ESTEE: A Simulation Toolkit for Distributed Workflow Execution

Jakub Beránek
IT4Innovations, VŠB – Technical University of Ostrava
Ostrava, Czech Republic

Stanislav Böhm
IT4Innovations, VŠB – Technical University of Ostrava
Ostrava, Czech Republic

Vojtěch Cima
IT4Innovations, VŠB – Technical University of Ostrava
Ostrava, Czech Republic

ABSTRACT

Task graphs provide a simple way to describe scientific workflows (sets of tasks with dependencies) that can be executed on both HPC clusters and in the cloud. An important aspect of executing such graphs is the used scheduling algorithm. Many scheduling heuristics have been proposed in existing works; nevertheless, they are often tested in oversimplified environments. We introduce a simulation environment designed for prototyping and benchmarking task schedulers. Our simulation environment, scheduler source codes and graph datasets are open in order to be fully reproducible. To demonstrate usage of Estee, as an example, we compare the performance of various workflow schedulers in an environment using two different network models.

ACM Reference Format:
Jakub Beránek, Stanislav Böhm, and Vojtěch Cima. 2019. ESTEE: A Simulation Toolkit for Distributed Workflow Execution. In *Proceedings of Supercomputing '19: Technical Track (SC '19).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Representing a computation by a directed task graph is a common programming model for defining programs for a distributed system or a parallel computer. The main advantage of such a program description is the possibility to capture parallelizable behavior of an application while allowing to abstract the computation from specific architectures and computational resources. Task graphs are used in several areas with various levels of task granularity. Fine-grained task graphs occur in the context of task-based programming models where tasks are usually short running fragments of code within a single program [1, 2]. In contrast, coarse-grained task graphs are used to represent complex workflows composed of a set of potentially long-running programs [3–5]. Although our benchmarks primarily focus on the latter category, the results are generalizable to a wider spectrum of task graph scheduling problems.

Finding the optimal schedule for a task graph is NP-hard even for very restricted formulations (without transfer costs and resource management) [6]. Plenty of heuristics have been proposed to tackle this problem, ranging from list-based scheduling to genetic algorithms. Many surveys and comparisons of scheduling algorithms were published in [7–10].

Most of scheduler surveys assume an environment with an oversimplified communication and computation model. Some works use more complex communication models that attempt to simulate more realistic network behavior [11–14]. However, none of them deal with two important properties that inevitably arise during the actual execution of real world task graphs - namely that the scheduling itself takes time and that the duration of the individual tasks may not be known in advance to the scheduler. Also, to our best knowledge, no previous survey provides source codes of the implemented schedulers. Together with the fact that they often do not explain important implementation decisions, it is hard to reproduce and verify the previous results. The primary objective of this work is provide an extensible simulation environment that facilitates prototyping and evaluation of task graph schedulers and network models.

2 RELATED WORK

Popular distributed environment simulators such as Simgrid [15] or CloudSim [16] focus on deployment and provisioning infrastructures with low granularity of resource requirements, but do not consider scheduling task workflows with task dependencies. This problem has been assessed by DAGSim [11] and SimDAG [17]. Nonetheless, there are limitations that block us from building on top of these solutions in our work. DAGSim only reports experimental results without providing the actual implementation which makes it difficult to extend the solution or reproduce the results. SimDAG does not consider task resource requirements (e.g. number of cores) and also does not allow to define custom network models.

3 SIMULATION ENVIRONMENT

We have implemented Estee[1], a flexible open-source simulation environment that is designed for benchmarking and experimenting with task schedulers. The implementation is very open-ended and allows to implement new schedulers, network models and workers easily. However, it also comes "battery-included" and provides implementations for all its components.

Estee is written in Python to provide a high-degree of flexibility that facilitates rapid prototyping. Estee is an open-source project provided under MIT license.

SC '19, November 17–22, 2019, Denver, CO
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

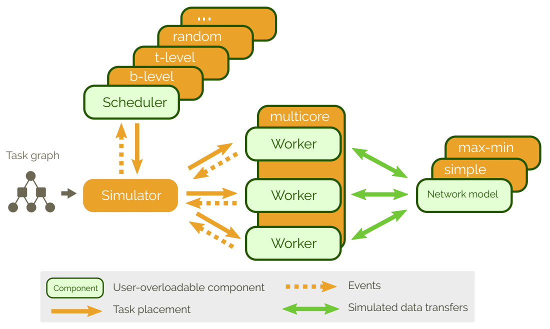


Figure 1: ESTEE architecture

3.1 Architecture

The architecture of ESTEE is depicted in Figure 1. The central component is the *Simulator*, which controls the whole simulation and communicates with the scheduler and workers. The *Scheduler* reads events about finished tasks and returns allocations of tasks to workers. A *Worker* simulates the execution of assigned tasks and also the transfer of task outputs between workers. The communication between workers is handled by a network model that informs them about download completion.

3.2 Schedulers

ESTEE features a set of schedulers inspired by classic scheduling heuristics. Originally these heuristics were mostly designed for environments with only one core per worker and one output per task; therefore, we had to slightly extend their implementation.

3.3 Task graph datasets

We use three task graph sets including simple elementary graphs as well as real world inspired graphs to test the behavior of schedulers in various situations. The first two sets are prepared by the authors, the third task graph set is derived from a set commonly used in other works. All graphs are published at [18].

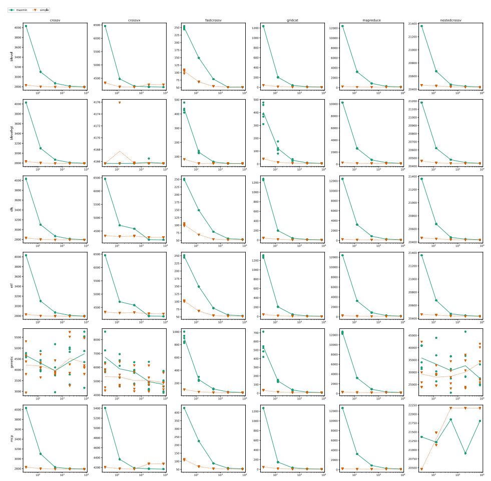
3.4 Network models

For simulating network connections, ESTEEprovide *max-min fairness* [19] and *simple* network models.

4 DEMONSTRATION

To demonstrate the usage of Estee, as an experiment, we compare scheduling algorithms in a distributed environments using two different network models with bandwidths ranging from 32 MiB/s to 8 GiB/s. Figure 2 compares makespans between the *simple* and *max-min* network models on selected task graphs using the *32x4* cluster for selected schedulers. It is clear that results obtained by using the *simple* model often under-approximate the resulting makespan. This is caused by the fact that network contention is not taken into account, which causes their transfer duration estimation to be overly optimistic. The differences vary based on the particular scheduler and task graph. Especially with slower bandwidths, the estimations produced by the two models can be an order of magnitude apart. Note that even small disparities are significant, since as shown in previous surveys [10] and in our provided results, the differences in produced makespans between existing scheduler heuristics are often very small and within a factor of two. As the bandwidth gets faster, the difference between the two models decreases, since network contention is lower and the *max-min* model starts to behave similarly to the *simple* model.

Figure 2: An example comparison of "maxmin" and "simple" netmodel; cluster 32x4



x axis: bandwidth [MiB/s]; y axis: makespan [s].

5 CONCLUSION

We have introduced ESTEE, an open-source framework that enables rapid prototyping and benchmarking of workflow schedulers in simulated distributed environments. We implemented a set of well known scheduling heuristics and prepared a dataset containing workflows of different types and scales.

As an example usage, we have shown that the complexity of the used network model may significantly affect the simulated workflow execution makespan. This confirms that it is important to consider the network behavior when applying scheduling heuristics in real-world applications and that it requires caution to refer to results that use simplified network models.

ESTEE, workflow datasets and scheduler implementations are open sourced, to make the results reproducible and extendable by the community. We believe that our results provide a comprehensive overview and comparison of workflow schedulers in various simulated conditions and that ESTEE has further potential to simplify the development and benchmarking of novel schedulers.

6 ACKNOWLEDGMENTS

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPS II) project "IT4Innovations excellence in science - LQ1602". This work was partially supported by the SGC grant No. SP2019/108 'Extension of HPC platforms for executing scientific pipelines', VŠB - Technical University of Ostrava, Czech Republic.

[1]https://github.com/it4innovations/estee

REFERENCES

[1] P. Thoman, K. Dichev, T. Heller, R. Iakymchuk, X. Aguilar, K. Hasanov, P. Gschwandtner, P. Lemarinier, S. Markidis, H. Jordan, T. Fahringer, K. Katrinis, E. Laure, and D. S. Nikolopoulos, “A taxonomy of task-based parallel programming technologies for high-performance computing,” *J. Supercomput.*, vol. 74, pp. 1422–1434, Apr. 2018.

[2] L. Dagum and R. Menon, “Openmp: An industry-standard api for shared-memory programming,” *Computing in Science & Engineering*, no. 1, pp. 46–55, 1998.

[3] S. Lampa, J. Alvarsson, and O. Spjuth, “Towards agile large-scale predictive modelling in drug discovery with flow-based programming design principles,” *Journal of Cheminformatics*, vol. 8, p. 67, Nov 2016.

[4] V. Cima, S. Böhm, J. Martinovič, J. Dvorský, K. Janurová, T. V. Aa, T. J. Ashby, and V. Chupakhin, “Hyperloom: A platform for defining and executing scientific pipelines in distributed environments,” in *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*, pp. 1–6, ACM, 2018.

[5] P. Amstutz, M. R. Crusoe, N. Tijanić, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, J. Kern, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic, “Common workflow language, v1.0,” 2016.

[6] J. D. Ullman, “Np-complete scheduling problems,” *J. Comput. Syst. Sci.*, vol. 10, pp. 384–393, June 1975.

[7] T. L. Adam, K. M. Chandy, and J. R. Dickson, “A comparison of list schedules for parallel processing systems,” *Commun. ACM*, vol. 17, pp. 685–690, Dec. 1974.

[8] Y.-K. Kwok and I. Ahmad, “Benchmarking the task graph scheduling algorithms,” in *ipps*, p. 0531, IEEE, 1998.

[9] T. Hagras and J. Janeček, “Static vs. dynamic list-scheduling performance comparison,” *Acta Polytechnica*, vol. 43, no. 6, 2003.

[10] H. Wang and O. Sinnen, “List-scheduling vs. cluster-scheduling,” *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[11] A. Jarry, H. Casanova, F. Berman, *et al.*, “Dagsim: A simulator for dag scheduling algorithms,” 2000.

[12] B. S. Macey and A. Y. Zomaya, “A performance evaluation of cp list scheduling heuristics for communication intensive task graphs,” in *IPPS/SPDP*, 1998.

[13] O. Sinnen and L. A. Sousa, “Communication contention in task scheduling,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, pp. 503–515, June 2005.

[14] X. Tang, K. Li, and D. Padua, “Communication contention in apn list scheduling algorithm,” *Science in China Series F: Information Sciences*, vol. 52, no. 1, pp. 59–69, 2009.

[15] H. Casanova, “Simgrid: A toolkit for the simulation of application scheduling,” in *Cluster computing and the grid, 2001. proceedings. first ieee/acm international symposium on*, pp. 430–437, IEEE, 2001.

[16] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[17] A. Zulianto, Y. S. Gondokaryono, *et al.*, “Hpc resources scheduling simulation using simdag,” in *Electronics Information and Emergency Communication (ICEIEC), 2016 6th International Conference on*, pp. 334–337, IEEE, 2016.

[18] “(omitted due to double-blind review),” 2019.

[19] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.