Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

LATENCY OPTIMIZED EXECUTION OF SEQUENTIAL SIMULATORS BY PARALLEL PARAMETER OPTIMIZATION

Till Köster
Nicola M. Drüeke
Adelinde M. Uhrmacher

Institute of Computer Science
University of Rostock
Albert-Einstein-Straße 22
18059 Rostock, GERMANY

ABSTRACT

Modern Hardware (even on small desktop systems) provides parallel execution capabilities, which are frequently not utilized by simulators. To increase the effective execution speed of a single simulation run, we will use these parallel hardware resources to execute different simulator configurations of the same model, explore the performance of different configurations per thread, and propagate superior performing configuration across parallel threads. We show the merit of this method in a small performance study.

1 Motivation

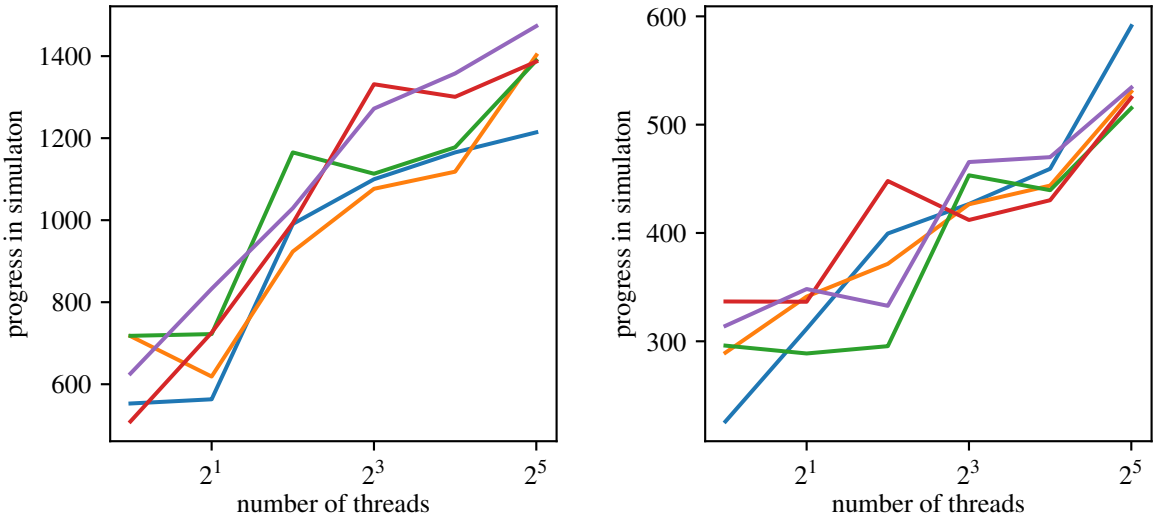
The parallelization of a simulation (including the synchronization of the different components) proved to be a challenging task. Especially with monolithic systems, parallization can be a very involved effort. In order to speed up the execution, an alternative is to optimize simulator parameter values such as the size of a buffer or the choice of a sorting algorithm (Helms 2017). We intend to combine those efforts. Specifically, each parallel thread runs its own method of finding optimal parameters.

One possibility to speed up simulation execution is to optimize simulator parameter values such as the size of a buffer or the choice of a sorting algorithm. In Helms (2017) reinforcement learning was used to select and configure simulation algorithms automatically. Here we intend to exploit parallel execution capabilities of modern hardware to identify the optimal simulation configuration for executing a single simulation run. Specifically, each parallel thread runs its own method of finding optimal parameters. One could employ any of the established methods here, like hill climbing or simulated annealing. During the execution we compare the progress of the threads several times and swap or replace parameter configurations. The simulation results are not affected by changing the parameters.

In the end, this is a rather wasteful use of resources and may increase power usage etc. However, for interactive applications, where not average throughput, but rather single simulation run performance is of interest (effectively a latency optimization problem), one might as well use those additional resources.

2 Parallel Optimization

For the initial testing of this approach, we utilized two basic optimization strategies on the single thread level. The advanced one employs hill climbing, whereas the naïve one simply uses different random configurations with no regard for their effectiveness. After a certain time, the more successful threads pass their configuration on to those threads that have less optimal ones. The objective of the exchange between the different threads is to spread a superior configuration among the other simulation runs.



(a) Advanced strategy.

(b) Naïve strategy.

Figure 1: Comparison of achieved results during the same wall-clock time for different configurations.

3 Results and Conclusion

For performance testing we used a system with 32 physical cores. The optimization was run with five representative values for the variables that control the number of comparisons. We used a simple dummy simulator with about 200 interdependent simulator parameters. As expected, the performance improves with the numbers of threads. In Figure 1 we can see the results. It can be noticed that the synchronization overhead decreases performance when moving from one to two threads, but with an increasing number of threads, the performance improves.

ACKNOWLEDGMENTS

This research is supported by the German Research Foundation (DFG) via research grants ESCeMMo (UH- 66/13).

REFERENCES

Helms, T. 2017. *Simulator Adaption at Runtime for Component-Based Simulation Software*. Ph. D. thesis, University of Rostock.