Simulating Batch and Application Level Scheduling Using GridSim and SimGrid

University of Basel

Ahmed Eleliemy, Ali Mohammed, and Florina. M. Ciorba
{ahmed.eleliemy,ali.mohammed, florina.ciorba}@unibas.ch

1. Introduction

In an HPC system, a collection of scheduling mechanisms at different levels, such as batch, application, process, and thread level is used. Using a simulation toolkit to study the performance of such schedulers is very interesting and revealing for research on scheduling and load balancing in the context of parallel computing. The research goals of this work are:

- Investigating the feasibility of using one simulation toolkit to study scheduling at different levels.
- Studying different batch level scheduling (BLS) and application level scheduling (ALS) approaches using two specific simulation frameworks: GridSim [1] and SimGrid [2].

Table 1—Selected ALS and BLS algorithms

ALS algorithms	BLS algorithms
Factoring (FAC)	First Come First Serve (FCFS)
Fixed Size chunk (FSC)	Earliest Deadline First (EDF)
Guided Self Scheduling (GSS)	Shortest Job First (SJF)

2. Challenges/Changes

Table 2—Extension challenges of each simulator to support ALS and BLS

Challenges/Changes \ Simulators	GS*	SG**
Implementing BLS algorithms FCFS, EDF, and SJF	No	Yes
Implementing ALS algorithms FAC, FSC, and GSS	Yes	Yes
Extending a non object oriented code	No	Yes
Number of lines of code (added, removed, or modified)	≅ 350 Yes	≅ 700 Yes
Extending a multi-threaded library	Yes	No
Implementing support for reading standard workload format (SWF) files	No	Yes

*GS: The extended Alea simulator based on GridSim library.
**SG: The extended simulator based on the SimDag interface of the SimGrid library.
Yes: Required an extension change and represents a challenge.
No: Did not require a change of the orginal code and does not represent a challenge.

3. Simulation Scenarios

- **Datasets for Application Level Scheduling**
  Lublin [3] has been used to generate tasks for four applications, each containing 1,115, 4,144, 16,715, and 65,703 tasks, respectively. All tasks are independent, with different execution lengths according to the Lublin configuration as used in [5].
- **Datasets for Batch Level Scheduling**
  The two job datasets provided in [4] were used in this work. The first dataset is obtained from the High Performance Computing Center North (HPC2N) in Sweden, and contains 3,100 jobs. The second dataset is obtained from the Czech National Grid Infrastructure(NGI) MetaCentrum and contains 17,800 jobs.
- **Simulated platforms**
  - In case of ALS algorithms, the platform consists of 128 single core nodes, with 1 FLOP/s core speed
  - In case of BLS algorithms, the platform consists of 240 single core nodes, with 1 FLOP/s core speed

4. Extending GridSim/Alea with support for ALS

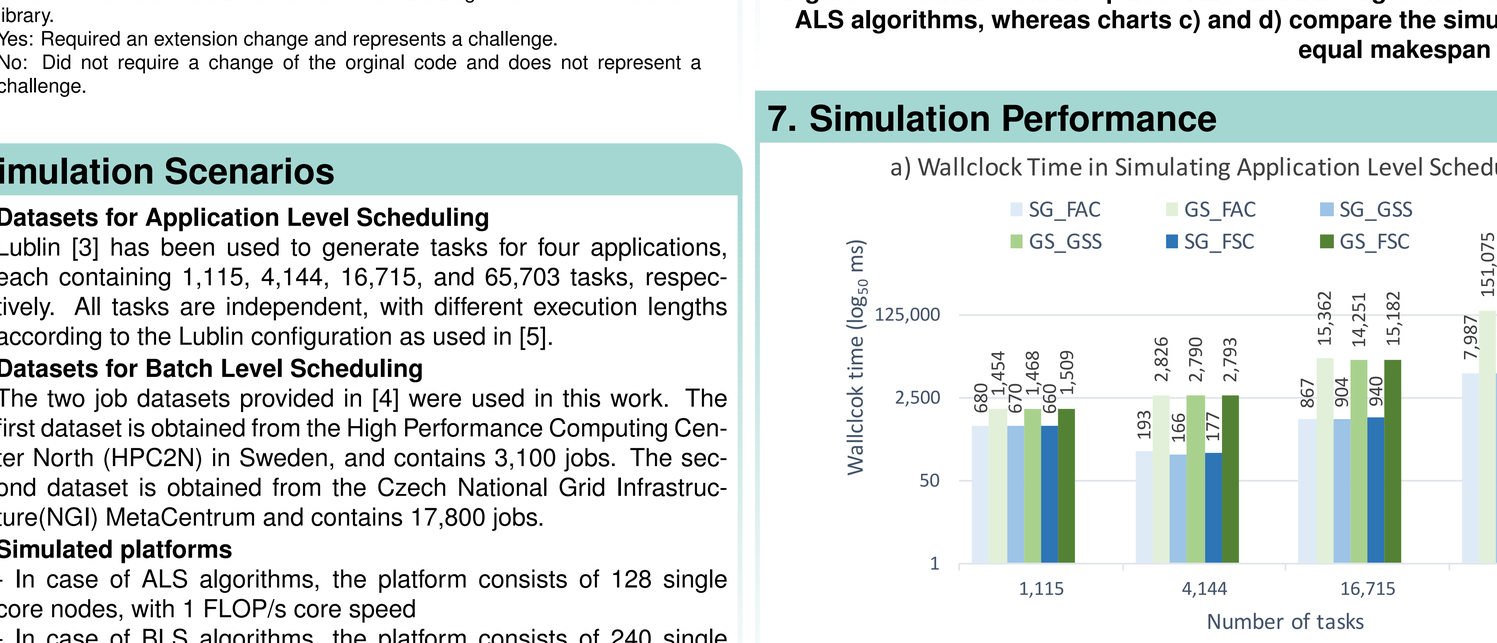

Figure 1—The execution workflow of GridSim/Alea and the extensions of this work to support ALS algorithms

5. Extending SimGrid/SimDag with support for BLS

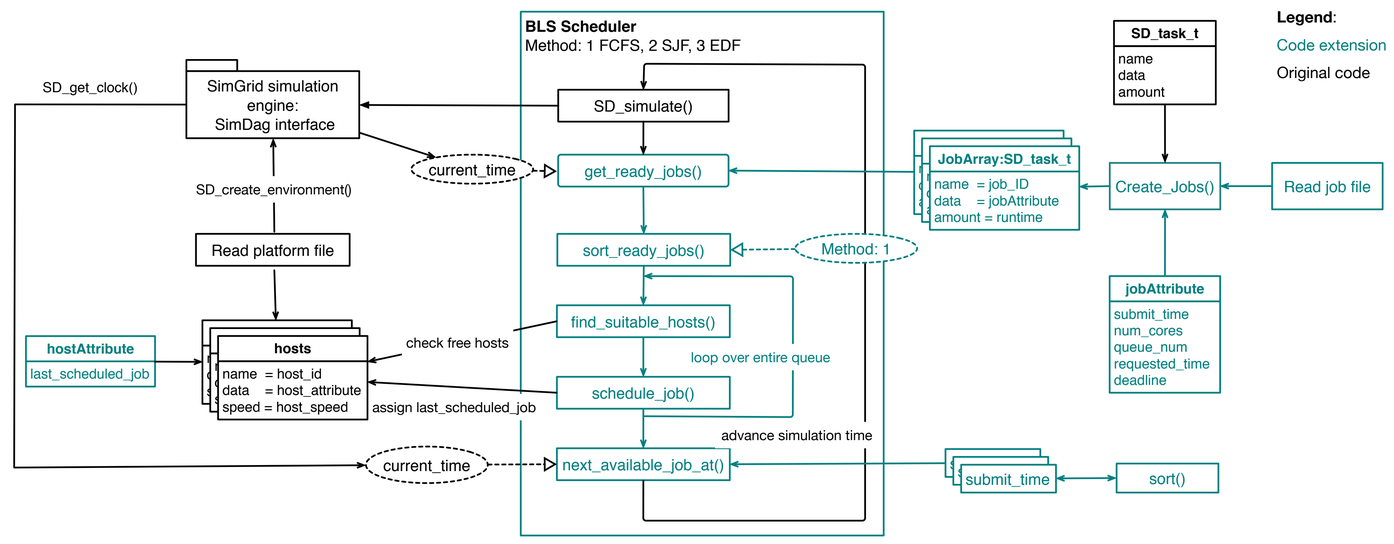

Figure 2—A conceptual diagram of SimGrid/SimDag execution and the extensions of this work to support BLS algorithms

6. Results

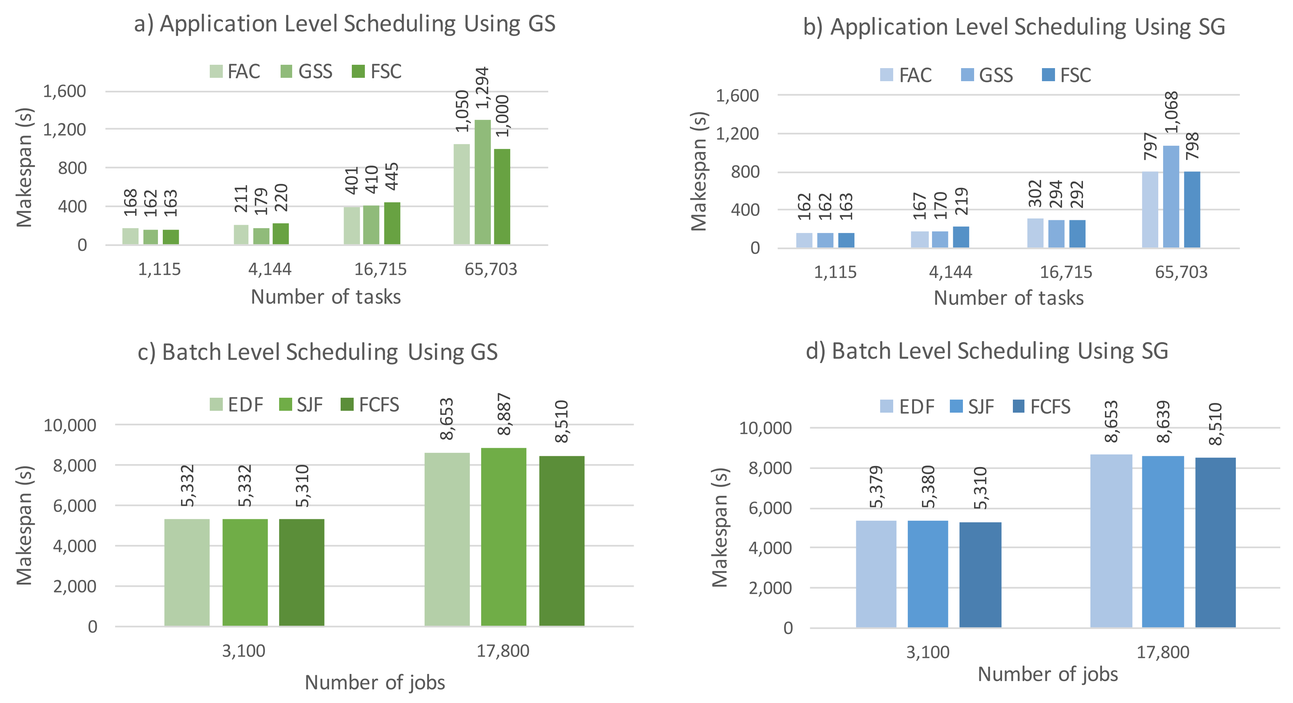

Figure 3—Simulated makespan of ALS and BLS algorithms using both simulators: charts a) and b) compare the simulated makespan for ALS algorithms, whereas charts c) and d) compare the simulated makespan BLS algorithms. Both simulators produced comparable or equal makespan for the ALS and BLS algorithms

7. Simulation Performance

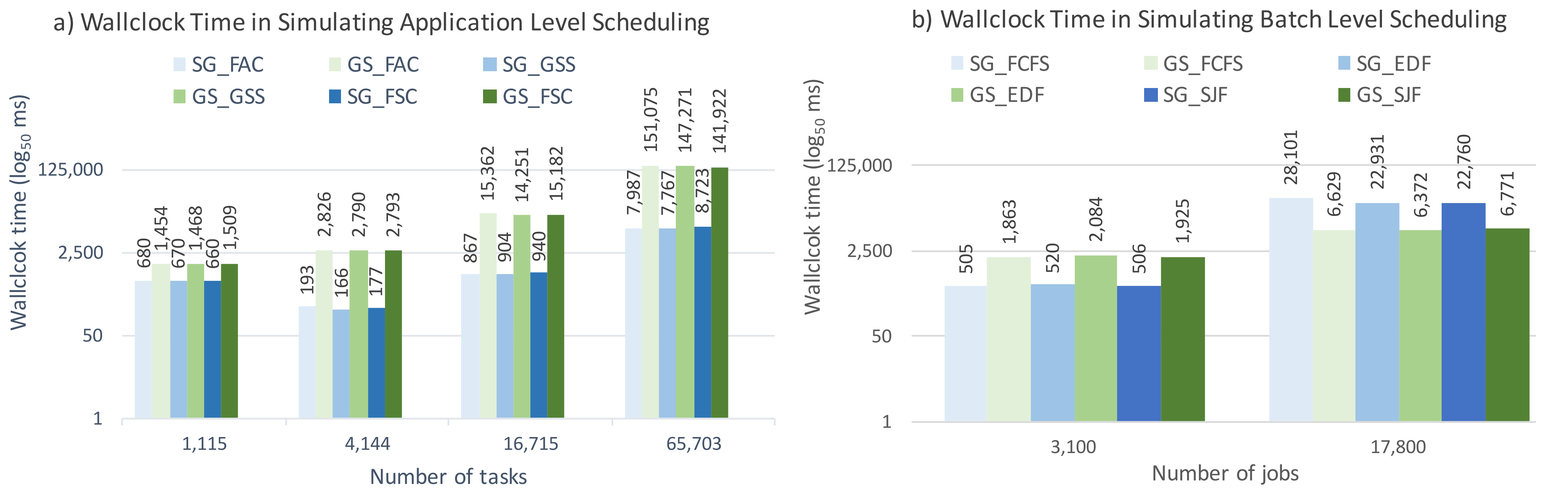

Figure 4—Performance of the extended simulators in terms of wallclock time for ALS (chart a) and BLS (char b). The results show that SG has an advantage in simulating ALS algorithms. However, GS has an advantage in the case of BLS algorithms for large workloads

8. Reproducibility of This Work

- Simulation platform
  - Macbook Pro, CPU intel Core i7 @2.5GHz
  - Operating system: OS X 10.11.5 (15F34)
- Libraries
  - GridSim v.5 and Alea v.4 [4]
  - SimGrid v.3.13 [2]
  - Lublin [3]
- Compilers
  - For SimGrid, clang v.7.3.0
  - For GridSim, javac v.1.8.0.91
- Extension code (ALS and BLS algorithms) and datasets
  - Available upon request, kindly contact the authors

9. Conclusion

- Application level and batch level scheduling support have been successfully implemented in both SimGrid/SimDag and GridSim/Alea
- For large workloads, SimGrid/SimDag outperforms GridSim/Alea in terms of simulation wallclock time in the case of ALS whilst GridSim/Alea outperforms SimGrid/SimDag in terms of simulation wallclock time in the case of BLS
- This work exposes the posssiblity of using one of the two simulation frameworks to simulate both ALS and BLS

References

[1] B. Rajkumar and M. Murshed. "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing", In Journal of Concurrency and Computation: Practice and Experience, vol. 14, no. 13, pp. 1175-1220, 2002.
[2] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter "Versatile, scalable, and accurate simulation of distributed applications and platforms", In Journal of Parallel and Distributed Computing, vol. 74, no. 10, pp. 2899-2917, 2014.
[3] U. Lublin and D. G. Feitelson "The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs", In Journal of Parallel and Distributed Computing, vol. 63, no. 11, pp. 1105-1122, 2003.
[4] D. Klusáček and H. Rudová "Alea 2 - job scheduling simulator", Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, pp. 61-71, 2010.
[5] 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", Proceedings of the 10th International Symposium on Parallel and Distributed Computing, pp. 86-93, 2011.