California **3** National Center for Computational Sciences, Oak Ridge National Laboratory ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 10 April 2022
Published: unpublished

Summary

Computer Science researchers in the field of parallel and distributed computing typically need to conduct large experimental campaigns to validate hypotheses and evaluate proposed approaches. These campaigns are often labor-intensive (due to the need to develop, configure, and deploy the required software stacks), and time- and energy-intensive (due to the need to obtain statistically significant numbers of experimental results that are never perfectly reproducible on real-world platforms). Furthermore, experimental scenarios are limited to those platforms configurations available to the researcher, thus making it difficult to explore a range of scenarios that span current, upcoming and/or hypothetical platform settings. One way to side-step these difficulties and constraints is to use *simulation*. The objective of the WRENCH framework is to provide high-level abstractions and corresponding APIs to make it straightforward to implement simulators of complex experimental scenarios that are accurate and scalable.

Statement of Need

Over the last couple of decades, many frameworks have been developed that can be used to build simulators of parallel and distributed computing applications, runtime systems, and platforms (Tikir et al., 2009) (Hoefler et al., 2010) (Buyya & Murshed, 2002) (Calheiros et al., 2011) (Núñez et al., 2011) (Kecskemeti, 2015) (Malik et al., 2014) (Qayyum et al., 2018) (H. Casanova et al., 2014) (Carothers et al., 2000), some of which are being actively developed and have garnered sizable user communities among Computer Science researchers. These frameworks differ drastically in terms of simulation accuracy, simulation scalability, and easy-of-use. Simulation accuracy should be the overriding user concern, and yet some popular (because they are easy to use) simulation frameworks have been shown to suffer from poor accuracy. Each framework makes different design choices for simulation models, thus achieving different trade-offs between scalability and accuracy. At one end of the spectrum are "microscopic models" that aim to capture most low-level behavior (e.g., packet-level network simulation, cycle-accurate CPU simulation) but as a result can suffer from high time- and space-complexity, which hinders scalability. At the other end are "macroscopic models" that abstract away most low-level behavior, typically via mathematical models and can thus suffer from poor accuracy.

The SimGrid project (H. Casanova et al., 2014) has placed a large emphasis on both simulation accuracy and simulation scalability, which have been achieved via state-of-the-art macroscopic models that have been validated extensively and implemented efficiently. These models are sufficiently scalable that complex simulations can be executed relatively quickly on a single core. SimGrid provides foundational simulation abstractions for building simulators for a broad

[*]Co-first author

range of applications, runtime systems, and hardware platforms. Essentially, it provides the necessary abstractions for simulating the execution sequential processes that run on compute resources and that can communicate by exchanging messages over network paths. While widely applicable, these abstractions are low-level. As a result, implementing SimGrid simulators for complex scenarios can be labor-intensive. Furthermore, many SimGrid users end up re-implementing essentially the same simulation components. There is thus a clear need for higher-level simulation abstractions that make it straightforward and low-labor to implement simulators of complex scenarios, but that build on SimGrid's low-level abstractions so as to achieve simulation accuracy and scalability. The WRENCH project (Henri Casanova et al., 2020) was established to answer this need.

Design and APIs

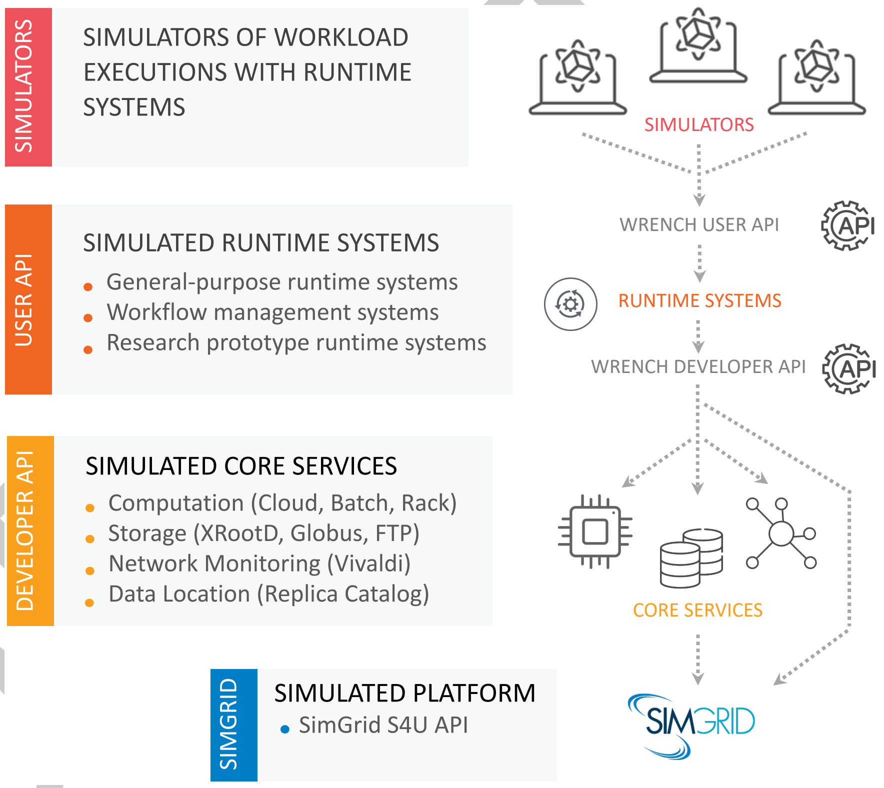
SIMULATORS
SIMULATORS OF WORKLOAD EXECUTIONS WITH RUNTIME SYSTEMS
USER API
SIMULATED RUNTIME SYSTEMS
General-purpose runtime systems
Workflow management systems
Research prototype runtime systems
DEVELOPER API
SIMULATED CORE SERVICES
Computation (Cloud, Batch, Rack)
Storage (XRootD, Globus, FTP)
Network Monitoring (Vivaldi)
Data Location (Replica Catalog)
SIMGRID
SIMULATED PLATFORM
SimGrid S4U API
SIMULATORS
WRENCH USER API
API
RUNTIME SYSTEMS
WRENCH DEVELOPER API
API
CORE SERVICES
SIMGRID

Figure 1: WRENCH Design and APIs.

Figure 1 depicts the overall WRENCH software architecture. At the bottom is SimGrid, which, as explained in the previous section, provides all foundational simulation abstractions as well as ways to describe a physical platform to simulate. Using SimGrid's S4U API, WRENCH implements a number of simulated "core services." These correspond to software services that are commonly deployed on hardware resources to make it possible to use these resources for performance computation, storing and serving data, monitoring network load, keeping track of data replicas, etc. A simulator specifies which particular services are deployed on which simulated hardware resources, and configure the behavior of these services. For instance, the simulated platform could be comprised of a cluster, and the simulator could specify that this cluster is managed by a batch service that provides access to the compute nodes in the cluster using some batch scheduling algorithm. WRENCH then provides an API, called the "Developer API" by which a user can implement a simulated runtime system that executes an application workload using the core services deployed on the hardware platform. WRENCH was initially designed specifically to simulate the execution of scientific workflow applications. The Developer API in WRENCH v2.0 has been made more general-purpose and the application workload can consist of arbitrary sets of independent and/or dependent actions to be performed on the deployed services. Finally, using the "User API", in a few lines of code one can initialize the simulated execution of a workflow with a runtime system using deployed services on the hardware platform, launch the simulation, and process the simulation output (which consists of time-stamped simulation event traces).

Software

WRENCH v2.0 was released in April 2022. The code is hosted on GitHub at (H. Casanova et al., 2020) and consists of C++ libraries, tool programs, and example programs. All information regarding the project including documentation is at https://wrench-project.org/. Note that WRENCH is not only used for Computer Science research purposes, but also for education as it provides the basis for the simulation-driven pedagogic modules for the EduWRENCH project. WRENCH has been actively developed for over 4 years with a software release every 4 months on average. Unit testing is implemented using Google Tests and code coverage has been between 80% and 90% for several years. Continuous Integration is performed on every push to the master branch using Docker images for a variety of OS and compiler versions. The main current development effort targets the implementation of a simulation REST API on top of which thin APIs for programming languages other than C++ can be quickly developed, starting with Python3.

Acknowledgments

This work is currently funded by NSF awards 2106059, 2106147, 2103489, and 2103508.

References

Buyya, R., & Murshed, M. (2002). GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *Concurrency and Computation: Practice and Experience*, *14*(13-15), 1175–1220.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, *41*(1), 23–50.

Carothers, C. D., Bauer, D., & Pearce, S. (2000). ROSS: A High-Performance, Low Memory, Modular Time Warp System. *Proc. Of the 14th ACM/IEEE/SCS Workshop of Parallel on Distributed Simulation*, 53–60.

Casanova, Henri, Ferreira da Silva, R., Tanaka, R., Pandey, S., Jethwani, G., Koch, W., Albrecht, S., Oeth, J., & Suter, F. (2020). Developing Accurate and Scalable Simulators of Production Workflow Management Systems with WRENCH. *Future Generation Computer Systems*, *112*, 162–175.

Casanova, H., Giersch, A., Legrand, A., Qinson, M., & Suter, F. (2014). Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing*, *75*(10), 2899–2917.

Casanova, H., McDonald, J., & Ferreira da Silva, R. (2020). WRENCH: Cyberinfrastructure simulation workbench. In *GitHub repository*. GitHub. https://github.com/wrench-project/wrench

Hoefler, T., Schneider, T., & Lumsdaine, A. (2010). LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model. *Proc. Of the ACM Workshop on Large-Scale System* *and Application Performance*, 597–604.

Kecskemeti, G. (2015). DISSECT-CF: A simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, *58*(2), 188–218.

Malik, A. W., Bilal, K., Aziz, K., Kliazovich, D., Ghani, N., Khan, S. U., & Buyya, R. (2014). CloudNetSim++: A toolkit for data center simulations in OMNET++. *Proc. Of the 2014 11th Annual High Capacity Optical Networks and Emerging/Enabling Technologies (Photonics for Energy)*, 104–108.

Núñez, A., Vázquez-Poletti, J., Caminero, A., Carretero, J., & Llorente, I. M. (2011). Design of a New Cloud Computing Simulation Platform. *Proc. Of the 11th Intl. Conf. On Computational Science and Its Applications*, 582–593.

Qayyum, T., Malik, A. W., Khan Khattak, M. A., Khalid, O., & Khan, S. U. (2018). FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment. *IEEE Access*, *6*, 63570–63583.

Tikir, M., Laurenzano, M., Carrington, L., & Snavely, A. (2009). PSINS: An Open Source Event Tracer and Execution Simulator for MPI Applications. *Proc. Of the 15th Intl. Euro-Par Conf. On Parallel Processing*, 135–148.