Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.

A PARALLEL ALGORITHM TO EXECUTE DEVS SIMULATIONS IN SHARED MEMORY ARCHITECTURES

Guillermo G. Trabes

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Dr.
Ottawa, ON K1S 5B6, CANADA

ABSTRACT

As the Discrete Event Systems Specification (DEVS) formalism becomes more popular and used in more fields of application, simulations are more complex and time consuming. For this reason, we need to develop new ideas to execute them efficiently. In this work we propose a parallel algorithm to execute DEVS simulationsin shared memory architectures. Our approach guarantees a simple and error free parallel execution without adding much overhead. In addition, we perform an experimental evaluation showing how we can accelerate the execution several times faster with our approach.

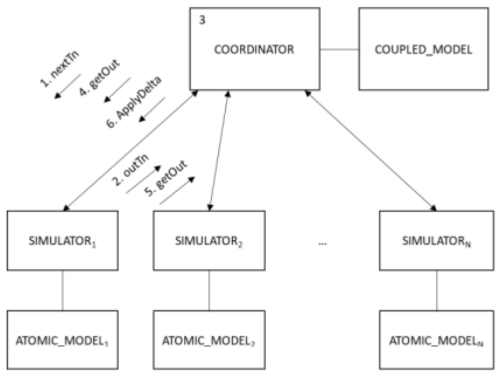
1 INTRODUCTION AND BACKGROUND

Discrete Event System Specification (DEVS) (Zeigler et al. 2000) is a well-known mathematical formalism that provides a theoretical framework to think about modeling using a hierarchical, modular approach. DEVS provides two types of models: atomics, that provide the model's behavior and coupled, that provide the model's structure. We can represent more complex problems by coupling the models.

In DEVS, users only define models and there are general mechanisms that execute the simulations. A structure known as PDEVS Abstract Simulator is generated, with one coordinator component for each coupled model and one simulator for each atomic model. To execute the PDEVS Abstract Simulator, there is a widely accepted mechanism called PDEVS Simulation Protocol. The steps presented in Figure 1 summarize this protocol and repeat until the simulation is over. Nevertheless, when dealing with big and complex DEVS simulations we need better techniques to execute them more efficiently. There have been several works published that utilize ideas from the Parallel and Distributed Simulation (PADS) field to execute DEVS simulations in parallel and improve performance. However, as stated in (Zeigler 2017) only a few proposed algorithms can execute correct simulations in all circumstances. For this reason, a new idea was proposed in (Zeigler 2017) to benefit from the intrinsic parallelism in the PDEVS simulation protocol in two circumstances: when more than one simulator 1) executes its output function and 2) executes its state transition. The theoretical results show how this approach can obtain good performance even compared to the ideal parallelization. The advantage of this approach is that it provides a simple parallelism without creating much overhead and guaranteeing correct simulation results.

2 PARALLEL SHARED MEMORY PDEVS SIMULATION PROTOCOL

In this work we propose a new approach intended for shared memory architectures. In these architectures, all message-passing between the coordinator and simulators is implemented as function calls and returns. Our algorithm divides the execution among threads and executes them on different processors in the architectures. We execute the PDEVS simulation protocol as follows: steps 1 and 2, are implemented as a function call and return respectively, and executed in parallel on every simulator. In step 3, the time for the next event is calculated in parallel. Each thread computes a partial minimum and then these results are compared to obtain the total minimum. Steps 4 and 5 are also implemented as a function call and return on every simulator and executed in parallel. Finally, step 6 is implemented as a function call without return value and are also executed in parallel on every simulator.



1. Coordinator sends a message to each simulator requesting the time for their next event.
2. Each simulator in the structure replies with the time for its next event.
3. Coordinator calculates the minimum time from all the times received.
4. Coordinator sends a message requesting for their output to each simulator and includes the minimum time in the message.
5. Each simulator checks if its minimum time is equal to the time received. If they are equal, the simulator calculates its output function and sends it to coordinator.
6. Coordinator sends a message requesting to apply their state transition to each simulator and includes the time calculated in step 3 in the message. In addition, if the simulator must receive an input value from any other component, these values are included in the message. Next, each simulator performs two checks: 1) if the time for its next event is equal to the time received; and 2) if is receiving any input value. According to these checks it will execute an internal, external, or confluent function; or it will not execute if has no events to compute.

Figure 1: PDEVS Simulation Protocol.

To evaluate our algorithm, we implemented a parallel version in C++ using the OpenMP library. We experimented with an epidemiological problem (White et. al 2007) modeled with the Cell-DEVS formalism. We vary the number of cells from 10.000 to 100.000 and executed for 500 simulation steps. The experiments were executed on two independent architectures: an 8-core i7-9700 processor and a 16-core machine with 2 8-core Xeon 2609V4 processors. The summary of the results can be seen in Table 1. The algorithm scales using more threads and the best results are obtained when using one thread per each core in both architectures.

Table 1: Result's Summary on an 8-core and 16-core architectures.

	Average speedup	
Number of threads	i7-9700	2 8-core 2609V4
2	1.91	1.82
4	3.60	3.44
8	5.89	6.35
16		8.00

3 CONCLUSIONS AND FUTURE WORK

In this work we presented an algorithm to execute DEVS simulations in parallel in shared memory architectures. The experimental results show that simulations can execute several times faster than the sequential version and scale when using more threads, up to the point when using all cores in the architectures. As future work, we plan to implement this algorithm on GPGPU architectures.

REFERENCES

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.