Parallel Processing Letters
© World Scientific Publishing Company

ONLINE TASK RESOURCE CONSUMPTION PREDICTION FOR SCIENTIFIC WORKFLOWS

RAFAEL FERREIRA DA SILVA[1], GIDEON JUVE[1], MATS RYNGE[1],
EWA DEELMAN[1], MIRON LIVNY[2]
[1] *University of Southern California, Information Sciences Institute, Marina Del Rey, CA, USA*
[2] *University of Wisconsin Madison, Madison, WI, USA*
{*rafsilva,gideon,rynge,deelman*}*@isi.edu, miron@cs.wisc.edu*

Received June 2015
Revised June 2015
Communicated by J. Dongarra

ABSTRACT

Estimates of task runtime, disk space usage, and memory consumption, are commonly used by scheduling and resource provisioning algorithms to support efficient and reliable workflow executions. Such algorithms often assume that accurate estimates are available, but such estimates are difficult to generate in practice. In this work, we first profile five 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 requirements based on these profiles. Our method estimates task runtime, disk space, and peak memory consumption based on the size of the 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 using a clustering technique. Task estimates are generated based on the ratio *parameter/input data size* if they are correlated, or based on the probability distribution function of the parameter. We then propose an *online* estimation process based on the MAPE-K loop, where task executions are monitored and estimates are updated as more information becomes available. Experimental results show that our online estimation process results in much more accurate predictions than an *offline* approach, where all task requirements are estimated prior to workflow execution.

Keywords: Scientific workflow, workflow characterization, online resource usage task estimation, MAPE-K loop

1. Introduction

Scientific workflows have been used by computational scientists to orchestrate complex simulations and analyses in a wide variety of domains [1]. Workflows enable users to easily express multi-step computational pipelines so that they can be efficiently and reliably executed on distributed infrastructures. Efficient workflow execution depends primarily on how resources are allocated and how tasks are scheduled. However, current algorithms for these problems suffer from the lack of accurate information about the resource requirements of workflows.

Task scheduling, for example, is known to be an NP-complete problem [2]. As a result, many heuristics have been developed over the years to address the allocation of tasks to resources. Heuristics such as Min-min, Max-min [3], HEFT [4], and others [5, 6, 7], have demonstrated the ability to improve the performance and the efficiency of workflow execution. However, all these algorithms assume the existence of accurate estimates for tasks resource requirements such as execution and communication times, disk space, or memory usage. In practice, it is difficult to generate estimates that are simple to compute, require little information about the internal composition of the tasks, and have good accuracy. Many task resource need estimation techniques used for scheduling are based on analytical modeling techniques that require significant effort to develop [8, 9], or statistical techniques that are typically not very accurate [10, 11].

Similarly, resource provisioning techniques benefit from accurate task estimates when determining the optimal number and characteristics of resources required to perform a computation. For instance, when a researcher uses a cloud infrastructure for processing scientific computations, accurate estimates of task requirements can have a significant impact on cost and resource utilization [12]. Ideally, the number and characteristics of resources used should be carefully chosen to minimize cost and maximize resource utilization.

In our previous work [13] we characterized the detailed resource usage of tasks in many real workflow executions. We developed monitoring tools [14, 15] to collect fine-grained resource usage profiles for I/O, runtime, memory usage, and CPU utilization. These tools use techniques that have sufficiently low overhead to use in production environments. In this work, we use these tools to profile five real workflow applications, and we propose, as our first contribution, a method to automatically estimate workflow task requirements such as runtime, disk usage, and memory consumption based on profiling data. Our method currently assumes that these parameters can be estimated based on the input data size, because this is a parameter that could be known in advance, and because the resource requirements of tasks in real workflows often depends on the size of input data [16, 17, 18, 19]. Our method looks for correlations between the size of input data and the desired parameters (runtime, disk usage, memory) by examining historical profiling data from previous executions of the workflow. If no correlation is detected, the profiling data is partitioned using a density clustering technique to identify subsets of the data that behave similarly. These subsets may have a higher correlation coefficient, or a lower standard deviation of the mean value. By repeatedly subdividing the data we generate a regression tree that can be used to estimate values for future executions of the workflow.

Task estimation for scientific workflows differs from the general case of task estimation [20, 21] because of the number of tasks involved in one workflow and the unique effects of dependencies between tasks. To illustrate, a poor estimate for the output of a task estimated before the workflow is submitted for execution (offline), will propagate and grow through the remainder of the workflow by skewing the estimates for the children and grandchildren of the task. To address this issue, we propose an online estimation process based on the MAPE-K loop (Monitoring, Analysis, Planning, Execution, and Knowledge) [22], where task executions are constantly monitored and estimations are updated upon task completion. This online process automatically corrects estimation errors as the workflow is running and prevents them from growing and spreading.

The main contributions of this paper are:

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

The characterization and estimation methods were introduced and evaluated on three real scientific workflows in [23]. In this paper, we extend our previous work by studying 1) the influence of workflow parameters for the characterization of scientific workflows; and 2) the advantage of using probability distribution functions to estimate task requirements instead of using simple mean values. In addition, two new workflow applications were added to the experiments, and the number of different datasets was increased from 3 to 10 for the Montage workflow, and from 3 to 6 for the Epigenomics workflow.

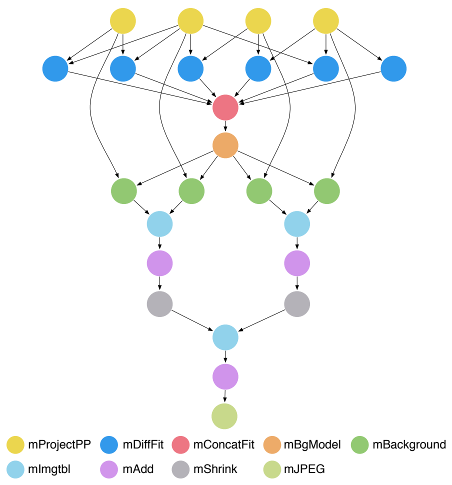
This paper is organized as follows. Section 2 describes the real scientific workflow applications used in this work. In Section 3, we present execution profiles of these workflows, and we introduce our automated method to characterize workflow executions. Our online task estimation process is presented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

2. Scientific workflows

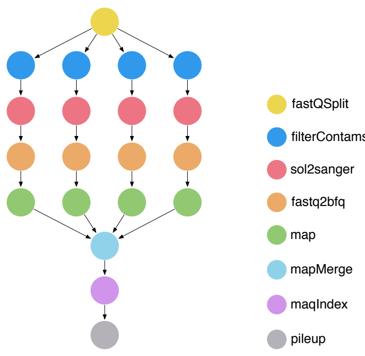
A scientific workflow describes a set of computational tasks and the dependencies between them. In many cases, workflows can be described as a directed acyclic graph (DAG), where the nodes represent tasks and the edges represent dependencies. This model (among others) is supported by several workflow management systems (WMS), including Pegasus [24, 25], Makeflow [26], Askalon [27], and Taverna [28].

In this work, we have used the following real scientific workflows:

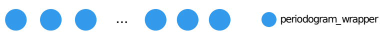
Montage. The Montage workflow [29] was created by the NASA Infrared Processing and Analysis Center (IPAC) as an open source toolkit that can be used to generate custom mosaics of astronomical images in the Flexible Image Transport System (FITS) format. In a Montage workflow, the geometry of the output mosaic 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. Fig. 1.a illustrates a small (20 node) Montage workflow.



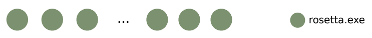
(a) Small (20 node) Montage workflow



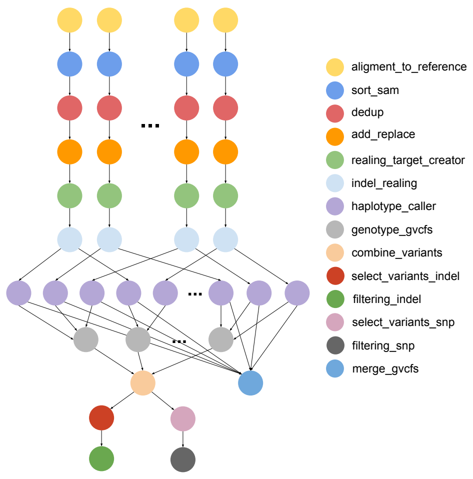
(b) Epigenomics



(c) Periodogram workflow



(d) Rosetta workflow



(e) SoyKB workflow

Fig. 1. Simplified visualization of the scientific workflows.

The size of the workflow depends on the number of images required to construct the desired mosaic. The workflow instances used in this paper generate 2, 4, or 8 degree square mosaics respectively using 275, 840, and 2640 input images from the 2 Micron All Sky Survey (2MASS).

Epigenomics. The USC Epigenome Center [30] is mapping the epigenetic state of human cells on a genome-wide scale. The Epigenomics workflow (Fig. 1.b) processes multiple sets of genome sequences in parallel. These sequences are split into subsets, the subsets are filtered to remove contaminants, reformatted, then mapped to a reference genome. The mapped sequences are finally merged and indexed for later analysis. In this paper, the Epigenome workflow was used to align 13 million genome sequence reads to a reference genome for human chromosome 21. The size of the workflow depends on the chunking factor used on the input data (`binsize`), which determines the number of sequence reads in each chunk.

Periodogram. The Periodogram workflow [31] searches for extra-solar planets, either by detecting "wobbles" in the radial velocity of a star, or dips in the star's intensity caused by orbiting and transiting planets. The workflow searches through time series datasets of star brightness called "light curves" to identify periodic variations in the brightness or color of the star caused by these orbiting planets. Fig. 1.c shows an illustration of a Periodogram workflow. The workflow is a simple bag-of-tasks consisting of independent `periodogram_wrapper` tasks. In this paper, we ran the Periodogram workflow on a dataset from NASA's Kepler mission consisting of 28,872 light curve files.

Rosetta. The Rosetta workflow [32] computes the energies of interactions within and between macromolecules to find the lowest energy structure for an amino acid sequence (protein-structure prediction) or a protein-protein complex, and to find the lowest energy amino acid sequence for a protein or protein-protein complex (protein design). Fig. 1.d shows an illustration of the Rosetta workflow. The workflow can be seen as a bag-of-tasks of `rosetta.exe` tasks, where each task can execute different search strategies. The workflow instances used in this work were composed of 20, 30, and 50 Protein Data Banks (PDBs).

SoyKB. The SoyKB workflow [33, 34] is a genomics pipeline that re-sequences soybean germplasm lines selected for desirable traits such as oil, protein, soybean cyst nematode resistance, stress resistance, and root system architecture. The workflow (Fig. 1.e) implements a SNP and injection/deletion (indel) identification and analysis pipeline using the GATK haplotype caller [35] and a soybean reference genome. The workflow analyzes samples in parallel to align them to the reference genome, to de-duplicate the data, to identify indels and SNPs, and to merge and filter the results. The results are then used for genome-wide association studies (GWAS) and genotype to phenotype analysis. The workflow instance used in this paper is based on a sample dataset of 50 sequence reads that requires less memory than a full-scale production workflow.

3. Workflow characterization

In this section, we characterize execution profiles for the scientific workflows described in the previous section by using the Kickstart [14, 15] profiling tool, and the Pegasus Workflow Management System (WMS) [24, 25].

Kickstart monitors and records information about the execution of individual workflow tasks. 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. This promotes workflow portability and reuse while enabling Pegasus to perform workflow optimizations, such as task clustering. Pegasus also intelligently manages workflow data by inferring the data transfers required by a workflow, by reusing data when available, and by registering data into information catalogs for later processing. In addition, Pegasus monitors the workflow using Kickstart and captures performance, resource usage, and provenance information into a database.

Runs of each workflow were performed with different datasets and input parameter options. For the Montage workflow, 10 datasets were used, and the `degree` input parameter was set to 2, 4, and 8 degrees. 6 datasets were used for the Epigenomics workflow, and the `binsize` input parameter was varied according to $10^n, n \in [3,5]$. 3 different input datasets were used for each of the other workflows. For each dataset and parameter value configuration, we conducted 5 runs of the workflow to ensure a 95% confidence interval of the measurements.

Workflows were executed on a 16-core cluster, composed of 5 dual core, 2.4 GHz AMD Opteron$^{\text{TM}}$ 250 processors with 8GB of RAM, and 3 dual core, 2.2 GHz AMD Opteron$^{\text{TM}}$ 275 processors with 8GB of RAM. We used a homogeneous cluster since our method does not yet consider resource characteristics to estimate task runtimes.

3.1. *Sensitivity analysis of input parameters*

In order to understand the effect of input parameters on task behavior, we examined workflow-level inputs for the Montage and Epigenomics workflows. We conducted a sensitivity study of the `degree` parameter for the Montage workflow and the `binsize` parameter for the Epigenomics workflow. These parameters were the only ones that we were able to change given the datasets that were available to us. For Montage workflows, the `degree` input parameter changes the size and structure of the workflow and the overall makespan of the workflow, but has no influence on the runtime, I/O read and write, and memory peak values of individual tasks in the workflow. As shown in Fig. 2 (left), the probability density functions (PDF) of the runtime parameter for different values of `degree` are nearly the same. For Epigenomics workflows, the `binsize` input parameter also affects the size and shape of the workflow, but only slightly influences tasks with very short runtimes (Fig. 2-right), or tasks that produce small outputs. Since the magnitude of these variations is very small, when compared to the magnitude of the overhead of scheduling such tasks, it is considered negligible. We observed no influence of `binsize` on memory usage. As a result of this analysis, these input parameters are not considered to be a major factor when characterizing these workflows. In future work we plan to identify other workflows, data sets, and execution platforms that will enable us to investigate how

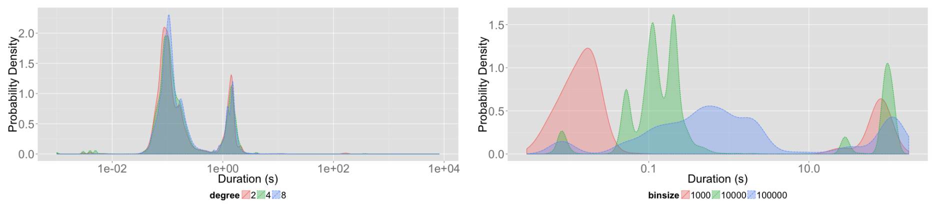


Fig. 2. Sensitivity analysis of the runtime parameter for the Montage (left) and Epigenomics (right) workflows.

different input parameters, environment configurations, and resource characteristics affect task behavior, and how those factors can be incorporated into our estimation techniques.

3.2. *Estimating target parameters based on the mean*

One simple approach for estimating the runtime, I/O and memory of future workflow tasks is to use the mean values of previous executions of similar tasks. Table 1 shows the execution profile for 15 runs of the SoyKB workflow, grouped by task type (executable). A few task types have small standard deviation values in comparison to the mean, thus task estimates could be based on mean values. However, for high standard deviation sets (e.g., runtime of `genotype_gvcfs`, and memory peak of `sort_sam`) task estimation based on the mean may lead to significant estimation errors. Most of the I/O write and memory peak tasks have small standard deviation values compared to the mean, thus the estimation of these requirements based on the mean value would yield reasonable accuracy. On the other hand, runtime standard deviation values are too high to have reasonable accuracy using the mean.

Table 1. Example of an execution profile of the SoyKB executions on a distributed platform.

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.
alignment_to_reference	255	18.07	13.81	4508.41	650.43	616.81	0.01	1873.61	14.73
sort_sam	255	1.84	1.55	322.06	87.20	471.65	133.11	1135.35	1243.36
dedup	255	2.94	2.10	399.04	51.97	591.13	79.55	2243.14	740.86
add_replace	255	1.83	1.25	350.79	83.09	518.39	129.34	1567.54	1205.49
realign_target_creator	255	347.20	106.11	2578.02	487.53	606.50	51.88	2451.24	492.60
indel_realign	255	11.46	22.84	431.45	81.71	575.54	97.24	2144.15	918.54
haplotype_caller	5,100	117,82	62.23	637.24	83.64	606.27	52.02	2441.65	493.40
genotype_gvcfs	300	801.09	588.51	536.58	90.06	595.62	74.01	2338.63	700.35
combine_variants	15	14.71	0.77	468.56	0.85	616.83	0.03	2550.55	36.22
select_variants_indel	15	71.13	4.68	935.50	1.79	1101.29	0.43	2564.81	61.65
filtering_indel	15	13.05	0.22	465.71	0.04	616.91	0.01	2551.17	37.98
select_variants_snp	15	72.71	1.61	934.83	0.96	1101.36	0.50	2528.77	1.84
filtering_snp	15	12.95	0.29	465.80	0.03	616.94	0.02	2526.00	2.19
merge_gcvf	15	19352.8	31081.1	2391.08	2080.98	431.88	375.03	2546.65	60.82

3.3. *Correlation between input data size and target parameters*

In many cases, standard deviations of the target parameters are not small enough to justify the use of mean values for estimating those parameters. In those cases, we assume that the target parameters, including runtime, I/O write, and memory, can be estimated based on the I/O read parameter, which is a reasonable proxy for input data size. This assumption is based on observed correlations between input data size and task requirements from previous research [16, 17, 18, 19] and observations of the data collected for this paper. This assumption is not unreasonable. Input data is typically read into in-memory data structures for processing, therefore there is often a correlation between memory use and input size. Similarly, output data size may be correlated with input data size when a task is applying a transformation to the input data, or it may have a constant size, for example, when the output data is a summary of the input data. Finally, runtime is often correlated with input data size when the task performs a set of computations for each value in the input data. Experiment results presented in Section 5 support this assumption.

The workflow data was partitioned by workflow application, and task type. For each partition, correlation statistics were computed to identify statistical relationships between the I/O read parameter and the target parameters (runtime, I/O write, and memory). Table 2 shows the Pearson's correlation (ρ) values for each task type of the Epigenomics and Rosetta workflows . We consider two parameters to be correlated if the absolute value of their correlation value, ρ, is greater than 0.8. The threshold value of 0.8 was selected somewhat arbitrarily after observing the data. Correlated parameters, highlighted in the table, are expected to yield accurate estimates using a simple linear relationship between the parameters.

Table 2. Correlation (ρ) values for the scientific workflows. Highlighted cells indicate high correlation values to the I/O read parameter.

Task	ρ		
	Runtime	I/O Write	Memory Peak
fastqSplit	0.43	0.99	0.00
filterContams	0.91	0.99	0.09
sol2sanger	0.93	0.99	0.10
fast2bfq	0.79	0.99	0.13
map	0.73	0.27	0.99
mapMerge	0.98	0.99	-0.16
pileup	0.72	0.99	0.99

(a) Epigenomics workflow.

Task	ρ		
	Runtime	I/O Write	Memory Peak
rosetta.exe	-0.05	0.15	0.06

(b) Rosetta workflow.

3.4. *Improving Correlations Using Density-Based Clustering*

Only about 51% of the parameters show high enough correlation values to be useful for estimation. For the other parameters, further processing is required to identify subsets of tasks that have higher predictability. We used density-based clustering [36] to identify high-density areas in the datasets where weak correlations were identified (e.g., Fig. 3). The goal of this clustering is to partition the dataset into smaller subsets that have higher correlation coefficients, or lower standard deviations, for the target parameters.

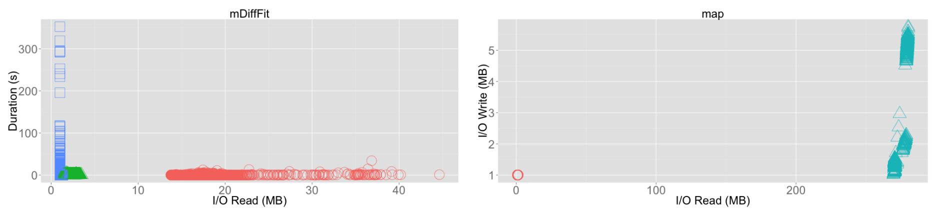


Fig. 3. Dataset clustering of runtime (Montage) and I/O write (Epigenomics) parameters.

We use DBSCAN [37] (density-based spatial clustering of applications with noise) as the clustering algorithm. DBSCAN was chosen because it is one of the most commonly used density-based clustering algorithms 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 the x axis represents the I/O read parameter value, and the y axis represents the target parameter value (runtime, I/O write, memory). For instance, Fig. 3 shows the dataset clustering of the runtime parameter for `mDiffFit` tasks of the Montage workflow, and I/O write parameter for `map` tasks of the Epigenomics workflow. In these datasets, clusters of related data 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 the plot shows a strong bend.

Correlation (ρ) and standard deviation (σ) values, and clusters (c) per task type are shown in Table 3. Datasets with high correlation values were not clustered (highlighted cells in Table 2), for example the I/O write parameter of `filterContams` (Epigenomics). Otherwise, subsets such as the runtime parameter of `periodogram_wrapper` (Table 3.b), have higher correlation values or smaller standard deviation values. In clusters where the correlation is null and the standard deviation is zero, the data is concentrated in a unique point, i.e. the parameter is a constant value independent of the workflow input dataset. Note that even though

Algorithm 1 DBSCAN algorithm.

```
inputs: $D$ dataset, $eps$, $minPts$
cluster $C = 0$
for $p \in D$ 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 $p' \in$ neighborPts do
    if $p'$ is unvisited then
        mark $p'$ as visited
        neighborPts' = regionQuery($p'$, $eps$, $D$)
        if neighborPts'.size $\geq minPts$ then
            neighborPts = neighborPts $\cup$ neighborPts'
        end if
    end if
    if $p' \notin$ any cluster then
        add $p'$ to $C$
    end if
end for

regionQuery ($p$, $eps$, $D$)
    return $D' \subseteq D$, where $distance(p, q) \leq eps$, $q \in D'$
```

the standard deviation value is shown as zero for some clusters shown in Table 3, the correlation value is defined. This is due to very small non-zero values of standard deviation. This is observed for most memory peak parameter of the Epigenomics workflows (Table 3.a). After clustering, some datasets have lower correlation values than before, but also have lower standard deviation values. Thus, task estimation errors based on the mean values are smaller. For datasets where the correlation values of the subsets are smaller and the standard deviation increases, our method discards the subsets and keeps the original dataset (e.g., `alignment_to_reference` for the SoyKB workflow). The `genotype_gvcfs` task type (SoyKB workflow) has the largest number of subsets. Two of the subsets have weak correlation values, while strong correlations are found for the two other subsets.

4. Task estimation process

Fig. 4 illustrates our estimation process for one parameter. This process is based on regression trees. The tree is constructed offline based on the analysis of the workflow characterizations presented in the previous section. First, tasks are classified by application (workflow), then by task type. The next step decides whether runtime, I/O write, or memory requirements should be estimated based on the input data size (I/O read). Note that a set of clustered datasets (Tables 3) is associated for each parameter. A cluster is selected according to the I/O read value of the task being estimated. If the parameter is strongly correlated to the input data within a cluster,

Table 3. Clustered datasets: 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
fastqSplit	1	-1.00	0.66	1	0.99	338.09	1	-	0.00
	2	0.84	6.87						
filterContams	1	0.91	0.45	1	0.99	6.09	1	0.09	0.02
sol2sanger	1	0.93	0.09	1	0.99	7.77	1	0.62	1.61
							2	-	0.00
fast2bfq	1	0.79	0.21	1	0.99	1.98	1	0.82	5.52
							2	-	0.00
map	1	0.73	38.67	1	0.93	0.77	1	0.99	64.10
				2	0.20	0.01			
mapMerge	1	0.98	10.99	1	0.99	184.02	1	0.42	9.11
							2	0.99	0.03
pileup	1	0.72	51.28	1	0.99	2347.7	1	0.99	53.57

(a) Epigenomics workflow.

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

(b) Periodogram workflow.

Task	Runtime			I/O Write			Memory Peak		
	c	ρ	σ	c	ρ	σ	c	ρ	σ
alignment_to	1	0.23	9.4	1	-0.06	0.1	1	-0.17	14.7
_reference	2	0.08	19.3	2	0.31	0.1			
	3	0.95	5.8						
sort_sam	1	0.86	1.8	1	0.99	133.1	1	0.99	1243.4
	2	0.21	1.2						
dedup	1	0.56	2.2	1	0.98	79.5	1	0.98	740.9
	2	0.76	0.1						
add_replace	1	0.59	1.5	1	0.99	129.3	1	0.99	1205.5
	2	0.51	0.1						
realign_target	1	-0.23	109.4	1	0.95	51.9	1	0.95	492.7
_creator	2	0.17	54.2						
indel_realign	1	0.07	24.8	1	0.99	97.2	1	0.99	918.5
	2	0.53	4.3						
haplotype_caller	1	0.26	64.4	1	0.96	52.0	1	0.96	493.4
	2	0.81	56.3						
genotype_gvcfs	1	0.35	588.5	1	0.98	74.0	1	0.98	700.3
	2	0.28	26.7						
	3	0.97	67.3						
	4	0.81	209.0						
combine_variants	1	0.82	0.77	1	0.98	0.4	1	0.99	36.2
				2	0.01	0.1			
select_variants_indel	1	-0.98	4.7	1	0.84	0.4	1	-0.84	61.6
filtering_indel	1	-0.88	0.2	1	0.91	0.1	1	0.98	38.0
select_variants	1	-0.98	1.6	1	0.99	0.5	1	0.98	2.5
_snp							2	0.01	0.1
filtering_snp	1	-0.98	0.4	1	0.62	0.1	1	-0.60	2.19
	2	0.01	0.1						
merge_gcvf	1	0.98	0.00	1	0.98	0.00	1	0.01	60.8

(c) SoyKB workflow.

values are estimated according to the ratio *parameter/input data size*. Otherwise, our estimation process performs a Kolmogorov-Smirnov goodness of fit test (K-S test) [38] to determine whether the parameter distribution of the subset fits a probability distribution. We conduct K-S tests where the null hypothesis H_0 is that the subset fits a Normal or a Gamma distribution. For subsets that do not fit one

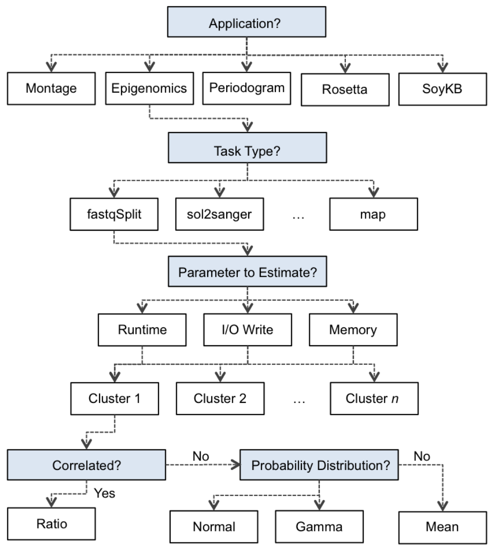


Fig. 4. Example of the estimation process for one parameter of the Epigenomics workflow.

of the two distributions, values are estimated according to the mean. The process outputs rules used to estimate future workflow executions. Fig. 5 shows examples of rules to estimate I/O write for the `rosetta.exe` task (Rosetta workflow).

```
if $workflow$ = ‘$Rosetta$’ and $taskType$ = ‘$rosetta.exe$’ and $parameter$ = ‘$write$’
  and $input\_size > 24815452$ then
    return [0.225, $ratio$] // ratio of output and input data
end if

if $workflow$ = ‘$Rosetta$’ and $taskType$ = ‘$rosetta.exe$’ and $parameter$ = ‘$write$’
  and $input\_size \leq 24815452$ then
    return [$gamma$, 5597415, 3953.37] // mean and standard deviation values for distribution
end if
```

Fig. 5. Rules for I/O write estimation of the Rosetta workflow.

Although our method requires significant amount of time (∼6 hours) to build the regression trees (which are built prior to the workflow execution), the overhead produced by our estimation process is negligible since it only parses rules to perform the predictions. As a result, this process can easily be used in practice by real workflow management systems.

Generating estimates that match a probability distribution. The estimation process uses the Marsaglia’s polar method [39] to generate values from a Normal distribution, and the Marsaglia and Tsang method [40] to generate values from a Gamma distribution.

Marsaglia’s polar method transforms from a two-dimensional continuous distribution to a two-dimensional bivariate normal distribution. The method chooses random points (u, v), uniformly and independently distributed $[-1, +1]$, until $s < 1$ $(s = u^2 + v^2)$. Then, x and y have a normal distribution with mean μ (defined as the median value of the probability density function) and variance σ^2 (where σ is standard deviation of the probability function) as follows:

$$x = \mu + \sigma \cdot \sqrt{\tfrac{-2 \ln s}{s}}, \qquad y = \mu + \sigma \cdot \sqrt{\tfrac{-2 \ln s}{s}} \tag{1}$$

When the prediction method asks for an estimate, x is returned and y is kept as a spare value for the next invocation.

The method for generating a Gamma variate assumes that methods for generating values from a standard Normal and Uniform distributions are available. Then, a Gamma variate w can be generated as $d \cdot v$, defined as:

$$d = \alpha - \tfrac{1}{3}, \qquad v = \left(1 + \tfrac{x}{\sqrt{9 \cdot d}}\right)^3 \tag{2}$$

where x is a generated value from a Normal distribution. Note that this method generates Gamma variates for a scale parameter $\theta = 1$. Therefore, we approximate w to $w' = \theta' \cdot d \cdot v$, where $\theta' = \frac{1}{\beta}$ and β is the rate parameter.

Generated estimates are then compared to the real values to determine the accuracy of the estimation method. If the generated estimates are not accurate, the probability distribution is then updated with the real value, thereby the probability of generating estimates for this real value is decreased. In addition, for skewed distributions (which is the case for most of Gamma distributions in our dataset), predictions based on the median value of the distribution would yield poor accuracy.

Offline and online estimation. An *offline* task estimation approach estimates the runtime, output data size (I/O write), and memory usage for all tasks in a workflow before any of the tasks have been executed. Since the outputs of each task in a workflow become inputs to subsequent tasks, and we use input size to estimate all the target parameters, poor output data size estimates for tasks at higher levels of the workflow may lead to a chain of increasing estimation errors for tasks at subsequent levels. Therefore, in addition to the *offline* estimation process, we also propose an *online* estimation process based on the MAPE-K loop (Monitoring, Analysis, Planning, Execution, and Knowledge), where task executions are constantly monitored [41, 42]. Upon task completion, estimated values for the task are updated with the real values, and, based on these values, a new prediction is generated (using the regression tree of Fig. 4) for subsequent tasks that have data dependencies from the completed task. Fig. 6 summarizes the online estimation process. Note that in a workflow, tasks may have multiple parents, thus at any given time their input data size may be a combination of estimated and real (for completed parent tasks) values.

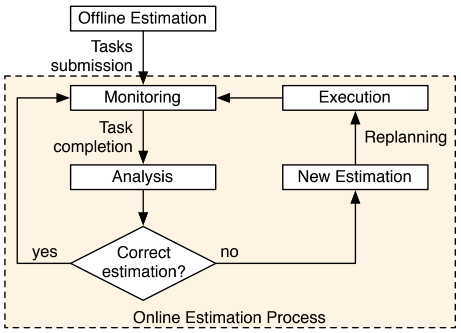


Fig. 6. Online estimation process.

5. Experiment and evaluation

Experiments were conducted to evaluate the accuracy of the online estimation process in comparison to the offline estimation process. In addition, we evaluate the benefit of using probability distribution functions in the estimation process instead of just the mean values as used in [23].

5.1. *Experiment conditions*

Trace analyzes were performed for the five scientific workflow applications described in Section 3: Montage, Epigenomics, Periodogram, Rosetta, and SoyKB. For each workflow, we conducted a leave-one-out cross-validation using the datasets used to characterize the workflows: one dataset is used to test the accuracy of the estimation process, while the others are reserved for training purposes (generation of rules). For each execution, we performed an analysis to test the accuracy of the prediction–results presented in the next subsection are an average of these analyses. We assume that a parameter is statistically correlated if its correlation coefficient ρ is greater than or equal to 0.8. Otherwise, K-S tests are conducted to determine whether the subset fits a Normal or Gamma distribution, or none of them.

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. For the online estimation process, we implemented both the process proposed in [23], named `Online-m`, and the estimation process proposed in this work, named `Online-p`, which uses probability density functions in addition to mean values.

Our analysis consists of replaying a workflow execution from its execution trace and estimating task parameters using both the offline and online approaches. Replaying a workflow execution trace means that our simulator processes each task in the same order as the real execution, using the same delays and task behaviors. The simulator estimates task requirements and compares them to the real values to assess estimation errors. We do not aim to evaluate the efficiency of the scheduling algorithms used in the real execution, but rather the accuracy of our estimation processes.

5.2. *Results and discussion*

Table 4 shows the average estimation error for the Montage workflow. The `Offline` process yields an average estimation error of 49% for runtime, 55% for I/O write, and 57% for memory peak, while the `Online-m` process yields average errors of 20%, 11%, and 14%, and the `Online-p` process yields average errors of 13%, 8%, and 11%. For the first-level tasks (`mProjectPP`) all process have nearly the same accuracy as tasks are estimated directly from the workflow input data. Note that the average runtime error for `Online-m` is higher due to the estimation based on the mean value. `Online-m` and `Online-p` produce the same estimate error rate for estimates based on the correlation (e.g., I/O write and memory peak for `mDiffFit`). Offline estimations for tasks such as `mDiffFit`, `mBackground`, `mImgtbl`, and `mAdd` are greatly affected by the propagation of estimation errors. For instance, the input data of a `mDiffFit` task are multiples `mProjectPP` output data. A bad estimation of the input data size may lead the process to select the wrong cluster. The online process initially faces the same issue of erroneous estimations, but upon task completion, wrong predictions are replaced by the actual value. In most cases, `Online-p` yields better accuracy than `Online-m`, since the subsets often fit Gamma distributions for runtime, and Normal distributions for I/O write and memory peak requirements. `Online-m` and `Online-p` have the same performance when the subsets are grouped into a single point (e.g., runtime for `mJPEG`), since both are estimated from the mean value.

Table 5 shows the average estimation error for the Epigenomics workflow. The average estimation errors for the `Offline` process are 30% for runtime, 56% for I/O write, and 46% for memory. For the `Online-m` process the errors are 14%, 5%, and 9%. And for the `Online-p` process the errors are 13%, 5%, and 8%. Similarly to Montage, first-level task estimations (`fastqSplit`) are nearly the same for all approaches. `Offline` estimates for `filterContams`, `sol2sanger`, `fast2bfq`, and `mapMerge` are significantly affected by the estimation errors of their parent tasks. Since most of the parameters are strongly correlated with input data size (Table 2.a), the `Online-p` process has no significant gain over `Online-m`. Also, most of the subsets that are not correlated, do not fit any of the two distributions, thereby the mean is used for prediction.

Table 6.a presents average estimation errors for the Periodogram workflow. As the workflow has only one task level (`periodogram_wrapper`, see Fig. 1.c), the online approaches produce the same result as `Offline` for runtime and memory peak requirements. For the I/O write parameter, the non-correlated subset fits a Normal distribution. As a result, `Offline` and `Online-p` yield better accuracy than `Online-m`.

Similarly, `Online-p` yields better runtime accuracy for Rosetta workflows (Ta-

Table 4. 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 (%)
mProjectPP	Offline	19.59	1.97	4.03
	Online-m	20.78	1.97	4.03
	Online-p	19.59	1.97	4.03
mDiffFit	Offline	201.62	172.64	103.21
	Online-m	46.22	50.41	53.27
	Online-p	41.25	50.41	53.27
mConcatFit	Offline	9.91	17.46	21.77
	Online-m	7.13	12.10	18.41
	Online-p	4.33	8.88	11.29
mBgModel	Offline	21.32	33.09	22.08
	Online-m	1.98	26.57	3.43
	Online-p	1.98	13.99	3.43
mBackground	Offline	69.35	97.17	104.62
	Online-m	48.36	1.41	1.84
	Online-p	47.88	1.41	1.84
mImgtbl	Offline	69.96	111.19	131.77
	Online-m	29.95	7.47	9.11
	Online-p	18.01	5.29	9.11
mAdd	Offline	29.46	116.60	107.07
	Online-m	18.12	3.82	10.22
	Online-p	8.99	3.82	10.22
mShrink	Offline	15.91	27.36	28.75
	Online-m	8.44	15.06	12.15
	Online-p	4.11	3.97	5.23
mJPEG	Offline	1.96	8.29	23.89
	Online-m	1.41	1.87	12.03
	Online-p	1.41	1.11	12.03

Table 5. Epigenomics: 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	10.11	4.76	0.95
	Online-m	10.11	4.76	0.95
	Online-p	10.11	4.76	0.95
filterContams	Offline	23.87	10.68	62.38
	Online-m	12.20	5.43	9.39
	Online-p	12.20	5.43	4.21
sol2sanger	Offline	59.34	84.16	76.90
	Online-m	14.42	1.48	2.33
	Online-p	14.42	1.48	2.17
fast2bfq	Offline	31.74	82.33	93.40
	Online-m	19.02	18.77	11.26
	Online-p	14.96	18.77	11.26
map	Offline	21.73	20.01	24.12
	Online-m	5.38	4.28	3.99
	Online-p	4.85	4.02	3.99
mapMerge	Offline	51.55	103.00	43.04
	Online-m	12.48	9.91	3.28
	Online-p	12.48	9.91	1.09
pileup	Offline	17.90	6.14	29.66
	Online-m	5.54	1.82	2.15
	Online-p	4.83	1.82	2.15

ble 6.b), since one of the subsets fits a Normal distribution. All processes result in high accuracy for I/O write and memory peak, since the parameters have nearly

Table 6. 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 (%)
periodogram_ wrapper	Offline	45.13	9.29	1.02
	Online-m	45.13	16.72	1.02
	Online-p	45.13	9.29	1.02

(a) Periodogram workflow.

Task	Estimation	Runtime Avg. Error (%)	I/O Write Avg.Error (%)	Memory Avg.Error (%)
rosetta.exe	Offline	11.79	1.89	1.68
	Online-m	35.08	1.89	1.68
	Online-p	11.79	1.89	1.68

(b) Rosetta workflow.

constant values.

Table 7 shows the average estimation error for the SoyKB workflow. The average estimation error for runtime, I/O write, and memory peak is 61%, 49%, and 44% for `Offline`, 20%, 6%, and 7% for `Online-m`, and 12%, 6%, and 7% for `Online-p`. Like Montage, most of the subsets for runtime that are not correlated fit a Gamma distribution. Subsets for I/O write and memory peak have very low standard deviation values, thus the estimation based on the mean yields high accuracy for both `Online-m` and `Online-p` (e.g., `alignment_to_reference` and `merge_gvcf`).

In the analyses for all 5 scientific workflows, the online processes are more accurate than the offline process. In particular, the use of probability distributions significantly improves the accuracy of the predictions, notably for task runtimes. The importance of using a loop to constantly monitor task executions to update estimations is emphasized in workflows due to their task dependency model. Although the online strategy counterbalances the propagation of estimation errors, the estimation of first-level tasks still 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, or the quality of the collected dataset (e.g., failures, time window, etc.).

6. Related Work

Workload archives are widely used for research in 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 [43], the Grid Workload Archive [44], the Failure Trace Archive [45], and the Grid Observatory [46], 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 sub-steps, and artifacts introduced by

Table 7. SoyKB: 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 (%)
alignment_to _reference	Offline	14.73	22.98	10.34
	Online-m	17.31	22.98	10.34
	Online-p	14.73	22.98	10.34
sort_sam	Offline	28.02	19.31	15.50
	Online-m	21.44	4.16	2.65
	Online-p	13.97	4.16	2.65
dedup	Offline	35.11	29.66	21.41
	Online-m	18.76	6.09	5.77
	Online-p	10.01	6.09	5.77
add_replace	Offline	59.55	29.35	25.84
	Online-m	22.14	5.98	4.08
	Online-p	9.08	5.98	4.08
realign_target _creator	Offline	63.22	31.04	40.69
	Online-m	31.18	8.57	10.15
	Online-p	27.83	8.57	10.15
indel_realign	Offline	51.02	20.92	37.41
	Online-m	29.47	3.78	7.09
	Online-p	18.15	3.78	7.09
haplotype _caller	Offline	103.77	94.17	76.23
	Online-m	28.39	7.90	8.44
	Online-p	14.06	7.90	8.44
genotype_gvcfs	Offline	88.50	44.11	51.98
	Online-m	21.96	4.99	5.53
	Online-p	7.14	4.99	5.53
combine _variants	Offline	22.27	30.53	18.34
	Online-m	8.44	5.16	3.10
	Online-p	8.44	5.16	3.10
select_variants _indel	Offline	17.89	16.45	22.32
	Online-m	3.12	9.02	10.43
	Online-p	3.12	9.02	10.43
filtering_indel	Offline	15.70	12.70	10.95
	Online-m	5.86	2.77	3.49
	Online-p	5.86	2.77	3.49
select_variants _snp	Offline	18.01	14.43	24.70
	Online-m	3.03	1.86	10.41
	Online-p	3.03	1.86	10.41
filtering_snp	Offline	13.45	28.14	37.08
	Online-m	2.93	7.29	18.16
	Online-p	2.93	7.29	18.16
merge_gvcf	Offline	37.30	42.68	49.99
	Online-m	4.91	2.04	1.88
	Online-p	4.91	2.04	1.88

application-level scheduling. Therefore, some efforts have been made to collect and publish traces and performance statistics for real scientific workflows. We recently published traces for a few workflows executed using Pegasus [47], and traces of several workflow executions obtained from a science-gateway [48]. We also published synthetic workflows based on statistics from real applications for use in simulations [47, 49]. Similarly, Ramakrishnan and Ganon [50] have provided information for many real workflow applications, and Ostermann et al. [51, 52] have provided analyses of workflow-based workload traces from the Austrian grid.

With respect to workload characterization in distributed environment, Iosup and Epema [53] and Hart [54] presented analyses of grid and HPC workload characteristics including system usage, user population, application characteristics, and characteristics of grid-specific application types. Ren et al. [55] 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. [56] identified cloud workload patterns based on behavioral characteristics and presented statistical techniques to understand them. They categorized the virtual machine workload with respect to the following patterns: periodicity, threshold, relationship, variability, and image similarity. Madougou et al. [57] provided a characterization of workflow executions using provenance data captured from a workflow management system. They analyzed usage and failure patterns at the 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. [10] presented GIPSy, a parameter sweep prediction framework that 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 yields good accuracy, but is not applicable to an online environment. Sonmez et al. [20] 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 with scheduling techniques that do not use prediction show that the use of these techniques does not imply better performance of the grid scheduling. Pacheco-Sanchez et al. [58] 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 series. Khan et al. [59] also proposed a method to characterize and predict workloads 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 common workload patterns. A method based on Hidden Markov Modeling is used to capture temporal correlations and to predict the changes of workload pattern. The use of Markov-based techniques provides good accuracy for workload prediction, but it adds a significant overhead to the application execution. In this work, our method builds regression trees ahead of the workflow execution, thus the overhead to perform the predictions is negligible. Recently, we performed an exploratory analysis of an HTC workload using the statistical recursive partitioning method and conditional inference trees to identify patterns that characterize particular behaviors of the workload [60].

On workflow estimation, Duan et al. [61] proposed a hybrid Bayesian neural network method for modeling and predicting execution time of workflow activities on grids. Contrary to this work, they use resource characteristics 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. [62] and Huang et al. [63] proposed heuristics and models to estimate the number of resources required to execute a workflow, but they assume task runtimes are available. Nadeem and Fahringer [64] 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. Recently, Pallipuram et al. [65] proposed a distribution-based frequency analysis tool to benchmark DAG-based workflows to estimate modeling window sizes to predict resource behavior on non-dedicated resources. Although their technique yields high levels of accuracy, it is limited to homogeneous DAGS, where all the tasks execute the same type of computation. Chirkin et al. [66] proposed an algorithm to estimate task runtimes for workflow scheduling based on distribution function models. The model was evaluated with simple pipelines composed of a few sequential tasks. For complex workflows, such as the ones used in this work, restricting the task estimation only for distribution functions would yield poor results, since some task resource need estimates do not often fit a distribution function. Pietri et al. [67], proposed a model to estimate the makespan of scientific workflows for a given number of resources. However, task runtimes are assumed to be generated from traces of past executions.

7. Conclusion

We presented a method for online estimation of fine-grained task requirements such as runtime, disk usage, and memory consumption in scientific workflows. We profiled five real scientific workflows using execution traces and defined a process to automatically predict task requirements based on these profiles. We assume that task requirements can be estimated based on the size of the input data. Our process looks for correlations between the target task parameters and the input data size. If no strong correlation is found, density-based clustering is performed to identify groups of high density areas. Smaller groups may have higher correlation, lower standard deviation values, or may fit a probability distribution. Then, we defined an online process, based on the MAPE-K loop, to estimate task requirements according to workflow execution profiles.

The method was evaluated using a set of workflow execution traces where the accuracy of our process was measured in comparison with the real value. We also compared the accuracy of the online method proposed in this work against an offline estimation process (all tasks of a workflow are estimated at once), and the online estimation process proposed in [23]. Experimental results show that our estimation process often produces more accurate estimates than the other methods. In addition, we showed that poor estimates of output data size lead to a chain of estimation errors in scientific workflows. This motivates the use of an online strategy where task executions are constantly monitored and estimates are updated as new data becomes available. In the future, we plan to analyze the impact of re-planning a workflow when using an online estimation strategy. We also plan to conduct a sensitivity analysis of the correlation value ρ. Finally, we intend to incorporate the estimation process to the Pegasus workflow management system.

Acknowledgments

This work was funded by DOE under the contract number ER26110, "dV/dt - Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science". We also thank Weiwei Chen for his valuable help.

References

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

[2] J. D. Ullman, Np-complete scheduling problems, J. Comput. Syst. Sci. 10 (3) (1975) 384–393.

[3] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, R. F. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, in: 8th Heterogeneous Computing Workshop, HCW '99, 1999, pp. 30–.

[4] H. Topcuouglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.

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

[6] S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, J. Wang, Cost-efficient task scheduling for executing large programs in the cloud, Parallel Computing 39 (45) (2013) 177 – 188.

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

[8] A. Wierman, M. Nuyens, Scheduling despite inexact job-size information, in: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2008, pp. 25–36. `doi:10.1145/1375457.1375461`.

[9] M. Dell'Amico, D. Carra, M. Pastorelli, P. Michiardi, Revisiting size-based scheduling with estimated job sizes, in: IEEE 22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2014.

[10] S. Verboven, P. Hellinckx, F. Arickx, 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. `doi:10.1109/APSCC.2008.189`.

[11] C. Yan, H. Luo, Z. Hu, X. Li, Y. Zhang, Deadline guarantee enhanced scheduling of scientific workflow applications in grid, Journal of Computers 8 (4).

[12] J. O. Gutierrez-Garcia, K. M. Sim, A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling, Future Generation Computer Systems 29 (7) (2013) 1682 – 1699. `doi:http://dx.doi.org/10.1016/j.future.2012.01.005`.

[13] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2013)

682 – 692, special Section: Recent Developments in High Performance Computing and Security. doi:10.1016/j.future.2012.08.015.

[14] J.-S. Vöckler, G. Mehta, Y. Zhao, E. Deelman, M. Wilde, Kickstarting remote applications, in: 2nd International Workshop on Grid Computing Environments, 2006.

[15] G. Juve, B. Tovar, R. Ferreira da Silva, C. Robinson, D. Thain, E. Deelman, W. Allcock, M. Livny, Practical resource monitoring for robust high throughput computing, Tech. rep., University of Southern California (2014).

[16] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future Generation Computer Systems 29 (3) (2014) 682–692.

[17] F. Nadeem, M. Yousaf, R. Prodan, T. Fahringer, Soft benchmarks-based application performance prediction using a minimum training set, in: 2nd IEEE International Conference on e-Science and Grid Computing, 2006.

[18] W. Tang, J. Bischof, N. Desai, K. Mahadik, W. Gerlach, T. Harrison, A. Wilke, F. Meyer, Workload characterization for mg-rast metagenomic data analytics service in the cloud, in: IEEE International Conference on Big Data, 2014.

[19] T. Shibata, S. Choi, K. Taura, File-access patterns of data-intensive workflow applications and their implications to distributed filesystems, in: 19th ACM International Symposium on High Performance Distributed Computing (HPDC), 2010.

[20] O. Sonmez, N. Yigitbasi, A. Iosup, D. Epema, Trace-based evaluation of job runtime and queue wait time predictions in grids, in: 18th ACM international symposium on High performance distributed computing, HPDC '09, ACM, 2009, pp. 111–120. doi:10.1145/1551609.1551632.

[21] D. Martínez-Rego, M. Pontil, Multi-task averaging via task clustering, in: E. Hancock, M. Pelillo (Eds.), Similarity-Based Pattern Recognition, Vol. 7953 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 148–159. doi:10.1007/978-3-642-39140-8_10.

[22] J. Kephart, D. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50. doi:10.1109/MC.2003.1160055.

[23] R. Ferreira da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, M. Livny, Toward fine-grained online task characteristics estimation in scientific workflows, in: Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science, WORKS '13, 2013, pp. 58–67. doi:10.1145/2534248.2534254.

[24] 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, D. S. Katz, Pegasus: A framework for mapping complex scientific workflows onto distributed systems, Sci. Program. 13 (3) (2005) 219–237.

[25] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systemsdoi:10.1016/j.future.2014.10.008.

[26] M. Albrecht, P. Donnelly, P. Bui, D. Thain, Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids, in: 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, SWEET '12, ACM, 2012, pp. 1:1–1:13. doi:10.1145/2443416.2443417.

[27] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto, Jr., H.-L. Truong, Askalon: a tool set for cluster and grid computing: Research articles, Concurr. Comput. : Pract. Exper. 17 (2-4) (2005) 143–169.

[28] 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, C. Wroe, Taverna: lessons in creating a workflow environment for the life sciences: Research articles, Concurr. Comput. : Pract. Exper. 18 (10) (2006) 1067–1100.

[29] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, M.-H. Su, Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, SPIE Conference 5493 (2004) 221–232. `doi:10.1117/12.550551`.

[30] USC epigenome center, http://epigenome.usc.edu/.

[31] Periodogram workflow, http://portal.futuregrid.org/projects/77.

[32] Rosetta Commons, http://www.rosettacommons.org.

[33] T. Joshi, B. Valliyodan, S. M. Khan, Y. Liu, J. V. Maldonado dos Santos, et al., Next generation resequencing of soybean germplasm for trait discovery on xsede using pegasus workflows and iplant infrastructure, in: XSEDE 2014, 2014.

[34] T. Joshi, M. R. Fitzpatrick, S. Chen, Y. Liu, H. Zhang, R. Z. Endacott, E. C. Gaudiello, G. Stacey, H. T. Nguyen, D. Xu, Soybean knowledge base (soykb): a web resource for integration of soybean translational genomics and molecular breeding, Nucleic Acids Research 42 (D1) (2014) D1245–D1252. `doi:10.1093/nar/gkt905`.

[35] GATK, https://www.broadinstitute.org/gatk.

[36] H.-P. Kriegel, P. Kröger, J. Sander, A. Zimek, Density-based clustering, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1 (3) (2011) 231–240. `doi:10.1002/widm.30`.

[37] M. Ester, H. P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: 2nd International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[38] I. Chakravarty, J. Roy, R. Laha, Handbook of methods of applied statistics, McGraw-Hill, 1967.

[39] G. Marsaglia, T. A. Bray, A convenient method for generating normal variables, Siam Review 6 (3) (1964) 260–264.

[40] G. Marsaglia, W. W. Tsang, A simple method for generating gamma variables, ACM Trans. Math. Softw. 26 (3) (2000) 363–372. `doi:10.1145/358407.358414`.

[41] R. Ferreira da Silva, T. Glatard, F. Desprez, Self-healing of operational workflow incidents on distributed computing infrastructures, in: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid'12, 2012, pp. 318–325. `doi:10.1109/CCGrid.2012.24`.

[42] R. Ferreira da Silva, T. Glatard, F. Desprez, Self-healing of workflow activity incidents on distributed computing infrastructures, Future Generation Computer Systems 29 (8) (2013) 2284–2294. `doi:10.1016/j.future.2013.06.012`.

[43] Parallel workloads archive.
`www.cs.huji.ac.il/labs/parallel/workload/`

[44] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, D. H. J. Epema, The grid workloads archive, Future Gener. Comput. Syst. 24 (7) (2008) 672–686. `doi:10.1016/j.future.2008.02.003`.

[45] D. Kondo, B. Javadi, A. Iosup, D. Epema, The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems, in: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGrid, IEEE, 2010, pp. 398–407.

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

[47] R. Ferreira da Silva, W. Chen, G. Juve, K. Vahi, E. Deelman, Community resources for

enabling and evaluating research on scientific workflows, in: 10th IEEE International Conference on e-Science, eScience'14, 2014, pp. 177–184. `doi:10.1109/eScience.2014.44`.

[48] R. Ferreira da Silva, 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, Vol. 7640 of Lecture Notes in Computer Science, 2013, pp. 79–88. `doi:10.1007/978-3-642-36949-0_10`.

[49] Workflow archive, http://workflowarchive.org.

[50] L. Ramakrishnan, D. Gannon, A survey of distributed workflow characteristics and resource requirements, Indiana University.

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

[52] S. Ostermann, R. Prodan, T. Fahringer, A. Iosup, D. Epema, A trace-based investigation of the characteristics of grid workflows, in: T. Priol, M. Vanneschi (Eds.), From Grids to Service and Pervasive Computing, Springer US, 2008, pp. 191–203. `doi:10.1007/978-0-387-09455-7_14`.

[53] A. Iosup, D. Epema, Grid computing workloads, Internet Computing, IEEE 15 (2) (2011) 19–26. `doi:10.1109/MIC.2010.130`.

[54] D. L. Hart, Measuring teragrid: workload characterization for a high-performance computing federation, International Journal of High Performance Computing Applications 25 (4) (2011) 451–465.

[55] Z. Ren, X. Xu, J. Wan, W. Shi, 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. `doi:10.1109/IISWC.2012.6402895`.

[56] S. Mahambre, P. Kulkarni, U. Bellur, G. Chafle, D. Deshpande, Workload characterization for capacity planning and performance management in iaas cloud, in: IEEE International Conference on Cloud Computing in Emerging Markets, 2012, pp. 1–7. `doi:10.1109/CCEM.2012.6354624`.

[57] S. Madougou, S. Shahand, M. Santcroos, B. van Schaik, A. Benabdelkader, A. van Kampen, S. Olabarriaga, Characterizing workflow-based activity on a production e-infrastructure using provenance data, Future Generation Computer Systems 29 (8) (2013) 1931 – 1942. `doi:http://dx.doi.org/10.1016/j.future.2013.04.019`.

[58] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, S. Dawson, Markovian workload characterization for qos prediction in the cloud, in: IEEE International Conference on Cloud Computing, 2011, pp. 147–154. `doi:10.1109/CLOUD.2011.100`.

[59] A. Khan, X. Yan, S. Tao, 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. `doi:10.1109/NOMS.2012.6212065`.

[60] R. Ferreira da Silva, M. Rynge, G. Juve, I. Sfiligoi, E. Deelman, J. Letts, F. Würthwein, M. Livny, Characterizing a high throughput computing workload: The compact muon solenoid (CMS) experiment at LHC, in: 2015 International Conference on Computational Science, ICCS 2015, 2015.

[61] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, T. Fahringer, A hybrid intelligent method for performance modeling and prediction of workflow activities in grids, in: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, 2009, pp. 339–347. `doi:10.1109/CCGRID.2009.58`.

[62] E.-K. Byun, Y.-S. Kee, E. Deelman, K. Vahi, G. Mehta, J.-S. Kim, Estimating resource needs for time-constrained workflows, in: eScience, 2008. eScience '08. IEEE

Fourth International Conference on, 2008, pp. 31–38. `doi:10.1109/eScience.2008.18`.

[63] R. Huang, H. Casanova, A. A. Chien, Automatic resource specification generation for resource selection, in: 2007 ACM/IEEE conference on Supercomputing, ACM, 2007, pp. 11:1–11:11. `doi:10.1145/1362622.1362638`.

[64] F. Nadeem, T. Fahringer, Using templates to predict execution time of scientific workflow applications in the grid, in: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, 2009, pp. 316–323. `doi:10.1109/CCGRID.2009.77`.

[65] V. K. Pallipuram, J. DiMarco, M. Taufer, Applying frequency analysis techniques to dag-based workflows to benchmark and predict resource behavior on non-dedicated clusters, in: IEEE International Conference on Cluster Computing, CLUSTER, 2014, pp. 29–37. `doi:10.1109/CLUSTER.2014.6968734`.

[66] A. M. Chirkin, A. S. Z. Belloum, S. V. Kovalchuk, M. X. Makkes, Execution time estimation for workflow scheduling, in: 9th Workshop on Workflows in Support of Large-Scale Science, WORKS'14, 2014, pp. 1–10. `doi:10.1109/WORKS.2014.11`.

[67] I. Pietri, G. Juve, E. Deelman, R. Sakellariou, A performance model to estimate execution time of scientific workflows on the cloud, in: 9th Workshop on Workflows in Support of Large-Scale Science, WORKS'14, 2014, pp. 11–19. `doi:10.1109/WORKS.2014.12`.