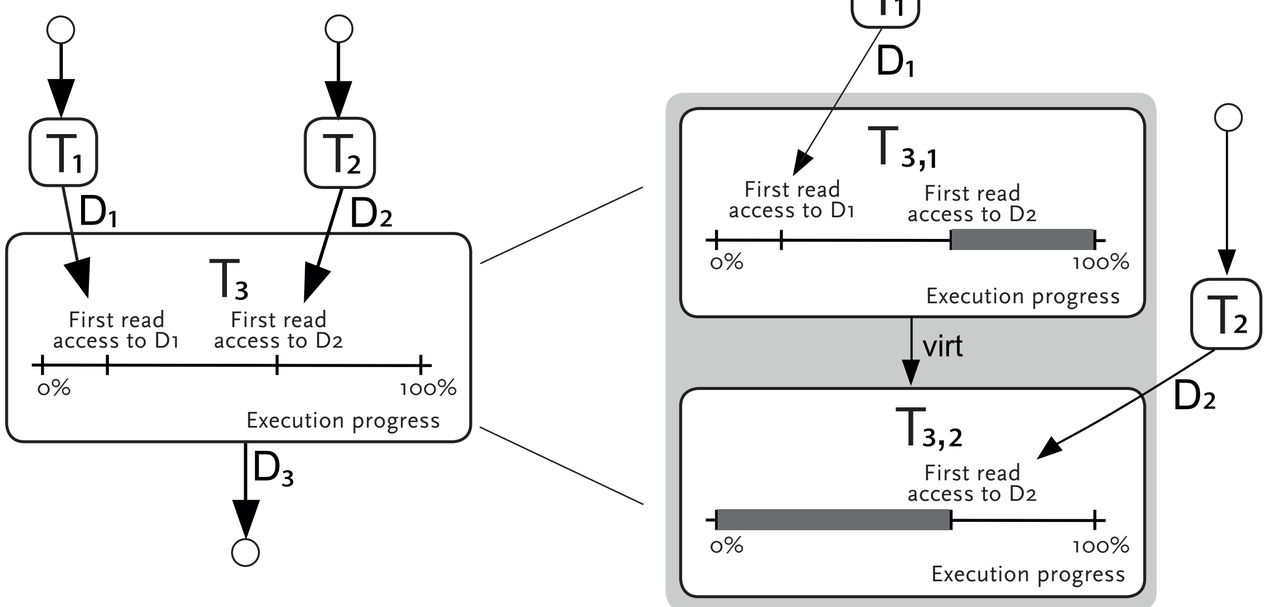
Starting Workflow Tasks Before They're Ready

Wladislaw Gusew and Björn Scheuermann
Computer Engineering Group, Humboldt University of Berlin
{gusewwly, scheuermann}@informatik.hu-berlin.de

Abstract

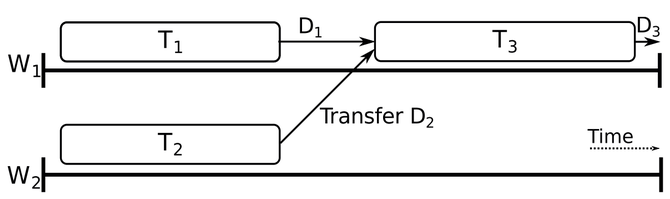
Today's science is more and more driven by collecting and evaluating increasing amounts of data. Utilizing Scientific Workflows is one suitable method how to organize processing pipelines for this purpose. In this work, we show that performance improvements on the execution of existing workflows can be achieved, if the conditions for starting selected tasks with certain data access characteristics are loosened. We provide a scheme how to identify eligible tasks in a given workflow and demonstrate a technique how an earlier start of tasks can be realized in Pegasus WMS by transforming the workflow DAG and by using a wrapper around the task executable during runtime. Our implemented wrapper handles the reading data accesses for task instances so that existing original workflows can be executed without the need to modify them. We evaluate our approach in simulations and experiments on real distributed computing resources, and are able to observe performance improvements for the Montage workflow by a significant reduction of total execution time.

Workflow Transformation

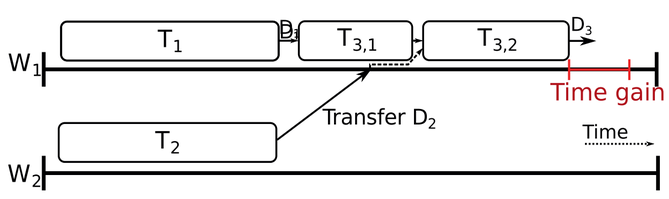


Original DAG with T3 and its input data characteristics.

Transformed DAG where T3 is virtually split at the point in time when D2 is accessed for the first time.



Exemplary schedule for the original DAG executed on two worker nodes W1 and W2.



Exemplary schedule for the transformed DAG with an earlier start of T3 resulting in a reduced makespan.

Several workflow execution engines consider a task as ready only after all parent tasks have finished execution. These semantics have proven to work for many practical workflows, but we show that the execution performance can be improved for workflows where two properties are present:

1) A task exists that merges multiple outputs of parent tasks (reduce pattern).
2) The reduce task accesses its input data in a consecutive way.

The figures to the left demonstrate the main idea behind our proposed optimization scheme. There an eligible task is split into multiple virtual tasks in order to be able to start the virtualized task earlier without the need to modify the implementation of the task or the execution engine. To this end, we utilized three different techniques in combination:

1) profiling black-boxed workflow tasks and infer from the profile the eligibility of the task for an earlier start,
2) transform the workflow DAG by refining the virtualized task as a cluster of tasks which represent calls to our specific wrapper,
3) a wrapper program which creates locally a virtual file system that provides the interface between the wrapper and the actual task executable.

Simulation Results

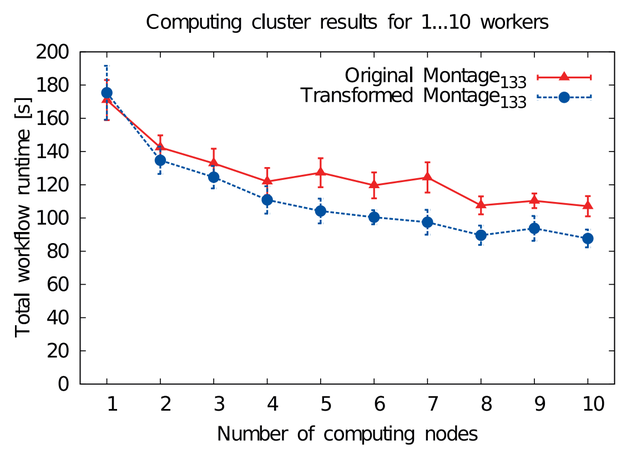
#VMs	#Tasks	Min-Min	Max-Min	Round-robin	HEFT	DHEFT	Random
		Scheduling and planning algorithms					
5	25	10.5	15.0	12.9	12.7	11.1	40.5
5	50	10.4	13.1	15.5	-22.8	15.2	39.9
5	100	10.1	11.1	12.1	8.7	13.4	12.6
5	1000	11.1	10.3	10.4	7.3	7.5	10.9
10	25	14.5	14.5	15.7	11.1	11.3	7.7
10	50	14.7	18.9	14.8	12.1	13.3	4.8
10	100	14.5	17.2	21.2	10.3	19.5	11.3
10	1000	17.0	16.4	16.1	8.6	10.5	-0.1
50	25	14.5	14.5	15.7	11.1	11.3	20.6
50	50	16.1	19.0	20.0	16.3	13.9	0.0
50	100	25.9	24.6	25.2	25.2	16.7	-0.8
50	1000	31.1	30.7	31.1	29.9	15.1	20.7
100	25	14.5	14.5	15.7	11.1	11.3	-8.4
100	50	16.1	19.0	20.0	16.3	13.9	1.4
100	100	24.8	26.6	27.6	25.1	16.7	33.9
100	1000	34.2	33.5	33.7	33.1	18.8	4.7

Reduction of total workflow runtime [%]: 40, 37, 34, 30, 27, 24, 20, 17, 14, 10, 7, 4, 0, -3, -6, -10, -13, -16, -20, -23

We evaluated our approach with the Montage workflow in a simulation by utilizing the *WorkflowSim* framework. Under many different configurations we analyzed how a split of the `mAdd` aggregation task into multiple virtual tasks affects the total runtime of the workflow.

In general, we are able to achieve performance gains by approx. 10% to 30% for simple scheduling algorithms in the simulation. Moreover, results indicate that the potential of outperforming the original workflow by the transformed workflow grows with an increasing computing cluster and workflow size.

Experimental Results



In order to evaluate our scheme in a realistic setting, we executed the Montage workflow containing 133 tasks in the Pegasus WMS on a small cluster with 10 computing nodes. The results underline our findings in the simulation that the transformation of a workflow in order to start aggregation tasks earlier can have a measurable impact on the execution performance. We could achieve a reduction of the total workflow runtime by up to 15%. When the number of workers is increased, both workflow executions can benefit from the increased parallelization potential. In the range from 8 to 10 nodes the curves seem to approach a saturation zone, where more available resources do not result in shorter runtimes, because the workflow structure does not provide any more parallelization potential.

eScience '16: 12th IEEE International Conference
Baltimore, Maryland, USA, October 2016