Toward Fine-Grained Online Task Characteristics Estimation in Scientific Workflows

Rafael Ferreira da Silva[1, 2], Gideon Juve[2], Ewa Deelman[2], Tristan Glatard[1, 3]
Frédéric Desprez[4], Douglas Thain[5], Benjamín Tovar[5], Miron Livny[6]

[1]University of Lyon, CNRS, INSERM, CREATIS, Villeurbanne, France
{rafael.silva,glatard}@creatis.insa-lyon.fr
[2]University of Southern California, Information Sciences Institute, Marina Del Rey, CA, USA
{gideon,deelman}@isi.edu
[3]McConnell Brain Imaging Centre, Montreal Neurological Institute, McGill University, Canada
[4]INRIA, University of Lyon, LIP, ENS Lyon, Lyon, France
Frederic.Desprez@inria.fr
[5]University of Notre Dame, Notre Dame, IN, USA
{dthain,btovar}@nd.edu
[6]University of Wisconsin Madison, Madison, WI, USA
miron@cs.wisc.edu

ABSTRACT

Task characteristics estimations such as runtime, disk space, and memory consumption, are commonly used by scheduling algorithms and resource provisioning techniques to provide successful and efficient workflow executions. These methods assume that accurate estimations are available, but in production systems it is hard to compute such estimates with good accuracy. In this work, we first profile three real scientific workflows collecting fine-grained information such as process I/O, runtime, memory usage, and CPU utilization. We then propose a method to automatically characterize workflow task needs based on these profiles. Our method estimates task runtime, disk space, and memory consumption based on the size of tasks input data. It looks for correlations between the parameters of a dataset, and if no correlation is found, the dataset is divided into smaller subsets by using a clustering technique. Task behavior estimates are done based on the ratio *parameter/input data size* if they are correlated, or based on the mean value. However, task dependencies in scientific workflows lead to a chain of estimation errors. To correct such errors, we propose an *online* estimation process based on the MAPE-K loop where task executions are constantly monitored and estimates are updated accordingly. Experiment results show that our online estimation process yields much more accurate predictions than an *offline* approach, where all task needs are estimated at once.

Keywords

Scientific workflow, workflow characterization, online task estimation, MAPE-K loop

1. INTRODUCTION

Scientific workflows have been widely used by computational scientist communities to run complex simulations and analyses [1]. They allow users to easily express multi-step computational tasks, for example retrieve data from an instrument or a database, reformat the data, and run an analysis. A successful and efficient workflow execution mainly depends on how tasks are planned and executed. Task scheduling is known to be an NP-complete problem [2], thus several heuristics have been developed to address this problem. For instance, classical heuristics such as Min-min, Max-min [3], and HEFT [4], or recent ones [5, 6, 7], have demonstrated good performance and improvements on task scheduling. In contrast, they share the same assumption that they have an accurate estimate of tasks needs such as execution and communication times, disk space, or memory usage. In production systems, it is hard to compute such estimates online and with good accuracy. Currently, scheduling algorithms use task estimation techniques that are not very accurate [8], or estimates are obtained from some distribution [9].

In addition, resource provisioning techniques may benefit from accurate task estimation to determine the number and characteristics of resources required to perform a computation. For instance, when a researcher uses a cloud infrastructure for processing scientific computations, accurate task needs estimates have direct impact on the cost of the computation and resource utilization [10], i.e. the user should request a number of resources so that the resource utilization is highest and the cost is minimal.

In our previous work [11], we presented workflow charac-

terization from the point of view of the performance of the individual workflow components and overall workflows. Profiling tools were developed to collect and summarize performance metrics of workflow applications. These tools collect fine-grained profile data such as process I/O, runtime, memory usage, and CPU utilization. In this work, we use these tools to profile three real scientific workflow executions, and we propose, as our first contribution, a method to automatically characterize workflow task needs such as runtime, disk usage, and memory consumption based on the workflow execution profiles. Our method assumes that these parameters can be estimated according to the input data size, because this is a parameter that could be known in advance, and the application execution time is usually dependent of the input data. Thus, it looks for correlations between the input data and the parameters. If no correlation is detected, execution datasets are divided into sub-datasets by a density clustering technique. Smaller datasets may have higher correlation coefficient, or lower standard deviation of the mean value.

Task estimation for scientific workflows differs from the general case of task estimation [12, 13] because of the dependencies between tasks. For instance, a bad estimation for a task output data, implies in poor estimations for dependent tasks, i.e. tasks where input data are dependent on the output data from the previous task. In a pipeline, estimation errors are propagated sequentially, however, in a workflow, estimation errors may be propagated successively. To address this issue, we propose an online estimation process based on the MAPE-K loop (Monitoring, Analysis, Planning, Execution, and Knowledge) [14] where task executions are constantly monitored and estimations are updated upon task completion.

We characterize three real scientific workflows using execution profiles obtained through workflow runs using the Pegasus workflow management system (WMS) [15] with the Kickstart profiling tool [16].

Our main contributions are summarized as:

1. an automated method that characterizes scientific workflow executions;
2. fine-grained characterization of three real scientific workflows;
3. an online estimation process to predict fine-grained task needs.

This paper is organized as follows. Section 2 presents the description of the scientific workflows used in this work. In Section 3, we present execution profiles of these workflows, and we introduce our automated method to characterize workflow executions. Our online task estimation process is presented in Section 4, and evaluated in Section 5. Section 6 presents the related work, and Section 7 concludes this paper.

2. SCIENTIFIC WORKFLOWS

A scientific workflow describes the dependencies between tasks. In most cases, the workflow is described as a directed acyclic graph (DAG), where the nodes are tasks and the edges denote task dependencies. This model is supported by several workflow management systems (WMS), such as Pegasus [15], Makeflow [17], Askalon [18], and Taverna [19].

In this work, we use the following real scientific workflows:

Montage. The Montage workflow [20] was created by the NASA/IPAC Infrared Science Archive as an open source toolkit that can be used to generate custom mosaics of the sky using input images in the Flexible Image Transport System (FITS) format. During the production of the final mosaic, the geometry of the output image is calculated from the input images. The inputs are then re-projected to have the same spatial scale and rotation, the background emissions in the images are corrected to have a uniform level, and the re-projected, corrected images are co-added to form the output mosaic. Figure 1 illustrates a small (20 node) Montage workflow. The size of the workflow depends on the number of images used in constructing the desired mosaic of the sky. The structure of the workflow changes to accommodate increases in the number of inputs, which corresponds to an increase in the number of computational tasks.

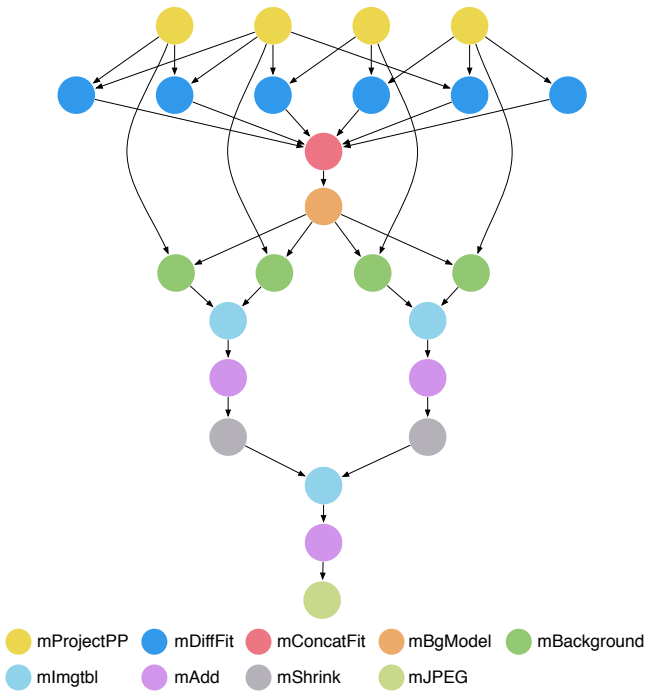

Figure 1: A small (20 node) Montage workflow.

Epigenomics. The USC Epigenome Center[1] is currently involved in mapping the epigenetic state of human cells on a genome-wide scale. The Epigenomics workflow (Figure 2) is a highly parallel application with multiple pipelines operating on independent chunks of data in parallel. The size of the workflow depends on the partitioning factor used on the input data.

[1] `http://epigenome.usc.edu`

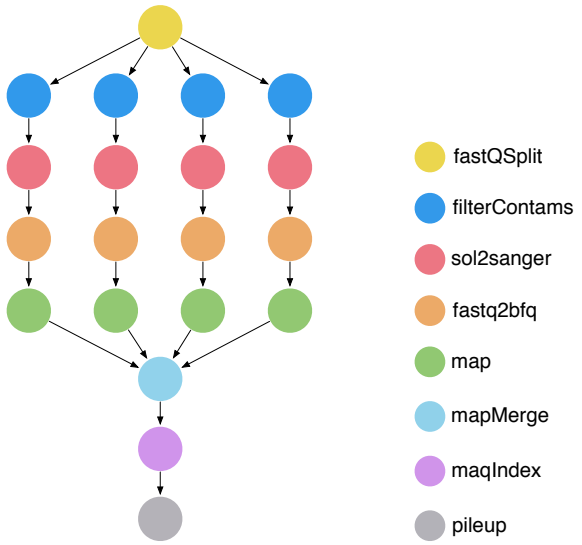

Figure 2: Epigenomics workflow.

Periodogram. The Periodogram workflow[2] searches for extra-solar planets, either by detecting "wobbles" in the radial velocity of a star, or dips in the starÕs intensity. In either case, the workflow searches for repeating variations over time in a sub-set of the light curves released by the Kepler project. Currently, three algorithms are available for computing periodograms from light curves: Lomb-Scargle (LS), Box-fitting Least Squares (BLS), and Plavchan. Figure 3 shows an illustration of a Periodogram workflow. The workflow can be seen as a bag-of-tasks of `periodogram_wrapper` tasks where each task can execute one of the available algorithms.

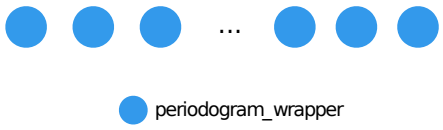

Figure 3: Periodogram workflow.

3. WORKFLOW CHARACTERIZATION

In this section, we characterize execution profiles for the three scientific workflows described in the previous section.

3.1 Workflow execution profiling

We profiled the three workflows by using the Kickstart [16] profiling tool, and the Pegasus WMS[3] for workflow execution. Kickstart monitors and records task execution in scientific workflows. It captures fine-grained profiling data such as process I/O, runtime, memory usage, and CPU utilization. Pegasus is a workflow management system that bridges the scientific domain and the execution environment by automatically mapping high-level abstract workflow descriptions onto distributed resources. It manages data on behalf of the user: infers the required data transfers, registers data into catalogs, and captures performance information while maintaining a common user interface for workflow submission.

Three runs of each workflow were performed with different data sets. Workflows were executed on a 16-core cluster, composed by 5 Dual core MP Opteron™ Processor 250 2.4GHz with 8GB of RAM, and 3 Dual core MD AMD Opteron™ Processor 275 2.2 GHz with 8GB of RAM.

Table 1 shows the execution profile for 3 runs of the Montage workflow. Some tasks have small standard deviation values compared to the mean, thus task estimation could be based on their mean values. However, for high standard deviation values of the mean, as for runtime of `mDiffFit`, and I/O write of `mBackground`, task estimation based on the average may lead to significant estimation errors. In particular, resource underestimation yields task failures while resource overestimation reduces resource utilization. Similarly, in the execution profile of the Epigenomics workflow (Table 2) `pileup` runtime values could be estimated based on the average, while `mapMerge` values would have more than 100% of estimation error if the average is used. This high estimation error case is also valid for the `periodogram_wrapper` task as shown in Table 3.

3.2 Automatic workflow execution characterization

We propose a method to characterize workflow tasks based on their estimation capability. We assume that task needs such as runtime, I/O write, and memory peak, can be estimated based on the I/O read parameter. Commonly, input data is read into memory, therefore there is a correlation between memory use and input size. Similarly, output data size may be correlated to the input data size, for example, when a task performs an image segmentation, or it may have a constant size, when the output data is a value. Experiment results presented in Section 5 support this assumption.

For each parameter that will be estimated, the method generates a dataset per task type containing information about the I/O read parameter and the actual parameter. Then, correlation statistics are enforced to the datasets to identify statistical relationships between parameters. Table 4 shows correlation (ρ) and standard deviation (σ) values for each task type for the three workflows, respectively. We consider that two parameters are correlated if their correlation value ρ is greater than 0.8. The threshold value of 0.8 was selected arbitrarily based on common sense. Correlated parameters, highlighted in the table, yield accurate estimations and no further analysis is required for them.

However, most correlation measures are sensitive to the data distribution. Therefore, density-based clustering [21] is performed to identify groups of high-density areas in datasets where no correlations were identified (e.g. Figure 4-top). Smaller datasets may increase the correlation coefficient, or lower standard deviation of the mean value.

[2] `https://portal.futuregrid.org/projects/77`
[3] `http://pegasus.isi.edu`

Task	Count	Runtime		I/O Read		I/O Write		Memory Peak	
		Mean (s)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.
mProjectPP	7965	2.59	0.69	4.24	0.19	16.20	0.80	9.96	0.40
mDiffFit	23733	1.25	0.92	24.08	5.76	1.35	1.11	5.32	0.90
mConcatFit	3	122.04	5.27	2.70	0.01	3.15	0.01	7.26	0.01
mBgModel	3	2008.08	88.50	4.14	0.04	0.27	0.00	14.41	0.01
mBackground	7965	2.14	1.68	13.67	6.78	13.05	6.44	11.75	5.78
mImgtbl	51	4.65	2.04	22.64	4.61	0.25	0.05	6.37	0.13
mAdd	51	47.69	14.03	2191.76	560.39	1574.22	383.86	21.66	3.40
mShrink	48	11.53	2.25	835.57	0.31	1.00	0.00	3.05	0.01
mJPEG	3	1.03	0.07	46.18	0.02	0.78	0.00	2.66	0.01

Table 1: Execution profile of Montage workflow executions for a 8 degrees square region of the sky.

Task	Count	Runtime		I/O Read		I/O Write		Memory Peak	
		Mean (s)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.
fastqSplit	15	22.94	9.00	755.85	297.11	755.94	297.15	1.92	0.01
filterContams	842	1.25	0.27	13.48	1.46	13.51	1.46	2.03	0.01
sol2sanger	842	0.56	0.32	24.58	2.11	18.46	1.49	3.57	0.01
fast2bfq	842	0.60	0.22	18.44	1.58	4.45	0.57	3.85	0.01
map	842	106.16	16.97	276.04	7.97	1.67	0.60	177.60	1.39
mapMerge	18	12.22	13.33	151.15	190.58	145.73	189.81	8.02	2.14
pileup	3	109.36	4.73	7347.57	576.47	6664.40	249.78	133.06	25.70

Table 2: Execution profile of Epigenomics workflow executions.

Task	Count	Runtime		I/O Read		I/O Write		Memory Peak	
		Mean (s)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.	Mean (MB)	Std. Dev.
periodogram_wrapper	86617	26.36	93.01	5.72	3.41	7.85	3.05	29.27	0.34

Table 3: Execution profile of Periodogram workflow executions.

Task	Runtime		I/O Write		Memory Peak	
	ρ	σ	ρ	σ	ρ	σ
mProjectPP	0.15	0.68	0.88	0.19	0.88	0.40
mDiffFit	0.46	0.91	0.01	1.19	0.01	1.08
mConcatFit	0.00	5.27	0.00	0.01	0.00	0.01
mBgModel	-0.99	88.50	1.00	0.00	0.96	0.01
mBackground	0.62	2.68	0.99	6.44	0.99	5.78
mImgtbl	0.54	2.84	0.92	0.05	0.88	0.13
mAdd	0.84	14.03	0.98	383.86	0.97	3.40
mShrink	-0.02	4.25	1.00	0.00	0.00	0.01
mJPEG	0.00	0.07	0.00	0.00	0.98	0.01

(a)

Task	Runtime		I/O Write		Memory Peak	
	ρ	σ	ρ	σ	ρ	σ
fastqSplit	0.98	9.00	1.00	297.15	0.00	0.01
filterContams	-0.03	0.27	0.99	1.46	0.00	0.01
sol2sanger	0.21	0.41	0.90	1.49	0.00	0.01
fast2bfq	0.18	0.27	0.56	0.87	0.00	0.01
map	0.02	18.96	0.06	0.70	0.01	1.43
mapMerge	0.98	13.33	0.99	189.81	-0.36	2.15
pileup	0.99	4.73	0.17	249.78	0.87	25.70

(b)

Task	Runtime		I/O Write		Memory Peak	
	ρ	σ	ρ	σ	ρ	σ
periodogram_wrapper	0.68	1333.12	0.69	189.81	0.83	0.34

(c)

Table 4: Correlation (ρ) and standard deviation (σ) values for the (a) Montage, (b) Epigenomics, and (c) Periodogram workflows. Highlighted cells indicate high correlation values.

Density-based clustering. We use DBSCAN [22] (density-based spatial clustering of applications with noise) as the clustering algorithm. The choice of DBSCAN was because it is one of the most common density-based clustering algorithm used in the scientific literature. DBSCAN's definition of a cluster is based on the notion of *density reachability*, i.e. a point q is directly *density-reachable* from a point p if the distance between them is smaller than a given distance ϵ, and p is surrounded by sufficiently many points such that one may consider p and q to be part of a cluster. A point is defined by a pair of parameter values, where in the x axis are represented the I/O read parameter values. For instance, Figure 4 shows the dataset clustering of the runtime and I/O write parameters for two task types (`mProjectPP` and `mDiffFit`) of the Montage workflow. In these datasets, 4 smaller datasets are identified where the correlation value is more significant or they converge to a unique point. Algorithm 1 shows the DBSCAN pseudocode. The value of the distance ϵ is chosen by using a *k-distance graph*, plotting the distance to the *minPts* nearest neighbors; good values of ϵ are where this plot shows a strong bend.

Algorithm 1 DBSCAN algorithm.

```
inputs: D dataset, eps, minPts
cluster C = 0
for p ∈ D and p is unvisited do
   mark p as visited
   neighborPts = regionQuery(p, eps, D)
   if neighborPts.size < minPts then
      mark p as noise
   else
      C = next cluster
      expandCluster(p, neighborPts, C, eps, minPts)
   end if
end for

expandCluster(p, neighborPts, C, eps, minPts)
add p to C
for p' ∈ neighborPts do
   if p' is unvisited then
      mark p' as visited
      neighborPts' = regionQuery(p', eps, D)
      if neighborPts'.size ≥ minPts then
         neighborPts = neighborPts ∪ neighborPts'
      end if
   end if
   if p' ∉ any cluster then
      add p' to C
   end if
end for

regionQuery (p, eps, D)
   return D' ⊆ D, where distance(p, q) ≤ eps, q ∈ D'
```

Correlation (ρ) and standard deviation (σ) values, and clusters (c) per task type are shown in Tables 5,6, and 7 for the Montage, Epigenomics, and Periodogram workflows, respectively. Datasets with high correlation values were not clustered (highlighted cells in Table 4), for example the I/O write parameter of `mAdd` for the Montage workflow, and `filterContams` for the Epigenomics workflow. Otherwise, subsets (clusters) of the datasets may have higher correlation values, and standard deviation values are smaller (e.g. runtime parameter of `periodogram_wrapper` in Table 7). In clusters where the correlation is null and the standard deviation is negligible, the data is concentrated in a unique point, i.e. the parameter is a constant value independent of the workflow input dataset. This is observed for most memory peak parameter of the Montage and Epigenomics workflows. After clustering, some datasets, as for the runtime parameter of `mImgtbl` for the Montage workflow, have lower correlation values than before, but also have lower standard deviation values. Thus, task estimation errors based on the mean values are smaller.

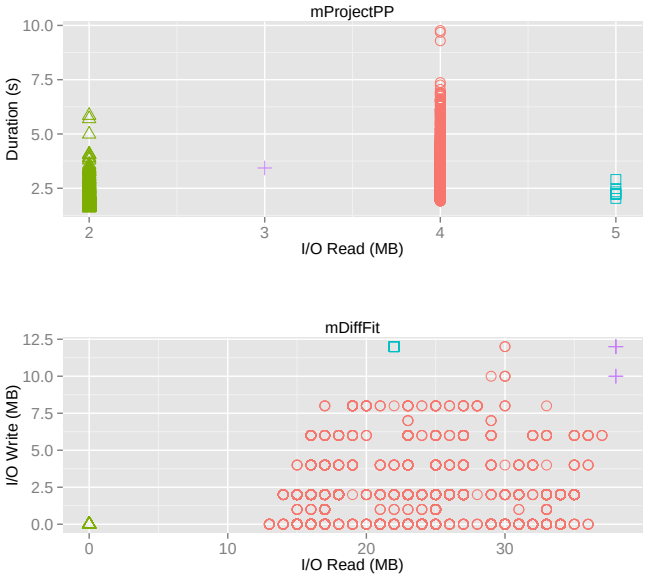

Figure 4: Dataset clustering of runtime and I/O write parameters for the Montage workflow.

Task	Runtime			I/O Write			Memory Peak		
	c	ρ	σ	c	ρ	σ	c	ρ	σ
mProjectPP	1	0.00	0.68	1	0.88	0.19	1	0.88	0.40
	2	0.00	0.54						
	3	0.00	0.28						
mDiffFit	1	0.04	1.08	1	0.01	1.17	1	0.01	1.03
	2	0.05	0.84	2	0.00	0.00	2	0.00	0.00
	3	0.07	0.61						
mConcatFit	1	0.00	5.27	1	0.00	0.01	1	0.00	0.01
mBgModel	1	-0.99	88.50	1	1.00	0.00	1	0.96	0.01
mBackground	1	-0.02	1.46	1	0.99	6.44	1	0.99	5.78
	2	0.00	0.00						
	3	-0.09	0.66						
mImgtbl	1	0.00	0.17	1	0.92	0.05	1	0.88	0.13
	2	0.28	1.85						
mAdd	1	0.84	14.03	1	0.98	383.86	1	0.97	3.40
mShrink	1	0.00	2.25	1	1.00	0.00	1	0.00	0.01
	2	0.00	1.58						
mJPEG	1	0.00	0.07	1	0.00	0.00	1	0.98	0.01

Table 5: Montage: clusters (c), correlation (ρ), and standard deviation (σ) values.

4. TASK ESTIMATION PROCESS

Figure 5 shows our general estimation process for one parameter. The process is based on regression trees. The tree is built offline from analyses of historical data. First, tasks are classified by application (workflow), then by task type. The next step decides whether runtime, I/O write, or memory parameters should be estimated based on the input data size. If the parameter is strong correlated to the input data, values are estimated according to the ratio *parameter/input data size*. Otherwise, values are estimated as the mean.

Task	Runtime			I/O Write			Memory Peak		
	c	ρ	σ	c	ρ	σ	c	ρ	σ
fastqSplit	1	0.98	9.00	1	1.00	297.15	1	0.00	0.01
filterContams	1	-0.03	0.27	1	0.99	1.46	1	0.00	0.01
	2	0.70	0.17						
sol2sanger	1	0.19	0.31	1	0.90	1.49	1	0.00	0.01
	2	0.39	0.31						
	3	0.17	0.08						
fast2bfq	1	0.12	0.21	1	0.24	0.73	1	0.00	0.01
	2	0.63	0.17	2	0.00	0.00			
map	1	-0.04	16.95	1	0.36	0.59	1	0.05	1.38
	2	0.41	14.10	2	0.37	0.55	2	0.54	0.89
mapMerge	1	0.98	13.33	1	0.99	189.81	1	0.55	1.98
							2	0.00	0.00
							3	0.00	0.00
pileup	1	0.99	4.73	1	0.17	249.78	1	0.87	25.70

Table 6: Epigenomics: clusters (c), correlation (ρ), and standard deviation (σ) values.

Task	Runtime			I/O Write			Memory Peak		
	c	ρ	σ	c	ρ	σ	c	ρ	σ
periodogram_wrapper	1	0.85	28.27	1	0.64	3.07	1	0.83	0.34
	2	-0.96	2937.36	2	-1.00	37.18			

Table 7: Periodogram: clusters (c), correlation (ρ), and standard deviation (σ) values.

An intermediate step could be added to the process to classify tasks by execution parameters (e.g. the 'degree' parameter for Montage workflows). The process outputs rules used to estimate future workflow executions. Figure 6 show examples of rules to estimate I/O write for the `periodogram_wrapper` task (Periodogram workflow).

An *offline* task estimation approach estimates at once task runtimes, output data sizes (I/O write), and memory peaks for all tasks in a workflow. In scientific workflows, poor output data estimations may lead to a chain of estimation errors: the output data of a task is the input data of another task in a subsequent level. Hence, runtime and memory peak may also be poorly estimated for the sub-sequential task—as our estimation process is based on the input data. Therefore, we propose an *online* task estimation process based on the MAPE-K loop (Monitoring, Analysis, Planning, Execution, and Knowledge), where task executions are constantly monitored [23]. Upon task completion, estimated values for the task are updated with the real values, and based on these values a new prediction is done (using the regression tree of Figure 5) for subsequent tasks of multiple levels (tasks that are data-dependent of the current task). Figure 7 summarizes the online estimation process. Note that in a workflow, tasks may have multiple parents, thus at an instant time t, their input data will be a composition of estimated and real (for completed parent tasks) values.

5. EXPERIMENT AND EVALUATION

The experiment presented hereafter aims at evaluating the accuracy of the online estimation process in comparison to the offline estimation process.

5.1 Experiment conditions

Trace analyzes were performed in the three workflow applications described in Section 3: Montage, Epigenomics, and Periodogram. For each workflow, we use the three different executions used to characterize the workflows. Two executions are reserved for training purposes (generation of rules), and the third one is used for test the accuracy of the estimation process (leave-one-out cross-validation). For each execution, we perform an analysis to test the accuracy of the prediction, thus results presented in the next subsection are an average of these analyzes. We assume that a parameter is statistically correlated if its correlation coefficient ρ is greater than or equal to 0.8. Otherwise, the mean value is used.

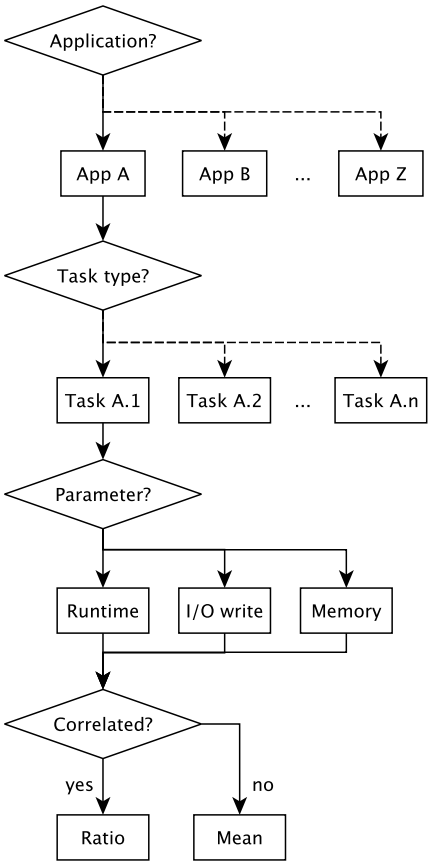

Figure 5: Estimation process for one parameter.

We implemented a simple DAG analyzer that parses a workflow description and spawns tasks and their dependencies. The analyzer implements both the offline and online estimation processes. An analysis consists in replaying a workflow execution, estimating tasks runtimes, I/O writes, and memory peaks at once, for the offline approach, or upon task completion, for the online one. Replaying a workflow execution means that our simulator processes each task in the same order of its real execution; tasks experience the same delays and resource performances. The simulator only computes task needs estimations, and compares them to the real values to assess estimation errors. We do not aim at evaluating the efficiency of scheduling algorithms, but the accuracy of our online estimation process.

5.2 Results and discussion

```
if workflow = 'Periodogram'
   and taskType = 'periodogram_wrapper'
   and parameter = 'write'
   and input_size ≤ 45088768 then
      return [7371489.28, mean] // mean value in bytes
end if

if workflow = 'Periodogram'
   and taskType = 'periodogram_wrapper'
   and parameter = 'write'
   and input_size > 45088768 then
      return [0.38, ratio] // ratio of output and input data
end if
```

Figure 6: Rules for I/O write estimation of the Periodogram workflow.

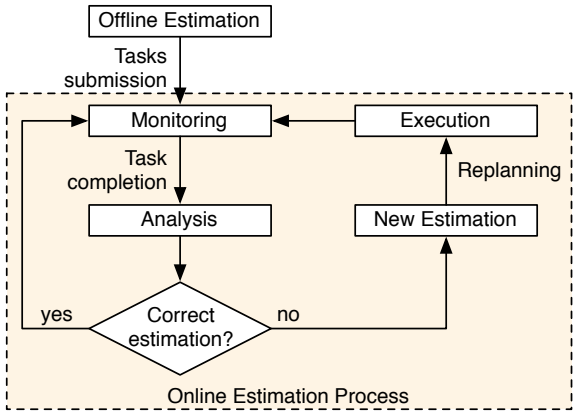

Figure 7: Online estimation process.

Table 8 shows the average estimation error of both estimation processes for the Montage workflow. In general, the online process has an average estimation error of 18% for runtime, 9% for I/O write, and 13% for memory peak, while the offline has 43%, 56%, and 53% respectively. For the first-level tasks (`mProjectPP`) both offline and online approaches have the same accuracy as tasks are estimated directly from the workflow input data. However, offline estimations for tasks such as `mDiffFit`, `mBackground`, `mImgtbl`, and `mAdd` are extremely affected by the propagation of estimation errors. For instance, the input data of a `mDiffFit` task are multiples `mProjectPP` output data. From Table 5, we notice that `mDiffFit` has low correlation values, thus mean values are used in the prediction. A bad estimation of the input data size may lead the process to select the wrong cluster. The online process initially faces the same problem of erroneous estimations, but upon task completion, wrong predictions are replaced by the actual value.

Table 9 shows the average estimation error for the Epigenomics workflow. The average estimation error for the offline process is 29% for runtime, 57% for I/O write, and 48% for memory, and for the online process is 13%, 5%, and 8% respectively. Similarly, first-level task estimations (`fastqSplit`) are the same for both approaches. Offline estimations for `filterContams`, `sol2sanger`, `fast2bfq`, and `mapMerge` are significantly affected by previous erroneous estimations of their parent tasks.

Task	Estimation	Runtime Avg. Error (%)	I/O Write Avg.Error (%)	Memory Avg.Error (%)
mProjectPP	Offline	18.95	1.63	2.80
	Online	18.95	1.63	2.80
mDiffFit	Offline	191.02	159.46	91.07
	Online	46.52	69.14	73.72
mConcatFit	Offline	4.38	0.00	7.62
	Online	4.03	0.00	6.22
mBgModel	Offline	23.83	0.00	22.08
	Online	1.17	0.00	3.43
mBackground	Offline	65.13	102.80	104.62
	Online	44.90	1.23	1.84
mImgtbl	Offline	61.27	127.29	126.58
	Online	29.15	5.53	8.35
mAdd	Offline	9.67	113.14	110.20
	Online	9.31	3.43	9.06
mShrink	Offline	13.72	0.34	0.00
	Online	7.61	0.33	0.00
mJPEG	Offline	1.61	0.00	19.09
	Online	1.37	0.00	11.40

Table 8: Montage: average estimation errors of task runtime, I/O write, and memory peak.

Task	Estimation	Runtime Avg. Error (%)	I/O Write Avg.Error (%)	Memory Avg.Error (%)
fastqSplit	Offline	8.36	3.28	9.14
	Online	8.36	3.28	9.14
filterContams	Offline	59.31	109.81	102.83
	Online	29.13	5.35	8.15
sol2sanger	Offline	54.93	98.20	96.68
	Online	34.74	1.23	1.96
fast2bfq	Offline	27.13	128.18	99.98
	Online	17.09	15.11	10.65
map	Offline	23.62	0.00	21.07
	Online	1.39	0.00	3.33
mapMerge	Offline	53.74	93.34	1.01
	Online	10.22	9.39	1.00
pileup	Offline	6.00	4.17	49.42
	Online	5.11	3.87	19.31

Table 9: Epigenomics: average estimation errors of task runtime, I/O write, and memory peak.

Table 10 presents average estimation errors for the Periodogram workflow. As the workflow has only one task level (`periodogram_wrapper`, see Figure 3), the online approach produces the same result as the offline. I/O write and memory estimation errors are low, but runtime predictions are correctly for a bit more than 50% of the tasks.

Task	Estimation	Runtime Avg. Error (%)	I/O Write Avg.Error (%)	Memory Avg.Error (%)
periodogram_wrapper	Offline	45.13	16.72	1.02
	Online	45.13	16.72	1.02

Table 10: Periodogram: average estimation errors of task runtime, I/O write, and memory peak.

In all analyzes for the 3 scientific workflows, the online process is more accurate when predicting task needs. The importance of using a loop to constantly monitor task executions to update estimations is emphasized on workflows due to their task dependency model. Although the online strategy counterbalances the propagation of estimation errors, the estimation of first-level tasks have strong influence in subsequent estimations. Therefore, efforts should be concentrated on techniques to address more accurate offline predictions. One approach to improve offline predictions would

be to consider other parameters such as command-line arguments when estimating workflow executions.

6. RELATED WORK

Workload archives are widely used for research on distributed systems, to validate assumptions, to model computational activity, and to evaluate methods in simulation or in experimental conditions. Available workload archives, such as the Parallel Workloads Archive [24], the Grid Workload Archive [25], and the Grid Observatory [26], provide workloads from parallel and grid execution environments. These workloads mainly capture information about task executions, but lack fine-grained information of scientific workflow executions, such as dependencies among tasks, task substeps, and artifacts introduced by application-level scheduling. Therefore, some efforts have been done to collect and publish traces and performance statistics for real scientific workflows. We recently published traces for a few workflows executed using Pegasus [27], and traces of several workflow executions obtained from a science-gateway [28]. We also published synthetic workflows based on statistics from real applications for use in simulations [29]. Similarly, Ramakrishnan and Ganon [30] have provided data statistics for many real workflow applications, and Ostermann et al. [31, 32] have provided analyzes of workflow-based workload traces from the Austrian grid.

On workload characterization in distributed environment, Iosup and Epema [33] and Hart [34] presented analyzes of grid and HPC workloads characteristics including system usage, user population, application characteristics, and characteristics of grid-specific application types. Ren et al. [35] presented an analysis of a MapReduce trace derived from a production Hadoop cluster, where they analyzed job characteristics such as CPU utilization, memory usage, slots allocation, I/O operations, and network transfers. Mahambre et al. [36] characterized a cloud workload into patterns based on their behavioral characteristics and presented statistical techniques to understand the patterns. They categorized the virtual machine workload in the following patterns: periodicity, threshold, relationship, variability, and image similarity. Recently, Madougou et al. [37] provided a characterization of workflow executions using provenance data captured from a workflow management system. They analyzed usage and failure patterns at workflow and task levels.

Workload estimations are generally used by resource allocation strategies and task scheduling algorithms in distributed platforms, such as clouds and grids. Verboven et al. [8] presented a parameter sweep prediction framework GIPSy, which estimates task runtimes based on previous runtime information. They performed evaluations using six different models: polynomial approach, radial basis functions, kriging models, neural networks, support vector machines, and nearest neighbor prediction. Their approach gives good accuracy, but is not applicable to an online environment. Sonmez et al. [12] studied job runtime and queue wait time prediction methods and their application in grid scheduling. They evaluated time series prediction methods when predicting job runtimes, and point-valued and upper-bound predictions when estimating queue wait times. A comparison to scheduling techniques that do not use prediction, show that the use of these techniques do not imply a better performance of grid scheduling. Pacheco-Sanchez et al. [38] proposed a Markovian Arrival Process (MAP) to predict HTTP workloads in cloud infrastructures. The process captures moments of the probability distribution, autocorrelation, and temporal dependencies of a time serie. Khan et al. [39] also proposed a method of characterizing and predicting workload in a cloud environment. Their method discovers and leverages repeatable workload patterns within groups of virtual machines (VMs) that belong to a cloud customer. They also developed a co-clustering technique for identifying such VM groups and the common workload patterns. A method based on Hidden Markov Modeling is used capture temporal correlations and to predict the changes of workload pattern. The use of Markov-based techniques provide good accuracy when predicting workloads. However, it adds a significant overhead to the application execution. In this work, we adopted an approach where the methods used to provide estimation should be computed online with negligible overheads.

On workflow workloads estimation, Duan et al. [40] proposed a hybrid Bayesian neural network method for modeling and predicting execution time of workflow activities in grids. Contrary to our work, they use resource information to estimate runtime. Thus, their approach is useful for the task scheduling problem, but it is not applicable for the resource provisioning problem. On the other hand, Eun-Kyu Byun et al. [41] and Huang et al. [42] proposed heuristics and models to estimate the number of resources required to execute a workflow. They assume that task runtimes are available. Nadeem and Fahringer [43] proposed a workflow performance prediction system using similarity templates. Templates are generated from different workflow attributes reflecting workflow performance at different grid infrastructure levels, and are evaluated through an exhaustive search method. The drawback of their approach is that they rely on an expert user to emphasize attributes when defining templates.

7. CONCLUSION

We presented a method to online estimate fine-grained task needs such as runtime, disk usage, and memory consumption. We profiled three real scientific workflow executions, and defined a process to automatic characterize these profiles. We assume that task needs can be estimated based on the size of the input data. Our process looks for correlations between the task need parameters and the input data size. If no correlation is found, density-based clustering is performed to identify groups of high density areas. Smaller groups may have higher correlation, or lower standard deviation values. Then, we defined a process, based on the MAPE-K loop, to online estimate task needs according to workflow execution characterizations.

The method was evaluated through the analysis of workflow execution traces where the accuracy of our process was measured in comparison with the real value. We also compared the accuracy of our online method against an offline estimation process, where all tasks of a workflow are estimated at once. Results shows that our online estimation process outcomes more accurate estimation than the offline method. In addition, we showed that poor output data estimations lead to a chain of estimation errors in scientific workflows,

hence the importance of using an online strategy where task executions are constantly monitored and estimations are updated accordingly. Future work includes the analysis of the impact of re-planning a workflow when using an online estimation strategy, and a sensitivity analysis of the correlation value ρ. We also plan to increase the number of workflow samples and to compare the results with other monitoring tools.

Acknowledgments

This work was funded by DOE under the contract number ER26110, "dV/dt - Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science". The research leading to this publication has also received funding from the EC FP7 Programme under grant agreement 312579 ER-flow = Building an European Research Community through Interoperable Workflows and Data, and the framework LABEX ANR-11-LABX-0063 of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

8. REFERENCES

[1] I. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer, 2007.

[2] J. D. Ullman, "Np-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.

[3] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proceedings of the Eighth Heterogeneous Computing Workshop*, ser. HCW '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 30–.

[4] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[5] M. Rahman, R. Hassan, R. Ranjan, and R. Buyya, "Adaptive workflow scheduling for dynamic grid and cloud computing environment," *Concurrency and Computation: Practice and Experience*, pp. n/a–n/a, 2013.

[6] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang, "Cost-efficient task scheduling for executing large programs in the cloud," *Parallel Computing*, vol. 39, no. 4âĂŞ5, pp. 177 – 188, 2013.

[7] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 638–645.

[8] S. Verboven, P. Hellinckx, F. Arickx, and J. Broeckhove, "Runtime prediction based grid scheduling of parameter sweep jobs," in *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, 2008, pp. 33–38.

[9] C. Yan, H. Luo, Z. Hu, X. Li, and Y. Zhang, "Deadline guarantee enhanced scheduling of scientific workflow applications in grid," *Journal of Computers*, vol. 8, no. 4, 2013.

[10] J. O. Gutierrez-Garcia and K. M. Sim, "A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1682 – 1699, 2013.

[11] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682 – 692, 2013, special Section: Recent Developments in High Performance Computing and Security.

[12] O. Sonmez, N. Yigitbasi, A. Iosup, and D. Epema, "Trace-based evaluation of job runtime and queue wait time predictions in grids," in *Proceedings of the 18th ACM international symposium on High performance distributed computing*, ser. HPDC '09. New York, NY, USA: ACM, 2009, pp. 111–120.

[13] D. Martínez-Rego and M. Pontil, "Multi-task averaging via task clustering," in *Similarity-Based Pattern Recognition*, ser. Lecture Notes in Computer Science, E. Hancock and M. Pelillo, Eds. Springer Berlin Heidelberg, 2013, vol. 7953, pp. 148–159.

[14] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[15] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Sci. Program.*, vol. 13, no. 3, pp. 219–237, Jul. 2005.

[16] J. s. VŽckler, G. Mehta, Y. Zhao, E. Deelman, and M. Wilde, "Kickstarting remote applications," in *2nd International Workshop on Grid Computing Environments*, 2006.

[17] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids," in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, ser. SWEET '12. New York, NY, USA: ACM, 2012, pp. 1:1–1:13.

[18] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto, Jr., and H.-L. Truong, "Askalon: a tool set for cluster and grid computing: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 17, no. 2-4, pp. 143–169, Feb. 2005.

[19] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, no. 10, pp. 1067–1100, Aug. 2006.

[20] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su, "Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand," vol. 5493, pp. 221–232, 2004.

[21] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.

[22] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[23] R. Ferreira da Silva, T. Glatard, and F. Desprez, "Self-healing of workflow activity incidents on distributed computing infrastructures," *Future Generation Computer Systems*, p. in press, 2013.

[24] Parallel workloads archive. [Online]. Available: www.cs.huji.ac.il/labs/parallel/workload/

[25] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. J. Epema, "The grid workloads archive," *Future Gener. Comput. Syst.*, vol. 24, no. 7, pp. 672–686, 2008.

[26] C. Germain-Renaud, A. Cady, P. Gauron, M. Jouvin, C. Loomis, J. Martyniak, J. Nauroy, G. Philippon, and M. Sebag, "The grid observatory," *IEEE International Symposium on Cluster Computing and the Grid*, pp. 114–123, 2011.

[27] Workflow gallery. [Online]. Available: http://pegasus.isi.edu/workflow_gallery

[28] R. Ferreira da Silva and T. Glatard, "A science-gateway workload archive to study pilot jobs, user activity, bag of tasks, task sub-steps, and workflow executions," in *Euro-Par 2012: Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, I. Caragiannis, M. Alexander, R. Badia, M. Cannataro, A. Costan, M. Danelutto, F. Desprez, B. Krammer, J. Sahuquillo, S. Scott, and J. Weidendorfer, Eds. Springer Berlin Heidelberg, 2013, vol. 7640, pp. 79–88.

[29] Workflow generator. [Online]. Available: http://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

[30] L. Ramakrishnan and D. Gannon, "A survey of distributed workflow characteristics and resource requirements," *Indiana University*, 2008.

[31] S. Ostermann, R. Prodan, T. Fahringer, A. Iosup, and D. Epema, "On the characteristics of grid workflows," in *CoreGRID Symposium - Euro-Par 2008*, 2008.

[32] ——, "A trace-based investigation of the characteristics of grid workflows," in *From Grids to Service and Pervasive Computing*, T. Priol and M. Vanneschi, Eds. Springer US, 2008, pp. 191–203.

[33] A. Iosup and D. Epema, "Grid computing workloads," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 19–26, 2011.

[34] D. L. Hart, "Measuring teragrid: workload characterization for a high-performance computing federation," *International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 451–465, 2011.

[35] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, "Workload characterization on a production hadoop cluster: A case study on taobao," in *Workload Characterization (IISWC), 2012 IEEE International Symposium on*, 2012, pp. 3–13.

[36] S. Mahambre, P. Kulkarni, U. Bellur, G. Chafle, and D. Deshpande, "Workload characterization for capacity planning and performance management in iaas cloud," in *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, 2012, pp. 1–7.

[37] S. Madougou, S. Shahand, M. Santcroos, B. van Schaik, A. Benabdelkader, A. van Kampen, and S. Olabarriaga, "Characterizing workflow-based activity on a production e-infrastructure using provenance data," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 1931 – 1942, 2013.

[38] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for qos prediction in the cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 147–154.

[39] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 1287–1294.

[40] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer, "A hybrid intelligent method for performance modeling and prediction of workflow activities in grids," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 339–347.

[41] E.-K. Byun, Y.-S. Kee, E. Deelman, K. Vahi, G. Mehta, and J.-S. Kim, "Estimating resource needs for time-constrained workflows," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 31–38.

[42] R. Huang, H. Casanova, and A. A. Chien, "Automatic resource specification generation for resource selection," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, ser. SC '07. New York, NY, USA: ACM, 2007, pp. 11:1–11:11.

[43] F. Nadeem and T. Fahringer, "Using templates to predict execution time of scientific workflow applications in the grid," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 316–323.