Llwaah F, Cala J, Thomas N.

Simulation of Runtime Performance of Big Data Workflows on the Cloud.

***In: 13th European Performance Engineering Workshop (EPEW)*. 2016, Chios, Greece: Springer.**

Simulation of Runtime Performance of Big Data Workflows on the Cloud

Faris Llwaah, Jacek Cała, Nigel Thomas

Newcastle University
Newcastle upon Tyne, UK
{f.llwaah,jacek.cala,nigel.thomas}@ncl.ac.uk

Abstract. Big data analysis has become a vital tool in many disciplines. Due to its intensive nature, big data analysis is often performed in cloud computing environments. Cloud computing offers the potential for large scale parallelism and scalable provision. However, determining an optimal deployment can be an expensive operation and therefore some form of prediction of performance prior to deployment would be extremely useful. In this paper we explore the deployment of one complex such problem, the NGS pipeline. We use provenance execution data to populate models simulated in WorkflowSim and CloudSim. This allows us to explore different scenarios for runtime properties.

Keywords: Big-Data, scalability, NGS pipeline, WorkflowSim, CloudSim

1 Introduction

A big data workflow is composed of many applications that may involve large input data sets and produce large amounts of data as an output [5]. The scale and demand of these applications is such that they might rapidly overwhelm stand alone computing systems. One solution to this problem is to deploy the workflow into a commercial cloud environment, which provides ample resources and elastic provision. However, hiring resources clearly costs money and the process of tuning the deployment to ensure sufficient and efficient use of resources can be a costly exercise in itself. Therefore some means of predicting the performance of deployed workflows would be extremely useful and could save money.

In this paper, we explore this problem by considering a complex genomics data processing Next Generation Sequencing (NGS) workflow-based pipeline deployed on the Microsoft Azure public cloud [1]. The NGS pipeline is used to discover variants in patients exome. The local deployment of this pipeline, processing a cohort of 24 patient samples, typically takes several days to execute. The Azure deployment can potentially run much faster, but given limited funds it is necessary to find an optimal or near-optimal deployment which minimises both execution time and cost.

Fortunately a number of simulation tools have become available in recent years which enable a workflow to be simulated in repeatable and reproducible

experiments, with no charge for testing environment [9]. These simulators have been a significant tool for the evaluation and improvement a single workflow[3], although there is a lack of support for simulating a pipeline (a set of workflows).

In this paper, we have modified a simulation platform to simulate the execution behavior of the NGS pipeline as implemented in e-Science central (e-SC). The main contributions of this paper is to propose a methodology for predicting the runtime and output data size using WorkflowSim/CloudSim, parameterised with realistic data from archived provenance file of e-Science central workflows. In order to achieve this we have translated the e-SC workflow enactment model into a Pegasus workflow suitable for input into WorkflowSim and used WorkflowSim and CloudSim to predict runtime and the output data size. To the best of our knowledge this kind of prediction is novel.

The remainder of this paper is structured as follows: the next section covers some background and related work. Section 3 presents the simulation model. Section 4 discusses the proposed prediction methodology which is followed by Section 5 covering the evaluation. Finally, conclusions and future work are presented in Section 6.

2 Background and Related work

Although providing cloud runtime estimation for big data workflows is a problem of significant interest, very few studies are currently available in the literature. This is partially, due to the complexity of the problem in terms of workflow performance behavior and due to the modernity of cloud simulation. Some notable contributions in this area include Rak *et al* [11], who presents a technique to evaluate the trade-off between costs and performance of the cloud application through benchmarks and simulation based on the mOSAIC framework. In [13] the authors extend this approach to consider *bag-of-tasks* scientific applications. The integrated framework with the cloud simulation environment is able to predict the behavior of development stage performance and cloud resource usage. Rozinat *et al* [12], describe a simulation system for operational decisions to support the context of workflow management. The proposed approach combines and extends the workflow management system YAWL and the process mining framework ProM. CloudProphet [8] aims to predict the performance and costs of legacy applications when executed on cloud infrastructures. The advantage of this approach is focused on applications cloud-aware by design, which means it takes into account the elasticity of elasticity rather than using a framework to predict the performance. The framework presented in [14] for performance prediction of parallel programs on hierarchical clusters is based on two principle steps:- one at the installation time of the parallel application and the other at the runtime. In order to model accurately the components, they are sketched those components. In the second step in this approach the generated model was used to the completion time estimation via the fast simulator MPI-PERF-SIM.

Our approach is built on to of WorkflowSim [4], which is an extension of the CloudSim simulator. This is done by providing multiple layers on top of the

existing task scheduling layer of CloudSim, such as workflow mapper, workflow engine, clustering engine, and workflow scheduler. Cloudsim [3] provides the realistic components such as data centre, host, policies and workloads.

3 NGS pipeline simulation

To explore the problem of simulating workflow deployment in the cloud, we have chosen a case study using the NGS pipeline [1]. These workflows are used to implementing WES data processing pipelines at the Institute for Genetic Medicine. In general, a pipeline consists of a composition of workflows that include typical NGS processing steps [10], which are alignment (BWA), cleaning (Picard), sequence recalibration, filtering, variant calling and recalibration (GATK), coverage analysis (bedTools), and annotation (Annovar). It consists of a top level, coordinating workflow that invokes 8 sub-workflows, each of which implements one step of the pipeline, see Figure 1. For each step, the sub-workflows are executed synchronously in parallel over a number of samples or sub-chromosomal regions. This means that the top-level workflow submits N sub-workflow invocations for a particular step, waits until all of them complete, and then moves on to the following step.

For most of the steps number of sub-workflow invocations N equals the number of samples, with the exception when the pipeline enters the variant discovery step (highlighted with dashed blue line in the figure). Then, the data is split according to, so called, *chromosome-split* i.e. intermediate data for all samples are gathered together and split chromosome by chromosome into a specified number of sub-chromosomal regions.

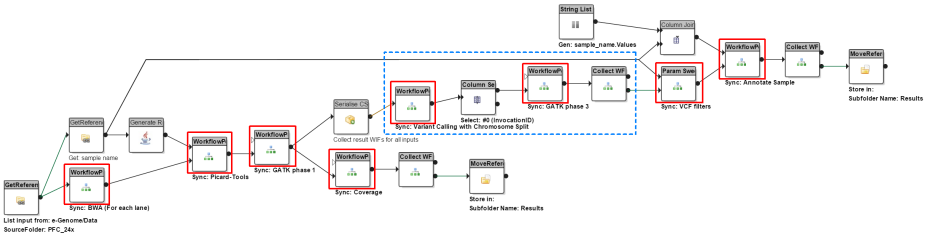

Fig. 1. NGS pipeline structure.

Both Pegasus and e-SC support enactment of scientific workflows which combine tasks into a directed acyclic graph (DAG). These systems share some common features but there are important differences between their deployment and workflow execution model.

3.1 e-SC architecture and workflow enactment model

e-SC consists of three main components: the server, database and workflow engine. It follows the common *master-worker* pattern in which the server orchestrates execution of workflows across one or more workflow engines. All e-SC components can be deployed on a single VM (*all-in-one deployment*) but in larger scale experiments, such as the NGS pipeline, they are deployed separately with single server and database VM and multiple engine VMs. The e-SC workflow enactment model is based on the *work stealing* approach: the server submits workflow invocations to a shared FIFO queue. From there invocations are pulled by the engines. Each engine can run one or more invocations concurrently in order to improve performance on a multi-core VMs.

The e-SC workflows can be of two types: basic and compound. Basic workflows execute within a single engine (within a single invocation thread on that engine), and so the data transfer between tasks is enclosed within a VM and can be very efficient. In addition there are compound workflows, which are workflows which submit one or more subworkflows. A subworkflow can again be compound or basic. Of course, data transfer between the parent and its child subworkflows is supported by the server. However, in the Cloud, workflow engines can directly communicate with scalable cloud storage such as Azure Blob Store or Amazon S3, which enables effective data transfer for large scale workflow applications. Moreover, links between blocks can transmit a list of input data and so a single parent workflow can start multiple subworkflows – one for each element on the list. Figure 1 shows the architecture of e-SC deployed in the Azure Cloud.

3.2 Pegasus architecture and workflow enactment model

WorkflowSim follows the execution model of the Pegasus WfMS. In Pegasus a workflow consists of tasks, each of which represents a node, and the task dependencies, denoted by the edges. Often, a workflow is modeled as a DAX to DAX relationship; we assume that $DAG = (V, A)$, where set of vectors $V = \{T_1, T_2, \ldots, T_n\}$ represents tasks in the workflow and set of arcs A represents data dependencies between these tasks. Moreover, data transfer between tasks is achieved using Condor File IO in the case of a non-shared file system setup.

3.3 Modelling the pipeline in WorkflowSim

Because, e-SC workflows can represent combinations of more fine-grained tasks and also due to the difference between the workflow model and possible invocation trace, we had to find a way to map an e-SC workflow into one that WorkflowSim could enact. The chosen approach was to represent the actual invocation trace of an e-SC workflow as a compatible Pegasus workflow, which could be done for the NGS pipeline using the provenance logs provided by e-SC. The provenance logs allow us to trace the complete graph of tasks and workflows that were involved in producing a specific output. They also include the block execution time and the amount of data transferred by each block. We used this

data to reconstruct our NGS pipeline workflow as a WorkflowSim workflow. Each task in Pegasus may run on different VM (unless there's clustering turned on) so we decided to model e-SCworkflows as WorkflowSim tasks. This seems to be an appropriate abstraction level, however subworkflows in e-SC can be enacted in the middle of the parent workflow, so we had to split every e-SC workflow into parts connected by the subworkflow submission blocks; this is depicted in Figure 1. Following this approach we were able to map an execution trace of the NGS pipeline ran on e-SC as a WorkflowSim workflow descriptor. Figure 2 illustrates the mapping. Note that for a different number of patient samples in the batch there is a differently sized DAX (DAG in XML) descriptor.

4 The prediction methodology

WorkflowSim requires as input a description of the execution environments, the workflow to be executed and the execution times for each task. WorkflowSim then simulates the deployment of the workflow on the environment. We wish to use this simulation to predict the performance of the NGS pipeline with different sample input sizes. From the above we can provide WorkflowSim with an input model of the NGS pipeline. However, as yet we have no idea about the task execution times or the parameters of the execution environment (MIPs and bandwidth). Our approach is therefore to gather provenance data from sample executions with small number of input samples, use this data as input to WorkflowSim to estimate the execution environment parameters and then use this to gain predictions for larger sample sizes from further WorkflowSim simulations. These predictions can then be used to find the best selection of resources (e.g. number of VMs) on which to deploy the pipeline for larger sample sizes.

4.1 Preparation phase

i) Providing parameters to WorkflowSim: A WorkflowSim toolkit must capture a complete description of the tasks, such as identification, runtime, input data sizes, and output data size. One of the trends is to make a WorkflowSim automation toolkit by providing it with an opportunity to predict the transformation parameters such as runtime and output data size. Therefore, this facility will help a user to use WorkflowSim in an easy and efficient way. As such, the NGS pipeline is launched only with the input data sizes. This will require using two prediction models and integrating them into WorkflowSim to perform this job.

ii) WorkflowSim input compatibility: This step of the preparation phase is related with the conversion from e-SC provenance traces to WorkflowSim workflow model, in order to obtain the schema of NGS pipeline that will be accepted as input workflow to the WorkflowSim.

Based on the workflow execution model description above, we convert the NGS pipeline from an execution model in e-SC Central to an execution model

in Pegasus. Figure 2 shows the result of a hierarchical pipeline graph which can be represented by DAX (DAG in XML) implementing a single NGS pipeline that represents one possible execution path of the workflow schema, where an execution consists of all nodes and edges within a workflow DAX beginning from the start node (i.e. 1-sample input) to the end node. For running 6 samples NGS pipeline, the implementation will be within 6 paths of the workflow schema (i.e. 6-sample input), and so on for running N NGS will be implemented within N paths.

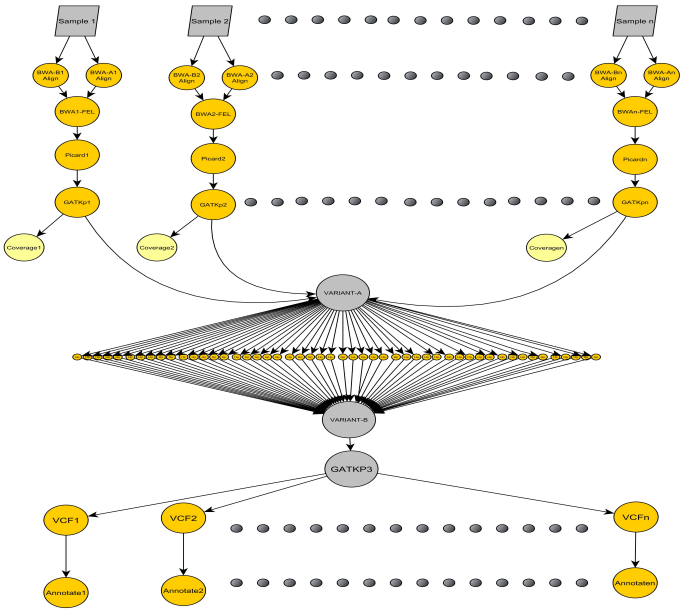

Fig. 2. Invocation graph of the NGS pipeline with N samples.

iii) Data submission format: Here we describe the workflow and present a new XML file format for workflow states that enable the WorkflowSim to interface NGS pipeline in an acceptable way. As we mentioned above, the original parser model of the WorkflowSim has been modified to analyse and parse a new XML file of the pipeline. In order to execute the e-SC workflow application in our modified WorkflowSim, workflows are described by users manually as DAXs, where the node represents individual workflow/subworkflow, and the edges represent execution dependencies between the (sub)workflows. Figure 2 illustrates the abstract description abstract as an XML file to represent the whole pipeline, which captures all logical identifiers with which the task should be invoked, such as input/output file, task identifier (id), and required run time (runtime). For

example, the following is a brief record on XML file that represents the BWA Aligan workflow.

```
id = BWA_A1_AL;
file = InputDir_BWA_A1_AL.dat;  size="7507423824";
file = BWA__A1_AL.out;          size = <->;
runtime = <->
```

In the case of size parameters, each is assigned by value of input sample, the second size and runtime parameters should be assigned by the prediction models.

4.2 Building the prediction models phase

In this phase, we describe the method to build a prediction model. To achieve this, we need the following steps:

i) Data collection and feature set The main task for this step is to shed light on the predicting modules by collecting information from the provenance file for each invocation which is needed for performance prediction. Thus,we use historical data, including the details information about the execution of the workflows in the pipeline such as invocation Id, Workflow Name, Block number, Block name, Start time, End time, Input data size, and Output data size. The following steps describe a method of extracting the above parameters from analytical provenance file:

- Sorting the information by an invocation Id and Start time. To obtain an ordering of the pipeline blocks as were executed in a real cloud. This facility helps us easily extract the actual execution time of each block.
- Extracting the runtime for each block, (i.e. runtime = End time Start time).
- Specifying the input and output data volume of each block.

Table 1 gives an example of the provenance data pertaining to the runtime and (input-output) data volume which is collected from an execution before the actual simulation of a workflow is started. All extracted parameters play a significant role in our experiments for time execution estimation of the task. Moreover, to derive the output data volume that will be used as input for next task.

ii) Extracting prediction equations To run NGS pipeline tasks, as in Figure 2, on WorkflowSim we need to know the execution time of each task before the submission is done. This aim prompted us to build prediction models for both runtime and data output sizes which are based on live performance data and using a statistical prediction approach for extracting the equations of prediction. The time parameters can be extracted from input data volume for each block and output data volume parameters. For the majority of the tasks that shown in Figure 2, it is possible to generate the prediction equations to estimate

Table 1. Basic characteristics of a selection of tasks of a 10-sample pipeline execution extracted from its provenance trace.

Pipeline step	Input data [MB]	Output data [MB]	Run time [s]
BWA1_FEL	15,862	11,336	18,807
BWA_A1_AL	7,507	7,963	9,871
PICARD1	11,342	7,411	8,941
GATKP1_1	7,417	2,766	28,703
VARIANTA	344	344	23,792
GATK phase3	55	43	943
VCF1	55	43	175
COVERAGE1	2,766	16	280
ANNOTATE1	43	204	1,206

execution time of each task based upon the size of the input data. For example, Figure 3 shows a prediction equation of the BWA1_FEL for three 6-sample input using a simple linear regression model. The same method was used to obtain further equations of the output data sizes.

However, using data input size to generate estimation equations was inadequate for the Haplotype workflows which are the entry-point to the part of the pipeline that runs under the chromosome-split regime. It means that each Haplotype workflow as the input uses data of all patient samples but is configured to read different chromosomal region of them. Thus, we used the region length as a division factor for different input sizes.

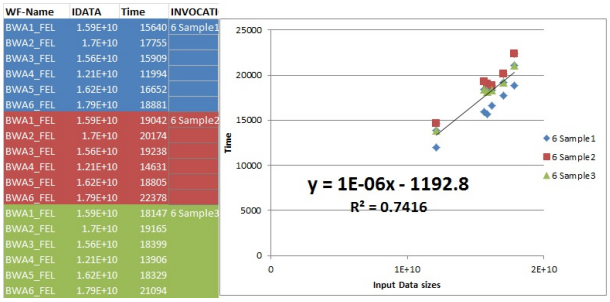

Fig. 3. Linear model of equation prediction.

iii) Set up training sets: From measurements we have obtained three data sets based on different numbers of input samples (6, 10 and 12 samples executes 3, 2 and 2 times each respectively). Each set can be used to train an input sample for execution environment parameter estimation. Thus, we can create different sized training sets for time prediction of a scalable input sample. For example, if we have 12-samples input, the prediction model can use a training set based on 6 and/or 10 samples. In the evaluation we can explore whether having more data points across all available training sets (in this case 6 and 10) provides a better prediction than using just the data from one set (6 or 10). Ideally we would get a good prediction from just using the 6 sample training set, as this would clearly be the cheapest to produce.

4.3 Integrating the derived equations phase

In this phase, we integrate the prediction equations that have been built in the building phase into WorkflowSim by considering two issues. Firstly, a run time prediction should be given for each task in case of submitting the tasks with predefined execution time to the WorkflowSim. Secondly, the output data volume should be calculated, which is a passing a factor to other tasks during running the pipeline. So, we have constructed two estimation models:

i) Runtime prediction model: One key benefit potential statistical prediction method has to match for parameters of input data to predict the parameters of the output data of task/job, making this prediction by using past information [7]. We have used the linear regression method as a solution to address the estimation. This approach manages the relationship between two variables, X is input variable (i.e. input data volume) and Y is dependent output variable (i.e. runtime), to extract Y from X.

Execution time prediction is an important factor in cloud computing and in simulation [6]. However, in a WorkflowSim, a task is assigned according to its size which is defined by the user. Therefore, it was necessary to develop a model in WorkflowSim for runtime estimation which is required to simulate the task.

ii) Output data volume prediction model: The approach taken by runtime prediction method to generate a model is followed in this method as well. However, the deference lies in two considered variables, i.e. X is input variable (i.e. input data volume) and Y is dependent output variable (i.e. output data volume), to extract Y from X.. The volume of data plays a crucial role for modelling execution time estimation. This parameter is gathered from real data in the provenance file where it is linked to the front line of the tasks (i.e. the first tasks that the pipeline execution is started). In the NGS pipeline, every task generates output data required by its child as input. This method is required for constructing a model and integrating it into WorkflowSim for output data estimation which is required to complete our models.

4.4 Extracting input parameters

In order to extract optimal metrics which have been used to set the WorkflowSim configuration for implementing our experiment, we have traced the simulator to generate these parameters, i.e. MIPs (Million instructions per second) and BW (bandwidth), based on the following steps.

1. Selecting minimum and maximum value of MIPs and BW parameters.
2. Running the WorkflowSim individually for each sample depending on the chosen MIPs and BW values with defined range in step 1 to generate estimated runtime of pipeline execution.
3. Applying an error function to find the error value between real time and estimated time for each running input sample by implementing the following formula.

$$errorratio = \sum_{j=1}^{N} (RT - ET)^2. \tag{1}$$

   Where RT and ET are Real time and Predicted time respectively. N is the number of input sample in one training set.
4. Repeat step 2 and 3 with fixed skip of MIPs and BW values until generating a minimum value of $errorratio$.

The input parameters have been generated from three scenarios as in Figures 4,5 and 6. Each scenario denotes one training set, i.e. scenario 1 denotes training set $\{6\}$, scenario 2 denotes training set $\{6+10\}$, and scenario 3 denotes training set $\{6+10+12\}$. Therefore, the above steps have been implemented on three scenarios to specifying input parameters value of the training sets.

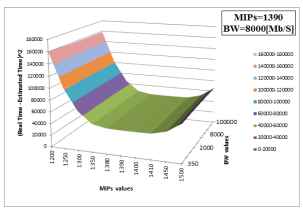

Fig. 4. Scenario 1 extracting parameters of 6-samples.

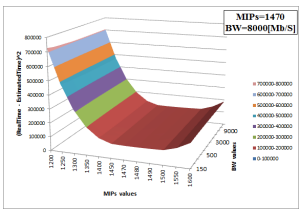

Fig. 5. Scenario 2 extracting parameters of (6+10)-samples.

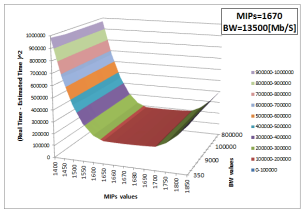

Fig. 6. Scenario 3 extracting parameters of (6+10+12)-samples.

5 Evaluation

This section presents our evaluation of NGS pipeline execution and its use with an adopted WorkflowSim to derive a scalability and performance optimization

based on estimated run time. Firstly, we describe the experiment setup, then we present our results on the accuracy of the NGS pipeline running with three input samples, finally we discuss the evaluation of the estimated results for relative errors between different running samples to derive the expectation of a Big-Data samples.

5.1 Experiment setup

The NGS pipeline is composed of 8 tasks for each path and between them there are 53 common workflows (VARIANT-A, HAPLOTYPE-CLEAR, VARIANT-B, and GATK-phase3). We ran the application with the size of Total Tasks = $N \times 9 + 53$, where N is the number of input samples. For example, if $N = 6$, then Total Tasks = $6 \times 8 + 53$. So, when we have 6 input samples, the experiment consists of 101 tasks. For each task we generate the estimation of the output data size and runtime using the simulation and then use the output data size to generate the input data size for the subsequent task. We configured WorkflowSim to simulate one datacentre and 12 virtual machines (VMs) to represent four VMs in the real cloud, each with three execution threads. We allocated the capacity of the computation unit with MIPs and bandwidth (BW) values derived from simulating the training set. Moreover, for data transfer delay, a shared file system has been used for one datacentre, where, the data transfer time is already considered in the task execution time and there is a varying setting of a BW value depending on which training set we will use. The space shared mode of the VMs has been defined as only one VM can run one task at a time. We have used three different sizes of the input sets with 6, 10, and 12 patient samples, based on the data we have available for training and validation.

5.2 Accuracy of prediction results

We conducted our experiment to evaluate the simulation accuracy prediction of the NGS pipeline execution. The goal of this experiment is to determine whether our methodology is able to predict a runtime and output data sizes of the NGS pipeline, with an enough accuracy to be useful to scale up the number of input samples.

In this case we extract input data from the provenence files from 6-sample 10-sample, and 12-sample executions and then use this to execute WorkflowSim over the same input scale to give an estimated execution time to compare with the real execution time from the provenance data. In this way, the implementing of pipeline to generate estimated run time on a specific training set that mentioned before. i.e. 6-sample would be trained on training set $\{6\}$, 10-sample would be trained on training set $\{6 + 10\}$, and 12-sample would be trained on training set $\{6 + 10 + 12\}$. As the real execution potentially vary with each run and WorkflowSim gives an single prediction, this predicted value will therefore be different to the real times. The equation that has been used to calculate a relative

error between estimated time and real time for each case as follows:

$$RelativeError = \frac{\sum_{j=1}^{N} |RT - ET|}{\sum_{j=1}^{N} (RT)} \quad (2)$$

Where RT and ET are Real time and Predicted time respectively. N is the number of input sample in one training set.

In each case the values of MIPs and BW which gives the minimum error over the training set are used to give the estimated time. The results in each case are shown in Figure 7. As expected the errors are relatively small ($< 10\%$). It might be slightly counter-intuitive that the error for the 6-sample (0.090) is larger than that for either the 10-sample or 12-sample (0.071 and 0.075 respectively). However, this is due almost entirely to one outlier in the measured execution times which has a disproportionate effect on the 6-sample as there are fewer data elements in the training set. Such experimental variance could clearly be reduced by ignoring the outlying value. However, we have chosen not to manipulate our results in this way as a) the outlying result is a genuine data point, and b) our number of data points in each training set is so small that we have no statistical basis on which to say which result is an outlier and which is not. However, as an aside, we have executed the experiments without this data point and the results do improve considerably.

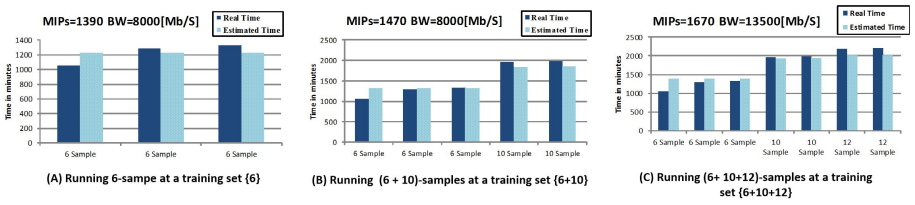

Fig. 7. The real and estimated time for different sizes of the training set.

5.3 Relative errors at input scalable

In the previous section, we verified the accuracy of the prediction model by comparing the actual and estimated runtime derived from the training sets. This process was self-reflective in that we included the provenance data from the sample size we were predicting within the training set. We now wish to consider a pure prediction of the 12-sample input by using the 6- and 10-sample sets as training data. This allows us to consider whether our approach can indeed give rise to a useful prediction in this scenario which might be used to procure infrastructure in the cloud. The results are shown in Figure 8.

Using the 6-sample data as the training set for the 12-sample case gave a relative error of approximately 0.187 (18.7%, see Figure 8A). Using the 6- and

10-sample data as the training set for the 12-sample case gave a relative error of approximately 0.112 (11.2%, see Figure 8B). In both cases the predictions underestimate the execution time. In the case of the 6-sample training set we already know that one outlying result in the provenance data is having an adverse effect on prediction and this will have had a greater effect here than in Figure 7C. What also appears to be significant is that the MIPs and BW estimations in WorkflowSim seem to be much larger for the 12-sample case than in the other two cases (see Figure 7), presumably reflecting the increased demands. Therefore, the under-estimation of MIPs and BW is causing an additional error in the prediction for the 12-sample case.

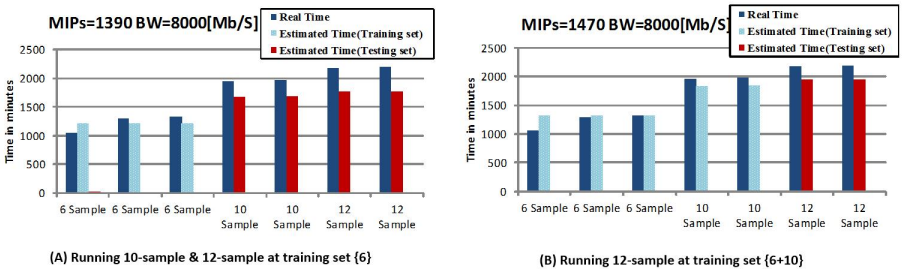

Fig. 8. The real and estimated time for different training and testing set sizes.

6 Conclusion and future work

In this paper we have described, through the use of a motivating case study, a method for predicting the runtime performance of a complex workflow based on existing simulation tools. This addresses an important and current research question. However, the effort needed to simulate this workflow should not be underestimated. There is a considerable amount of work which needs to be done to translate the e-SC workflow model into a Pegasus workflow and to obtain the baseline data which we use as a training set for the predictions. Although this cost would be reduced for subsequent workflows due to the experience of this case study, the future effort would also be significant due to the bespoke nature of each simulation set up. In our experiments we had only three data sets to consider, corresponding to 6, 10 and 12 input samples. This meant that we could validate predictions for 10 samples, using the 6 sample data as training, and for 12 samples, using the 6 and/or 10 samples for training. Ideally, the 6 sample data would have proved sufficient for good prediction as this would enable a relatively cheap means of data collection. However, the results clearly show that for 12 samples the addition of the 10 sample data offers better training. It is disappointing that the results are not more convincing. While a reasonable level of accuracy has been shown to be achieved, a similar level of accuracy

could potentially be obtained by deriving execution data at a number of scales and performing a linear extrapolation. However, one advantage of our simualation approach is that we are able to make predictions based on a single set of observations, which is clearly less costly in terms of access to the cloud provider.

The results in this paper relate to one complex workflow from one application domain. Clearly if we are to draw any general conclusions from this work then we need to conduct more case studies with different applications with different workflow structures.

References

1. Cała, J., Marei, E., Xu, Y., Takeda, K., Missier, P.: Scalable and efficient whole-exome data processing using workflows on the cloud. Future Generation Computer Systems. (2016).
2. Cała, J., Xu, Y., Wijaya, E., Missier, P.: From Scripted HPC-Based NGS Pipelines to Workflows on the Cloud. 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. (2014).
3. Calheiros, R., Ranjan, R., Beloglazov, A., De Rose, C., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract. Exper. 41, 23-50 (2010).
4. Chen, W., Deelman, E.: WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. 2012 IEEE 8th International Conference on E-Science. (2012).
5. Deelman, E.Gil, Y.: Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges. 2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06). (2006).
6. Fan, C., Chang, Y., Wang, W., Yuan, S.: Execution Time Prediction Using Rough Set Theory in Hybrid Cloud. 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing. (2012).
7. Iverson, M., Ozguner, F., Potter, L.: Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. IEEE Transactions on Computers. 48, 1374-1379 (1999).
8. Li, A., Zong, X., Kandula, S., Yang, X., Zhang, M.: CloudProphet. ACM SIGCOMM Computer Communication Review. 41, 426 (2011).
9. Long, W., Yuqing, L., Qingxin, X.: Using CloudSim to Model and Simulate Cloud Computing Environment. 2013 Ninth International Conference on Computational Intelligence and Security. (2013).
10. Pabinger, S., Dander, A., Fischer, M., Snajder, R., Sperk, M., Efremova, M., Krabichler, B., Speicher, M., Zschocke, J., Trajanoski, Z.: A survey of tools for variant analysis of next-generation genome sequencing data. Briefings in Bioinformatics. 15, 256-278 (2014).
11. Rak, M., Cuomo, A., Villano, U.: Cost/Performance Evaluation for Cloud Applications Using Simulation. 2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. (2013).
12. Rozinat, A., Wynn, M., van der Aalst, W., ter Hofstede, A., Fidge, C.: Workflow Simulation for Operational Decision Support Using Design, Historic and State Information. Lecture Notes in Computer Science. 196-211 (2008).

13. Rak, M., Turtur, M., Villano, U.: Early Prediction of the Cost of HPC Application Execution in the Cloud. 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. (2015).
14. Achour, S., Ammar, M., Khmili, B., Nasri, W.: MPI-PERF-SIM: Towards an Automatic Performance Prediction Tool of MPI Programs on Hierarchical Clusters. 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing. (2011).