Runtime Performance Prediction of Big Data Workflows with I/O-aware Simulation

Faris Llwaah*
School of Computing Science
Newcastle University
College of Computer Sciences and Mathematics, Mosul University, Iraq
f.llwaah@ncl.ac.uk

Jacek Cała
School of Computing Science
Newcastle University
jacek.cala@ncl.ac.uk

Nigel Thomas
School of Computing Science
Newcastle University
nigel.thomas@ncl.ac.uk

ABSTRACT

Modelling and simulation of Big Data analytics processes running in the cloud is a difficult problem which introduces many challenges. The major one is the collection of training data which is scarce and costly to obtain, due to large scale and long runtime of those processes. In our previous work, we proposed a methodology to model, simulate and predict the runtime of Big Data processes such as complex Next Generation Sequencing (NGS) pipelines. The major contribution of our simulation methodology is that it provides a reasonable prediction of runtime for testing data much larger than the training inputs. To further improve the accuracy of prediction we present now an extension of our previous work that can model cloud data storage. Our simulation framework is based on CloudSim and WorkflowSim, to which we have added a *shared storage* component. We present the design and implementation of the storage extension together with an evaluation performed on selected scientific workflows: the Pegasus Montage workflow and NGS pipeline implemented in e-Science Central. The evaluation shows that the proposed extension works correctly and can improve prediction accuracy for our largest 390 GB input dataset by about 16% when compared to previous results.

CCS CONCEPTS

• **Computing methodologies** → *Simulation tools*;

KEYWORDS

Big data, Data-intensive simulation, WorkflowSim, Next generation sequencing pipeline

ACM Reference Format:
Faris Llwaah, Jacek Cała, and Nigel Thomas. 2017. Runtime Performance Prediction of Big Data Workflows with I/O-aware Simulation. In *Proceedings of Valuetools (Valuetools 2017).* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4

*Faris is a PhD student at Newcastle University, UK.

VALUETOOLS 2017, December 5–7, 2017, Venice, Italy

ACM ISBN 978-1-4503-6346-4/17/12…$15.00
https://doi.org/10.1145/3150928.3150943

1 INTRODUCTION

Big Data simulation integrates existing simulation tools and performance characteristics of Big Data applications. The need for such integration is driven by the rise of Big Data analytics in the cloud [12] as an answer to tackle increasingly large amounts of information generated every day. One of the main advantages of an accurate simulation of Big Data processes in the cloud is cost savings. Whilst tuning and optimisation of the system configuration to match the requirements of the user workload has always been difficult and time consuming, considering Big Data applications deployed in the cloud, this process can become very costly.

Recently, we proposed a methodology that allowed us to simulate behaviour of complex workflow-based applications [14]. We have used this approach to predict runtime performance of our next generation sequencing (NGS) pipeline deployed on Microsoft Azure public cloud. The pipeline, implemented as a workflow in e-SC) [13], is an application used to discover and annotate variants in human exome (part of the human genome). Typically, to find variants in a cohort of 24 patients the pipeline needs to process about 400 GB of compressed data, which takes over thousand of CPU hours. Working with such applications is difficult, not only because of the volume of input data and significant time it takes to process them, but also due to intricate dependencies between tasks, data and the cloud that can cause failures, lower performance of the system and increase time and monetary cost [3].

In such a setting an accurate simulation framework can greatly help tune the configuration of the processing platform. Using our simulation we were able to predict runtime of the pipeline with accuracy close to 90%. However, due to limitations of the underlying simulation toolkit (WorkflowSim/CloudSim), the predictions of runtime for large problem sizes were less accurate. One of the contributing factors that caused inaccuracies is the simplified network and data storage model of the toolkit. Neither CloudSim nor WorkflowSim capture the network and I/O contention that occurs when multiple VMs access shared cloud storage, although this is increasingly important during the simulation of Big Data application where such contention is inevitable. In this paper we present our extension of the WorkflowSim/CloudSim environment to simulate contention in I/O operations against the shared cloud storage. We also show how the extension improves the estimation of runtime performance of our Big Data NGS workflow. The main contributions of this work are:

- a proposition of a simple model of a shared cloud storage together with its design and implementation as an extension to WorkflowSim/CloudSim,

- an evaluation of the proposed storage component using a selected data-intensive Pegasus workflow and runtime information collected from the actual execution of the NGS pipeline in e-SC deployed on the Azure cloud,
- a demonstration of the improved accuracy of our runtime predictions and discussion about the advantages and limitations of the proposed component.

The remainder of this paper is structured as follows: the next section covers background and related work. In Section 3 we formulate the main problem when working with Big Data applications. Section 4 describes the proposed shared storage component, which is followed by its evaluation in Section 5. Finally, discussion and conclusions are included in Section 6.

2 BACKGROUND AND RELATED WORK

One of the most widely used examples of a simulation environment for workflows is WorkflowSim [6]. It is an extension of the CloudSim simulator [5] and enables modelling and simulation of scientific workflows (data flows) in the cloud. The simulator follows the approach proposed by the Pegasus WMS and contains a workflow mapper, engine and scheduler and also components like the clustering engine [8]. These components allow WorkflowSim users to evaluate and optimise variety of algorithms and techniques related to workflow execution and resource allocation, which if done in the real cloud, would be time consuming and costly.

In our previous work [14] we proposed a methodology to predict runtime performance of complex Big Data applications. The methodology based on the WorkflowSim/CloudSim simulation environment allows us to use the provenance information of past application executions from our own WMS – e-SC and forecast runtime for larger and/or different user workloads. We have developed an extension to WorkflowSim which is able to predict runtime and output data size of single tasks, and so enables prediction of more complex scientific workflows including NGS pipelines.

WorkflowSim, and particularly CloudSim, have been widely used in the research community in variety of applications and problems including scheduling and provisioning algorithms, energy-aware resource allocation [1, 2, 16]. Much less commonly, research has targeted runtime prediction for Big Data applications in which modelling of storage performance plays fundamental role.

Long and Zhao [15] present an extension to CloudSim to model a new kind of distributed storage technology in the cloud to store and manage Big Data. This extension has been achieved by adding the file striping and data replica management functions, making it into a simulation platform for computing and storage. This new feature can help implement the data cloud entities and test specific data layout and replica management strategies. It is less useful, however, in modelling workflow-based applications. Sturm *et al* [18] developed a storage component in CloudSim. However, the extension focused on the mechanism to simulate object storage-based cloud services (STaaS or Storage-as-a-Service) and, again, did not consider workflows. Grozev and Buyya [11] proposed a hard disk drive (HDD) processing element representing the I/O capacity of a storage disk as an extension to the CloudSim simulator. However, that alone is not enough to model the I/O contention in the cloud.

Louis *et al* [17] proposed CloudSimDisk, another extension to CloudSim aimed at modelling and simulation of energy-aware storage in cloud systems. It was based on an analytical energy consumption model for hard drives and considered transaction time and energy consumption related to adding and retrieving binary files used by VMs.

Of the work not related directly to CloudSim or WorkflowSim, Costa *et al* [7] prediction mechanisms to estimate the performance of a workflow application or storage operation. Based on the target application's characteristics, the proposed mechanisms can speed up the exploration of the configuration space. The effectiveness of this mechanism was evaluated in some scenarios, including different system scale, hardware platform and configuration choices. The mechanisms provided high enough accuracy to support the user's decisions about configuration and provisioning the storage system, while being less resource-intensive than running the real applications. This work is conceptually similar to our work to detect and fix potential performance prediction issues, but differs in the development of a complex and error-prone distributed storage system specialised in workflow applications

To the best of our knowledge the existing research in the area of cloud simulation falls short in addressing the problem of I/O contention that arises from multiple VMs accessing the shared cloud storage. Therefore, in this paper we propose a model and extension to the WorkflowSim/CloudSim toolkit that can help address this gap.

3 PROBLEM FORMULATION

As mentioned earlier, in our previous work we proposed a method for predicting the runtime performance of complex Big Data workflows. The method is able to take into account two main factors that may affect the runtime of the applications: the size of computational tasks vs CPU speed of the simulated environment and the size of the input and output data vs network bandwidth. However, due to the nature of the data intensive problems, the scarce availability of training data and limitations of the simulation environment, our solution was able to produce predictions with limited accuracy. Specifically, using a small 6-sample input datasets over 12 VMs our method was able to predict runtime of 10-sample and 12-sample workloads with relative error of about 15% and 19%, respectively; Fig. 1 shows results for a training set consisting of 6-samples.

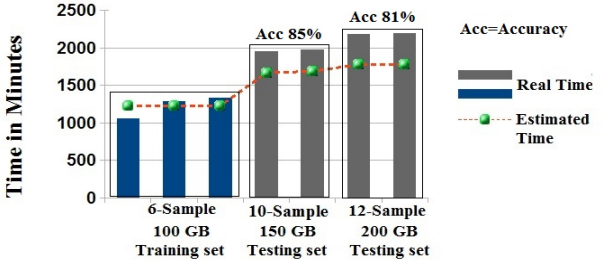

Figure 1: The real and estimated time for different sizes of the training set.

In all cases our predictor underestimated the real execution time. The bigger the gap between the training and testing set, the larger

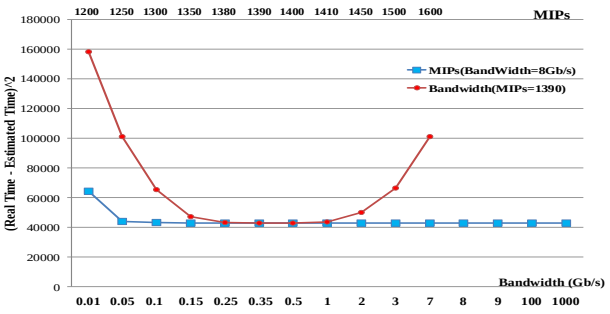

Figure 2: MIPs and BandWidth curves relative with error rate.

the underestimation. Each sample involving transfer and processing of hundreds of gigabytes of data, which suggested that the I/O operations were a major factor of inaccuracies. We confirmed this hypothesis by running a sensitivity analysis of our simulation with respect to MIPs and bandwidth. As shown in Fig. 2, for narrow range between 1350–1450 MIPs the error value drops significantly. Conversely, changes in the network bandwidth affected the error only when it was about 10 Mb/s and below, a value unrealistically small when considering public cloud environments, such as Microsoft Azure. Moreover, extremely large values of bandwidth (10Gb/s and more) did not increase the error which indicated serious flaws in the network model of the WorkflowSim and CloudSim environments.

Using WorkflowSim to calculate the transfer time of data between VMs only considers the size of the data and network bandwidth, while CloudSim ignores any contention that may arise from multiple VMs accessing the same shared data. Without analysing the source code, we confirmed this by running a simple experiment in which 1–100 VMs executed a one-task workflow that accessed the same input file and produced one output file. In all cases the execution time was the same, clearly indicating that no network contention model was in place.

4 I/O SIMULATION MODEL

As indicated above, one source of inaccuracy in our prediction method comes from a simplistic model of network and storage in the simulation environment we used. To address this problem we decided to design and implement a shared storage component to simulate contention between VMs accessing data in CloudSim and WorkflowSim simulation. Importantly, we focus this work on the shared storage only and do not model aspects related to a shared network environment. Our goal is to cover both the relevant network and storage issues under a single I/O contention model.

The existing I/O model for scientific workflows implemented by WorkflowSim approximately follows the data transfer mechanisms of Pegasus. Briefly, each workflow task requires all its input data to be staged in to the machine where the task is going to execute, and then after the processing is finished, the output data is shared by the workflow management system with the downstream tasks. This simple model, illustrated in Fig. 3, involves three distinct periods in the task lifecycle following the VM allocation event: data stage-in or (R)ead, (P)rocess, and data stage-out or (W)rite. Notably, in Big Data workflows data stage-in and -out take significant amounts of time and are the main source of contention between multiple VMs executing the pipeline.

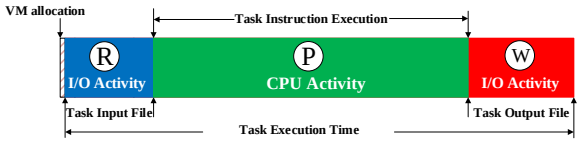

Figure 3: Time task model before I/O response time considered

To model the data transfer speed between task VMs and shared storage WorkflowSim uses single parameter – network *bandwidth*. By changing bandwith the user can influence the amount of time a task spends in the stage in/out periods. However, the stage in/out time is calculated simply as $data\_size/bandwidth$, and so no contention is simulated at all.

4.1 Pipeline Adaptation

WorkflowSim was primarily designed to simulate Pegasus workflows, thus modelling our NGS pipeline designed in e-SC required a certain level of adaptation. Both systems are similar in that they support execution of scientific workflows (data flows), yet there are some important differences between them.. Specifically, e-SC workflows are more fine-grained and can operate at two levels: *basic* and *composite*. A basic workflow may include many tasks, yet all of them run on the same machine (workflow engine). That helps to optimise execution of small, short-lived tasks because data transfer between them is local. A composite workflow also consists of tasks but some of them invoke basic and/or composite *subworkflows*. Each of these subworkflows may run on a different workflow engine, which helps handle data and task parallelism that often occurs in scientific analyses.

The distinction between basic and composite workflows is especially important when Big Data workflows are designed because it heavily affects how data is transferred between tasks. Tasks of a basic workflow pass data via a local filesystem, whereas in composite workflows data needs to be shared between VMs, and so transferred to/from the shared storage provided by e-SC. Yet, finding a balance between effective parallel execution across a number of machines and using fast local data transfer is not obvious, as discussed in [3].

Another difference between the systems is that e-SC workflows give a lot of leeway in how and at which stage workflows share their data. Similarly to Pegasus, they may follow the read-process-write pattern where all input data is staged in before the core tasks execute and then staged out afterwards. But they may also share data and invoke subworkflows in the middle of the execution.

Our NGS pipeline largely follows the read-process-write pattern, so at the end of processing the outputs of a pipeline step are transferred to the e-SC storage and then to the engine that is going to execute the following step. But in a few cases NGS subworkflows break this rule, so we split their WorkflowSim models into parts such that each part consists of the three (R)-(P)-(W) stages in order.

4.2 Enhanced I/O Model

To simulate I/O contention we propose a simple serialised storage model based on a single FIFO queue (Fig. 4). Following the existing task model, we assume that before task execution all required input files have to be staged in from the shared storage. That is collectively represented by a single read request (R). And, similarly, upon task completion all output files are staged out to the shared storage, denoted by a single write request (W). Then, using the queue we can introduce contention as the calculated delay in response to read/write requests.

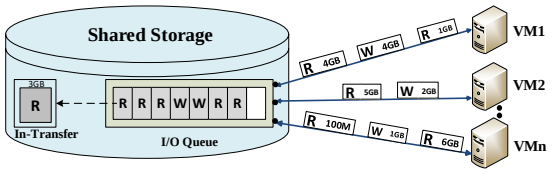

Figure 4: The VMs uses single queue to the shared storage.

In order to keep the enhanced I/O model simple we also use single parameter to compute the storage response time. This time, however, *bandwidth* represents the cumulative delays of network and storage elements involved in transfer and storing/retrieving data. This simplification follows the perception of the cloud user who has no easy way to distinguish between these two types of delay in the real cloud environment. To compute storage response time T_n of request n we use the following formula:

$$T_n = \sum_{i=1}^{n} t_i + r_0 \tag{1}$$

where t_i is the transfer time of the i-th request in the queue, n is the length of the queue after adding the new request and r_0 is the remainder of transfer time t_0 of the request being in transfer. Briefly, remainder $r_0 \in \langle 0, t_0 \rangle$ indicates how much time is needed to complete the request in transfer when a new I/O request arrives. As previously, the data transfer time is calculated as:

$$t_i = \frac{data\_size}{bandwidth} \tag{2}$$

To illustrate the proposed enhancements we conducted a simple experiment with three tasks running on three VMs and show the tasks' life-cycle on the Gantt chart in Fig. 5. The figure explicitly highlights I/O delay time (D) which is the difference between the response time returned by the shared storage and the data transfer time as if there was no contention.

Initially, when the tasks are submitted, they are scheduled to run on three VMs in parallel, and so all issue their read requests to the shared storage at the same time. While Task_1 starts reading immediately, the other two are delayed such that Task_2 waits until Task_1 finishes reading and Task_3 waits until the other two finish. Similarly, at the end the Task_3 write operation is delayed because it needs to wait for Task_2 write request to complete. Coincidentially, Task_1 ends execution close to the moment when Task_2 issues a write requests, so the latter task experiences no delay in writing. The chart clearly shows the serialisation of the requests and the way we are going to simulate I/O contention.

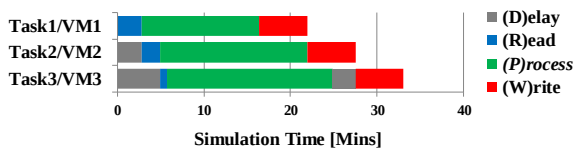

Figure 5: Gantt chart of three tasks with I/O read and write operations.

4.3 Extending the Simulation Framework

The implementation of the proposed shared storage mechanism involved extending the WorkflowSim environment as well as modifying some components of CloudSim, such as Cloudlet. Fig. 6 shows an overview of our newly added component SharedStorage which is highlighted in red and its communication with two other components from the workflowSim layer: WorkflowDatacenter and WorkflowScheduler.

Figure 6: Relational Diagram of Shared Storage Model

The WorkflowDatacenter component is an extension of the Datacenter component in CloudSim. They contribute a list of functions to process a Cloudlet submission and verify if some Cloudlet inside in it already finished. WorkflowDatacenter has own functions to calculate data transfer time, update the submission time of the Cloudlet and register a file to the storage if it is an output file. The WorkflowScheduler component is used to match a Cloudlet to a VM based on a user-defined scheduling algorithm. It assigns VM management, as VM creation, submission of task to this VMs and destruction of VMs. Finally, we focus on the SharedStorage component description and its ability of associating and coordinating with other components. However, the framework works based on the events which are generated dynamically and executed chronologically. It supports modelling of Cloud entities such as Workflowdatacenter, WorkflowScheduler, etc. Therefore, to enable this feature, we adopt an event-based approach component. Where this component maintains itself, checks whether it has received an event (i.e. Task status IN-WRITE) and whether it should delay a task to keep VM allocated until delay finishes, then send a task to the sender WorkflowScheduler. The function of our proposed model can be achieved by tracking the following mechanism:

► At a particular time, a WorkflowSim task scheduler decides which tasks can be scheduled for execution on free VMs. While each scheduled task receives a VM allocation it starts reading, then WorkflowDatacenter interacts with the SharedStorage component to send (R) request to the I/O Queue (Storing the request at the Queue End).

► Whenever SharedStorage receives a Read request it interacts with the WorkflowDatacenter to add the required transfer and delay time of the task read operation (c.f. grey and blue slices in the Gantt charts).

► At a specific time, a completed task is available inside WorkflowDatacenter component, then it interacts with the SharedStorage component to send (W) request to the I/O Queue (Storing the request at the Queue End).

► During the (R/W) requests are waiting in I/O Queue, the SharedStorage will process them to calculate a transfer and delay time of the task, the process will repeatedly be done as in Alg. 1.

► When a task has completed, WorkflowDatacenter schedules it as an event to the SharedStorage to change a task status from IN-EXECUTION to IN-WRITE for delay and consequently to extend VM allocation time.

► Immediately upon delay finishes the SharedStorage will send task completion to the WorkflowScheduler.

The above mechanism steps must be performed iteratively until the simulation finished.

Algorithm 1: Process (R/W) Request in SharedStorage

```
1  QueueTime – the completion time of the last I/O request (0 at the simulation
     start) ;
2  Input: cl : Cloudlet ;
3  cl.setIOArrivalTime := CloudSim.Clock() ;
4  if CloudSim.Clock()>QueueTime then
5  |   QueueTime :=CloudSim.Clock();
6  end
7  cl.setIOStartTime := QueueTime;
8  FTT := compute_file_transfer_time (see Eq. 2);
9  QueueTime := QueueTime + FTT;
10 cl.setIOEndTime := QueueTime;
```

5 EVALUATION

To evaluate the proposed shared storage extension we conducted two experiments. First involved the Pegasus Montage workflow [10] and allowed us to validate the correctness of the implementation. In the second we used our NGS pipeline to observe influence of the extension on the prediction accuracy.

5.1 Running the Montage workflow

To validate the correctness of the implementation we tested our shared storage extension by running the Montage workflow, a workflow far more complex than the initial one used in the experiment presented earlier in Fig. 5. The Montage workflow is an astronomy application used to generate custom mosaics of the sky based on a set of input images and is considered as data-intensive [9]. Most of the tasks in Montage have little CPU needs and spend their execution mainly on file read and write operations. In our experiment we used the variety of the workflow consisting of 25 tasks, as shown in Fig. 8.

To run the simulation we used the latest available WorkflowSim 1.1.0 extended with our shared storage component. The simulation environment included 5 VMs, each VM had 1 CPU-core (MIPs = 1000) with 512 MB of RAM. Bandwidth was set to 1000.

The experiment results, presented in Fig. 7, show the serialisation of the read and write operations enforced by the storage component. The operations were delayed such that they did not overlap with each other and once the current operation in-transfer had completed, the next in the I/O queue was immediately taken over by the shared storage. Given the results from the execution of a relatively complex workflow, we were assured that the proposed implementation behaved as expected and could be used in our prediction framework, as presented next.

5.2 Evaluation of Runtime Performance

The overall goal of this work is to improve the accuracy of runtime prediction of complex Big Data workflows such as NGS pipelines. Analysing the past results of our prediction experiments, we observed that the source of inaccuracies might stem from WorkflowSim and CloudSim not being able to simulate I/O contention. Thus, we added the shared storage extension to our prediction framework and re-executed our earlier experiments with a new dataset. The new dataset was needed to increase the number of samples and enlarge the test sets. That allowed us to test the prediction when changing both the number of samples and the number of VMs.

5.2.1 Experiment setup. For the purpose of evaluation we ran the NGS pipeline in e-SC deployed in the Azure cloud over 3, 6 and 12 VMs. Each VM was of class D13 with 8-core CPU, 56 GiB of RAM and 400 GB of local SSD storage and was used to run 4 concurrent workflow execution threads. The tests involved sequencing of 6, 12 and 24 patient samples with data size in range of 98–390 GB. As the training data we used the smallest executions of 6 patient samples ran on 3 VMs (12 workflow execution threads). For each *evaluation point*, a specific number of VMs and input samples, we collected runtime information of three executions in the cloud. The size of the input sets was a trade-off between what is used in clinical practice (30–40 patient samples) and the cost it takes to run the pipeline in the cloud. The smallest 6-sample training set was the minimal input size for which the pipeline could complete successfully.

The simulation environment was configured such that each simulated VM represented a workflow execution thread in the real cloud. We trained our model on 12 WorkflowSim VMs running 6 patient samples and then tested it on 12, 24 and 48 simulated VMs. As one VM could run only one task at a time, we used the space shared mode for task scheduling.

Both training and testing phases were performed twice – with and without the shared storage component. Training the model gave us the optimal simulation parameters of: MIPs = 1430 and bandwidth = 50 Mb/s in simulations without I/O contention and MIPs = 1305 and bandwidth = 985 Mb/s in simulations with I/O contention switched on; clearly, the value of 985 Mb/s × 12 is much closer to the maximum throughput of the Azure Cloud Storage set at 10 and 15 Gb/s per storage account for ingress and egress access, respectively.

The original NGS pipeline is composed of three stages: the first and last running in the *sample-split* mode and the middle one running in, so called, *chromosome-split* mode. The sample-split stages are largely sequential and involve eight and two tasks, respectively. Depending on the number of input samples, they are replicated such that each input sample runs a separate sequence of tasks. The chromosome-split stage includes one join task in front, then a fixed number of tasks each running in parallel over a dedicated chromosomal region, and then two tasks at the end. Overall, the pipeline consists of $9 \times N + 53$ tasks, where N is the number of input samples

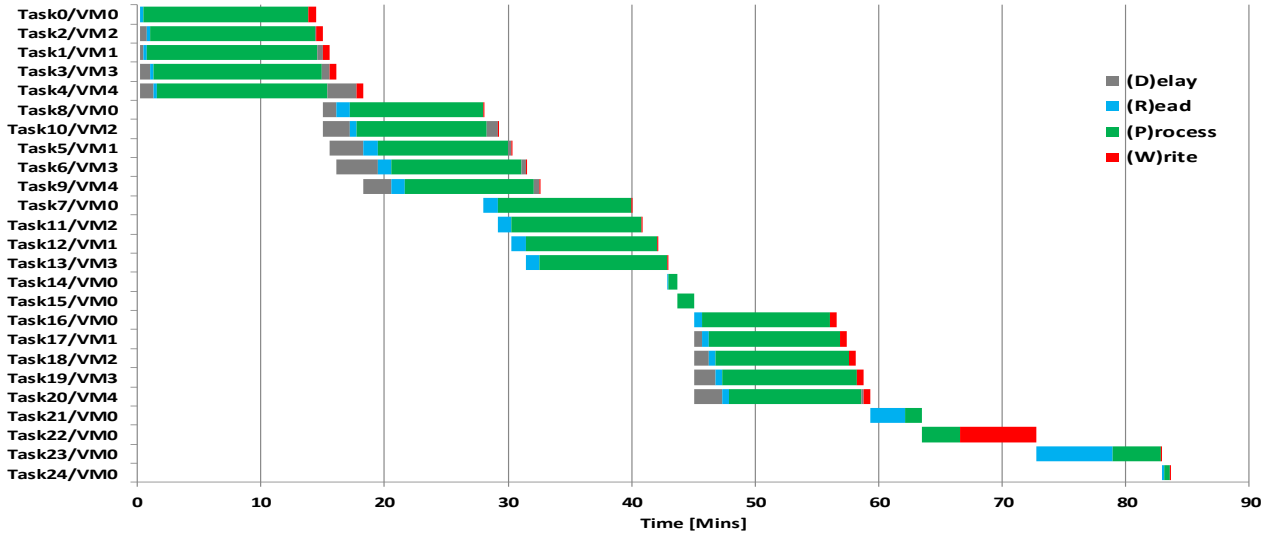

Figure 7: The Gantt chart of the Montage workflow.

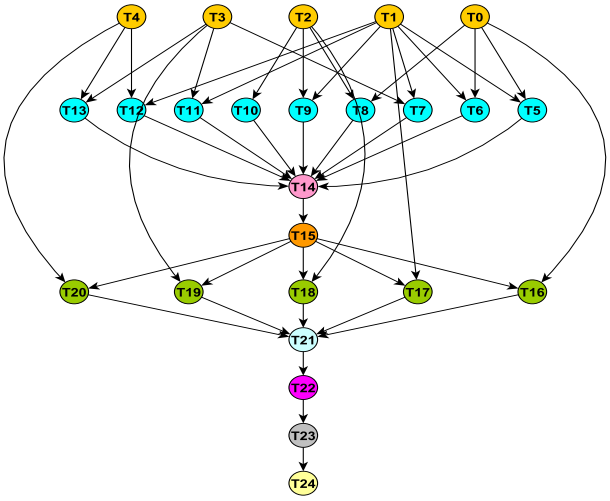

Figure 8: The structure of the 25-task Montage workflow.

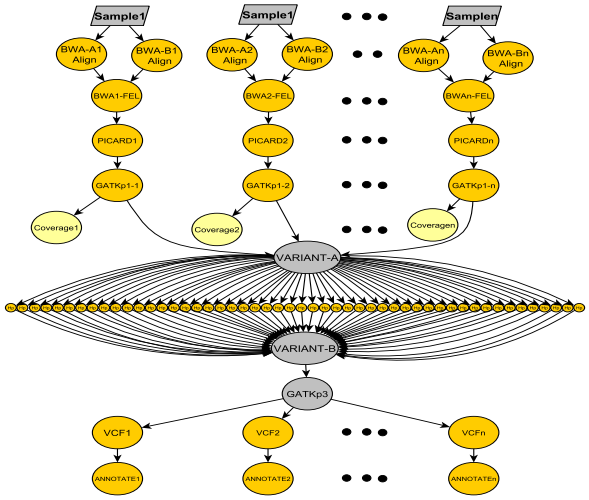

Figure 9: The structure of our NGS pipeline modelled in WorkflowSim.

(see Fig. 9). For example, in the biggest setting our WorkflowSim simulation included 269 tasks. As mentioned earlier, however, in the real pipeline each of the simulated tasks consists of a number of workflow blocks modelled in e-SC such that the 24-sample run involves execution of thousands of tasks and requires thousands of CPU hours to complete.

5.2.2 Results. Analysing the provenance information collected by e-SC, we obtained the exact times and data sizes of each task in the real pipeline workflow. We converted them into the time and input/output data size of each simulated task and compared these real execution times (RT) with estimated time simulated by the framework with shared storage switched off (ET) and on (ET+SS). The results are shown in Fig. 10.

The graph indicates significant improvement of accuracy when simulations used the shared storage component. Using only one evaluation point (6-sample input over 12 VMs) as the training set, the framework was able to predict much larger experiments with relative error as small as 2% for 12 and 24 samples running on 12 VMs. Whereas, without the shared storage the error was over 15% for 12 samples running on 12 VMs (See Scope from Fig. 1 in Fig. 10). There is a difference in the accuracy of about 4% if compared with Fig.1 because we have used a new dataset.

Simulations with shared storage rendered prediction across all the testing set with relative error no larger than 15%. The largest error we observed was for the samples running on 48 VMs, the

evaluation point most distant from the training set. Although the results are much more promising than what we can achieve without the shared storage component they still indicate some room for improvement.

In order to compare the NGS pipeline execution in the Azure cloud with that in our Framework simulation, so from Fig. 11-A we can see that for all set of input samples tested over 12 VMs behaved well as most likely Azure, but when tested them over 24 VMs behaved partially like Azure. A bit overestimation time against the real Azure time when they were tested over 48 VMs because a larger sample input and many VMs can cause congestion in the I/O Queue.

Fig. 11-B shows the Framework throughput (samples per day) about the number of simulation VMs. It is compared with the Azure running time, from one point (6-Sample, 12 VMs) we can predict an ideal linear speed-up and describes gains in the processing by increasing VMs.

Finally, Fig. 11-C shows how well the different configurations scale when compared to baseline with 12 VMs, using a measure of Relative Processing Effectiveness (RPE; the higher, the better). With a particular input sample size s, the $Ts(n)$ is an estimated time for a configuration with n VMs , we define the RPE of the VMs configuration relative to the baseline b-VM configuration as:

$$RPEs(b, n) = \frac{bTs(b)}{nTs(n)} \tag{3}$$

100% effectiveness is achieved when $Ts(n) = \frac{b}{n}Ts(b)$. For example, resources are perfectly utilised when doubling the number of VMs (n = 2b) results in the halving of the estimated time relative to the baseline ($Ts(2b) = 12Ts(b)$) on the same input size. In our experimental simulation, we have used baseline b=12, n=24 and n=48, and s ranging from 6 to 24. In a case of comparing both configurations (n=24 and n=48) with their actual and estimated time, as seen in our chart, are most similar behavior. So, from this graph we can investigate how much running NGS faster when doubling the number of engines on the s-sample input. For example, on of the estimated time in our chart, $\mathrm{RPE}_{12}(12, 24) = 70\%$, indicates that doubling the number of VMs on the 12-sample input is only of about 1.4× faster than the baseline; ideally, it would be $2times$ faster. These results show that for larger configurations the estimated time grows slowly with the number of samples as in actual behavior of Azure cloud. For the smallest, 6-sample, input we observed very little gain when increasing simulation VMs (c.f. throughput). Only for the biggest, 24-sample, input the pipeline showed good effectiveness about 92% when running over 24 VMs (see Fig. 11-C).

Thus, for a complex NGS pipeline, our Framework can deliver enough scalability information to ensure that the application workload with appropriate resources deployment in an efficient way.

6 DISCUSSION AND CONCLUSIONS

Runtime prediction of Big Data analyses plays an important role in the design and deployment of data analytics systems. Cloud platforms make it very easy to provision virtually infinite resources, but they give little support in finding the optimal configuration for a given user workload. In Big Data analyses that is particularly important because such analyses consume significant amount of cloud time which directly translates into monetary cost, and so any mismatch between the provisioned resources and actual workload is likely to increase the cost.

Big Data applications operate on an amount of data that is difficult to handle and takes significant amount of time and cost to transfer and process. The influence of these large amounts of data becomes even more apparent when the processing nodes of the data analytics system work in parallel, which can generate significant contention in access to the data storage. In this paper we have presented a model and implementation of a simple serialised shared storage component embedded into the WorkflowSim environment to simulate I/O contention.

The storage component is based on a single FCFS queue and handles one read/write request at a time. Despite its simplicity the evaluation results show that the proposed extension is a promising approach to improve the accuracy of the runtime prediction. Using only a very small training set consisting of 3 measurements of the smallest executions, we were able to predict runtime of much larger configurations with relative error in the range of 0.4–15% and median 8%. That gives users the ability to tailor the resource configuration to their workload and potentially save a lot of costs in finding the configuration that matches their needs, i.e. can process the workload in a predictable expected time.

Interestingly, the proposed simple storage component can simulate with good accuracy much more complex data access mechanism of the real cloud. In Azure, where our NGS pipeline was deployed, upload and download of blobs (data objects) is intrinsically parallel – each blob can be accessed independently of others [4]. The reason why our serialised storage component can well simulate the parallel blob storage in Azure stems from the fact that the actual NGS pipeline implementation is very synchronous in nature and due to large amounts of data it saturates the bandwidth available for a single cloud storage account. Thus, although each pipeline step runs many subworkflows in parallel across multiple VMs, before moving to the following step the pipeline waits until all subworkflows complete. And these synchronisation points at each step make the serialised and parallel access to storage consume similar amount of time. Whether the proposed sequential read/write storage would be able to simulate more sophisticated workflow enactment models is left for the future work. Nevertheless, as shown in our previous work [3], asynchronous workflow enactment models can introduce their own set of issues and not necessarily are the best to support Big Data applications.

The results we achieved and presented indicate that there is still some room for improvement in the runtime prediction. Thus, an implementation of a more sophisticated parallel storage component for WorkflowSim is on our list of future work.

Software availability: The source code has been released as open source code and can be downloaded from:

https://github.com/Farisllwaah/NGS-Framework. It includes a demonstration of how to use it and several examples.

7 ACKNOWLEDGMENTS

A very special thanks to Paul Watson for his funding to collecting the new dataset. Also this work was partially supported by EPSRC

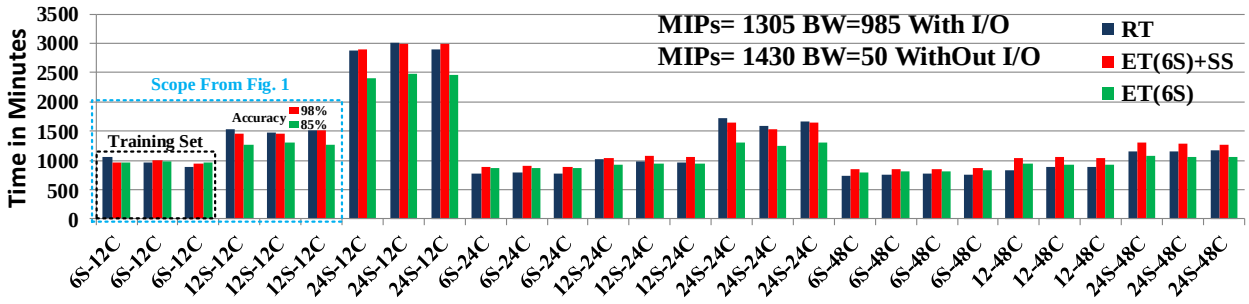

Figure 10: The Runtime Results with/without I/O

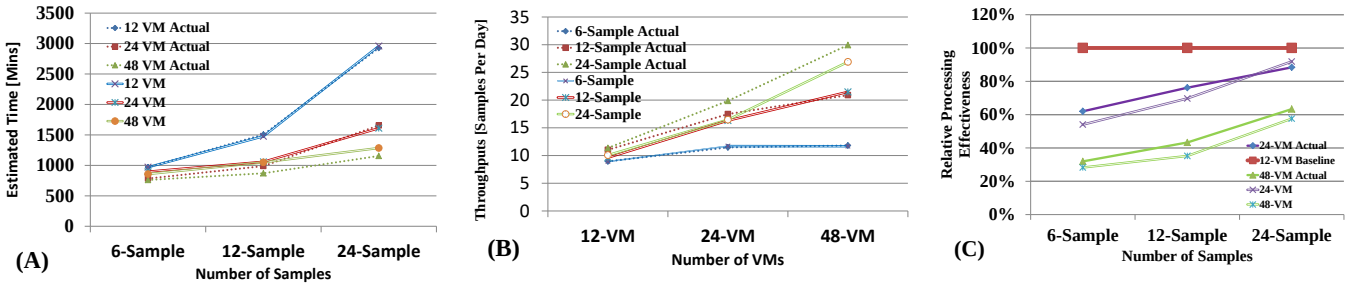

Figure 11: Response time, throughput and relative processing effectiveness of the pipeline with the increasing number of input samples and different number of workflow engines in the system.

grant no. EP/N01426X/1 in the UK and by a grant from the Microsoft Azure Research programme.

REFERENCES

[1] Ghalem Belalem, Fatima Zohra Tayeb, and Wieme Zaoui. 2010. *Approaches to Improve the Resources Management in the Simulator CloudSim.* Springer Berlin Heidelberg, Berlin, Heidelberg, 189–196. https://doi.org/10.1007/978-3-642-16167-4_25

[2] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems* 28, 5 (2012), 755 – 768. https://doi.org/10.1016/j.future.2011.04.017 Special Section: Energy efficiency in large-scale distributed systems.

[3] J. Cała, E. Marei, Y. Xu, K. Takeda, and P. Missier. 2016. Scalable and efficient whole-exome data processing using workflows on the cloud. *Future Generation Computer Systems* 65, Supplement C (2016), 153 – 168. https://doi.org/10.1016/j.future.2016.01.001 Special Issue on Big Data in the Cloud.

[4] Brad Calder, Ju Wang, Aaron Ogus, and et al. 2011. Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles – SOSP '11.* ACM Press, New York, New York, USA, 143–157. https://doi.org/10.1145/2043556.2043571

[5] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 41, 1 (Jan 2011), 23–50. https://doi.org/10.1002/spe.995

[6] Weiwei Chen and Ewa Deelman. 2012. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th International Conference on E-Science.* IEEE, 1–8. https://doi.org/10.1109/eScience.2012.6404430

[7] Lauro Beltrão Costa, Samer Al-Kiswany, Hao Yang, and Matei Ripeanu. 2014. Supporting storage configuration for I/O intensive workflows. *ICS '14 Proceedings of the 28th ACM international conference on Supercomputing* (2014), 191–200. https://doi.org/10.1145/2597652.2597679

[8] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. 2004. *Pegasus: Mapping Scientific Workflows onto the Grid.* Springer Berlin Heidelberg, Berlin, Heidelberg, 11–20. https://doi.org/10.1007/978-3-540-28642-4_2

[9] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. 2008. The Cost of Doing Science on the Cloud : The Montage Example. In *SC 08 Proceedings of the 2008 ACM/IEEE conference on Supercomputing.* Austin, TX, USA.

[10] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, Anastasia Laity, Joseph C Jacob, and Daniel S Katz. 2005. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific programming.* 13, 3 (2005), 219–237.

[11] Nikolay Grozev and Rajkumar Buyya. 2013. Performance modelling and simulation of three-tier applications in Cloud and Multi-Cloud environments. *Computer Journal* 58, 1 (2013), 1–22. https://doi.org/10.1093/comjnl/bxt107

[12] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems* 47 (2015), 98–115. https://doi.org/10.1016/j.is.2014.07.006

[13] Hugo Hiden, Simon Woodman, Paul Watson, and Jacek Cala. 2012. Developing cloud applications using the e-Science Central platform. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 371, 1983 (2012). https://doi.org/10.1098/rsta.2012.0085 arXiv:http://rsta.royalsocietypublishing.org/content/371/1983/20120085.full.pdf

[14] Faris Llwaah, Jacek Cała, and Nigel Thomas. 2016. Simulation of Runtime Performance of Big Data Workflows on the Cloud. In *European Workshop on Performance Engineering.* Springer, 141–155.

[15] S. Long and Y. Zhao. 2012. A Toolkit for Modeling and Simulating Cloud Data Storage: An Extension to CloudSim. (Dec 2012), 597–600. https://doi.org/10.1109/ICCECT.2012.160

[16] Wang Long, Lan Yuqing, and Xia Qingxin. 2013. Using CloudSim to Model and Simulate Cloud Computing Environment. (2013), 323–328.

[17] Baptiste Louis, Karan Mitra, Saguna Saguna, and Christer Ahlund. 2015. CloudSimDisk: Energy-Aware Storage Simulation in CloudSim. (Dec 2015), 11–15. https://doi.org/10.1109/UCC.2015.15

[18] Tobias Sturm, Foued Jrad, and Achim Streit. 2014. Storage CloudSim a simulation environment for cloud object storage infrastructures. *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science* (2014), 186–192. https://doi.org/10.5220/0004956401860192