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

Ali Mohammed
Department of Mathematics and Computer Science
University of Basel, Switzerland
ali.mohammed@unibas.ch

Ahmed Eleliemy
Department of Mathematics and Computer Science
University of Basel, Switzerland
ahmed.eleliemy@unibas.ch

Florina M. Ciorba
Department of Mathematics and Computer Science
University of Basel, Switzerland
florina.ciorba@unibas.ch

ABSTRACT

Simulation is considered as the third pillar of science, following experimentation and theory. Bridging the native and simulated executions of parallel applications is needed for attaining trustworthiness in simulation results. Yet, bridging the native and simulated executions of parallel applications is challenging. This work proposes a methodology for bridging the native and simulated executions of message passing parallel applications on high performance computing (HPC) systems in two steps: Expression of the software characteristics, and representation and verification of the hardware characteristics in the simulation. This work exploits the capabilities of the SimGrid [3] simulation toolkit's interfaces to reduce the effort of bridging the native and simulated executions of a parallel application on an HPC system. For an application from computer vision, the simulation of its parallel execution using straightforward parallelization on an HPC cluster approaches the native performance with a minimum relative percentage difference of 5.6%.

KEYWORDS

Native execution, simulated execution, performance, scheduling

1 INTRODUCTION

Bridging the native and simulated execution of parallel applications denotes creating a connection between the two scientific approaches, in the sense that each instance (native or simulated) is representative of the other. If a parameter has a certain effect in native execution, it should have the same effect in the simulated execution. Bridging the native and simulated execution of parallel applications is important as it connects two pillars of science, i.e., experimentation and simulation. Bridging native and simulated execution allows the acceptance of the simulation results with a higher degree of confidence than otherwise. This work proposes a methodology for bridging the native and simulated execution of parallel applications on high performance computing (HPC) systems.

In this work, the parallel spin-image algorithm (PSIA) [5] is considered as the application under study. The PSIA is a parallel version of the spin-image algorithm (SIA) [6]. It converts a 3D object representation to a set of 2D images considered as shape descriptor for that object. The SIA is used in 3D object recognition, categorization, and 3D face recognition [4]. In the PSIA, the generation of spin-images is equally distributed among the available parallel processors, i.e., each processor is assigned a certain number of spin-images to generate. The time to generate a single spin-image is not constant per and among processors. Therefore, dynamic loop scheduling (DLS) techniques are needed to balance the generation of spin-images among the available parallel processors and enhance application performance.

An overview of the most efficient DLS techniques is given in [1]. Experimenting with the use of all the available DLS techniques and examining the effects of using each method on the application performance using native execution may not be feasible, due to the following limitations: (1) Application execution time can be very large due to the problem size and (2) It is not feasible to control all the parameters that affect the execution time in native experiments.

Simulation can alleviate certain limitations encountered in native experimentation. To enable simulated experimentation, the SimGrid [3] simulation toolkit is used to represent the simulated performance of the PSIA application on an HPC system. SimGrid is a framework to simulate distributed systems with the following user interfaces: *MSG*, *SimDag*, and *SMPI*. The use of *SimDag* and *SMPI* in this work is described in Section 2.

2 BRIDGING APPROACH

In this work, a methodology for bridging the native and simulated executions of parallel applications onto HPC systems is proposed, which involves two steps: (1) Expression of applications; and (2) The representation and verification of computing systems. The advantages of the proposed methodology are the reduction of the complexity of the representation and verification process and the separation of the concerns between software and hardware representation. The separation of concerns leads to the ease of identifying the source of inaccurate simulation results and, hence, places more effort into improving the software and hardware representation.

2.1 Expression of Applications in SimGrid

To enable the simulation of the application of interest on the system represented in the platform file, the application characteristics need to be expressed in *SimGrid-SimDag* or *SimGrid-MSG*. The *SimGrid-SimDag* interface has been selected in this work as it has two simple types of tasks that can represent the PSIA application: sequential computation tasks (FLOP) and end to end communication tasks (B). To describe the application in the *SimGrid-SimDag* interface, the *SimGrid-SMPI* interface is used to produce a special type of trace of the application's simulated execution, i.e., time independent trace (TiT). In a TiT, the size of each computation or communication event is specified in FLOP or number of communicated elements, respectively. As shown in the first step in Figure 1, the information from the TiT is used to express the application in the *SimGrid-SimDag* and obtain the proper values of each computation and communication tasks.

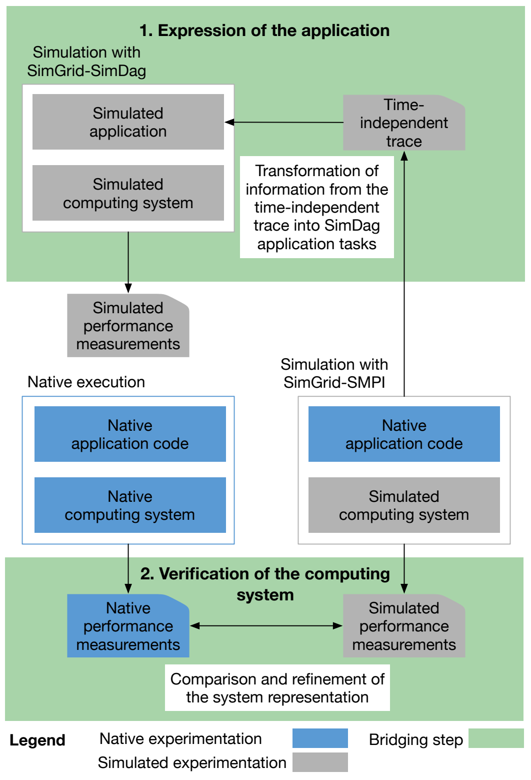
1. Expression of the application
Simulation with SimGrid-SimDag
Simulated application
Simulated computing system
Time-independent trace
Transformation of information from the time-independent trace into SimDag application tasks
Simulated performance measurements
Native execution
Simulation with SimGrid-SMPI
Native application code
Native computing system
Native application code
Simulated computing system
2. Verification of the computing system
Native performance measurements
Simulated performance measurements
Comparison and refinement of the system representation
Legend
Native experimentation
Simulated experimentation
Bridging step

Figure 1: Methodology for bridging native and simulated executions of parallel applications on HPC systems.

2.2 Verification of Computing Systems Representation

The simulation of an unmodified parallel application using the *SimGrid-SMPI* interface requires a description of the computing system hardware in the platform file. By comparing the performance of the native execution of the application and the simulated performance with the *SimGrid-SMPI*, one can verify the representation of the computing system in the platform file, as shown in the second step in Figure 1. In addition, the *SimGrid-SMPI* calibration tool [2] is used to modify the network bandwidth and latency in the simulation according to message sizes and to add function call overheads for MPI send and receive functions.

3 EXPERIMENTS AND RESULTS

Experiments: The PSIA application is executed to generate 8, 000 spin-images to represent the Ramesses object described in [7]. Generating spin-images is equally distributed by a master MPI rank between eight worker MPI ranks. All ranks execute on separate compute nodes (Intel Xeon Broadwell E5-2640 v4 processor), interconnected via a single level fat tree (OmniPath fabric). The master rank is only responsible for distributing the work among the workers and does not generate spin-images. The same aforementioned native experimental setting is reproduced in the simulated experimental setting.

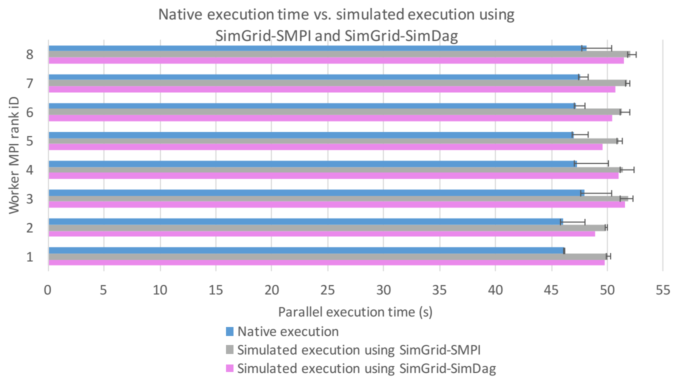

Figure 2: Comparison between the native execution time of PSIA and its simulated execution time using SimGrid-SMPI and SimGrid-SimDag. Execution with nine MPI ranks (one master and eight workers), using straightforward parallelization. Generation of spin-images is equally distributed by the master rank between the eight worker ranks. Error bars represent the differences between the native and simulated execution times of 20 runs for each experiment.

Results: The results of the native execution of the PSIA application are compared to the results of the simulated execution using *SimGrid-SMPI* and *SimGrid-SimDag* in Figure 2. The simulation results from both the *SimGrid-SMPI* and the *SimGrid-SimDag* follow the same trend of the native execution within certain differences. The minimum, the maximum, and the average relative percentage differences between the results of the native execution and the simulation using *SimGrid-SMPI* are 8.1%, 8.7%, and 8.5%, respectively. For the results of the *SimGrid-SimDag* simulation, the minimum, the maximum, and the average relative percentage differences are 5.6%, 7.9%, and 6.9%, respectively. These differences are attributed to the use of simulation abstractions that do not account for certain significant computing system properties, such as the memory subsystem.

4 CONCLUSION AND FUTURE WORK

In this work, a methodology for bridging the native and simulated executions of message-passing parallel applications on HPC systems is presented. The usage of *SimGrid-SMPI* to verify a computing system's representation in a platform file is introduced. The proposed methodology has the advantages of reducing the effort of bridging the native and simulated executions of parallel applications and separating the concerns in software and hardware representations.

A verified simulation of the application execution can be used to examine the effects of using different DLS techniques on the PSIA to enhance its performance on existing or future HPC systems.

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.

REFERENCES

[1] I. Banicescu and R. L. Cariño. 2005. Addressing the stochastic nature of scientific computations via dynamic loop scheduling. *Electronic Transactions on Numerical Analysis* 21 (2005), 66–80.

[2] P. Bedaride, A. Degomme, S. Genaud, A. Legrand, G. S. Markomanolis, M. Quinson, M. Stillwell, F. Suter, and B. Videau. 2013. Toward better simulation of MPI applications on Ethernet/TCP networks. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 158–181.

[3] 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.

[4] 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.

[5] 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 Modelling and Simulation*. 623–628.

[6] A. E. Johnson. 1997. *Spin-Images: A Representation for 3-D Surface Matching*. Ph.D. Dissertation. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[7] 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.