A Methodology for Bridging the Native and Simulated Executions of Parallel Applications

University of Basel

Ali Mohammed, Ahmed Eleliemy, and Florina. M. Ciorba
Department of Mathematics and Computer Science, University of Basel, Switzerland
{ali.mohammed, ahmed.eleliemy, florina.ciorba}@unibas.ch

1. Introduction

Problem statement

- Bridging native and simulated executions refers to creating a connection between two pillars of science, i.e., experimentation and simulation
- Many native experiments are needed in the process of optimizing the performance of an application on a high performance computing (HPC) system
- Native experiments can be time consuming, not feasible due to hardware unavailability, and it is very hard to control all the parameters that affect the performance
- Simulation uses abstractions and can alleviate certain limitations encountered in native experimentation

Advantages of the proposed bridging methodology described in Section 4 are

- It separates the concerns in the representation and the verification of the application software and of the computing hardware
- It reduces the effort needed to bridge the native and simulated executions of a parallel application on an HPC system

2. Native and Simulated Experimental Setups

1. Parallel Application

- The parallel spin-image algorithm (PSIA) [3] is a parallel version of the spin-image algorithm (SIA) [4]
- SIA applications: 3D object recognition, categorization, and 3D face recognition [2]
- SIA converts a 3D object to a set of 2D images which describes that 3D object
- The time to generate a single spin-image is not constant per and among workers and depends on the input data
- Variability of spin-image generation time results in unbalanced load among worker processes and uneven processors' finishing times

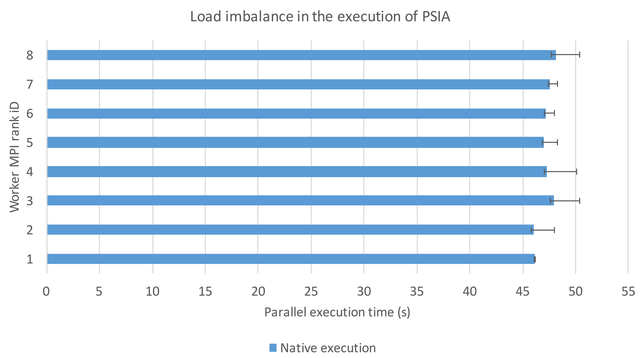

Figure 1: The performance of the parallel execution of PSIA for generating 8,000 spin-images using eight workers and one master

2. Native Computing System Hardware

- Processor: Intel Xeon Broadwell E5-2640 v4
- Main memory: 64 GB DDR4
- Network: single level fat tree OmniPath interconnection fabric

3. Experimental Setup

a. Native Experiments

- Each MPI rank is pinned to a single core, on an individual compute node
- PSIA generates 8,000 spin-images of the Ramesses object [5]
- Generation of spin-images is equally distributed using straightforward parallelization by the master among eight worker MPI ranks
- The native execution of the PSIA is repeated 20 times
- The average difference between the ranks execution time and the average ranks execution time for all the 20 runs is 3.7%

b. Simulated Experiments

- The native experimental setup is reproduced in the simulated experimental setting

3. The Use of SimGrid Simulation Toolkit

1. SimGrid is a framework to simulate distributed computer systems [1]
2. SimGrid has three user interfaces
   - MSG represents concurrent processes and simulates distributed applications
   - SMPI simulates the execution of MPI programs on a simulated computing system
   - SimDag represents tasks with dependencies (data or control) represented as directed acyclic graphs (DAG)
3. SimGrid requires a `platform file` to represent the characteristics of the computing system

4. Bridging Approach

4.1. Expression of Applications in SimGrid (steps)

1. Representation of the computing system in simulation as a `platform file`
2. Simulation of the application using *SimGrid-SMPI* and the `platform file` that represents the computing system
3. Generation of a time independent trace (TiT) of the application execution using *SimGrid-SMPI*
4. Analysis of the TiT and comparison against the application source code
5. Creation of the *SimGrid-SimDag* computation and communication tasks that represent the application using the information obtained from the TiT

4.2. Verification of Computing Systems Representation (steps)

1. Execution of the application natively and reporting its performance
2. Simulation of the application using *SimGrid-SMPI* and the `platform file` that represents the computing system
3. Comparison of the performance of native and simulated executions
4. Tuning the representation of the computing system in the `platform file`
5. Using *SimGrid-SMPI* calibration tool to obtain the calibration factors for the simulated network and MPI functions

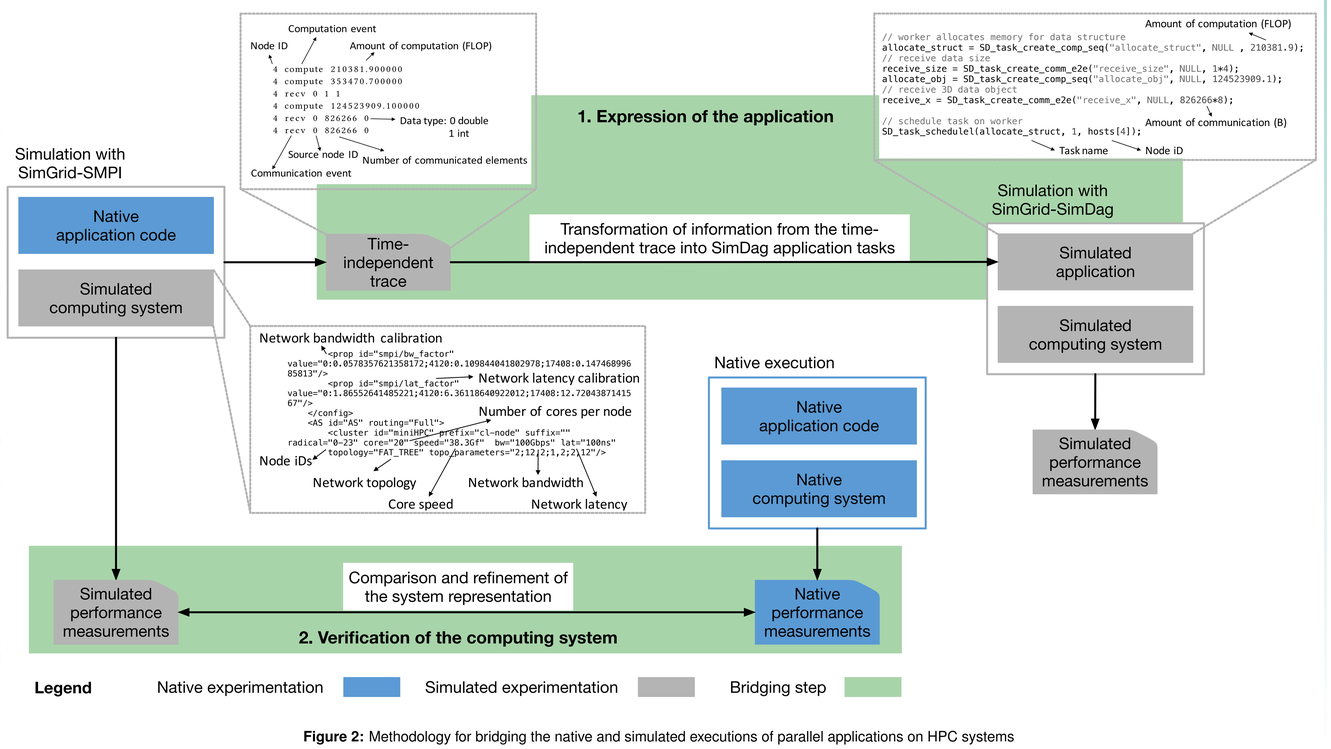

Figure 2: Methodology for bridging the native and simulated executions of parallel applications on HPC systems

5. Results of Native and Simulated Executions

- The simulation of the PSIA using *SimGrid-SMPI* is repeated 20 times and the simulated execution time per worker MPI rank is reported. The average error between all the 20 runs is 1.3%

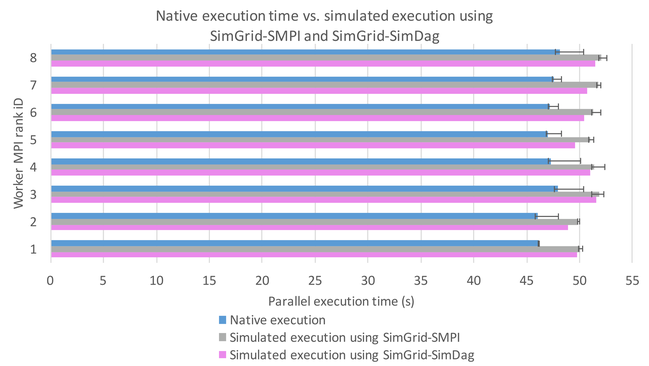

Figure 3: Comparison between the native execution time of PSIA and its simulated execution time using *SimGrid-SMPI* and *SimGrid-SimDag*. Error bars represent the differences between the native and simulated execution times of 20 runs for each experiment.

- Relative percentage differences between native and simulated results are calculated using

$$\text{relative percentage difference} = \left(\frac{\text{simulated execution time}}{\text{native execution time}} - 1\right) \times 100\%$$

Table 1: Relative percentage difference between the native execution time of the PSIA and the simulated execution time using *SimGrid-SMPI* and *SimGrid-SimDag*

Simulation	Minimum	Maximum	Average
SimGrid-SMPI	8.1%	8.7%	8.5%
SimGrid-SimDag	5.6%	7.9%	6.9%

6. Conclusion and Future Work

- The intermediate step of simulation using the *SimGrid-SMPI*, separates the concerns in the verification of software and hardware representation and reduces the uncertainties incurred in the process of representing the application execution
- Information from time-independent traces, generated with *SimGrid-SMPI*, is used in the representation of the application flow and the size of its tasks in the *SimGrid-SimDag* simulation
- The proposed methodology is a first step towards automating the process of bridging the native execution of parallel applications on HPC systems with their corresponding simulated execution
- The verified simulated experiments can be used to evaluate the performance of different scheduling methods in simulation

7. Reproducibility of This Work

The information on the software and the hardware of the computing systems used in this work is summarized below

Table 2: Native and simulated platforms' characteristics

Software		Hardware	
Simulation toolkit	SimGrid v.3.15 [1]	Processor	Intel(R) Xeon(R) CPU E5-2640 v4
Compiler	GNU GCC v. 6.3	Number of cores	20
MPI	OpenMPI v.2.0.2	Main memory	64 GB DDR4
Compilation flag	-O3	Topology	Single level fat tree
Operating system	CentOS Linux release 7.2.1511	Network	Intel OmniPath interconnect fabric

The raw results, scripts to run experiments, and the source codes of the application and the simulation can be accessed by scanning the QR code

References

[1] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, 2014. Versatile, scalable, and accurate simulation of distributed applications and platforms. Journal of Parallel and Distributed Computing, 74, 10 (2014), 2899-2917.
[2] K.-S. Choi and D.-H. Kim, 2013. Angular-partitioned spin image descriptor for robust 3D facial landmark detection. Electronics Letters, 49, 23 (2013), 1454-1455.
[3] A. Eleliemy, M. Fayze, R. Mehmood, I. Katib, and N. Aljohani, 2016. Loadbalancing on Parallel Heterogeneous Architectures: Spin-image Algorithm on CPU and MIC. In Proceedings of the 9th EUROSIM Congress on Modeling and Simulation, 623-628.
[4] A. E. Johnson, 1997. Spin-Images: A Representation for 3-D Surface Matching. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
[5] K. Wang, G. Lavoué, F. Denis, A. Baskurt, and X. He, 2010. A benchmark for 3D mesh watermarking. In Proceedings of the 9th IEEE International Conference on Shape Modeling and Applications. 231-235.

Acknowledgment

This work is in part supported by the Swiss National Science Foundation in the context of the "Multi-level Scheduling in Large Scale High Performance Computers" (MLS) grant, number 169123