Towards a Realistic Scheduler for Mixed Workloads with Workflows

Ilyushkin, Alexey; Epema, Dick

DOI
10.1109/CCGrid.2015.74

Publication date
2015

Document Version
Accepted author manuscript

Published in
15th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing

Towards a Realistic Scheduler for Mixed Workloads with Workflows

Alexey Ilyushkin
Delft University of Technology
Delft, the Netherlands
a.s.ilyushkin@tudelft.nl
PhD student

Dick Epema
Delft University of Technology
Delft, the Netherlands
d.h.j.epema@tudelft.nl
Supervisor

***Abstract*—Many fields of modern science require huge amounts of computation, and workflows are a very popular tool in e-Science since they allow to organize many small, simple tasks to solve big problems. They are used in astronomy, bioinformatics, machine learning, social network analysis, physics, and many other branches of science. Workflows are notoriously difficult to schedule, and the vast majority of research on workflow scheduling is concerned with scheduling single workflows with known runtimes. The goal of this PhD research is to bring more realism to the problem of workflow scheduling in actual systems. First, in real systems, multiple workflows may be contending for the available resources. Second, task runtime estimates are not always known, and task runtime estimates may be wrong. Third, workflows are usually not the only type of jobs submitted to a system, there may for instance also be parallel applications and bags-of-tasks. Accordingly, the purpose of this PhD research is to create and analyze policies for online scheduling of workloads of workflows with and without known task runtimes that also contain jobs of other types. We are in the process of simulating policies, and we will validate our results by means of an implementation and real-world experiments with the KOALA-W workflow processing system.**

***Keywords*-scheduling; workloads; workflows;**

I. Problem Statement

Workflows (WF) are a very popular tool in e-Science [1], [2]. A typical application of WFs is DNA sequencing in bioinformatics. There they allow to perform massively parallelized sequencing of millions of nucleic acids, and the amount of data generated by these WFs is often gigabytes to terabyte [3]. Among all the job types in the workloads of modern large computing environments such as clusters and datacenters, WFs constitute one of the most difficult types of jobs to schedule [4]. Their sizes can reach thousands or hundreds of thousands of tasks and are virtually unbounded, and the amount of required resources for a single WF can significantly fluctuate during its execution. Almost all of the research in WF scheduling has focused on scheduling single WFs with known task runtimes. However, in reality, schedulers may have to schedule an arrival stream of WFs, their task runtimes may not be known in advance, and workloads usually do not solely consist of WFs, but of other application types as well. Therefore, the goal of my PhD thesis is to design and analyze policies for scheduling WFs with unknown task runtimes that are part of mixed workloads that arrive at a system—both by means of simulations, and by means of the design and implementation of, and experiments with the WF scheduler KOALA-W as part of the KOALA multicluster scheduler.

Many techniques for offline WF scheduling [5]–[7] have been investigated, but the amount of research in online scheduling workloads partially or completely consisting of WFs is much smaller. The streaming nature of a job arrival process makes it important to consider online scheduling algorithms that are able to solve scheduling problems in a dynamic way. There only a few works that consider an online stream of WFs. For example, in [8] the authors use a *Directed Acyclic Graph* (DAG) composition approach and task runtime information to prioritize the tasks using HEFT [5]. (A WF can be represented in a form of a DAG and these two notions are often used interchangeably.) In [9], a stream of arriving WFs is used and the ideas from [8] are extended by considering the critical path for each WF. In [10], the Pegasus planner [11] and the DAGMan [12] batch WF executor are used. However, all of these works suppose the knowledge of WF task runtimes.

Although in much of the research on WF scheduling, knowledge of the runtimes of WF components (tasks) is assumed, in many cases these runtimes are actually not known in advance. Several runtime estimation algorithms [13], [14] that usually use statistical methods to estimate the runtime of tasks or of whole WFs have been investigated, but these methods almost always provide inaccurate results. Some systems employ user runtime estimates, but often these are even more inaccurate. Therefore, it is a challenge for a scheduler to process jobs with unknown runtimes or with inaccurately predicted runtimes.

Of course, there are many other job types and the WFs are not the only group of interest for researchers. Here could be considered single query jobs like http requests, MapReduce jobs, parallel applications (MPI), and parameter sweep applications (bags-of-tasks). These job types can be represented in the system model as a background load which exists by default and the scheduler should cope with it.

First, we have proposed a set of online scheduling policies to schedule workloads of WFs in a distributed environment

with unknown task runtimes [4]. We have investigated these policies with simulations with realistic synthetic workloads. Each WF in our experiments consists of 30–600 tasks. To compare the proposed policies and to assess their performance we have utilized different system-oriented and user-oriented metrics.

Second, we will introduce runtime estimates in our model and we will analyse and compare the old and new simulation results to adjust our policies. In particular, we are planning to check how the runtime estimation error affects the scheduling quality and how the structure of a WF can help to improve the runtime prediction to get better scheduling results. In order to check how our design can manage bigger WF sizes, we are going to start experimenting with medium-sized WFs containing thousands of tasks. If these experiments are successful, we will try to process very large WFs with hundreds of thousands of tasks, such as our BTWorld workflow [15].

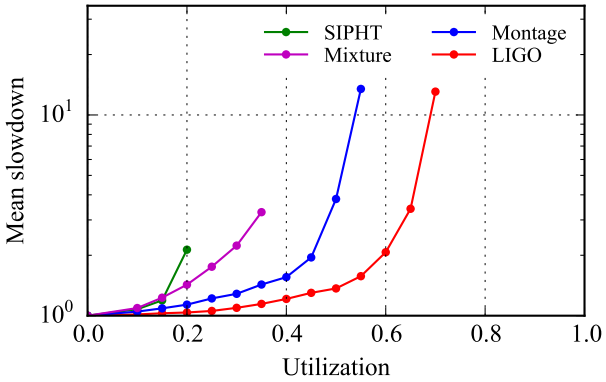
Third, to bring realism to our work, we will extend the model of our simulated computing system. We are planning to consider multi-core processors, other resource types such as storage, and communication delays. Moreover, we will implement the ideas obtained from the simulations in the KOALA[1] scheduler by adding to it a WF scheduling module KOALA-W with a runtime estimation algorithm. To validate our simulation results, we will perform real experiments on our DAS-4[2] system. Furthermore, we are going to address the problem of simultaneous execution of multiple large WFs with multiple independent schedulers, when they need to compete for shared computing resources.

II. Unknown Task Runtimes

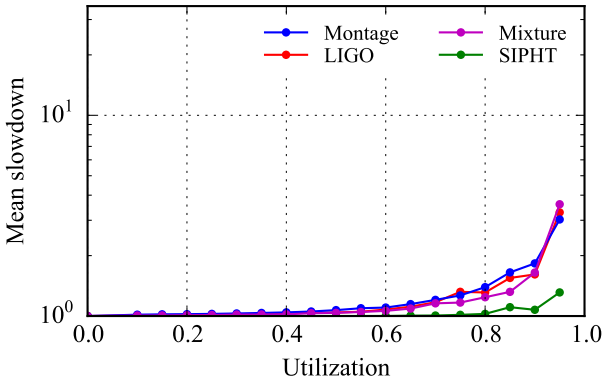
As the first part of this PhD research, which has been completed, we have proposed and analyzed with simulations four scheduling policies for scheduling workloads of WFs with unknown task runtimes. The main distinguishing feature of these policies is the number of processors they reserve for WFs towards the head of the waiting queue in order to deal with fluctuations in their *Level of Parallelism* (LoP). The LoP of a WF at a certain point before or during its execution is the maximum number of processors the WF can ever need during the remainder of its execution. We have also proposed a simple LoP estimation algorithm. We have tested it on five popular WF structures and showed its accuracy. The proposed WF scheduling policies are:

1) The *Strict Reservation* (SR) policy tries to reserve the LoP processors and can be considered as an implementation of the classical FCFS policy for WF scheduling.
2) The *Scaled LoP* (SLoP) policy behaves similar to the SR policy albeit with a lower reservation target, and in the boundary case when the scaling factor $f = 1$ it is equal to it.
3) The *Future Eligible Sets* (FES) policy analyses the workflow DAG to a certain depth to calculate the required number of processors for reservation. It employs the same algorithm we use to calculate LoP, and for lookup depth ∞ it is similar to the SR policy.
4) The *Backfilling* (BF) policy does not use the processor reservation at all and places any eligible tasks on the processors sequentially processing WFs in the queue.

[1] www.pds.ewi.tudelft.nl/koala
[2] www.cs.vu.nl/das4



(a) Strict Reservation policy



(b) Backfilling policy

Figure 1: The mean slowdown of workflows as a function of the utilization for the different policies and for each of the four workload types (the vertical axis is in log scale).

In Figure 1 we show the simulation results for two the most diverse scheduling policies which we proposed for online scheduling of WFs with unknown runtimes. We used three WF structures, namely Montage, LIGO and SIPHT [4], and four workloads. Three workloads consist only of a single WF type, and the fourth one contains an equal mixture of all the considered WF structures. As can be seen from Figure 1a and Figure 1b, the mean job slowdowns significantly differ between the Strict Reservation policy and the Backfilling policy. Our experiments showed that any form of preemptive processor reservation when scheduling workloads of WFs

with unknown task runtimes only decreases the system's performance.

III. The Benefit of Known Task Runtimes

As the second part of this PhD research, which we have just started at the time of writing, we will, again with simulations, investigate the benefit of using (estimates of) task runtimes in online WF scheduling. The execution of multiple similar WF applications can be used to collect and analyze statistical runtime information, such as the execution times of WF tasks, their memory consumption, etc. However, when the workload is highly diverse, this approach may not be very useful. Thus, in our planned simulations we are going to introduce task runtime estimates which we will vary in a certain range to see how the estimation error affects the quality of the scheduling decisions and the values of our performance metrics. The runtimes for workloads we are going to generate are based on the traces data from real systems using the information from workload archives. It is also possible to drive our simulations using real traces.

Knowing runtimes could be really useful when executing WFs with fluctuating LoP, since then the scheduler has much more information about tasks that can unlock other tasks to fill the gaps in the schedule. Obviously, exploiting the structural information about a WF makes no sense when runtime estimates are not available. The only important property which can be derived then is the LoP.

IV. Scheduling Mixed Workloads

It is hard to imagine a computing system completely dedicated to running WFs. Rather, systems will have to deal with mixed workloads consisting of multiple application types, with in addition to WFs also parallel applications, MapReduce jobs, bags-of-tasks and parameter sweeps, etc. In fact, WFs may only make up a small fraction of actual workloads. The scheduling of multiple job types in a single system is complicated because different application types have different resource and time requirements. Additionally, inappropriate coordination may cause race conditions and interlocking when multiple jobs compete for the same resource.

In addition to WF we are planning to consider two extra application types, which are also quite common in e-Science. These are conventional parallel applications (PA) which require a fixed set of processors for a certain amount of time, and bags-of-tasks (BoT) which represent a group of small single-processor tasks. Any other system or user activity can be easily represented as a background load. Here we can see two possible approaches for processing the mixture workload. The first one supposes the existence of a single queue for all the job types. The second one uses distinct queues for each job type. The priorities for each application type (or for each queue) can be assigned automatically to achieve, for instance, fair shares of computing resources or to prioritize a certain job type.

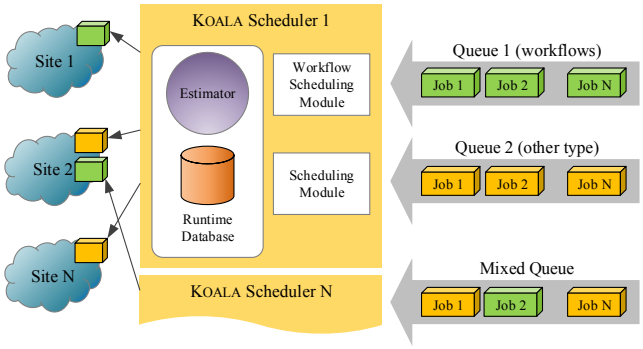


Figure 2: The KOALA scheduler.

For example, when WFs, PAs and BoTs arrive to the system, and the required number of processors for WFs and PAs is high, then BoTs with their small tasks can easily "flood" the system, delaying the processing of WFs and PAs. In opposite, a large WF or PA can easily occupy a significant part of system's resources and delay BoTs. Moreover, the WFs can introduce even more uncertainty with their LoP fluctuations. It means that static prioritization in this situation can be harmful.

We will also try to apply approaches which we designed for WFs to process other simpler job types, for instance, by dynamically combining BoTs in WF-alike structures for load balancing purposes.

V. An Implementation of Workflow Scheduler

The final step of my PhD research will be the implementation of the scheduling policies in a real system to check their practicability and detect any possible technical problems. For this purpose we will use KOALA scheduler. The implementation will be deployed on the Dutch supercomputer DAS-4 (or even on the future DAS-5). Figure 2 shows the diagram of KOALA scheduler with the KOALA-W module for WF scheduling when operating in multi-scheduler mode. Each scheduler can have multiple queues for different job types or a single shared queue. Runtime information are collected and consolidated in the runtime database. The prediction module provide necessary information for scheduling subsystems of KOALA, such as runtime and memory consumption estimates. Multiple schedulers can share a common resource space and can compete for the resources.

VI. Timeline

The timeline of my PhD research includes the following five points (Table I):

1) The development of four scheduling policies for online workflow scheduling and their implementation in the

Table I: My PhD Research Timeline.

Phase	Duration	Results (future)
Scheduling workloads of workflows with unknown runtimes	1.5 years	Simulator, LoP estimation, 4 policies
Scheduling workfloads of workflows with runtime estimates	6 months	New scheduling policies, runtime estimation algorithms
Scheduling mixed workloads	6 months	Improvements in the simulator, new policies
Scheduling very large workflows	6 months	Review of the existing policies
Improving system model, implementation of KOALA-W	1 year	Better system model, working workflow scheduler

DGSim [16] simulator. Additionally, we proposed the *Level of Parallelism* approximation technique.

2) The implementation of runtime estimations in the simulated model and the creation of new scheduling policies.
3) Modification in the simulator to schedule mixed workloads. Running trace-driven simulations.
4) Scheduling of very large workflows and adjustment, if necessary, of the policies.
5) Improving the system model, transferring the successful approaches validated in the simulator to the KOALA scheduler and performing real-world experiments on the DAS-4 (DAS-5) system.

ACKNOWLEDGMENT

This research is supported by the Dutch national program COMMIT.

REFERENCES

[1] I. J. Taylor *et al.*, *Workflows for e-Science*. Springer-Verlag London Limited, 2007.

[2] G. Juve *et al.*, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, pp. 682–692, 2013.

[3] C. Cantacessi *et al.*, "A practical, bioinformatic workflow system for large data sets generated by next-generation sequencing," *Nucleic Acids Research*, vol. 38, pp. e171–e171, 2010.

[4] A. Ilyushkin, B. Ghit, and D. Epema, "Scheduling workloads of workflows with unknown task runtimes," in *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2015.

[5] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 260–274, 2002.

[6] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, pp. 1400–1414, 2012.

[7] H. Zhao and R. Sakellariou, "Scheduling multiple DAGs onto heterogeneous systems," in *20th International Parallel and Distributed Processing Symposium*, 2006.

[8] Z. Yu and W. Shi, "A planner-guided scheduling strategy for multiple workflow applications," in *International Conference on Parallel Processing-Workshops*, 2008.

[9] C.-C. Hsu, K.-C. Huang, and F.-J. Wang, "Online scheduling of workflow applications in grid environments," *Future Generation Computer Systems*, vol. 27, pp. 860–870, 2011.

[10] K. Lee *et al.*, "Adaptive workflow processing and execution in Pegasus," *Concurrency and Computation: Practice and Experience*, vol. 21, pp. 1965–1981, 2009.

[11] E. Deelman *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, pp. 219–237, 2005.

[12] J. Frey, "Condor DAGMan: Handling inter-job dependencies," Tech. Rep., 2002.

[13] L. Yang, A. Bundy, C. Hughes, and D. Berry, "Fast, but approximate, workflow-runtime estimation using the bell-curve calculus," 2007.

[14] A. M. Chirkin, A. Belloum, S. V. Kovalchuk, and M. X. Makkes, "Execution time estimation for workflow scheduling," in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*, 2014.

[15] T. Hegeman *et al.*, "The BTWorld use case for big data analytics: Description, mapreduce logical workflow, and empirical evaluation," in *IEEE International Conference on Big Data*, 2013.

[16] A. Iosup, D. H. J. Epema, T. Tannenbaum, M. Farrellee, and M. Livny, "Inter-operating grids through delegated matchmaking," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2007.