Automated Calibration of a Simulator of MPI Application Executions

Yick Ching Wong*, Frédéric Suter†, Kshitij Mehta†, Henri Casanova* and Jesse McDonald*
* University of Hawai'i at Mānoa, Honolulu, HI, USA
email: {wongy, henric, jamcd}@hawaii.edu
† Oak Ridge National Laboratory, Oak Ridge, TN, USA
email: {suterf, mehtakv}@ornl.gov

***Abstract*—The traditional approach for assessing the performance of scientific applications on HPC platforms consists in executing these applications on these platforms. But conducting these real-world experiments comes with several difficulties. Besides being often time-, labor-, and resource-intensive, experiments are limited to application and platform configurations at hand, thus precluding the exploration of "what if?" scenarios. A way to resolve these difficulties is to resort to simulation. The main concern, then, is that of simulation accuracy. For a simulation to be accurate, the parameters that define the behavior of the simulation models can be calibrated with respect to ground-truth executions. Simulation calibration, in the current state of the art, relies, at best, on labor-intensive manual procedures. We propose an automated simulation calibration approach, and apply this approach to the specific context of the simulation of MPI applications on leadership class HPC platforms. This poster will motivate the development of this approach and detail our methodology and results.**

I. Introduction

High-Performance Computing (HPC) applications are key to tackling scientific problems, the solutions to which enable groundbreaking research and innovation. It is crucial for the performance behaviors of these applications on HPC platforms be well understood to identify bottlenecks, guide resource provisioning decisions, application configuration decisions, and the coordination of application execution campaigns. Due the current and increasing complexity of modern HPC platforms and applications, the most reliable approach to conduct performance assessments is to run real applications on real platforms. Unfortunately, this approach faces several challenges. First, real-world experiments are typically time-, labor-, and resource-intensive, which can prevent comprehensive performance assessment. Second, these experiments are subject to platform "noise" and can be impacted by platform unavailability periods and reconfigurations/upgrades. Third, real-world executions are not always perfectly observable due to limits of what logging features are implemented in the applications and/or deployed on the platform. Fourth, and perhaps most important, the scope of real-world experiments are limited to whatever application and platform configurations are available. That is, they cannot be used to explore "what if?" scenarios (e.g., running experiments at larger scales than that of the currently available platforms).

A way to obviate the above experimental challenges is to resort to simulation, as a replacement or in addition to real-world experiments for conducting performance assessment. Simulation experiments are perfectly repeatable and observable, are almost always less time-, labor-, and resource-intensive than real-world experiments, and can be conducted for arbitrary application and platform settings. The main concern with simulation is accuracy. A simulation implements simulation models that abstract the performance behavior of software and hardware components of the real-world system. These simulation models, to allow for versatility, come with configuration parameters that influence their behaviors. The accuracy of simulation results, i.e., whether simulated executions are in line with real-world executions, depends on the chosen values for these simulation model parameters. Choosing these values is not easy because many parameters do not map directly to the hardware and software characteristics of the components of the real-world system due to the abstraction inherent in the models. The way to choose good parameter values is to *calibrate* simulation parameters with respect to ground-truth real-world executions. An exploration of the simulation calibration state of the art, in the field of parallel and distributed computing, reveals that simulation parameter calibration is often not done, or done via labor-intensive, mostly manual procedures [1].

In this work we develop a method for automated simulation calibration, and apply this method in the context of MPI application. Specifically, we focus on the SMPI simulator [2], for simulating executions on the Summit supercomputer at the Oak Ridge Leadership Computing Facility (OLCF), using ground-truth execution data obtained by executing the Intel MPI benchmarks on this platform. Details on our methodology are given in the next section.

II. Methodology

To investigate the feasibility and usefulness of automated simulation calibration, we require ground-truth data from the execution of MPI applications or benchmarks, a simulator of MPI application executions, and a calibrator that automatically

discovers good values for the configuration parameters of the simulator. We detail these three components hereafter:

Ground-Truth Data – For this work, we have chosen the Intel® MPI benchmark suite, a set of benchmark MPI programs that measures the performance of point-to-point and global communication operations. We have opted for using these benchmarks for our ground-truth data as they are freely available, can be executed at arbitrary scales and for arbitrary message sizes, and are straightforward to install and run on any platform. As a result, they can be used to obtain large and diverse ground-truth datasets. Our main hypothesis is that these datasets sufficiently capture communication performance behaviors for calibrating our simulator so that it can accurately simulate the execution of arbitrary MPI applications. We have obtained a large ground-truth dataset, for both point-to-point and collective communications, by running these benchmarks on the Summit platform at the Oak Ridge Leadership Computing Facility (OLCF).

Simulator – We use the SMPI simulator [2] provided as part of the popular SimGrid [3] simulation framework, which implements accurate and scalable foundational simulation models. SMPI allows for the simulation of unmodified MPI applications, such as the above MPI benchmarks, on arbitrary platform configurations. SimGrid also provides a programmatic API to specify simulation compute platforms, which has allowed us to describe Summit in under 80 lines of C++ code. The MPI benchmarks repeat many iterations of the same communication operations because real-world platforms are subject to "noise", making performance experiments not completely deterministic and reproducible. As a result, these benchmarks run for a significant amount of time. Although SMPI is relatively fast, to perform calibration we need to simulate possibly enormous numbers of benchmark executions (to search for the best simulation parameter values in a large, multi-dimensional parameter space). Because simulated benchmark executions are perfectly deterministic, unlike their real-world counterparts, we can obtain valid (simulated) performance results with only a few iterations. We have thus modified the benchmark implementation to detect when performance measures have converged to a fixed value and terminate without completing any further iterations.

Calibrator – We use the SimCal (code on GitHub [4]) simulator calibration framework. It provides a Python API for specifying simulation parameter spaces to invoke any arbitrary simulator configured by these parameters and to define arbitrary simulation accuracy metrics as loss functions. It implements several calibration algorithms that attempt to minimize the loss function over the parameter space: grid search, random search, gradient descent, and Bayesian optimization. Using SimCal has made it possible for us to implement our simulation calibration procedure in under 200 lines of Python, the goal being to calibrate 29 parameters: 21 parameters that configure the behavior of the SMPI simulation kernel; 6 parameters that specify the hardware characteristics of compute nodes in the simulated platform; and 2 parameters that specify the hardware characteristics of the interconnect. We define our loss function as the explained variance between the real-world and the simulated data transfer rates in MB/sec.

Given the above three components, we can now evaluate the effectiveness of automated simulation calibration. At the time of writing, evaluation experiments are underway, according to the following evaluation plan:

1) The initial valuation focuses on a subset of the MPI benchmarks, namely the *IMB-P2P* point-to-point communication benchmarks, which includes shared memory transport: PingPing, PingPong, Birandom, Corandom, SendRecv_Replace, Unirandom, Stencil2D, Stencil3D. We will execute these benchmarks on messages size ranging from 1B to 4MB.
2) A subset of the ground-truth benchmark executions (Stencil2D, Stencil3D) will be held back as a validation set to verify the accuracy of the automatically calibrated simulator when used to simulate scenarios that were not included in the calibration process.
3) We will compare the quality of the calibration when computed with the different algorithms described above, as well as when determined by a human expert.
4) There is an intriguing tension between the amount of ground-truth used to compute the calibration and the ability to explore the parameter space. More ground-truth data is desirable to ensure that the obtained calibration is generalizable. But more ground-truth data also increases the loss function evaluation time (due to the need to run more simulations for this evaluation), which hinders the parameter space exploration. As a result, given a time budget to compute the calibration, the question of how much ground-truth data should be used is not straightforward. We will explore this question by performing calibration with different amounts of ground-truth data for our particular use case.

The results from the above evaluation will yield not only a calibrated simulator for our particular use case on Summit, but also findings regarding the amount of ground-truth data and the calibration algorithms that are the most effective.

ACKNOWLEDGEMENTS

This research is partially supported by Laboratory Directed Research and Development Strategic Hire funding No. 11134 from Oak Ridge National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy.

REFERENCES

[1] J. McDonald, M. Horzela, F. Suter, and H. Casanova, "Automated Calibration of Parallel and Distributed Computing Simulators: A Case Study," in *Proc. of the 25th IEEE Int. Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC)*, 2024. [Online]. Available: https://arxiv.org/abs/2403.13918

[2] A. Degomme, A. Legrand, G. Markomanolis, M. Quinson, M. Stillwell, and F. Suter, "Simulating MPI applications: the SMPI approach," *IEEE TPDS*, vol. 18, no. 8, pp. 2387–2400, 2017.

[3] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *JPDC*, vol. 74, no. 10, pp. 2899 – 2917, 2014.

[4] "The SimCal simulator calibration framework," https://github.com/wrench-project/Grand-Unified-Calibration-Framework, 2024.