Simulating Batch and Application Level Scheduling Using GridSim and SimGrid

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

***Abstract*—Modern high performance computing (HPC) systems are increasing in the complexity of their design and in the levels of parallelism they offer. Studying and enhancing scheduling in HPC became very interesting for two main aspects. First, scheduling decisions are taken by different types of schedulers such as batch, application, process, and thread schedulers. Second, simulation has become an important tool to examine the design of HPC systems. Therefore, in this work, we study the simulation of different scheduling levels. We used two well-known simulation toolkits, SimGrid and GridSim, in order to support two different scheduling levels, batch and application level scheduling. Each toolkit is extended to support both levels. Moreover, three different scheduling algorithms for each level are implemented and their performance is examined through a real workload dataset. Finally, a comparison for the extension challenges of the two simulators is conducted.**

I. Introduction

Modern HPC systems offer parallelism at different levels starting from multiple cores at the CPU level to multiple computing nodes at the cluster level. Thus, there is a different type of scheduling associated with each of these levels. For instance, the batch level scheduling is associated with the cluster level whereas the application level scheduling is associated with the core level. It is important to study how different scheduling algorithms at different levels can cooperate and integrate to achieve higher utilization results. Given the complexity of HPC systems, simulation is an established method to examine and assess the design of different algorithms in HPC [1]. There is a wide set of parallel and distributed systems simulators [2], [3], [4], [5] which can be used to simulate various scheduling algorithms of different levels. However, to the best of our knowledge, there is no simulator that can support more than one level of scheduling at the same time. Hence, Simulators such as SimGrid [4] and GridSim [2] are used for application level scheduling and batch level scheduling, respectively. The main idea is to study the extension challenges to use single simulation toolkit to simulate different scheduling levels. This work is considered as preliminary step to how to extend the available simulation toolkits in such a way that they can be used to simulate different scheduling levels simultaneously. Our main contribution is to provide the necessary simulation toolkit to examine the effect of different combinations of scheduling algorithms at different levels on HPC systems utilization.
In this work, two well-known simulation toolkits have been selected: Alea (GridSim based simulator) [3] and SimGrid [4]. They are intended to simulate batch and application level scheduling, respectively.

II. Extended GridSim

GridSim [2] is a Java based simulation toolkit that enables users to simulate and model large scale distributed and parallel systems with different configurations. The ease of use and the reliable results of GridSim encourages many researchers to use it, hence there are many simulators that use the GridSim toolkit as its basis. Alea [3] is an instance of these simulators, Alea allows the study of advanced scheduling techniques for planning various types of jobs. it has been selected in this work due to two reasons. First, it implements a wide range of batch level scheduling algorithms such as first come first serve (FCFS), earliest deadline first (EDF), and shortest job first (SJF). Second, Alea has been successfully extended to support application level scheduling [6]. Compared to [6], this extension has three main advantages. First, it supports application level scheduling while the same binaries can still support batch level scheduling. Second, it is based on the latest version of Alea and GridSim. Third, all changes do not affect the Alea batch scheduling functionality.

III. Extended SimGrid

SimGrid [4] is a library that provides functionality to simulate large-scale distributed systems. It is fast, scalable, with small memory footprint, and simulation models are theoretically and experimentally assessed. Application level scheduling is simulated using SimGrid's SimDag interface assuming a master worker execution model. Our current implementation supports three dynamic loop scheduling algorithms: fixed size chunk(FSC), guided self-scheduling(GSS), and factoring(FAC).To simulate batch jobs, Simbatch [5] is an existing extension of SimGrid that extends MSG interface to support batch scheduling. However, we chose to build our extension upon SimDag to be comparable to the application level scheduling performance implemented in SimDag and also to provide support for jobs with dependencies in the future. Our extension of SimDag is built upon SimGrid v.3.13 library and it is designed to be simple and extensible. Currently our extension supports three BLS scheduling algorithms: FCFS, SJF, and EDF.

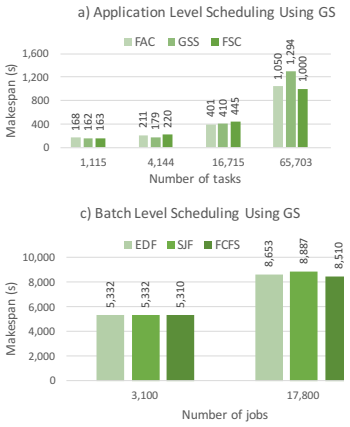

Fig. 1. Application and batch level scheduling makespan simulated by Alea/GridSim

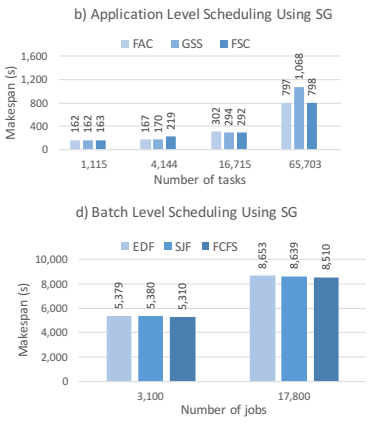

Fig. 2. Application and batch level scheduling makespan simulated by SimDag/SimGrid

IV. Experiments and Results

Two experiments have been designed to test our extension of GridSim and SimGrid. For the application level scheduling experiment, we used Lublin [7] to obtain tasks of four applications, each application contains 1115, 4144, 16715, 65703 tasks, respectively. All tasks are independent, with different execution length according to Lublin configuration. For batch level scheduling experiment, we used the same two job datasets provided by Alea in [3]. First dataset is obtained from High Performance Computing Center North (HPC2N) in Sweden, and it contains 3100 jobs. Second dataset is obtained from the Czech national Grid infrastructure MetaCentrum and it contains 17800 jobs. Figures 1 and 2 shows the makespan obtained by running the experiments on both simulators Alea and SimGrid, respectively.

Moreover to evaluate the efficiency of the used simulators, we have also measured the wall clock times of the simulators while running the experiments. Figure 3 shows the wall clock times of both simulators in simulating application and batch level scheduling for different problem sizes.

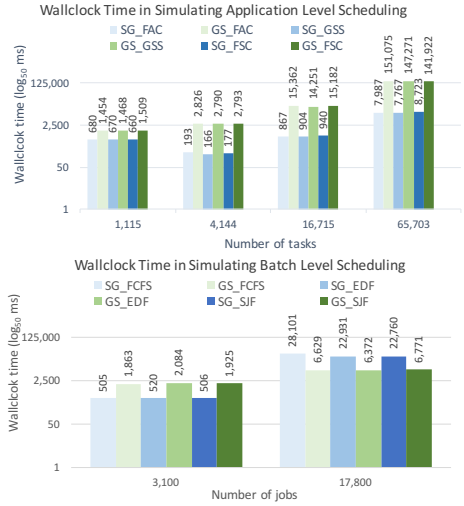

Fig. 3. Wall clock times of both simulators as problem size increase

V. Conclusion and Future Work

Both SimGrid and Alea were able to simulate both application and batch level scheduling algorithms and obtain comparable results. However, when it comes to scheduling large workloads on large scale computing systems SimGrid outperforms Alea with regards to simulator wall clock time on application level scheduling, whereas Alea outperforms SimGrid on batch level scheduling. The next step is to study how to enable simulation of these two scheduling levels using one of these successful simulator extensions.

References

[1] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale hpc network systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[2] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience CCPE*, vol. 14, no. 13, pp. 1175–1220, 2002.

[3] D. Klusáček and H. Rudová, "Alea 2 – job scheduling simulator," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools 2010)*. ICST, 2010.

[4] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899 – 2917, 2014.

[5] Y. Caniou and J. S. Gay, *Simbatch: An API for Simulating and Predicting the Performance of Parallel Resources Managed by Batch Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 223–234.

[6] S. Srivastava, I. Banicescu, F. M. Ciorba, and W. E. Nagel, "Enhancing the functionality of a gridsim-based scheduler for effective use with large-scale scientific applications," in *2011 10th International Symposium on Parallel and Distributed Computing*, July 2011, pp. 86–93.

[7] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: Modeling the characteristics of rigid jobs," *J. Parallel Distrib. Comput.*, vol. 63, no. 11, pp. 1105–1122, Nov. 2003.