A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud

Ilia Pietri*, Gideon Juve†, Ewa Deelman†, Rizos Sakellariou*
*School of Computer Science, University of Manchester, U.K
†Information Sciences Institute, University of Southern California, USA

***Abstract*—Scientific workflows, which capture large computational problems, may be executed on large-scale distributed systems such as Clouds. Determining the amount of resources to be provisioned for the execution of scientific workflows is a key component to achieve cost-efficient resource management and good performance. In this paper, a performance prediction model is presented to estimate execution time of scientific workflows for a different number of resources, taking into account their structure as well as their system-dependent characteristics. In the evaluation, three real-world scientific workflows are used to compare the estimated makespan calculated by the model with the actual makespan achieved on different system configurations of Amazon EC2. The results show that the proposed model can predict execution time with an error of less than 20% for over 96.8% of the experiments.**

I. Introduction

In many scientific domains, a workflow, typically modelled as a Directed Acyclic Graph (DAG), is used to abstract complex computational jobs and describe data dependencies between them [1]. Large-scale distributed systems, such as clusters and grids, have been widely used to execute workflow applications [2], [3]. Recently, the use of cloud computing infrastructures is gaining popularity by offering users several options and benefits compared to traditional high performance environments, especially when it comes to provisioning resources on demand.

A challenge that arises in a cloud computing environment is to determine the number of resources (or slots) to allocate for the cost-efficient execution of scientific workflows. Using a large number of resources may result in small execution time, however, at the expense of a high monetary cost. In some cases, a slightly longer execution time may be tolerated if this comes at a significantly lower monetary cost. Thus, in order to use a cloud infrastructure efficiently, some ability to predict the workflow execution time (or makespan) is needed in order to decide how many resources to provision.

Typically, the makespan is affected by the number of resources used, the structure of the scientific workflow but also task and data communication characteristics. For example, independent tasks may be executed in parallel; then, by allocating a large number of resources a small application makespan can be obtained. In other cases where the tasks can only be executed sequentially, execution time will not be affected if additional resources are added.

SC14, November 16-21, 2014, New Orleans, Louisiana, USA
978-1-4799-5500-8/14/$31.00 ©2014 IEEE

In this paper, a model to estimate the makespan of scientific workflows for a given number of resources is proposed. This performance model takes into account the structure of the scientific workflow and the runtime characteristics of its tasks, using task runtime information from different runs. To capture the structure of the workflow, a *level-based estimation model*, which divides workflow tasks into levels on the basis of data dependencies between them and calculates the characteristics of each level based on task runtimes, is proposed to predict workflow execution time. As opposed to related work [4], [5] the main difference of the performance model proposed in this paper is that it focuses on the structure of the scientific workflow, which enables prediction using minimal information about runtime characteristics of the tasks. This is achieved by assigning (grouping) the tasks of a workflow into levels and estimating the performance at each level. Thus, knowing the runtime characteristics of the tasks when executed on a small number of resources is sufficient to provide good predictions for cases where a different (and perhaps larger) number of resources is available to use. This property (of making use of runtime characteristics for only a small number of resources) may be particularly useful in a cloud environment. The validity of the proposed model is demonstrated using a number of experiments with real-world workflows on Amazon EC2.

The remainder of the paper is structured as follows. Section 2 presents related work. Section 3 describes the problem and environment. Section 4 presents the proposed model. Section 5 evaluates its performance using real experiments on Amazon EC2. Finally, Section 6 concludes the paper.

II. Related Work

Performance modeling and analysis of parallel applications has been a topic with a long history of research [6].

Predicting job runtime for parallel applications has been the focus of many studies, such as [7], [8], [9], [10], [11], [12], [13], [14], [15]. In [7], a prediction method for the runtime of online tasks in high performance computing environments is presented. The model requires historical data from different runs at the same configuration of the computing platform, while the difference from the actual measurements in the clusters is significant. In [8], an algorithm for duration forecasting of workflow activities is proposed and evaluated using both real world examples and simulations. In [9], a method to predict the runtime of jobs in grids, consisting of geographically distributed and/or heterogeneous resources, is developed and evaluated with experiments in real systems. In

[15], a methodology to accurately predict fine-grained task needs for scientific workflows using an online estimation process is presented.

Also, research specifically related to performance modeling of workflow-based parallel applications when running on different infrastructures has been done. In [16], job execution times and data transfer between jobs are considered to model workflow performance using a probabilistic model. Grid overheads are also incorporated in the model using a random variable. A medical image analysis application is used to evaluate the model, providing insight about the impact of grid behavior and variability when executing scientific workflows. In [17], a cost model to be used when scheduling a scientific workflow in clouds is proposed. The aim is to find the best configuration of Virtual Machines (VMs) in terms of both execution time and cost choosing from different choices of instance types and number of VMs. Performance and cost benefit analysis is also considered in [18] focusing on the comparison between commercial and academic clouds.

Finally, prediction of the execution time for parallel workflows is the subject in many studies, such as in [4], [5], [19], [20], [21]. In [4], a machine learning method is proposed that takes into account application input features and historical data. A limitation of this work is that systematic selection of the candidate input features and prediction models is required in order to improve the accuracy of the prediction. Machine learning models are also used in [5] to predict workflow execution time in grid systems taking into account input features and system characteristics. However, the model uses system-performance attributes and its success depends on the availability of historical and monitoring data. In [19], a local learning approach for grid environments is presented weighting application attributes depending on their impact on workflow runtime, while in [20] a method to predict workflow makespan online using similarity templates is proposed. In [21], a framework to predict the execution time of workflow components is described, with the emphasis being on connecting the amount of data consumed and the amount of data produced by components, as opposed to modeling complex workflow structures.

In this paper, the key difference is that runtime estimation is based primarily on workflow characteristics and does not require data from lots of runs on a different number of available resource slots. We demonstrate that we can still get good insight into the number of slots to allocate in order to achieve a desired level of performance when running in cloud environments.

III. Problem Description

In the target cloud environment setting a user submits a workflow for execution under a certain system configuration, where the user specifies the number of resources (or slots – note that the two terms, resources or slots, are assumed to be identical in this paper) to be provisioned. As the execution time and the monetary cost of the workflow depend on the number of resources used, it is crucial for the user to determine how many resources should be provisioned to avoid unnecessary costs and/or wastage of resources. To do so, the execution time of the workflow when using a specific number of resources has to be estimated. The runtime of each task of the application may vary when running on different cloud environments, but it is assumed that task runtime under a specific system configuration can be predicted with some good accuracy. The estimation model can then be used to determine the number of resources required to achieve a certain level of performance and calculate the cost associated with the use of these resources.

Application Model: This paper considers scientific workflows, applications that comprise inter-related tasks with data dependencies between them [1]. Workflows can be represented as DAGs where the nodes represent computational tasks and the edges data dependencies between them. The execution of a task can only start after all data transfer from its predecessors has finished. The time required to execute a task may vary when running on different cloud systems, but it is assumed that information about task runtimes on a specific system is available to the model. The user-defined tasks of the workflow are assigned to jobs to be submitted for execution. In some cases a number of tasks are clustered together into a single job. Task clustering can be done in a number of ways to meet different optimization criteria [22], [23], [24], [25].

Cloud Model: A cloud model similar to Amazon Elastic Compute Cloud (EC2) [26] is assumed in this paper, with VMs of a single instance type being provisioned on demand. Jobs have exclusive access to VMs consuming all of their capacity. The number of available slots is equal to the total number of CPUs of the provisioned VMs, assuming that VMs are exclusively used for the execution of the jobs in the workflow. Unused slots remain idle, while a job can be assigned to a free slot when data dependency constraints are met, that is, data transfer from its predecessors has finished. The provider can charge for the use of resources in terms of the number of slots and the time they were used. However, the specifics of the pricing model employed are orthogonal to the proposed performance model in this paper, as the performance model provides estimates of the workflow execution time without requiring information related to the pricing model. The pricing model can be used in addition only if it is necessary to estimate the monetary cost building on the performance model.

IV. Makespan Estimation

The key idea of the model proposed for estimating the makespan of a scientific workflow is to take into account the workflow structure and to divide tasks into levels based on the data dependencies between them so that tasks assigned to the same level are independent to each other. Then, for each level, its execution time (which is equal to the time required for the execution of the tasks in the level) can be calculated considering the overall runtime of the tasks of the level (that is, the sum of the individual task runtimes). The assignment in levels can be done using either a top-down or a bottom-up approach that assigns a level to each task by taking into account the level of its predecessors or successors, respectively. Once this assignment has taken place, a level-based estimation model takes into account level characteristics to provide an overall performance estimate for the workflow.

A. Level Assignment Approaches

The two approaches used to assign levels to the tasks of the workflow are a top-down and a bottom-up approach, explained

Algorithm 1 Level-based Estimation Model

```
Require: the workflow with task runtimes and number of available slots
 1: procedure RuntimeEstimation($runtime_t$, $slots$)
 2:     Top-Down or Bottom-Up approach, resp. Eq. 1 and 2      ▷ Assign levels to tasks
 3:     for each level $l$ do
 4:         $tasksByLevel_l$                                     ▷ the number of tasks in the level
 5:         $runtime_l = \sum_{t \in l} runtime_t$               ▷ the maximum runtime of the level
 6:         $minRuntime_l = max_{t \in l}\{runtime_t\}$          ▷ the minimum runtime of each level
 7:         $maxSlots_l = tasksByLevel_l$                        ▷ the maximum number of slots that can be used in the level
 8:     end for
 9:     for each level $l$ do
10:         Compute $makespan_l$ using Equation 3                ▷ level makespan
11:     end for
12:     $makespan_{slots} = \sum_l makespan_l + Delay_l$         ▷ compute estimated makespan by adding all level makespans
13:     return $makespan_{slots}$                                ▷ return the estimated makespan for the given number of slots
14: end procedure
```

below.

Top-Down Approach (TDA): In a top-down approach the level of a task is given by the longest path from an entry node of the workflow. To do so, the tasks of the workflow are ordered in topological order and the level of each task is given by Equation 1, where the level of task t, $Level_t$, is the maximum level of its predecessors, $pred_t$, increased by 1 with the level of all entry nodes being 0.

$$Level_t = max_{p \in pred_t}\{Level_p\} + 1 \tag{1}$$

Bottom-Up Approach (BUA): In a bottom-up approach the level of each task is given by the longest path from an exit node of the workflow. More specifically, the level of a task t is calculated in reverse order as the maximum level of its successors, $succ_t$, increased by 1 with the level of all exit nodes being 0.

$$Level_t = max_{s \in succ_t}\{Level_s\} + 1 \tag{2}$$

Note that other approaches for assigning tasks into levels may be used. For example, one could use as many levels as tasks in the longest path from an entry node to an exit node and then try to make assignments for the remaining tasks in ways that respect dependencies.

B. Level-based Estimation Model

The model to predict execution time of a workflow when allocating a certain number of slots is described in Algorithm 1. Firstly, it calls the level assignment algorithm (top-down or bottom-up approach) to assign levels to the tasks of the workflow (step 2). Then, the characteristics of each level are calculated, including the number of tasks assigned to the level, the maximum runtime of all the tasks of the level ($runtime_l$), being the total time required to execute the tasks assigned in the level sequentially, and its minimum runtime ($minRuntime_l$), which is the longest time a task assigned to this level requires to run (steps 4-6). Also, the maximum number of slots that can be used in a level ($maxSlots_l$) is equal to the number of tasks assigned to this level (step 7). As every task of a level can use a maximum of only one slot, there is no benefit in allocating more slots than the number of tasks at a level. Step 10 estimates the execution time of each level given the available slots using the following equation:

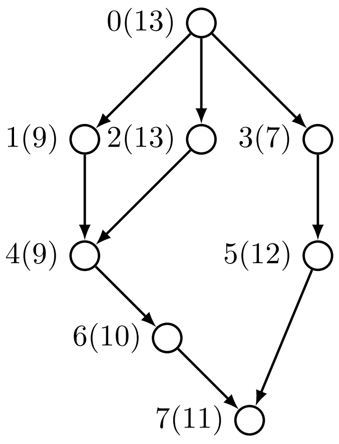
$$makespan_l = max\left(\frac{runtime_l}{min(slots, maxSlots_l)}, minRuntime_l\right). \tag{3}$$

The rationale of this equation is that the execution time for a single level ($makespan_l$) cannot be less than $minRuntime_l$, that is the time required to run the longest task of the level. Similarly, it is taken into account that more slots than the number of tasks at the level will not result in a performance improvement. Finally, the overall workflow makespan ($makespan_{slots}$) is calculated in step 12 by adding the makespan of each assigned level. To this value a fixed delay for the workflow may be incorporated into the model to represent job submission delays that often happen in real environments.

In the case of different configurations, for example when resources with a different number of cores per machine are used, the execution time of each job may vary. Also, the use of shared resources, such as the memory and network, may result in an increase in the execution time of the tasks. To deal with the variation in the execution time of the tasks due to system overheads and make the model more accurate for different configuration scenarios, a scaling factor can be introduced in the calculation of the estimated makespan in Equation 3. For example, a scaling factor based on the average of the runtime variation of each task can be used to scale the execution time of each level. Understanding how to scale execution time is the subject of future work.

C. Job Submission Delays

In many real-world environments the jobs of a workflow are submitted through a queue-based system [2]. This may introduce an additional delay to the execution of the workflow as jobs are submitted to the queue in a way that preserves dependencies. For example, to validate the model proposed in this paper, the Pegasus Workflow Management system [27] is used to plan and execute the workflows. Pegasus makes use of DAGMan [28] to manage data dependencies



(a) DAG

$Level$	min $Runtime_l$	max $Runtime_l$	max $Slots_l$	$Tasks_l$
0	13	13	1	0
1	13	29	3	1, 2, 3
2	12	21	2	4, 5
3	10	10	1	6
4	11	11	1	7

(b) TDA

$Level$	min $Runtime_l$	max $Runtime_l$	max $Slots_l$	$Tasks_l$
4	13	13	1	0
3	13	22	2	1, 2
2	9	16	2	3, 4
1	12	22	2	5, 6
0	11	11	1	7

(c) BUA

Fig. 1: An example using the two makespan prediction approaches.

between executable jobs, while Condor manages the individual execution of each job. Job runtimes used as input for the estimation model are generated using the logs in DAGMan [28]. However, as the results of the model are compared to overall workflow runtimes managed by Condor, system delays, such as Condor/DAGMan delays or queueing time, need to be accounted for [2]. In the simplest form, these delays can be approximated through a system-dependent delay added per level, which will have to be estimated separately for different environments and platforms. In the evaluation of the model in this paper, a coarse-grain assumption for a constant delay of 25 seconds per level of the workflow is made.

D. Example

Figure 1a shows an example of how the execution time prediction is performed for a simple DAG. Every node of the DAG is annotated with two numbers. The first number is the task id, the number inside brackets is the estimated task runtime.

Using TDA and two slots the predicted makespan is:
$TDA_{2slots} = 13 + 14.5 + 12 + 10 + 11 = 60.5$.
The makespan is reduced for 4 slots to:
$TDA_{4slots} = 13 + 13 + 12 + 10 + 11 = 59$.
It can be seen that only one slot can be used in level 0 that contains one task (task 0), while the estimated makespan of level 1 is reduced to the minimum level makespan, the runtime of longest task of the level (task 2), in the case of 4 available slots. In the case of BUA, the predicted makespan for 2 slots is:
$BUA_{2slots} = 13 + 13 + 9 + 12 + 11 = 58$.
The same makespan, 58, is achieved when using BUA with 4 slots. The predicted makespan differs from the estimation of TDA, as tasks 3 and 5 are assigned to different levels, starting from the exit node. The system-dependent delays, in this example, are considered to be 0. Tables 1b and 1c show the detailed values used in the calculation of the estimated makespan according to Equation 3.

It is noted that the two different approaches for assigning tasks to levels, TDA and BUA, result in a different makespan. This is because different tasks are assigned to different levels. Thus, in both BUA and TDA, the top level consists of task 0. Then, TDA assigns tasks into the next levels as follows: $\{1, 2, 3\}$, $\{4, 5\}$, $\{6\}$, $\{7\}$. Conversely, BUA assigns tasks into levels as follows: $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$, $\{7\}$. This is also shown in the last column of Tables 1b and 1c.

In general, the level assignment approach may be chosen based on the scheduling scenario, taking into account the workflow structure and the scheduling algorithm to be used to allocate the tasks on the available resources. For example, BUA assigns tasks to the levels according to the successors of the tasks starting from the exit node, while TDA assigns the levels based on the level assignment of the predecessors starting from the root. As a result, BUA may be more suitable when the scheduling aims at exploiting the latest finish time of the tasks, while TDA may be chosen when the scheduling aims at executing the tasks as early as possible.

As already mentioned, a good reason for having a workflow execution estimation model for a cloud platform is to assess execution time against the monetary cost for using the resources. To illustrate this, we apply a simple pricing model to the example DAG and the estimation results above. In this pricing model, the monetary cost is proportional to the overall time that slots were used, which corresponds to the overall value of the makespan. More specifically, an upper bound on the monetary cost to execute the workflow on a number of slots, $slots$, can be computed based on the estimated makespan. The cost of provisioning a slot, r, for a period equal to the scheduling makespan, $makespan_{slots}$, is computed as

$$Cost_r = p * makespan_{slots}, \tag{4}$$

with the slot being charged at a price p per time unit. The overall cost required for the execution of the workflow is the sum of the provisioning cost of each slot used, with an upper bound of

$$CostBound_{slots} = Cost_r * slots, \tag{5}$$

when using the available number of $slots$.

In the case of the example, assume that each resource used is charged at the price of $1 for each time unit. Then, the cost incurred by the user is expected to be $58*2=$116 for BUA and $60.5*2=$121 for TDA, when two resources are provisioned. The cost is expected to increase when provisioning a larger number of resources. Indeed, when using four resources, the monetary cost is $58*4=$232 for BUA and $59*4=$236 for TDA. This assessment suggests that a slight delay in workflow execution may be tolerated to avoid a significant increase in the monetary cost and strike a good balance between the execution time and application cost for using the cloud resources.

V. EVALUATION

A. Experimental Setup

To validate the model, experiments on Amazon EC2 using three real-world workflows were carried out. The configuration used to execute the workflows on EC2 is described next.

Firstly, a submit host runs outside the cloud to manage the workflows and set up the execution environment using the Pegasus Workflow Management system [27]. The Nimbus Context Broker [29] is also installed to provision and configure the virtual cluster, consisting of the configured VMs. To do so, worker nodes to execute the workflow jobs inside the cloud are deployed on Amazon EC2 using the c1.xlarge and m1.xlarge instance types in the experiments. The c1.xlarge instance type is configured with an eight-core 2.33-2.66 GHz Xeon processor, 7.5 GB RAM and 1680 GB local disk storage and the m1.xlarge instance type is configured with two dual-core 2.0-2.6 GHz Opteron processors, 15 GB RAM and 1680 GB local disk storage. For the execution of the workflows different storage systems running inside the cloud are used in the experiments to store the input and output data of the workflow jobs, including Amazon S3 [30], NFS [31], GlusterFS [32], PVFS [33], P2P file sharing and local disk usage.

Amazon S3 [30] is a distributed system that provides storage for objects, such as files, through web service interfaces. The network file system NFS [31] uses a centralized node, a file server, connected to a group of machines in order to provide access to files over the network. GlusterFS [32] is a distributed storage system with client and server components in each node, supporting various configurations (such as non-uniform file access and distributed configurations). Each remote server exports a local volume and merges it with the volumes of the other machines in order to compose the final volume, while a hashing algorithm can be used in the configuration to distribute the files to the nodes more uniformly. In the case of the parallel file system PVFS [33], data striping is used to distribute file data across multiple nodes and provide parallel access by the tasks. Software RAID (RAID 0) is also used to improve I/O performance by striping data across the available ephemeral storage devices [3].

B. Workflows

Three real scientific workflows were used in the experiments, namely Montage [34], Epigenome [35] and Broadband [36]. The basic structure of the three workflows is shown in Figure 2.

Montage is a scientific workflow that generates image mosaics of the sky and can be characterized as I/O intensive [3], [37]. Most of the jobs in Montage have low CPU utilization and short runtime, spending their execution time mainly on I/O operations to read and write files. A simple example of the structure of Montage is shown in Figure 2a. The number of the jobs in each level may vary depending on the size of the generated workflow and the number of clusters used. A

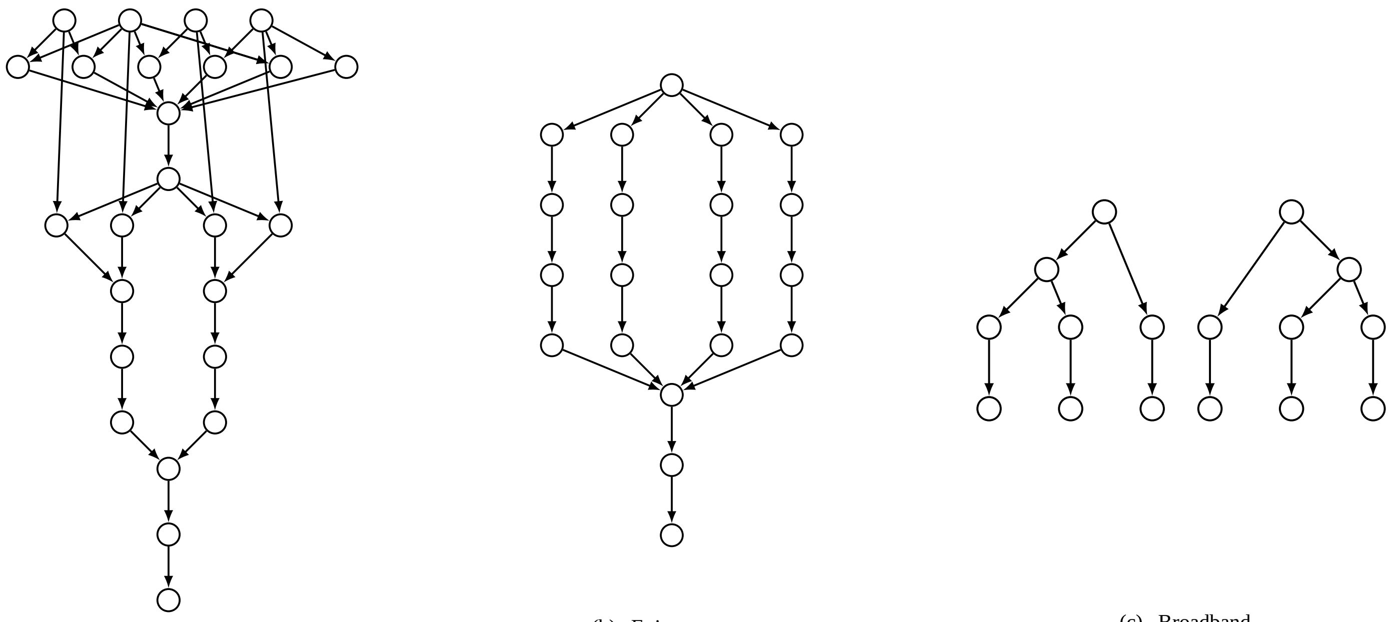

(a) Montage (b) Epigenome (c) Broadband

Fig. 2: The structures of the workflows used.

Montage workflow with 10429 tasks was generated and used in the experiments. The tasks were clustered to create a total of 31, 55, 103-104, 152, 248, 440 and 823 jobs when using 4, 8, 16, 32, 64, 128 and 256 clusters per level, respectively. The jobs can be divided into 13 levels taking into account data dependencies between them.

The Epigenome workflow maps the epigenetic state of human cells and can be classified as a CPU-bound application [3], [37] with several parallel jobs operating in independent data files, as shown in Figure 2b. Most of the jobs are computationally intensive, while only a few jobs, that split/convert the input files to multiple/formatted output files, have low CPU utilization. As a result, most of the runtime is spent in the CPU and only a small amount of time is spent on other operations. An Epigenome workflow of 529 tasks was used in the experiments, with the tasks clustered to create a total of 50-51, 83, 147, 275 and 529-531 jobs using 8, 16, 32, 64 and 128 clusters per level, respectively. The Epigenome workflow consists of 11 job levels.

Broadband is a data-intensive workflow application with high memory utilization that integrates earthquake motion simulation codes [3], [37]. Jobs with high memory requirements (more than 1 GB of memory) have the longest runtime consuming more than 75% of the total execution time of the workflow, while some data are being accessed several times. In the experiments, a workflow of 768 tasks was generated which was executed using 770 jobs (one task per job including two extra jobs to create the working directory and copy the files to it) in 6 levels.

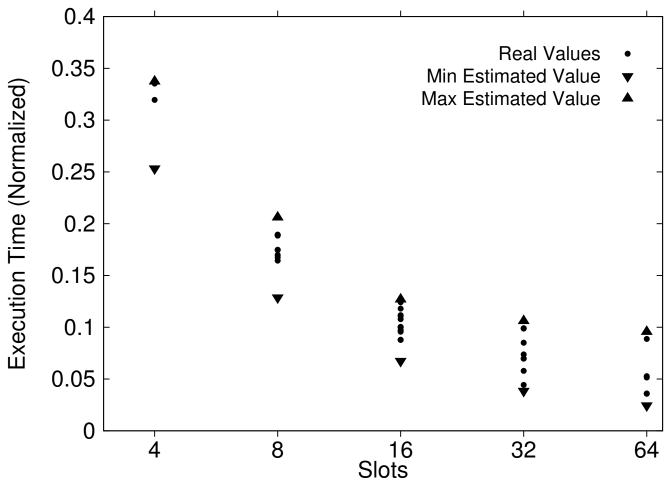
C. *Results*

To evaluate the model, experiments on Amazon EC2 using the three workflows (and task clustering as mentioned), different storage configurations and running on 4, 8, 16, 32, or 64 slots were carried out. In total, there were 35 experiments for Montage, 25 experiments for Epigenome and 35 experiments for Broadband.

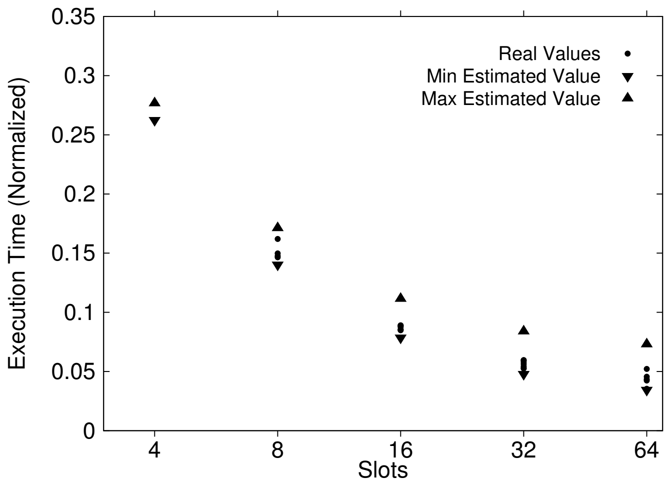
a) Comparison of estimated makespan with actual measurements: To validate the accuracy of the model, individual job runtimes obtained from actual workflow runs for a given number of slots were used as input to the model to come up with estimations of the workflow execution time (makespan) according to Algorithm 1. The error between the estimated and the real (actual) makespan was calculated using the equation:

$$\epsilon = \left| \frac{makespan_{real} - makespan_{pred}}{makespan_{real}} \right|. \qquad (6)$$

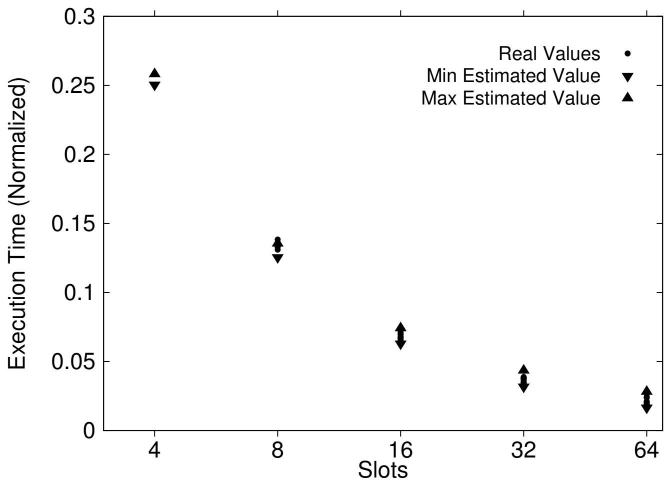
In the case of Montage 33 out of 35 experiments gave an error, ϵ, less than 10%. The prediction accuracy is lower for Epigenome with an error less than 10% in 17 out of the 25 experiments. The results are identical regardless of whether TDA or BUA is used to assign levels. In the case of Broadband, the accuracy of the model is higher using TDA with 27 out of 35 experiments resulting in an error of less than 10%. Using BUA 25 out of 35 experiments had an error of less than 10%. This is due to Broadband's structure, which results in a significantly different level assignment depending on whether TDA or BUA is used. For all three workflows ($35+25+35 = 95$ experiments) there were only 2 experiments



(a) Montage



(b) Epigenome



(c) Broadband

Fig. 3: Execution time prediction compared with real measurements.

in Epigenome and 1 experiment in Broadband where the error was higher than 20%. In all three cases, it appears that the model overestimates, which suggests that there was minimal job submission delays in these three cases. Overall, in 92 out of 95 (or 96.8%) experiments, the model gives an error of less than 20%, while in 77 out of 95 (or 81%) experiments there was an error of less than 10%. In this exercise, the accuracy of the model was evaluated for the case of perfectly accurate task runtime estimates. To do this, we used the actual task runtimes of a given run to predict the workflow makespan of that run and compared the estimated makespan with the actual makespan of the run. Later (in paragraph c) we deal with inaccurate predictions, as estimated and actual task runtimes may vary in practice.

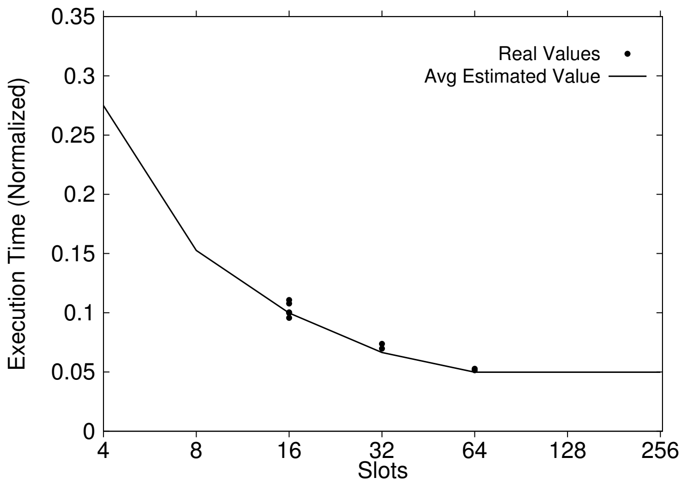
b) Using the model to make predictions for different number of slots: In this exercise, we used individual job runtimes from each experiment (on a given number of slots) to predict the workflow makespan for 4, 8, 16, 32 and 64 slots. This gives a range of predictions (one prediction for every experiment used) for each number of available slots, out of which we considered the minimum and the maximum values. These values, along with the real measurements for every workflow and every different number of slots are shown in Figure 3. The triangular symbols (normal and upside down) indicate the maximum and minimum predicted values, respectively, while the dots correspond to real measurements on EC2. Execution time is normalized using the sum of the runtimes of the jobs (equivalent to the time needed to run all jobs on one slot). This is because, even with the same number of slots, task runtime performance may vary when using different storage systems [3]. It can be seen that the real measurements are within the prediction range. The variation between the minimum and maximum prediction is different for each workflow and it is more profound in the case of Montage. This is expected and it is due to the impact of storage systems on workflow I/O activity. Montage processes many files and a large amount of data is processed and generated resulting in high I/O utilization, which means that the storage system performance has a big impact on workflow performance. In contrast, storage systems affect the performance of Epigenome less, as this is a CPU-intensive workflow where most of the time is spent operating data in memory, not on I/O. Finally, in Broadband the variation is small as the workflow involves a lot of input data, but relatively little output data; much of the input data is common, so the workflow makes more effective use of the file system cache than Montage. Clearly, as data storage systems influence execution time, predictions can be improved if they are estimated for every storage system separately (this is demonstrated later in this section).

c) Model performance evaluation with inaccurate job runtimes: So far, we used as an input of the model actual (measured) job runtimes. In reality, the individual job runtimes used as an input to the model may deviate from actual runtimes. Although the assumption that the runtimes of the jobs can be adequately predicted may be reasonable (see for example the large body of work focusing on job runtime prediction and/or workflow characterization [8], [9], [10], [11], [12], [37]), in reality these models may overestimate or underestimate the execution time of the jobs, especially as there have been cases where job performance appear to vary even for cloud instances of the same type from the same cloud provider [38]. In order to investigate the impact of inaccurate individual job runtimes on the overall workflow makespan predicted by the model, job runtimes (rt for short) were varied using a random error in the range of $\pm 5\%$, $\pm 10\%$ and $\pm 15\%$. We generated 100 random values for each of the three ranges and for each of the $(35 + 25 + 35 = 95)$ experiments for the three workflows. The error, ϵ, between estimated makespan by the model and actual values on EC2 was computed in each case. Then, the percentage of the experiments with ϵ being less than 10%, 15% and 20% using TDA or BUA is presented in Table I. These percentages relate to 3500 error values for each case in Montage (35 experiments $\times$ 100 random values), 2500 error values for each case in Epigenome and 3500 error values for Broadband. It can be seen that even when there is inaccuracy in individual job runtime estimates used by the model, the performance of the model is not different to the results obtained with accurate job runtime estimates. As noted in the first paragraph of this section (comparison of estimated makespan with actual measurements), in 33 out of 35 experiments (or 94.28%) for Montage, in 17 out of 25 experiments (or 68%) for Epigenome and in 27 out of 35 experiments (or 77.14%) for Broadband (using TDA) the model gave a prediction error of less than 10%. The first two rows of the table (corresponding to $\epsilon < 10\%$) suggest that even a small inaccuracy in job runtimes does not affect the performance of the model significantly.

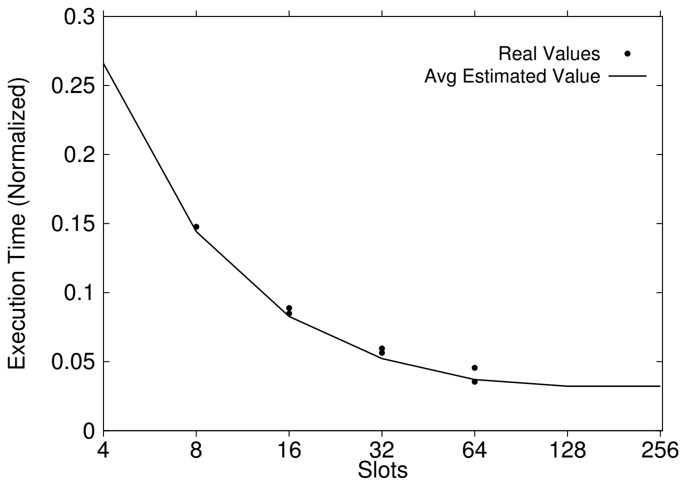
d) Estimating workflow execution time in practice: In the previous paragraphs, we evaluated different aspects of the model and compared actual measurements with model predictions. In reality, the model may be used with some individual job estimates to predict the workflow execution time for a number of slots for which there are no actual workflow runs. Such a prediction may be important in order

		Montage			Epigenome			Broadband		
ϵ	Model	rt± 5%	rt± 10%	rt± 15%	rt± 5%	rt± 10%	rt± 15%	rt± 5%	rt± 10%	rt± 15%
$< 10\%$	TDA	94.49	89.94	79.94	68.28	68.08	65.40	77.40	80.31	80.51
	BUA	94.37	89.89	79.29	68.24	68.08	65.80	71.98	74.26	77.26
$< 15\%$	TDA	98.34	98.23	97.03	79.76	79.96	79.64	88.77	88.89	88.57
	BUA	98.49	98.51	97.20	80.20	80.20	79.92	85.54	86.51	86.66
$< 20\%$	TDA	100.00	100.00	99.86	90.88	89.64	89.04	97.14	97.03	96.91
	BUA	100.00	100.00	99.91	90.44	90.20	89.52	97.14	96.83	96.46

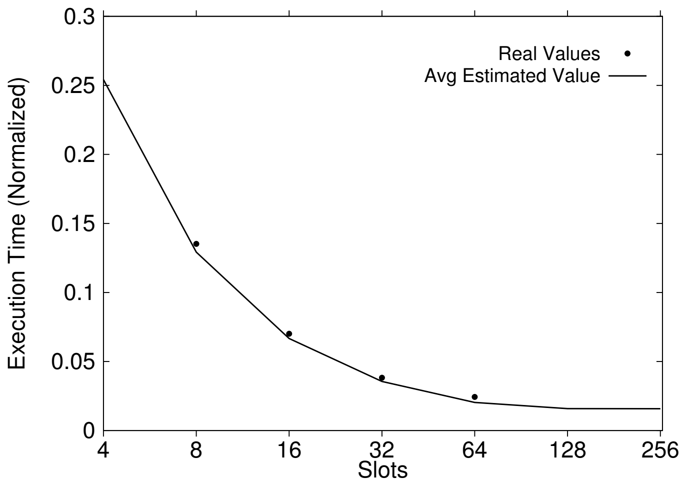
TABLE I: Percentage of experiments within a certain prediction accuracy.



(a) Montage - GFS



(b) Epigenome - NFS



(c) Broadband - Amazon S3

Fig. 4: Execution time prediction for different slots and a single storage system.

to determine in a cost-efficient manner the number of slots to provision for every different workflow. In this exercise, we considered for each workflow the measurements on a specific storage system and we tried to estimate the performance for any different number of slots. To do this, we run the model multiple times using as an input individual job runtimes with a random variation of $\pm 10\%$. More specifically, we used as a starting point individual job runtimes from: 9 different measurements for Montage using GFS on 16, 32 and 64 slots; 7 different measurements for Epigenome using NFS on 8, 16, 32, and 64 slots; and 4 different measurements for Broadband using Amazon S3 on 8, 16, 32, and 64 slots. For each of these measurements 100 values with a variation of $\pm 10\%$ were generated for individual job runtimes and used as input to the model. The predicted workflow execution times (900 predictions for Montage, 700 predictions for Epigenome and 400 predictions for Broadband) were averaged for each workflow and number of slots and the results are shown in Figure 4. It can be seen that the (average) estimated values fit well with the measurements we had, also giving a prediction for a number of slots from 4 to 256. These predictions also indicate that there is no performance improvement for Montage if more than 64 slots are used or for Epigenome and Broadband if more than 128 slots are used. Having this information in practice is important for the user to decide what number of slots to provision.

VI. Conclusion

In this paper we considered the problem of execution time prediction for scientific workflows. A model to estimate workflow makespan based on information about the workflow structure and individual job characteristics is described using two different approaches to assign levels to the jobs. The performance of the model was evaluated and compared with real experiments on Amazon EC2 using three scientific workflows. The results show that the model has good prediction accuracy.

Future work could improve the accuracy of the model using more elaborate estimates for system overheads or job runtimes, for instance estimating runtimes in relation to the number of slots used, introducing a scaling factor for the variation of the execution time of the tasks. Another interesting observation is that other level assignment approaches can be developed for workflow structures where several tasks can be distributed in more than one levels sometimes without affecting the level assignment of their predecessors and successors. For example, in such cases tasks with long execution time may be distributed in more than one level or tasks with short runtimes can be assigned to different levels depending on the structure of the workflow. Level assignment approaches to deal with these cases is another future direction to be followed. Finally, the proposed model can be used to estimate operational costs and find cost-efficient configurations.

Acknowledgment

This work was supported in part by the U.S. Department of Energy Office of Science under Award Number ER26110, “dV/dt - Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science”.

References

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

[2] S. Bharathi and A. Chervenak, "Data staging strategies and their impact on the execution of scientific workflows," in *Proceedings of the 2nd International Workshop on Data-aware Distributed Computing (DADC)*. ACM, 2009.

[3] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Data sharing options for scientific workflows on Amazon EC2," in *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC10)*. IEEE Computer Society, 2010, pp. 1–9.

[4] T. Miu and P. Missier, "Predicting the execution time of workflow activities based on their input features," in *High Performance Computing, Networking, Storage and Analysis (SCC), SC Companion*. IEEE Computer Society, 2012, pp. 64–72.

[5] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE Computer Society, 2010, pp. 495–504.

[6] V. S. Adve, R. Bagrodia, J. C. Browne, E. Deelman, A. Dube, E. N. Houstis, J. R. Rice, R. Sakellariou, D. J. Sundaram-Stukel, and P. J. Teller, "Poems: End-to-end performance design of large parallel adaptive computational systems," *IEEE Transactions on Software Engineering*, vol. 26, no. 11, pp. 1027–1048, 2000.

[7] M. Tao, S. Dong, and L. Zhang, "A multi-strategy collaborative prediction model for the runtime of online tasks in computing cluster/grid," *Cluster Computing*, vol. 14, no. 2, pp. 199–210, 2011.

[8] X. Liu, Z. Ni, D. Yuan, Y. Jiang, Z. Wu, J. Chen, and Y. Yang, "A novel statistical time-series pattern based interval forecasting strategy for activity durations in workflow systems," *Journal of Systems and Software*, vol. 84, no. 3, pp. 354–376, 2011.

[9] M. Dobber, R. van der Mei, and G. Koole, "Effective prediction of job processing times in a large-scale grid environment," in *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC)*. IEEE Computer Society, 2006, pp. 359–360.

[10] W. Smith, I. Foster, and V. Taylor, "Predicting application run times using historical information," in *Job Scheduling Strategies for Parallel Processing (JSSPP)*. Springer, 1998, pp. 122–142.

[11] V. Taylor, X. Wu, J. Geisler, and R. Stevens, "Using kernel couplings to predict parallel application performance," in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC)*. IEEE Computer Society, 2002, pp. 125–134.

[12] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Modeling user runtime estimates," in *Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*. Springer, 2005, pp. 1–35.

[13] ——, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 18, no. 6, pp. 789–803, 2007.

[14] E. J. H. Yero and M. A. A. Henriques, "Contention-sensitive static performance prediction for parallel distributed applications," *Performance Evaluation*, vol. 63, no. 4, pp. 265–277, 2006.

[15] R. F. da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and 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)*. ACM, 2013, pp. 58–67.

[16] T. Glatard, J. Montagnat, and X. Pennec, "A probabilistic model to analyse workflow performance on production grids," in *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*. IEEE Computer Society, 2008, pp. 510–517.

[17] V. Viana, D. de Oliveira, and M. Mattoso, "Towards a cost model for scheduling scientific workflows activities in cloud environments," in *Proceedings of IEEE Services*. IEEE Computer Society, 2011, pp. 216–219.

[18] G. B. Berriman, E. Deelman, G. Juve, M. Rynge, and J.-S. Vöckler, "The application of cloud computing to scientific workflows: a study of cost and performance," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, 2013.

[19] F. Nadeem and T. Fahringer, "Predicting the execution time of grid workflow applications through local learning," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, Proceedings (SC09)*. IEEE Computer Society, 2009, pp. 1–12.

[20] ——, "Using templates to predict execution time of scientific workflow applications in the grid," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*. IEEE Computer Society, 2009, pp. 316–323.

[21] H. Hiden, S. Woodman, and P. Watson, "A framework for dynamically generating predictive models of workflow execution," in *Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science (WORKS'13)*. ACM, 2013, pp. 77–87.

[22] G. Singh, M.-H. Su, K. Vahi, E. Deelman, B. Berriman, J. Good, D. S. Katz, and G. Mehta, "Workflow task clustering for best effort systems with Pegasus," in *Proceedings of the 15th ACM Mardi Gras Conference*. ACM, 2008.

[23] W. Chen, E. Deelman, and R. Sakellariou, "Imbalance optimization in scientific workflows," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing (ICS)*. ACM, 2013, pp. 461–462.

[24] W. Chen and E. Deelman, "Fault tolerant clustering in scientific workflows," in *IEEE 8th World Congress on Services*. IEEE, 2012, pp. 9–16.

[25] W. Chen, R. F. D. Silva, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," in *Proceedings of the 9th IEEE International Conference on e-Science*. IEEE Computer Society, 2013, pp. 188–195.

[26] Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2.

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

[28] Team Condor, DAGMan: A Directed Acyclic Graph Manager, http://www.cs.wisc.edu/condor/dagman.

[29] K. Keahey and T. Freeman, "Contextualization: Providing one-click virtual clusters," in *Proceedings of the 4th International Conference on eScience*. IEEE Computer Society, 2008, pp. 301–308.

[30] Simple Storage Service (S3), http://aws.amazon.com/s3.

[31] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the sun network filesystem," in *Proceedings of the Summer USENIX Conference*, 1985, pp. 119–130.

[32] Gluster, Inc., http://www.gluster.org.

[33] R. B. Ross and R. Thakur, "PVFS: A parallel file system for linux clusters," in *Proceedings of the 4th Annual Linux Showcase and Conference*, 2000, pp. 391–430.

[34] D. S. Katz, J. C. Jacob, E. Deelman, C. Kesselman, G. Singh, M.-h. Su, G. Berriman, J. Good, A. Laity, and T. A. Prince, "A comparison of two methods for building astronomical image mosaics on a grid," in *Proceedings of the IEEE International Conference on Parallel Processing Workshops (ICPPW)*. IEEE Computer Society, 2005, pp. 85–94.

[35] USC Epigenome Center, http://epigenome.usc.edu.

[36] Southern California Earthquake Center, http://www.scec.org/cme.

[37] 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, 2012.

[38] L. Gillam, B. Li, J. O'Loughlin, and A. P. S. Tomar, "Fair benchmarking for cloud computing systems," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, 2013.