School of Computing

Simulation of the Performance of Complex Data-Intensive Workflows

Faris Llwaah

Submitted for the degree of Doctor of Philosophy in the School of Computing, Newcastle University

August 2018

Abstract

Recently, cloud computing has been used for analytical and data-intensive processes as it offers many attractive features, including resource pooling, on-demand capability and rapid elasticity. Scientific workflows use these features to tackle the problems of complex data-intensive applications. Data-intensive workflows are composed of many tasks that may involve large input data sets and produce large amounts of data as output, which typically runs in highly dynamic environments. However, the resources should be allocated dynamically depending on the demand changes of the workflow, as over-provisioning increases the cost and under-provisioning causes Service Level Agreement (SLA) violation and poor Quality of Service (QoS). Performance prediction of complex workflows is a necessary step prior to the deployment of the workflow. Performance analysis of complex data-intensive workflows is a challenging task due to the complexity of their structure, diversity of big data, and data dependencies, in addition to the required examination to the performance and challenges associated with running their workflows in the real cloud.

In this thesis, a solution is explored to address these challenges, using a Next Generation Sequencing (NGS) workflow pipeline as a case study, which may require hundreds/thousands of CPU hours to process a terabyte of data. We propose a methodology to model, simulate and predict runtime and the number of resources used by the complex data-intensive workflows. One contribution of our simulation methodology is that it provides an ability to extract the simulation parameters (e.g., MIPs and BW values) that are required for constructing a training set and a fairly accurate prediction of the runtime for input for cluster sizes much larger than ones used in training of the prediction model. The proposed methodology permits the derivation of runtime prediction based on historical data from the provenance files. We present the runtime prediction of the complex workflow by considering different cases of its running in the cloud such as execution failure and library deployment time. In case of failure, the framework can apply the prediction only partially considering the successful parts of the pipeline, in the other case the framework can predict with or without considering the time to deploy libraries. To further improve the accuracy of prediction, we propose a simulation model that handles I/O contention.

Declaration

I hereby declare that this thesis is my own work unless otherwise stated. No part of this thesis has been previously submitted for a degree or any other qualification at Newcastle University or any other institution.

Faris Llwaah

I confirm that, to the best of my knowledge, this thesis is from the student's own work and has been submitted with my approval.

Supervisor

Dr. Nigel Thomas

Publications

CONFERENCE

Over the course duration of my PhD I have published in conference some of the work. Below are the articles that declared my contributions in producing them, they are being used in this thesis (shown in parenthesis):

1. F. Llwaah, J. Cała, and N. Thomas. Runtime Performance Prediction of Big Data Workflows with I/O-aware Simulation. In 11th EAI International Conference on Performance Evaluation Methodologies and Tools. ACM, 2017. (Chapter 6)
2. F. Llwaah, J. Cała, and N. Thomas. Simulation of runtime performance of big data workflows on the cloud. In European Workshop on Performance Engineering, pages 141-155. Springer, 2016. (Chapter 4)
3. F. Llwaah, N. Thomas, and J. Cała. Improving MCT scheduling algorithm to reduce the makespan and cost of workflow execution in the cloud. In: UK Performance Engineering Workshop. 2015, Leeds, UK: University of Leeds.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Nigel Thomas for his professional guidance, continuous support, unreserved advice and enthusiastic encouragement throughout my whole PhD study. He has supported me academically and morally from the beginning to the end of finishing this thesis.

My appreciation also extend to Dr. Jacek Cała who directly contributed to the discussion of my thesis work with great patience and immense care. He has been demonstrative and generous in his guidance and support valuable advices throughout my whole PhD study.

I would also like to thank my examiners, Professor Stephen Jarvis and Dr. Phillip Lord for the valuable viva discussion. I am also grateful for reviewing my thesis and also for providing constructive feedbacks and detailed comments.

I would like to thank my home country Iraq, the Ministry of Higher Education and Scientific Research in Iraq and Mosul University for providing the opportunity for me to undertake my doctoral study in Newcastle University.

During my study journey, I have had the pleasure of knowing alongside numerous kind and helpful people, I would also like to thank all the members of staff at the School of Computing at Newcastle University.

Some special words of gratitude go to my friend who has always been a major source of support when I have been through a difficult time, Man Balola. I would also express my gratitude to the following people for their support during the study period, Rawaa Qasha, Soulaf Kakel, Choman Abdullah, Osama Alrajeh, Said Kamil, Ayman Noor, Devki Nandan, Mohammed Alotaibi, and Ali Alssaiari.

Nobody has been more important to me in the pursuit of this stage than the members of my family. I would like to thank my lovely family. Most importantly, I wish to thank my loving and supportive wife, and my three wonderful children, Mohammed, Aya and Mayar, who provide unending inspiration.

Contents

List of Figures

List of Tables

Acronyms

AEF	Automated Emulation Framework
CPU	Central Processing Unit
CSV	Comma Separated Values, is a type of data format
DAG	Directed Acyclic Graph
DAX	DAG in XML
DCG	Directed Cyclic Graph
DWFSs	Data Workflow Systems
FCFS	First-come, first-served FGCS
FIFO	First-In First-Out
GAE	Google App Engine
GPCG	Computational Genomics Queuing Network
GUI	Graphical User Interface
HDD	Hard Disk Drive
HPC	High Performance Computing
IaaS	Infrastructure as a Service
MIPs	Million Instruction Per second
NFS	Network File System
NGS	Next-Generation Sequencing
OS	Operating System
PaaS	Platform as a Service
QN	Queuing Network
QoS	Quality of Service
SaaS	Software as a Service
SLA	Service Level Agreement
VM	Virtual Machine
WMS	Workflow Management System

1 Introduction

Contents

Introduction

In the last decade, Cloud computing has addressed the challenges posed by Big Data analytics due to its scalable, reliable and cost-effective nature, where it can be adopted without organisations having to own their local computing infrastructure [88]. A lot of research is currently taking place in the cloud, as the scientific workflow provides a possibility for dealing with the growing scale and computational complexity of Big Data problems in many fields such as Astronomy, Social Science, Bioinformatics, etc.

Scientific workflows are typically composed of several different tasks with complex execution dependencies among them. This notion, which allows scientific applications to be disciplined as directed graphs of tasks that are executed to analyse large input data sets and generate large amounts of data as output when they are implemented in extremely dynamic environments [154]. Data-intensive workflows are one type workflow that has been widely recognised as a significant technology for large-scale data analysis and can provide users with an easy way to specify complex data-analysis tasks easily within the distributed resources in the cloud. However, the assumption that all tasks in a workflow can run on all resources is not always true because of the resource constraints, for instance, resource capacities for workloads or the impact of authorization of the resources [70].

A key issue is that runtime pushes the users to specify an appropriate configuration and number of IaaS cloud resources that need to fulfill running a given workflow. This action becomes increasingly complex due to the relation between the cost of a given configuration of resources and its potential influence on the performance of a given target workflow under a certain demand levels. This can be hard to determine without actually deploying and running the workflow in the cloud in a particular configuration [44, 84]. But, using the actual deployment is costly and challenging; the process of tuning the deployment to ensure sufficient and efficient use of resources can be costly, complex and highly demanding process, especially for data-intensive workflows. On the other hand, if such a relation is not determined or defined adequately in advance by the owner of the workflow, there is a risk of miss-provisioning of resources that supports the user, where resources used might be more or less than what is needed

[15]. Both cases may lead to unsatisfactory results, such as over-provisioning hight costs, under-provisioning caused Service Level Agreement (SLA) violation and poor Quality of Service (QoS).

Performance evaluation can be used to answer performance related issues [67] for workflows during workflow design, capacity planning at deployment time or during system operation. Therefore, the performance prediction of complex data-intensive execution time is a solution to be adopted for the above challenge. Due to the analysis of such applications, performance evaluation cannot be accomplished using this heavy-weight technique because of the complexity of workflow structure, diversity of big data and dependencies. This research applies a new technique that is repeatable and controllable methodology in performance evaluation of complex applications before their actual deployment using simulation tools. As a consequence, these simulation tools are a more viable solution that provides an appropriate evaluation of the performance of workflows, where the workflow system is simulated to the cloud. Cloud simulation can offer some benefits to workflow, including evaluating and optimising different algorithms and techniques related to workflow execution and resource allocation to enable research. A simulation framework is typically required for early testing of results before the real deployment is performed. In general, simulators support real execution by two main advantages [121]: 1) it provides a possibility to investigate a larger number of parameter configurations, machine sizes, or scenarios that could be difficult to achieve in practice; 2) can help to save time and cost in the real execution.

However, a longer runtime of the workflow is bound to substantially increase the cost. In addition, a long run with a large number of resources may significantly increase the energy consumption. Therefore, the key metrics for calculating both the cost and energy are the runtime and the resource utilisation. The cost metric has not been considered directly in this work because the commercial cloud providers typically charge the users with a daily-based pricing model and the running of the data-intensive workflow may stretch over a period of days. Therefore, the experiment for estimating a cost is not calculated based on the real amount of resources used, but according to the initial cost of the setup at day one, also taking into account the cost of existing shared storage which cannot be distinguished because it is all shared within the subscriptions.

Moreover, the energy metric is not considered because it is a non-trivial metric to compute in this scenario, depending on the actual running conditions of the services such as cooling and what else is running on that machine, which is clearly outside the individual workflow control.

In this thesis, the focus is on addressing the problem of performance prediction of complex data-intensive workflows through using simulation tools. As an example of Big Data workflows, we will consider complex Next Generation Sequencing (NGS) pipeline workflow as a case study, where the case processes a terabyte of data that may require thousands of CPU hours. The simulators will need some of the adaptation steps to track this type of workflow to be simulated.

1.1 Thesis statement

With the proliferation of using cloud infrastructure as a service (IaaS) for data-intensive workflow executions and answering resource sizing questions, execution time questions for deployment to the cloud is increasingly essential. With cloud simulation, there is an opportunity for interpolation and extrapolation of system behaviour. Hence, we have an aspiration to adapt an existing simulation toolkit and to use it to model the behaviour of complex data-intensive workflows, as this is important for processing and lowers costs required in tuning their configuration in the cloud. Accordingly, in this spectrum an adapted framework and its prediction model can answer questions related to overall prediction and performance of data-intensive workflow applications; it may further answer performance related questions such as “what is the best number of Cloud VMs to process a cohort of patients effectively?” Or “how long does it take to process the cohort on a given set of resources?”. As an example, Figure 1.1 shows two unknown values. The first is the red bar (24-Sample), which represents the framework that can be used to predict the execution time for much larger input data according to a specific number of the resources. The second is the green bar (24-Sample), which represents the framework that can be used to predict the number of resources to determine the defined execution time.

The fundamental objectives of this research are to accomplish the following:

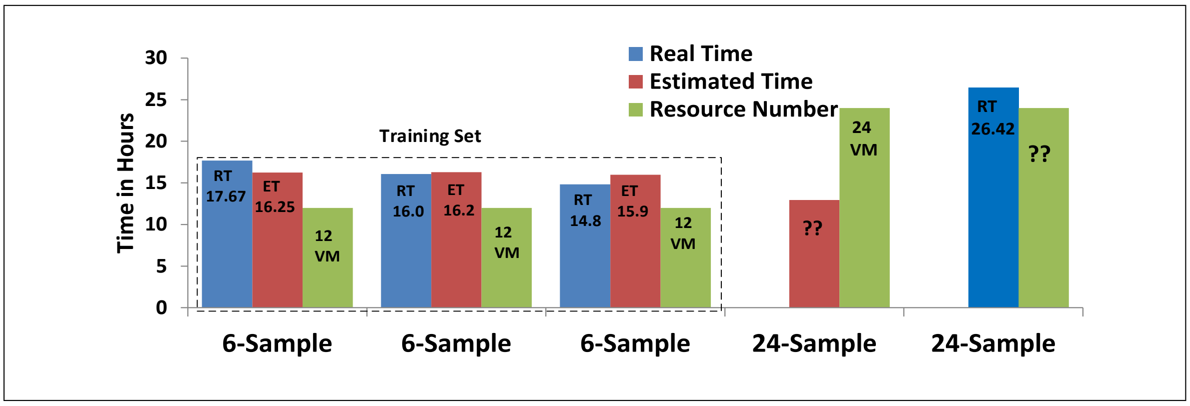


Figure 1.1: An example prediction graph.

(1)- **Performance prediction framework for the complex scientific workflows**

This addresses if we can adapt and develop framework for performance prediction of complex data-intensive workflows. Within an adapted framework, this investigates if it will be possible to answer different questions about some form of performance before the deployment of workflows into the cloud?

(2)- **Methodology for coordinating a framework to function properly**

This introduces a methodology for predicting the runtime and output data size that is equipped with realistic data from archived provenance files of the workflow's execution. These are required for constructing a training set and a fairly accurate prediction of runtime for input and cluster sizes that are much larger than ones used in the training of the prediction model.

(3)- **Enhancement performance prediction of the framework**

Two main perspectives guide the prediction improvement: First, how can we develop the framework by adding a new component that improves the prediction. Second, how can we explore the deployment of the pipeline workflow at an input that is scalable with different scenarios through runs on the actual cloud. This action supports a proposed methodology that presents the runtime prediction of the complex pipeline workflow by considering different cases of its implementation in the cloud, such as executing failure and libraries deployment time that is then could be represented in the framework.

(4)- **Complex scientific workflow modelling**

Develop a way to model the scientific workflow from one system to another enactment system to facilitate the task of the simulator. Modelling should provide a comprehensive description of the workflow, its services, required dependencies and relationships between these components. A workflow description should enable the simulator to simulate its behavior to find an optimal deployment in a range of clouds.

(5)- **Extracting the necessary information**

Analyze the provenance data and attempt to derive optimum benefit from the construction of the prediction model that will assist in building and adapting the framework.

1.2 Contributions

The original contribution of this work is a framework that adapts and develops runtime performance prediction and forecasting of the required resources for complex data-intensive workflows, where reflects its benefit on the deployment. Specifically:

(1)− A new technique to measure the performance of complex scientific workflows, based on simulation tools which coordinate with the prediction model. The technique allows the owner of the workflow to test large input data on the cloud without restriction of the configuration, where it forecasts a better deployment.

(2)− Another contribution of the research is migrating complex workflows between cloud platforms: the opportunity is to switch running a complex workflow from one cloud to another and for a better price and customer service. However, the critical step is determining how long the workflow will run and the cost of execution, which is particularly relevant for data-intensive workflows that typically takes many hours or days. Furthermore, another critical step is to choose the number and types of instances to be provisioned in the new cloud environment and then use them to deploy a complex workflow. Thus, the decision to treat the above critical steps can be made by the framework.

(3)− The main contribution within developing such tools is simulating an I/O contention problem with read or write operations. This improvement is conducted to further improve the accuracy of prediction by modelling an I/O contention. A significant part of the work is in WorkflowSim, which models the complex workflow and makes relevant changes that can be achieved in CloudSim.

(4)− A way to determine analytically the optimal number of engines is found; the complexity of the hierarchical workflow engines as shown in the pipeline structure is compounded with the fact that different stages in the pipeline exhibit different degrees of data parallelism that would be difficult to determine.

(5)− Because virtual resources in IaaS clouds are diverse, the owner of the workflow methodically evaluates the performance of their workflows using our framework after it is supported by information of the provenance from different cloud providers. This would allow users to clearly identify the resource configuration and providers that would offer better and cheaper services for their workflows.

(6)− Using the real deployment to tune an efficient use of the resources has a risk in term of security. In particular in Grid or Cloud environments, the implementation of security policies where its own organisation's computational resources reside [69]. It is, therefore, necessary to follow security policies when one wants to calculate the performance of the executing workflows. Our framework provides a private testbed to find the performance of the execution of the workflow.

1.3 Thesis Structure

Chapter 1 describes the motivation behind the work of the thesis and its aim; it highlights the research problem and major contributions of the research.

Chapter 2 presents background material required for the methods of the thesis and a summary of work closely related to the original research.

Chapter 3 presents a summary to outlying the selected simulators and reasons for their selection. The adaptation of these simulators to employ them for building a framework to simulate a complex data-intensive workflow prediction and the proposed methodology in a generic way are demonstrated in this chapter.

Chapter 4 outlines the runtime estimation methodology by adopting a complex genomics data processing pipeline workflow as a case study. A method of modelling an e-Science Central workflow into another system is explained. The steps to adapt a framework that allows us to predict the runtime when the user wants to scale up input samples is also described. Evaluation of the impact of running the simulation in term of execution time of the pipeline is conducted.

Chapter 5 presents extraction of a new dataset to provide an enhancement of the framework for runtime and more factors such as resource numbers required to execute the NGS pipeline. The pipeline simulation particularly used over the framework, is addressed and how it benefits the pipeline workflow for many fields mentioned. The implementation of the experiments are shown by how these enhancements work.

Chapter 6 presents the extension of the framework environment to model shared storage and simulating contention in I/O operations as equally to the cloud. It also shows how the extension improves the estimation of runtime performance of the pipeline. Additionally, the required time of the deployment library, as reality observed is demonstrated.

Chapter 7 summarises and highlights the main conclusions derived from this work; it proposes a number of directions for future work in this area.

2

Literature review

Contents

2.1 Summary

This chapter introduces the background knowledge and the related work that motivated the work proposed and presented in this thesis. Section 2.2 is inaugural to provide a brief preliminary on the Cloud computing. Section 2.3 starts describing Big-Data application in the Cloud computing and some of background information regarding the scientific workflow and its challenges. Section 2.4 illustrates performance prediction of cloud applications, which is the main focus of this thesis and some related works based on time and data volume are presented in this section. Furthermore, a cloud simulation will be observed in Section 2.5 in the case of describing the requirements of our research, also the simulation-based works are related to the research scope stated in this section. Meanwhile, gaps in the research topic are highlighted, where we describe how this work can address these issues. Finally, the chapter is summarised in Section 2.6.

2.2 Cloud Computing

Over the past decade, Cloud computing has been a new emergence in the field of information technology. This paradigm provides a novel operating model for both commercial and scientific users to access and deploy their applications at anytime from anywhere in the world at reasonable prices depending on their Quality of Service (QoS) specification in a pay-per-use basis. The cloud has been defined by NIST [114] as *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

In general, Cloud computing can be divided into two main parts [150]: hardware and services. The hardware (Datacenter) is a layer of resources that provides a service on demand basis. The services layer can have three popular computing models [115]: 1) **Infrastructure as a service (IaaS)**- where the cloud resources are offered in the form of independent raw virtual machines. Users take responsibility for and have the flexibility to install operating systems and software, such as using Amazon EC2

[77] and Microsoft Azure [5]. They also require 2) **Platform as a Service (PaaS)**- in this layer, where the cloud provider offers a computing platform which consists of an operating system, programming language execution environments, web servers and databases. This facility provides an environment for developers to create, host and deploy applications, saving developers from managing the complexities of the infrastructure (set up and configuration). Example of PaaS include Google App Engine (GAE) [78] and Salesforce.com [79]. Afterwards 3) **Software as a Service (SaaS)**- in this layer, the cloud provider install and operate application software on the cloud and end users can access the software from cloud clients. Examples of SaaS include Microsoft office 365 and Dropbox.

Hence, Cloud computing has brought a significant opportunity for data-intensive workflows in various fields [55, 75, 112, 116], but it has introduced many challenges. Exploiting the benefits of cloud infrastructure in the deployment of data-intensive workflows requires providing a functional infrastructure that meets Big Data application requirements, which can be a challenge. For example, efficient storage and scalable parallel programming are still challenging for scientific efforts [98]. A further challenge is the heterogeneous nature of the cloud resources and a dynamic, changing performance of the Cloud computing infrastructure [128]. Another challenge is the resource provisioning relative to the monetary cost and performance optimisation of applications [155].

2.3 Big-Data application in the Cloud

Big Data applications are applied to many scientific disciplines such as astronomy, genomics, atmospheric research and other areas. However, scientific workflows have become more popular for domain scientists for formalising and structuring a complex scientific process in the cloud. Therefore, an increasing volume of Big Data applications have motivated researchers to use the power of Cloud computing to solve their analyses.

Evidently, Cloud computing is a solution to Big Data problems [37, 90] because:- 1) it provides suitable storage for massive data storage issues, 2) it offers a cost-effective solution to the problem based on a pay-as-you-go system, 3) it gives a scalable solution for computing hardware requirements and 4) it facilitates the resources parallelisation required to execute data-intensive workflows. Given these reasons referenced allow us to solve Big Data problem with Cloud computing. The following subsection introduces the workflow.

2.3.1 Scientific Workflow

A workflow is an automation process for data tasks or sequencing activities logically which works on a set of predefined rules [101]. The workflow can be classified into business workflows and scientific workflows. The business workflow describes a work process in the business environment as sequential steps, and it comprises of procedures, people and tools involved in each step of a business process [74]. The benefit of using business workflow is to provide adequate and a reliable operational business process [16].

Scientific workflows have been implemented widely to allow scientists and engineers to implement more and more complex applications for accessing and processing large data repository and run a scientific experiment on the Grid or Cloud [133]. However, workflows can be executed in a different environment, where they behave somewhat differently when the same software (applications) runs in a different environment. Therefore, there are many efforts that have been devoted towards the development of scientific workflow tools that discover their behaviour, and information about the current state of the workflow running. The work in this thesis applies a simulation tool to address the challenge for complex scientific workflow' behaviour.

The workflow consists of a set of connected tasks/jobs that represent data flows and/or control dependencies. Each task/job represents a single step of work that is composed of one logical step in the overall process. The composition of a scientific workflow allows a researcher to represent the required steps and dependencies of the desired overall analysis. There are a number of mechanisms to specify data flows or dependencies. In one case, workflows are designed directly by representing both the computational steps and the data flows through them. Examples of this mechanism are shown in Figure 2.1 within a DAX structure (will be explained afterwards). As examples, MONTAGE, EPIGENOMICS, SIPHT, LIGO and CYBERSHAKE workflows. In another case,

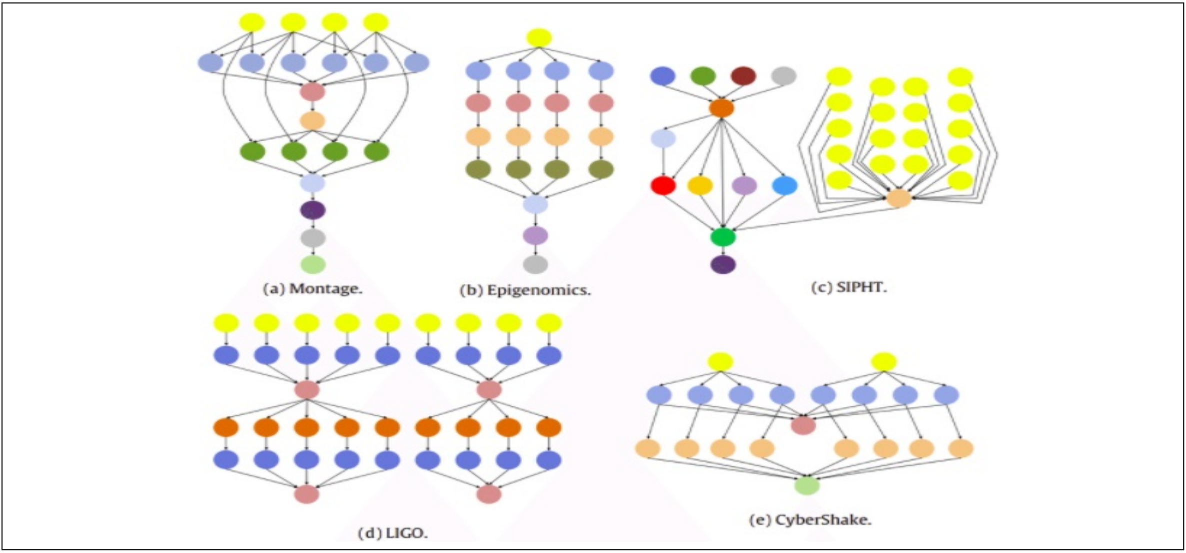


Figure 2.1: The structure of five pragmatic scientific workflows from Pegasus [19].

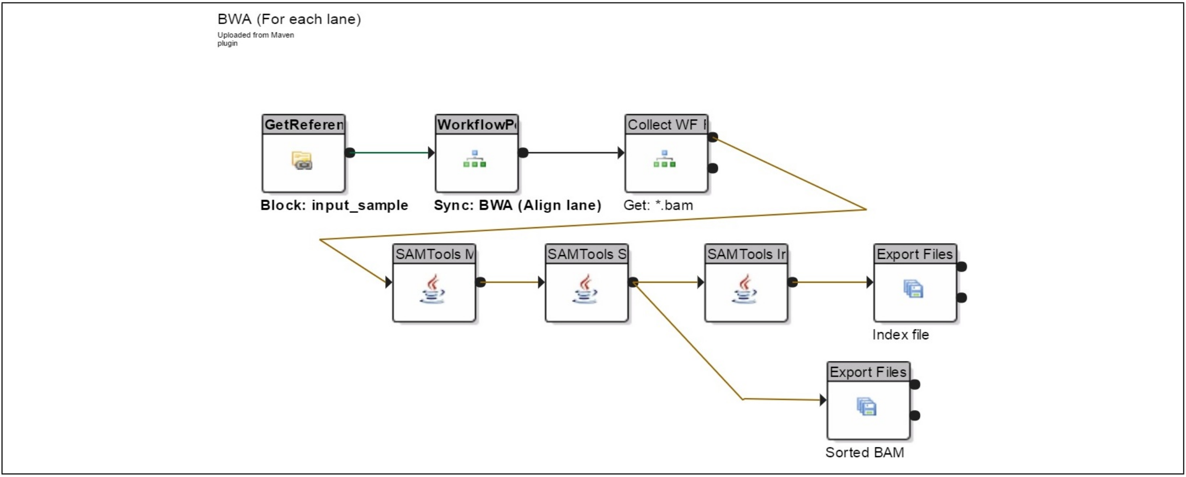


Figure 2.2: The structure of the e-scientific workflow from NGS pipeline.

the composition of the workflow contains two levels, where the first one works as an abstract level which describes the high-level workflow and second level consist of instances of the abstract level with actual data [73]. As example of this type is presented in Figure 2.2 within a (BWA) workflow that uses the sequence alignment of the reads [97].

The most common method in which scientific workflows can be presented is a Directed Acrylic Graphs (DAG) or Directed Cyclic Graphs (DCG) [48]. Indeed, there are two entities that describe scientific workflow, which are actions and tasks. On the one hand, an action is a logical step needed to achieve a part of the workflow [24]. On the other hand, a task is an instance of an activity [40]. Therefore, the task would be taken as a representation of the execution of an activity. Although most systems use an XML based representation in addition to using graphs, such as Pegasus having different representation for its abstract workflow definition, the DAX (DAG in XML), which is the input to the mapping process of the Pegasus. Another example is e-Science Central [72], where the abstract representation of the workflow is perhaps as a linked set of individual software components (blocks) that run in sequence using items of data.

2.3.2 Workflow Pipeline

In this subsection, we describe a workflow-based software architecture, where it enables end-to-end, high-level workflow processing in a cloud environment consisting of many heterogeneous resources connected to each other along with input and output data products [137, 153]. These components may include local software, scripts, services, libraries, etc.

First, the term "Workflow" is used to describe a set of high-level specifications for the collection of computational processes and dependencies that are required in order to accomplish a specific goal. Second, "pipeline" is used as a term to refer to a series of processes, usually linear, which process or transform data. In general, the processes are considered to be running concurrently. The graphical representation of the pipeline data flow does not normally branch or loop. A first step takes raw data as input does something to it, its results are then sent to second process, where this then continues as needed. Eventually, the workflow ends with the final results being produced by the last process in the pipeline.

Thus, mixing both terms, a term as "workflow pipeline" can be used to refer to an abstract architecture of the process as a series of stages from the generation of a goal description, through execution, to the completed point, where the results are collated, then staged out to the user. According, the workflow pipeline is defined as a computation and data analysis processes of the pipelines that should be accomplished by sequencing and concurrent tasks that handle streams of data [99].

In principle, data workflow systems (DWFSs) efficiently support data-intensive problems within parallelism over big data sets by providing scalable, distributed deployment and automation execution of the workflow over a cloud infrastructure. These characteristics make workflows an essential choice for implementing Big Data processing workflow pipelines. For example, works in the literature, such as Taverna [120, 143], Taverna2-Galaxy [144], Graphical pipeline for Computational Genomics (GPCG) [54], and Google Genomic Cloud, Pegasus [51, 60, 61], demonstrate such pipelines. These works demonstrate how the pipeline experiments are executed on computing platforms, where these processes are required to be mapped to a pool of executable software components. This execution stage of the pipeline is known as "concrete". The next section describes this process in more detail by presenting the workflow pipeline life-cycle.

2.3.3 Workflow Pipeline life-cycle

There are many definitions of the workflow life-cycle in the literature; all have proposed their owns phases of the life-cycle [48, 64, 111]. Three phases of the life-cycle have been suggested by Katharina *et al* [64]. Also, four phases were suggested by Bertram *et al* [111], whereas additional phase has been included to define stage-in data within computing resources before the execution phase starts. Of course, capturing provenance data is important for the workflow execution, which provides information for workflow reproducibility. Therefore a new phase is suggested by Deelman *et al.* [48]. In this subsection, the focus is on the three main phases of the workflow and pipeline lifecycle summarised in: *Specification, Realisation and Execution* (See Figure 2.3). In the following subsection, provenance data are discussed as one more phase.

Specification: (Description of the goal), This stage of the pipeline covers the programs involved in defining a workflow, the language used, how the workflow can be presented and ways that the user can specify the requirements of enactment of the workflow, where they eventually wish to have an abstract workflow in this stage. An abstract workflow is a description of the computation process in terms, relevant files and logical transformations; the dependencies between the workflow components can be specified in the abstract [47].

Realisation: (Abstract and concrete workflow), This stage can accept the abstract of the workflow, which is generated in the specification stage. The aim of this stage is validating a workflow and mapping its elements to the specific resources (scheduling

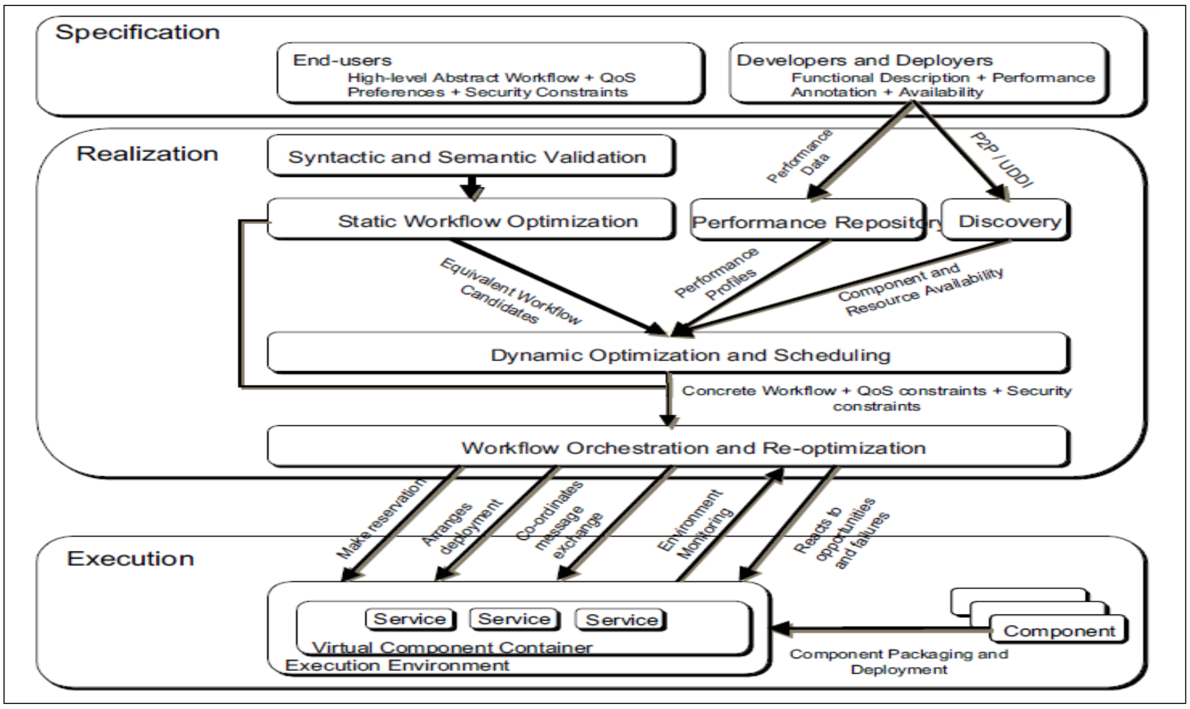

Specification
End-users
High-level Abstract Workflow + QoS Preferences + Security Constraints
Developers and Deployers
Functional Description + Performance Annotation + Availability
Realization
Syntactic and Semantic Validation
Performance Data
P2P / UDDI
Static Workflow Optimization
Performance Repository
Discovery
Equivalent Workflow Candidates
Performance Profiles
Component and Resource Availability
Dynamic Optimization and Scheduling
Concrete Workflow + QoS constraints + Security constraints
Workflow Orchestration and Re-optimization
Make reservation
Arranges deployment
Co-ordinates message exchange
Environment Monitoring
Reacts to opportunities and failures
Execution
Service
Service
Service
Virtual Component Container
Execution Environment
Component
Component Packaging and Deployment

Figure 2.3: The workflow pipeline lifecycle [134].

and deployment). Furthermore, it implements preparation for execution. This stage needs more careful observation due to its function of carrying out various optimisation to the abstract workflow, which otherwise might have caused the computationally intensive and time-consuming processes.

Execution: (workflow enactment) While the above stages are complete, the workflow moves into an execution stage, which is the workflow enactment in the cloud environment. Due to the complexity of Cloud computing heterogeneous compositions, deploying different parts of the workflow in different locations is evident. Therefore, this stage of the pipeline does not involve simply placing the compiled code on a specified resource and letting it execute. Once execution of the workflow pipeline begins, execution time may be considered as a significant factor in performance. This performance needs to be managed well according to the requirements of the workflow owner.

High-performance: The workflow pipeline is both compute intensive and data intensive that uses the distributed resources. As a consequence, this poses significant challenges regarding an execution time of the workflow pipeline. The execution time is the time needed to execute the workflow pipeline. This time can be calculated from a first component starting until the last one is completed [113]. It includes the time to deploy workflows and their libraries, which fully consume time on the cloud, excluding idle time for components to complete and transition data time between different parts. Therefore, to achieve high performance could be accomplished by mapping the complex workflow pipeline to the complex resources or proposing an augmented framework which contains the simulation annotation that corresponds to the behaviour of the workflow. For example, GridFlow [36, 118] is a simulation-based tool which uses a performance-driven strategy to determine the best resource for scheduling a task. In this thesis, the focus mainly is on the complex data-intensive workflows and describes how it can benefit from developing a new framework to simulate its behaviour and measure its performance.

2.3.4 Provenance

Provenance in the workflow is an approach that captures and records the history of the data object. This includes a graph structure for processes, where the time stamp, program version number, component or service version number, execution host, library versions, user data, intermediate data products that link back to the source data used for initialisation and potentially other data are recorded. For the scientific workflow, data provenance gives users the capacity to reproduce and verify results. The workflow can also be optimised. A single experiment of pipeline workflows may require weeks to run, even in the cloud. Thus, there is a need for scientists to monitor the experiment during its execution. Runtime provenance analysis allows for scientists to monitor workflow execution, allowing them to take actions before the end of the workflow execution.

Data provenance is also required for transformed workflow execution. However, the inputs of a workflow are outputs of the components of the executed workflow. Therefore, by tracking the provenance of the executed workflow, outputs can be immediately reused. Nevertheless, different ways to record data provenance are often needed in a workflow system. Some use internal structures to manage provenance information, while others rely on external services which could be generic. As one example, Triana has its own internal format to record provenance information and for interacting with external services [38]. Another is Karma [130], where the provenance system represents large independent workflows. That system can gather data from various workflow systems, providing a search-able database of data provenance that have extensive capabilities for formulating data provenance queries. Provenance has, therefore, introduced significant functionality for Big Data projects [63]. Dealing with provenance data is then often required when an estimation of execution time is needed for workflows [146]. To satisfy the prediction requirements of the developed framework, a statistical process, to be explained in Chapter 3, for building a training set for predicting execution time based on historical data from provenance files is required.

2.3.5 Requirements and Challenges

Workflows have become a tool capable of representing complex analyses of intensive data. They also provide a useful representation used to solve large-scale computation. These representations bring many challenges for Big Data problems, ranging from the design of the processing system in relation to scalability and flexibility to data analysis that has a higher level. As stated, a workflow pipeline consists of a composition of configurable library packages and tools that are implemented for analysis algorithms and many stage-in/stage-out operations managed during execution time. In this subsection, we briefly describe the major requirements that should be available in the pipeline processing system. Likewise, we discuss some of the most notable challenges, including facing the development and use of the workflow pipeline for Big Data analytics in the cloud environment.

- A first requirement is a Scalability: [27]

As a pipeline consuming and producing large amounts of distributed data with a cloud-based platform, Big Data processing must be able to scale simultaneously in the use of input data. For example, in Bioinformatics [124], the data analysis has been growing exponentially from small data as an input about 15GB and to output results about 1TB over the execution time.

- A second requirement is a Flexibility and Extensibility: [29]

The system must have the ability to track developments in technology by making a rapid adaptation of an existing pipeline. One key is re-configuring tools; this means exploring different options by changing their configuration parameters so that the new tools are added rapidly, according to required versions that track their evolution. If a new version of the algorithms has been released, old sequences require re-processing, as the set of variants are called by the new version even if there are only slight difference compared to the original. Actually, even slightly different configurations for applied tools within a single pipeline will generally produce qualitatively different results.

- A third requirement is a Traceability:[29]

This requirement relates to the changeability in variant lists that are produced over different times; an explanation is required through these records of variants, in terms of different production for the pipelines. Consequently, this will bring a requirement of producing more details about pipeline execution, where recording and storing the provenance of every set of variants, in a method that makes it responsive and easy to explain the difference observed through the output datasets, is applied.

- A fourth requirement is a Reproducibility:[29]

The most important requirement for pipeline processing is support for reproducibility. However, the process must have the ability to be repeatable at different times; this will give the possibility to compare different results obtained at different times. Therefore, it is possible to use different versions of the tools.

The tackling of Big Data pipeline challenges can be managed by advanced technology and Cloud computing. There are, however, challenges. Therefore, the following elaborate on the challenges and issues that are associated with pipeline management in distributed heterogeneous environments.

- A first challenge is pipeline performance variability:

Workflow management systems for extreme-scale systems, including speed up, have been optimised for supporting single application performance. Unfortunately, these capabilities are not available on complex workflows [92]. Because these workflows differ from single applications in terms of execution in one homogeneous environment in a number of significant ways; for one, they are a combination of multiple different programming/execution models and heterogeneous execution platforms. One key to the challenge is understanding how application performance variability may arise due to the number of workflow and system components, in addition to their heterogeneity and strong interdependence. Moreover, hardware and software reaction ability and user-created events at runtime will add another level of complexity.

- A second challenge is platform heterogeneity:

The heterogeneous platform makes the challenge of accessing clouds of various vendors and provision virtual machines uniform. Since there exists heterogeneity in the cloud platform, it will obstruct workflow pipeline management at different levels in the cloud for the following reasons [87]: 1- the connection to the cloud will be specified by various cloud providers; 2- resource provisioning, where there exists no standard for provisioning resources as different providers provision cloud resources; 3- migrating workflows between cloud platform can be inconsistent for instance types that complicate the migration of the workflow from one cloud to another.

- A third challenge is Resource Selection:

Choosing the number and types of virtual machines affects running a workflow on the cloud. Hence, optimal run time is hard to be determined, while the scheduling problem is NP-complete in general [138]. Thus, it is a challenge to choose the best-determined configuration for a given workflow.

- A fourth challenge is Increasing Data Size:

In order to do this, several thousand of computing nodes may be needed to handle very large data requirements, complexity storage and processing. The challenge is to manage data flow during workflow execution and the execution environment (e.g. coordinating the requirements and data volumes). However, there are many issues such as cost-related trade offs, low latency and high throughput when the data are distributed across many servers [89]. By considering the volume of big data, the complexities of challenges is magnified as data scales to greater levels.

- A fifth challenge is security and authorization:

Security is a significant issue with regards to big data workflows, especially for those with sensitive data. The activities in these workflows require computing resources as well as human resources. Where there is human involvement additional security concerns may be introduced. In planning or allocating resources capacities, we often assume that the task is allocated to resource and start execution once the processor becomes available. However, the security policies impose further constraints on task execution and therefore may affect application performance. In addition, authorization is an important aspect of security [70]. So that, the authorization constraints must be taken into account when resource allocation strategies would be considered for both human resources and computing resources.

- A Sixth challenge is information for cloud provider and customer:

For cloud customers, the challenge is a possibility to know how their applications will behave in a set of resources and the costs before renting resources. It involves potential in growing and shrinking the required resources pool [14]. Another challenge for the service provider is to know the application response time, which is affected by variable demand for the service [117]. Because resources are shared among multiple users, they can simultaneously request the service. The service provider may also want to scale-up the resource pool to meet Quality of Service (QoS) for the users. However, time and cost are involved in the process of scaling up or scaling down, both the rendering software and computing resources.

One of the methods that may be used to investigate the performance of Big Data applications depends on the number of requests machines is via deployment of the application. But, this method requires a real deployment of the application in a physical infrastructure, which limits the scale of the experiments in terms of cost and time for running tests. Moreover, reproducing the experiment with different configurations requires generation, submission and processing of user requests, which may be time consuming.

2.4 The performance prediction of Cloud application

Cloud application development has come a long way in recent years, the enterprises are already reaping benefits for users such as in saving costs. On the other hand, cloud users benefit from reducing execution time. The performance of the cloud service and

Big Data applications are dependent on each other, and this task is a challenge due to the diversity of Big Data applications and the complexity of cloud resources. However, the resources should be allocated dynamically depending on demand changes of applications, where over-provisioning increases the cost and under-provisioning causes Service Level Agreement (SLA) violation and Quality of Service dropping. Therefore, performance prediction of the application is a necessary step before deploying them to the infrastructure resources. For this purpose, a performance prediction of an application running on Cloud computing platforms has become significantly important [20]. Overall, performance prediction is complex and shall consider, both parallel computation process and data transformation time.

Choosing performance models and different design options should be considered along with various aspects including [20]: 1) process-level parallelism, 2) distribution of tasks on multi-core platforms, 3) application related parameters and 4) characteristics of the datasets. In the following subsection, we introduce some proposed models, for the framework, which leverages the performance prediction of cloud applications. Two impact factors are focused on, including runtime and data size for predicting performance as they are adopted in this thesis.

2.4.1 Performance Prediction Models

The clusters and grids are distributed systems that have been used for executing workflow applications [86]. Recently, the Cloud computing infrastructure is gaining popularity due to offering several choices and benefits to the users compared to traditional high-performance environments, mainly when an infinity computing resources will be available on demand for provisioning. However, using a large number of the resources may result in short execution time, but at the expense of a high monetary cost. In contrast, slightly longer execution time may be tolerated if this comes at a lower monetary cost. In order to decide how many resources are needed to execute the workflow efficiently, efficient use of the platform resources, including computation and communication (e.g., CPU and BW), might be required as well as shared storage. Therefore, we need an ability to predict the application's performance such as execution time.

Moreover, execution time of the application is affected by the number of resources used, the application's structure and its complexities; data volume and their dependencies also affect execution time. For example, tasks that have no dependencies may execute in parallel; so that by allocating a large number of resources a small workflow execution time can be obtained. In different cases, where the tasks have dependencies and are executed sequentially, execution time will not be affected even by adding additional resources.

Furthermore, the expectations of cloud application customers receive a certain level of application performance as in SLA; an application performance should be guaranteed at execution time [151]. To preserve the cloud application performance, the process typically runs in highly dynamic environments.

Therefore, determining the performance of a complex application, its massive computational needs and data richness, execution time of the application needs to be predicted. This benefits more efficient running and development of a complex application. Toward this end, researchers have worked with the topic of performance prediction modelling and analysis of complex applications [13].

In general, there are four groups of the predictive models, which are the table-driven methods, control theory, queuing theory and machine learning techniques [14].

- Table driven method: In this method, the application's behaviour is recorded in the table for different values of the workload intensity and their resources which have been allocated to it. The interpolation is used for calculating the values that are not found in table [17]. This method is inappropriate for scalability due to the consuming time for building the table, as it needs many numbers of executed applications, different states of the resource allocation and different types of workloads to fill the table.

- Control Theory: This model can be used for automatic resource allocation for multi-tier applications. Determining the relationship between the resource utilisation and the performance measurement is used to design a feedback controller [103]. Therefore, a processing model of the applications should be determined correctly.

- Queuing Theory: The performance prediction of the application can be achieved by the Queuing Network (QN). It works for modelling the relationship between workload and the performance criteria [148]. The queuing system contains an allocated server of the application and departs from one queue arrival at another queue. The specification of required parameters such as the request arrival rate and the average resources requirement can be estimated by solving some equations resulting from system evaluation [139]. Regardless, it does not need a training phase and is very sensitive to the parameters estimation, whereas the estimation of its parameters is expensive.

- Machine learning: This method can be considered as the newest proposed approach. Machine learning is a method used in different prediction aspects such as predicting future behaviour of resources, SLA violations, the application performance and the execution time of jobs. It works based on a training phase to expect the application behaviour. Therefore, building a model can predict the application behaviour depending on past observations of the application behaviour [20].

Because machine learning learns through historical data of the application; therefore, it is impossible to make an immediate, accurate prediction. Furthermore, to obtain a better performance of the prediction needs, a more significant collection of historical data for feeding is required for a training set [23]. Another problem of machine learning is its capability for error. Brynjolfsson and McAfee [23] state that diagnosing and correcting errors are difficult due to the need for going through the complexity of the algorithms and related processes.

Work in this thesis investigates the performance prediction of the application in large-scale data by using a training phase (i.e., small training set) to expect the application's behavior coupled with simulation tools. The limitations of the methods mentioned above are observed in our framework.

2.4.1.1 Runtime aware-driven performance

In fact, since time is the fundamental measurement of performance, many studies have focused on a prediction methods for runtime-based performance, such as. Liu *et al* [102] proposed an algorithm for duration estimation of the workflow, they achieved an evaluation using both real-world examples and simulations. Also, research related to performance modelling is considered, while the infrastructures have been used to run the workflow.

Glatard *et al* [62], they have used a probabilistic model to forecast workflow performance, when both jobs execution time and transfer between jobs have been considered. Spooner *et al* [131] presented an existing performance-aware grid management system (TITAN) for understanding the performance implications of scheduling grid workflows. Zhao and Jarvis [152] described a methodology for predicting the execution time of parallel applications composed of multiple interacting components. Viana *et al* in [140] proposed a cost model that can be used for scheduling a scientific workflow in clouds. The aim is to help develop adequate configuration of the cloud environment, according to restrictions imposed by the user. The configuration composes the Virtual Machines (VMs) regarding both execution time and costs selecting from different choices of instance types and number of VMs.

Silva *et al* [46] proposed a model that works based on input data of the tasks that predict the task parameters such as runtime, the disk space and the memory consumption. Firstly, the classification of the tasks is achieved by the workflow type and the task type. The correlation between each parameter and the size of input data can be calculated depending on the collected dataset. In case a parameter is not correlated with the size of input data, a clustering technique might be applied to split the dataset into smaller groups. Eventually, if there is a correlation between the parameters and input data size, it is estimated according to the ratio (parameter/input data size), otherwise, it is predicted a mean value.

Pham *et al* in [125] proposed a two-stage approach using a machine learning method to predict a task execution time of the workflow for various input data in the Cloud. However, the first stage is dedicated to predicting the value of the runtime parameters based on historical data for a task on different clouds. The second stage predicts the execution time of the task by applying all predicted runtime parameters together with pre-runtime parameters that have been collected before. The purpose of this method is to evaluate the use of different machine learning regression methods.

Although the majority of the works that are mentioned above are often short runs (a few seconds or minutes), and because of that, they only predict execution time of tasks and not a complete prediction execution time of the workflow. In contrast, our work in this thesis can achieve prediction performance by using a methodology (will be explained in Chapter 4) to estimate an execution time over a large set of workflows (pipeline).

2.4.1.2 Data sizes aware-driven performance

Another indicator is a data size that often complementary to performance prediction of data-intensive analysis applications. Specifically, this indicator has an important role in completing Big Data applications under different numbers of resources in the cloud, which serves users to expose completion time of their application as well as the providers to evaluate QoS of the cloud center [129].

There exists many Big Data workflow platforms that are designed for homogeneous clusters of cloud resources, for instance, Apache YARN, Mesos, Apache Spark. The expectation in these platforms is for workflow administrators to determine the number and configurations of cloud allocated resources. These platforms request from workflow administrators determine the amount and configuration of allocated cloud resource. Public cloud providers (Amazon, Azure) can provide branded price calculators, which allow comparison of cloud resource leasing costs. However, Big Data processing frameworks have directed divers for QoS measures. These calculators are unable in recommending or comparing configurations across workflow activities [128].

For instance, many examples have attempted to automate the configuration selection of Hadoop frameworks over heterogeneous cloud-based virtualisation resources. Herodotou and Babu [71] they use a relative-box model to estimate cost statistic fields and use analytical models to predict data flow and cost fields based on their proposing to use a What-if engine for MapReduce optimisation. Tian and Chen [136] they study components in MapReduce for incorporating a cost function for measuring the relationship between input dataset size and available resources. They produced a successful predicted scale performance showing how to determine an optimal number of virtual machines which are required for allocation. Jarvis *et al* [83] present performance prediction based on internal structures that exhibit highly variable runtime depending on a certain degree of data-dependence. The developed model is used in the context of an interactive scheduling system which provides rapid feedback to the users and can make a decision to either submit the workloads to available resources or allocate more resources to schedule their workloads.

The above works are based on virtual machine configuration (CPU Speed, RAM Size, cloud location, etc.). However, our work is different from selection VMs configuration, where it based on QoS and SLA requirements for mapping any complex workflow for Big Data programming frameworks and define resources number.

In this work, we show how to build a data prediction process that starts with an input data size feed as the input and uses our simulation framework to generate estimated time forecasts as output and output data size with all workflow pipeline components.

2.5 Cloud Computing Simulation

Recently, companies are transferring applications into the cloud at significantly greater rates. Therefore, the evaluation of performance level of the application as well as the cloud system itself becomes necessary. Actually, experimenting in a real environment is not recommended because the cloud system deployment requires using many hardware resources, network resources, storage resources, etc. In addition, an evaluation of critical scenarios and failure in a real system is difficult. Moreover, the facility for repeating experiments in a real cloud is almost impossible. Furthermore, having certain knowledge of networking fundamentals, cloud resource management, and cloud security is required when performing experiments within a real system. Performing experiments in the cloud can come at a high cost in financial and time resources.

To overcome these problems with a more viable solution, cloud simulation tools are one possibility. These tools can provide appropriate evaluation of the performance of cloud applications at early development stages. Simulation tools have spread in industry and academic fields. Simulations provide a free environment that can replicate a real cloud's behavioural environment [25]. Therefore, using simulation tools, experiments need less effort for preparing them.

There are several simulation tools which carried out research in the cloud. Fakhfakh *et al* [57] discuss more details of the most existing simulators in the literature that concerning Cloud computing. Most of all, well-known cloud simulator and its extension have been presented such as the CloudSim [34]. Here, the target is a particular use of the simulators that are most relevant for simulation-based performance prediction of the workflows.

2.5.1 Mapping a Workflow to Simulation

As mentioned in a previous section, the abstract workflow is composed of many tasks and data dependencies between them; they need to be mapped to a Big Data programming framework and cloud resources. The mapping or graph of data analysis activities for cloud resources demands a well decision to select relevant configurations from an abundance of possibilities and make decisions that take into account workflow execution time. Mapping can be performed either by the user or by the workflow system directly. Therefore, finding the best mapping can lead to a significant problem. Integrating the workflow system with resource provisioning technologies to determine the appropriate amount of resources that are required for workflow execution may hurt the performance and deteriorate the QoS if they are under-provided. Additional costs or unneeded costs occur when there is over-provisioning. Instead, users can apply the distribution of their workflow to target execution resources through low-commands. These, however, need advance information to identify the number of required resources for workflow execution.

The Pegasus workflow management system (WMS) approach can achieve bridging between the scientific domain and the execution environment, performing required mapping of the high-level workflow (i.e., abstract) to an executable workflow description of the computation [53]. In particular, the cloud simulation environment supports workflow systems' categories, which includes mapping [41]. There is also modelling of the (WMS) that is defined similarly to that one of Pegasus WMS [47]. Therefore, a mapping potentiality exists in the simulation of the cloud that provides us with an opportunity to conduct the complex workflow.

2.5.2 Simulation Performance Prediction aware of Big-Data

The execution of a Big Data application in real environment is 1) expensive due to its requirements of availability of many resources over an execution time; 2) time costly because it requires an actual deployment and execution under different heavy loads; 3) costly and difficulty in repeating the experiment because the number of testing is undefined to reach appropriate results. Therefore, a simulation is another technique used as a solution for performance prediction of applications. In the following, existing tools are highlighted in the literature, where these tools attempt to predict the performance of the application by testing through simulation tools.

The EMUSIM [33]: This simulation tool tends to estimate performance and costs for an application, focusing on supporting modelling, evaluation and validation of application. It combines Automated Emulation Framework (AEF) [32] for producing an improvement in the input parameters for simulations and CloudSim for simulating.

The mOSAIC [126]: There was a notable contribution in this area by Rak *et al*, who present a technique to evaluate trade-off between costs and performance of the cloud application through benchmarks and simulation. In [127] the authors extended this approach to consider *bag-of-tasks* scientific applications. The integrated framework applying a cloud simulation environment is able to predict the behavior of the development stage performance and cloud resource usage. Rozinat *et al* [80], 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.

COMPASS [93]: This tool is used for automating performance modelling. That is based on statistical analysis and the integration of a structure performance model (ASPEN Model) from the parallel application code. However, available parallelism and data movement should be indicated by the user to generate an accurate model. Otherwise, COMPASS uses Banerjee-Wolfe dependency analysis, which cannot detect data dependency through memory operations and generates a conservative parallelism profile. Additionally, COMPASS is not efficient with irregular applications due to the computation and memory access patterns which are data dependent.

ElasticSim [26]: This tool aims to evaluate the performance of scheduling and resource provisioning algorithms. The ElasticSim participates in development of the CloudSim to support resource runtime auto-scaling and modelling stochastic task execution time. This tool works as the promising platform for an evaluation of the practical performance of the workflow throughout GUI advantages.

CloudAnalyst [142]: This simulator provides a user facility to model scenarios while Software as a Service (SaaS) data centres and users are in a different location geographically. Therefore, its purpose is an evaluation of the performance of large scale-distributed applications on the cloud. It works based on CloudSim.

iCanCloud [119]: This simulator was developed by Nunez *et al.* The iCanCloud can predict trade-offs between cost and performance of a given set of applications executed in specific hardware. It provides a GUI for designing and running the experiments. Furthermore, it allows parallel execution of one experiment over several machines. It also supports simulation of federated cloud environments that contain inter-network resources from both public and private domains.

MPI-PERF-SIM [12]: This tool applies performance prediction of parallel programs on hierarchical clusters which is based on two main steps:- one at the installation time of the parallel application and the other at the runtime. In order to model the components accurately, 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.

GroudSim [122]: This simulator developed to simulate the execution of scientific applications in a computational grid or cloud. GroudSim focuses on IaaS service and can be extended to support additional models. Moreover, it provides several features for simulating complex scenarios.

All tools as mentioned above have limitations such as no consideration for data-intensive applications or they lack capabilities to quantify resources in advance. The proposed framework in this thesis has a novel contribution, where it targets to adders these limitations.

2.5.3 Big Data Workflows Deployment Optimisation

Cloud-based computing and storage services aid with the rapid deployment of cloud infrastructures; Big Data workflows have been increasingly shifted or are on the way to relocating to clouds. There are several efforts have been made to develop workflow optimisation in the Big Data era. One of the efforts is considered workflow engines, such as Hadoop ecosystem that runs on cloud platforms with virtual resources [3]. Other efforts are devoted towards works that relate to task scheduling or module mapping for workflows. For example [149], authors create a mathematical model for pipeline workflow and network components; they conduct an exhaustive investigation that performs an accurate prediction of the execution time of a computing module in a real network. Also, Coetzee and Jarvis [42] present a new semi-automated approach that's based on a semantically rich type system which needs little programming expertise from the user. This approach uses composition, planning, code generation and performance tuning of scalable hybrid analytics with online and offline usage of the analytic environments.

However, the performance and cost of deploying a cloud cluster are highly dependent on exploiting cloud resource abundance [94]. So that, the critical factor to optimise the deployment is to determine an optimal resource usage contributing to resource efficiency. Therefore, by optimising the workflow deployment or execution, this can lead to 1) reducing execution time and monetary cost, 2) increasing reliability, 3) improvement in QoS and 4) improvement in resources utilisation.

Working on Big Data workflows, owners must be careful when deploying them in various environments. For instance, configuration, provisioning, libraries, and shared components adding to optimisation in performance. So that, the owners face difficulty with characterised elements without help from providers or means.

Typically, the number of the resources allocated to a workflow based on the load demand and the QoS requirements can be defined by the customer. However, the selection of an appropriate value is crucial, where a good choice leads to significant performance and financial benefits for the cloud customer. The cloud operator can benefit from improved flexibility and ability to meet QoS requirements.

It is, therefore, through the optimisation, the requirements of improving the QoS and cost-efficiency of execution workflow; ideally, can be managed by using proactive tools, which predict execution time and provisions resources based on: 1) avoiding under-provision if there is always enough capacity to handle the workflow tasks, 2) reducing over-provisioning through decrease the number of resources that will not be at any given time and keep cost minimal. Performance optimisation is a significant branch of workflow optimisation research in the cloud. Simulation has brought new and exciting benefits such as flexibility and high fidelity models for the target architecture environment. Therefore, concerning is how to evaluate a performance of the workflows before deployment. This thesis addresses an issue of performance deployment through modelling and optimisation optimising of data-intensive workflow.

2.5.3.1 Simulation Performance Prediction aware of Workflows Deployment

The deployment of complex data-intensive workflows on the cloud should be offered where they are suitable for predictable performance and use cost. Currently, this is difficult because it is inadequate for technologies at the level of modelling and analytics that identifies key characteristics of the data-intensive application and their impact on performance. There is also a lack of information that addresses the system's operation and infrastructure in overall performance. On the other hand, testing data-intensive workflow on the real cloud is difficult due to their massive data processing and consumption time.

The feature of cloud deployment is scalability. However, cloud instances can be deployed only when they have required it; therefore, the payment is just for the utilised applications and data storage. Another feature is elasticity, where the cloud can be scaled depending on user demand to the IT system. One issue is assessing the costs involved due to on-demand; and another issue is the provider cannot guarantee SLA due to the scalability and availability. Therefore, the enterprises will become reluctant to move to different cloud platforms without technologies that allow users to make proactive, knowledge-driven decisions as it will enable them to have future trends and behaviours predicted.

The advantages of the approach that predicts the performance of applications without actual deployment onto the real cloud are: 1) avoiding the burden that can be faced with fully deployment of all applications components in various cloud platforms, 2) the potential of re-deploying multiple times using different deployment configuration, 3) the payment will be once for cloud resources that will be used by the application with high utilisation because it has tested before.

There are some tools that attempted to predict the performance of the application under testing when not deployed in the cloud. Examples of performance prediction tools are as follows:

CloudCmp [95]: This framework is a systematic comparator of the performance of different cloud services. This tool focused on comparing low level performance of cloud services such as CPU and Network throughput. It aids customers to select more suitable cloud provider for a given application based on the measured metrics that collected during the execution of a suite of benchmark tasks that stress different types of virtual resources provided by each target platform.

CloudProphet [96]: This tool aims to predict the performance and costs of legacy applications when executed on cloud infrastructures. The purpose of this approach is to focus on applications' that are cloud-aware by design. That means it takes into account the tracing of resource usage events (i.e., CPU, memory, disk I/O, and network) of the on-premise application, extracts the dependencies between those events, and finally replays them on the target cloud platform. The various accuracy of the tool prediction depends on how the event traces collected on-premise correctly and are replayed in the cloud that represent the runtime behaviour of the application.

CDOSIM [59]: This simulation tool aids to estimate the cost and performance properties of different cloud deployment options (CDO) for a given application. The modelling performance of different cloud resources can be observed in terms of Mega Integer Plus Instructions Per Second (MIPIPS) and the required resources that are needed for all application statements. CDOSIM has extended the cloud simulator, CloudSim, and integrated it into the CloudMIG framework that can be implemented to support VM migration.

CloudAdvisor [85]: This tool supports customers with a potential method of comparing different cloud providers in terms of their estimated price and performance for a given application workload and subject several user preferences (e.g., maximum budget, throughput expectation, and energy saving). The CloudAdvisor involves a detailed characterisation of the performance capability for each resource type (e.g., CPU, Memory, Disk I/O, and Network) offered by the target cloud platforms, which can be performed by executing resource-specific benchmark suites in each cloud. This is where measuring the estimation quality depends on how well the provided resource characterisation would match the actual resource needs of a given application.

In this thesis, a deeper understanding of the above issues, using a novel combination of estimation runtime performance of complex workflow couple with simulation method, along with a scheme that estimates by adapting the simulation tool. The research focuses on predicting exemplary application in runtime compared with actual execution time from provenance data and the case study that considers in this thesis is a complex data-intensive workflow.

2.5.4 I/O Contention in the Cloud Simulation

The performance of Big Data applications is dominated by time that is required by computation processes, read and write operations, and response time of the requested resources such as storage. However, virtualisation in data centre supports concurrent use of I/O (input/output) resources by several Virtual Machines (VMs). Big data workflows are more massive in workloads that can be allocated to many VMs. There is a concurrent sharing use of a storage device by running several VMs at the same. Therefore, I/O contention occurs when there is several VMs compete to have access to the shared storage for reading and writing operations, causing latency and bottlenecks. This issue can affect performance degradation as a result of the delay of response time [91].

In the context of a performance prediction simulator of the data-intensive application, most existing a modelling and simulation of a parallel I/O covers this limitation. Based on the survey of the latest cloud simulation tools in [57], several simulation tools are proposed in the literature. They considered various aspects such as modelling cloud environments and simulating several workloads running on them. Nevertheless, as shown in the previous works, there is no simulator that has considered I/O contentions. However, much less commonly, a tool has targeted runtime prediction for Big Data applications in which modelling of storage performance plays a fundamental role. Here, we give an overview of those that are developed in CloudSim.

Long and Zhao [107] present an extension of 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, its utility in modelling workflow-based applications. Sturm *et al* [132] 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 [66] 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* [109] 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.

Other works, less related to the simulation, include Costa *et al* [43] 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 accuracy to support the user's decisions about configuration and provisioning the storage system, while being less resource-intensive than running real applications. This work is conceptually similar to our work in detecting and fixing 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, 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, modelling the performance of I/O contention is

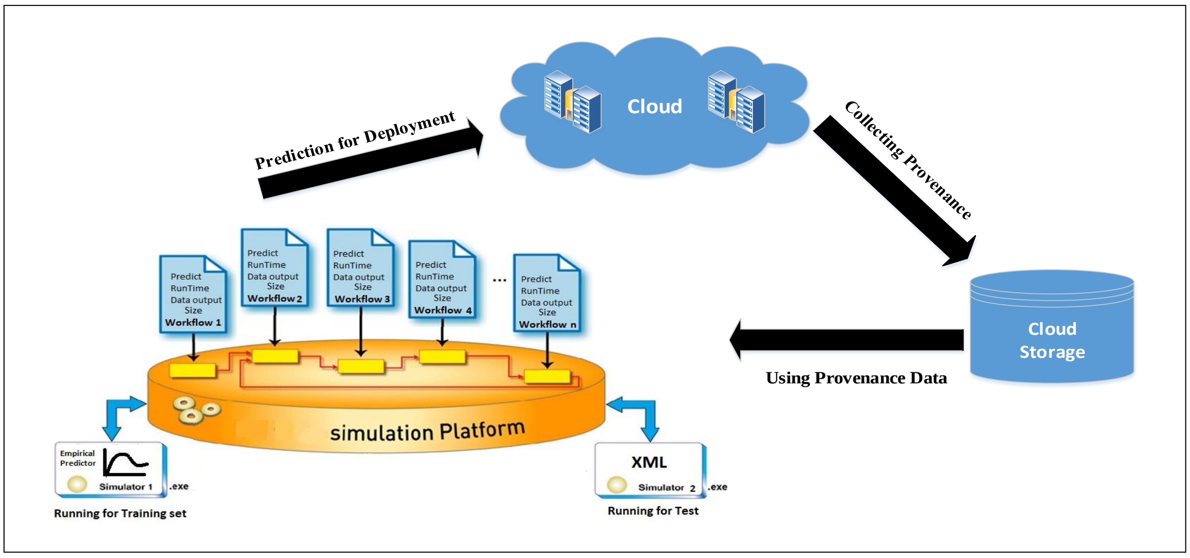

Cloud
Prediction for Deployment
Collecting Provenance
Cloud Storage
Using Provenance Data
Predict RunTime Data output Size Workflow 1
Predict RunTime Data output Size Workflow 2
Predict RunTime Data output Size Workflow 3
Predict RunTime Data output Size Workflow 4
Predict RunTime Data output Size Workflow n
simulation Platform
Empirical Predictor
Simulator 1 .exe
Running for Training set
XML
Simulator 2 .exe
Running for Test

Figure 2.4: Overview of simulation performance prediction aware of workflow deployment

a challenging due to the dealing with I/O flows issued by several VMs and because of the complexity of Big Data workflows. In this thesis, we tackle this problem by introducing a simple model for I/O contention in our framework environment, where many VMs can access shared storage and make read and write operations considering response time. This solution is described in chapter 6.

2.6 Conclusion

Deploying complex workflow activities to cloud resources regarding execution time must be considered due to a right decision that must be taken through a given configuration from an abundance of possibilities. Therefore, each complex workflow needs to predict an execution time prior to deployment onto the cloud without running on the real cloud.

The solution of this issue is to use cloud simulation tools that can measure an evaluation of the performance of cloud applications before deploying them in a real cloud environment. There is no support for a complex data-intensive workflow in previous works that have predictable performance execution time couple with simulation tools as shown in previous sections. The runtime estimation for different input and cluster sizes that are much larger than ones feeding the training set can be predicted; then information about cluster sizes becomes available before the deployment; it offers us the possibility to highlight the best one.

In this thesis, the focus is on presenting a proposed methodology that is coupled with an adaption of the simulator. This is a novel contribution where it targets to cover the gaps, see Figure 2.4, it provides a summarised overview of our framework that can predict execution time of the complex data-intensive workflows. The next chapter presents the preparation and key background information with selected simulators and relevant case study.

3

Framework for workflows prediction

Contents

Summary

This chapter describes the simulators that will be used and reasons for their selection. The preparation steps for adapting the selected simulators to employ them for building a framework for data-intensive workflows prediction are demonstrated. It shows how the proposed solution methods can be associated with the source code of these simulators and how it becomes feasible to implement complex workflows.

3.1 Introduction

Employing a cloud simulator to accurately predict runtime performance of data-intensive (DI) workflows executing within a cloud environment is an attractive area for researchers. The availability of these tools for providing an accurate prediction of the time taken for the DI workflows to execute would be beneficial for both scalability and performance optimisation (See Chapter 2). To meet the research aim, it is necessary to build a simulation model that can interpolate and extrapolate the behaviour of the DI workflow. A methodology is proposed, integrating its modelling within an existing cloud simulation toolkit for estimating the execution time of such applications, that requires a different modification.

This chapter will present a summary of what will be modified in a selected simulation toolkit: the WorkflowSim adaptation. The processes are: the method of filling in the synthetic workflow template with some parameters collected from provenance, constructing the estimation module, the automation method to generate missing capacity values for both the CPU unit and network in the data centre, and determine an efficient way to make compatible the differences between the enactment workflow model.

In the remainder of this chapter, Section 3.2 will give a detailed function of CloudSim and WorkflowSim and their components, as well it shows some of the existing examples of the Pegasus workflows. Section 3.3 describes simulation environment for the prediction that shows the methodology phases, including Subsection 3.3.1 models the input workflow into the simulator template. Subsection 3.3.2 demonstrates an *estima-*

tion module and its main role in the work of the framework. Also, Subsection 3.3.3 explains the description of provenance information and capturing the relevant information. Finally, what has been performed is discussed.

3.2 Using WorkflowSim and CloudSim

As mentioned in the previous chapter, the simulation tools are applicable solutions for studies that evaluate the performance of cloud applications. And it was also underlined that the performance prediction of the DI workflows must be achieved before they are deployed, ensuring that the cloud infrastructure effectively provide the services according to the SLA agreement with the desired QoS.

However, the right decision in choosing a proper simulation tool, considering the above matters, must be taken according to some feature availability. For instance, the simulator should be supported with more realistic for modelling the cloud environments and applications; another characteristic is the deployment of a workflow with communicating its jobs or tasks (i.e., tasks or data dependencies). In the following subsections, a brief explanation for both simulators, CloudSim and WorkflowSim, and their potential to support the stated goal is presented, highlight the reasons for choosing them.

3.2.1 Overview of CloudSim

CloudSim has become a well-known and an expansible simulator tool, where about 57% of the other simulators are an extension of the CloudSim [57], as many researchers around the world have started using it. For instance, researchers at the National Research Centre for Intelligent Computer Systems (Beijing) have used CloudSim for SLA-oriented management and optimisation of the cloud environments. Additionally, researchers at Kookmin University (Seoul, Korea) used it for their investigation on workflow scheduling in clouds. What's more, it offers the following features [34]:

- it facilitates the modelling and simulating of a large scale of the data centre.

- it facilitates the modelling and creation of more than one virtual machine (VMs) with policies for provisioning host resources to VMs.

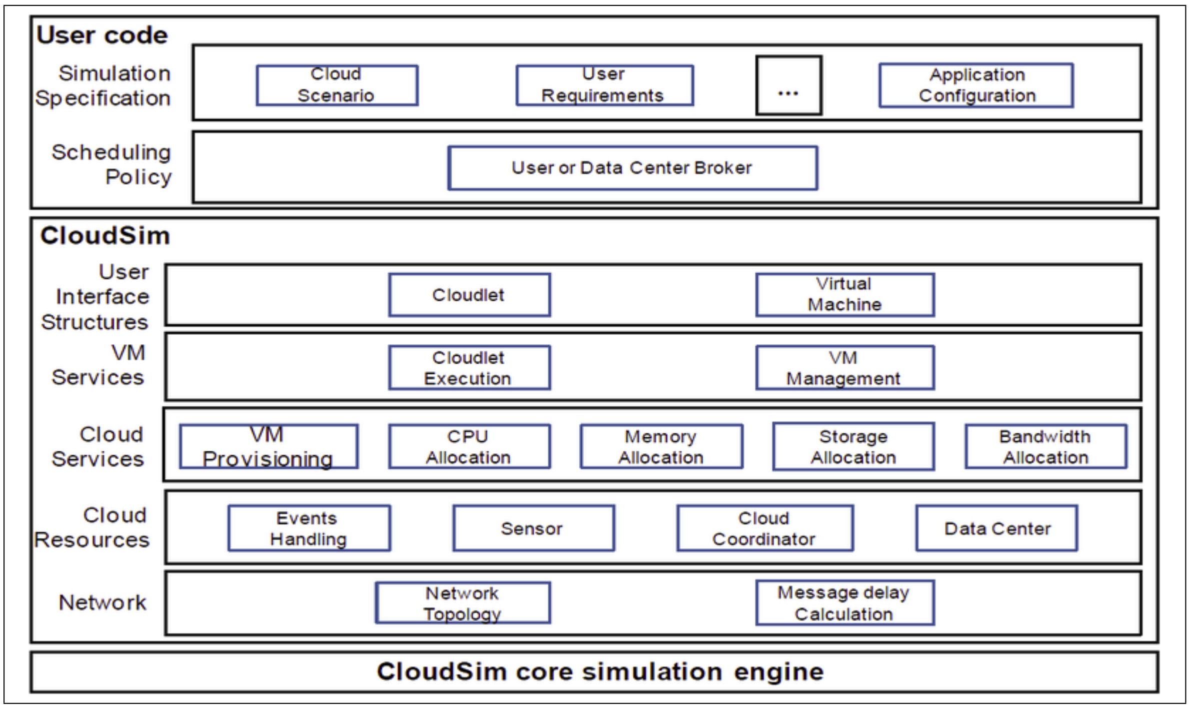


Figure 3.1: The CloudSim components and simulation engine [34].

- it facilitates the modelling and simulating of energy aware allocation heuristics provision data centre resources.

- it also facilitates the modelling and simulating the data centre topologies and message passing applications.

- it facilitates the modelling and simulating the diversity of cloud types such as federated.

- it has ability of dynamic insertion of simulation elements, stopping and resuming simulation.

- it provides the users with an ability for defining policies for allocation of the hosts and policies for allocating VM.

The components and core simulation engine of the CloudSim framework are shown in Figure 3.1. CloudSim realises a discrete event-driven simulation as a technique that is used to pass an event between the entities in the simulation framework. This technique supports the function of several cores, such as queuing and processing of events, creating entities of cloud system (services, host, data centre, broker, VMs), communicate the components in the framework, and managed the simulation clock. The CloudSim components represent an abstraction of components of the Cloud computing environment such as the virtual machine (VM), data centre, CPU, and Bandwidth [34].

Modelling the cloud computing environment

The resources of Cloud computing are interconnected together; they can scale up or down dynamically. Therefore, CloudSim abstracts cloud computing resources as a set of entities; one entity is an instance of a component, which is described by the data center, including many physical machines. These machines can use any virtualisation technique to provide multiple virtual machines (VMs). The applications can be submitted to these VMs; the number of the VMs can be changed dynamically over time. In the following, more details related to components of CloudSim [34, 35], which are shown in Figure 3.1, are discussed.

Data center : It encapsulates a set of computing hosts that can be either homogeneous or heterogeneous concerning their hardware configurations (memory, cores, capacity, and storage). Where the host in a CloudSim denotes to the physical node in a cloud, such as assigned in a pre-configured processing capability, that is expressed in millions of instruction per second- MIPs.

Cloud Coordinator : This component is an extension of the data centre; it is responsible for exporting the cloud infrastructure and platform level to the federation. It keeps monitoring the internal state of data centre elements, conducting negotiations with the cloud providers in a federation to deal with any unexpected peak in demand for resources in the local resources. It ensures that the agreed SLAs are delivered through monitoring the execution lifecycle of the application.

Virtual Machine VM : This is a modelling of the virtual machine component in CloudSim, which is managed and hosted by the host components. Each VM has related characteristics such as accessible memory, processing, storage size, and the VM's internal provisioning policy (space-shared and time-shared), based on these policies the host can simultaneously instantiate multiple VMs and allocate the cores in the VMs.

Cloudlet : It is modelling of the cloud-based application services (jobs or tasks), which are commonly deployed and executed in the data centre. Every job/task has predefined instruction length and volume of data that have been taken into account to be hosted successfully.

VM Provisioning : This models the provisioning policy of allocating VMs to the hosts. The main function of the VM provisioning is selecting an available host in the data centre, considering the VM deployment requirements such as memory, storage, and a processing element (PE). CloudSim package depends on a simple VM Provisioner policy to allocate VMs, which is selecting the first available Host that contains abundant elements and are meeting the VM deployment requirements.

Memory Provisioning : Models the provisioning policy for allocating memory to VMs. This component can model the allocating physical memory space to the rival VMs. Once a memory provisioner component determines the space that is requested for the new VM deployment, then the deployment and execution of the VM will be possible.

BW Provisioner : This models the provisioning policy of bandwidth which is required to deploy a VM on a host component. This component provides an allocation of network bandwidth for a set of deployed VMs on the data centre. To meet the application requirements, developers and researchers can extend it with their specific policies such as priority and QoS.

Storage Allocation : Storage area in the CloudSim for storing large chunks of data is modeled. SANStorage is implemented as an interface to simulate saving and retrieving data. The bandwidth is a fundamental factor to measure a speed of transmitting the data. Where extra time can add to task unit execution as a delay time, which is consumed for transferring data files through the data centre over the network speed.

Network Topology : This models the information about network behavior in CloudSim. It commonly uses the information that is generated by the BRIT topology generator.

Sensor : It is modelling a sensor component that can benefit a cloud coordinator to observe the specific parameters such as energy consumption, resource utilisation.

As we mentioned above the CloudSim is an event-driven simulation tool, which all components are dealing with the message, such that each entity communicate with another one by passing the event as a message. In particular, the CloudSim has a limitation for performing simulation and executing only individual job/task [35]. While the workflow cannot be simulated in the CloudSim. Therefore, there needs to be an exploration of the simulation tool that deals with the workflows and undertake their dependencies between the jobs/tasks.

3.2.2 Overview of WorkflowSim

One of the most widely used examples of a simulation environment for workflows is WorkflowSim, developed by Chen *et al* [41]. It is an extension of the CloudSim simulator and enables modelling and simulation of scientific workflows (data flow) with the DAG form in the cloud. The simulator follows the approach proposed by the Pegasus WMS, contains a workflow mapper, engine and scheduler, and components such as the clustering engine [47]. These components allow WorkflowSim users to evaluate and optimise different algorithms and techniques related to workflow execution and resource allocation, which if done in the real cloud, would be time consuming and costly.

WorkflowSim is based on the identification of popular workflow systems for scientific workflows and their components (e.g., composition, mapping, and execution) to support some acceptable features of workflows and optimisation techniques. It is designed to be on a top layer of CloudSim specifically at an existing task scheduling layer, including workflow Mapper, Workflow Engine, Clustering Engine, Workflow Scheduler, Failure Generator and Failure Monitor etc. This layer consists of four main components as follows:

Workflow Mapper : this supports the importing DAG files that are formatted in DAX and other metadata information such as the size of the file from a workflow generator [45].

Workflow Engine : it supports handling the state of the tasks by releasing them to the Clustering Engine and also managing to ensure that tasks do not release until their parents complete the execution successfully.

Clustering Engine : approach merges tasks into jobs intending to reduce the scheduler overhead. Usually, an atomic unit that can be seen by the execution system is a job, which includes a set of tasks and are viably executed, either sequentially or in parallel.

Workflow Scheduler : this matches jobs to the resources based on the criteria which are predefined by users. WorkflowSim depends on CloudSim to support an accurate and reliable job-level execution model, such as the time-shared model and space-shared model.

3.2.3 Why these tools

Based on the possibilities of the simulators that offered above, we decided to choose CloudSim and WorkflowSim. Typically, the CloudSim can provide a high fidelity model on the specifications of cloud hardware. This makes it a powerful architecture for modelling and simulating complex DI workflows without suffering from a lack of modelling of the main components of the cloud environments, which can be handled and would be slight at most.

As a notable simulation facility that fits the purpose of simulating the workflows on the cloud. WorkflowSim is a significant option because it supports building the framework as efficiently as possible. However, one potential problem is that WorkflowSim cannot accept a complex DI workflow which implemented in e-Science Central platform [72]. Because only it can model Pegasus workflows, five recognised scientific applications have been implemented in this simulator, which are: LIGO Inspiral analysis [6], Montage [18], CyberShake [65], Epigenomics [7], and SIPHT [4]. Therefore more progress is needed, such as a method to map e-SC workflows into what WorkflowSim can enact (will be explained in Chapter 4).

One of the benefits of WorkflowSim is that it enables a workflow to be simulated in repeatable and reproducible experiments, with no charge for testing environment [108]. Another benefit of the simulators is that they are based on Java, open source code that allows one to write classes and develop the code as required.

3.2.4 Existing Examples of Pegasus Workflow Experiments

Experiment 1

To simulate the Pegasus workflow in WorkflowSim and to be familiar with using its configuration parameters, we conducted the LIGO Inspiral Analysis workflow [22], which is an application used to investigate gravitational wave signature for data that collected by large-scale interferometers. The workflow's mission helps to detect and determine the influence of gravity changes in which gravity occurs due to the curvature of time and space. The LIGO Inspiral workflow was divided into multiple groups of interconnected tasks. Figure 3.2 shows the structure of a small LIGO Inspiral workflow. The figure presents a graph of the abstract of the workflow as tasks, with each task having a task ID and expected execution time that was collected from workflow gallery [10]. For more information about the abstract, see DAX file in [11].

The experiment here is to evaluate Inspiral execution under different scenarios. The value of MIPs was varied in each execution using three values (1000, 1500 and 2000) as well as applying variations of BW with two setting(15 and 100). By controlling the number of VMs ranged from 1 – 10 at any setting of both parameters, we can see the entire execution time of the workflow across the virtual machines (VMs). Figure 3.3 shows a graph of all scenarios as the execution varies. The optimal parameters values can be determined experimentally through the training that is beneficially for many studies, such as scheduling, resources utilisation, cost mandatory etc.

Experiment 2

One benefit achieved through using the WorkflowSim is the scheduling algorithms development. In this experiment, WorkflowSim was used as a tool to show how a simple extension of the Minimum Completion Time (MCT) scheduling algorithm produced a benefit of the execution of the workflow in reducing the Makespan and cost [106]. This work also allowed us to be in better understanding the available simulated environment in the WorkflowSim and proved to be very useful for the workflow.

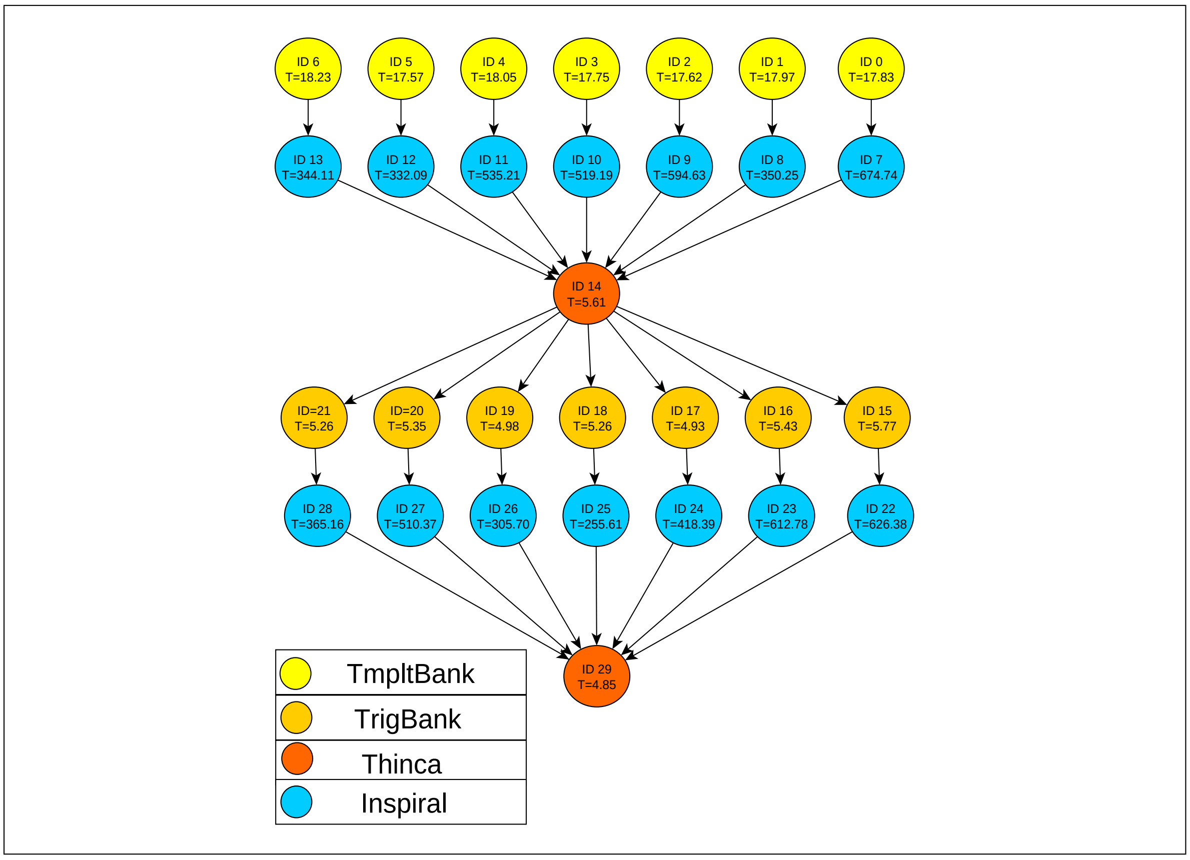


Figure 3.2: The structure of the LIGO Inspiral workflow.

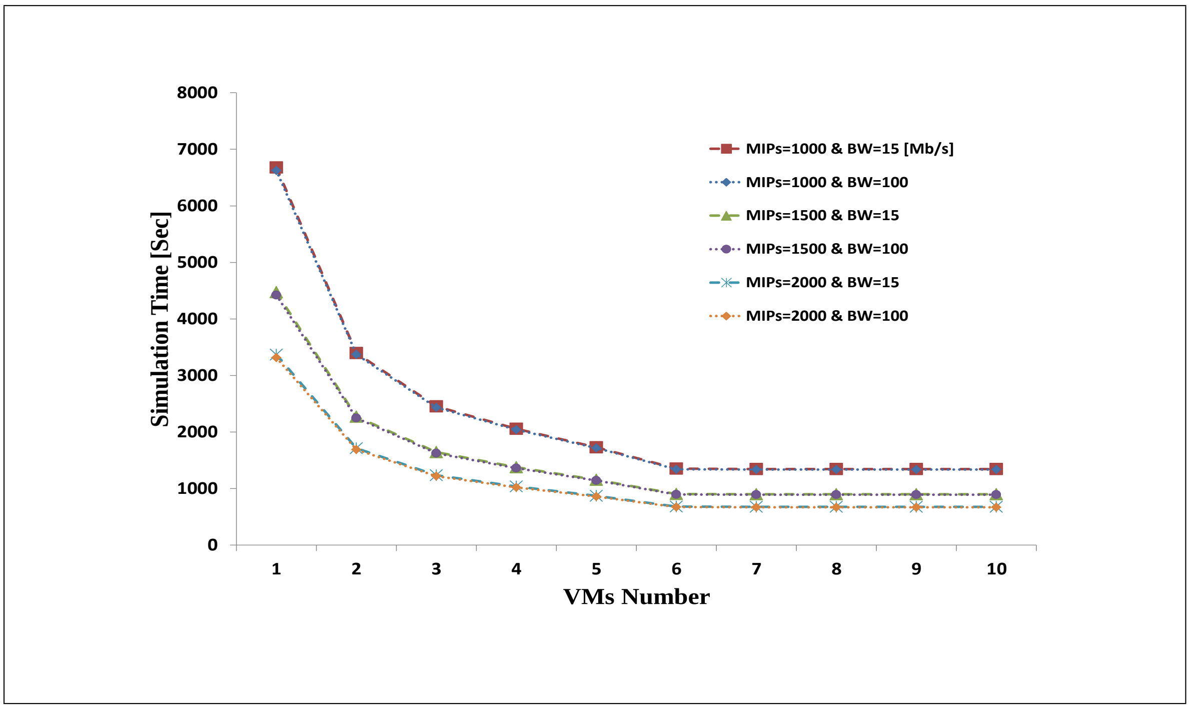


Figure 3.3: The experiment to show variations of MIPs and BW.

3.3 Simulation Environment for Prediction

As mentioned previously, CloudSim and WorkflowSim were selected for building a simulation-based performance prediction framework of complex DI workflows. In order to address the overall goal, we need to develop the framework as automation tool of its input parameters and predictive platform for the execution time.

Once a standard parameterisation of the WorkflowSim was used to run a workflow, the environmental parameters are assigned as a sequence of manual configuration steps; these parameters' values can be taken from a dedicated set of values from the previous real execution. An overview of the methodology used in preparing the selected simulators to achieve the target is presented in Figure 3.4. We illustrate the methodology phases and their mechanisms in the following subsection.

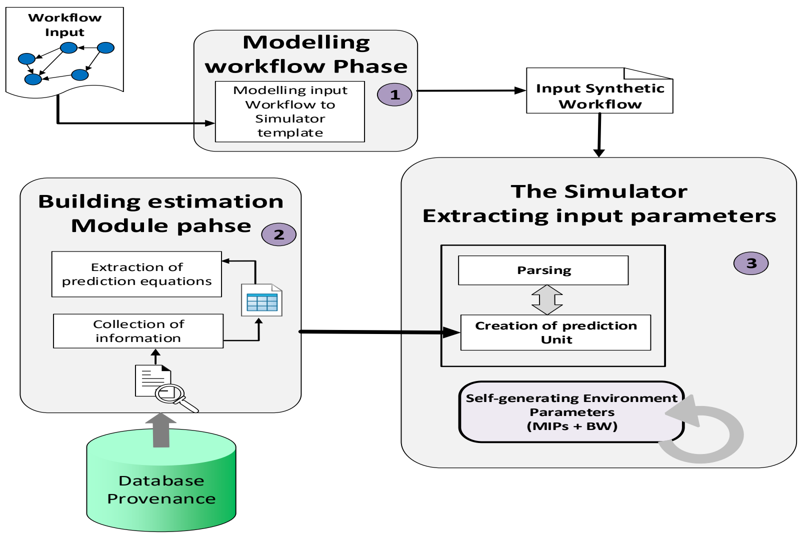

Workflow Input
Modelling workflow Phase
Modelling input Workflow to Simulator template
1
Input Synthetic Workflow
Building estimation Module pahse
2
Extraction of prediction equations
Collection of information
The Simulator Extracting input parameters
3
Parsing
Creation of prediction Unit
Self-generating Environment Parameters (MIPs + BW)
Database Provenance

Figure 3.4: The Methodology of the proposed prediction Framework

3.3.1 Modelling Workflow Phase

It is apparent from the methodology figure that the phase represents a modelling input workflow into the simulator template. However, WorkflowSim was primarily designed to simulate Pegasus workflows. Therefore, this required a way to convert an input workflow into a form acceptable to the WorkflowSim before a submission process is done. This mechanism would be achieved manually. For example, to model the NGS pipeline designed and implemented 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, but 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 sub-workflows. Each of these sub-workflows 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 considered 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, thus it is transferred to/from the shared storage provided by e-SC. However, finding a balance between effective parallel execution across some machines and using fast local data transfer is not obvious, as discussed in [27].

After the modelling phase, the generation of the synthetic workflow stage is arisen. But, the synthetic workflows include two steps: the structure that has included identification of individual tasks and their composition. Consistent with a technique that was used by Pegasus WFMS to create a synthetic workflow, the same technique to fill out the synthetic template what belongs to a new workflow needs to be followed.

Input Synthetic Workflow

Workflow traces have been collected and published in the workflow Gallery [10] through Pegasus WMS. The toolkit called Workflow Generator can produce synthetic workflows by gathering metadata (e.g., input/output data sizes, runtime of the tasks) to facilitate the evaluation of the workflow performance under diverse configurations within the simulation. From these traces, the workflow DAX file in the XML format can be filled and used as input to the WorkflowSim.

The XML file contains the representation of abstract workflow description, which comprise of all the computational tasks, the execution of the tasks' order, the required inputs of each task, the expected output of each task, and the argument with which the task should be invoked. Finally, the synthetic workflow that is generated by a workflow generator can be simulated by WorkflowSim, and many examples of the DAXes files are available in [11].

Consequently, constructing and populating the data to create a new synthetic workflow relies upon relevant data in the provenance file. However, the ultimate objective is simulating the DI workflows which have no existing information in the provenance, in particular when a large size of input sample is considered. Therefore, a method to populate the missing parameters in the synthetic template had to be found, where these parameters need to be populated by estimating them. For example, the sizes of output data and the execution of times processed for each task. This is explained in phase 3.

3.3.2 Building Estimation Module Phase

Provenance provides a very significant service to workflow applications in such services as verification of process and review of recorded data [76]. However, an abstract template that was obtained from the previous phase is not completely adequate and requires more detailed information that can be extracted from the provenance. Therefore, in this phase 2, the focus is on the usage of the provenance data for building an *estimation module* and present extracted data and how it can be utilised. This phase can be prepared by the steps described below.

i- Data Collection and Analysis

Once the workflow's process is complete in the cloud, the description of steps leading to the results is often stored away as a file. However, the decision as to what to retain the provenance data in the cloud storage is a trade-off regarding the long-term storage cost and can be considered by the users [145]. Therefore, weeks or months may pass before the scientist realises a produced result. Data provenance contains information on what the datasets generated, which is important for this work in two respects: to create the synthetic workflow and to extract the prediction equations.

As we have seen in Pegasus WFMS is used in Workflow Generator as a tool to create synthetic workflows. But based on what is know, there is a lack of the provenance information discovery for the end user, in particular the DI workflows. Therefore, a method to discover and collect the provenance data for this workflow needs to be determined. For instance, the cloud platform for data analysis, e-Science Central supports secure storage for sharing and runs workflows for analysis data [72]. It facilitates the collection of the provenance information and uses the fragment of the code written in the Prolog language and supported by the ReComp team [1] (Appendix A). The e-SC provenance files of an executed workflow can be downloaded. The Prolog program is provided with the "Invocation id" number, which refers to top-level workflow; after that, the program exports information as a CSV file.

At the time that provenance files are exported as CSV files, an analysis of particular information can be performed (i.e., execution time, input data size and output data size). This could deliver a prediction based on a set of input data. Thus, prediction provides the basis for building an *estimation module*. However, the prediction of both execution time and the size of the output data for the data-intensive workflow within all its components (e.g., sub-workflow) is necessary, where having statistics on the past execution of the workflows is needed. Therefore, a historical database is used, with the information of the workflow's execution stored and the prediction model uses this information. For example, details about execution time and the size of input/output data of the pipeline workflow are stored in the provenance storage system in e-SC Central [147]. This file contains the data which can be summarised in Table 3.1. This is used to extract the execution time and input/output data size for each component in the workflow.

ii- Extracting Prediction Equations

Typically, DI workflow is executed multiple times, with each execution having different input data size to predict its performance (e.g., its execution time with various resource numbers). The differences in the number of required resources of the workflow execution can be attributed to the size of the input data. However, the performance metrics are for exploring the relationships between the input data size of the workflow and the execution time, including accurate estimation of these requirements being cru-

Table 3.1: Provenance information

Label Name	Description
1- Invocation id	*Refer to the Workflow identification*
2- Workflow Name	*Refer to the Workflow name (BWA, PICARD, etc.)*
3- Block Name	*Refer to the name of each block in the Workflow (block_1,block_2,etc.)*
4- Block Label	*Each block has a label (GZip, PrepareLibrary, etc.)*
5- Start Time	*Beginning time to execute the Workflow*
6- End Time	*Finishing time in executing the Workflow*
7- Data Consumed	*The size of input data*
8- Data Produced	*The size of output data*

cial. Moreover, it is important to examine how variation in all input data sizes over workflow components may affect workflow execution time and thus affect performance. Therefore, a statistical analysis is needed to extract these relationships.

For the purpose of the development of our framework, a linear least squares regression method is used for extracting the prediction equations to build an estimation module and in turn to predict runtime and input/output data size [141]. This simple method has been chosen because the data elicited from the provenance information exhibits a linear dependency. The accurate indicator of a generated function can be controlled by r-squared (R^2) which have values between 0 and 1. Figure 3.5 shows the formula of the regression used.

The experimental results show that the linear regression approach, when compared with the polynomial regression in the description of the runtime and output data size of the NGS pipeline, offers the best accuracy for the prediction studied, including cases where the predicted resource usage presents linear dependencies on the input data size.

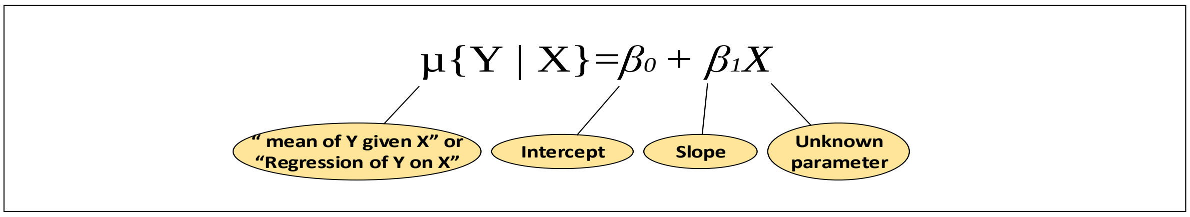

$\mu\{Y \mid X\} = \beta_0 + \beta_1 X$
" mean of Y given X" or "Regression of Y on X"
Intercept
Slope
Unknown parameter

Figure 3.5: A simple linear regression model.

1) Runtime Estimation

An accurate estimation of the execution time becomes a dominant factor in different cloud research fields. For example, to investigate an efficient scheduling [100]. Execution time of the workflow is commonly used as a metric to scale workflow performance

[56]. The scope of the work and goal, in the broader context of performance prediction that requires estimation the runtime of the DI workflow. For example, the focus is on execution time estimation in the complex structure for the e-SC workflow pipeline. The execution time for the entire workflow pipeline (i.e., workflow and sub workflow) needs to be estimated to determine a reasonable runtime that can be used to manage the deployment.

The *estimation module* is used; it is described in subsection 3.3.2 to estimate the execution time of each component of the workflow-based pipeline. The runtime estimation approach uses the data sizes that are collected from provenance information as input to the *estimation module*; the prediction is based on prediction equations which are also extracted from the provenance by the least squares linear regression. Thus, in this way, the execution time can be predicted with accuracy for each component of the workflow pipeline.

2) Output Data Size Estimation

The aim of this prediction differs from the runtime approach in that with the concern is predicting the execution time of each workflow in the whole pipeline. The concept of output data size prediction is similar to applying a similar result for runtime prediction, but with the resulting output data size instead. That means both input and output of the *estimation module* are data sizes; prediction occurs sequentially for the whole pipeline.

However, the different sizes of the input data in the sample may lead to significant differences in the output over the process. Exploration of the data is begun using the relationships between input and output data size of the workflow that represents performance metrics and are crucial to estimate it. Statistical analysis (i.e., Least Squares Linear Regression) is used to extract these relationships; the assessment of how the output data size is affected is based on variations in the input data size. Hence, this prediction is mostly performed on the initial input data only. An *estimation module* is provided with specific prediction equations of data sizes such that the ability enables prediction not only for execution time but also output data size.

Variance: The performance prediction framework is used to estimate the execution time of the data-intensive workflow; the predicted runtime usually lies in an interval around the actual execution time of the workflow pipeline. Variances are given for actual times that tell us how spread out the estimated times are, where the closer variance is to zero. When working with sample data sets, the following formula to calculate variance can be used.

$$S^2 = \frac{\sum(X_i - \overline{X})^2}{n-1} \tag{3.1}$$

where S^2 is the variance; X_i is the term in the dataset; $\overline{X}$ is the sample mean and n is the sample size.

iii- Integrating Estimation Module

The performance prediction model supplies estimated activity for the execution time and the input/output data sizes of the workflow for the WorkflowSim. Such activity has not been built in the original version. Therefore, it was necessary to devise a predictive model and integrate it into the framework to complete the research objectives. This model should provide the simulator with a training-based approach. The reason is that this can cover cases for testing workflow, which has input data much larger than the training set. Another reason is building a training set without cost, hence a re-training set if it is required, can be done by rerunning the simulator.

The *estimation module* consists of two main branches: (1) runtime estimation, which predicts the execution time of each component of the DI workflow (e.g., workflow pipeline); (2) input/output data sizes estimation, by this the output data size is predicted and will be input for the next component of the DI workflow. Both activities of prediction depend on the data size as input to the model of estimating the runtime and output data size.

Building a prediction model using a statistical method over the input dataset depends on the predicted equations described above. That involves multiple phases: (1) creating the runtime and output data sizes estimation units; (2) generating a training set and its configuration; (3) extracting and filling information in the DAX files of the synthetic workflow; (4) testing a scale-up input data that is outside the training set.

3.3.3 Extracting Input Parameters Phase

Phase 3 is an extractor that extracts unknown parameters that are required for the simulator as input. There are two unknown input parameters that are needed for the operation of the WorkflowSim and inputs used to complete missing information in the synthetic workflow file. Both types are generated during running the simulator in a coordinated manner while fulfilling its prediction potential, as explained below.

i- Extracting Simulator Parameters

The WorkflowSim as any software requires the setting of its input parameters that presented an initial step before the running. For us to be able to run this simulator, the following steps describe how to set up an execution environment and run DI workflow. An execution environment consists of a set of input parameters have to be set with value. Here, we learn how to make self-generation parameters by the simulator.

To configure modelled resources in the simulation, a set of the parameters need to be specified with proper values, depending on the users' desired intent, with the most parameters used in the WorkflowSim shown in Table 3.2. Two primary input parameters are focused on that significantly affect the execution time of the workflow, including MIPS and BW. The input parameter (MIPs) is a million instructions per second that scale the computation process time (i.e., CPU speed) [9]. The bandwidth (BW) is a bit-rate measurement that defines a transmission capacity over a network communication system [8].

However, at the beginning of a simulation the workflow, we have no idea about the parameters of the execution environment (i.e., MIPs and BW). Therefore, an empirical approach is applied, running the simulator a number of times, across a given range of both the MIPs and the BW values , to tune a setting value of the MIPs and BW (i.e., the bandwidth within the data center).

As described above, there needs to be an allocation of the capacity of the computation and transmission units with MIPs and bandwidth values. They will be derived from a simulation to construct the training set. The work relies on the perspective of the training set to predict the runtime of the workflows, which is based on fixing the functional training set to be more influential on testing sets outside of the training

Table 3.2: Input Metrics

Parameters dependence	Parameters	Description
VM parameters	**size**	*Image size (MB)*
	ram	*VM memory (MB)*
	mips	*CPU speed*
	bw	*Capacity BW in VM*
	pesNumber	*Number of CPUs*
	Vmm	*VM name*
Global parameters	**VmNum**	*Number of VMs*
	num_user	*Number of the users*
	MaxTransferRate	*The bandwidth within a data centre in MB/s*
Data centre parameters	**arch="x86"**	*System architecture*
	Os	*Operating system*
	Cost	*The cost of using processing in this resource*
	CostperStorage	*The cost of using storage in this resource*
	CosetperBw	*The cost of using BW in this resource*

boundaries.

To begin with the initial stage, we need to build the training set and configure its parameters, so that we have proposed to use an empirical approach for running WorkflowSim repeatedly to extract these parameters. The most significant function of an empirically the simulator is to predict continuous values until finding a minimum relative error between the actual and estimated execution times on the samples from the training set. Meanwhile, the runtime *estimation module* that will be summarised in next section would be cooperated to support the training set with the estimated runtime, consequently to improve the MIPs and BW values. Thus, at a minimum relative error, we can pick up the MIPs and BW values and be considering them as optimal capacity parameters for the testing sets, (will be explained in more details in the next chapter in the methodology).

ii- Extracting Synthetic Workflow Parameters

As an output from the phase 1, the template for the synthetic workflow still suffers from inadequate information. This information must be collected and placed in the context of the template to enable execution of the WorkflowSim on a new workflow. In this step, a necessary extraction method to derive the missing parameters of the synthetic workflow is planned.

In this regard, the simulator's parser and an *estimation module* have a fundamental role in addressing this problem. However, the original parser model of the simulator was designed to parse an XML file which constructed based on Pegasus system. Therefore, to make the simulator accept a new format of the workflow from a different system, we have to develop the parse to be tailored with the new workflow's format, at the same time the parser should manage a treatment of missing parameters.

The workflow template generating phase is followed by predicting the missing parameters by assigning different values (from *estimation module*) to the parameters (e.g., expected runtime, input data size, and output data size) in the template. The prediction data are assigned to these missing parameters according to the input data size and the *estimation module* is then used to make predictions. While assigning data to the parameters, the parser model parses and creates the internal template within full information for each identified node. Both of those units (i.e., the parser and the estimation units) have been working together to generate the missing parameters with taking into account the sequence. Firstly, parsing and predicting to create the output data volume overall workflow template, and secondly parsing and predicting to generate the execution time because the execution time of each node depends on the input data size as an input parameter to estimation function.

3.4 Conclusion

In this chapter, the simulators have been introduced, which will be used to build a framework to predict the performance of the data-intensive workflow with better accuracy in runtime. The key steps require an adaption of these simulators and make them adequate to accept complex workflows rather than the Pegasus workflow system. The complex workflow that will be applied as a case study is implemented in the e-SC system as well as a pipeline of the workflows it will be described in the next chapter.

We have presented approaches that can extract the framework parameters and predict execution times, input data size, and output data size. The approach also uses these parameter values for populating the synthetic workflow with data that are conveyed by WorkflowSim. The detailed steps are followed in conjunction with a coordinator methodology that clarifies the approach to enable such complex workflows can be simulated and predicted for their performance. Two experiments were conducted using WorkflowSim and provided us confidence to work with these tools.

In the next chapter, more detail about the case study and description of the processes conjunction methodology to build the framework for runtime performance prediction of the complex data-intensive workflows is given.

4

NGS PIPELINE PERFORMANCE

Contents

Summary

The simulation, together with a newly proposed runtime estimation methodology, provides an overview of the performance and benefits to predict the resource number associated with data-intensive processing. Such an approach has implications on fields that include health care. In this chapter, a complex genomics data processing Next Generation Sequencing (NGS) workflow-based pipeline is implemented as a case study because there is a lack of evidence on how well NGS pipelines behave with an increase in the number of input samples. A way of converting e-SC pipeline workflow into one that can be accepted by the WorkflowSim is explained. An integration method for the methodology phases is described with the WorkflowSim to produce a new framework that allows one to predict the runtime if the NGS pipeline's user wants to scale up input samples. How this methodology is working correctly is shown with implementing some of the experiments that have already been executed in the cloud with the same input samples.

4.1 Introduction

A big data workflow is composed of many applications that may involve large input data and produce large amounts of data as an output [49]. 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 that 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 chapter, this problem is explored by considering the pipeline workflow deployed on the Microsoft Azure public cloud [27]. 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 minimising execution time and cost.

As pointed out earlier, the selected simulators have been a significant tool for the evaluation and improvement a single workflow, although there is a lack of support for simulating a pipeline (a set of workflows). A simulation platform should be modified to simulate the execution behavior of the NGS pipeline as implemented in e-Science Central e-SC . The main contributions of this chapter 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-SC workflows. In order to achieve this, the e-SC workflow enactment model has been translated 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 is a novel approach of prediction.

The current chapter is structured as follows: In the next section 4.2, the conceptual structure of the NGS pipeline and its characteristic is discussed. Section 4.3 will describe the methodology phases, which included modelling the NGS pipeline to the synthetic workflow, building an estimation module, and WorkflowSim for extracting input parameters. Section 4.4 shows the experiments that evaluate and examine the methodology which is proposed for prediction the sample input data over the training data set. Finally, the conclusion of the chapter is presented.

4.2 A Case Study of data-intensive Workflow- NGS Pipeline

To explore the problem of simulating workflow deployment in the cloud, we have chosen a case study using the NGS pipeline [27]. These workflows were designed following the WES [1] data processing pipelines used at the Institute for Genetic Medicine, Newcastle University. In general, a pipeline consists of a composition of workflows that include typical NGS processing steps [123], which are raw sequence alignment (BWA), cleaning (Picard), sequence recalibration, filtering, variant calling and recalibration (GATK),

[1]The Whole Exome Sequencing is an application of the NGS, represents an efficient approach for studying the genetic basis of human phenotypes [135].

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 4.1.

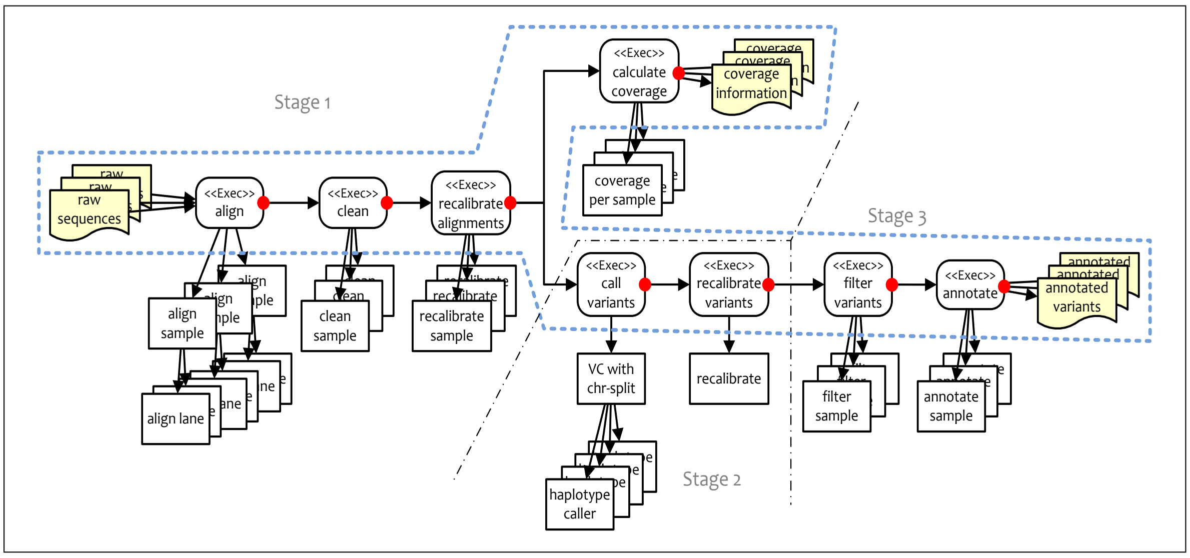


Figure 4.1: The structure of the NGS pipeline; highlighted in dashed blue is the top-level workflow; red dots indicate the synchronisation points when the top-level invocation waits for all child invocations to complete. [27]

For each step, the sub-workflows are executed synchronously in parallel over a number of samples or sub-chromosomal regions. Each patient sample includes 2-lane, pair-end raw sequence reads. The average size of the input is about 150 Gbases per sample, which is provided as compressed files of nearly 15 GiB size. The top-level workflow processes N of these samples ($N \geq 6$) are executed by submitting N sub-workflow invocations for a particular step. Then, the process waits until all of the submission are complete, before then moving on to the following step.

Overall, the pipeline involves three key stages:

(1)- preparation of the raw sequences for variant discovery and coverage calculation.

(2)- variant calling and recalibration.

(3)- variant filtering and annotation.

Stages 1 and 3 are executed in a loop so that all tools involved are invoked on each sample separately. As there is no dependency between samples in these two stages, paralellisation is straightforward. Conversely, Stage 2 runs only once for all input samples, thus parallel processing across samples is no longer possible. However, since the tools used in Stage 2 can operate independently on individual chromosomes (or even on smaller sub-chromosomal regions), the pipeline exploits that property by splitting each exome within each sample along chromosome boundaries. Then M variant calling sub-workflows are submitted ($M \geq 23$) each with a particular chromosome region taken from all input samples.

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.

4.2.1 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 that 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 4.1 shows the architecture of e-SC deployed in the

Azure Cloud.

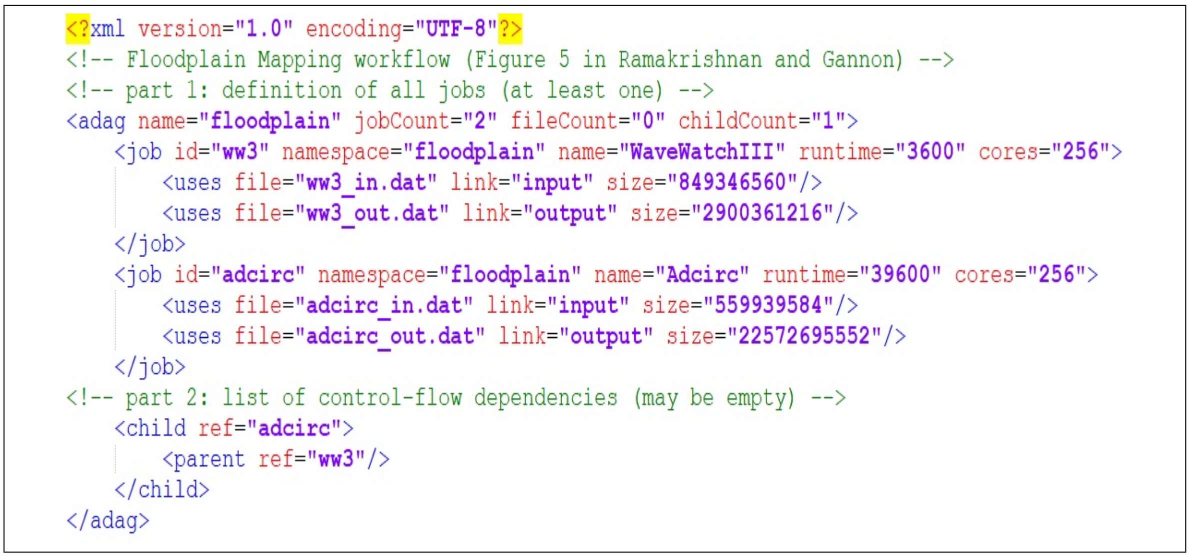
4.2.2 Pegasus architecture and workflow enactment model

Pegasus is a workflow compiler that cooperates with the centralised workflow execution system, Condor DAGMan [110]. Although workflow instances have input data that are required for computation, they are independent of the execution resources. It, therefore, creates Pegasus as an executable workflow by mapping the workflow instance onto the execution resources. And then, the workflow engine executes the workflow tasks in order, depending on their dependencies. Once the jobs are ready to run, this can be determined by the Condor DAGMan, which submits them to the HTCondor queue for the execution. For data storage, the shared file system can be configured by the user, such as for the Network File System (NFS). Additionally, Pegasus can plan the workflow using an object store such as Amazon S3.

As indicated in Chapter 2, the description of the abstract workflow in XML format is used as an input to Pegasus and called DAX. This file is captured in all the tasks that achieve computation, required inputs and expected output data for each task, the dependencies of these tasks, and the estimated execution time. The logical identifier represents all input/output datasets and executable tasks. WorkflowSim follows the execution model of the Pegasus WFMS. Often, a workflow is modelled as a DAX relationship. Of the assumption is that $DAG = (V, A)$, where the 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. However, a workflow consists of tasks, each one of them represents a node of computation, and the edges can denote the task dependencies between these tasks that specify the order in which they can be executed. Figure 4.2 illustrates the Floodplain Mapping workflow that contains two nodes “ww3” and “adcir” and their dependencies, where this description represents the abstract to an executable synthetic workflow by WorkflowSim.

4.2.3 NGS Pipeline Dataset

We have used NGS pipeline execution information that is supported by ReComp team [1] that included the Blocks names, Input data size, Output data size, and execution



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Floodplain Mapping workflow (Figure 5 in Ramakrishnan and Gannon) -->
<!-- part 1: definition of all jobs (at least one) -->
<adag name="floodplain" jobCount="2" fileCount="0" childCount="1">
    <job id="ww3" namespace="floodplain" name="WaveWatchIII" runtime="3600" cores="256">
        <uses file="ww3_in.dat" link="input" size="849346560"/>
        <uses file="ww3_out.dat" link="output" size="2900361216"/>
    </job>
    <job id="adcirc" namespace="floodplain" name="Adcirc" runtime="39600" cores="256">
        <uses file="adcirc_in.dat" link="input" size="559939584"/>
        <uses file="adcirc_out.dat" link="output" size="22572695552"/>
    </job>
<!-- part 2: list of control-flow dependencies (may be empty) -->
    <child ref="adcirc">
        <parent ref="ww3"/>
    </child>
</adag>
```

Figure 4.2: Part of the Floodplain Mapping workflow (XML with 3 tasks)

time. This information was collected from seven real runs of the NGS pipeline, where these had different input samples 6, 10, and 12 on 12 VMs. We analysed this information to extract the data that was indicated in Chapter 3, where these data are used to generate the prediction equations that are required to build the *estimation module*. The traces data and analyses are available in Appendix B for 6-sample NGS execution.

Prediction Mechanism: The prediction mechanism relies on both the training set and the testing set. The training set can be constructed from the provenance data as in Section 4.3 by considering only a small input sample of the NGS pipeline, where the framework uses the training to produce a fairly accurate prediction of the runtime of the NGS pipeline when one needs to test large input sample with a different VM's number. The testing set represents an operation of forecasting the runtime and the resource numbers which are required to run the NGS pipeline on the cloud hence the produced estimated time from the framework based on the training set can be compared with an actual execution time of the NGS pipeline to test how accurate it is.

4.3 Methodology

In this section, we demonstrate the stated methodology to simulate unlike Pegasus workflow in the WorkflowSim and normalising this simulator as a framework which targets for predicting the runtime execution of the NGS pipeline workflow. The methodology contains three basic phases:

(1-) modelling NGS to synthetic workflow.

(2-) building the estimation module.

(3-) extracting input parameters, these phases are shown in Figure 4.3.

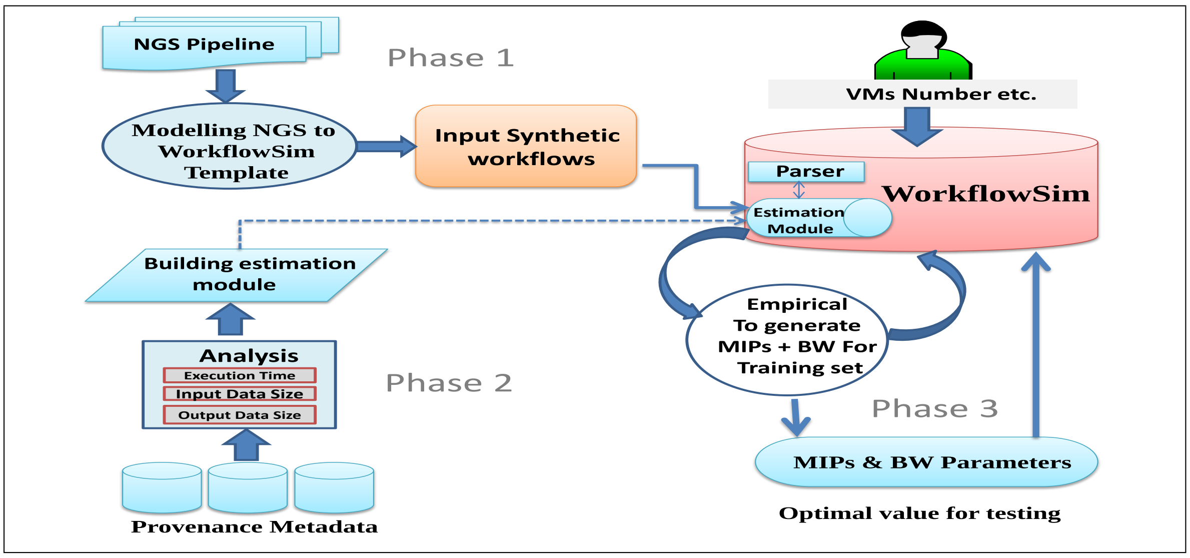


Figure 4.3: The methodology diagram of runtime prediction for the NGS pipeline using WorkflowSim.

4.3.1 Phase 1 (Modelling NGS to Synthetic Workflow)

i- 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, a way to map an e-SC workflow into one that WorkflowSim could enact had to be found. 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 one to trace the complete graph of tasks and workflows that were involved in producing a specific output. They include the block execution time and the amount of data transferred by each block. These data were used to reconstruct the applied NGS pipeline workflow as a WorkflowSim workflow. Each task in Pegasus may run on different VM, unless there's clustering turned on, so that is possible to model e-SC workflows 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. The mean that every e-SC workflow has to be split into parts connected by the subworkflow submission blocks; this is depicted in Figure 4.1. Following this approach, an execution trace can be mapped on the NGS pipeline, which runs on e-SC as a WorkflowSim workflow descriptor. Figure 4.4 illustrates this mapping. For different number of patient samples in the batch, there is a differently sized DAX (DAG in XML) descriptor.

The next step in this phase is to fill out the synthetic pipeline template for the task list and the dependencies of the task; such steps are summarised the constructed abstract level of the workflow.

ii- WorkflowSim Input Synthetic Workflow

The function of this step is related to the conversion from e-SC provenance traces to WorkflowSim workflow model, in order to complete the modelling approach. This is done to fill out the synthetic workflow template, for the list of the tasks correctly, for the dependencies of the tasks, for the input and output data relevant to the tasks. The filling in the template depends on the provenance data and the NGS pipeline invocation in the collection of manual data.

Firstly, all computation units in the NGS pipeline invocation (i.e. workflow and sub workflow) need to be specified and identified with an abbreviated name (e.g., BWA-A1 Align, BWA-B1 Align, Picard1, VCF1, etc). These abbreviations are considered as tasks in the abstract, where each task refers to a computation node. Secondly, however, these tasks have a hierarchical order to organise their execution correctly. Therefore, there is a need to extract this relationship between them as it appeared in the pipeline invocation. Finally, to specify the data dependencies of these tasks, there is a need to map the dependencies' definitions by defining which task is a parent or child. To this end, the constructed form is the synthetic workflow, which structurally identifies the abstract but does not fully contain the information about the input and output data and the execution time of each task. The filling action of these parameters in the next phase will be described. The approach taken here can be applied to other workflow systems.

Accordingly, as was explained above about the description of the workflow execution model, the NGS pipeline from an execution model in e-SC was converted to an execution model in Pegasus. Figure 4.4 shows the result of a hierarchical pipeline graph that can be represented by DAG 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 pipelines, the implementation is part of 6 paths of the workflow schema (i.e., 6-sample input); for running N samples pipeline , it will be implemented within N paths.

4.3.2 Phase 2 (Building Estimation Module)

An abstract template that was outputted from the previous phase is incomplete and still needs the process to fill out missing information that is concerned with the execution of the tasks such as the required input data, expected output, and estimated execution time. In this phase, the method of building an *estimation module* is described. To achieve this action, we need to apply the following steps:

i- Data collection and analysis

The primary task for this step is to shed light on the *estimation module*, which includes constructing and integrating it into WorkflowSim. This provides an approach for filling out the missing parameters to complete the synthetic template. This can be achieved by analysing and collecting information from the provenance file of the past execution of the NGS pipeline; this information is required for extracting the prediction equations.

Initially, the information (data) is usually downloaded in a CSV file by a Prolog program as mentioned in Chapter 3. This needs to be arranged before extracting the specific elements. In addition, the data file needs manipulation and analysis, therefore, it can be edited in Excel Spreadsheet, containing all relevant fields of information

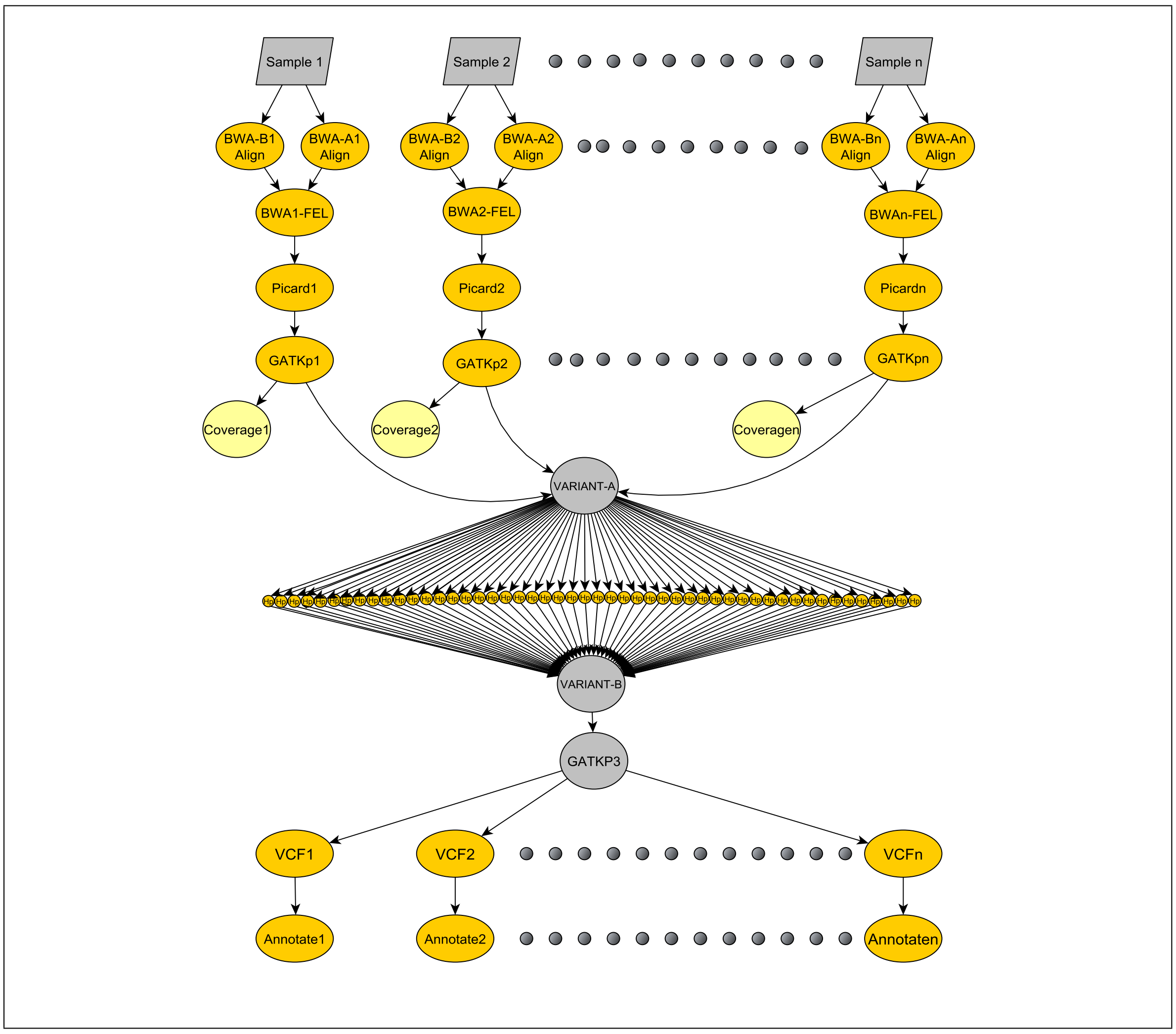


Figure 4.4: Invocation graph of the NGS pipeline with N samples; as in Figure 4.1 but in a form that would be acceptable to the WorkflowSim.

Table 4.1: 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

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 helps 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 4.1 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. This is also to derive the output data volume that will be used as input for next task.

ii- Extracting prediction equations

In the previous step, a method of collecting and analysing a required data from the e-SC provenance file was focused on. In this step, the extracted data is focused on by using the valuable factors to derive the prediction equations for building an *estimation module*. The reasons that prompted us to build like this prediction model are:

(1) Running the NGS pipeline tasks in the WorkflowSim that takes into consideration its structure, as in Figure 4.4 needs to know the execution time of each task before the submission is done. This is one reason to complete the synthetic workflow template with an expected runtime of each task. Therefore, a prediction model for estimating the runtime needs to be built.

(2) The basic structure of the NGS pipeline is presented in Figure 4.4, which shows a number of linked blocks. Each block can process a given amount of data, that means the execution consumes data which takes time and then produces an output data. The product data from the first block is then propagated to the second block of the NGS pipeline; in a similar way for the second block, this will propagate amount of output data for the third block until the last block. However, the synthetic workflow needs to be filled out for the input and output data size for each task before the submission to the WorkflowSim. Therefore, the prediction equations need to be setup prior to the execution as it determines the output data size for missing information, eventually that will lead to building a unit acting as a predictor.

To achieve the intended purpose, which is handling missing values, an *estimation module* is established that is capable of prediction for both runtime and output data size of each task. Through the analysis of data provenance, it is evident that input data size and the execution time have a relation as a linear relationship of time and data size. The relationship in identifiable in a linear regression. Providing an adequate number (even less number) in association of the observed data between input data size and execution time is relevant for this regression. For example, Figure 4.5 shows a prediction equation of the BWAn_FEL (The integer n is 1,2,..,6) for three 6-sample input using a simple linear least squares regression model. Looking at Figure 4.5, the table shows the number of observations of the execution block (BWAn_FEL) while looking at the same Figure (to the right) shows a linear regression line that has an equation of the form: Y = bX + a, where X is the explanatory variable (e.g., represents input data size) and Y is the dependent variable (e.g., representing n predicted time). In the same Figure 4.5, the accurate indicator of a generated function can be controlled by the R^2 that have values between 0 and 1, and the function is more accurate when R^2 is close to 1. Additionally, a same method is depended on that is used to obtain further equations of the output data sizes.

It may be seen through an analysis of data provenance that the runtime does not scale linearly with the input data size. In such cases, we need to find a solution, either by using the upper or lower bounds on the execution time, if possible, or by using a different factor, such as used in Haplotype workflows. Hence, the main indicator is that the Haplotype workflows uses the same input data sizes of all patient samples but is configured to read different chromosomal regions. Thus, the region's length was used as a division factor for different input sizes. Therefore, the region length was used as a division row matrix factor to compute function for the same input sizes (e.g., the interval row matrix such as ["chrX:1-77635280"], ["chrX:77635281-155270560"], etc., where there is the same input value "1.8767E+10"). For completion, building an *estimation module* is used to extract two type of the prediction equations that are described as follows:

1- Runtime prediction model

Execution time prediction is an important factor in cloud computing and in simulation [58]. To support completion of building an *estimation module*, a set of predictions for the equations (each block in the pipeline needs equation, see 4.1) that work as an execution time estimation model, which statistically estimates the runtime of the tasks using past observations by using past information [82]. Where these data exist, these equations in the *estimation module* help to estimate runtime of the task as a function of one input parameter which is the data size.

A least squares linear regression method as a solution to address this estimation is used. This approach manages the relationship between two variables, *size*, which represents an input data volume, and *runtime*, which represents a runtime to extract *runtime* from *size*. Following the structure of NGS pipeline as in figure 4.4, each set of tasks represents a group of observations, which have one set of prediction equations. These observations should be under the same name and may be collected from one or more invocation. For example, the tasks (Picard1, Picard2,.., Picard6) are in the 6-sample input pipeline; they have one prediction equation as follows:

$$runtime = 8.95734E - 07 * size - 1840.06; \tag{4.1}$$

Thus, it is a necessary that such type of estimation in the framework supplies expected execution time of each task in the synthetic pipeline. Concurrently, the prediction equations can be used to extract runtime of large-scale input data when the framework is used for a training test.

2- Output data volume prediction model

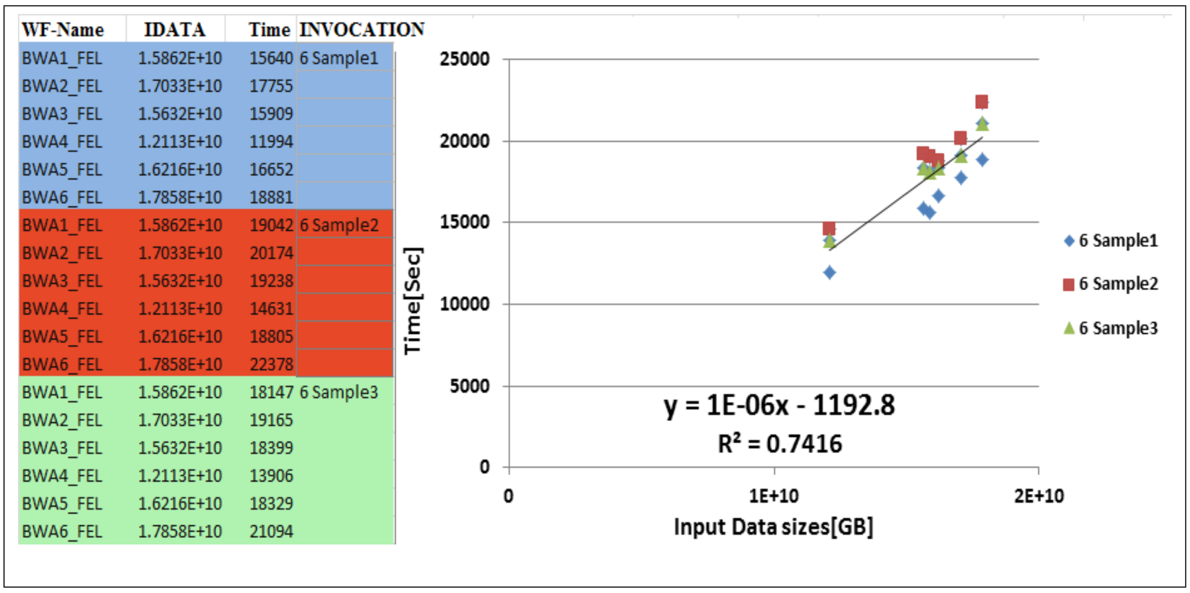
The approach taken by runtime prediction method to generate a model is followed in this method as well. However, the difference lies in two considered variables, that is the *size* input variable (i.e. input data volume) and *output*, which is a dependent output variable (i.e. output data volume) to extract *output* from *size*. For example, the tasks the tasks (Picard1, Picard2,.., Picard6) are in the 6-sample input pipeline, and they have one prediction equation as follows:

$$output = 0.981865508 * size - 1220190052; \tag{4.2}$$

In addition, to support completion of building an *estimation module* by adding a set of prediction equations, each block in the pipeline needs equation as the equation (4.2) that work as an output data size estimation model. 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.

3- Managing the same input of the Workflows issue

In particular, there is a case where using data input size to generate estimation equations was inadequate for the Haplotype workflows that was the entry-point to the part of the pipeline that runs under the chromosome-split regime. This indicates that the Haplotype workflow as the input uses data of all patient samples but is configured to read different chromosomal regions. Thus, the region's length was used as a division factor for different input sizes. The region length was used as a division row matrix factor to compute function for the same input sizes (e.g., the interval row matrix such as ["chrX:1-77635280"], ["chrX:77635281-155270560"], etc., where there is the same input value "1.8767E+10").



WF-Name	IDATA	Time	INVOCATION
BWA1_FEL	1.5862E+10	15640	6 Sample1
BWA2_FEL	1.7033E+10	17755	
BWA3_FEL	1.5632E+10	15909	
BWA4_FEL	1.2113E+10	11994	
BWA5_FEL	1.6216E+10	16652	
BWA6_FEL	1.7858E+10	18881	
BWA1_FEL	1.5862E+10	19042	6 Sample2
BWA2_FEL	1.7033E+10	20174	
BWA3_FEL	1.5632E+10	19238	
BWA4_FEL	1.2113E+10	14631	
BWA5_FEL	1.6216E+10	18805	
BWA6_FEL	1.7858E+10	22378	
BWA1_FEL	1.5862E+10	18147	6 Sample3
BWA2_FEL	1.7033E+10	19165	
BWA3_FEL	1.5632E+10	18399	
BWA4_FEL	1.2113E+10	13906	
BWA5_FEL	1.6216E+10	18329	
BWA6_FEL	1.7858E+10	21094	

Figure 4.5: Linear model of equation prediction of the BWAn−FEL workflow that extracted from three runs of the NGS pipeline 6-sample1, 6-sample2 and 6-sample3; where n is from 1 to 6, Y is a predicted runtime, x is an input data size.

iii- Integrating the derived equations phase

In this step, the equations extracted in the previous step are gathered in an *estimation module* that supports the framework for prediction. This module should be integrated and work in coordination with the other WorkflowSim equipment, i.e. it works in coordination with the parser. The existence of such a mechanism in the framework carried out a function of filling in missing parameters in the synthetic pipeline according to the prediction sequence (i.e., using first the output data set and second the execution time). There is then a framework that can then predict information about the pipeline without consulting the provenance file to explore the runtime and output data size. This facility should be available in the framework because its functionality is testing a pipeline with a large input data set that has no information on the provenance file at this point.

Some operational preparations for the framework need to be undertaken, including a working method of this *estimation module* predicting the relevant parameters of the WorkflowSim environment. In the next phase, this is presented.

4.3.3 Phase 3 (WorkflowSim for Extracting Input Parameters)

i- Filling in Runtime and input/output Data Size

The output of phase 1 is describing and presenting an abstract of the pipeline with a new XML file format that is not composed of all parameters (i.e., it lacks the parameters such as execution time, output data size, and the input data size). For example, the following is a brief record on XML file of the NGS pipeline that represents the “PICARD1” workflow (task) with the missing parameters.

```
<task id= "PICARD" namespace= "NGSprocessingPipeline" name= "Picard-
   Tools-Workflow" runtime= "---">

        <uses file= "BWA1_FEL_out.dat" link="input" size="----"/>
        <uses file="PICARD1_out.dat" link="output" size"----"/>

</task>
```

As indicated earlier, the remaining structure of the whole pipeline as presented in Figure 4.4 was constructed manually as an XML file. Since the node represents individual workflow/subworkflow that identified the tasks abbreviation (id), where the logical identifiers with the task should be invoked, that contains a list of input and output file names which are relevant to the task. To complete the XML file and make it ready to simulate what is needed to fill out the parameters runtime, input size and output size with their estimated values are used.

Through the WorkflowSim running steps, the parsing is the first step in which the proceeding is being of syntax analysing the XML file. However, the original parser module of the WorflowSim cannot parse the uncompleted XML file of NGS pipeline. Therefore, we have modified a new version of the parser which is working under the coordination of the proposed *estimation module* to overcome this problem and complete the missing parameters. At the beginning of the operation of the parser, two matters are considered: a runtime estimation should be given for each task before the simulating; the output data volume should be with a predefined execution time because the execution time depends on the input data size. The following segment of Java code shows a method of filling in the "Picard" node in the XML with input and output data sizes before moving into populating the execution time stage.

```
if(GetBlockName.PICARD.contains(nodeName)){

    List list = node1.getChildren();
    Element file0 = (Element) list.get(0);
    Element file1 = (Element) list.get(1);
    String inout = file0.getAttributeValue("link");
    Attribute attribute = file0.getAttribute("size");
    attribute.setValue(BWA_FEL_common.get(BWA_FEL_count));// Get the parent output
        data
    BWA_FEL_count++;
    fileSize1 = file0.getAttributeValue("size");
    size1 = Double.parseDouble( fileSize1);
    size1 = data1.getOutputData(nodeName,inout ,size1); // estimated input data size
    NumberFormat formatter = new DecimalFormat("###############");
    String estimatedValue = formatter.format(size1);
    Attribute attribute1 = file1.getAttribute("size");
    attribute1.setValue(estimatedValue);
    PICARD_common.add(estimatedValue);                    // Save estimated data in XML

                                              }
```

In the case of runtime parameters, each is assigned by the value of predicted execution time that will be extracted by the *estimation module* from input data size.

ii- Extracting Input Parameters

There will be several sets of parameters with a required value to operate the WorkflowSim; this step represents a description of the execution environments, where the NGS pipeline needs to be simulated and extract execution time for each task. Then we will be able to conduct a realistic simulation of scenarios like in a real cloud within the simulation environment.

The main target of the framework is to gain a better simulation time to predict the whole runtime of the pipeline with different sample input sizes. For this purpose, we can submit the pipeline to our framework with only the size of the input sample (small size of the input sample). However, as yet we have no idea about the parameters of the execution environment (e.g., MIPs and bandwidth). This step can be considered as an important step towards building a generic training set that will be used for the testing set.

As a solution to the above, an approach to allocate the capacity of the computation and transmission units is proposed by driving the framework to construct the training set in an empirical method (the description of this mission was in Chapter 3). The approach is, therefore, to gather provenance data from sample executions with small input sample numbers (i.e., the actual execution time of the whole pipeline is short), use this data as input to WorkflowSim to make a comparison with an estimated simulation time, and find a relative error between them. This process is repeated a couple of times that are identified across the two boundary values of both the MIPs and bandwidth. As a result of running the framework many times, the relative error matrix can be generated, which can determine the optimal values of the parameters by searching the matrix for a minimum relative error.

The framework configuration is retrieved by tracing the simulator itself many times to generate unknown parameters values (e.g., MIPs and BW), with the approach's steps summarised as follows:

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.

   $$errorratio = \sum_{j=1}^{N}(RT - ET)^2. \tag{4.3}$$

   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 across a given range and detect the minimum $errorratio$.

4.4 Experiments and Evaluation

This section presents testing of the methodology with the NGS pipeline execution on an adapted framework. A small data set is used that is supported by ReComp team to run the experiments. In these experiments, we wanted to measure the capacity values of the MIPs and BW parameters by following up on the methodology. We have focused on finding these parameters values of the 6, 10, and 12-sample input data. As a result of these experiments, we constructed two small training sets which are $\{6\}$ and $\{6 + 10\}$. Also, these training sets would enable us to test larger input data such as 10 and 12-sample input on the small training set $\{6\}$ or test 12-sample input on the training set $\{6 + 10\}$. Finally, we discuss the evaluation of the estimated results for relative errors between different running samples to derive the expectation of the enhancement our framework.

4.4.1 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 $= N \times 8 + 53$, 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, the estimation of the output data size and runtime using the simulation and then using the output data size to generate the input data size for the subsequent task is generated.

We configured WorkflowSim to simulate one datacentre and 12 virtual machines (VMs) to represent three engines in the Azure cloud, each with four execution threads. The capacity of the computation unit with MIPs and bandwidth (BW) values are derived and allocated by using the application of an empirical method (see Chapter 3) in running the framework multiple times. Moreover, for data transfer delay, a shared file system is 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 is used. The space shared mode of the VMs has been defined as only one VM can running 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.

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. And the 6-sample input set was the minimal size for which the pipeline completed successfully.

4.4.2 Experiment Training set

To create a training data set, firstly, we have to collect the operational characteristics of this training data set on the framework. For example, finding the capacity of the computation unit and transmission data unit values (i.e., MIPs and BW). There is also the need to extract the real execution time of the NGS pipeline with the sample input data. Knowing how long it takes to execute the process on the cloud and using this time to be compared with estimated time is needed. This execution time is found in the provenance file of the previous execution and can be extracted to be used for comparing with an estimated time to find the relative errors. Following the completion of extracting the parameters, we can use all sample data as the training set for the 6-, 10-, and 12-sample cases, each according to its set. In general, this experiment seems to be an evaluation of the simulation permits prediction by implementing the case study. This determines whether the methodology can predict a runtime and output data sizes of the pipeline, with enough accuracy that is useful to scale up the number of input samples.

We conducted our experiment by using the dataset on the previous seven executions for three kinds of samples as input data to the pipeline, including 6-sample, 10-sample, and 12-sample (See Subsection 4.2.3). The runtime (whole execution time of the pipeline) has to be extracted for each past execution from the provenance file and then used to run our framework over the same input scale to give an estimated execution time to compare with the real execution time. While running the simulation multiple times to explore a minimum error, and after finding such value, we will rely on it to determine both parameters values. Depending on a minimum relative error we can select MIPs and BW values which have an optimal value at that point, i.e., when they were in the setting, and a minimum error was obtained. In this way we were able to find the parameters' value (i.e, MIPs and BW). The same process was replicated over all 6-, 10- and 12-sample input data.

The simulation environment is now ready for implementation; the pipeline to generate an estimated time on a specific training set mentioned above can be implemented.The 6-sample would be trained on the training set $\{6\}$; the 10-sample would be trained on the training set $\{6 + 10\}$; the 12-sample would be trained on the training set $\{6 + 10 + 12\}$. As real executions potentially vary with each run and WorkflowSim gives a 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 is as follows:

$$RelativeError = \frac{\sum |X - ET|}{\sum_{j=1}^{N}(RT)} \tag{4.4}$$

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

Within scenario one, the 6-sample set parameters were found and used as training data in predicting the 6-sample input case as shown in the Figure 4.6. Within scenario two, the parameters of the 6- and 10-sample were found and set as training data to be applied for prediction the 6- and 10-sample input case as shown in Figure 4.7. Finally, within scenario three, the parameters of the 6-, 10-, and 12-sample set were found and set as training data to be used for prediction of the 6-, 10-, and 12-sample input case as shown in Figure 4.8.

In each case the values of MIPs and BW that gives the minimum error over the training set are used to give the estimated time. The results in each case are shown in Figures 4.6, 4.7, and 4.8. 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

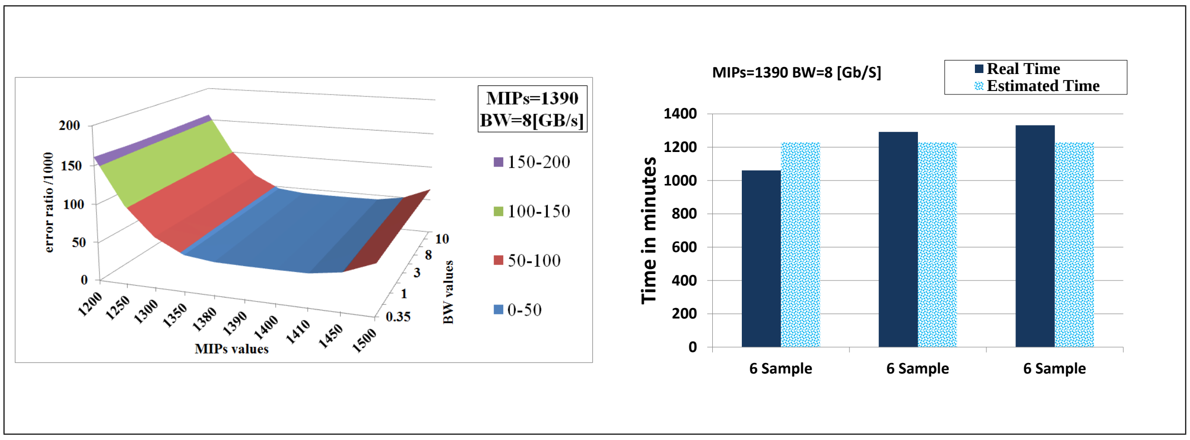


Figure 4.6: Scenario One: extracting parameters of 6-sample data and predicting itself sample set; where on the left side MIPs and BW values; on the right side 6-sample training set.

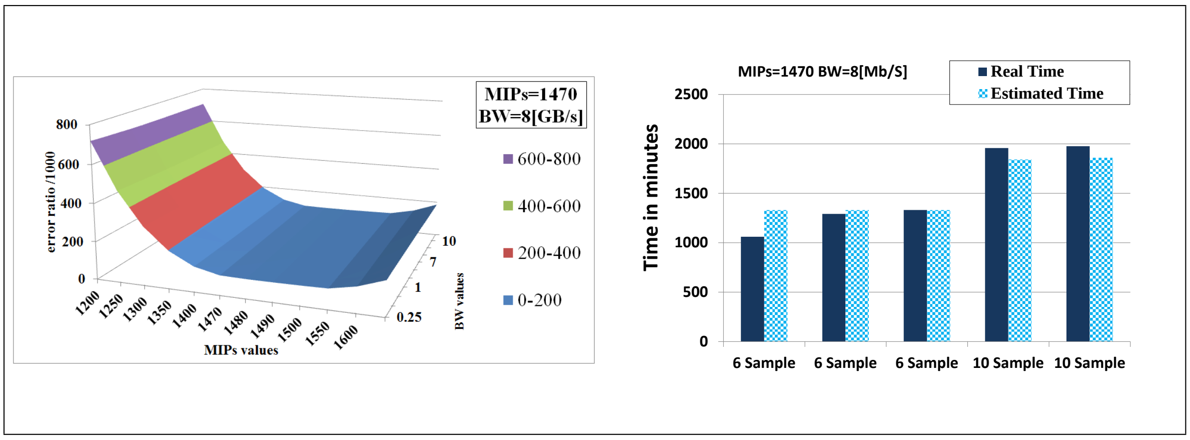


Figure 4.7: Scenario Two: extracting parameters of (6+10)-samples data and predicting itself sample set; where on the left side MIPs and BW values; on the right side (6+10)-sample training set.

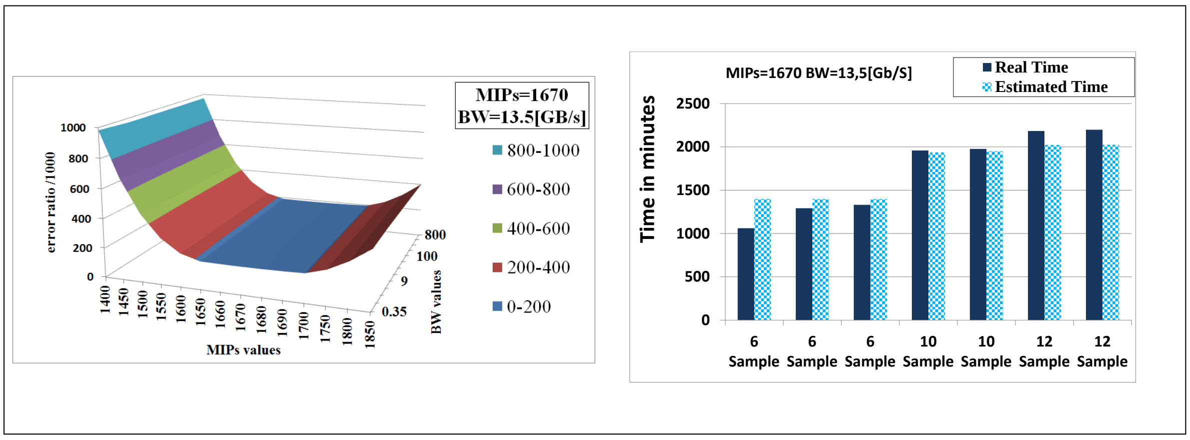


Figure 4.8: Scenario Three: extracting parameters of (6+10+12)-samples data and predicting itself sample set; where on the left side MIPs and BW values; on the right side (6+10+12)-sample training set.

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.

4.4.3 Prediction at Input Scalable

In the previous section, the accuracy of the prediction model by comparing the actual and estimated runtime derived from the training sets was verified. This process was self-reflective in that it included the provenance data from the sample size 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 Figures 4.9 and 4.10.

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 4.9). 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 4.10). In both cases the predictions under-estimate 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 4.8. 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 4.8), 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.

Our framework emphasizes the need for more historical data to generate a better prediction. However, based on a given data set, the framework was able to be about 80% accurate. Therefore, to increase the prediction accuracy, we need to collect more historical data set of the NGS pipeline, which is costly. However, the aim was to use a small dataset, hence, what was achieved in terms of accurately may be considered to be sufficient.

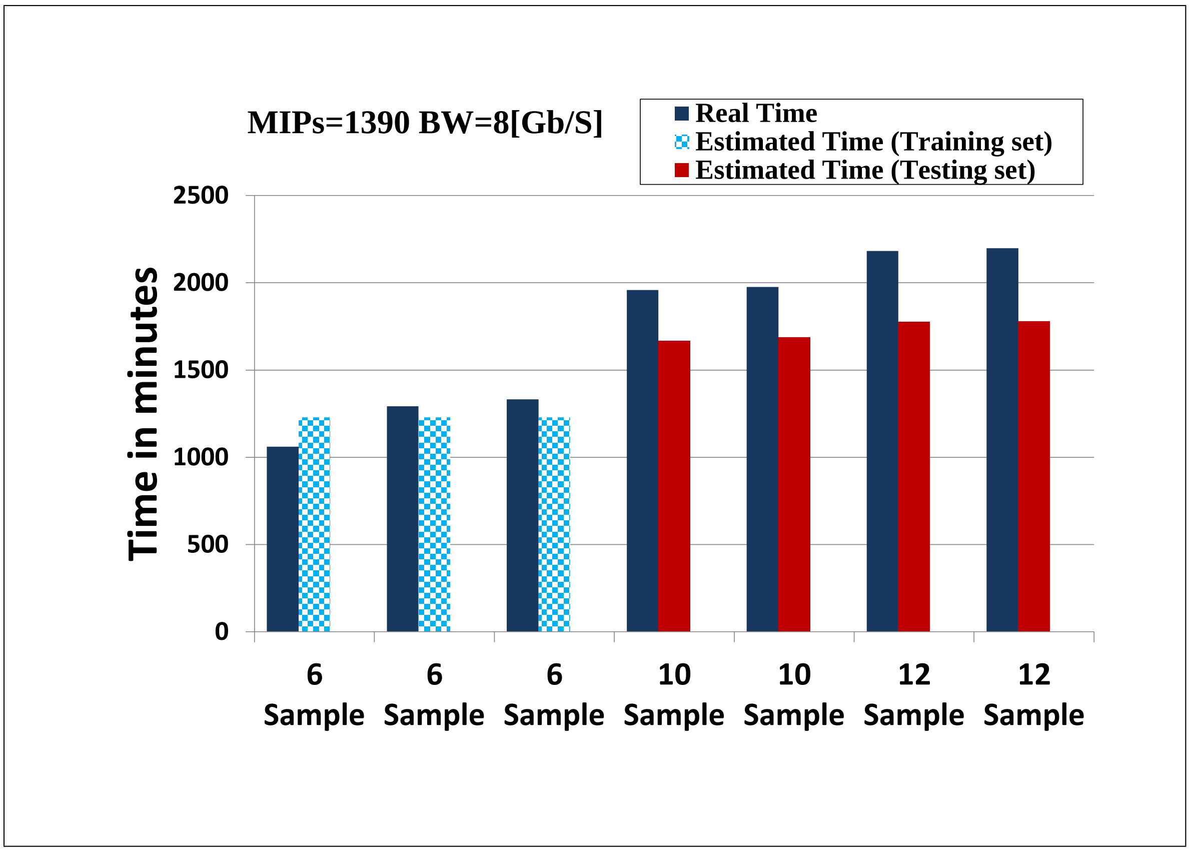


Figure 4.9: The pure prediction of the 10- and 12-sample input by using the 6-sample training set.

4.5 Conclusion

The scientific data-intensive workflows are growing increasingly complex, often consisting of a set of workflow and sub workflow with data dependencies in relations among them often referred to as a pipeline. However, there are no appropriate means on how well these pipelines can behave in a state of increasing the number of input samples. On the other hand, measuring and testing of these pipelines are uncontrollable if it could be done over computing system because it will be costly, time-consuming and most often unsuccessful. Therefore, as a solution for this issue was, we have built the framework that can predict an execution time of the pipeline in case of there is any increased in the input samples.

In this chapter, we have considered this problem by implementing the NGS pipeline workflow as a case study. The core function of the NGS pipeline is to discover vari-

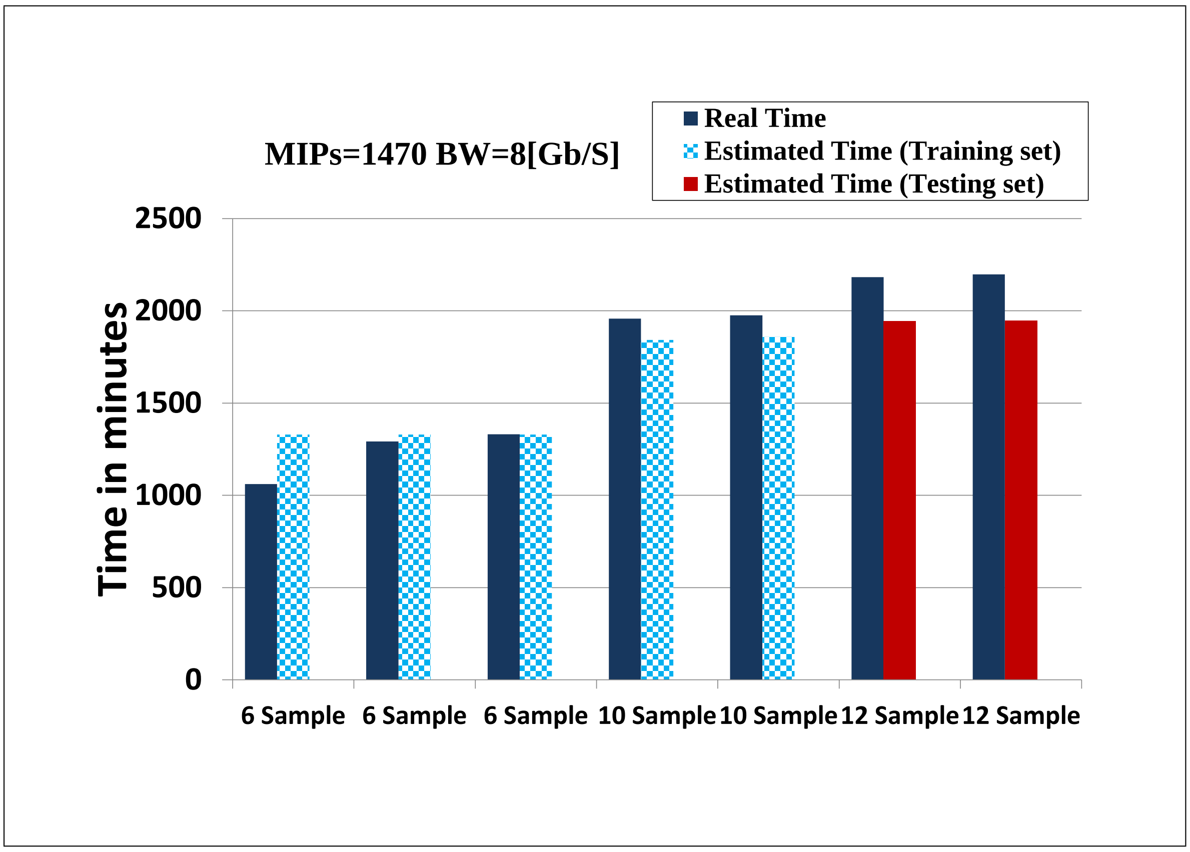


Figure 4.10: The pure prediction of the 12-sample input by using the 6- and 10-sample training set.

ants in patient's exome. With limited funds, a near-optimal deployment can be found, which minimises both the execution time and cost. This has been achieved by modifying a simulation platform to simulate the behaviour of the NGS pipeline. The operation was conducted by proposed methodology for predicting the runtime and output data size of the pipeline and integrating it with WorkflowSim and CloudSim by taking advantage from the real data that was archived in the provenance file.

Some of the experiments were conducted with an available small dataset that performs validation checks on the methodology results so that it indicated that it would enable us to build a training set. Results of the pure prediction experiments for two training data set are used to predict outside input data; these show that the framework's prediction is under-estimate for the real execution time, particularly for the 6-sample data as the training set for the 12-sample resulted in a poor prediction. However, the main role in our framework is minimising the number of training set while achieving better accuracy. Therefore, to make good prediction accuracy, we have to investigate a problem by examining a new dataset and give further consideration to the impact on parameters which are reflecting runtime prediction. We will illustrate that in the subsequent chapter.

5

Framework extension to cover enhancement

Contents

Summary

The adapted framework, together with a new dataset from a real execution, provides an enhancement of the prediction for both runtime and the resource number associated with the NGS pipeline simulation. In this chapter, the challenge of determining VMs numbers on the outside of the training set is addressed by the framework. It demonstrates an interesting thing which is to how simulating the pipeline, particularly over the framework, this enhances capability may be benefiting the pipeline workflow at in many fields that are mentioned. How these enhancements work is shown with implementing some of the experiments taking into account the new dataset.

5.1 Introduction

The primary key to several goals for critical system design and deployment decision is a predictive analysis of resource usage [81]. There are different predictive resource strategies presented in the previous works; a commonality is they find a precise prediction about how many resources are probably required for executing the computational tasks [21]. As far as our knowledge, no such strategy can predict the number of the resources needed for running the NGS pipeline. Hence, the proposed prediction framework uses a novel method which is able to speculate the increase of the execution time requirement, enabling proactive scaling to handle increases in input samples.

To address this challenge, this chapter presents a way about how to enhance the predictive *estimation module* that was set out in the previous chapter. Such enhancement is reflected positively on the entire methodology that is proposed in Chapter 4 and presented in [104]; it also improves the quality of the prediction. This enhancement is done through finding a new dataset to improve prediction and extend the ability of the resource prediction (e.g., predicting the number of the VMs). However, in addition to the main objective for collecting a new dataset, it is worth of increasing the number of the observations which will be extracted and can eventually improve the quality of the prediction by creating a new training set.

To this end, the question the framework answers is if the resources are sufficient for ensuring proper implementation of the NGS pipeline that can be deployed efficiently?. Therefore, the interest is to see if the framework can be incurred to determine the resource number of the target input samples. In other words, it is worth noting if the new dataset with more and different observations can be beneficial to achieving the target purpose. If this is the case, then we can eventually improve the prediction model and the resulting framework in the next step.

To demonstrate our enhancement, we have collected a new dataset by running NGS pipeline on the Azure cloud with a different number of both the input samples and engines. During the implementation there was a failure of the pipeline execution in a certain block. This problem led to finding a solution that enhanced the methodology in a way that can make a prediction only partially according to the successful blocks of the pipeline.

This chapter shows that our framework is not only able make a runtime prediction but is also skilled enough to forecast resources requirements (i.e., the VMs number) for projections that match the execution of the time.

The rest of this chapter is structured as follows: Section 5.2 presents in details the requirements of the new dataset to enhance the framework in two aspects including: adding a new prediction parameter (VM number) and improving the prediction. Section 5.3 describes the pipeline failures in the cloud and how it is motivated for flexibility for simulation that deals with an unsuccessful pipeline. In Section 5.4 the partitioning pipeline workflow approach over the simulation prediction is presented. Then the results and experiments are shown in Section 5.5. Finally, Section 5.6 summarises and concludes this chapter.

5.2 NGS Pipeline a New Dataset

During the building of the *estimation module* in the previous chapter, the dataset with specified information about the resources number was used, because it ran on three engines amounting to 12 VMs. This dataset was taken from the Recomp team. Additionally, they have one observation that worked as an outliers in the measured execution times effect on the 6-sample as the training set, as well there was an underestimation because fewer data elements in the training set were used. Therefore, to overcome these obstacles, we decided to collect a new dataset by running the NGS pipeline on the Azure cloud.

However, the proposed framework works based on the training dataset that predict an execution time on the large data input sample and a significant progress in the prediction is required. Therefore, there needs to be an improvement that can feed a variety of information while building a training set; these can be collected from a new dataset. For example, if the collected information contains a database on engine numbers, then the framework can expect a runtime for the NGS pipeline to be the difference in the number of engines. On the other hand, the prediction model is evaluated based on the accuracy of the predicted results. Where, the reliability can be measured by the outputs, that is if it is closer to the actual runtime, then it would be more reliable. Thus this indicates that the accurate prediction is dependent on the extracted information from the pipeline in the past.

Actually, the dataset is a collection of information through the execution of the pipeline on the Azure cloud. We ran the NGS pipeline in e-SC deployed over 3, 6 and 12 VMs, where 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 purpose of this exercise is to focus on enhancing the quality and coverage of information to cover instances that demonstrate a variety for the number of sources running the results. The total runs of the NGS dataset are contained as 6-, 12-, and 24-sample inputs, which were executed over 12, 24, and 48 cores. Table 5.1 shows the runs on the NGS pipeline.

Table 5.1: The actual running of the NGS on Azure

The cores number	**12 Cores**		**24 Cores**		**48 Cores**	
Input samples	***Success***	***Failed***	***Success***	***Failed***	***Success***	***Failed***
6-sample	3	1	3	6	3	3
12-sample	3	2	3	0	3	1
24-sample	3	0	3	0	4	0
Totally 41 times have been executed NGS pipeline The number refers to replicate running (Number of times)						

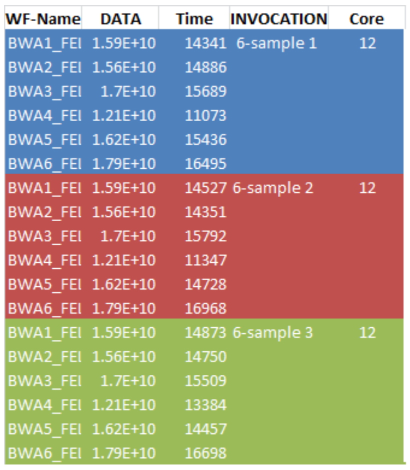
5.2.1 New Metrics prediction- VMs

Starting the application over the cloud requires the user or developer to choose the number of resources that can be used. There is also a possibility of scaling up/down these resources at run time. Resources prediction is guaranteed by the developed framework; it helps in deployment decisions. It is, therefore, necessary to promote measures to enhance the framework to tackle this issue. Here, the estimation focuses on VMs prediction by comparing the number of cores and execution time of actual runs and predictions. For example, a 6-sample training set on 12 cores as the cases of data set can use 24 cores or 48 cores. Where the object of our prediction is the VM's number, so that there is prediction of two cases through changing parameters of the simulation environment, the "VmNum" number and setting a value to either 24 or 48 for the prediction accuracy can be reported.

5.2.2 Improving Prediction

Improving prediction is based on certain assumptions, such as a normal distribution for errors ranges. In such cases, assuming a normal distribution allows the use of standard deviation values that enable a given probability for estimating estimates. In the observed points, having a high R^2 value in a linear prediction makes it possible to predict with relatively low error using a linear model. In this case, the R^2 limited dispersion among values. However, when R^2 is low in a linear least squares regression method, the data fit is potentially not sufficient, where greater data disparity is evident. This shows that the prediction method may be less effective in estimating what the empirical value should be. Therefore, model improvement can be done by increasing the R^2 value. An improvement in the prediction model depends on the data focused on. Therefore, collecting a new dataset provides an opportunity to make prediction improvement. As seen in Chapter 4, the extracted R^2 from previous data was 0.7416, but with a new dataset, R^2 raised to 0.889, as in Figure 5.1. This included the improvement in the *estimation module* that led to the improvement of final prediction. This improvement is made possible by a closer fit for the data to the linear model, making it more predictable using this approach.

Improving the data through other regression could potentially reflect improving the runtime prediction of the pipeline. However, weighing observation of each workflow/-subsorkflow will be increased (as seen in the results), where they are then reweighed using a least squares regression to generate the prediction equations. There are still potential for outliers points that cause inaccuracies in the prediction model so the treatment may be assigned to find prediction through the new observations from both datasets.



WF-Name	DATA	Time	INVOCATION	Core
BWA1_FEI	1.59E+10	14341	6-sample 1	12
BWA2_FEI	1.56E+10	14886		
BWA3_FEI	1.7E+10	15689		
BWA4_FEI	1.21E+10	11073		
BWA5_FEI	1.62E+10	15436		
BWA6_FEI	1.79E+10	16495		
BWA1_FEI	1.59E+10	14527	6-sample 2	12
BWA2_FEI	1.56E+10	14351		
BWA3_FEI	1.7E+10	15792		
BWA4_FEI	1.21E+10	11347		
BWA5_FEI	1.62E+10	14728		
BWA6_FEI	1.79E+10	16968		
BWA1_FEI	1.59E+10	14873	6-sample 3	12
BWA2_FEI	1.56E+10	14750		
BWA3_FEI	1.7E+10	15509		
BWA4_FEI	1.21E+10	13384		
BWA5_FEI	1.62E+10	14457		
BWA6_FEI	1.79E+10	16698		

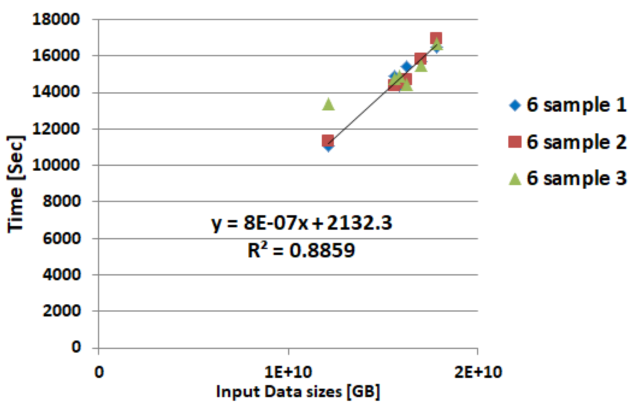


Figure 5.1: Improving the Linear model of equation prediction.

5.3 NGS Pipeline Failures in Cloud

In general, workflow execution failures may happen because resources are down due to common reasons such as: data becoming unavailable, networks are down, bugs in the system software or the application components appeared, and many other reasons. The approach that is used by Pegasus is tightens collaboration between planning and execution processes of the workflow. Although this method does not prevent failures, it provides robust mappings by keeping track of the intermediate produced data and by allowing for the presence of the latest information about resources [47]; this facility will enable the process of the workflow to restart execution again at the breakpoint.

During the collection of information in running the pipeline, execution breaking becomes a notable problem. One of the main issues in the NGS pipeline execution is the ability of failures to occur. As we cleared that up the NGS pipeline was deployed on

Azure Microsoft cloud, it works as black box therefore there is no an explanation from our side for why such breakings occur at that time. Common types of failures in the NGS pipeline in Azure are listed below:

1) Azure VM failures; 2) JRE errors; 3) tool errors (Haplotype Caller, GATK ph3); 4) not enough memory; 5) not enough disk space; 6) I/O bottlenecks; 7) occasional slow response from VMs (+24 vs 5 hrs); 8) misconfiguration of the tools (not enough memory for BWA or samtools; not enough heap space for GATK).

Unfortunately, Pegasus' recovery capability from failure mentioned above does not exist in the e-SC system. It means that once errors happen, we want to start a new pipeline execution from the beginning instead of starting from the point of failure. Therefore, the pipeline is restarted again with the same input sample from the beginning. Based on our knowledge and what happened during the NGS implementation, failures happened at blocks within the tail of the pipeline. Figure 5.2 defines most block breaks happening.

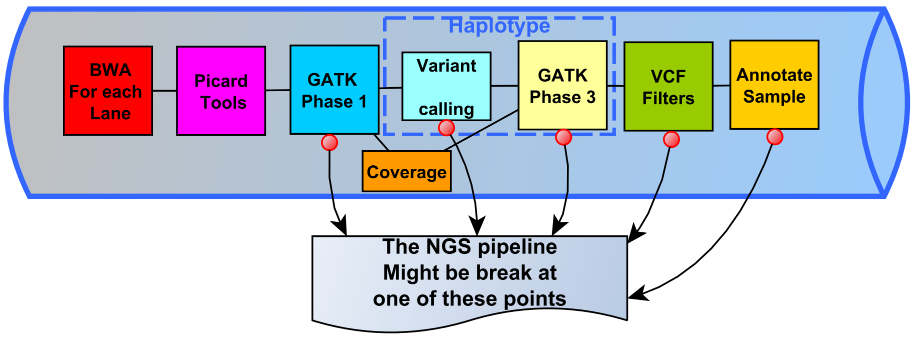


Figure 5.2: NGS pipeline breaking points.

Motivation

Most current work in the cloud simulation focus on improving modelling to meet the workflows' simulation requirements, particularly when there is a workflow that is changing; this workflow needs to test changes before putting it into practice. Based on current work, no simulator has considered pipeline partitioning for performance prediction over the simulation environment. In general, this potential becomes available in the simulator tool, which may have the following benefit:

(1-) It can facilitate a task for finding a fault in the real workflow deployment; however, the complex data-intensive workflow typically takes a long time to execute up to days or weeks. When the error happens within the real execution in a short time before the successful completion, which is often reproducible; this will facilitate an approach for the user/developer to monitor the pipeline execution at a specific time instead of waiting and observing the whole execution.

(2-) Facilitate a potential for researchers who want to do their experiments of partitioning a complex or even a simple workflow on such a framework that provides this facility for simulating a workflow partition.

(3-) In the prediction framework, this facility will support the beneficiary to plan and estimate a runtime performance for parts of the pipeline it depends on and what are required.

5.4 Partitioning Pipeline Workflows Over Simulation Prediction

The approach of dividing a workflow into many sub-workflows and submitting them to different sites (or VMs) is called workflow partitioning [39]. Partitioning workflows helps to improve runtime performance because there will be more execution sites available, thus parallelism is developed. However, this may not be the case where the workflow platform is the simulator. Providing enhanced framework partitioning capabilities when appropriately is constructed can increase simulation flexibility, resulting in a significant performance increase concerning such aspects as designing experiments, scheduling experiments, other experiments and simulations.

This section introduces a simulation pipeline partitioning, which provides automation and flexibility in how simulation prediction experiments are conducted. These particular pipeline templates are typically stored as XML documents, which are discussed in Chapter 4 in Section 4.3.1, in a pipeline that can be simulated by the framework. On the other hand, this pipeline component is dependent of the successful parts of any NGS run; the whole pipeline structure except the failed components can be used, where they can be removed from the synthetic workflow with their dependencies.

Approach of Partitioning

This step introduces the partitioning approach as follows:

1- Taking the original synthetic pipeline contianed as a while as described in Chapter 4 and cutting out only the broken componenets as well their dependencies, finally we can obtain a particular pipeline. For example, if the fault was at the COVERAGE block, this will then be removed until the end (i.e., ANNOTATE).

2- Extracting the actual execution time from the provenance data from the beginning to the fault point. For example, if the fault point was at the VCF block, then the execution time will be accounted for all blocks except COVERAGE and the ANNOTATE relevant blocks.

3- The simulation for testing is used to fill out missing values and to predict runtime, finding the relative error comparison between the prediction's time with extracted time from step 2.

5.5 Results of experiments

This section presents testing of the methodology with the NGS pipeline execution on an adapted framework, this time with a wide variety dataset. We have used a new data set that was collected from a real execution on the Azure cloud to run our experiments. In these experiments, the capacity values of the MIPs and BW parameters were measured by following a stated methodology. We have focused on finding these parameters values of the 6-sample input data set. As a result of these experiments, we constructed one small training set which is {6}. This training set would enable us to test of larger input data such as 6-sample input on 24 VMs or 48 VMs. Additionally, 12 and 24-sample input on 12, 24, and 48 VMs are tested on the small training set. Finally, the evaluation of the estimated results for relative errors between different running samples to derive the expectation of the enhancement of the framework is applied.

5.5.1 Experiment setup

The dataset has resulted in candidate configurations of the D13 VM with 8-core CPU, 56 GiB of RAM and 400 GB of local SSD storage that was used to run four workflows concurrently as execution threads. The different running of the pipeline with 6, 12 and 24-Sample in e-SC executed in the Azure cloud over 12, 24 and 48 as the VMs' simulation. These runs to collect the data have not been randomised because the order of the runs was dictated by cost efficiency of running the experiment setup.

Because the pipeline is simulated based on the following formula: Total Tasks = $N \times 8 + 53$, where N is the number of input samples. In this experiment, 6-, 12-, and 24-sample sets are applied; therefore, the number of tasks can be diverse as 101, 149, and 245 tasks respectively. Firstly, we have to run the framework to generate the environmental parameters, that is MIPs and BW, after that these are used for testing all input samples with various cores setting. Simultaneously, a stage creation step provides the *estimation module*, generating runtime and input/output data size for each task. The capacity of the computation unit with MIPs and bandwidth (BW) values is allocated and derived by using the application of an empirical method that runs the framework multiple times by setting the MIPs, which is arranged between 900-1500, and the BW arranged between 10-1000. As the training data we have used the smallest executions of 6 patient samples ran on 3 engines (12 workflow execution threads, equivalent to the 12 simulation VMs). Where, the training model produces optimal simulation parameters of MIPs = 1430 and bandwidth = 50 Mb/s.

Also, one datacentre has been created, where the data transfer time is already considered in the task execution time and extracted BW value can be adopted within each case training set. We will tune setting VM within three values such as 12 for the training/testing data set, 24 or 48 for the other testing sets.

5.5.2 Experiment Training set and Input Scalable

Minimising the number of training data sets and achieving better prediction is an important goal. For the NGS pipeline workflow, collecting more data points can help to extract the better prediction equations, but has time and cost associated with the data collection. For this reason, only the training data set for 6-sample data points is considered; this gives the impression that the framework is applied for predicting the large sample numbers.

This experiment shows three invocations of 6-sample were used as a training set. We were planned to conduct experiments on the new dataset that consists of different numbers of VMs (12, 24 and 48) driven by the NGS pipeline execution. There are three scenarios: The first creates and configures the training set {6} with given a fixed number of VMs (i.e., 12 VMs), and then testing the NGS pipeline with 12- and 24-sample inputs on 12 VMs. The second uses the same training data set in increasing the order of VMs to test 6-, 12-, and 24-sample on 24 VMs. The third scenario uses the same training data set by increasing the order of VMs to 48 to train the 6-, 12-, and 24-sample inputs again.

Figure 5.3 shows scenario one of an actual and predicted pipeline execution time with different input samples. For this experiment, the small size input sample is the training data for considering a pure prediction of the 12- and 24-sample input. It allows us to verify whether the enhancement can indeed give rise to useful prediction in this scenario that might be used to procure space resources forecast in the cloud.

Using the 6-sample data as the training set for the 12-sample input case gave relative error of approximately 15.0%; using the same training set for the 24-sample input case gave a relative error of approximately 16.0%, see Figure 5.3. Furthermore, comparisons with results of past dataset, which were presented in Chapter 4 for the 12-sample input, clearly shows that time improvement in the prediction rose about around 4.0%; however, it real time underestimates in large samples such as a 24-sample set. Obviously, there needs to be improvement for the runtime prediction of the pipeline supply through improved, collected information from the past.

The next experiment demonstrates how the prediction accuracy changes when sample size and number of VMs are varied.

5.5.3 Prediction at Different VMs

In the previous section, we estimated different samples of the pipeline and compared to actual runtime with predicted. In reality, the prediction may be used with some large samples to predict the pipeline execution time for the number of VMs for which there were no actual pipeline runs in the past. Such prediction may be important

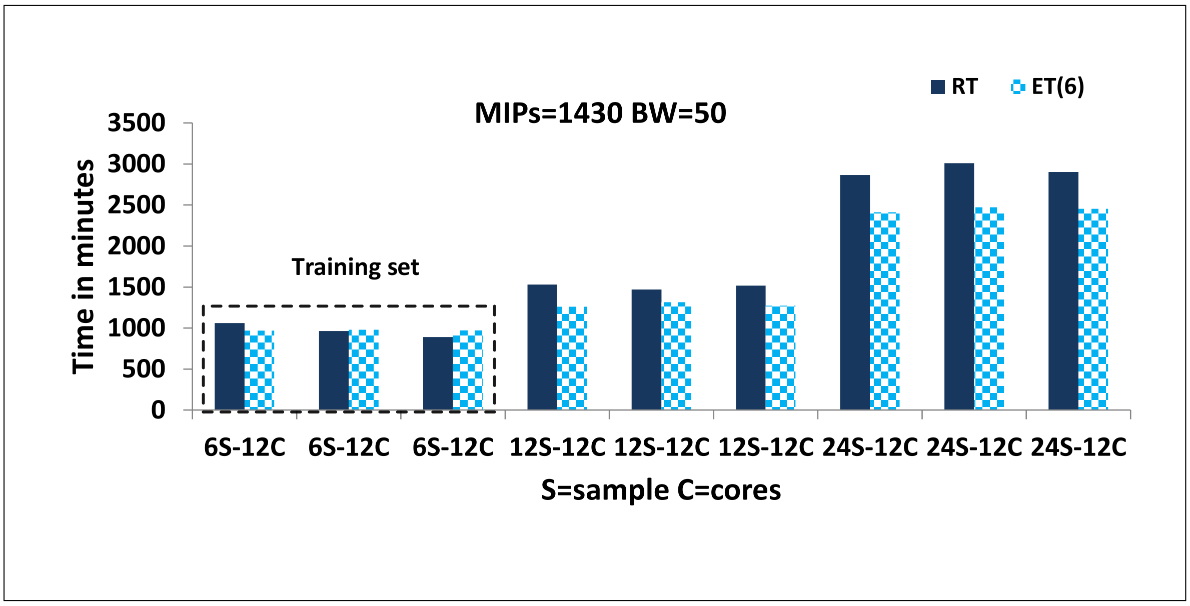


Figure 5.3: The Runtime Results Scenario 1; where (RT) is Real Time and (ET(6)) is Estimated Time based on 6-Sample training set.

for determining, in a deployment-efficient manner the number of resources required to execute the pipeline.

The accuracy of the estimation is evaluated across different sample numbers of all implementation status' of the pipeline and for a different number of VMs. Over the evaluation, each test of predicted execution times was implemented by simulating a pipeline with the specific considerations as follows:

1) The number of input samples; 2) The number of VMs; 3) The actual execution time for which the pipeline runs on the Azure cloud were implemented, in addition to relying on the training set configuration to set necessary variables for the framework. The results are shown in Figure 5.4.

Figure 5.4 shows scenarios 2 and 3 indicating significant result in accuracy by using only one evaluation point(6-sample input over 12 VMs) as the training set. The framework was able of predicting much larger experiments with relative error as small as 5% for a 12-sample set on 24 VMs and a larger 22% error for 24-sample set on 24 VMs. Both situations underestimated real time. The largest error we observed was from the sample running on 48 VM, where the evaluation point was the most distance from the training set. Also to be noted in the graph, there are overestimation on the real time for the 6-sample set on 24C and 6-sample set on 48C. The 12-sample set on

48C was overestimated because the sample number used was small while using a large number of resources in the execution.

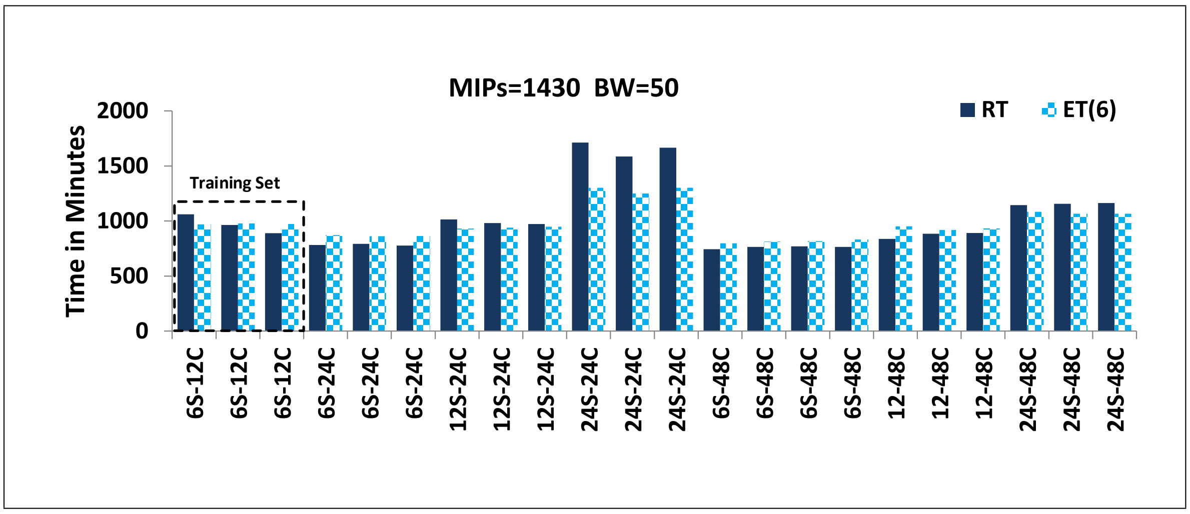


Figure 5.4: The Runtime Results Scenarios 2 and 3; where (RT) is Real Time and (ET(6)) is Estimated Time based on 6-Sample training set.

5.5.4 Prediction at Particularly Point

As a proof of simulating a pipeline, particularly and also to provide the framework flexibility, we experiment with varying only on pipeline broken situation. We chose real broken executions because they have the deficiency in completion, pipeline execution time, and to present results in a real situation. Figure 5.5 shows: First, the difference between the pipeline execution time to VCF missing block.

As proof of simulating a pipeline and to demonstrate framework facility, broken pipeline was tested. A real broken execution was chosen because it is an empirical case that can be measured for what applied solution is faster. This allows different measures of time to be taken on solutions to the broken pipeline using the framework. Figure 5.5 shows: the difference between the pipeline execution time to the VCF missing block. The training set is also given. From this, it is evident that the 6S-12C-VCF application achieves better time results. The difference between the real time (RT) and estimated time (ET) is also smaller in the 6S-12C-VCF case. In effect, this demonstrates the best results in the scenarios, whereas sets requiring greater resources such as 24S-24C-VCF show greater gap between ET and RT. Additionally, these cases require noticeably more time.

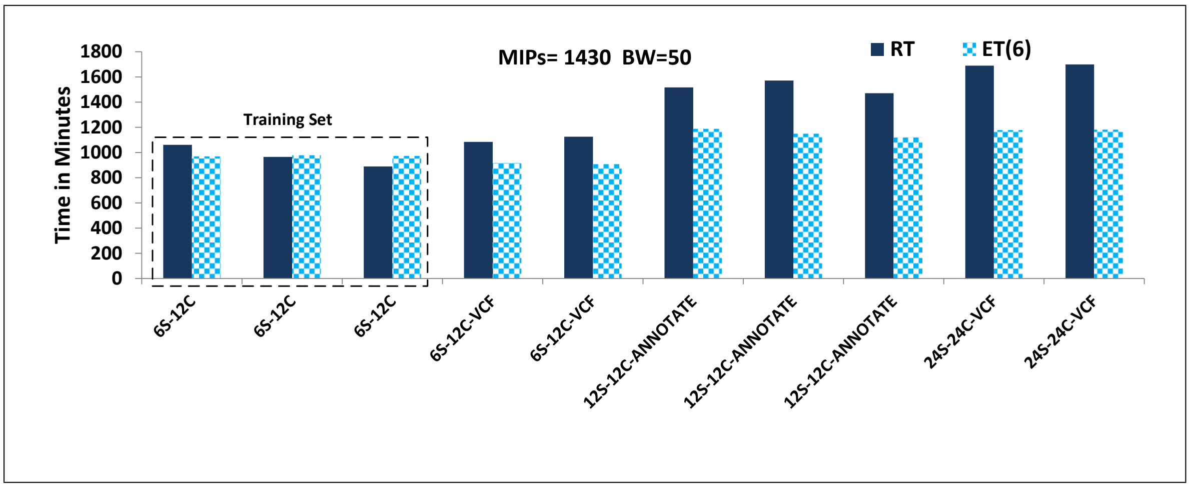


Figure 5.5: The Runtime Results at Broken points; where (RT) is Real Time and (ET(6)) is Estimated Time based on 6-Sample training set.

5.6 Conclusion

Predicting performance behavior is critical for scientists to enable the use of a framework that creates an effective NGS pipeline workflow, where it allows better decision-making in deployment. Therefore, an improvement of the prediction methodology is needed, while the framework also needs to predict the resource number. In this chapter, the new dataset was collected by running the NGS pipeline on the Azure cloud, where the existence of such a dataset was necessary for improving the prediction and predictability of resources.

Through a real implementation a broken NGS execution in some blocks is evident, which motivated pipeline partitioning in the simulation as it has benefits that are mentioned in this chapter. Some of the experiments were conducted with an extracted dataset that performs validation checks of the enhancement. Results of the resources prediction experiments for six testing data sets are used to predict the outside training set; these show that the framework's prediction is under-estimated for real execution times; the largest error was observed for the sample running on 48 VMs because of the evaluation point was most distant from the training set. Therefore, to make good predictive accuracy, we have to investigate a problem by looking for simulation environment and give further consideration to the impact of parameters which most affect runtime prediction. We will illustrate that in the subsequent chapter.

6

Enhanced I/O Shared Storage for Framework

Contents

Summary

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 the large-scale and long runtime of those processes. In the previous chapter, we acquired a new dataset to improve the accuracy of predicting runtime of the pipeline. Obviously, the emphasis was focused towards improving the data that was used to construct the prediction unit, but it is crucial that we now focus on developing the operational tools. To achieve a major contribution of the simulation methodology is that it provides a reasonable prediction of runtime for testing data much larger than the training inputs.

Therefore, in order to improve the effectiveness of estimation of framework, we present now an extension of framework 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.

6.1 Introduction

Big Data simulation integrates existing simulation tools and performance characteristics of Big Data workflows. The need for such integration is driven by the rise of Big Data analytics in the cloud [68] as an answer to tackle increasingly large amounts of data generated every day. One of the main advantages of accurate simulation of Big Data workflows in the cloud is time and cost saving. Whilst tuning and optimisation of the system configuration to match the requirements of the workflow has always been difficult and time-consuming, considering pipeline workflow deployed in the cloud, this process can become very costly.

In the previous chapter, the framework has revealed to be a powerful tool to deal with the challenges associated with pipeline workflow deployment. This solution is easy and free of charge to monitor, expand, and optimise the pipeline workflow in term of performance. However, the basic design of the framework was following a hierarchical schema that includes modules, basic systems of the Pegasus and does not consider the complexity of the e-SC system. Therefore, to further improve the accuracy of the prediction we focus on developing the framework environment via presenting an extension to the shared storage of the framework components that can model the I/O contention. All of what has been stated in the paper [105] applies to developing the framework and presents in this chapter.

Since, this approach to predict runtime performance of the NGS pipeline. 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 [27].

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 chapter 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.

On the other hand, as a significant requirement of the pipeline running it is resolving block dependencies automatically, and the engines can do that. However, each workflow contains software/ libraries dependencies that must be deployed them. Therefore, before running a workflow, any unavailable libraries should be downloaded from the server on demand. This process requires time, which will be added to the whole execution time of the pipeline. The case to complete the pipeline workflow simulation in the framework, we considered a library deployment time in this chapter.

The main contributions of this chapter 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.
- a simulation of the library deployment time is demonstrated and to be considered as a case in the methodology.

The remainder of this chapter is structured as follows: Section 6.2 identifies the problem through interpreting the experimental results to address the solution to improve the accuracy. Section 6.3 will describe the I/O simulation model by focusing on the shared storage and includes: a description of the pipeline I/O, enhanced I/O model, and extending the simulation framework. In Section 6.4 the pipeline library deployment time will be demonstrated and considered in the prediction time of the entire execution. Then the experiments and the results are shown in Section 6.5 as well the correctness of the model implementation has been proofed in this section. Finally, the section 6.6 has concluded this chapter.

6.2 Problem Formulation

As discussed earlier, in the previous chapter the method for predicting the runtime performance of complex data-intensive workflows was proposed. 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; Figure 6.1 shows results for a training set consisting of 6-samples.

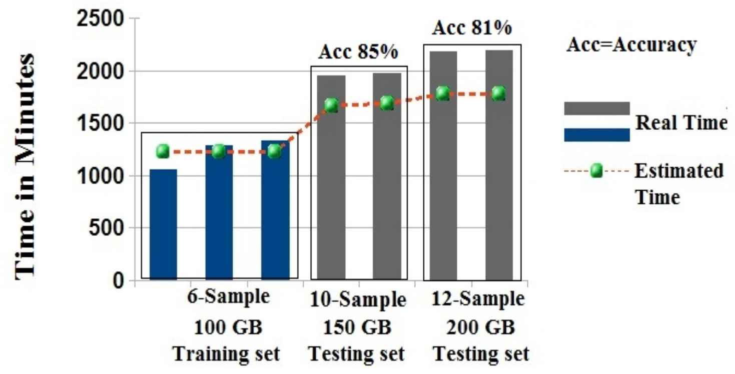


Figure 6.1: The real and estimated time for different sizes of the training set.

Moreover, considering the results of the experiment in chapter 5, when we used a 6-sample training data set as case 24-sample over 24 VMs our method was able to predict with relative error of about 22%. See Figure 5.4. In all cases our predictor underestimated the real execution time. The bigger the gap between the training and testing set, the larger 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.

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.

On the other hand, the results have been obtained in chapter 5 showed that the contention exists through actual execution of the pipeline in the cloud as a 6-sample input on 12 cores was completed with an average 16 hours. But, when executing the pipeline with the same 6-sample input on 24 cores was completed with an average 13 hours. The actual execution time for the second case (on 24 cores) was about 19% less than the first execution (on 12 cores) for the same input samples, whereas the cores were increased up to $2x$ times, and it had made little improvement in the runtime due to the I/O shared storage bottlenecks, see Figure 5.4. This contention is introduced by the blocks that require a data wherein all the engines from multiple workflows that are run simultaneously need to read the input data from and write the outputs to the same shared storage.

There are some approaches for modelling I/O contention such as prioritizing VM access to the storage resource, using many shared storages, multiple queues for ordering the read/write request and serialised storage with a single queue. We have chosen a simple serialised storage model based on a single FIFO queue because this component can accurately simulate the parallel blob storage in Azure stems by the fact that the actual NGS pipeline implementation is very synchronous in nature and due to the large amounts of data saturates the bandwidth available for a single cloud storage account. Thus, although each pipeline step runs many sub-workflows in parallel across multiple VMs, before moving to the following step the pipeline waits until all sub-workflows complete. These synchronisation points at each step make the serialised and parallel access to storage consume a similar amount of time. In fact, for this reasons, we considered the simple serialised model which is perfectly appropriate to our case but for other approaches might be considered as future work with different cases.

Therefore, to address the problem of I/O contention that can be arisen from multiple VMs accessing the shared storage the required I/O model to extent WorkflowSim is proposed. To the best of our knowledge a simulation of the I/O contention that arises from multiple request to the shared storage is novel and needs to be addressed, as outlined in Chapter 2 the existing research in the cloud simulation falls short in treatment this problem.

6.3 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 Figure 6.2, 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.

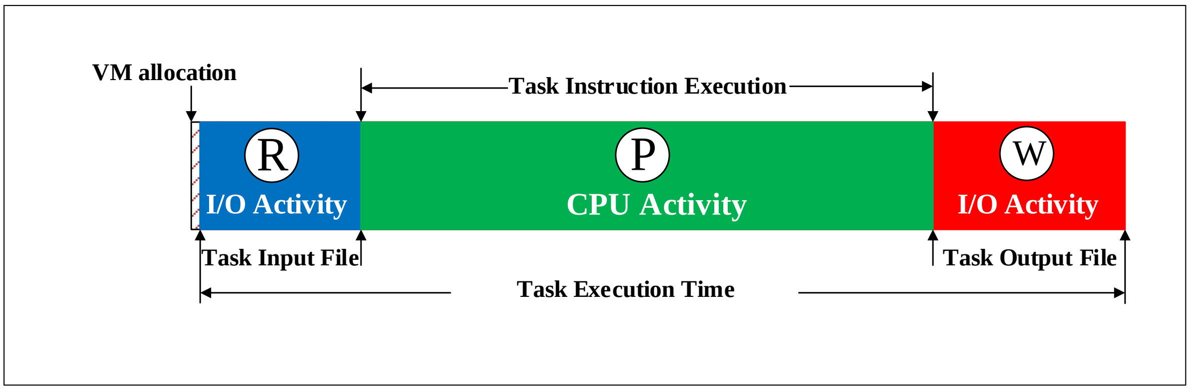


Figure 6.2: Time task model before I/O response time considered

To model the data transfer speed between task VMs and shared storage WorkflowSim uses single parameter – network *bandwidth*. By changing bandwidth 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.

6.3.1 Pipeline I/O

In the pipeline, the workflows have utility to import and export files that means to transfer data from/to the shared data storage, for example, Blob storage is one service of Azure storage service to store the files externally [31]. The data in the shared storage is characterised by sharing and for the common use of the workflows. However, the pipeline is more massive of the blocks that can be allocated to many engines and all concurrent workflow use the I/O storage. Therefore, increasing the engines is the main reason for increasing response time in the shared storage.

There are some important differences between both systems the e-SC and the Pegasus. 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 [27].

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.

6.3.2 Enhanced I/O Model

To simulate I/O contention we propose a simple serialised storage model based on a single FIFO queue (Figure 6.3). 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.

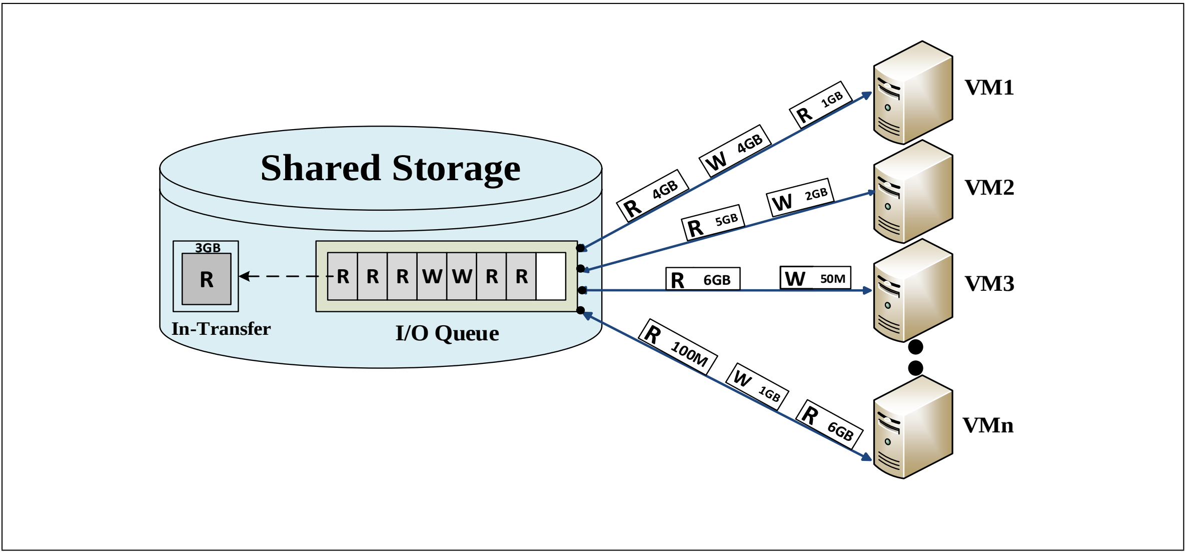


Figure 6.3: The VMs uses single queue to the shared storage.

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:

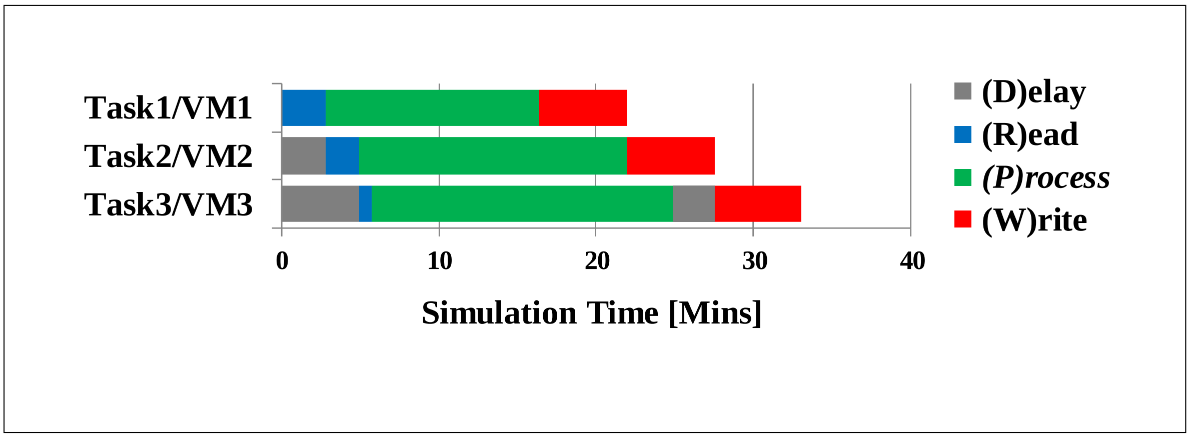


Figure 6.4: Gantt chart of three tasks with I/O read and write operations.

$$T_n = \sum_{i=1}^{n} t_i + r_0 \tag{6.1}$$

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:

$$t_i = \frac{data\_size}{bandwidth} \tag{6.2}$$

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 Figure 6.4. 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. Coincidentally, 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.

6.3.3 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. Figure 6.5 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.

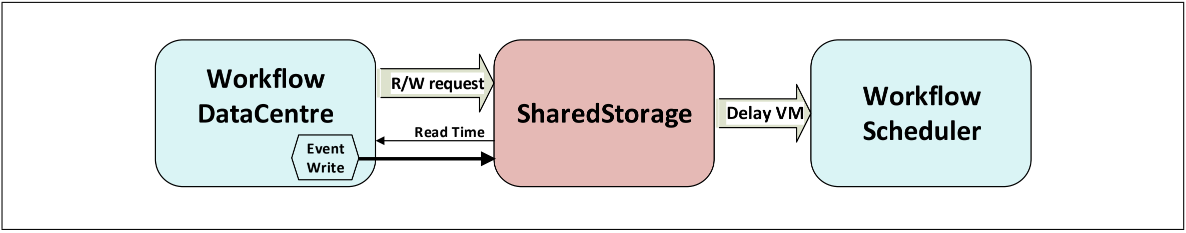


Figure 6.5: Relational Diagram of Shared Storage Model

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 Algorithm 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. 6.2);
9  QueueTime := QueueTime + FTT;
10 cl.setIOEndTime := QueueTime;
```

6.4 Pipeline Execution Model- Library Deployment

As discussed earlier in Chapter 4, e-SC has depended on a DAG to represent a data flow model of the NGS pipeline since the pipeline is built as a set of connected blocks. The edges describe the flow of data, and the vertices present the blocks see Figure 6.6. There are multiple input and output ports in the blocks, multiple input properties and an output status [28].

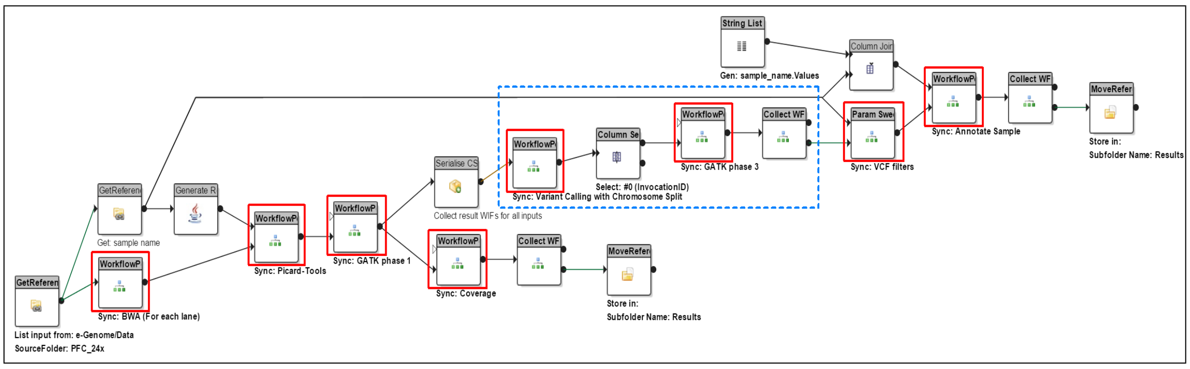


Figure 6.6: The NGS pipeline as Blocks.

In general, a pipeline is made up of a composition of configurable library packages and tools that implement genome analysis algorithms, and which are expressed as blocks. It can be classified into five different types: Java, R, Octave, GnuPlot and JavaScript, each has a specific execution environment. For example, R blocks execute the code that is written directly in R.

The dispatching operation can be summarised as that the pipeline invocation is sent to a single message queue from which the engines acquire it. This work-stealing approach fits better with the cloud platform than explicit task scheduling may be for two reasons. Firstly, during the operation, the worker nodes may be restarted or taken offline. It may be caused by the Azure self-heading service or automatic upgrade of the OS. Secondly, there is a facility for adding or removing a node from the resource pool during the execution. Therefore, there is no need for rescheduling tasks when the pool size changes [28].

As a part of the smoothing-running of the pipeline that lies with the engines are capable of resolving block dependencies automatically; however, a block contains software dependencies that must be found to start it, in addition to the declaration of input ports that needs to run. For example, the number of workflows in the pipeline need to access the human genome (HG19 from UCSC) as a shared library, and this request is defined in the block descriptor as a library dependency. Therefore, before running a workflow (just for the first run of the pipeline), any unavailable libraries are downloaded from the server on demand. Once all software dependencies as the operational conditions are fulfilled, the engines start execution the pipeline. This process needs time, which is added to the pipeline execution time.

6.4.1 The New Dataset- aware Library Deployment Time

In general, for reviewing the deployment library time to ensure that most cases in the pipeline can be considered and our Framework is accurately estimated. Thus, the simulation methodology to predict the runtime of the NGS pipeline with library deployment time needs to be applied. However, the previously collected dataset has not considered a library deployment time for each NGS running on the cloud. Therefore, to deal with such circumstance a new dataset is required; we decided to collect a new dataset by running the NGS pipeline on the Azure cloud which exclusively aimed to consider library deployment time and must be taken fully into account for each execution. When the library deployment time is included in building a training set must be matched by the testing sets that have to encompass the library deployment time. The total runs of the NGS dataset are contained as 6-, 12-, and 24-sample inputs, which were executed over 12, 24, and 48 cores. Table 6.1 shows the runs on the NGS pipeline.

Table 6.1: The actual running of the NGS on Azure by considering library deployment

The Cores Number	**12 Cores**		**24 Cores**		**48 Cores**	
Input Samples	**Success**	**Failed**	**Success**	**Failed**	**Success**	**Failed**
6-Sample	3	1	2	0	2	0
12-Sample	2	0	2	1	2	0
24-Sample	2	0	2	0	1	0
Totally 20 times have been executed NGS pipeline- with library deployment time The number refers to replicate running (Number of times)						

6.4.2 The problem with Library Deployment Time

In an analysis of the new dataset to extract library deployment time, there was an unexplained delay time in the NGS pipeline blocks; this time gap has happened between the *End–Time* of the prepare library block and the *Start–Time* of the next block (BWA_Mem). This time has appeared with BWA (Align lane) workflows in the pipeline across all executions of the pipeline with deployment library that listed in the Table 6.1. We were not able to determine an exact cause of this delay time, it seems such as a technical bug in the pipeline.

However, this time gap was ranging between 1-5 hours, it is always changing and do not have a similar pattern in the engine or cannot be fixed time for each block. Therefore, this issue caused to generate the small values R-squared less than 1% for both BWA (Align lane) and BWA (For each lane) workflows when there was an intention to extract the prediction equations which are required for *estimation module*. For example, the prediction equation of the BWA (For each lane) was plotted the regression line with low R^2= 0.0837 and the prediction equation of the BWA (Align lane) was plotted the regression line with low R^2=0.0182.

The Suggested Solutions

• Working on the same information that extracted from the dataset even there is a time gap, i.e. with the bugs we will produce an inaccurate estimation.

• Removing the time gap that appeared as bugs from the extracted time for each observation and then applying a regression model with a larger R^2, for example, after removing the time gap we were obtained R^2=0.8175.

• Considering this issue as resource properties, therefore we need to apply a new model that takes into account a waiting time for ordering the library storage requesting, where this solution can be considered as future work.

6.5 Evaluation

To evaluate the proposed shared storage extension we conducted two experiments. First involved the Pegasus Montage workflow [52] 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.

6.5.1 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 Figure 6.4. 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 [50]. 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 Figure 6.8.

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 Figure 6.7, 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.

6.5.2 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 using the same dataset in Chapter 5. As mentioned before,

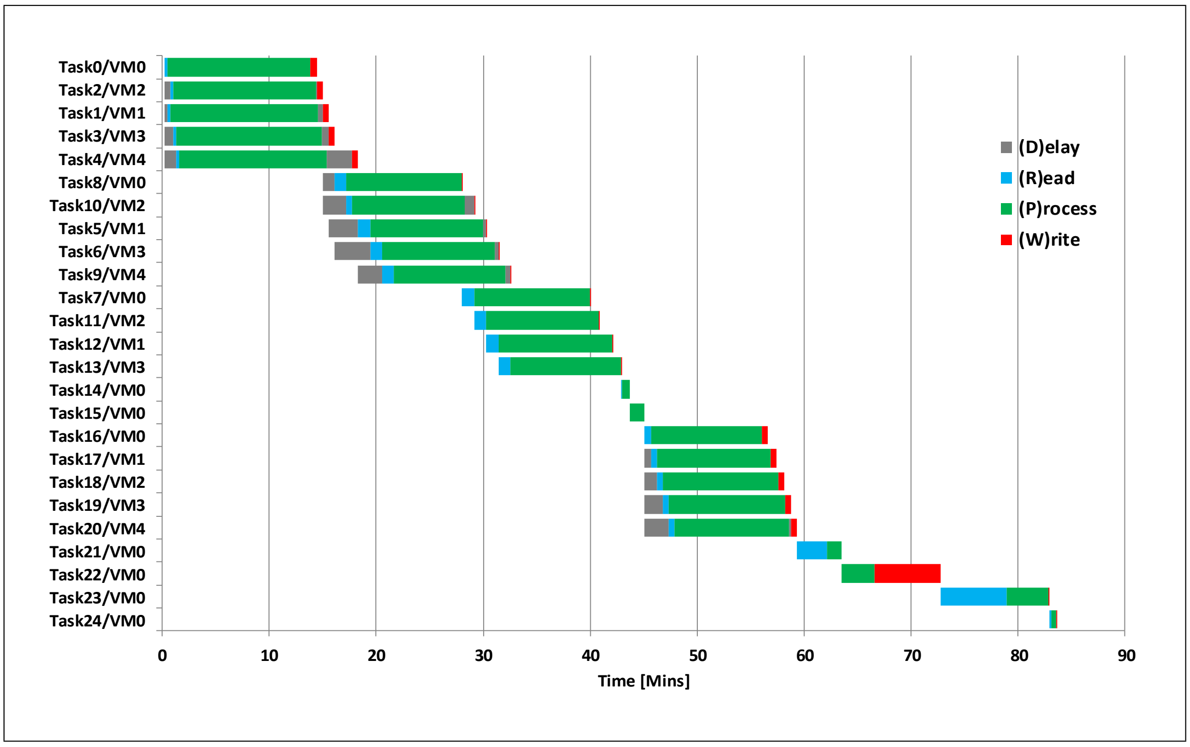


Figure 6.7: The Gantt chart of the Montage workflow.

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.

6.5.2.1 Experiment setup.

The dataset has resulted in candidate configurations of the D13 VM with 8-core CPU, 56 GiB of RAM and 400 GB of local SSD storage that was used to run four workflows concurrently as execution threads. The different deployment of the pipeline in e-SC executed in the Azure cloud over 12, 24 and 48 as the VMs' simulation. 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 with the shared storage component. Training the model gave us the optimal simulation parameters of: MIPs = 1305 and bandwidth = 985 Mb/s in simulation with I/O contention. Apparently, the value of 985 Mb/s $\times$ 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. But the optimal simulation parameters that predicted in Chapter 5 by training were different, MIPs= 1430 and bandwidth = 50 Mb/s because there was no I/O contention consideration.

As outlined 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 $8 \times N + 53$ tasks, where N is the number of input samples (see Figure 4.4). For example, in the biggest setting our WorkflowSim simulation included 245 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.

6.5.2.2 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

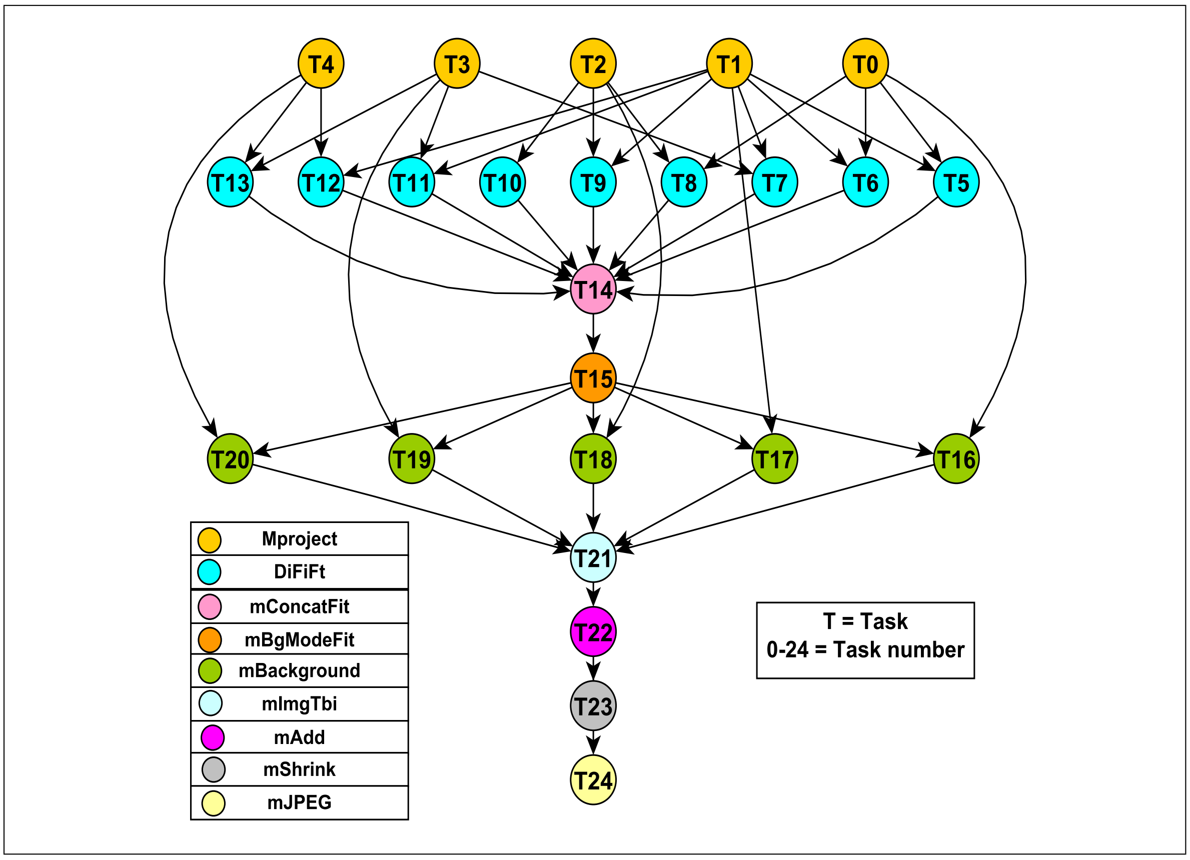


Figure 6.8: The structure of the 25-task Montage workflow.

execution times (RT) with estimated time simulated by the framework with shared storage as in (ET+SS) and comparing it with estimated time simulated in Chapter 5 without shared storage as in (ET-CH5).The results are shown in Figures 6.9, 6.10, 6.11.

The graphs indicate 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 Figure 6.1 in Figure 6.9). There is a difference in the accuracy of about 4% if compared with Figure 6.1 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.

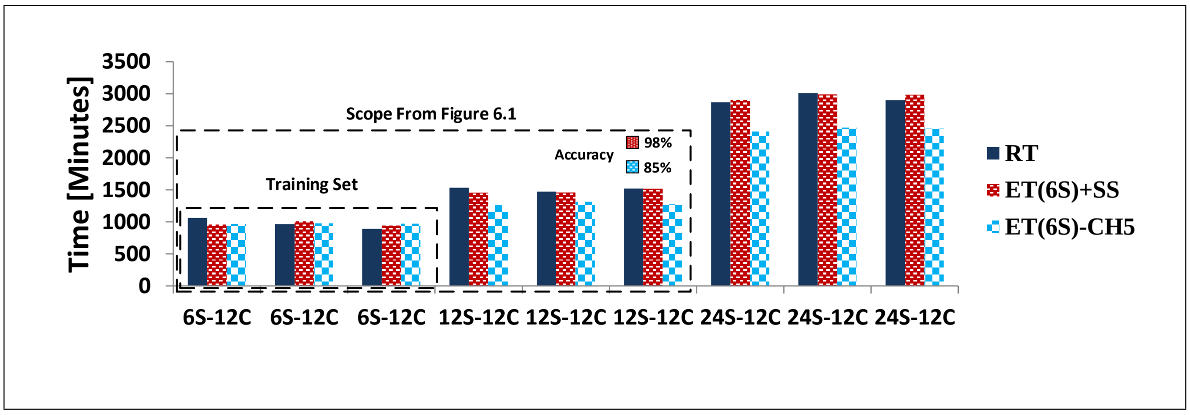


Figure 6.9: The Runtime Results with I/O Compared without I/O as in Chapter 5- Testing over 12 VMs; where (RT) is Real Time and (ET(6S)+SS) is Estimated Time with Shared Storage based on (6S) 6-Sample training set and (ET(6S)-CH5) is Estimated Time as resulted in Chapter 5

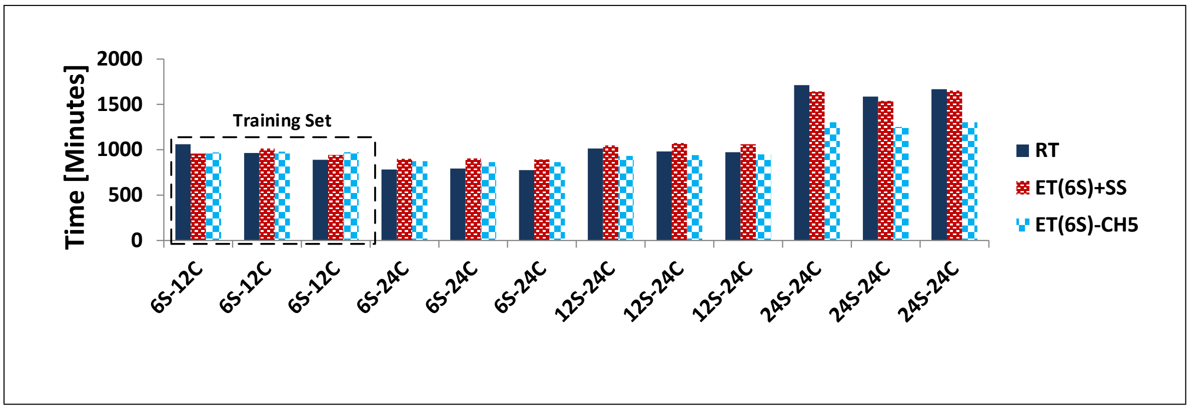


Figure 6.10: The Runtime Results with I/O Compared without I/O as in Chapter 5- Testing over 24 VMs; where (RT) is Real Time and (ET(6S)+SS) is Estimated Time with Shared Storage based on (6S) 6-Sample training set and (ET(6S)-CH5) is Estimated Time as resulted in Chapter 5.

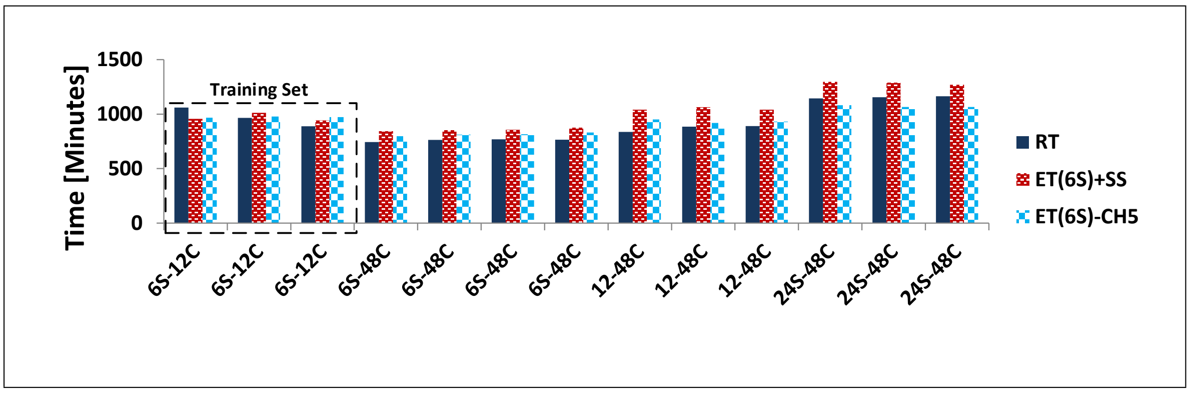


Figure 6.11: The Runtime Results with I/O Compared without I/O as in Chapter 5- Testing over 48 VMs; where (RT) is Real Time and (ET(6S)+SS) is Estimated Time with Shared Storage based on (6S) 6-Sample training set and (ET(6S)-CH5) is Estimated Time as resulted in Chapter 5.

In order to compare the NGS pipeline execution in the Azure cloud with that in our Framework simulation, so from Figure 6.12 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.

Figure 6.13 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, Figure 6.14 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:

$$RPEs(b,n) = \frac{bTs(b)}{nTs(n)} \tag{6.3}$$

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, $\text{RPE}_{12}(12,\ 24) = 70\%$, 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 $2 times$ 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 Figure 6.14).

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.

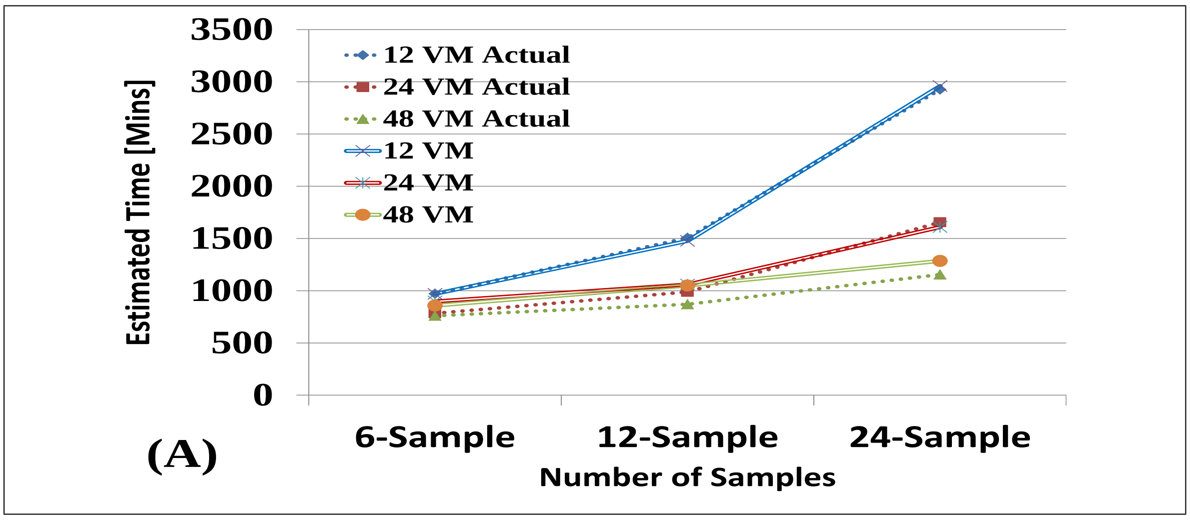


Figure 6.12: Estimated time and behavior of the pipeline with the increasing number of input samples and different number of workflow engines in the system.

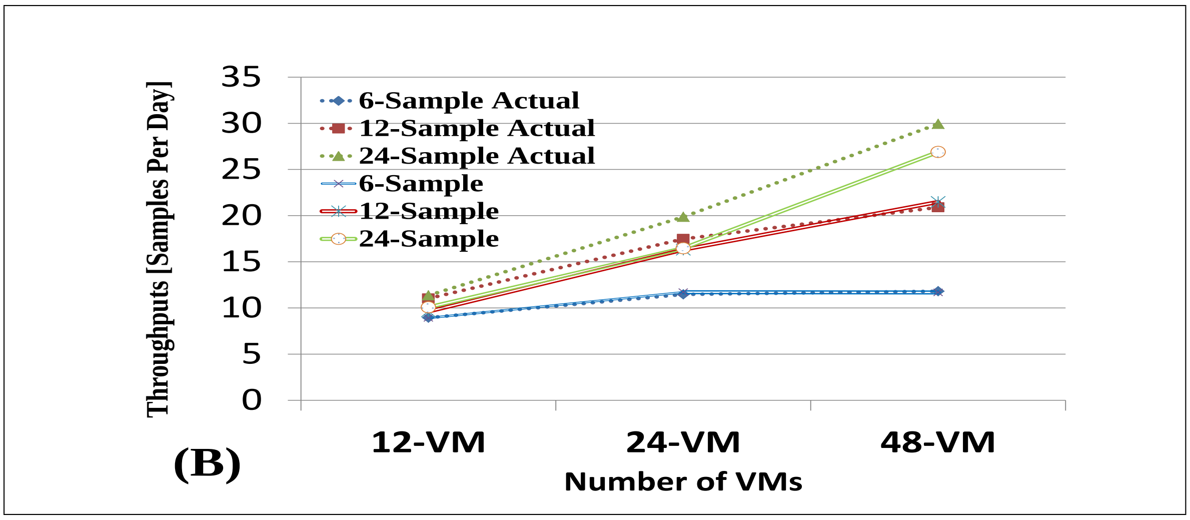


Figure 6.13: Throughput of the pipeline with the increasing number of input samples and different number of workflow engines in the system.

6.5.3 Evaluating and Analysis of Error Bars

Once the methodology is fitted for predicting runtime of the data-intensive workflows, we assess it against the testing sets for much larger input data sizes, according to a

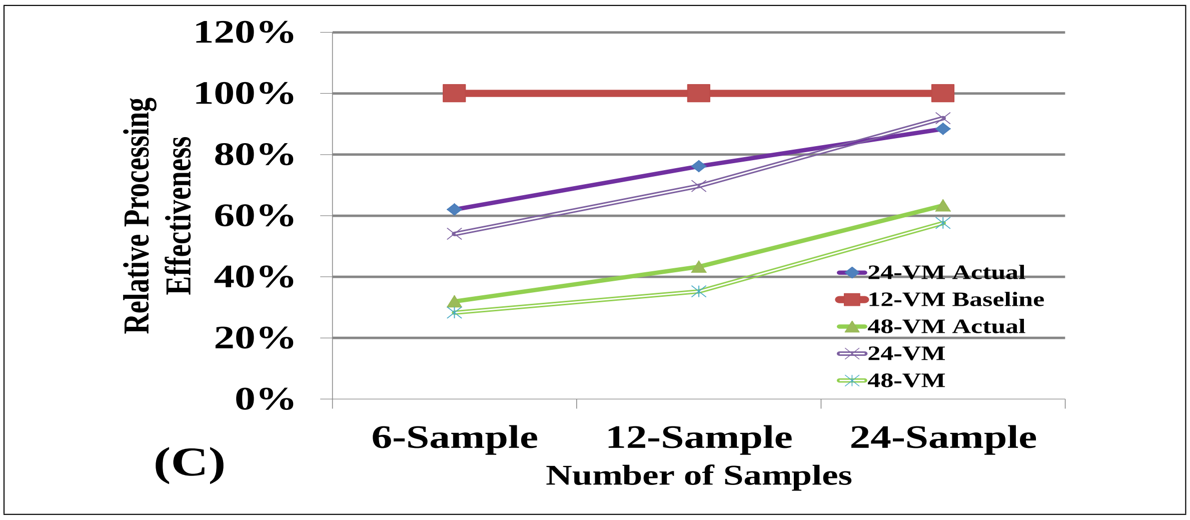


Figure 6.14: The relative processing effectiveness of the pipeline with the increasing number of input samples and different number of workflow engines in the system.

specific number of resources (See Figure 6.15), with prediction runtime $X_1..X_{28}$ and measured actual time $Y_1..Y_{28}$. Hence, the error bars can be revealed for each of these actual runtimes as a probabilistic prediction time that is, a mean of actual time f(Y_i) and standard deviation STD f(Y_i). This is a way to show the error bars which help to indicate estimated error or uncertainty to give a general overview of how precise a predicted runtime is. We evaluate the error bars by counting the standard error of the mean (SEM) for each chunk, where the chunk is three or four values of the actual time that falls into the bars against of the estimated points might fall into the actual bars confidence. This analysis of the predicted runtime is shown in Figure 6.15.

For actual performance, the figure shows the predictions are consistent with the expectation because the estimated time is inside the error range of the measured time. For example, the predicted runtimes for 6-, 12-, and 24-sample over 12 cores, which were enough to guarantee a confidence level. For the cases when the predictions are outside the error bars that means the prediction is less reliable as an estimate of actual time. For example, the predicted runtimes for the 6- and 12-sample over 12 cores and 6-, 12- and 24-sample over 48 cores.

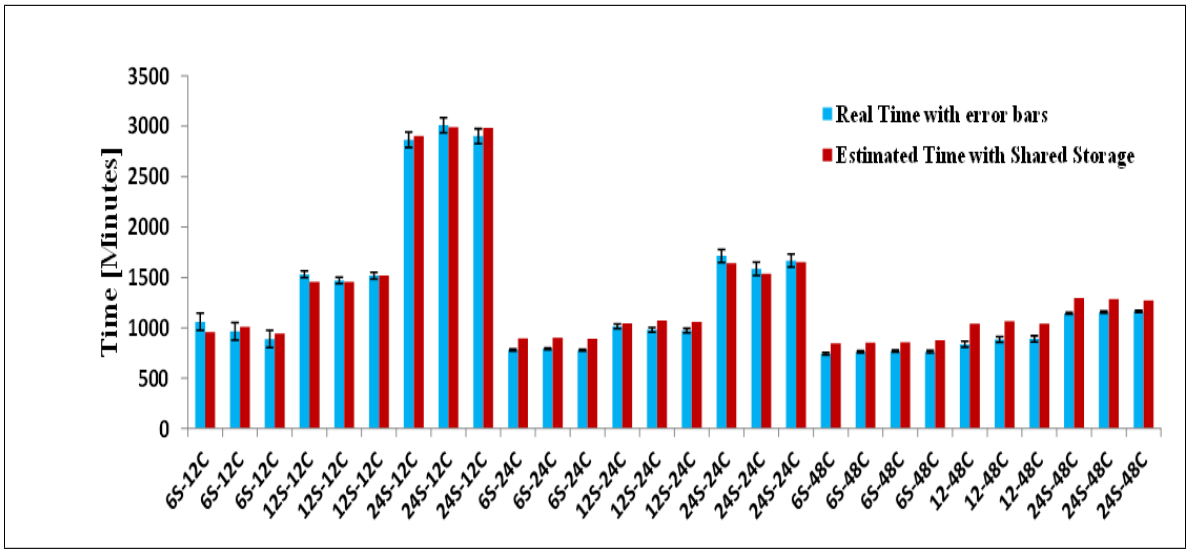


Figure 6.15: Predicted runtime versus measured actual runtime with presenting the error bars for different input data sizes ranged as (6-, 12-, and 24-sample) over various numbers of the cores ranged as (12, 24 and 48cores).

6.6 Conclusion

Cloud simulation remains an essential tool for providing decision makers with guidance to implement applications into the cloud. In this chapter, we focused on the development of deep in the framework components to simulate the I/O contention problem.

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 chapter 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 [30]. 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 the previous work [27], 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. The source code has been released as open source code and can be downloaded from [2].

7 Conclusion

Contents

Summary

This chapter summarises the research work presented in this thesis. The following section concludes the novel prediction technique we developed for estimating the runtime and resource numbers of the data-intensive workflows, by emphasising the main functions for each chapter. The last section describes future research avenues that are worth further exploration.

7.1 Thesis Summary

This thesis concentrated on building a framework dedicated to the common goal which is the performance prediction of complex data-intensive workflows that utilised in the cloud. This work explores how the prediction technique has used many real cases in the simulation tool to support an enhancement for prediction accuracy and the framework development. A new performance estimation technique was proposed and evaluated with actual data-intensive workflows.

Chapter 2 gave background information concerning the main topic of this thesis, including Cloud computing, Big-Data application in the cloud, performance prediction of cloud applications, and Cloud computing simulation. More details are given on scientific workflows, which represent the pipeline workflow and their life-cycle. In addition, a method for capturing and recording the history of the workflow execution is represented in the provenance section. Furthermore, the requirements and challenges for Big-Data workflows were outlined regarding scalability, flexibility and variability in challenges such as platform heterogeneity and resource selection. There is scarcity of research on complex, data-intensive workflows that measure performance prediction concerning required resources at deployment time. However, most existing techniques and models had limitations and do not consider the complexity of the deployment of Big-Data workflow workflows. These challenges were addressed by proposing and developing new techniques to model and simulate the behaviour of this type of workflows and effectively support the deployment decision by estimating the runtime and resource numbers.

Chapter 3 introduced the simulation tools selected to build the framework and a methodology was described largely in general terms. Additionally, the adaptation steps of the selected simulators are outlined. The method to convert a workflow from one system to another system to be accepted under some template was presented. The method can extract the framework parameters and predict execution times, input data size and output data size is showed. Because the methodology based on these parameters' values for filling out the synthetic workflow with data which are required by WorkflowSim. Steps have been shown for the methodology to enable such complex workflows to be simulated and predict their performance. Further, the case study follows a proposed methodology that formed the basis to achieve runtime prediction for deploying the workflows in the Cloud as described in Chapter 4.

Chapter 4 considered the NGS pipeline as the example to apply the thesis' questions and study related issues. In the case study, a near-optimal deployment is found to minimize the execution time and the cost of implementing the pipeline on the cloud. More details about modifying a simulation platform to simulate the behaviour of the NGS pipeline were learned. A methodology for predicting runtime and output size of the pipeline was conducted and integrated with the framework function. The problem addressed in this chapter focused on two things: 1) how to configure a training set from a small input sample, and 2) how to build a *estimate module* that predicts runtime and output data sizes as a function of all blocks in the pipeline. For the cases, the data collected during a prior run of the pipeline are insufficient as well as cases that estimate the runtime is inaccurate due to less data overlap between the training set and the testing set. The experiments showed the prediction of the framework is under-estimated for the real execution time and further improvements are needed.

Chapter 5 intended to develop the framework by enriching the existing prediction methodology and tool components with information that enables scientist to acquire better decision-making in deployment. Collecting a new dataset by running the NGS pipeline on the Azure cloud was applied, where having such prepared new data sets helped improve the prediction and predictability of the VM numbers. Additionally, the pipeline execution that was broken at some points motivated a consideration of the pipeline workflow partitioning as a case to be simulated. This can be used for research purposes of the data-intensive workflows such as detecting a fault. The experiments were conducted with a new dataset; the validation of the enhancement was checked. Results were produced after improvement showed the prediction of the framework was under-estimated for real execution time. The largest error was shown for the sample running on large numbers of VMs. Therefore, there has been a trend towards developing a framework tool after detecting the problem of I/O contention, which has not been considered in the WorkflowSim. Chapter 6 takes responsibility to cover and solve this issue.

Chapter 6 explained one of the enhancement works to improve the runtime prediction of the pipeline through the development of the framework. The proposed simple storage component is added to the framework that has introduced an improvement with good accuracy of the estimation. We proved the correctness of a new component by testing the Montage workflow and the Gantt chart showed the storage component is based on a single FCFS queue to save the request read/write and handle them correctly. Finally, a very small training set consisting of three measurements of the smallest execution were able to estimate the performance.

7.1.1 Contributions of the Performance Simulation of Complex Workflows

The collective result of the work in this thesis used and developed technology for the existing simulations. The main innovative features of the new prediction technique is summarised as follows: 1) rely on minimal information stored in the provenance to generate a training set and extract the simulation environments. 2) Devise a method to simulate the behaviour of big-data workflows which are from disparate systems; there is difficulty when one wants to analyse them on real systems. 3) Fair runtime prediction can be obtained from a framework for running complex Big-Data workflows with scaling-up their input data size that can used to define sub-optimal deployment. 4) It will further help promote the predictability of the number of resources that can be determined before their deployment. 5) It introduces a generic methodology that applies to any workflow of parallel and/or serial processes when one wants to use simulation tools for prediction.

One important contribution of our framework was to accurately predict for large datasets using only a small dataset to train and simulate the NGS pipeline workflow. In order to prove that, we have conducted an experiment which used a small data set (three invocations) as the training ones to generate and collect information about the simulation environment. The derived parameters are then used to predict the runtime for new large data sets. Therefore, there was a sensitivity analysis on different data set sizes and resource numbers. Our framework demonstrates that there is flexibility for the user to choose a better configuration for achieving optimal performance before deployment. Our framework can predict performance under different scenarios of configuration and help the user to decide the appropriate resource configuration.

The results of the runtime prediction for the pipeline for the execution of the 6-sample input datasets and 48-sample input datasets are plotted in Figure 7.1. We used 6-Sample datasets, as a training set to generate the parameter' values for configuration the framework. They were MIPs= 995 and BW= 795; two tests have been done, with the first was testing the 48-sample input on 12 cores and the second testing a 48-sample input on 48 cores. It was observed that our framework can accurately predict the execution time even when the input datasets became large, where it accurately predicted about of 94% and 71% respectively, see Figure 7.1. Summarizing the figure, we can see that this new technique consistently achieves predicted results in good agreement with actual ones under different pipeline parameters such as dataset size and number of clusters.

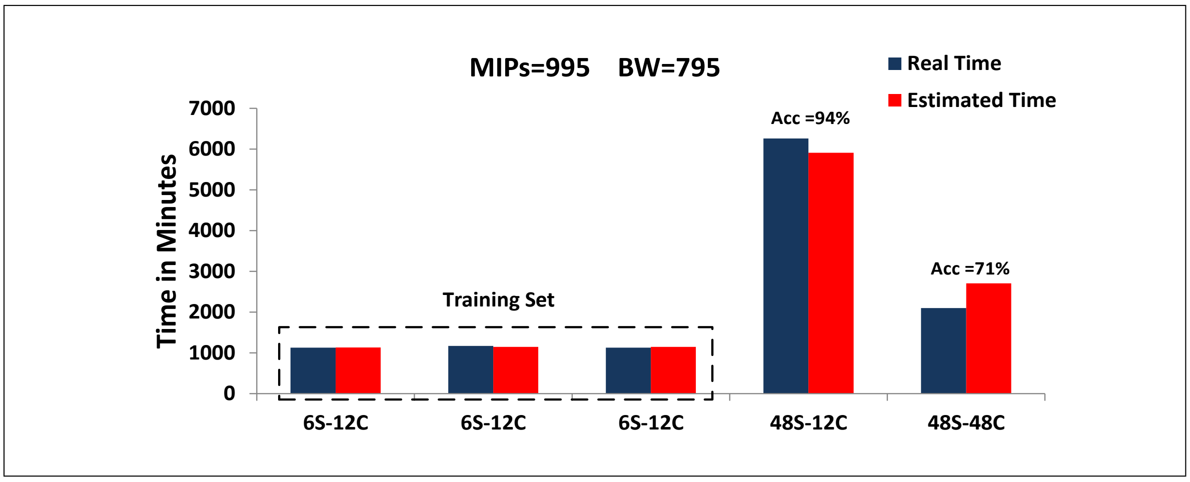


Figure 7.1: The real and estimated time for large input sample.

7.2 Future Research Directions

During the course of this thesis, a number of areas for future work have been identified.

7.2.0.1 Automatic Modelling of Big-Data Workflows into Simulation

One step of the prediction methodology is modelling the pipeline workflow to be accepted by the WorkflowSim as explained in Chapter 3 and 4. Synthetic workflow supports the definition and description of the workflows for the simulator by using templates. While the workflows differ from one type to another; therefore, the automatic technique for modelling various workflows is needed to fill-out the template of the synthetic workflow. The first phase of this technique is centred on the activities in the workflow to be considered as a process/task. The next phase involves dealing with gathering information, which is relevant to the processes/tasks. The third phase focuses on identifying and addressing the needs of input and output files for each process. These phases can be utilised to gather a workflow description that is composed of tasks or even sub-workflows.

7.2.0.2 Method of Capturing Provenance Data and Keeping the Framework Up to Date

In general, the applied prediction method used by framework was crafted for structuring and characterising data-intensive workflow based on provenance data. Although such methodology provides accurate predictions, it requires the supervision of efforts for constructing and tuning the prediction module. Such requirement invalidates one of the leading advantages of data-intensive workflows: simplicity for the user. Therefore, the suggestion to automate a capability for constructing the prediction module in future work is suggested. If such automated methods exist, then it will supply the framework to be accessible from existing provenance data for any provider and support it with prediction equations. Potentially, for many organizations which use cloud solution decisions, it can be easier in relation to prediction of runtime, hiring needed resources and developing an expectation of execution behavior for the complex pipeline workflows based on input data size. Additionally, such benefits may mean that these organisations may not hesitate in using cloud resources.

7.2.0.3 Fault-Tolerance Prediction of Big-Data workflows Support the Framework

The WorkflowSim promises to support an evaluation platform for study areas such as fault tolerance of the Pegasus workflow; we have modelled e-SC workflow; therefore, we can exploit this ability and employ it in developing the framework to predict the fault in the cloud when the NGS pipeline is implemented. However, fault tolerance is a significant issue in such type of the workflows, particularly in the deployment mission because it takes a long time in execution. To address this issue, we must create a failure predictor module in the framework that can estimate failures in the pipeline when it will be implemented in the cloud. The predictor module must be provided with most popular execution failures of the pipeline.

7.2.0.4 Solving the Problem of Congestion

The extension solved the problem of the I/O contention and produced a good estimate, especially for those running with a large number of the data samples and resources. On the other hand, it caused overestimation for the small input sample over small resources due to congestion. One reason for the congestion was using one queue to simulate the I/O contention. Therefore, it was suggested to double the number of queues to tackle congestion, which means modelling more than one queue. In case of using multiple queues, the scheduling algorithm that arrange the VMs' requests are needed. This solution, currently is difficult to apply with WorkflowSim.

7.3 Limitations

1- Within the original dataset, only input information for the framework was extracted and a limited summary of the runtime prediction was obtained by simulating the NGS pipeline. As outlined in Chapter 5, we extended the dataset by including VMs' information in addition to information included to improve prediction equations. This dataset was instrumental in achieving a number benefits. First, it provided greater insight in the prediction enhancement over the datasets. Secondly, it enables focus on the development of the tool to establish consistency between the simulated and empirical cloud environments. Finally, we extended the dataset to consider the deployment libraries' time in additionally predicting to collect a run time for the 48-sample. Therefore, new datasets are always needed due to ongoing developments by cloud providers. However, the pipeline with the 6-sample input was run in 2014 on an Azure cloud, taking 21:31 hours to run. The same run of the same input was also executed on the Azure cloud and took 18:51 hours. Therefore, no consideration to the cloud resources' improvement was considered in our framework.

2- In this thesis, we presented a predictive framework for the NGS pipeline workflow. Predicting the execution time of these types of workflow is particularly difficult as the execution time is highly dependent on the input data size, but for different types, it needs more care to investigate a suitable relationship between the input data and the runtime for building the estimation model.

3- The training set and testing set always were small because we had limited funds to run the NGS pipeline on the Azure cloud. Therefore, we tried to make a balance between the training and testing sets and that was affected on the randomizing the order of running our real experiments.

8
Appendix

Contents

8.1 Appendix -A

```
  :- use_module(library(csv)).

traverse_workflow_invocation_(WfInv, Out) :-
    hdb_store_2:wasAssociatedWith(_, WfInv, _, WfDoc, _),
    once(hdb_store_2:program(WfDoc, WfDocProps)),
    the HDB the call to program must be wrapped in once/1
    get_dict('prov:label', WfDocProps, WfName),
    hdb_store_2:wasPartOf(SubEx, WfInv),
    hdb_store_2:execution(SubEx, St, Et, SubExProps),
    get_dict('prov:type', SubExProps, Types),
    (memberchk("esc:workflowExecution", Types) ->
        traverse_workflow_invocation_(SubEx, Out)
    ;
        hdb_store_2:wasAssociatedWith(_, SubEx, WfEng, _,
            AssocAttrs),
        SubExProps >:< _{ 'prov:label':BlockLabel, 'esc:blockName'
            :BlockName, 'esc:dataConsumed':InData, 'esc:
            dataProduced':OutData },
        AssocAttrs >:< _{ 'esc:ConcurrentServiceRuns' : ConcurSrvs
            },
        %get_dict('prov:label', SubExProps, BlockLabel),
        %get_dict('esc:blockName', SubExProps, BlockName),
        csv_write_stream(Out, [row(WfInv, WfName, BlockName,
            BlockLabel, St, Et, InData, OutData, WfEng, ConcurSrvs)
            ], []),
        fail
    ).

dump_invocation_data(WfInv, OutputFile) :-
    setup_call_cleanup(
        open(OutputFile, write, Out),
        (
            csv_write_stream(Out, [row("Invocation␣Id", "Workflow␣
                Name", "Block␣Name", "Block␣Label", "Start␣Time", "
                End␣Time", "Data␣Consumed", "Data␣Produced", "
                Workflow␣Engine", "Concurrent␣Blocks")], []),
            traverse_workflow_invocation_(WfInv, Out)
        ),
        close(Out)
    ).

dump_toplevel :-
    hdb_store_2:workflow(WfDoc, WfProps),
    get_dict('prov:label', WfProps, "Top-level␣(Yaobo)"),
    hdb_store_2:wasAssociatedWith(_, Ex, _, WfDoc, _),
    split_string(Ex, "/", "/", Tokens),
    last(Tokens, InvId),
    dump_invocations("esc://recomp-esc/invocation/", [InvId]).
dump_invocations(Prefix, InvList) :-
    member(InvId, InvList),
    atomics_to_string(["inv-", InvId, ".csv"], FileName),
    atomics_to_string([Prefix, InvId], ExecId),
    dump_invocation_data(ExecId, FileName).


batch_4(["54039", "56126", "58213", "60301", "60708", "61115", "
   61522", "61857", "62211", "62618", "63025" ]).
batch_5(["63432"]).
batch_6(["63839", "64284", "64788", "65435", "66083", "67210", "
   68338", "68673", "69081", "70857", "69729" ]).
batch_7(["47171", "47578", "47985", "48392", "49129", "49537", "
   50184", "50831", "51479", "52606", "63432" ]).

test_batch(["62192"]).
```

8.2 Appendix-B

NodeId	334414
TYPE	Workflow Run
invocationId	23689
name	Top-level (Yaobo)

Invocation Id	Engine I	Workflow Name	Block No.	Block Name	Start Time	End Time	Elapsed Time	Start-Start Time				
23689		Top-level (Yaobo)	1	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:02				
23689		Top-level (Yaobo)	2	GetReferenceInfo	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01				
23689		Top-level (Yaobo)	3	GenerateRGData	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01				
23689		Top-level (Yaobo)	4	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:05	0:00:02				
23689		Top-level (Yaobo)	125	WorkflowPerInvocation	12/15/2014 13:37	12/15/2014 13:37	0:00:05	0:00:04				
23689		Top-level (Yaobo)	174	WorkflowPerInvocation	12/15/2014 16:38	12/15/2014 16:38	0:00:04	0:00:01				
23689		Top-level (Yaobo)	259	SerializeCSV	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01				
23689		Top-level (Yaobo)	260	WorkflowPerFile	12/16/2014 1:40	12/16/2014 1:40	0:00:03	0:00:02				
23689		Top-level (Yaobo)	718	ColumnSelect	12/16/2014 5:12	12/16/2014 5:12	0:00:01	0:00:01				
23689		Top-level (Yaobo)	719	WorkflowPerInvocation	12/16/2014 5:12	12/16/2014 5:12	0:00:02	0:00:01	TOTAL	22:11:30		1331:30:00
23689		Top-level (Yaobo)	729	CollectWFResults	12/16/2014 5:26	12/16/2014 5:26	0:00:01	0:00:01				
23689		Top-level (Yaobo)	730	ParameterSweep	12/16/2014 5:26	12/16/2014 5:27	0:00:05	0:00:03				
23689		Top-level (Yaobo)	767	WorkflowPerInvocation	12/16/2014 5:29	12/16/2014 5:29	0:00:05	0:00:00				
23689		Top-level (Yaobo)	792	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:02				
23689		Top-level (Yaobo)	793	ColumnJoin	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:01				
23689		Top-level (Yaobo)	794	WorkflowPerInvocation	12/16/2014 5:35	12/16/2014 5:35	0:00:05	0:00:03				
23690		BWA (For each lane)	6	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:02	0:00:01			2416	BWA1_FEL
23690		BWA (For each lane)	8	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:02	0:00:00			68	
23690		BWA (For each lane)	87	CollectWFResults	12/15/2014 10:20	12/15/2014 10:30	0:10:04	0:01:39			15861953049	I
23690		BWA (For each lane)	95	SAMTools-Merge	12/15/2014 10:39	12/15/2014 11:27	0:48:10	0:00:19			15861972195	
23690		BWA (For each lane)	106	SAMTools-Sort	12/15/2014 11:27	12/15/2014 12:26	0:59:34	0:01:35	Wait	2:56:16	11335839402	O
23690		BWA (For each lane)	113	SAMTools-Index	12/15/2014 12:26	12/15/2014 12:30	0:03:18	0:03:18	BWA_A	0:00:04	6242968	
23690		BWA (For each lane)	114	ExportFiles	12/15/2014 12:30	12/15/2014 12:30	0:00:01	0:00:01	BWA_B	2:21:02	???	
23690		BWA (For each lane)	115	ExportFiles	12/15/2014 12:30	12/15/2014 12:41	0:11:45	0:00:32	TOTAL	5:17:22	???	
23691		BWA (For each lane)	5	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:00			2416	BWA2_FEL
23691		BWA (For each lane)	7	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:02	0:00:01			68	
23691		BWA (For each lane)	96	CollectWFResults	12/15/2014 10:39	12/15/2014 10:50	0:10:52	0:01:02			17033454431	I
23691		BWA (For each lane)	99	SAMTools-Merge	12/15/2014 10:50	12/15/2014 11:41	0:51:12	0:03:19			17033452787	
23691		BWA (For each lane)	108	SAMTools-Sort	12/15/2014 11:41	12/15/2014 12:46	1:05:04	0:24:51	Wait	3:14:49	12187484531	O
23691		BWA (For each lane)	119	SAMTools-Index	12/15/2014 12:46	12/15/2014 12:50	0:03:30	0:03:31	BWA_A	0:00:03	6253544	
23691		BWA (For each lane)	120	ExportFiles	12/15/2014 12:50	12/15/2014 12:50	0:00:00	0:00:01	BWA_B	2:21:22	???	
23691		BWA (For each lane)	121	ExportFiles	12/15/2014 12:50	12/15/2014 13:00	0:10:40	0:31:55	TOTAL	5:36:14	???	
23692		BWA (For each lane)	9	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01			2416	BWA3_FEL
23692		BWA (For each lane)	10	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:0:-1			68	
23692		BWA (For each lane)	91	CollectWFResults	12/15/2014 10:30	12/15/2014 10:40	0:09:51	0:0:-1			15631861654	I
23692		BWA (For each lane)	97	SAMTools-Merge	12/15/2014 10:40	12/15/2014 11:28	0:48:25	0:04:50			15631835494	
23692		BWA (For each lane)	107	SAMTools-Sort	12/15/2014 11:28	12/15/2014 12:30	1:01:49	0:12:38	Wait	3:05:59	11316144438	O
23692		BWA (For each lane)	116	SAMTools-Index	12/15/2014 12:30	12/15/2014 12:33	0:03:17	0:03:17	BWA_A	0:00:02	6432376	
23692		BWA (For each lane)	117	ExportFiles	12/15/2014 12:33	12/15/2014 12:33	0:00:02	0:00:02	BWA_B	2:14:37	???	

23692		BWA (For each lane)	118	ExportFiles	12/15/2014 12:33	12/15/2014 12:45	0:11:12	0:12:34	TOTAL	5:20:38	???	
23693		BWA (For each lane)	11	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01			2416	BWA4_FEL
23693		BWA (For each lane)	12	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:0:-2			68	
23693		BWA (For each lane)	72	CollectWFResults	12/15/2014 9:41	12/15/2014 9:48	0:06:44	0:01:41			12113188483	I
23693		BWA (For each lane)	76	SAMTools-Merge	12/15/2014 9:53	12/15/2014 10:30	0:37:23	0:01:13			12113202634	
23693		BWA (For each lane)	94	SAMTools-Sort	12/15/2014 10:30	12/15/2014 11:18	0:47:15	0:08:08	Wait	2:16:46	8788010149	O
23693		BWA (For each lane)	102	SAMTools-Index	12/15/2014 11:18	12/15/2014 11:20	0:02:32	0:00:55	BWA_A	0:00:02	6299848	
23693		BWA (For each lane)	104	ExportFiles	12/15/2014 11:20	12/15/2014 11:20	0:00:03	0:00:03	BWA_B	1:47:03	???	
23693		BWA (For each lane)	105	ExportFiles	12/15/2014 11:20	12/15/2014 11:28	0:07:36	0:06:27	TOTAL	4:03:51	???	
23694		BWA (For each lane)	13	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01			2416	BWA5_FEL
23694		BWA (For each lane)	14	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:02	0:00:01			68	
23694		BWA (For each lane)	86	CollectWFResults	12/15/2014 10:19	12/15/2014 10:29	0:10:01	0:01:24			16216042490	I
23694		BWA (For each lane)	90	SAMTools-Merge	12/15/2014 10:29	12/15/2014 11:19	0:49:39	0:01:06			16216048444	
23694		BWA (For each lane)	103	SAMTools-Sort	12/15/2014 11:19	12/15/2014 12:21	1:02:49	0:01:38	Wait	2:54:52	11733168987	O
23694		BWA (For each lane)	110	SAMTools-Index	12/15/2014 12:21	12/15/2014 12:25	0:03:24	0:03:24	BWA_A	0:00:03	6407800	
23694		BWA (For each lane)	111	ExportFiles	12/15/2014 12:25	12/15/2014 12:25	0:00:01	0:00:01	BWA_B	2:18:30	???	
23694		BWA (For each lane)	112	ExportFiles	12/15/2014 12:25	12/15/2014 12:37	0:12:36	0:01:28	TOTAL	5:13:25	???	
23695		BWA (For each lane)	15	GetFileReferences	12/15/2014 7:24	12/15/2014 7:24	0:00:01	0:00:01			2416	BWA6_FEL
23695		BWA (For each lane)	16	WorkflowPerFileWithRef	12/15/2014 7:24	12/15/2014 7:24	0:00:02	0:00:00			68	
23695		BWA (For each lane)	100	CollectWFResults	12/15/2014 10:53	12/15/2014 11:05	0:12:20	0:12:20			17858114745	I
23695		BWA (For each lane)	101	SAMTools-Merge	12/15/2014 11:05	12/15/2014 12:06	1:00:24	0:12:18			17858132616	
23695		BWA (For each lane)	109	SAMTools-Sort	12/15/2014 12:06	12/15/2014 13:21	1:15:41	0:15:37	Wait	3:28:57	12587829996	O
23695		BWA (For each lane)	122	SAMTools-Index	12/15/2014 13:21	12/15/2014 13:25	0:03:50	0:03:50	BWA_A	0:00:03	6612096	
23695		BWA (For each lane)	123	ExportFiles	12/15/2014 13:25	12/15/2014 13:25	0:00:01	0:00:01	BWA_B	2:43:58	???	
23695		BWA (For each lane)	124	ExportFiles	12/15/2014 13:25	12/15/2014 13:37	0:11:41	0:11:48	TOTAL	6:12:58	???	
23696		BWA (Align lane)	26	ImportDirectory	12/15/2014 7:24	12/15/2014 7:30	0:05:33	0:00:03			8056050385	I BWA_A1_AL
23696		BWA (Align lane)	38	GZip	12/15/2014 7:30	12/15/2014 7:40	0:10:18	0:00:05			19831433538	
23696		BWA (Align lane)	59	PrepareLibrary	12/15/2014 7:40	12/15/2014 7:40	0:00:02	0:00:02			4657	
23696		BWA (Align lane)	60	BWA_Mem	12/15/2014 7:40	12/15/2014 9:54	2:14:04	0:00:19			23522125554	
23696		BWA (Align lane)	78	SAMTools-SAM2BAM	12/15/2014 9:55	12/15/2014 10:22	0:26:49	0:00:25			8549708632	O
23696		BWA (Align lane)	93	ExportFiles	12/15/2014 10:30	12/15/2014 10:39	0:08:41	0:00:15	TOTAL	3:14:48	???	
23697		BWA (Align lane)	21	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:04:52	0:00:02			7507423824	I BWA_B1_AL
23697		BWA (Align lane)	32	GZip	12/15/2014 7:29	12/15/2014 7:38	0:09:21	0:00:07			18502392392	
23697		BWA (Align lane)	47	PrepareLibrary	12/15/2014 7:38	12/15/2014 7:38	0:00:02	0:00:02			4657	
23697		BWA (Align lane)	48	BWA_Mem	12/15/2014 7:38	12/15/2014 9:35	1:56:17	0:00:00			21948489227	
23697		BWA (Align lane)	70	SAMTools-SAM2BAM	12/15/2014 9:35	12/15/2014 9:59	0:24:44	0:02:55			7963230281	O
23697		BWA (Align lane)	81	ExportFiles	12/15/2014 9:59	12/15/2014 10:08	0:08:44	0:03:35	TOTAL	2:44:00	???	
23698		BWA (Align lane)	24	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:05:25	0:00:01			8000668868	I BWA_A2_AL
23698		BWA (Align lane)	36	GZip	12/15/2014 7:29	12/15/2014 7:40	0:10:25	0:00:05			19687368280	
23698		BWA (Align lane)	57	PrepareLibrary	12/15/2014 7:40	12/15/2014 7:40	0:00:01	0:00:01			4657	
23698		BWA (Align lane)	58	BWA_Mem	12/15/2014 7:40	12/15/2014 9:53	2:12:52	0:0:-1			23352524680	
23698		BWA (Align lane)	77	SAMTools-SAM2BAM	12/15/2014 9:54	12/15/2014 10:20	0:26:20	0:01:10			8483745799	O
23698		BWA (Align lane)	92	ExportFiles	12/15/2014 10:30	12/15/2014 10:39	0:08:30	0:00:08	TOTAL	3:14:28	???	
23699		BWA (Align lane)	23	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:05:21	0:0:-1			7386052462	I BWA_B2_AL
23699		BWA (Align lane)	35	GZip	12/15/2014 7:29	12/15/2014 7:39	0:09:43	0:00:03			18352931750	

23699		BWA (Align lane)	51	PrepareLibrary	12/15/2014 7:39	12/15/2014 7:39	0:00:02	0:00:03			4657	
23699		BWA (Align lane)	52	BWA_Mem	12/15/2014 7:39	12/15/2014 9:42	2:03:19	0:00:06			21770360909	
23699		BWA (Align lane)	73	SAMTools-SAM2BAM	12/15/2014 9:42	12/15/2014 10:07	0:24:39	0:02:54			7831858962	O
23699		BWA (Align lane)	83	ExportFiles	12/15/2014 10:07	12/15/2014 10:15	0:07:34	0:03:59	TOTAL	2:50:39	???	
23700		BWA (Align lane)	19	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:04:44	0:00:01			7452878642	I BWA_A3_AL
23700		BWA (Align lane)	31	GZip	12/15/2014 7:29	12/15/2014 7:38	0:09:22	0:00:08			18362808298	
23700		BWA (Align lane)	43	PrepareLibrary	12/15/2014 7:38	12/15/2014 7:38	0:00:02	0:00:02			4657	
23700		BWA (Align lane)	44	BWA_Mem	12/15/2014 7:38	12/15/2014 9:34	1:55:29	0:00:06			21784127062	
23700		BWA (Align lane)	69	SAMTools-SAM2BAM	12/15/2014 9:34	12/15/2014 9:58	0:24:40	0:00:54			7898722768	O
23700		BWA (Align lane)	80	ExportFiles	12/15/2014 9:58	12/15/2014 10:07	0:08:17	0:00:58	TOTAL	2:42:35	???	
23701		BWA (Align lane)	20	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:04:58	0:0:-1			7356633844	I BWA_B3_AL
23701		BWA (Align lane)	33	GZip	12/15/2014 7:29	12/15/2014 7:40	0:10:33	0:00:00			18299259420	
23701		BWA (Align lane)	55	PrepareLibrary	12/15/2014 7:40	12/15/2014 7:40	0:00:02	0:00:03			4657	
23701		BWA (Align lane)	56	BWA_Mem	12/15/2014 7:40	12/15/2014 9:41	2:01:10	0:00:16			21707561700	
23701		BWA (Align lane)	75	SAMTools-SAM2BAM	12/15/2014 9:48	12/15/2014 10:13	0:25:07	0:05:13			7800002692	O
23701		BWA (Align lane)	88	ExportFiles	12/15/2014 10:22	12/15/2014 10:30	0:08:02	0:04:29	TOTAL	3:05:57	???	
23702		BWA (Align lane)	18	ImportDirectory	12/15/2014 7:24	12/15/2014 7:28	0:04:08	0:0:-2			5727243157	I BWA_A4_AL
23702		BWA (Align lane)	30	GZip	12/15/2014 7:28	12/15/2014 7:38	0:10:04	0:00:34			14195215904	
23702		BWA (Align lane)	45	PrepareLibrary	12/15/2014 7:38	12/15/2014 7:38	0:00:02	0:00:02			4657	
23702		BWA (Align lane)	46	BWA_Mem	12/15/2014 7:38	12/15/2014 9:14	1:35:38	0:0:-3			16837764904	
23702		BWA (Align lane)	66	SAMTools-SAM2BAM	12/15/2014 9:14	12/15/2014 9:33	0:19:28	0:12:26			6083441789	O
23702		BWA (Align lane)	68	ExportFiles	12/15/2014 9:33	12/15/2014 9:40	0:06:34	0:00:16	TOTAL	2:15:54	???	
23703		BWA (Align lane)	17	ImportDirectory	12/15/2014 7:24	12/15/2014 7:28	0:03:45	0:00:01			5677081672	I BWA_B4_AL
23703		BWA (Align lane)	29	GZip	12/15/2014 7:28	12/15/2014 7:37	0:09:07	0:00:23			14085154522	
23703		BWA (Align lane)	41	PrepareLibrary	12/15/2014 7:37	12/15/2014 7:37	0:00:02	0:00:02			4657	
23703		BWA (Align lane)	42	BWA_Mem	12/15/2014 7:37	12/15/2014 9:07	1:30:20	0:01:10			16707771454	
23703		BWA (Align lane)	65	SAMTools-SAM2BAM	12/15/2014 9:07	12/15/2014 9:26	0:19:03	0:06:38			6029746694	O
23703		BWA (Align lane)	67	ExportFiles	12/15/2014 9:26	12/15/2014 9:32	0:05:49	0:07:02	TOTAL	2:08:08	???	
23704		BWA (Align lane)	27	ImportDirectory	12/15/2014 7:24	12/15/2014 7:30	0:05:28	0:00:00			7674281065	I BWA_A5_AL
23704		BWA (Align lane)	37	GZip	12/15/2014 7:30	12/15/2014 7:39	0:09:43	0:00:02			19058807824	
23704		BWA (Align lane)	53	PrepareLibrary	12/15/2014 7:39	12/15/2014 7:39	0:00:02	0:00:02			4657	
23704		BWA (Align lane)	54	BWA_Mem	12/15/2014 7:39	12/15/2014 9:45	2:06:05	0:00:15			22608256865	
23704		BWA (Align lane)	74	SAMTools-SAM2BAM	12/15/2014 9:45	12/15/2014 10:11	0:25:44	0:02:09			8146434485	O
23704		BWA (Align lane)	84	ExportFiles	12/15/2014 10:11	12/15/2014 10:19	0:07:45	0:01:32	TOTAL	2:54:48	???	
23705		BWA (Align lane)	25	ImportDirectory	12/15/2014 7:24	12/15/2014 7:30	0:05:36	0:0:-2			8490980528	I BWA_B5_AL
23705		BWA (Align lane)	39	GZip	12/15/2014 7:30	12/15/2014 7:40	0:10:33	0:00:29			22202616440	
23705		BWA (Align lane)	61	PrepareLibrary	12/15/2014 7:40	12/15/2014 7:40	0:00:01	0:00:02			4657	
23705		BWA (Align lane)	62	BWA_Mem	12/15/2014 7:40	12/15/2014 9:55	2:14:53	0:00:14			26371087511	
23705		BWA (Align lane)	85	SAMTools-SAM2BAM	12/15/2014 10:13	12/15/2014 10:45	0:32:03	0:06:15			8922975618	O
23705		BWA (Align lane)	98	ExportFiles	12/15/2014 10:45	12/15/2014 10:53	0:08:16	0:05:00	TOTAL	3:28:55	???	
23706		BWA (Align lane)	22	ImportDirectory	12/15/2014 7:24	12/15/2014 7:29	0:04:56	0:00:00			7602869847	I BWA_A6_AL
23706		BWA (Align lane)	34	GZip	12/15/2014 7:29	12/15/2014 7:38	0:09:16	0:00:24			18901953208	
23706		BWA (Align lane)	49	PrepareLibrary	12/15/2014 7:38	12/15/2014 7:38	0:00:02	0:00:02			4657	
23706		BWA (Align lane)	50	BWA_Mem	12/15/2014 7:38	12/15/2014 9:37	1:59:10	0:00:49			22423059118	
23706		BWA (Align lane)	71	SAMTools-SAM2BAM	12/15/2014 9:37	12/15/2014 10:03	0:25:24	0:03:20			8069608005	O

23706		BWA (Align lane)	82	ExportFiles	12/15/2014 10:03	12/15/2014 10:11	0:08:15	0:04:16	TOTAL	2:47:04	???	
23707		BWA (Align lane)	28	ImportDirectory	12/15/2014 7:24	12/15/2014 7:30	0:06:04	0:03:44			8503011732	I BWA_B6_AL
23707		BWA (Align lane)	40	GZip	12/15/2014 7:30	12/15/2014 7:40	0:10:21	0:06:47			22205300398	
23707		BWA (Align lane)	63	PrepareLibrary	12/15/2014 7:40	12/15/2014 7:41	0:00:02	0:00:03			4657	
23707		BWA (Align lane)	64	BWA_Mem	12/15/2014 7:41	12/15/2014 9:56	2:15:01	1:26:45			26374822681	
23707		BWA (Align lane)	79	SAMTools-SAM2BAM	12/15/2014 9:56	12/15/2014 10:26	0:30:53	0:02:46			8935139127	O
23707		BWA (Align lane)	89	ExportFiles	12/15/2014 10:26	12/15/2014 10:36	0:09:15	0:02:29	TOTAL	3:11:37	???	
23756		Picard-Tools	126	ImportDirectory	12/15/2014 13:37	12/15/2014 13:45	0:08:04	0:0:-1			11342082370	I PICARD1
23756		Picard-Tools	138	PickFile	12/15/2014 13:45	12/15/2014 13:46	0:00:58	0:00:22			11335839402	
23756		Picard-Tools	143	Picard-CleanSAM	12/15/2014 13:46	12/15/2014 14:19	0:33:14	0:22:55			11450605131	
23756		Picard-Tools	145	Picard-MarkDuplicates	12/15/2014 14:19	12/15/2014 15:08	0:48:07	0:00:08			7315131348	
23756		Picard-Tools	151	Picard-AddOrReplaceRG	12/15/2014 15:08	12/15/2014 15:36	0:28:00	0:04:58			7410818872	O
23756		Picard-Tools	159	SAMTools-Index	12/15/2014 15:36	12/15/2014 15:38	0:02:12	0:02:12			6235720	
23756		Picard-Tools	160	ExportFiles	12/15/2014 15:38	12/15/2014 15:38	0:00:01	0:00:01			???	
23756		Picard-Tools	161	ExportFiles	12/15/2014 15:38	12/15/2014 15:44	0:06:21	0:04:54	TOTAL	2:06:59	???	
23757		Picard-Tools	127	ImportDirectory	12/15/2014 13:37	12/15/2014 13:45	0:07:23	0:00:02			12193738075	I PICARD2
23757		Picard-Tools	134	PickFile	12/15/2014 13:45	12/15/2014 13:46	0:01:01	0:00:17			12187484531	
23757		Picard-Tools	139	Picard-CleanSAM	12/15/2014 13:46	12/15/2014 14:21	0:35:39	0:00:10			12311571714	
23757		Picard-Tools	148	Picard-MarkDuplicates	12/15/2014 14:21	12/15/2014 15:13	0:51:32	0:05:24			7872487516	
23757		Picard-Tools	153	Picard-AddOrReplaceRG	12/15/2014 15:13	12/15/2014 15:43	0:29:53	0:02:38			7975311426	O
23757		Picard-Tools	162	SAMTools-Index	12/15/2014 15:43	12/15/2014 15:45	0:02:22	0:02:22			6246504	
23757		Picard-Tools	163	ExportFiles	12/15/2014 15:45	12/15/2014 15:45	0:00:01	0:00:02			???	
23757		Picard-Tools	164	ExportFiles	12/15/2014 15:45	12/15/2014 15:52	0:07:04	0:02:09	TOTAL	2:15:01	???	
23758		Picard-Tools	128	ImportDirectory	12/15/2014 13:37	12/15/2014 13:45	0:07:38	0:0:-1			11322576814	I PICARD3
23758		Picard-Tools	135	PickFile	12/15/2014 13:45	12/15/2014 13:46	0:00:57	0:0:-1			11316144438	
23758		Picard-Tools	140	Picard-CleanSAM	12/15/2014 13:46	12/15/2014 14:20	0:33:47	0:00:01			11436941982	
23758		Picard-Tools	146	Picard-MarkDuplicates	12/15/2014 14:20	12/15/2014 15:13	0:52:57	0:00:45			8962734019	
23758		Picard-Tools	152	Picard-AddOrReplaceRG	12/15/2014 15:13	12/15/2014 15:47	0:34:43	0:00:17			9078276010	O
23758		Picard-Tools	165	SAMTools-Index	12/15/2014 15:47	12/15/2014 15:50	0:02:41	0:02:41			6428448	
23758		Picard-Tools	166	ExportFiles	12/15/2014 15:50	12/15/2014 15:50	0:00:01	0:00:01			???	
23758		Picard-Tools	167	ExportFiles	12/15/2014 15:50	12/15/2014 15:58	0:08:28	0:01:19	TOTAL	2:21:14	???	
23759		Picard-Tools	129	ImportDirectory	12/15/2014 13:37	12/15/2014 13:43	0:05:24	0:00:00			8794309997	I PICARD4
23759		Picard-Tools	132	PickFile	12/15/2014 13:43	12/15/2014 13:43	0:00:44	0:00:44			8788010149	
23759		Picard-Tools	133	Picard-CleanSAM	12/15/2014 13:43	12/15/2014 14:09	0:25:49	0:01:14			8879132809	
23759		Picard-Tools	144	Picard-MarkDuplicates	12/15/2014 14:09	12/15/2014 14:49	0:39:55	0:10:19			7057978345	
23759		Picard-Tools	150	Picard-AddOrReplaceRG	12/15/2014 14:49	12/15/2014 15:16	0:26:59	0:18:32			7151140292	O
23759		Picard-Tools	155	SAMTools-Index	12/15/2014 15:16	12/15/2014 15:18	0:02:06	0:02:07			6297576	
23759		Picard-Tools	156	ExportFiles	12/15/2014 15:18	12/15/2014 15:18	0:00:01	0:00:01			???	
23759		Picard-Tools	157	ExportFiles	12/15/2014 15:18	12/15/2014 15:25	0:06:36	0:17:12	TOTAL	1:47:35	???	
23760		Picard-Tools	130	ImportDirectory	12/15/2014 13:37	12/15/2014 13:45	0:07:38	0:00:01			11739576787	I PICARD5
23760		Picard-Tools	136	PickFile	12/15/2014 13:45	12/15/2014 13:46	0:00:59	0:00:16			11733168987	
23760		Picard-Tools	141	Picard-CleanSAM	12/15/2014 13:46	12/15/2014 14:20	0:34:31	0:00:20			11858587245	
23760		Picard-Tools	147	Picard-MarkDuplicates	12/15/2014 14:20	12/15/2014 15:15	0:55:07	0:00:57			9409037758	
23760		Picard-Tools	154	Picard-AddOrReplaceRG	12/15/2014 15:15	12/15/2014 15:51	0:35:49	0:00:34			9531142797	O
23760		Picard-Tools	168	SAMTools-Index	12/15/2014 15:51	12/15/2014 15:54	0:02:49	0:02:49			6403104	

23760		Picard-Tools	169	ExportFiles	12/15/2014 15:54	12/15/2014 15:54	0:00:01	0:00:02			???	
23760		Picard-Tools	170	ExportFiles	12/15/2014 15:54	12/15/2014 16:03	0:08:53	0:29:08	TOTAL	2:25:50	???	
23761		Picard-Tools	131	ImportDirectory	12/15/2014 13:37	12/15/2014 13:45	0:07:52	0:05:23			12594442092	I PICARD6
23761		Picard-Tools	137	PickFile	12/15/2014 13:45	12/15/2014 13:46	0:01:03	0:00:10			12587829996	
23761		Picard-Tools	142	Picard-CleanSAM	12/15/2014 13:46	12/15/2014 14:27	0:40:33	0:00:05			12766210063	
23761		Picard-Tools	149	Picard-MarkDuplicates	12/15/2014 14:27	12/15/2014 15:35	1:08:41	0:22:22			11643289399	
23761		Picard-Tools	158	Picard-AddOrReplaceRG	12/15/2014 15:35	12/15/2014 16:23	0:47:54	0:00:14			11800280496	O
23761		Picard-Tools	171	SAMTools-Index	12/15/2014 16:23	12/15/2014 16:27	0:03:40	0:03:40			6613616	
23761		Picard-Tools	172	ExportFiles	12/15/2014 16:27	12/15/2014 16:27	0:00:01	0:00:01			???	
23761		Picard-Tools	173	ExportFiles	12/15/2014 16:27	12/15/2014 16:38	0:10:39	0:10:45	TOTAL	3:00:24	???	
23786		GATK phase 1	177	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			47	GATKP1_1
23786		GATK phase 1	178	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-1			60	
23786		GATK phase 1	188	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:00			4657	
23786		GATK phase 1	193	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:00			4924	
23786		GATK phase 1	199	ImportDirectory	12/15/2014 16:38	12/15/2014 16:43	0:04:44	0:0:-1			7417054592	I
23786		GATK phase 1	206	PickFile	12/15/2014 16:43	12/15/2014 16:43	0:00:38	0:00:06			7410818872	
23786		GATK phase 1	209	GATK-RealignerTargetCre	12/15/2014 16:43	12/15/2014 17:24	0:41:14	0:00:07			3597127	
23786		GATK phase 1	218	GATK-IndelRealigner	12/15/2014 17:24	12/15/2014 18:11	0:46:14	0:02:01			7449553429	
23786		GATK phase 1	223	GATK-BaseRecalibrator	12/15/2014 18:11	12/15/2014 19:01	0:50:06	0:02:57			916648	
23786		GATK phase 1	228	GATK-PrintReads	12/15/2014 19:01	12/15/2014 20:25	1:24:28	0:01:32			12740661801	
23786		GATK phase 1	236	GATK-ReduceReads	12/15/2014 20:25	12/15/2014 22:08	1:42:51	0:10:55			2766450004	O
23786		GATK phase 1	243	ExportFiles	12/15/2014 22:08	12/15/2014 22:11	0:02:53	0:02:53				
23786		GATK phase 1	244	SAMTools-Index	12/15/2014 22:11	12/15/2014 22:12	0:00:48	0:00:48			6215392	
23786		GATK phase 1	245	ExportFiles	12/15/2014 22:12	12/15/2014 22:12	0:00:02	0:14:57	TOTAL	5:34:10		
23787		GATK phase 1	175	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			47	GATKP1_2
23787		GATK phase 1	176	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:00			60	
23787		GATK phase 1	187	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			4657	
23787		GATK phase 1	191	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:02			4924	
23787		GATK phase 1	200	ImportDirectory	12/15/2014 16:38	12/15/2014 16:43	0:04:50	0:00:02			7981557930	I
23787		GATK phase 1	207	PickFile	12/15/2014 16:43	12/15/2014 16:43	0:00:40	0:00:23			7975311426	
23787		GATK phase 1	210	GATK-RealignerTargetCre	12/15/2014 16:43	12/15/2014 17:26	0:43:07	0:00:13			3735131	
23787		GATK phase 1	219	GATK-IndelRealigner	12/15/2014 17:26	12/15/2014 18:19	0:52:50	0:06:35			8016830706	
23787		GATK phase 1	225	GATK-BaseRecalibrator	12/15/2014 18:19	12/15/2014 19:04	0:44:22	0:20:12			916539	
23787		GATK phase 1	230	GATK-PrintReads	12/15/2014 19:04	12/15/2014 20:36	1:32:33	0:01:21			13276459043	
23787		GATK phase 1	237	GATK-ReduceReads	12/15/2014 20:36	12/15/2014 22:27	1:50:34	0:42:28			2824430083	O
23787		GATK phase 1	246	ExportFiles	12/15/2014 22:27	12/15/2014 22:30	0:02:47	0:02:48				
23787		GATK phase 1	247	SAMTools-Index	12/15/2014 22:30	12/15/2014 22:30	0:00:49	0:00:49			6224816	
23787		GATK phase 1	248	ExportFiles	12/15/2014 22:30	12/15/2014 22:30	0:00:02	0:14:48	TOTAL	5:52:45		
23788		GATK phase 1	180	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			47	GATKP1_3
23788		GATK phase 1	183	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-1			60	
23788		GATK phase 1	192	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-1			4657	
23788		GATK phase 1	198	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-1			4924	
23788		GATK phase 1	202	ImportDirectory	12/15/2014 16:38	12/15/2014 16:44	0:05:48	0:00:00			9084704458	I
23788		GATK phase 1	212	PickFile	12/15/2014 16:44	12/15/2014 16:44	0:00:47	0:00:44			9078276010	
23788		GATK phase 1	214	GATK-RealignerTargetCre	12/15/2014 16:45	12/15/2014 17:33	0:48:32	0:00:21			4816333	

23788		GATK phase 1	220	GATK-IndelRealigner	12/15/2014 17:33	12/15/2014 18:39	1:06:26	0:00:06			9126965264	
23788		GATK phase 1	226	GATK-BaseRecalibrator	12/15/2014 18:40	12/15/2014 19:34	0:54:09	0:08:43			916757	
23788		GATK phase 1	232	GATK-PrintReads	12/15/2014 19:34	12/15/2014 21:19	1:45:00	0:15:04			15308823442	
23788		GATK phase 1	238	GATK-ReduceReads	12/15/2014 21:19	12/15/2014 23:33	2:14:10	0:15:54			3430581549	O
23788		GATK phase 1	250	ExportFiles	12/15/2014 23:33	12/15/2014 23:36	0:03:27	0:02:56				
23788		GATK phase 1	252	SAMTools-Index	12/15/2014 23:36	12/15/2014 23:37	0:01:00	0:01:00			6405264	
23788		GATK phase 1	253	ExportFiles	12/15/2014 23:37	12/15/2014 23:37	0:00:02	0:01:00	TOTAL	6:59:38		
23789		GATK phase 1	179	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			47	GATKP1_4
23789		GATK phase 1	184	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			60	
23789		GATK phase 1	194	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			4657	
23789		GATK phase 1	196	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			4924	
23789		GATK phase 1	201	ImportDirectory	12/15/2014 16:38	12/15/2014 16:42	0:04:34	0:00:01			7157437868	I
23789		GATK phase 1	205	PickFile	12/15/2014 16:42	12/15/2014 16:43	0:00:37	0:00:09			7151140292	
23789		GATK phase 1	208	GATK-RealignerTargetCre	12/15/2014 16:43	12/15/2014 17:24	0:41:11	0:00:11			3994757	
23789		GATK phase 1	217	GATK-IndelRealigner	12/15/2014 17:24	12/15/2014 18:14	0:49:25	0:00:13			7188890298	
23789		GATK phase 1	224	GATK-BaseRecalibrator	12/15/2014 18:14	12/15/2014 19:02	0:48:43	0:05:40			916212	
23789		GATK phase 1	229	GATK-PrintReads	12/15/2014 19:02	12/15/2014 20:24	1:21:17	0:01:19			12044923155	
23789		GATK phase 1	235	GATK-ReduceReads	12/15/2014 20:24	12/15/2014 22:04	1:40:40	0:01:39			2787113930	O
23789		GATK phase 1	240	ExportFiles	12/15/2014 22:04	12/15/2014 22:07	0:02:41	0:02:42				
23789		GATK phase 1	241	SAMTools-Index	12/15/2014 22:07	12/15/2014 22:08	0:00:48	0:00:48			6279120	
23789		GATK phase 1	242	ExportFiles	12/15/2014 22:08	12/15/2014 22:08	0:00:02	0:00:20	TOTAL	5:30:09		
23790		GATK phase 1	182	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:03			47	GATKP1_5
23790		GATK phase 1	186	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:00			60	
23790		GATK phase 1	189	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:02			4657	
23790		GATK phase 1	195	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:01			4924	
23790		GATK phase 1	204	ImportDirectory	12/15/2014 16:38	12/15/2014 16:44	0:05:44	0:04:35			9537545901	I
23790		GATK phase 1	211	PickFile	12/15/2014 16:44	12/15/2014 16:44	0:00:49	0:00:06			9531142797	
23790		GATK phase 1	213	GATK-RealignerTargetCre	12/15/2014 16:44	12/15/2014 17:33	0:48:45	0:00:07			4982446	
23790		GATK phase 1	221	GATK-IndelRealigner	12/15/2014 17:33	12/15/2014 18:48	1:15:04	0:14:02			9581291374	
23790		GATK phase 1	227	GATK-BaseRecalibrator	12/15/2014 18:48	12/15/2014 19:49	1:00:30	0:12:36			916212	
23790		GATK phase 1	233	GATK-PrintReads	12/15/2014 19:49	12/15/2014 21:35	1:45:50	0:30:02			15893371461	
23790		GATK phase 1	239	GATK-ReduceReads	12/15/2014 21:35	12/15/2014 23:36	2:01:12	0:29:44			3290229376	O
23790		GATK phase 1	251	ExportFiles	12/15/2014 23:36	12/15/2014 23:38	0:02:32	0:00:32				
23790		GATK phase 1	254	SAMTools-Index	12/15/2014 23:38	12/15/2014 23:39	0:00:57	0:00:57			6378272	
23790		GATK phase 1	255	ExportFiles	12/15/2014 23:39	12/15/2014 23:39	0:00:02	1:56:00	TOTAL	7:01:36		
23791		GATK phase 1	181	GATK-ReadFilter-Init	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-2			47	GATKP1_6
23791		GATK phase 1	185	GATK-ReadFilter-BadCigar	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-1			60	
23791		GATK phase 1	190	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:0:-2			4657	
23791		GATK phase 1	197	PrepareLibrary	12/15/2014 16:38	12/15/2014 16:38	0:00:01	0:00:00			4924	
23791		GATK phase 1	203	ImportDirectory	12/15/2014 16:38	12/15/2014 16:45	0:07:00	0:0:-2			11806894112	I
23791		GATK phase 1	215	PickFile	12/15/2014 16:45	12/15/2014 16:46	0:01:02	0:01:02			11800280496	
23791		GATK phase 1	216	GATK-RealignerTargetCre	12/15/2014 16:46	12/15/2014 17:47	1:01:18	0:38:21			5557020	
23791		GATK phase 1	222	GATK-IndelRealigner	12/15/2014 17:47	12/15/2014 19:05	1:17:51	0:23:31			11855629971	
23791		GATK phase 1	231	GATK-BaseRecalibrator	12/15/2014 19:05	12/15/2014 20:19	1:13:45	0:28:39			917288	
23791		GATK phase 1	234	GATK-PrintReads	12/15/2014 20:19	12/15/2014 22:45	2:26:25	0:04:53			19968818285	

23791		GATK phase 1	249	GATK-ReduceReads	12/15/2014 22:45	12/16/2014 1:35	2:50:04	0:47:39			3629593202	O
23791		GATK phase 1	256	ExportFiles	12/16/2014 1:35	12/16/2014 1:39	0:03:22	0:03:22				
23791		GATK phase 1	257	SAMTools-Index	12/16/2014 1:39	12/16/2014 1:40	0:01:03	0:01:04			6600528	
23791		GATK phase 1	258	ExportFiles	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:07	TOTAL	9:01:59		
23824		Variant Calling with Chro	261	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:02			4657	
23824		Variant Calling with Chro	262	GenerateIntervals	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			1213	
23824		Variant Calling with Chro	263	ImportFile	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:02			49	
23824		Variant Calling with Chro	264	ParameterSweep	12/16/2014 1:40	12/16/2014 1:40	0:00:20	0:00:01	Wait	3:30:13	588	
23824		Variant Calling with Chro	715	CollectWFResults	12/16/2014 5:11	12/16/2014 5:11	0:00:29	0:00:29			186484721	
23824		Variant Calling with Chro	716	GATK-CatVariants	12/16/2014 5:11	12/16/2014 5:12	0:00:43	0:00:43			185888349	
23824		Variant Calling with Chro	717	ExportFiles	12/16/2014 5:12	12/16/2014 5:12	0:00:14	0:00:16	TOTAL	3:32:04		
23828		Haplotype Caller	265	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:03			4657	
23828		Haplotype Caller	270	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:02			0	
23828		Haplotype Caller	274	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			49	
23828		Haplotype Caller	281	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			158	
23828		Haplotype Caller	290	CollectWFResults	12/16/2014 1:40	12/16/2014 1:59	0:18:25	0:00:02			18766501536	
23828		Haplotype Caller	325	FileListJoin	12/16/2014 1:59	12/16/2014 2:01	0:02:25	0:00:01			18766501536	
23828		Haplotype Caller	328	PickFiles	12/16/2014 2:01	12/16/2014 2:03	0:02:21	0:00:00			18728398144	
23828		Haplotype Caller	340	GATK-HaplotypeCaller	12/16/2014 2:03	12/16/2014 2:04	0:01:02	0:00:00			50866	
23828		Haplotype Caller	344	ExportFiles	12/16/2014 2:04	12/16/2014 2:04	0:00:02	0:00:01	TOTAL	0:24:25	???	
23829		Haplotype Caller	266	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			4657	
23829		Haplotype Caller	271	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:01			0	
23829		Haplotype Caller	275	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:0:-1			49	
23829		Haplotype Caller	280	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-2			158	
23829		Haplotype Caller	291	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:49	0:0:-2			18766501536	
23829		Haplotype Caller	337	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:00:00			18766501536	
23829		Haplotype Caller	356	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:10	0:00:00			18728398144	
23829		Haplotype Caller	365	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 3:19	1:10:38	0:00:00			6340750	
23829		Haplotype Caller	527	ExportFiles	12/16/2014 3:19	12/16/2014 3:19	0:00:02	0:00:03	TOTAL	1:38:59	???	
23830		Haplotype Caller	267	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			4657	
23830		Haplotype Caller	276	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:01			0	
23830		Haplotype Caller	284	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			49	
23830		Haplotype Caller	289	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			158	
23830		Haplotype Caller	302	CollectWFResults	12/16/2014 1:40	12/16/2014 2:01	0:21:03	0:00:00			18766501536	
23830		Haplotype Caller	331	FileListJoin	12/16/2014 2:01	12/16/2014 2:04	0:03:11	0:00:04			18766501536	
23830		Haplotype Caller	345	PickFiles	12/16/2014 2:04	12/16/2014 2:08	0:03:10	0:00:04			18728398144	
23830		Haplotype Caller	359	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:25	0:17:19	0:00:03			2917574	
23830		Haplotype Caller	370	ExportFiles	12/16/2014 2:25	12/16/2014 2:25	0:00:02	0:00:04	TOTAL	0:44:53	???	
23831		Haplotype Caller	268	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:02			4657	
23831		Haplotype Caller	278	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:02			0	
23831		Haplotype Caller	287	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:0:-1			49	
23831		Haplotype Caller	294	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			158	
23831		Haplotype Caller	306	CollectWFResults	12/16/2014 1:40	12/16/2014 1:59	0:18:26	0:0:-1			18766501536	
23831		Haplotype Caller	326	FileListJoin	12/16/2014 1:59	12/16/2014 2:01	0:02:24	0:00:02			18766501536	
23831		Haplotype Caller	329	PickFiles	12/16/2014 2:01	12/16/2014 2:03	0:02:22	0:00:01			18728398144	

23831		Haplotype Caller	342	GATK-HaplotypeCaller	12/16/2014 2:03	12/16/2014 2:44	0:40:40	0:00:59			4867783	
23831		Haplotype Caller	433	ExportFiles	12/16/2014 2:44	12/16/2014 2:44	0:00:01	0:00:04	TOTAL	1:04:00	???	
23832		Haplotype Caller	269	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-3			4657	
23832		Haplotype Caller	283	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:01			0	
23832		Haplotype Caller	292	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:00			49	
23832		Haplotype Caller	299	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			158	
23832		Haplotype Caller	309	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:48	0:00:00			18766501536	
23832		Haplotype Caller	336	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:00:00			18766501536	
23832		Haplotype Caller	355	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:10	0:00:00			18728398144	
23832		Haplotype Caller	364	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:34	0:25:44	0:00:01			4111348	
23832		Haplotype Caller	406	ExportFiles	12/16/2014 2:34	12/16/2014 2:34	0:00:03	0:00:06	TOTAL	0:54:03	???	
23833		Haplotype Caller	272	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			4657	
23833		Haplotype Caller	286	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:00			0	
23833		Haplotype Caller	296	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:0:-1			49	
23833		Haplotype Caller	303	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			158	
23833		Haplotype Caller	313	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:38	0:00:01			18766501536	
23833		Haplotype Caller	334	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:00:10			18766501536	
23833		Haplotype Caller	352	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:04	0:00:09			18728398144	
23833		Haplotype Caller	361	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:28	0:19:55	0:00:08			3044161	
23833		Haplotype Caller	382	ExportFiles	12/16/2014 2:28	12/16/2014 2:28	0:00:02	0:00:04	TOTAL	0:47:58	???	
23834		Haplotype Caller	273	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			4657	
23834		Haplotype Caller	288	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:04			0	
23834		Haplotype Caller	297	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:0:-1			49	
23834		Haplotype Caller	305	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			158	
23834		Haplotype Caller	315	CollectWFResults	12/16/2014 1:40	12/16/2014 1:59	0:18:26	0:00:03			18766501536	
23834		Haplotype Caller	327	FileListJoin	12/16/2014 1:59	12/16/2014 2:01	0:02:23	0:02:22			18766501536	
23834		Haplotype Caller	330	PickFiles	12/16/2014 2:01	12/16/2014 2:03	0:02:20	0:00:13			18728398144	
23834		Haplotype Caller	341	GATK-HaplotypeCaller	12/16/2014 2:03	12/16/2014 2:42	0:38:12	0:00:01			3795361	
23834		Haplotype Caller	424	ExportFiles	12/16/2014 2:42	12/16/2014 2:42	0:00:03	0:00:05	TOTAL	1:01:31	???	
23835		Haplotype Caller	277	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-2			4657	
23835		Haplotype Caller	293	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			0	
23835		Haplotype Caller	300	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			49	
23835		Haplotype Caller	308	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			158	
23835		Haplotype Caller	316	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:46	0:0:-1			18766501536	
23835		Haplotype Caller	335	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:00:01			18766501536	
23835		Haplotype Caller	353	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:11	0:00:00			18728398144	
23835		Haplotype Caller	363	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:31	0:23:08	0:00:00			3130472	
23835		Haplotype Caller	394	ExportFiles	12/16/2014 2:31	12/16/2014 2:32	0:00:02	0:00:04	TOTAL	0:51:26	???	
23836		Haplotype Caller	279	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:02			4657	
23836		Haplotype Caller	295	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:01			0	
23836		Haplotype Caller	304	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:00			49	
23836		Haplotype Caller	311	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-1			158	
23836		Haplotype Caller	317	CollectWFResults	12/16/2014 1:40	12/16/2014 2:01	0:21:05	0:0:-1			18766501536	
23836		Haplotype Caller	332	FileListJoin	12/16/2014 2:01	12/16/2014 2:04	0:03:10	0:00:11			18766501536	
23836		Haplotype Caller	346	PickFiles	12/16/2014 2:04	12/16/2014 2:08	0:03:09	0:00:00			18728398144	

23836		Haplotype Caller	360	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:26	0:18:32	0:00:27			3579073	
23836		Haplotype Caller	376	ExportFiles	12/16/2014 2:26	12/16/2014 2:26	0:00:02	0:00:04	TOTAL	0:46:06	???	
23837		Haplotype Caller	282	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			4657	
23837		Haplotype Caller	298	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:04			0	
23837		Haplotype Caller	307	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:03			49	
23837		Haplotype Caller	312	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			158	
23837		Haplotype Caller	318	CollectWFResults	12/16/2014 1:40	12/16/2014 2:01	0:21:17	0:00:02			18766501536	
23837		Haplotype Caller	333	FileListJoin	12/16/2014 2:01	12/16/2014 2:04	0:02:52	0:00:21			18766501536	
23837		Haplotype Caller	343	PickFiles	12/16/2014 2:04	12/16/2014 2:06	0:01:53	0:00:03			18728398144	
23837		Haplotype Caller	358	GATK-HaplotypeCaller	12/16/2014 2:06	12/16/2014 2:34	0:28:14	0:01:21			3651278	
23837		Haplotype Caller	412	ExportFiles	12/16/2014 2:34	12/16/2014 2:35	0:00:02	0:00:04	TOTAL	0:54:26	???	
23838		Haplotype Caller	285	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:0:-1			4657	
23838		Haplotype Caller	301	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:01			0	
23838		Haplotype Caller	310	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:00			49	
23838		Haplotype Caller	314	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:0:-2			158	
23838		Haplotype Caller	319	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:46	0:00:07			18766501536	
23838		Haplotype Caller	338	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:0:-1			18766501536	
23838		Haplotype Caller	357	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:11	0:01:03			18728398144	
23838		Haplotype Caller	366	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:30	0:21:29	0:08:46			2921003	
23838		Haplotype Caller	388	ExportFiles	12/16/2014 2:30	12/16/2014 2:30	0:00:02	0:00:05	TOTAL	0:49:47	???	
23839		Haplotype Caller	320	PrepareLibrary	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			4657	
23839		Haplotype Caller	321	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:02			0	
23839		Haplotype Caller	322	ImportDirectory	12/16/2014 1:40	12/16/2014 1:40	0:00:01	0:00:01			49	
23839		Haplotype Caller	323	ParseCSV-d	12/16/2014 1:40	12/16/2014 1:40	0:00:02	0:00:02			158	
23839		Haplotype Caller	324	CollectWFResults	12/16/2014 1:40	12/16/2014 2:02	0:21:33	0:18:08			18766501536	
23839		Haplotype Caller	339	FileListJoin	12/16/2014 2:02	12/16/2014 2:05	0:03:10	0:01:21			18766501536	
23839		Haplotype Caller	354	PickFiles	12/16/2014 2:05	12/16/2014 2:08	0:03:04	0:00:02			18728398144	
23839		Haplotype Caller	362	GATK-HaplotypeCaller	12/16/2014 2:08	12/16/2014 2:32	0:24:10	0:00:08			3335092	
23839		Haplotype Caller	400	ExportFiles	12/16/2014 2:32	12/16/2014 2:33	0:00:16	0:00:19	TOTAL	0:52:19	???	
23840		Haplotype Caller	347	PrepareLibrary	12/16/2014 2:04	12/16/2014 2:04	0:00:02	0:00:02			4657	
23840		Haplotype Caller	348	ImportDirectory	12/16/2014 2:04	12/16/2014 2:05	0:00:02	0:00:02			0	
23840		Haplotype Caller	349	ImportDirectory	12/16/2014 2:05	12/16/2014 2:05	0:00:01	0:00:01			49	
23840		Haplotype Caller	350	ParseCSV-d	12/16/2014 2:05	12/16/2014 2:05	0:00:01	0:00:01			158	
23840		Haplotype Caller	351	CollectWFResults	12/16/2014 2:05	12/16/2014 2:17	0:12:34	0:00:26			18766501536	
23840		Haplotype Caller	367	FileListJoin	12/16/2014 2:17	12/16/2014 2:19	0:02:20	0:02:21			18766501536	
23840		Haplotype Caller	368	PickFiles	12/16/2014 2:19	12/16/2014 2:22	0:02:09	0:02:09			18728398144	
23840		Haplotype Caller	369	GATK-HaplotypeCaller	12/16/2014 2:22	12/16/2014 2:45	0:23:15	0:03:16			3694312	
23840		Haplotype Caller	441	ExportFiles	12/16/2014 2:45	12/16/2014 2:45	0:00:02	0:00:07	TOTAL	0:40:28	???	
23841		Haplotype Caller	371	PrepareLibrary	12/16/2014 2:25	12/16/2014 2:25	0:00:01	0:00:02			4657	
23841		Haplotype Caller	372	ImportDirectory	12/16/2014 2:25	12/16/2014 2:25	0:00:02	0:00:02			0	
23841		Haplotype Caller	373	ImportDirectory	12/16/2014 2:25	12/16/2014 2:25	0:00:03	0:00:03			49	
23841		Haplotype Caller	374	ParseCSV-d	12/16/2014 2:25	12/16/2014 2:25	0:00:01	0:00:01			158	
23841		Haplotype Caller	375	CollectWFResults	12/16/2014 2:25	12/16/2014 2:37	0:12:18	0:01:04			18766501536	
23841		Haplotype Caller	418	FileListJoin	12/16/2014 2:37	12/16/2014 2:39	0:01:36	0:01:36			18766501536	
23841		Haplotype Caller	419	PickFiles	12/16/2014 2:39	12/16/2014 2:41	0:01:36	0:00:26			18728398144	

23841		Haplotype Caller	421	GATK-HaplotypeCaller	12/16/2014 2:41	12/16/2014 3:01	0:20:11	0:00:28			2815801	
23841		Haplotype Caller	467	ExportFiles	12/16/2014 3:01	12/16/2014 3:01	0:00:02	0:00:06	TOTAL	0:35:52	???	
23842		Haplotype Caller	377	PrepareLibrary	12/16/2014 2:26	12/16/2014 2:26	0:00:02	0:00:02			4657	
23842		Haplotype Caller	378	ImportDirectory	12/16/2014 2:26	12/16/2014 2:26	0:00:01	0:00:01			0	
23842		Haplotype Caller	379	ImportDirectory	12/16/2014 2:26	12/16/2014 2:26	0:00:03	0:00:04			49	
23842		Haplotype Caller	380	ParseCSV-d	12/16/2014 2:26	12/16/2014 2:26	0:00:02	0:00:03			158	
23842		Haplotype Caller	381	CollectWFResults	12/16/2014 2:26	12/16/2014 2:39	0:13:02	0:01:37			18766501536	
23842		Haplotype Caller	420	FileListJoin	12/16/2014 2:39	12/16/2014 2:41	0:01:38	0:01:10			18766501536	
23842		Haplotype Caller	423	PickFiles	12/16/2014 2:41	12/16/2014 2:43	0:01:36	0:00:27			18728398144	
23842		Haplotype Caller	431	GATK-HaplotypeCaller	12/16/2014 2:43	12/16/2014 3:02	0:19:28	0:00:30			2470320	
23842		Haplotype Caller	474	ExportFiles	12/16/2014 3:02	12/16/2014 3:02	0:00:02	0:00:06	TOTAL	0:35:57	???	
23843		Haplotype Caller	383	PrepareLibrary	12/16/2014 2:28	12/16/2014 2:28	0:00:02	0:00:02			4657	
23843		Haplotype Caller	384	ImportDirectory	12/16/2014 2:28	12/16/2014 2:28	0:00:01	0:00:01			0	
23843		Haplotype Caller	385	ImportDirectory	12/16/2014 2:28	12/16/2014 2:28	0:00:02	0:00:02			49	
23843		Haplotype Caller	386	ParseCSV-d	12/16/2014 2:28	12/16/2014 2:28	0:00:01	0:00:01			158	
23843		Haplotype Caller	387	CollectWFResults	12/16/2014 2:28	12/16/2014 2:41	0:12:53	0:01:41			18766501536	
23843		Haplotype Caller	422	FileListJoin	12/16/2014 2:41	12/16/2014 2:43	0:02:06	0:00:01			18766501536	
23843		Haplotype Caller	432	PickFiles	12/16/2014 2:43	12/16/2014 2:45	0:01:37	0:00:51			18728398144	
23843		Haplotype Caller	440	GATK-HaplotypeCaller	12/16/2014 2:45	12/16/2014 3:03	0:18:08	0:00:06			2613790	
23843		Haplotype Caller	480	ExportFiles	12/16/2014 3:03	12/16/2014 3:03	0:00:01	0:00:04	TOTAL	0:34:53	???	
23844		Haplotype Caller	389	PrepareLibrary	12/16/2014 2:30	12/16/2014 2:30	0:00:01	0:00:02			4657	
23844		Haplotype Caller	390	ImportDirectory	12/16/2014 2:30	12/16/2014 2:30	0:00:03	0:00:03			0	
23844		Haplotype Caller	391	ImportDirectory	12/16/2014 2:30	12/16/2014 2:30	0:00:03	0:00:03			49	
23844		Haplotype Caller	392	ParseCSV-d	12/16/2014 2:30	12/16/2014 2:30	0:00:02	0:00:02			158	
23844		Haplotype Caller	393	CollectWFResults	12/16/2014 2:30	12/16/2014 2:43	0:12:32	0:01:23			18766501536	
23844		Haplotype Caller	430	FileListJoin	12/16/2014 2:43	12/16/2014 2:44	0:01:37	0:00:02			18766501536	
23844		Haplotype Caller	439	PickFiles	12/16/2014 2:44	12/16/2014 2:46	0:01:36	0:00:32			18728398144	
23844		Haplotype Caller	448	GATK-HaplotypeCaller	12/16/2014 2:46	12/16/2014 3:04	0:18:06	0:01:31			2243047	
23844		Haplotype Caller	487	ExportFiles	12/16/2014 3:04	12/16/2014 3:04	0:00:03	0:00:07	TOTAL	0:34:05	???	
23845		Haplotype Caller	395	PrepareLibrary	12/16/2014 2:32	12/16/2014 2:32	0:00:01	0:00:01			4657	
23845		Haplotype Caller	396	ImportDirectory	12/16/2014 2:32	12/16/2014 2:32	0:00:02	0:00:02			0	
23845		Haplotype Caller	397	ImportDirectory	12/16/2014 2:32	12/16/2014 2:32	0:00:02	0:00:02			49	
23845		Haplotype Caller	398	ParseCSV-d	12/16/2014 2:32	12/16/2014 2:32	0:00:01	0:00:01			158	
23845		Haplotype Caller	399	CollectWFResults	12/16/2014 2:32	12/16/2014 2:46	0:14:00	0:00:43			18766501536	
23845		Haplotype Caller	447	FileListJoin	12/16/2014 2:46	12/16/2014 2:47	0:01:41	0:00:11			18766501536	
23845		Haplotype Caller	449	PickFiles	12/16/2014 2:47	12/16/2014 2:49	0:01:49	0:00:02			18728398144	
23845		Haplotype Caller	454	GATK-HaplotypeCaller	12/16/2014 2:49	12/16/2014 3:10	0:20:49	0:00:03			3442306	
23845		Haplotype Caller	505	ExportFiles	12/16/2014 3:10	12/16/2014 3:10	0:00:01	0:00:06	TOTAL	0:38:30	???	
23846		Haplotype Caller	401	PrepareLibrary	12/16/2014 2:33	12/16/2014 2:33	0:00:01	0:00:01			4657	
23846		Haplotype Caller	402	ImportDirectory	12/16/2014 2:33	12/16/2014 2:33	0:00:03	0:00:03			0	
23846		Haplotype Caller	403	ImportDirectory	12/16/2014 2:33	12/16/2014 2:33	0:00:04	0:00:04			49	
23846		Haplotype Caller	404	ParseCSV-d	12/16/2014 2:33	12/16/2014 2:33	0:00:03	0:00:03			158	
23846		Haplotype Caller	405	CollectWFResults	12/16/2014 2:33	12/16/2014 2:47	0:14:36	0:01:12			18766501536	
23846		Haplotype Caller	451	FileListJoin	12/16/2014 2:47	12/16/2014 2:49	0:01:47	0:00:43			18766501536	
23846		Haplotype Caller	455	PickFiles	12/16/2014 2:49	12/16/2014 2:51	0:01:42	0:01:00			18728398144	

23846		Haplotype Caller	458	GATK-HaplotypeCaller	12/16/2014 2:51	12/16/2014 3:11	0:19:42	0:01:14			3462913	
23846		Haplotype Caller	511	ExportFiles	12/16/2014 3:11	12/16/2014 3:11	0:00:03	0:00:07	TOTAL	0:38:01	???	
23847		Haplotype Caller	407	PrepareLibrary	12/16/2014 2:34	12/16/2014 2:34	0:00:01	0:00:01			4657	
23847		Haplotype Caller	408	ImportDirectory	12/16/2014 2:34	12/16/2014 2:34	0:00:02	0:00:02			0	
23847		Haplotype Caller	409	ImportDirectory	12/16/2014 2:34	12/16/2014 2:34	0:00:02	0:00:02			49	
23847		Haplotype Caller	410	ParseCSV-d	12/16/2014 2:34	12/16/2014 2:34	0:00:01	0:00:01			158	
23847		Haplotype Caller	411	CollectWFResults	12/16/2014 2:34	12/16/2014 2:48	0:13:55	0:00:12			18766501536	
23847		Haplotype Caller	452	FileListJoin	12/16/2014 2:48	12/16/2014 2:50	0:02:04	0:00:51			18766501536	
23847		Haplotype Caller	456	PickFiles	12/16/2014 2:50	12/16/2014 2:52	0:01:55	0:00:22			18728398144	
23847		Haplotype Caller	459	GATK-HaplotypeCaller	12/16/2014 2:52	12/16/2014 3:08	0:16:11	0:02:39			2070713	
23847		Haplotype Caller	499	ExportFiles	12/16/2014 3:08	12/16/2014 3:08	0:00:02	0:00:09	TOTAL	0:34:14	???	
23848		Haplotype Caller	413	PrepareLibrary	12/16/2014 2:35	12/16/2014 2:35	0:00:02	0:00:02			4657	
23848		Haplotype Caller	414	ImportDirectory	12/16/2014 2:35	12/16/2014 2:35	0:00:02	0:00:02			0	
23848		Haplotype Caller	415	ImportDirectory	12/16/2014 2:35	12/16/2014 2:35	0:00:02	0:00:02			49	
23848		Haplotype Caller	416	ParseCSV-d	12/16/2014 2:35	12/16/2014 2:35	0:00:01	0:00:02			158	
23848		Haplotype Caller	417	CollectWFResults	12/16/2014 2:35	12/16/2014 2:47	0:12:45	0:02:43			18766501536	
23848		Haplotype Caller	450	FileListJoin	12/16/2014 2:47	12/16/2014 2:49	0:01:36	0:00:03			18766501536	
23848		Haplotype Caller	453	PickFiles	12/16/2014 2:49	12/16/2014 2:51	0:01:35	0:00:10			18728398144	
23848		Haplotype Caller	457	GATK-HaplotypeCaller	12/16/2014 2:51	12/16/2014 3:07	0:16:17	0:00:20			2844295	
23848		Haplotype Caller	493	ExportFiles	12/16/2014 3:07	12/16/2014 3:07	0:00:03	0:00:05	TOTAL	0:32:26	???	
23849		Haplotype Caller	425	PrepareLibrary	12/16/2014 2:42	12/16/2014 2:42	0:00:01	0:00:01			4657	
23849		Haplotype Caller	426	ImportDirectory	12/16/2014 2:42	12/16/2014 2:42	0:00:03	0:00:03			0	
23849		Haplotype Caller	427	ImportDirectory	12/16/2014 2:42	12/16/2014 2:42	0:00:02	0:00:02			49	
23849		Haplotype Caller	428	ParseCSV-d	12/16/2014 2:42	12/16/2014 2:42	0:00:01	0:00:01			158	
23849		Haplotype Caller	429	CollectWFResults	12/16/2014 2:42	12/16/2014 2:55	0:13:06	0:00:55			18766501536	
23849		Haplotype Caller	460	FileListJoin	12/16/2014 2:55	12/16/2014 2:57	0:01:44	0:01:44			18766501536	
23849		Haplotype Caller	461	PickFiles	12/16/2014 2:57	12/16/2014 2:58	0:01:43	0:00:20			18728398144	
23849		Haplotype Caller	464	GATK-HaplotypeCaller	12/16/2014 2:58	12/16/2014 3:23	0:24:14	0:01:04			3405218	
23849		Haplotype Caller	539	ExportFiles	12/16/2014 3:23	12/16/2014 3:23	0:00:02	0:00:05	TOTAL	0:40:57	???	
23850		Haplotype Caller	434	PrepareLibrary	12/16/2014 2:44	12/16/2014 2:44	0:00:01	0:00:01			4657	
23850		Haplotype Caller	435	ImportDirectory	12/16/2014 2:44	12/16/2014 2:44	0:00:03	0:00:03			0	
23850		Haplotype Caller	436	ImportDirectory	12/16/2014 2:44	12/16/2014 2:44	0:00:01	0:00:02			49	
23850		Haplotype Caller	437	ParseCSV-d	12/16/2014 2:44	12/16/2014 2:44	0:00:01	0:00:01			158	
23850		Haplotype Caller	438	CollectWFResults	12/16/2014 2:44	12/16/2014 2:57	0:12:42	0:00:03			18766501536	
23850		Haplotype Caller	462	FileListJoin	12/16/2014 2:57	12/16/2014 2:59	0:02:27	0:00:34			18766501536	
23850		Haplotype Caller	465	PickFiles	12/16/2014 2:59	12/16/2014 3:02	0:02:45	0:00:40			18728398144	
23850		Haplotype Caller	473	GATK-HaplotypeCaller	12/16/2014 3:02	12/16/2014 3:39	0:37:00	0:00:02			4103700	
23850		Haplotype Caller	567	ExportFiles	12/16/2014 3:39	12/16/2014 3:39	0:00:02	0:00:02	TOTAL	0:55:03	???	
23851		Haplotype Caller	442	PrepareLibrary	12/16/2014 2:45	12/16/2014 2:45	0:00:02	0:00:02			4657	
23851		Haplotype Caller	443	ImportDirectory	12/16/2014 2:45	12/16/2014 2:45	0:00:01	0:00:01			0	
23851		Haplotype Caller	444	ImportDirectory	12/16/2014 2:45	12/16/2014 2:45	0:00:02	0:00:02			49	
23851		Haplotype Caller	445	ParseCSV-d	12/16/2014 2:45	12/16/2014 2:45	0:00:02	0:00:02			158	
23851		Haplotype Caller	446	CollectWFResults	12/16/2014 2:45	12/16/2014 2:57	0:12:20	0:00:34			18766501536	
23851		Haplotype Caller	463	FileListJoin	12/16/2014 2:57	12/16/2014 3:00	0:02:30	0:00:49			18766501536	
23851		Haplotype Caller	466	PickFiles	12/16/2014 3:00	12/16/2014 3:03	0:02:59	0:00:46			18728398144	

23851		Haplotype Caller	481	GATK-HaplotypeCaller	12/16/2014 3:03	12/16/2014 3:24	0:20:33	0:00:01			3181198	
23851		Haplotype Caller	546	ExportFiles	12/16/2014 3:24	12/16/2014 3:24	0:00:02	0:00:04	TOTAL	0:38:35	???	
23852		Haplotype Caller	468	PrepareLibrary	12/16/2014 3:01	12/16/2014 3:01	0:00:02	0:00:02			4657	
23852		Haplotype Caller	469	ImportDirectory	12/16/2014 3:01	12/16/2014 3:01	0:00:02	0:00:03			0	
23852		Haplotype Caller	470	ImportDirectory	12/16/2014 3:01	12/16/2014 3:01	0:00:02	0:00:02			49	
23852		Haplotype Caller	471	ParseCSV-d	12/16/2014 3:01	12/16/2014 3:01	0:00:02	0:00:02			158	
23852		Haplotype Caller	472	CollectWFResults	12/16/2014 3:01	12/16/2014 3:13	0:12:23	0:01:04			18766501536	
23852		Haplotype Caller	517	FileListJoin	12/16/2014 3:13	12/16/2014 3:15	0:01:37	0:01:37			18766501536	
23852		Haplotype Caller	518	PickFiles	12/16/2014 3:15	12/16/2014 3:17	0:01:36	0:00:03			18728398144	
23852		Haplotype Caller	521	GATK-HaplotypeCaller	12/16/2014 3:17	12/16/2014 3:53	0:36:27	0:00:03			4253654	
23852		Haplotype Caller	606	ExportFiles	12/16/2014 3:53	12/16/2014 3:53	0:00:02	0:00:04	TOTAL	0:52:14	???	
23853		Haplotype Caller	475	PrepareLibrary	12/16/2014 3:02	12/16/2014 3:02	0:00:01	0:00:01			4657	
23853		Haplotype Caller	476	ImportDirectory	12/16/2014 3:02	12/16/2014 3:02	0:00:01	0:00:02			0	
23853		Haplotype Caller	477	ImportDirectory	12/16/2014 3:02	12/16/2014 3:02	0:00:01	0:00:02			49	
23853		Haplotype Caller	478	ParseCSV-d	12/16/2014 3:02	12/16/2014 3:02	0:00:01	0:00:01			158	
23853		Haplotype Caller	479	CollectWFResults	12/16/2014 3:02	12/16/2014 3:15	0:12:44	0:00:36			18766501536	
23853		Haplotype Caller	519	FileListJoin	12/16/2014 3:15	12/16/2014 3:17	0:01:36	0:01:28			18766501536	
23853		Haplotype Caller	522	PickFiles	12/16/2014 3:17	12/16/2014 3:18	0:01:38	0:00:07			18728398144	
23853		Haplotype Caller	525	GATK-HaplotypeCaller	12/16/2014 3:18	12/16/2014 3:41	0:22:55	0:00:22			2931491	
23853		Haplotype Caller	574	ExportFiles	12/16/2014 3:41	12/16/2014 3:41	0:00:01	0:00:06	TOTAL	0:39:03	???	
23854		Haplotype Caller	482	PrepareLibrary	12/16/2014 3:03	12/16/2014 3:03	0:00:02	0:00:02			4657	
23854		Haplotype Caller	483	ImportDirectory	12/16/2014 3:03	12/16/2014 3:03	0:00:03	0:00:03			0	
23854		Haplotype Caller	484	ImportDirectory	12/16/2014 3:03	12/16/2014 3:03	0:00:02	0:00:02			49	
23854		Haplotype Caller	485	ParseCSV-d	12/16/2014 3:03	12/16/2014 3:03	0:00:01	0:00:01			158	
23854		Haplotype Caller	486	CollectWFResults	12/16/2014 3:03	12/16/2014 3:17	0:13:39	0:00:49			18766501536	
23854		Haplotype Caller	523	FileListJoin	12/16/2014 3:17	12/16/2014 3:19	0:02:36	0:01:23			18766501536	
23854		Haplotype Caller	533	PickFiles	12/16/2014 3:19	12/16/2014 3:22	0:02:11	0:00:22			18728398144	
23854		Haplotype Caller	537	GATK-HaplotypeCaller	12/16/2014 3:22	12/16/2014 4:05	0:43:17	0:00:47			3923740	
23854		Haplotype Caller	647	ExportFiles	12/16/2014 4:05	12/16/2014 4:05	0:00:03	0:00:04	TOTAL	1:01:57	???	
23855		Haplotype Caller	488	PrepareLibrary	12/16/2014 3:04	12/16/2014 3:04	0:00:01	0:00:01			4657	
23855		Haplotype Caller	489	ImportDirectory	12/16/2014 3:04	12/16/2014 3:04	0:00:02	0:00:02			0	
23855		Haplotype Caller	490	ImportDirectory	12/16/2014 3:04	12/16/2014 3:04	0:00:02	0:00:02			49	
23855		Haplotype Caller	491	ParseCSV-d	12/16/2014 3:04	12/16/2014 3:04	0:00:01	0:00:01			158	
23855		Haplotype Caller	492	CollectWFResults	12/16/2014 3:04	12/16/2014 3:17	0:12:21	0:02:44			18766501536	
23855		Haplotype Caller	520	FileListJoin	12/16/2014 3:17	12/16/2014 3:18	0:01:37	0:00:05			18766501536	
23855		Haplotype Caller	524	PickFiles	12/16/2014 3:18	12/16/2014 3:20	0:01:35	0:00:09			18728398144	
23855		Haplotype Caller	534	GATK-HaplotypeCaller	12/16/2014 3:20	12/16/2014 3:43	0:23:06	0:00:49			4996005	
23855		Haplotype Caller	580	ExportFiles	12/16/2014 3:43	12/16/2014 3:43	0:00:02	0:00:04	TOTAL	0:38:51	???	
23856		Haplotype Caller	494	PrepareLibrary	12/16/2014 3:07	12/16/2014 3:07	0:00:02	0:00:02			4657	
23856		Haplotype Caller	495	ImportDirectory	12/16/2014 3:07	12/16/2014 3:07	0:00:02	0:00:02			0	
23856		Haplotype Caller	496	ImportDirectory	12/16/2014 3:07	12/16/2014 3:07	0:00:02	0:00:02			49	
23856		Haplotype Caller	497	ParseCSV-d	12/16/2014 3:07	12/16/2014 3:07	0:00:01	0:00:01			158	
23856		Haplotype Caller	498	CollectWFResults	12/16/2014 3:07	12/16/2014 3:19	0:11:34	0:01:15			18766501536	
23856		Haplotype Caller	526	FileListJoin	12/16/2014 3:19	12/16/2014 3:21	0:01:54	0:00:17			18766501536	
23856		Haplotype Caller	535	PickFiles	12/16/2014 3:21	12/16/2014 3:23	0:02:00	0:00:12			18728398144	

23856		Haplotype Caller	540	GATK-HaplotypeCaller	12/16/2014 3:23	12/16/2014 3:45	0:22:36	0:00:00			4417900	
23856		Haplotype Caller	586	ExportFiles	12/16/2014 3:45	12/16/2014 3:45	0:00:02	0:00:04	TOTAL	0:38:14	???	
23857		Haplotype Caller	500	PrepareLibrary	12/16/2014 3:09	12/16/2014 3:09	0:00:02	0:00:02			4657	
23857		Haplotype Caller	501	ImportDirectory	12/16/2014 3:09	12/16/2014 3:09	0:00:02	0:00:03			0	
23857		Haplotype Caller	502	ImportDirectory	12/16/2014 3:09	12/16/2014 3:09	0:00:01	0:00:03			49	
23857		Haplotype Caller	503	ParseCSV-d	12/16/2014 3:09	12/16/2014 3:09	0:00:01	0:00:02			158	
23857		Haplotype Caller	504	CollectWFResults	12/16/2014 3:09	12/16/2014 3:21	0:12:07	0:01:21			18766501536	
23857		Haplotype Caller	536	FileListJoin	12/16/2014 3:21	12/16/2014 3:22	0:01:36	0:00:49			18766501536	
23857		Haplotype Caller	538	PickFiles	12/16/2014 3:22	12/16/2014 3:24	0:01:37	0:00:07			18728398144	
23857		Haplotype Caller	552	GATK-HaplotypeCaller	12/16/2014 3:24	12/16/2014 3:53	0:28:48	0:00:23			4660487	
23857		Haplotype Caller	599	ExportFiles	12/16/2014 3:53	12/16/2014 3:53	0:00:03	0:00:03	TOTAL	0:44:21	???	
23858		Haplotype Caller	506	PrepareLibrary	12/16/2014 3:10	12/16/2014 3:10	0:00:01	0:00:01			4657	
23858		Haplotype Caller	507	ImportDirectory	12/16/2014 3:10	12/16/2014 3:10	0:00:02	0:00:03			0	
23858		Haplotype Caller	508	ImportDirectory	12/16/2014 3:10	12/16/2014 3:10	0:00:04	0:00:04			49	
23858		Haplotype Caller	509	ParseCSV-d	12/16/2014 3:10	12/16/2014 3:10	0:00:01	0:00:01			158	
23858		Haplotype Caller	510	CollectWFResults	12/16/2014 3:10	12/16/2014 3:24	0:14:07	0:00:22			18766501536	
23858		Haplotype Caller	553	FileListJoin	12/16/2014 3:24	12/16/2014 3:26	0:02:04	0:00:17			18766501536	
23858		Haplotype Caller	555	PickFiles	12/16/2014 3:26	12/16/2014 3:28	0:01:40	0:00:13			18728398144	
23858		Haplotype Caller	557	GATK-HaplotypeCaller	12/16/2014 3:28	12/16/2014 3:54	0:25:53	0:00:35			5656653	
23858		Haplotype Caller	613	ExportFiles	12/16/2014 3:54	12/16/2014 3:54	0:00:02	0:00:04	TOTAL	0:43:56	???	
23859		Haplotype Caller	512	PrepareLibrary	12/16/2014 3:11	12/16/2014 3:11	0:00:02	0:00:02			4657	
23859		Haplotype Caller	513	ImportDirectory	12/16/2014 3:11	12/16/2014 3:11	0:00:02	0:00:02			0	
23859		Haplotype Caller	514	ImportDirectory	12/16/2014 3:11	12/16/2014 3:11	0:00:01	0:00:01			49	
23859		Haplotype Caller	515	ParseCSV-d	12/16/2014 3:11	12/16/2014 3:11	0:00:01	0:00:01			158	
23859		Haplotype Caller	516	CollectWFResults	12/16/2014 3:11	12/16/2014 3:25	0:13:49	0:02:33			18766501536	
23859		Haplotype Caller	554	FileListJoin	12/16/2014 3:25	12/16/2014 3:27	0:01:59	0:01:47			18766501536	
23859		Haplotype Caller	556	PickFiles	12/16/2014 3:27	12/16/2014 3:29	0:02:01	0:01:27			18728398144	
23859		Haplotype Caller	558	GATK-HaplotypeCaller	12/16/2014 3:29	12/16/2014 3:57	0:27:54	0:02:42			4096182	
23859		Haplotype Caller	625	ExportFiles	12/16/2014 3:57	12/16/2014 3:57	0:00:02	0:0:-2	TOTAL	0:45:54	???	
23860		Haplotype Caller	528	PrepareLibrary	12/16/2014 3:19	12/16/2014 3:19	0:00:01	0:00:01			4657	
23860		Haplotype Caller	529	ImportDirectory	12/16/2014 3:19	12/16/2014 3:19	0:00:01	0:00:01			0	
23860		Haplotype Caller	530	ImportDirectory	12/16/2014 3:19	12/16/2014 3:19	0:00:02	0:00:02			49	
23860		Haplotype Caller	531	ParseCSV-d	12/16/2014 3:19	12/16/2014 3:19	0:00:01	0:00:01			158	
23860		Haplotype Caller	532	CollectWFResults	12/16/2014 3:19	12/16/2014 3:31	0:12:18	0:00:18			18766501536	
23860		Haplotype Caller	559	FileListJoin	12/16/2014 3:31	12/16/2014 3:33	0:01:51	0:01:52			18766501536	
23860		Haplotype Caller	560	PickFiles	12/16/2014 3:33	12/16/2014 3:35	0:01:50	0:01:12			18728398144	
23860		Haplotype Caller	562	GATK-HaplotypeCaller	12/16/2014 3:35	12/16/2014 4:03	0:27:26	0:00:24			5255692	
23860		Haplotype Caller	640	ExportFiles	12/16/2014 4:03	12/16/2014 4:03	0:00:04	0:00:10	TOTAL	0:43:35	???	
23861		Haplotype Caller	541	PrepareLibrary	12/16/2014 3:23	12/16/2014 3:23	0:00:02	0:00:02			4657	
23861		Haplotype Caller	542	ImportDirectory	12/16/2014 3:23	12/16/2014 3:23	0:00:02	0:00:02			0	
23861		Haplotype Caller	543	ImportDirectory	12/16/2014 3:23	12/16/2014 3:23	0:00:02	0:00:02			49	
23861		Haplotype Caller	544	ParseCSV-d	12/16/2014 3:23	12/16/2014 3:23	0:00:02	0:00:02			158	
23861		Haplotype Caller	545	CollectWFResults	12/16/2014 3:23	12/16/2014 3:34	0:11:45	0:00:49			18766501536	
23861		Haplotype Caller	561	FileListJoin	12/16/2014 3:34	12/16/2014 3:36	0:01:57	0:00:38			18766501536	
23861		Haplotype Caller	564	PickFiles	12/16/2014 3:36	12/16/2014 3:38	0:01:50	0:01:00			18728398144	

23861		Haplotype Caller	566	GATK-HaplotypeCaller	12/16/2014 3:38	12/16/2014 3:56	0:18:13	0:00:50			3768985	
23861		Haplotype Caller	624	ExportFiles	12/16/2014 3:56	12/16/2014 3:57	0:00:03	0:00:09	TOTAL	0:33:56	???	
23862		Haplotype Caller	547	PrepareLibrary	12/16/2014 3:24	12/16/2014 3:24	0:00:02	0:00:02			4657	
23862		Haplotype Caller	548	ImportDirectory	12/16/2014 3:24	12/16/2014 3:24	0:00:02	0:00:02			0	
23862		Haplotype Caller	549	ImportDirectory	12/16/2014 3:24	12/16/2014 3:24	0:00:02	0:00:02			49	
23862		Haplotype Caller	550	ParseCSV-d	12/16/2014 3:24	12/16/2014 3:24	0:00:01	0:00:02			158	
23862		Haplotype Caller	551	CollectWFResults	12/16/2014 3:24	12/16/2014 3:36	0:11:46	0:00:16			18766501536	
23862		Haplotype Caller	563	FileListJoin	12/16/2014 3:36	12/16/2014 3:37	0:01:55	0:00:55			18766501536	
23862		Haplotype Caller	565	PickFiles	12/16/2014 3:37	12/16/2014 3:39	0:01:42	0:00:50			18728398144	
23862		Haplotype Caller	568	GATK-HaplotypeCaller	12/16/2014 3:39	12/16/2014 3:51	0:12:10	0:00:01			2690337	
23862		Haplotype Caller	592	ExportFiles	12/16/2014 3:51	12/16/2014 3:51	0:00:02	0:00:06	TOTAL	0:27:43	???	
23863		Haplotype Caller	569	PrepareLibrary	12/16/2014 3:39	12/16/2014 3:39	0:00:02	0:00:02			4657	
23863		Haplotype Caller	570	ImportDirectory	12/16/2014 3:39	12/16/2014 3:39	0:00:02	0:00:02			0	
23863		Haplotype Caller	571	ImportDirectory	12/16/2014 3:39	12/16/2014 3:39	0:00:02	0:00:02			49	
23863		Haplotype Caller	572	ParseCSV-d	12/16/2014 3:39	12/16/2014 3:39	0:00:01	0:00:01			158	
23863		Haplotype Caller	573	CollectWFResults	12/16/2014 3:39	12/16/2014 3:52	0:12:41	0:02:00			18766501536	
23863		Haplotype Caller	598	FileListJoin	12/16/2014 3:52	12/16/2014 3:54	0:01:36	0:00:52			18766501536	
23863		Haplotype Caller	612	PickFiles	12/16/2014 3:54	12/16/2014 3:55	0:01:35	0:00:29			18728398144	
23863		Haplotype Caller	621	GATK-HaplotypeCaller	12/16/2014 3:55	12/16/2014 4:06	0:11:16	0:00:57			1877908	
23863		Haplotype Caller	656	ExportFiles	12/16/2014 4:06	12/16/2014 4:06	0:00:02	0:00:06	TOTAL	0:27:17	???	
23864		Haplotype Caller	575	PrepareLibrary	12/16/2014 3:41	12/16/2014 3:41	0:00:01	0:00:01			4657	
23864		Haplotype Caller	576	ImportDirectory	12/16/2014 3:41	12/16/2014 3:41	0:00:02	0:00:03			0	
23864		Haplotype Caller	577	ImportDirectory	12/16/2014 3:41	12/16/2014 3:41	0:00:02	0:00:02			49	
23864		Haplotype Caller	578	ParseCSV-d	12/16/2014 3:41	12/16/2014 3:41	0:00:01	0:00:01			158	
23864		Haplotype Caller	579	CollectWFResults	12/16/2014 3:41	12/16/2014 3:53	0:11:32	0:01:25			18766501536	
23864		Haplotype Caller	605	FileListJoin	12/16/2014 3:53	12/16/2014 3:55	0:01:42	0:00:04			18766501536	
23864		Haplotype Caller	620	PickFiles	12/16/2014 3:55	12/16/2014 3:56	0:01:36	0:00:25			18728398144	
23864		Haplotype Caller	623	GATK-HaplotypeCaller	12/16/2014 3:56	12/16/2014 4:13	0:16:31	0:00:09			1959426	
23864		Haplotype Caller	674	ExportFiles	12/16/2014 4:13	12/16/2014 4:13	0:00:02	0:00:04	TOTAL	0:31:32	???	
23865		Haplotype Caller	581	PrepareLibrary	12/16/2014 3:43	12/16/2014 3:43	0:00:02	0:00:03			4657	
23865		Haplotype Caller	582	ImportDirectory	12/16/2014 3:43	12/16/2014 3:43	0:00:02	0:00:02			0	
23865		Haplotype Caller	583	ImportDirectory	12/16/2014 3:43	12/16/2014 3:43	0:00:02	0:00:02			49	
23865		Haplotype Caller	584	ParseCSV-d	12/16/2014 3:43	12/16/2014 3:43	0:00:02	0:00:02			158	
23865		Haplotype Caller	585	CollectWFResults	12/16/2014 3:43	12/16/2014 3:54	0:11:20	0:02:05			18766501536	
23865		Haplotype Caller	619	FileListJoin	12/16/2014 3:54	12/16/2014 3:56	0:01:37	0:00:16			18766501536	
23865		Haplotype Caller	622	PickFiles	12/16/2014 3:56	12/16/2014 3:58	0:01:37	0:00:15			18728398144	
23865		Haplotype Caller	637	GATK-HaplotypeCaller	12/16/2014 3:58	12/16/2014 4:16	0:18:21	0:01:21			3495260	
23865		Haplotype Caller	683	ExportFiles	12/16/2014 4:16	12/16/2014 4:16	0:00:02	0:00:03	TOTAL	0:33:08	???	
23866		Haplotype Caller	587	PrepareLibrary	12/16/2014 3:45	12/16/2014 3:45	0:00:01	0:00:01			4657	
23866		Haplotype Caller	588	ImportDirectory	12/16/2014 3:45	12/16/2014 3:45	0:00:01	0:00:02			0	
23866		Haplotype Caller	589	ImportDirectory	12/16/2014 3:45	12/16/2014 3:45	0:00:02	0:00:03			49	
23866		Haplotype Caller	590	ParseCSV-d	12/16/2014 3:45	12/16/2014 3:45	0:00:01	0:00:01			158	
23866		Haplotype Caller	591	CollectWFResults	12/16/2014 3:45	12/16/2014 3:57	0:12:04	0:05:55			18766501536	
23866		Haplotype Caller	636	FileListJoin	12/16/2014 3:57	12/16/2014 3:59	0:01:36	0:00:15			18766501536	
23866		Haplotype Caller	638	PickFiles	12/16/2014 3:59	12/16/2014 4:01	0:01:35	0:01:37			18728398144	

23866		Haplotype Caller	639	GATK-HaplotypeCaller	12/16/2014 4:01	12/16/2014 4:31	0:29:51	0:01:52			2966339	
23866		Haplotype Caller	701	ExportFiles	12/16/2014 4:31	12/16/2014 4:31	0:00:02	0:00:34	TOTAL	0:45:19	???	
23867		Haplotype Caller	593	PrepareLibrary	12/16/2014 3:51	12/16/2014 3:51	0:00:01	0:00:01			4657	
23867		Haplotype Caller	594	ImportDirectory	12/16/2014 3:51	12/16/2014 3:51	0:00:02	0:00:02			0	
23867		Haplotype Caller	595	ImportDirectory	12/16/2014 3:51	12/16/2014 3:51	0:00:02	0:00:02			49	
23867		Haplotype Caller	596	ParseCSV-d	12/16/2014 3:51	12/16/2014 3:52	0:00:01	0:00:01			158	
23867		Haplotype Caller	597	CollectWFResults	12/16/2014 3:52	12/16/2014 4:04	0:12:21	0:00:27			18766501536	
23867		Haplotype Caller	646	FileListJoin	12/16/2014 4:04	12/16/2014 4:06	0:01:38	0:01:03			18766501536	
23867		Haplotype Caller	653	PickFiles	12/16/2014 4:06	12/16/2014 4:07	0:01:37	0:00:27			18728398144	
23867		Haplotype Caller	662	GATK-HaplotypeCaller	12/16/2014 4:07	12/16/2014 4:32	0:24:30	0:00:25			4038244	
23867		Haplotype Caller	703	ExportFiles	12/16/2014 4:32	12/16/2014 4:32	0:00:01	0:00:36	TOTAL	0:40:15	???	
23868		Haplotype Caller	600	PrepareLibrary	12/16/2014 3:53	12/16/2014 3:53	0:00:02	0:00:02			4657	
23868		Haplotype Caller	601	ImportDirectory	12/16/2014 3:53	12/16/2014 3:53	0:00:01	0:00:01			0	
23868		Haplotype Caller	602	ImportDirectory	12/16/2014 3:53	12/16/2014 3:53	0:00:02	0:00:02			49	
23868		Haplotype Caller	603	ParseCSV-d	12/16/2014 3:53	12/16/2014 3:53	0:00:01	0:00:02			158	
23868		Haplotype Caller	604	CollectWFResults	12/16/2014 3:53	12/16/2014 4:06	0:13:19	0:00:02			18766501536	
23868		Haplotype Caller	655	FileListJoin	12/16/2014 4:06	12/16/2014 4:08	0:01:37	0:00:06			18766501536	
23868		Haplotype Caller	665	PickFiles	12/16/2014 4:08	12/16/2014 4:10	0:01:37	0:01:14			18728398144	
23868		Haplotype Caller	668	GATK-HaplotypeCaller	12/16/2014 4:10	12/16/2014 4:50	0:40:46	0:00:36			7678349	
23868		Haplotype Caller	710	ExportFiles	12/16/2014 4:50	12/16/2014 4:50	0:00:02	0:01:52	TOTAL	0:57:31	???	
23869		Haplotype Caller	607	PrepareLibrary	12/16/2014 3:53	12/16/2014 3:53	0:00:01	0:00:02			4657	
23869		Haplotype Caller	608	ImportDirectory	12/16/2014 3:53	12/16/2014 3:53	0:00:01	0:00:01			0	
23869		Haplotype Caller	609	ImportDirectory	12/16/2014 3:53	12/16/2014 3:53	0:00:02	0:00:02			49	
23869		Haplotype Caller	610	ParseCSV-d	12/16/2014 3:53	12/16/2014 3:53	0:00:01	0:00:01			158	
23869		Haplotype Caller	611	CollectWFResults	12/16/2014 3:53	12/16/2014 4:06	0:12:42	0:00:18			18766501536	
23869		Haplotype Caller	654	FileListJoin	12/16/2014 4:06	12/16/2014 4:08	0:01:35	0:00:21			18766501536	
23869		Haplotype Caller	663	PickFiles	12/16/2014 4:08	12/16/2014 4:09	0:01:37	0:00:04			18728398144	
23869		Haplotype Caller	666	GATK-HaplotypeCaller	12/16/2014 4:09	12/16/2014 5:10	1:01:18	0:00:05			9054605	
23869		Haplotype Caller	714	ExportFiles	12/16/2014 5:10	12/16/2014 5:11	0:00:03	0:00:03	TOTAL	1:17:22	???	
23870		Haplotype Caller	614	PrepareLibrary	12/16/2014 3:54	12/16/2014 3:54	0:00:02	0:00:02			4657	
23870		Haplotype Caller	615	ImportDirectory	12/16/2014 3:54	12/16/2014 3:54	0:00:02	0:00:02			0	
23870		Haplotype Caller	616	ImportDirectory	12/16/2014 3:54	12/16/2014 3:54	0:00:01	0:00:01			49	
23870		Haplotype Caller	617	ParseCSV-d	12/16/2014 3:54	12/16/2014 3:54	0:00:01	0:00:02			158	
23870		Haplotype Caller	618	CollectWFResults	12/16/2014 3:54	12/16/2014 4:08	0:13:23	0:00:14			18766501536	
23870		Haplotype Caller	664	FileListJoin	12/16/2014 4:08	12/16/2014 4:10	0:02:36	0:00:20			18766501536	
23870		Haplotype Caller	670	PickFiles	12/16/2014 4:10	12/16/2014 4:12	0:01:42	0:01:26			18728398144	
23870		Haplotype Caller	673	GATK-HaplotypeCaller	12/16/2014 4:12	12/16/2014 4:38	0:26:13	0:00:58			3822407	
23870		Haplotype Caller	705	ExportFiles	12/16/2014 4:38	12/16/2014 4:38	0:00:02	0:02:07	TOTAL	0:44:03	???	
23871		Haplotype Caller	626	PrepareLibrary	12/16/2014 3:57	12/16/2014 3:57	0:00:01	0:00:01			4657	
23871		Haplotype Caller	627	ImportDirectory	12/16/2014 3:57	12/16/2014 3:57	0:00:02	0:00:02			0	
23871		Haplotype Caller	628	ImportDirectory	12/16/2014 3:57	12/16/2014 3:57	0:00:02	0:00:02			49	
23871		Haplotype Caller	629	ParseCSV-d	12/16/2014 3:57	12/16/2014 3:57	0:00:01	0:00:01			158	
23871		Haplotype Caller	630	CollectWFResults	12/16/2014 3:57	12/16/2014 4:09	0:12:33	0:00:02			18766501536	
23871		Haplotype Caller	667	FileListJoin	12/16/2014 4:09	12/16/2014 4:12	0:02:23	0:00:19			18766501536	
23871		Haplotype Caller	671	PickFiles	12/16/2014 4:12	12/16/2014 4:13	0:01:40	0:00:15			18728398144	

23871		Haplotype Caller	680	GATK-HaplotypeCaller	12/16/2014 4:13	12/16/2014 4:46	0:32:58	0:00:15			9852787	
23871		Haplotype Caller	709	ExportFiles	12/16/2014 4:46	12/16/2014 4:46	0:00:03	0:04:04	TOTAL	0:49:44	???	
23872		Haplotype Caller	631	PrepareLibrary	12/16/2014 3:57	12/16/2014 3:57	0:00:01	0:00:02			4657	
23872		Haplotype Caller	632	ImportDirectory	12/16/2014 3:57	12/16/2014 3:57	0:00:02	0:00:02			0	
23872		Haplotype Caller	633	ImportDirectory	12/16/2014 3:57	12/16/2014 3:57	0:00:03	0:00:03			49	
23872		Haplotype Caller	634	ParseCSV-d	12/16/2014 3:57	12/16/2014 3:57	0:00:02	0:00:02			158	
23872		Haplotype Caller	635	CollectWFResults	12/16/2014 3:57	12/16/2014 4:10	0:13:17	0:00:35			18766501536	
23872		Haplotype Caller	669	FileListJoin	12/16/2014 4:10	12/16/2014 4:12	0:01:42	0:00:02			18766501536	
23872		Haplotype Caller	672	PickFiles	12/16/2014 4:12	12/16/2014 4:14	0:01:40	0:00:01			18728398144	
23872		Haplotype Caller	681	GATK-HaplotypeCaller	12/16/2014 4:14	12/16/2014 4:52	0:38:39	0:02:28			4695939	
23872		Haplotype Caller	711	ExportFiles	12/16/2014 4:52	12/16/2014 4:52	0:00:03	0:02:15	TOTAL	0:55:32	???	
23873		Haplotype Caller	641	PrepareLibrary	12/16/2014 4:03	12/16/2014 4:03	0:00:01	0:00:01			4657	
23873		Haplotype Caller	642	ImportDirectory	12/16/2014 4:03	12/16/2014 4:03	0:00:02	0:00:03			0	
23873		Haplotype Caller	643	ImportDirectory	12/16/2014 4:03	12/16/2014 4:03	0:00:02	0:00:02			49	
23873		Haplotype Caller	644	ParseCSV-d	12/16/2014 4:03	12/16/2014 4:03	0:00:01	0:00:02			158	
23873		Haplotype Caller	645	CollectWFResults	12/16/2014 4:03	12/16/2014 4:16	0:13:11	0:01:01			18766501536	
23873		Haplotype Caller	682	FileListJoin	12/16/2014 4:16	12/16/2014 4:18	0:01:40	0:00:02			18766501536	
23873		Haplotype Caller	689	PickFiles	12/16/2014 4:18	12/16/2014 4:19	0:01:40	0:00:30			18728398144	
23873		Haplotype Caller	691	GATK-HaplotypeCaller	12/16/2014 4:19	12/16/2014 4:40	0:21:00	0:00:05			3454972	
23873		Haplotype Caller	707	ExportFiles	12/16/2014 4:40	12/16/2014 4:40	0:00:02	0:02:41	TOTAL	0:37:43	???	
23874		Haplotype Caller	648	PrepareLibrary	12/16/2014 4:05	12/16/2014 4:05	0:00:01	0:00:01			4657	
23874		Haplotype Caller	649	ImportDirectory	12/16/2014 4:05	12/16/2014 4:05	0:00:02	0:00:02			0	
23874		Haplotype Caller	650	ImportDirectory	12/16/2014 4:05	12/16/2014 4:05	0:00:03	0:00:03			49	
23874		Haplotype Caller	651	ParseCSV-d	12/16/2014 4:05	12/16/2014 4:05	0:00:01	0:00:01			158	
23874		Haplotype Caller	652	CollectWFResults	12/16/2014 4:05	12/16/2014 4:18	0:13:06	0:00:24			18766501536	
23874		Haplotype Caller	690	FileListJoin	12/16/2014 4:18	12/16/2014 4:20	0:01:56	0:01:10			18766501536	
23874		Haplotype Caller	693	PickFiles	12/16/2014 4:20	12/16/2014 4:22	0:01:39	0:01:36			18728398144	
23874		Haplotype Caller	695	GATK-HaplotypeCaller	12/16/2014 4:22	12/16/2014 5:05	0:42:59	0:01:54			4674642	
23874		Haplotype Caller	713	ExportFiles	12/16/2014 5:05	12/16/2014 5:05	0:00:02	0:05:39	TOTAL	0:59:52	???	
23875		Haplotype Caller	657	PrepareLibrary	12/16/2014 4:07	12/16/2014 4:07	0:00:01	0:00:01			4657	
23875		Haplotype Caller	658	ImportDirectory	12/16/2014 4:07	12/16/2014 4:07	0:00:02	0:00:02			0	
23875		Haplotype Caller	659	ImportDirectory	12/16/2014 4:07	12/16/2014 4:07	0:00:01	0:00:01			49	
23875		Haplotype Caller	660	ParseCSV-d	12/16/2014 4:07	12/16/2014 4:07	0:00:01	0:00:02			158	
23875		Haplotype Caller	661	CollectWFResults	12/16/2014 4:07	12/16/2014 4:19	0:12:52	0:00:31			18766501536	
23875		Haplotype Caller	692	FileListJoin	12/16/2014 4:19	12/16/2014 4:22	0:02:18	0:00:42			18766501536	
23875		Haplotype Caller	694	PickFiles	12/16/2014 4:22	12/16/2014 4:24	0:01:57	0:00:04			18728398144	
23875		Haplotype Caller	696	GATK-HaplotypeCaller	12/16/2014 4:24	12/16/2014 4:43	0:19:20	0:02:20			1703470	
23875		Haplotype Caller	708	ExportFiles	12/16/2014 4:43	12/16/2014 4:43	0:00:02	0:03:12	TOTAL	0:36:37	???	
23876		Haplotype Caller	675	PrepareLibrary	12/16/2014 4:13	12/16/2014 4:13	0:00:01	0:00:01			4657	
23876		Haplotype Caller	676	ImportDirectory	12/16/2014 4:13	12/16/2014 4:13	0:00:02	0:00:02			0	
23876		Haplotype Caller	677	ImportDirectory	12/16/2014 4:13	12/16/2014 4:13	0:00:02	0:00:02			49	
23876		Haplotype Caller	678	ParseCSV-d	12/16/2014 4:13	12/16/2014 4:13	0:00:01	0:00:01			158	
23876		Haplotype Caller	679	CollectWFResults	12/16/2014 4:13	12/16/2014 4:26	0:13:02	0:00:17			18766501536	
23876		Haplotype Caller	697	FileListJoin	12/16/2014 4:26	12/16/2014 4:29	0:02:30	0:02:30			18766501536	
23876		Haplotype Caller	698	PickFiles	12/16/2014 4:29	12/16/2014 4:31	0:02:32	0:00:06			18728398144	

23876		Haplotype Caller	702	GATK-HaplotypeCaller	12/16/2014 4:31	12/16/2014 4:54	0:23:21	0:00:31			1649343	
23876		Haplotype Caller	712	ExportFiles	12/16/2014 4:54	12/16/2014 4:55	0:00:02	0:10:21	TOTAL	0:41:34	???	
23877		Haplotype Caller	684	PrepareLibrary	12/16/2014 4:16	12/16/2014 4:16	0:00:01	0:00:02			4657	
23877		Haplotype Caller	685	ImportDirectory	12/16/2014 4:16	12/16/2014 4:16	0:00:02	0:00:02			0	
23877		Haplotype Caller	686	ImportDirectory	12/16/2014 4:16	12/16/2014 4:16	0:00:01	0:00:01			49	
23877		Haplotype Caller	687	ParseCSV-d	12/16/2014 4:16	12/16/2014 4:16	0:00:02	0:00:02			158	
23877		Haplotype Caller	688	CollectWFResults	12/16/2014 4:16	12/16/2014 4:29	0:12:26	0:01:29			18766501536	
23877		Haplotype Caller	699	FileListJoin	12/16/2014 4:29	12/16/2014 4:30	0:01:49	0:01:49			18766501536	
23877		Haplotype Caller	700	PickFiles	12/16/2014 4:30	12/16/2014 4:32	0:01:45	0:00:04			18728398144	
23877		Haplotype Caller	704	GATK-HaplotypeCaller	12/16/2014 4:32	12/16/2014 4:40	0:08:00	0:05:53			817530	
23877		Haplotype Caller	706	ExportFiles	12/16/2014 4:40	12/16/2014 4:40	0:00:02	0:00:10	TOTAL	0:24:09	???	TOTAL Ha=
23980		GATK phase 3	720	PrepareLibrary	12/16/2014 5:12	12/16/2014 5:12	0:00:02	0:00:02			4657	
23980		GATK phase 3	721	PrepareLibrary	12/16/2014 5:12	12/16/2014 5:12	0:00:09	0:00:10			14247	
23980		GATK phase 3	722	PrepareLibrary	12/16/2014 5:12	12/16/2014 5:12	0:00:03	0:00:04			4982	
23980		GATK phase 3	723	ImportDirectory	12/16/2014 5:12	12/16/2014 5:12	0:00:08	0:00:08			185888349	I
23980		GATK phase 3	724	GATK-VariantRecalibrator	12/16/2014 5:12	12/16/2014 5:14	0:01:17	0:01:18			6300068	
23980		GATK phase 3	725	GATK-VariantRecalibrator	12/16/2014 5:14	12/16/2014 5:24	0:10:12	0:10:14			46653308	
23980		GATK phase 3	726	GATK-ApplyRecalibration	12/16/2014 5:24	12/16/2014 5:25	0:01:09	0:01:09			209954106	O
23980		GATK phase 3	727	GATK-ApplyRecalibration	12/16/2014 5:25	12/16/2014 5:26	0:01:03	0:01:03			212984157	
23980		GATK phase 3	728	ExportFiles	12/16/2014 5:26	12/16/2014 5:26	0:00:16	0:00:18	TOTAL	0:14:24		
23983		VCF Filters	732	PrepareLibrary	12/16/2014 5:26	12/16/2014 5:27	0:00:02	0:00:02			4657	VCF1
23983		VCF Filters	738	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:00:01			12069334	
23983		VCF Filters	743	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:09	0:00:01			212984157	
23983		VCF Filters	748	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:26	0:0:-7			46576188	I
23983		VCF Filters	756	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:00			37136921	O
23983		VCF Filters	762	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:03	0:00:00	TOTAL	0:02:17	???	
23984		VCF Filters	731	PrepareLibrary	12/16/2014 5:27	12/16/2014 5:27	0:00:01	0:0:-2			4657	VCF2
23984		VCF Filters	736	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:0:-1			12069334	
23984		VCF Filters	749	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:11	0:00:08			212984157	
23984		VCF Filters	754	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:28	0:01:14			46920150	I
23984		VCF Filters	759	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:00			37171640	O
23984		VCF Filters	766	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:04	0:00:05	TOTAL	0:02:24	???	
23985		VCF Filters	733	PrepareLibrary	12/16/2014 5:27	12/16/2014 5:27	0:00:01	0:0:-1			4657	VCF3
23985		VCF Filters	739	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:00:01			12069334	
23985		VCF Filters	745	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:10	0:00:02			212984157	
23985		VCF Filters	751	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:17	0:00:02			51741463	I
23985		VCF Filters	755	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:07			37711895	O
23985		VCF Filters	761	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:04	0:00:07	TOTAL	0:02:09	???	
23986		VCF Filters	734	PrepareLibrary	12/16/2014 5:27	12/16/2014 5:27	0:00:01	0:00:02			4657	VCF4
23986		VCF Filters	741	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:00:02			12069334	
23986		VCF Filters	744	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:09	0:00:00			212984157	
23986		VCF Filters	750	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:25	0:00:01			46760185	I
23986		VCF Filters	757	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:03			36702782	O
23986		VCF Filters	763	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:05	0:00:03	TOTAL	0:02:18	???	
23987		VCF Filters	735	PrepareLibrary	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:00:01			4657	VCF5

23987		VCF Filters	742	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:0:-1			12069334	
23987		VCF Filters	746	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:09	0:0:-1			212984157	
23987		VCF Filters	752	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:29	0:0:-1			50714303	I
23987		VCF Filters	760	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:21			38144466	O
23987		VCF Filters	765	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:05	0:00:01	TOTAL	0:02:23	???	
23988		VCF Filters	737	PrepareLibrary	12/16/2014 5:27	12/16/2014 5:27	0:00:01	0:0:-1			4657	VCF6
23988		VCF Filters	740	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:02	0:0:-1			12069334	
23988		VCF Filters	747	ImportFile	12/16/2014 5:27	12/16/2014 5:27	0:00:09	0:00:07			212984157	
23988		VCF Filters	753	GATK-SelectVariants	12/16/2014 5:27	12/16/2014 5:28	0:01:24	0:00:02			65603496	I
23988		VCF Filters	758	GATK-SelectVariants	12/16/2014 5:28	12/16/2014 5:29	0:00:35	0:00:04			38746310	O
23988		VCF Filters	764	ExportFiles	12/16/2014 5:29	12/16/2014 5:29	0:00:04	0:00:04	TOTAL	0:02:18	???	
24001		Coverage	768	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:01	0:00:04			4964	COVERAGE1
24001		Coverage	781	ImportDirectory	12/16/2014 5:29	12/16/2014 5:32	0:02:42	0:02:39			2766450004	I
24001		Coverage	783	Bedtools-Coverage	12/16/2014 5:32	12/16/2014 5:34	0:02:25	0:-2:-39			16125251	O
24001		Coverage	788	ExportFiles	12/16/2014 5:34	12/16/2014 5:34	0:00:02	0:00:20	TOTAL	0:05:11		
24002		Coverage	769	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:00	0:0:-1			4964	COVERAGE2
24002		Coverage	774	ImportDirectory	12/16/2014 5:29	12/16/2014 5:31	0:02:03	0:00:01			2824430083	I
24002		Coverage	776	Bedtools-Coverage	12/16/2014 5:31	12/16/2014 5:34	0:02:30	0:-2:-5			16126851	O
24002		Coverage	786	ExportFiles	12/16/2014 5:34	12/16/2014 5:34	0:00:03	0:00:12	TOTAL	0:04:38		
24003		Coverage	772	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:01	0:00:01			4964	COVERAGE3
24003		Coverage	784	ImportDirectory	12/16/2014 5:29	12/16/2014 5:32	0:03:11	0:03:12			3430581549	I
24003		Coverage	785	Bedtools-Coverage	12/16/2014 5:32	12/16/2014 5:35	0:02:57	0:01:22			16132338	O
24003		Coverage	791	ExportFiles	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:03	TOTAL	0:06:12		
24004		Coverage	770	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:01	0:00:02			4964	COVERAGE4
24004		Coverage	777	ImportDirectory	12/16/2014 5:29	12/16/2014 5:31	0:02:20	0:00:02			2787113930	I
24004		Coverage	779	Bedtools-Coverage	12/16/2014 5:31	12/16/2014 5:34	0:02:26	0:00:01			16126073	O
24004		Coverage	787	ExportFiles	12/16/2014 5:34	12/16/2014 5:34	0:00:03	0:00:18	TOTAL	0:04:51		
24005		Coverage	771	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:01	0:0:-1			4964	COVERAGE5
24005		Coverage	778	ImportDirectory	12/16/2014 5:29	12/16/2014 5:32	0:02:34	0:02:19			3290229376	I
24005		Coverage	782	Bedtools-Coverage	12/16/2014 5:32	12/16/2014 5:34	0:02:51	0:00:03			16123415	O
24005		Coverage	790	ExportFiles	12/16/2014 5:34	12/16/2014 5:35	0:00:02	0:00:43	TOTAL	0:05:29		
24006		Coverage	773	PrepareLibrary	12/16/2014 5:29	12/16/2014 5:29	0:00:01	0:00:00			4964	COVERAGE6
24006		Coverage	775	ImportDirectory	12/16/2014 5:29	12/16/2014 5:31	0:02:20	0:02:03			3629593202	I
24006		Coverage	780	Bedtools-Coverage	12/16/2014 5:31	12/16/2014 5:34	0:03:03	0:-2:-24			16090846	O
24006		Coverage	789	ExportFiles	12/16/2014 5:34	12/16/2014 5:34	0:00:02	0:00:03	TOTAL	0:05:26		
24019		Annotate Sample	796	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:01			56	ANNOTATE1
24019		Annotate Sample	801	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:04	0:00:03			37136921	I
24019		Annotate Sample	806	ANNOVAR	12/16/2014 5:35	12/16/2014 5:50	0:15:00	0:00:00			37136921	
24019		Annotate Sample	814	IGM-AnnoTool	12/16/2014 5:50	12/16/2014 5:51	0:01:03	0:00:19			67761043	
24019		Annotate Sample	820	ParseCSV	12/16/2014 5:51	12/16/2014 5:52	0:00:10	0:00:10			179873014	
24019		Annotate Sample	822	ColumnSelect	12/16/2014 5:52	12/16/2014 5:53	0:01:28	0:00:10			203018676	
24019		Annotate Sample	832	ExonicFilter	12/16/2014 5:53	12/16/2014 5:54	0:01:21	0:00:27			194904501	O
24019		Annotate Sample	838	SerializeCSV	12/16/2014 5:54	12/16/2014 5:55	0:00:15	0:00:15			31993390	
24019		Annotate Sample	840	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:03	0:00:03			29676207	
24019		Annotate Sample	842	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:04	0:00:14	TOTAL	0:19:30		

24020		Annotate Sample	795	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:0:-2			56	ANNOTATE2
24020		Annotate Sample	799	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:03	0:00:01			37171640	I
24020		Annotate Sample	805	ANNOVAR	12/16/2014 5:35	12/16/2014 5:51	0:15:49	0:0:-2			37171640	
24020		Annotate Sample	817	IGM-AnnoTool	12/16/2014 5:51	12/16/2014 5:52	0:01:04	0:00:03			67773669	
24020		Annotate Sample	827	ParseCSV	12/16/2014 5:52	12/16/2014 5:52	0:00:10	0:00:03			180391240	
24020		Annotate Sample	829	ColumnSelect	12/16/2014 5:52	12/16/2014 5:54	0:01:29	0:00:03			203553561	
24020		Annotate Sample	835	ExonicFilter	12/16/2014 5:54	12/16/2014 5:55	0:01:22	0:00:07			195438989	O
24020		Annotate Sample	849	SerializeCSV	12/16/2014 5:55	12/16/2014 5:56	0:00:15	0:00:08			31903616	
24020		Annotate Sample	851	ExportFiles	12/16/2014 5:56	12/16/2014 5:56	0:00:04	0:00:04			29598059	
24020		Annotate Sample	852	ExportFiles	12/16/2014 5:56	12/16/2014 5:56	0:00:04	0:00:05	TOTAL	0:20:26		
24021		Annotate Sample	797	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:00			56	ANNOTATE3
24021		Annotate Sample	803	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:03	0:0:-1			37711895	I
24021		Annotate Sample	809	ANNOVAR	12/16/2014 5:35	12/16/2014 5:51	0:15:19	0:00:02			37711895	
24021		Annotate Sample	816	IGM-AnnoTool	12/16/2014 5:51	12/16/2014 5:52	0:01:02	0:00:30			68946224	
24021		Annotate Sample	824	ParseCSV	12/16/2014 5:52	12/16/2014 5:52	0:00:10	0:00:11			183815048	
24021		Annotate Sample	825	ColumnSelect	12/16/2014 5:52	12/16/2014 5:54	0:01:38	0:00:00			207447678	
24021		Annotate Sample	834	ExonicFilter	12/16/2014 5:54	12/16/2014 5:55	0:01:24	0:00:23			199171146	O
24021		Annotate Sample	844	SerializeCSV	12/16/2014 5:55	12/16/2014 5:55	0:00:15	0:00:14			32799538	
24021		Annotate Sample	846	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:03	0:00:01			30423869	
24021		Annotate Sample	848	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:06	0:00:04	TOTAL	0:20:06		
24022		Annotate Sample	798	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:00:02			56	ANNOTATE4
24022		Annotate Sample	804	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:04	0:00:05			36702782	I
24022		Annotate Sample	808	ANNOVAR	12/16/2014 5:35	12/16/2014 5:50	0:14:58	0:00:01			36702782	
24022		Annotate Sample	813	IGM-AnnoTool	12/16/2014 5:50	12/16/2014 5:51	0:01:03	0:00:01			67069287	
24022		Annotate Sample	819	ParseCSV	12/16/2014 5:51	12/16/2014 5:52	0:00:11	0:00:01			178263299	
24022		Annotate Sample	821	ColumnSelect	12/16/2014 5:52	12/16/2014 5:53	0:01:27	0:00:00			201181335	
24022		Annotate Sample	831	ExonicFilter	12/16/2014 5:53	12/16/2014 5:54	0:01:21	0:00:01			193152468	O
24022		Annotate Sample	837	SerializeCSV	12/16/2014 5:54	12/16/2014 5:55	0:00:15	0:00:00			31987892	
24022		Annotate Sample	839	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:03	0:00:01			29666177	
24022		Annotate Sample	841	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:04	0:00:00	TOTAL	0:19:29		
24023		Annotate Sample	800	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:0:-3			56	ANNOTATE5
24023		Annotate Sample	807	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:04	0:00:01			38144466	I
24023		Annotate Sample	810	ANNOVAR	12/16/2014 5:35	12/16/2014 5:51	0:15:50	0:0:-4			38144466	
24023		Annotate Sample	818	IGM-AnnoTool	12/16/2014 5:51	12/16/2014 5:52	0:01:04	0:00:08			69423203	
24023		Annotate Sample	828	ParseCSV	12/16/2014 5:52	12/16/2014 5:53	0:00:11	0:00:08			184954222	
24023		Annotate Sample	830	ColumnSelect	12/16/2014 5:53	12/16/2014 5:54	0:01:34	0:00:31			208765193	
24023		Annotate Sample	836	ExonicFilter	12/16/2014 5:54	12/16/2014 5:56	0:01:24	0:00:19			200422039	O
24023		Annotate Sample	850	SerializeCSV	12/16/2014 5:56	12/16/2014 5:56	0:00:15	0:00:07			32138763	
24023		Annotate Sample	853	ExportFiles	12/16/2014 5:56	12/16/2014 5:56	0:00:04	0:00:04			29813642	
24023		Annotate Sample	854	ExportFiles	12/16/2014 5:56	12/16/2014 5:56	0:00:05		TOTAL	0:20:34		
24024		Annotate Sample	802	StringList	12/16/2014 5:35	12/16/2014 5:35	0:00:01	0:0:-1			58	ANNOTATE6
24024		Annotate Sample	811	ImportDirectory	12/16/2014 5:35	12/16/2014 5:35	0:00:04	0:00:05			38746310	I
24024		Annotate Sample	812	ANNOVAR	12/16/2014 5:35	12/16/2014 5:51	0:15:14	0:14:54			38746310	
24024		Annotate Sample	815	IGM-AnnoTool	12/16/2014 5:51	12/16/2014 5:52	0:01:04	0:00:02			70539351	
24024		Annotate Sample	823	ParseCSV	12/16/2014 5:52	12/16/2014 5:52	0:00:12	0:00:01			189485156	

24024		Annotate Sample	826	ColumnSelect	12/16/2014 5:52	12/16/2014 5:54	0:01:33	0:00:20			213871485	
24024		Annotate Sample	833	ExonicFilter	12/16/2014 5:54	12/16/2014 5:55	0:01:27	0:00:05			205326140	O
24024		Annotate Sample	843	SerializeCSV	12/16/2014 5:55	12/16/2014 5:55	0:00:15	0:00:02			31951423	
24024		Annotate Sample	845	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:03	0:00:02			29651358	
24024		Annotate Sample	847	ExportFiles	12/16/2014 5:55	12/16/2014 5:55	0:00:06	0:00:02	TOTAL	0:20:01		

Bibliography

[1] About - digital institute newcastle university
. http://www.ncl.ac.uk/digitalinstitute/about/.

[2] About - the framework simulation. https://github.com/Farisllwaah/NGS-Framework.

[3] Apache hadoop. http://hadoop.apache.org.

[4] Application showcase pegasus WMS. https://pegasus.isi.edu/application-showcase/.

[5] Microsoft Azure Pricing. http://azure.microsoft.com/en-gb/pricing/calculator/?scenario=virtual-machines.

[6] The Laser Interferometer Gravitational-Wave Observatory (LIGO). https://www.ligo.caltech.edu/.

[7] USC molecular genomics core. http://epigenome.usc.edu/.

[8] What is bandwidth? webopedia definition
. https://www.webopedia.com/TERM/B/bandwidth.html.

[9] What is MIPS? Webopedia Definition
. https://www.webopedia.com/TERM/M/MIPS.html.

[10] Workflow Gallery Pegasus WMS
. https://pegasus.isi.edu/documentation/workflow_gallery.php.

[11] WorkflowGenerator−Pegasus - Pegasus Workflow Management System. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

[12] S Achour, M Ammar, B Khmili, and W Nasri. MPI-PERF-SIM: Towards an automatic performance prediction tool of MPI programs on hierarchical clusters. *Proc. - 19th Int. Euromicro Conf. Parallel, Distrib. Network-Based Process. PDP 2011*, pages 207–211, 2011.

[13] V. S Adve, R Bagrodia, J. C Browne, E Deelman, A Dube, E. N Houstis, J. R Rice, R Sakellariou, D. J Sundaram-Stukel, P. J Teller, *et al.* Poems: End-to-end performance design of large parallel adaptive computational systems. *IEEE transactions on Software Engineering*, 26(11):1027–1048, 2000.

[14] M Amiri and L Mohammad-Khanli. Survey on prediction models of applications for resources provisioning in cloud. *Journal of Network and Computer Applications*, 2017.

[15] M Armbrust, A Fox, R Griffith, A. D Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, I Stoica, *et al.* A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[16] A Barker and J Van Hemert. Scientific workflow: a survey and research directions. In *International Conference on Parallel Processing and Applied Mathematics*, pages 746–753. Springer, 2007.

[17] M. N Bennani and D. A Menasce. Resource allocation for autonomic data centers using analytic performance models. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pages 229–240. IEEE, 2005.

[18] G. B Berriman, E Deelman, J. C Good, J. C Jacob, D. S Katz, C Kesselman, A. C Laity, T. A Prince, G Singh, and M.-H Su. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *Optimizing Scientific Return for Astronomy through Information Technologies*, volume 5493, pages 221–233. International Society for Optics and Photonics, 2004.

[19] S Bharathi, A Chervenak, E Deelman, G Mehta, M.-H Su, and K Vahi. Characterization of scientific workflows. In *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, pages 1–10. IEEE, 2008.

[20] J Bhimani, N Mi, M Leeser, and Z Yang. Fim: performance prediction for parallel computation in iterative data processing applications. In *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*, pages 359–366. IEEE, 2017.

[21] M Borkowski, S Schulte, and C Hochreiner. Predicting cloud resource utilization. In *Utility and Cloud Computing (UCC), 2016 IEEE/ACM 9th International Conference on*, pages 37–42. IEEE, 2016.

[22] D. A Brown, P. R Brady, A Dietz, J Cao, B Johnson, and J McNabb. A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. In *Workflows for e-Science*, pages 39–59. Springer, 2007.

[23] E Brynjolfsson and A Mcafee. The business of artificial intelligence. *Harvard Business Review*, 2017.

[24] M Bux and U Leser. Parallelization in scientific workflow management systems. *arXiv preprint arXiv:1303.7195*, 2013.

[25] R Buyya, R Ranjan, and R. N Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pages 1–11. IEEE, 2009.

[26] Z Cai, Q Li, and X Li. Elasticsim: A toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times. *Journal of Grid Computing*, 15(2):257–272, 2017.

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

[28] J Cała, H Hiden, S Woodman, and P Watson. Cloud computing for fast prediction of chemical activity. *Futur. Gener. Comput. Syst.*, 29(7):1860–1869, September 2013.

[29] J Cała, Y. X Xu, E. A Wijaya, and P Missier. From scripted HPC-based NGS pipelines to workflows on the cloud. In *Procs. C4Bio workshop, co-located with the 2014 CCGrid conference*, Chicago, IL, 2014. IEEE.

[30] B Calder, J Wang, A Ogus, and et al. 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*, pages 143–157, New York, New York, USA, 2011. ACM Press.

[31] B Calder, J Wang, A Ogus, N Nilakantan, A Skjolsvold, S McKelvie, Y Xu, S Srivastav, J Wu, H Simitci, *et al.* Windows azure storage: a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 143–157. ACM, 2011.

[32] R. N Calheiros, R Buyya, and C. A De Rose. Building an automated and self-configurable emulation testbed for grid applications. *Software: Practice and Experience*, 40(5):405–429, 2010.

[33] R. N Calheiros, M. A Netto, C. A De Rose, and R Buyya. Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications. *Software: Practice and Experience*, 43(5):595–612, 2013.

[34] R. N Calheiros, R Ranjan, A Beloglazov, C. A. F De Rose, and R Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, Jan 2011.

[35] R. N Calheiros, R Ranjan, C. a. F De Rose, and R Buyya. CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. *arXiv preprint arXiv:0903.2525*, page 9, 2009.

[36] J Cao, S. A Jarvis, S Saini, and G. R Nudd. Gridflow: Workflow management for grid computing. In *null*, page 198. IEEE, 2003.

[37] C. P Chen and C.-Y Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. In *Information Sciences*, volume 275, pages 314 – 347, 2014.

[38] L Chen, V Tan, F Xu, A Biller, P Groth, S Miles, J Ibbotson, M Luck, and L Moreau. A proof of concept: Provenance in a service oriented architecture. In *In Proceedings of the UK OST e-Science Second All Hands Meeting 2005 (AHM05)*, 2005.

[39] W Chen and E Deelman. Partitioning and scheduling workflows across multiple sites with storage constraints. In *International Conference on Parallel Processing and Applied Mathematics*, pages 11–20. Springer, 2011.

[40] W Chen and E Deelman. *Partitioning and Scheduling Workflows across Multiple Sites with Storage Constraints*, pages 11–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[41] W Chen and E Deelman. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th International Conference on E-Science*, pages 1–8. IEEE, Oct 2012.

[42] P Coetzee and S Jarvis. Goal-based composition of scalable hybrid analytics for heterogeneous architectures. *Journal of Parallel and Distributed Computing*, 108:59–73, 2017.

[43] L. B Costa, S Al-Kiswany, H Yang, and M Ripeanu. Supporting storage configuration for I/O intensive workflows. *ICS '14 Proceedings of the 28th ACM International Conference on Supercomputing*, pages 191–200, 2014.

[44] M Cunha, N Mendonca, and A Sampaio. Investigating the impact of deployment configuration and user demand on a social network application in the amazon ec2 cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 746–751. IEEE, 2011.

[45] R. F Da Silva, W Chen, G Juve, K Vahi, and E Deelman. Community resources for enabling research in distributed scientific workflows. In *e-Science (e-Science), 2014 IEEE 10th International Conference on*, volume 1, pages 177–184. IEEE, 2014.

[46] 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 *WORKS@ SC*, pages 58–67, 2013.

[47] E Deelman, J Blythe, Y Gil, C Kesselman, G Mehta, S Patil, M.-H Su, K Vahi, and M Livny. *Pegasus: Mapping Scientific Workflows onto the Grid*, pages 11–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[48] E Deelman, D Gannon, M Shields, and I Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[49] E Deelman and Y Gil. Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges. In *2006 Second IEEE Int. Conf. e-Science Grid Comput.*, pages 144–144. IEEE, December 2006.

[50] E Deelman, G Singh, M Livny, B Berriman, and J Good. 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, 2008.

[51] E Deelman, G Singh, M.-H Su, J Blythe, Y Gil, C Kesselman, G Mehta, K Vahi, G. B Berriman, J Good, *et al.* Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[52] E Deelman, G Singh, M.-H Su, J Blythe, Y Gil, C Kesselman, G Mehta, K Vahi, G. B Berriman, J Good, A Laity, J. C Jacob, and D. S Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific programming.*, 13(3):219–237, 2005.

[53] E Deelman, K Vahi, G Juve, M Rynge, S Callaghan, P. J Maechling, R Mayani, W Chen, R Ferreira da Silva, M Livny, and K Wenger. Pegasus, a workflow management system for science automation. *Futur. Gener. Comput. Syst.*, 46:17–35, May 2015.

[54] I. D Dinov, F Torri, F Macciardi, P Petrosyan, Z Liu, A Zamanyan, P Eggert, J Pierce, A Genco, J. A Knowles, *et al.* Applications of the pipeline environment for visual informatics and genomics computations. *BMC bioinformatics*, 12(1):304, 2011.

[55] J Ekanayake, S Pallickara, and G Fox. Mapreduce for data intensive scientific analyses. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 277–284. IEEE, 2008.

[56] T Fahringer, R Prodan, R Duan, J Hofer, F Nadeem, F Nerieri, S Podlipnig, J Qin, M Siddiqui, H.-L Truong, *et al.* Askalon: A development and grid computing environment for scientific workflows. In *Workflows for e-Science*, pages 450–471. Springer, 2007.

[57] F Fakhfakh, H. H Kacem, and A. H Kacem. Simulation tools for cloud computing: A survey and comparative study. In *Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on*, pages 221–226. IEEE, 2017.

[58] C.-T Fan, Y.-S Chang, W.-J Wang, and S.-M Yuan. Execution Time Prediction Using Rough Set Theory in Hybrid Cloud. *Ubiquitous Intell. Comput. 9th Int. Conf. Auton. Trust. Comput. (UIC/ATC), 2012 9th Int. Conf.*, pages 729–734, 2012.

[59] F Fittkau, S Frey, and W Hasselbring. Cdosim: Simulating cloud deployment options for software migration support. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on the*, pages 37–46. IEEE, 2012.

[60] Y Gil, J Kim, V Ratnakar, and E Deelman. Wings for pegasus: A semantic approach to creating very large scientific workflows. In *OWLED*06 Workshop on OWL:Experiences and Directions*, 2006.

[61] Y Gil, V Ratnakar, E Deelman, G Mehta, and J Kim. Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1767. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[62] T Glatard, J Montagnat, and X Pennec. A probabilistic model to analyse workflow performance on production grids. In *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pages 510–517. IEEE, 2008.

[63] B Glavic. Big data provenance: Challenges and implications for benchmarking. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, In Specifying Big Data Benchmarks:72–80, 2014.

[64] K Görlach, M Sonntag, D Karastoyanova, F Leymann, and M Reiter. Conventional workflow technology for scientific simulation. In *Guide to e-Science*, pages 323–352. Springer, 2011.

[65] R Graves, T. H Jordan, S Callaghan, E Deelman, E Field, G Juve, C Kesselman, P Maechling, G Mehta, K Milner, *et al.* CyberShake: A physics-based seismic hazard model for Southern California. *Pure and Applied Geophysics*, 168(3-4):367–381, 2011.

[66] N Grozev and R Buyya. Performance modelling and simulation of three-tier applications in Cloud and Multi-Cloud environments. *Computer Journal*, 58(1):1–22, 2013.

[67] S. D Hammond, G. R Mudalige, J Smith, S. A Jarvis, J Herdman, and A Vadgama. Warpp: a toolkit for simulating high-performance parallel scientific codes. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 19. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[68] I. A. T Hashem, I Yaqoob, N. B Anuar, S Mokhtar, A Gani, and S Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

[69] L He, M Calleja, M Hayes, and S. A Jarvis. Performance prediction for running workflows under role-based authorization mechanisms. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[70] L He, N Chaudhary, and S. A Jarvis. Developing security-aware resource management strategies for workflows. *Future Generation Computer Systems*, 38:61–68, 2014.

[71] H Herodotou and S Babu. A what-if engine for cost-based mapreduce optimization. *IEEE Data Eng. Bull.*, 36(1):5–14, 2013.

[72] H Hiden, S Woodman, P Watson, and J Cała. 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.

[73] H Hiden, S Woodman, P Watson, and J Cała. 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), 2013.

[74] D Hollingsworth and U. K Hampshire. Workflow management coalition: The workflow reference model. *Document Number TC00-1003*, 19, 1995.

[75] D. H Hu, Y Wang, and C.-L Wang. Betterlife 2.0: Large-scale social intelligence reasoning on cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 529–536. IEEE, 2010.

[76] M Imran and H Hlavacs. Applications of provenance data for cloud infrastructure. In *Semantics, Knowledge and Grids (SKG), 2012 Eighth International Conference on*, pages 16–23. IEEE, 2012.

[77] A Inc. *Amazon Elastic Compute Cloud (Amazon EC2).* Amazon Inc., http://aws.amazon.com/ec2.

[78] G Inc. *Google App Engine.* Google Inc., code.google.com/appengine.

[79] S Inc. *Salesforce.com.* Salesforce Inc., salesforce.com.

[80] E. P International, P Drennan, and D. a Keeffe. QUT Digital Repository : This is the author-version of the work . Conference proceedings published , by http://www.springer.de/comp/lncs/ Lecture Notes in Computer Science Virtual Consumption : Using Player Types to Explore Virtual Consumer Behavior. pages 466–469, 2007.

[81] S Islam, J Keung, K Lee, and A Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, 2012.

[82] M. a Iverson, F Özgüner, and L Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. *IEEE Trans. Comput.*, 48(12):1374–1379, 1999.

[83] S. A Jarvis, B Foley, P Isitt, D. P Spooner, D Rueckert, and G. R Nudd. Performance prediction for a code with data-dependent runtimes. *Concurrency and Computation: Practice and Experience*, 20(3):195–206, 2008.

[84] D Jayasinghe, S Malkowski, J Li, Q Wang, Z Wang, and C Pu. Variations in performance and scalability: An experimental study in iaas clouds using multi-tier workloads. *IEEE Transactions on Services Computing*, 7(2):293–306, 2014.

[85] G Jung, T Mukherjee, S Kunde, H Kim, N Sharma, and F Goetz. Cloudadvisor: A recommendation-as-a-service platform for cloud configuration and pricing. In *Services (SERVICES), 203 IEEE Ninth World Congress on*, pages 456–463. IEEE, 2013.

[86] 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 *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, pages 1–9. IEEE, 2010.

[87] A Kashlev and S Lu. A system architecture for running big data workflows in the cloud. In *Services Computing (SCC), 2014 IEEE International Conference on*, pages 51–58. IEEE, 2014.

[88] A Khajeh-Hosseini, D Greenwood, J. W Smith, and I Sommerville. The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience*, 42(4):447–465, 2012.

[89] S Khan, K. A Shakil, and M Alam. Workflow-based big data analytics in the cloud environment present research status and future prospects. *arXiv preprint arXiv:1711.02087*, 2017.

[90] S Khan, K. A Shakil, and M Alam. *Cloud-Based Big Data Analytics—A Survey of Current Research and Future Directions*, pages 595–604. Springer Singapore, 2018.

[91] Y Koh, R Knauerhase, P Brett, M Bowman, Z Wen, and C Pu. An analysis of performance interference effects in virtual environments. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 200–209. IEEE, 2007.

[92] S. N Laboratories, U. S. D of Energy. Office of Science, U. S. D of Energy. Office of Scientific, and T Information. *The Future of Scientific Workflows. Report of the DOE NGNS/CS Scientific Workflows Workshop (Sandia Contributions).* United States. Department of Energy. Office of Science, 2015.

[93] S Lee, J. S Meredith, and J. S Vetter. Compass: A framework for automated performance modeling and prediction. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pages 405–414. ACM, 2015.

[94] Y. C Lee, H Han, A. Y Zomaya, and M Yousif. Resource-efficient workflow scheduling in clouds. *Knowledge-Based Systems*, 80:153–162, 2015.

[95] A Li, X Yang, S Kandula, and M Zhang. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2010.

[96] A Li, X Zong, S Kandula, X Yang, and M Zhang. CloudProphet: towards application performance prediction in cloud. *ACM SIGCOMM Comput. Commun. Rev.*, 41:426–427, 2011.

[97] H Li and R Durbin. Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[98] X Li and J Qiu. *Cloud Computing for Data-Intensive Applications*, volume 1. Springer, 2014.

[99] C. S Liew, M. P Atkinson, M Galea, T. F Ang, P Martin, and J. I. V Hemert. Scientific workflows: moving across paradigms. *ACM Computing Surveys (CSUR)*, 49(4):66, 2017.

[100] C Lin and S Lu. Scheduling scientific workflows elastically for cloud computing. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 746–747. IEEE, 2011.

[101] J Liu, E Pacitti, P Valduriez, and M Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, 13(4):457–493, Dec 2015.

[102] 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*, 84(3):354–376, 2011.

[103] X Liu, X Zhu, S Singhal, and M Arlitt. Adaptive entitlement control of resource containers on shared servers. In *Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on*, pages 163–176. IEEE, 2005.

[104] F Llwaah, J Cała, and N Thomas. Simulation of runtime performance of big data workflows on the cloud. In *European Workshop on Performance Engineering*, LNCS, volume 9951, pages 141–155. Springer, 2016.

[105] F Llwaah, J Cała, and N Thomas. Runtime Performance Prediction of Big Data Workflows with I/O-aware Simulation. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS'17, 2017.

[106] F Llwaah, N Thomas, and J Cała. Improving mct scheduling algorithm to reduce the makespan and cost of workflow execution in the cloud. In *Proceedings of the 31st UK Performance Engineering Workshop*. School of Computing, University of Leeds, 2015.

[107] S Long and Y Zhao. A toolkit for modeling and simulating cloud data storage: An extension to cloudsim. In *2012 International Conference on Control Engineering and Communication Technology, IEEE*, pages 597–600, Dec 2012.

[108] W Long, L Yuqing, and X Qingxin. Using cloudsim to model and simulate cloud computing environment. In *Computational Intelligence and Security (CIS), 9th International Conference*, pages 323–328. IEEE, 2013.

[109] B Louis, K Mitra, S Saguna, and C Ahlund. Cloudsimdisk: Energy-aware storage simulation in cloudsim. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 11–15, Dec 2015.

[110] R Lovas, G Dózsa, P Kacsuk, N Podhorszki, and D Drótos. Workflow support for complex grid applications: Integrated and portal solutions. In *Grid Computing*, pages 129–138. Springer, 2004.

[111] B Ludäscher, M Weske, T McPhillips, and S Bowers. Scientific workflows: Business as usual? *Business Process Management*, pages 31–47, 2009.

[112] A Matsunaga, M Tsugawa, and J Fortes. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 222–229. IEEE, 2008.

[113] A. S McGough, J Cohen, J Darlington, E Katsiri, W Lee, S Panagiotidi, and Y Patel. An end-to-end workflow pipeline for large-scale grid computing. *Journal of Grid Computing*, 3(3):259–281, Sep 2005.

[114] P Mell, K Scarfone, and S Romanosky. Common vulnerability scoring system. *Security Privacy, IEEE*, 4(6):85–89, Nov 2006.

[115] P. M Mell and T Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.

[116] S Nepal and S Pandey. Guest editorial: Cloud computing and scientific applications (ccsa)–big data analysis in the cloud. *The Computer Journal*, 59(3):285–286, 2016.

[117] M. A Netto and R Buyya. Offer-based scheduling of deadline-constrained bag-of-tasks applications for utility computing systems. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–11. IEEE, 2009.

[118] G. R Nudd and S. A Jarvis. Performance-based middleware for grid computing. *Concurrency and computation: practice and experience*, 17(2-4):215–234, 2005.

[119] A Núñez, J. L Vázquez-Poletti, A. C Caminero, G. G Castañé, J Carretero, and I. M Llorente. icancloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

[120] T Oinn, M Addis, J Ferris, D Marvin, M Senger, M Greenwood, T Carver, K Glover, M. R Pocock, A Wipat, *et al.* Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[121] S Ostermann, K Plankensteiner, and R Prodan. Using a new event-based simulation framework for investigating resource provisioning in clouds. *Scientific Programming*, 19(2-3):161–178, 2011.

[122] S Ostermann, K Plankensteiner, R Prodan, and T Fahringer. Groudsim: an event-based simulation framework for computational grids and clouds. In *European Conference on Parallel Processing*, pages 305–313. Springer, 2010.

[123] S Pabinger, A Dander, M Fischer, R Snajder, M Sperk, M Efremova, B Krabichler, M. R Speicher, J Zschocke, and Z Trajanoski. A survey of tools for variant analysis of next-generation genome sequencing data. *Brief. Bioinform.*, 15(2):256–278, 2014.

[124] S Pabinger, A Dander, M Fischer, R Snajder, M Sperk, M Efremova, B Krabichler, M. R Speicher, J Zschocke, and Z Trajanoski. A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in bioinformatics*, 15(2):256–278, 2014.

[125] T. P Pham, J. J Durillo, and T Fahringer. Predicting workflow task execution time in the cloud using a two-stage machine learning approach. *IEEE Transactions on Cloud Computing*, 2017.

[126] M Rak, A Cuomo, and U Villano. Cost/Performance Evaluation for Cloud Applications Using Simulation. *2013 Work. Enabling Technol. Infrastruct. Collab. Enterp.*, pages 152–157, 2013.

[127] M Rak, M Turtur, and U Villano. Early prediction of the cost of HPC application execution in the cloud. *Proc. - 16th Int. Symp. Symb. Numer. Algorithms Sci. Comput. SYNASC 2014*, pages 409–416, 2015.

[128] R Ranjan, S Garg, A. R Khoskbar, E Solaiman, P James, and D Georgakopoulos. Orchestrating bigdata analysis workflows. *IEEE Cloud Computing*, 4(3):20–28, 2017.

[129] C Shen, W Tong, J.-N Hwang, and Q Gao. Performance modeling of big data applications in the cloud centers. *The Journal of Supercomputing*, 73(5):2258–2283, 2017.

[130] Y Simmhan, B Plale, D Gannon, and S Marru. Performance evaluation of the karma provenance framework for scientific workflows. In L Moreau and I Foster, editors, *Provenance and Annotation of Data*, volume 4145 of *Lecture Notes in Computer Science*, pages 222–236. Springer Berlin Heidelberg, 2006.

[131] D. P Spooner, J Cao, S. A Jarvis, L He, and G. R Nudd. Performance-aware workflow management for grid computing. *The Computer Journal*, 48(3):347–357, 2005.

[132] T Sturm, F Jrad, and A Streit. Storage CloudSim a simulation environment for cloud object storage infrastructures. *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science*, pages 186–192, 2014.

[133] D Talia. Workflow systems for science: concepts and tools. Hindawi Publishing Corporation, 2013.

[134] I. J Taylor, E Deelman, D. B Gannon, and M Shields. *Workflows for e-Science: scientific workflows for grids*. Springer Publishing Company, Incorporated, 2014.

[135] J. K Teer and J. C Mullikin. Exome sequencing: the sweet spot before whole genomes. *Human molecular genetics*, 19(R2):R145–R151, 2010.

[136] F Tian and K Chen. Towards optimal resource provisioning for running mapreduce programs in public clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 155–162. IEEE, 2011.

[137] A Tsalgatidou, G Athanasopoulos, M Pantazoglou, C Pautasso, T Heinis, R Grønmo, H Hoff, A.-J Berre, M Glittum, and S Topouzidou. Developing scientific workflows from heterogeneous services. *ACM Sigmod Record*, 35(2):22–28, 2006.

[138] J. D Ullman. Np-complete scheduling problems. *Journal of Computer and System sciences*, 10(3):384–393, 1975.

[139] B Urgaonkar, P Shenoy, A Chandra, P Goyal, and T Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1, 2008.

[140] V Viana, D De Oliveira, and M Mattoso. Towards a cost model for scheduling scientific workflows activities in cloud environments. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 216–219. IEEE, 2011.

[141] S Weisberg. *Non-linear regression*, volume 528. A JOHN WILEY & SONS, INC., PUBLICATION, 2005.

[142] B Wickremasinghe, R. N Calheiros, and R Buyya. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 446–452. IEEE, 2010.

[143] M. D Wilkinson, B Vandervalk, and L McCarthy. The Semantic Automated Discovery and Integration (SADI) web service design-pattern, API and reference implementation. *Journal of Biomedical Semantics*, 2(1):8, 2011.

[144] K Wolstencroft, R Haines, D Fellows, A Williams, D Withers, S Owen, S Soiland-Reyes, I Dunlop, A Nenadic, P Fisher, *et al.* The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, 41(W1):W557–W561, 2013.

[145] S Woodman, H Hiden, and P Watson. Workflow provenance: an analysis of long term storage costs. In *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science*, page 9. ACM, 2015.

[146] S Woodman, H Hiden, and P Watson. Applications of provenance in performance prediction and data storage optimisation. volume 75, pages 299–309. Elsevier, 2017.

[147] S Woodman, H Hiden, P Watson, and P Missier. Achieving reproducibility by combining provenance with service and workflow versioning. In *Proceedings of the 6th workshop on Workflows in support of large-scale science*, pages 127–136. ACM, 2011.

[148] H Wu, W Zhang, J Zhang, J Wei, and T Huang. A benefit-aware on-demand provisioning approach for multi-tier applications in cloud computing. *Frontiers of Computer Science*, 7(4):459–474, 2013.

[149] Q Wu and Y Gu. Optimizing end-to-end performance of data-intensive computing pipelines in heterogeneous network environments. *Journal of Parallel and Distributed Computing*, 71(2):254–265, 2011.

[150] Q Zhang, L Cheng, and R Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, may 2010.

[151] Y Zhang, Z Zheng, and M. R Lyu. Real-time performance prediction for cloud components. In *Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2012 15th IEEE International Symposium on*, pages 106–111. IEEE, 2012.

[152] L Zhao and S. A Jarvis. Predictive performance modelling of parallel component compositions. *Cluster Computing*, 10(2):155–166, 2007.

[153] Y Zhao, X Fei, I Raicu, and S Lu. Opportunities and challenges in running scientific workflows on the cloud. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, pages 455–462. IEEE, 2011.

[154] Y Zhao, Y Li, I Raicu, S Lu, W Tian, and H Liu. Enabling scalable scientific workflow management in the cloud. *Future Generation Computer Systems*, 46:3–16, 2015.

[155] A. C Zhou, B He, and S Ibrahim. eScience and Big Data Workflows in Clouds: A Taxonomy and Survey. *Big Data: Principles and Paradigms*, pages 431–455, 2016.