Using Simulation to Improve Workflow Scheduling in Heterogeneous Computing Systems

Alexey Nazarenko and Oleg Sukhoroslov ✉

Institute for Information Transmission Problems of the Russian Academy of Sciences,
Moscow, Russia
nazar@phystech.edu, sukhoroslov@iitp.ru

Abstract. Workflows is an important class of parallel applications that consist of many tasks with logical or data dependencies. A multitude of scheduling algorithms have been proposed to optimize the workflow execution in heterogeneous computing systems. However, in order to be efficiently applied in practice, these algorithms require accurate estimates of task execution and communication times. In this paper two modifications of the well-known HEFT algorithm are investigated that use simulation instead of simple analytical models in order to better estimate data transfer times. The results of experimental study show that the proposed approach can improve makespan for data-intensive workflows with high parallelism and communication-to-computation ratio.

Keywords: Workflow · Scheduling · Simulation · Heterogeneous systems · Distributed computing

1 Introduction

Heterogeneous computing systems (HCSs) composed of different computational units or standalone resources, which can be local or geographically distributed, are widely used nowadays for executing parallel applications. Workflows [14] is an important class of such applications that consist of many tasks with logical or data dependencies which can be modeled as directed acyclic graphs (DAGs).

The efficiency of executing workflows in HCS critically depends on the methods used to schedule the workflow tasks, i.e. decide when and which resource must execute the tasks of the workflow. The main objective is to minimize the overall completion time or makespan subject to possible additional constraints such as meeting a deadline or using a fixed budget. In comparison to homogeneous systems, the task scheduling problem in HCS is more complicated because of the different execution rates of individual resources and different communication rates of links between these resources.

The DAG scheduling problem has been shown to be NP-complete [9], even for the homogeneous case. This makes it practically impossible to obtain the optimal schedule even for the simplest formulations of practical interest. Therefore the research effort in this field has been mainly to obtain low complexity heuristics that produce good schedules. Since the late 1990s and until now, a multitude of workflow scheduling algorithms [18] based on different heuristics and metaheuristics have been proposed. However, in order to be efficiently applied in practice, these algorithms require accurate estimates of task execution and communication times.

In this paper we focus on the accuracy of models used for estimation of data transfer times. The presented experimental results provide a strong evidence against the widely used approach based on simple Hockney's model [11] that disregard network topology and bandwidth allocation. The schedules produced by static algorithms using this model clearly demonstrate that even for the modestly parallel workloads with sufficiently large data items the effect of competing data transfers may lead to the drastic underestimation of the communication time and the makespan degradation.

To address this issue we propose to incorporate simulation inside a workflow scheduling algorithm in order to improve the data transfer time estimates. Simulation, involving computer modeling of the process of application execution in HCS, has been actively used in scheduling algorithm research. The main advantage of simulation in comparison to the real-world experiments is the ability to perform a statistically significant number of experiments in a reasonable amount of time while ensuring the reproducibility and having moderate hardware resource requirements. However, while being widely used to evaluate the scheduling algorithms, the simulation has been rarely used inside the algorithms.

In this paper we investigate the use of more accurate simulation models instead of simple analytical models inside a workflow scheduling algorithm. Two modifications of the well-known HEFT algorithm [15] are proposed that use simulation in order to estimate data transfer times. The proposed modifications are compared with original HEFT and other scheduling algorithms using the developed simulation framework. The obtained experimental results show that the proposed approach can improve the makespan for workflows with high parallelism and communication-to-computation ratio.

The paper is structured as follows. Section 2 describes the used system and application models along with the used simulation framework. Section 3 provides an overview of HEFT algorithm and presents the proposed algorithm modifications. Section 4 presents and discusses the results of simulation experiments. Section 5 concludes and discusses future work.

2 Simulation Framework

To study the workflow scheduling algorithms in this paper we use simulation by modeling the process of application execution in a distributed computing system. In comparison with the full-scale experiments on real systems, simulation allows to significantly reduce the time needed to run an experiment and to ensure the reproducibility of produced results, while having moderate requirements to the used hardware resources. However, when using simulation it is important to ensure the accuracy, i.e. minimal deviation from the results of real-world experiments, and the scalability, i.e. the ability to conduct large-scale experiments, of the used simulation model.

The simulation model used in this paper is implemented on the base of SimGrid[1] [6], a simulation toolkit for studying the behaviour of large-scale distributed systems. The toolkit provides the required fundamental abstractions for the discrete-event simulation of parallel applications in distributed environments. The choice of SimGrid was motivated by the maturity of the toolkit, the soundness and high level of verification of embedded models, and the active support of developers. An important factor is also the versatility of the toolkit that allows one to simulate grids, cloud infrastructures, peer-to-peer systems and MPI applications.

Many studies also used WorkflowSim [7], an open source toolkit for simulating scientific workflows based on CloudSim simulator. We avoided the use of WorkflowSim as it has been shown that CloudSim among other simulators has flaws in its network model [17].

The heterogeneous computing system is modeled as a set of hosts and network links between them as depicted on Figure 1. Each host is characterized by its performance expressed in FLOPS. In this study it is assumed that each host can process a single task at a time. The execution of any task is considered non-preemptive. Network links are characterized by their bandwidth and latency.

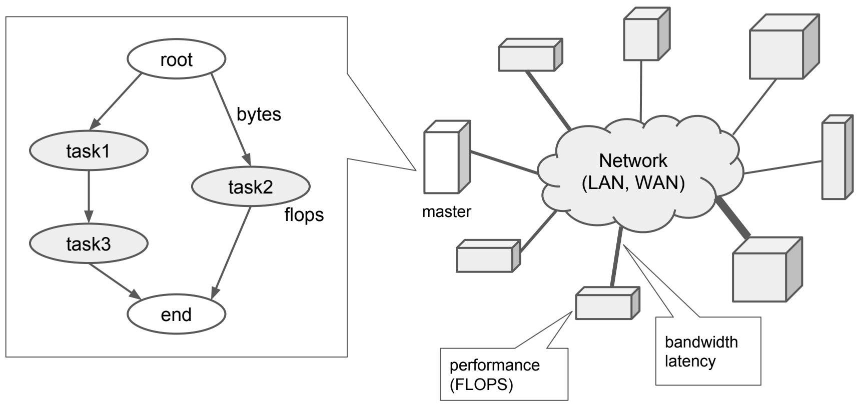

Fig. 1. Workflow and heterogeneous computing system models.

While the simulation is widely used to assess scheduling algorithms, the researchers often neglect the accuracy of the used models, especially network ones. In particular, in many papers authors assume a contention-free network model in which a network host can simultaneously send to or receive data from as many hosts as possible without experiencing any performance degradation. However,

[1] http://simgrid.gforge.inria.fr/

this model is not representative of real world networks. In this study we use the bounded multiport model provided by SimGrid. In this model, a host can communicate with several other hosts simultaneously, but each communication flow is limited by the bandwidth of the traversed route, and communications using a common network link have to share bandwidth. This scheme corresponds well to the behavior of TCP connections on a LAN. The validity of this network model has been demonstrated in [16].

SimGrid supports simulation of various network topologies including hierarchies and combinations of autonomous systems with different internal routing strategies. In this study we consider systems with a simple topology where each host is connected to a central backbone via a dedicated link as depicted on Figure 1, and a route between any two hosts contains the two respective links. The backbone, which can correspond to the LAN switch or the WAN, doesn't impose additional latency or bandwidth constraints in this model. Therefore the rate of communication between any pair of hosts is determined only by characteristics of the corresponding pair of links.

The workflow application is modeled as a directed acyclic graph (DAG), whose vertices correspond to individual tasks and directed edges represent the data dependencies between tasks as depicted on Figure 1. Each vertex is characterized by its size, i.e. the amount of computations in flops associated with the corresponding task. Similarly, each edge is characterized by the amount of communication in bytes between the corresponding pair of tasks. The size of task input data equals to the sum of sizes of incoming edges.

The two special tasks with zero size are introduced in order to model the staging of workflow input and output data. The *root* task passes the input data to the initial tasks, i.e. those that do not depend on other workflow tasks. The *end* task receives the output data from the final tasks, i.e. those that do not pass their data to other workflow tasks.

The *root* and *end* tasks are executed on a dedicated host called *master*, which does not participate in computations. This host corresponds to the machine, which stores the input data and where the output data should be placed after the application execution. In practice, this host often performs submission and management of the workflow.

While the SimGrid toolkit has been used previously for studying workflow scheduling algorithms [12, 2], to the best of our knowledge there are no published open source implementations of such algorithms for SimGrid. Therefore we have implemented a number of well-known static and dynamic algorithms, such as HEFT [15], HCPT [10], Lookahead [5], PEFT [1], OLB [3], MCT [13], MinMin [8, 13], MaxMin [8, 13] and Sufferage [13], following their original papers.

To simplify the implementation of scheduling algorithms for our experiments we have developed a *pysimgrid* library[2]. This library implements a thin wrapper around the native SimGrid API and provides a convenient interface for development of scheduling algorithms in Python language. The library also includes

[2] https://github.com/alexmnazarenko/pysimgrid

auxiliary tools for generation of synthetic systems and workflows, batch execution of simulation experiments and analysis of simulation results.

3 Modifications of HEFT Algorithm Using Simulation

3.1 HEFT Overview

Heterogeneous Earliest Finish Time (HEFT) [15] is probably the most cited workflow scheduling algorithm. Being relatively simple and proved to be consistently more efficient than other algorithms, HEFT is commonly used as a reference for evaluation of new algorithms.

HEFT can be described as a variant of static list scheduling algorithms that prioritize tasks having the most influence on the total workflow execution time (makespan). Such algorithms operate in two phases. During the first phase the algorithm computes the rank of each task according to some criterion that takes into account the position of the task in the DAG, its dependencies, etc. The output of the ranking phase is a list of tasks sorted by their rank. During the second phase the algorithm iterates over the list and assigns each task to a host that minimizes some criterion, for example task completion time.

The rank of a task T_i in HEFT is recursively defined by

$$rank(T_i) = \overline{EET}(T_i) + \max_{T_j \in succ(T_i)} \left(\overline{ECOMT}(data_{ij}) + rank(T_j)\right) \ , \tag{1}$$

where $\overline{EET}(T_i)$ is the average execution time of the task across all hosts, $succ(T_i)$ is the set of immediate successors of the task, $ECOMT(c_{ij})$ is the average communication time corresponding to the transfer of $data_{ij}$ bytes via edge (i, j).

The $\overline{EET}(T_i)$ is computed by averaging the estimated execution time $EET(T_a, H_i)$ of a task on each host H_i which is assumed to be known beforehand. In our model we compute accurate estimates using the task size and host performance.

The $ECOMT(data_{ij})$ is computed in HEFT using the Hockney's model [11] as

$$\overline{ECOMT}(data_{ij}) = \overline{L} + \frac{data_{ij}}{\overline{B}} \ , \tag{2}$$

where $\overline{L}$ is the average latency and $\overline{B}$ is the average bandwidth of communication links between the hosts in the system.

The tasks in HEFT are scheduled in decreasing order of their rank. Each task is scheduled to a host with a minimum estimated completion time

$$ECT(T_i, H_j) = EST(T_i, H_j) + EET(T_i, H_i) \ , \tag{3}$$

where $EST(T_i, H_j)$ is the earliest start time of the task on a given host

$$\begin{aligned} EST(T_i, H_j) &= \\ \max\{avail(H_j), \max_{T_k \in pred(T_i)} (ECT(T_k, H_k) &+ ECOMT(data_{ki}, H_k, H_i))\} \ , \end{aligned} \tag{4}$$

where $avail(H_j)$ is the earliest time the host is ready for task execution, $pred(T_i)$ is the set of immediate predecessors of the task.

Note the important feature of the rank function — it defines a valid topological order for the tasks. All tasks are scheduled after their parents, so it is possible to compute the required estimates of parent tasks' completion and communication times.

The communication time between tasks T_i and T_j running on hosts H_i and H_j respectively is computed as

$$ECOMT(data_{ij}, H_i, H_j) = L_{ij} + \frac{data_{ij}}{B_{ij}} \ , \tag{5}$$

where L_{ij} and B_{ij} are the latency and the bandwidth of the link between the given hosts.

3.2 Modified HEFT Versions

The simple linear model used in HEFT to estimate communication times doesn't take into account network topology and bandwidth allocation. A shown in Section 4, even for the modestly parallel workflows with sufficiently large data dependencies the effect of competing data transfers may lead to the drastic underestimation of the communication time and the degradation of HEFT performance. This is due to the fact that the inaccurate estimates of $ECOMT(data_{ij}, H_i, H_j)$ lead to inaccurate estimates of $ECT(T_i, H_j)$, and these inaccuracies accumulate during the scheduling of subsequent tasks. The ranking function also doesn't take into account the bandwidth contention by using a simple $\overline{ECOMT}(data_{ij})$ estimate.

To address this problem we modified HEFT to use simulation instead of analytical models to improve the used estimates. The proposed HEFT modification, hereinafter referred as SimHEFT, uses the same ranking phase as HEFT. However, during the task assignment phase SimHEFT uses simulation instead of analytical models to compute $ECT(T_i, H_j)$. For each host H_j, the execution of the workflow subgraph including already scheduled tasks and the current task T_i assigned to H_j is simulated. Note that these simulations are independent and, therefore, can be run in parallel. Then the task is scheduled to a host that corresponds to a minimum task completion time observed in simulations.

We also have tried to change the criterion used for selection of hosts during the task scheduling. Indeed, by optimizing the completion time of individual task it is possible to significantly degrade the completion times of already scheduled tasks due to the communication interference. The SimHEFT* variant schedules each task on a host that minimizes the overall makespan of currently scheduled subgraph instead of the task completion time. The intuition behind this variant is to minimally degrade the overall makespan during the scheduling of individual tasks.

The main advantage of the proposed approach is the minimal modification of the original algorithm. However, it is not clear without the experimental evaluation whether it is sufficient to improve only the task assignment phase while keeping the original ranking function and task scheduling order. Another concern is the additional overhead of simulation that can significantly increase the algorithm execution time. Note, however, that the proposed modifications allow running multiple simulations in parallel during the task assignment phase.

4 Experimental Evaluation

In this section we present the results of simulation experiments that compare the performance of proposed HEFT modifications with original HEFT and other workflow scheduling algorithms for a range of workflow and system configurations using the described simulation framework.

Besides HEFT we used two well-known dynamic algorithms - OLB and MCT. Opportunistic Load Balancing (OLB), which is widely used in modern HCSs, assigns available tasks to resources currently being idle without any a priori information about tasks. Minimum Completion Time (MCT) assigns each available task to a resource that is expected to finish the task the earliest.

We use the *makespan*, i.e. the measured total run time of a workflow in a given system according to a schedule produced by an algorithm, as the basis for comparison of algorithm performance. For each simulated pair system-application we run all algorithms and then normalize their makespans by the makespan achieved by the simplest algorithm - OLB. Finally, to reduce the variance, we compute the mean of normalized makespans across all simulations.

The experiments use a fixed set of workflows while varying the system characteristics. The used workflows are based on real world scientific applications [4]:

– **LIGO Inspiral:** analyses and filters the time-frequency data from the Laser Interferometer Gravitational Wave Observatory experiment (LIGO);
– **Epigenomics:** automates various genome sequencing operations (USC Epigenome Center);
– **Montage:** stitches together multiple images of the sky to create large-scale custom mosaics (NASA/IPAC);
– **CyberShake:** characterizes earthquake hazards in a region (SCEC).

The simulated systems have 5, 10 or 20 hosts with performance varying in a range of 1 to 4 GFlops. The network links have identical characteristics selected to be close to the Gigabit Ethernet network (bandwidth: 100 MBytes/sec, latency: 100 µs). For each host count 100 distinct systems are randomly generated.

The mean normalized makespans achieved by each algorithm in the experiments are presented in Table 1.

As it can be seen, HEFT outperforms the dynamic algorithms for the LIGO, Epigenomics and Montage workflows. The maximum speedup achieved in comparison to OLB varies among the workflows due to the different amount of inherent parallelism. However, for the CyberShake workflow both dynamic algorithms show similar results and outperform HEFT. The analysis of this workflow

Table 1. Mean normalized makespan

Hosts count	OLB	MCT	HEFT	SimHEFT	SimHEFT*
LIGO Inspiral, 100 tasks					
5	1.0000	0.9839	0.9651 (0.9608)	0.9652	1.0229
10	1.0000	0.9182	0.8792 (0.8602)	0.8791	1.0338
20	1.0000	0.7885	0.6898 (0.6865)	0.6898	0.9384
Epigenomics, 100 tasks					
5	1.0000	0.9753	0.9376 (0.9311)	0.9376	0.9368
10	1.0000	0.9014	0.8459 (0.8405)	0.8458	0.8437
20	1.0000	0.7942	0.7093 (0.6740)	0.7099	0.7067
Montage, 100 tasks					
5	1.0000	0.9791	0.9769 (0.9683)	0.9766	0.9766
10	1.0000	0.9639	0.9635 (0.9478)	0.9629	0.9636
20	1.0000	0.9109	0.9165 (0.9023)	0.9156	0.9172
CyberShake, 100 tasks					
5	1.0000	1.0104	1.0616 (0.5074)	1.0760	1.0395
10	1.0000	0.9972	1.0846 (0.3789)	1.1354	1.0325
20	1.0000	0.9845	1.1038 (0.2958)	1.3803	1.0244

revealed that it has two distinguishing properties — high parallelism and high communication-to-computation ratio (CCR). This could lead to a network contention resulting in a significant mismatch between the simple network model used in HEFT for estimation of *ECOMT* and the accurately modeled network in the simulator.

To confirm this hypothesis, we obtained the estimated makespan from the internal state of HEFT. These values, normalized to the simulated OLB makespan, are presented in brackets after the simulated HEFT makespan in Table 1. As it can be seen for the CyberShake workflow, HEFT expects to achieve a drastically different makespan than the one produced after the simulation. Ignoring the network contention effect resulted in more than 200% error in the makespan estimation. This result emphasizes the importance of accurate estimations of communication times during the workflow scheduling.

As for SimHEFT, it fails to improve the HEFT makespan for the Cybershake workflow while having a similar performance for other workflows. Contrary to expectations, SimHEFT behaves even worse than HEFT on Cybershake by showing up to 25% makespan degradation. We hypothesize that by optimizing the completion time of individual task it is possible to significantly degrade the completion times of already scheduled tasks due to the communication interference.

The SimHEFT* results confirm the above hypothesis. The results from Table 1 show that SimHEFT* was able to improve the Cybershake makespan by 2-7% in comparison to HEFT while having a similar performance for Epigenomics and Montage. However, SimHEFT* behaves significantly worse than HEFT and SimHEFT on the LIGO workflow. This can be explained by the fact that this workflow has the lowest CCR ratio and therefore is less sensitive to errors in estimated communication time. In this case the modified scheduling criterion doesn't bring any improvements over the original criterion and, as it can be seen, can even worsen the schedule.

While improving the Cybershake makespan, SimHEFT* is still up to 4% worse than dynamic MCT algorithm. This could indicate that it is not sufficient to improve only the task assignment phase while keeping the original HEFT ranking function and task scheduling order intact.

5 Conclusion and Future Work

In this paper we have investigated the use of simulation instead of simple analytical models inside a workflow scheduling algorithm to improve the estimation of communication times. Two extensions of the well-known HEFT algorithm that use simulation during the task assignment phase have been proposed. The experimental study of proposed modifications showed that it is not sufficient to simply plug simulation into the HEFT assignment phase (SimHEFT variant). However, by modifying the host selection criterion it is possible to improve the makespan for workflows with high parallelism and communication-to-computation ratio (SimHEFT* variant). As was demonstrated, such workflows suffer the most from inaccurate estimations of simple analytical models.

While it is demonstrated that the proposed approach have some potential, there are remaining challenges and room for improvement. The SimHEFT* variant is still behind simple dynamic algorithms for Cybershake and doesn't work well for workflows with low CCR ratio. The possible improvements here include modifications of the ranking phase and adapting the algorithm behaviour depending on the CCR ratio. The use of simulation significantly (up to two orders) increased the scheduling time. However, it is possible to decrease this time, e.g. by running the simulations in parallel during the task assignment phase. We plan to address the mentioned challenges in the future work and to perform an extended experimental study across a wide range of synthetic workflows.

Acknowledgments

This work is supported by the Russian Foundation for Basic Research (projects 15-29-07068, 15-29-07043).

References

1. Arabnejad, H., Barbosa, J.G.: List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems 25(3), 682–694 (March 2014)
2. Arabnejad, H., Barbosa, J.G., Prodan, R.: Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. Future Generation Computer Systems 55, 29–40 (2016)
3. Armstrong, R., Hensgen, D., Kidd, T.: The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh. pp. 79–87. IEEE (1998)
4. Bharathi S., Chervenak A., Deelman E., Mehta G., Su M. H., Vahi K.: Characterization of scientific workflows. In: 2008 Third Workshop on Workflows in Support of Large-Scale Science. pp. 1–10 (Nov 2008)
5. Bittencourt, L.F., Sakellariou, R., Madeira, E.R.M.: Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm. In: 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. pp. 27–34 (Feb 2010)
6. Casanova H., Giersch A., Legrand A., Quinson M., Suter F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. Journal of Parallel and Distributed Computing 74(10), 2899–2917 (2014)
7. Chen, W., Deelman, E.: Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In: E-science (e-science), 2012 IEEE 8th International Conference on. pp. 1–8. IEEE (2012)
8. Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., et al.: Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In: Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh. pp. 184–199. IEEE (1998)
9. Graham R. L., Lawler E. L., Lenstra J. K., Kan A. R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of discrete mathematics 5, 287–326 (1979)
10. Hagras, T., Janecek, J.: A simple scheduling heuristic for heterogeneous computing environments. In: Parallel and Distributed Computing, 2003. Proceedings. Second International Symposium on. pp. 104–110 (Oct 2003)
11. Hockney, R.W.: The communication challenge for mpp: Intel paragon and meiko cs-2. Parallel computing 20(3), 389–398 (1994)
12. Hunold, S., Rauber, T., Suter, F.: Scheduling dynamic workflows onto clusters of clusters using postponing. In: Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on. pp. 669–674. IEEE (2008)
13. Maheswaran M., Ali S., Siegal H.J., Hensgen D., Freund R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth. pp. 30–44. IEEE (1999)
14. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M.: Workflows for e-Science: scientific workflows for grids. Springer Publishing Company, Incorporated (2014)
15. Topcuoglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems 13(3), 260–274 (Mar 2002)

16. Velho, P., Legrand, A.: Accuracy study and improvement of network simulation in the simgrid framework. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques. p. 13. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2009)
17. Velho, P., Schnorr, L.M., Casanova, H., Legrand, A.: On the validity of flow-level tcp network models for grid and cloud simulations. ACM Transactions on Modeling and Computer Simulation (TOMACS) 23(4), 23 (2013)
18. Yu J., Buyya R., Ramamohanarao K.: Workflow scheduling algorithms for grid computing. In: Metaheuristics for scheduling in distributed computing environments, pp. 173–214. Springer (2008)