A HIGH-PERFORMANCE DEVS SIMULATOR FOR MULTI-GPU PLATFORMS

Guillermo G. Trabes

Department of Systems and Computer Engineering
Carleton University
and Universidad Nacional de San Luis
1125 Colonel By
Ottawa. ON, K1S 5B6, CANADA

ABSTRACT

As Discrete Event Systems Specification (DEVS) formalism is increasingly adopted across various fields, simulations become more complex and time-consuming. To improve the performance of DEVS simulations, we propose a parallel algorithm designed for modern high-performance platforms with multiple GPU devices. Our approach guarantees a simple, error-free parallel execution. We also present experimental results demonstrating that our approach can accelerate a real-world problem up 14.13X using one GPU and up to 22.89X using two GPUs, compared to sequential execution.

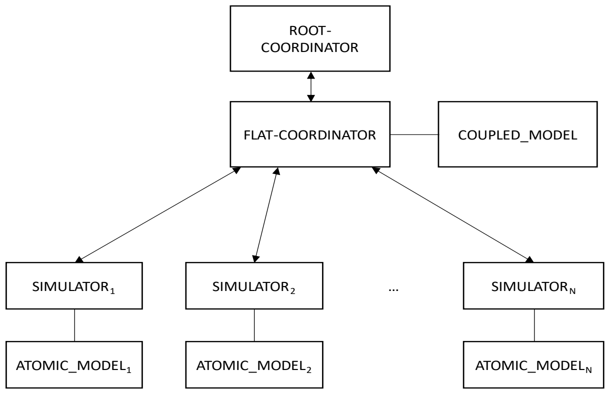
Keywords: simulation, DEVS, high-performance computing, GPU, Multi-GPU

1 INTRODUCTION AND BACKGROUND

Discrete Event System Specification (DEVS) (Zeigler et al. 2000) is a known mathematical formalism commonly used for modeling complex systems in a modular and hierarchical way. DEVS enables the modeling of problems by coupling atomic and coupled models, connecting them in a hierarchical structure. In DEVS, users combine these two types of models, coupled (providing structure) and atomics (providing behavior), to define their models, and there are general mechanisms that execute the simulations. To simplify the hierarchical structure, we can flatten it, resulting in a structure with only one coupled model that maintains all atomic models and couplings defined by the user. From there, we create a PDEVS Abstract Simulator structure, consisting of a root-coordinator at the top of the tree structure, one flat-coordinator component, and one simulator for each atomic model. To execute simulations on this structure, there is a well-known algorithm: the PDEVS Simulation Protocol. Although DEVS provides a general mechanism to execute simulations, when dealing with large and complex DEVS simulations, we need better techniques to execute them efficiently. In the past, several techniques have been proposed to execute DEVS simulations in parallel, including those from the Parallel and Distributed Simulation (PADS) field. However, only a few of these algorithms can execute correct simulations in all circumstances. In (Zeigler 2017) a new approach was proposed, that uses the PDEVS protocol's inherent parallelism in the execution of output and state transitions execution. This idea was extended with more parallelism in multicore architectures in Trabes et al (In press).

2 PARALLEL PDEVS SIMULATION PROTOCOL ON MULTI-GPUS

In this work we improved previous approaches on the parallel execution of the PDEVS simulation protocol by adapting it to modern high-performance platforms, composed by multicore central processing units (CPUs) and several graphics processing units (GPUs). Our algorithm parallelizes the execution assigning different functions to CPU threads, where each thread is responsible to execute on a different GPU. We assign to each thread a subset of the simulators and each their functions on a GPU. In Figure 1 we can see a summary of this algorithm, which repeats until the simulation finishes. After each task in the simulation,



1. Execute outputs : ROOT-COORDINATOR calls execute_output_functionson FLAT-COORDINATOR, which calls execute_outputs_function on every SIMULATOR subcomponent in parallel. Each thread executes on a subset of SIMULATORs. SIMULATOR executes its output function and stores it in its outbox.
2. Route messages: ROOT-COORDINATOR calls route_messages on FLAT-COORDINATOR, which routes messages among atomic models in parallel. Each thread routes messages to a subset of atomic models. SIMULATOR inserts output bag to input bag of component if not empty.
3. Execute transitions: ROOT-COORDINATOR calls execute_transition_functions on FLAT-COORDINATOR, which calls execute_transition function on every SIMULATOR subcomponent in parallel. Each thread executes on a subset of SIMULATORs. SIMULATOR checks for imminent component and empty inbox and executes corresponding transition function.
4. Obtain next event: ROOT-COORDINATOR calls next_time function on FLAT-COORDINATOR, which obtains next_time for each SIMULATOR subcomponent in parallel and determines minimum value through parallel reduction.

Figure 1: Parallel PDEVS Simulation Protocol.

the threads synchronize to guarantee the completion of each task before starting the new one. This way we achieve to deploy the algorithm on all GPUs in the system, and they all collaborate on the execution of the simulations. To evaluate our , we implemented a parallel version in C++ using the OpenMP and CUDA libraries. We experimented with an epidemiological problem (White et. al 2007) modeled with the Cell-DEVS formalism. We evaluated a model with one million cells and executed it increasing the number of simulation steps. The experiments were executed on platform with two NVIDIA 1660 GPUs. Our approach achieved a significant speedup of up to 14.13X with one GPU and up to 22.89X with two GPUs compared to the sequential version run on a dual Intel Xeon E5-2609V4 multicore CPU. In comparison, the multicore parallelization with 16 cores on the same computer provided only a 3.03X acceleration.

3 CONCLUSIONS AND FUTURE WORK

In this work we presented an algorithm to execute DEVS simulations in multi-GPU platforms. The experimental results show that simulations can execute several times faster than the sequential version and scale when using more GPUs. As future work, we plan to implement this extend this approach to execute on distributed computers with multi-GPU nodes.

ACKNOWLEDGMENTS

This research was conducted in collaboration with my PhD supervisors, Dr. Gabriel A. Wainer and Dr. Veronica Gil-Costa.

REFERENCES

Trabes, G. G, G. A. Wainer, and V. Gil-Costa. (In press). "A Parallel Algorithm to Accelerate DEVS Simulations in Shared Memory Architectures". IEEE Transactions on Parallel and Distributed Systems.

White, S. H., A. M. Del Rey, and G. R. Sánchez. (2007). Modeling Epidemics Using Cellular Automata. Applied mathematics and computation 186(1):193–202.

Zeigler, B. P., H. Praehofer, and T. Kim. 2000. Theory of Modeling and Simulation. 2nd ed. Orlando, FL, USA: Academic Press, Inc.

Zeigler, B. P. 2017. "Using the Parallel DEVS Protocol for General Robust Simulation with Near Optimal Performance". Computing in Science and Engineering 19(3):68-77.