Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 16(1) March 2015

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 16, Number 1, March 2015

TABLE OF CONTENTS

Special issue on Cloud for Health:

DOI 10.12694/scpe.v16i1.1056
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON CLOUD FOR HEALTH

Dear SCPE readers,

It is a pleasure to present this special issue covering subjects related to applying cloud computing to bioinformatics, biomedicine, and health, including solutions to area problems and architectural adaptation of cloud systems to fit those problems. The special issue includes papers selected from the workshops C4Bio 2014 and CCGrid-Health 2014 that, after further extension and additional review, were selected for publication. C4Bio 2014 and CCGrid-Health 2014 were held in Chicago, during May 26-29th 2014 within the framework of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014).

The special issue begins with the review paper "Cloud Computing in Healthcare and Biomedicine". Calabrese and Cannataro explore the current academic and commercial cloud-based solutions developed in the healthcare and biomedicine domains, designed in order to satisfy the specific needs (e.g. massive scalable computing and storage, data sharing, on-demand access anytime and anywhere) in these fields. Subsequently, the paper summarizes main issues regarding the use of Cloud Computing in healthcare and biomedicine.

The article "Selected Approaches and Frameworks to Carry out Genomic Data Analysis on the Cloud", by Church and Goscinski, examines commonly used cloud-based genomic analysis services, introduces the approach of exposing data as services and proposes two new solutions, namely HPCaaS and Uncinus, which aim to automate service development, deployment process and data provision. By comparing and contrasting these solutions, the authors identify key mechanisms of service creation, execution and data access required to support non-computing specialists employing clouds.

Ahmad, Hasan, and Siddique propose to formally reason about molecular pathways of Zsyntax based deduction using the HOL4 theorem prover and developing a biologist friendly graphical user interface. "On the Formalization of Zsyntax with Applications in Molecular Biology" shows additionally a use case with the formal analysis of cancer-related molecular pathway, i.e., TP53 degradation and metabolic pathway, known as Glycolysis.

The article "Accelerating Comparative Genomics Workflows in a Distributed Environment with Optimized Data Partiotioning and Workflow Fusion", by Choudhury et al., discusses different strategies for alleviating the computational constraints experienced by several data-intensive bioinformatics applications, presenting a workflow-based framework to parallelize the compute-intensive tasks of genome alignment and variant calling, the two most important stages in most comparative genomics applications. This framework provides workflow fusion to merge multiple sequential workflows into a single optimized workflow that benefits from caching, eliminating choke points, and stacked partitioning.

In "Performance Comparison and Tuning of Virtual Machines For Sequence Alignment Software", Estrada et al. explore the performance cost of virtualization for the fast growing application domain of genomics to inform the feasibility of running an NGS pipeline in a cloud, and in doing so consider two prevalent short-read sequence alignment programs, BWA and Novoalign. They execute these applications in three separate open-source system virtualization solutions (KVM hypervisor, Xen paravirtualized hypervisor, and Linux Containers) and compare the runtime in each environment against the runtime of the same system without virtualization and measure the relative performance of each hypervisor, presenting tuning suggestions for cloud implementors and users.

The article "Extending XNAT towards a Cloud-based Quality Assessment Platform for Retinal Optical Coherence Tomographies", by Jansen et al., presents a platform designed for automatic quality assessment of retinal OCTs and provided as a cloud-based service employing OpenStack. It extends the image management platform XNAT by services to calculate and store quality measures. It is also extensible regarding new quality measure algorithms, allowing the developer to upload, compile and test code for the system's architecture.

Finally, the paper "A Tool for Managing the X1.V1 Platform on the Cloud", by Marzini et al., proposes a framework for managing the execution of the X1.V1 platform on the Cloud. This framework enables an easy, quick, and secure management of the Cloud resources allocation and reallocation to X1.V1 Virtual Machines, in order to enhance the platform performance, optimize resource utilization and, consequently, reduce the whole services cost.

We would like to thank all the reviewers for their effort, cooperation, and valuable feedback and all the authors who submitted papers to C4Bio 2014 and CCGrid-Health 2014 and to this Special Issue.

Jesus Carretero and Javier García-Blas, Universidad Carlos III de Madrid, Spain
Sandra Gesing, University of Notre Dame, USA

DOI 10.12694/scpe.v16i1.1057
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 1–18. http://www.scpe.org

CLOUD COMPUTING IN HEALTHCARE AND BIOMEDICINE

BARBARA CALABRESE*AND MARIO CANNATARO†

Abstract. High throughput platforms available in clinical settings or in research laboratories, such as magnetic resonance imaging, microarray, mass spectrometry and next-generation sequencing, are producing an increasing volume of clinical and omics data that poses new issues in terms of secure data storage, models for data integration and analysis, and high performance computing. Cloud Computing offers large scalable computing and storage, data sharing, on-demand anytime and anywhere access to resources and applications, and it supports easy but powerful distributed computing models, for facing those issues. In fact, in the recent years it has been adopted for the deployment of several applications in healthcare and bioinformatics both in academia and in the industry. However, cloud computing presents several issues regarding the security and privacy of data, that are particularly important when analyzing patients data, such as in personalized medicine. This paper reviews main cloud-based healthcare and biomedicine applications; with a special focus on healthcare, biomedicine and bioinformatics solutions and underlines main issues and problems related to the use of such platforms for the storage and analysis of patients data.

Key words: cloud computing, healthcare, biomedicine, bioinformatics, virtualization

AMS subject classifications. 68M14, 68U01

1. Introduction. Research in life sciences, as well as, clinical practice in medicine, are more and more based on the large datasets produced by high throughput platforms used for the investigation of the human body (e.g. magnetic resonance imaging), as well as the cell machinery (e.g mass spectrometry, microarray, and next generation sequencing). For instance, medical imaging is producing large datasets of biomedical images, while omics sciences, such as genomics, proteomics and interactomics, are producing large datasets of experimental data. Some recent advances in biomedicine and healthcare are based on the integrated analysis of clinical and omics data, as in pharmacogenomics. Pharmacogenomics is an important branch of genomics that studies the impact of genetic variation (e.g. Single Nucleotide Polimorphisms - SNPs) on drug response in patients and is at the basis of the so-called "personalized medicine", where drugs are chosen or optimized to meet the genetic profile of each patient. The application of omics studies on large populations is producing an increasing amount of experimental and clinical data, as well as specialized databases spread over the Internet. However, the storage, preprocessing and analysis of experimental data are becoming the main bottleneck of the analysis pipeline.

Cloud computing is a computing model that has spread very rapidly in recent years for the supply of IT resources (hardware and software) of different nature, through services accessible via the network. Cloud can provide Virtual Machines (VM) in the sense of hardware resources, platforms to deploy applications or ready-to-use services. The cloud resources are dynamically scalable, virtualized and accessible on the Internet [1]. This model provides new advantages related to massive and scalable computing resources available on demand, virtualization technology and payment for use as needed [2].

Thus, cloud computing may play an important role in many phases of the bioinformatics analysis pipeline, from data management and processing, to data integration and analysis, including data exploration and visualization.

Healthcare requires continuous and systematic innovation in order to remain cost effective, efficient and to provide high-quality services. Many managers and experts predict that cloud computing can improve healthcare services and benefit healthcare research. The work [3] summarizes main opportunities and challenges of cloud computing to improve healthcare services. The author recalls that Schweitzer [4], Haughton [5], and Kabachinski [6] believe that cloud computing can reduce electronic health record (EHR) startup expenses, and licensing fees, and thus this will encourage cloud computing adoption.

Different works reported the successful application of cloud computing in healthcare and biomedicine [7, 8,

*Laboratory of Bioinformatics, Department of Medical and Surgical Sciences, University "Magna Graecia", viale Europa 88100, Catanzaro, Italy (calabreseb@unicz.it).

†Corresponding Author. Laboratory of Bioinformatics, Department of Medical and Surgical Sciences, University "Magna Graecia", viale Europa 88100, Catanzaro, Italy(cannataro@unicz.it).

9, 10]. Besides academic researchers, many world-class software companies have heavily invested in the cloud, such as Microsoft HealthVault (www.healthvault.com).

Despite the many benefits associated with cloud computing to healthcare, there are also several management, technology, security, and legal issues to be addressed. For example, the storing of personal health information into a party, remote data center raises serious issues related to patient privacy. The possibility that patient data could be lost, misused or fall into the wrong hands, could affects rapid adoption.

The aim of this paper is to describe and discuss the most significant applications of cloud computing in the healthcare and biomedicine. The examined fields include bioinformatics, medical informatics, telemedicine and bioimaging. Then the paper focuses on specific requests and issues of such applications on cloud computing. The paper is organized as follows: in the Section 2 cloud computing definition is discussed. Service and delivery models are presented in order to define the cloud-related background. Successively, in the Section 3 the paper focuses on the application of cloud computing in bioinformatics, molecular modeling, medical informatics (EHR), medical imaging, telemedicine. Section 4 summarizes the main problems to be faced when moving biomedical applications on the cloud. Finally, Setion 5 concludes the paper and outlines future developments.

2. Background on Cloud Computing. Even though cloud computing is now becoming the key technology for the storage and analysis of large data sets both in academia and industry, it is not a totally new concept. In fact, it has some relations with grid computing and other technologies such as utility computing, clustering, virtualization systems, and distributed systems.

Specifically, High Performance Computing (HPC) covers all aspects of computing that require a lot of computing power and memory [11]. Common HPC computer environments are multi-core processors, clusters, grids, graphics processing units (GPUs). These computing environments have one thing in common: serial code will not run faster on these computers than on any other normal computer. The power of HPC environments is in fact gained through the possibility of parallel or distributed computing. In other words, exploiting HPC requires extensive rewritting of serial code. On the other hand, cloud computing can be interpreted as the next generation of grid computing because the delivery of computing is seen as a service rather than a product, and moreover, it supports easy to implement distributed computing models such as map-reduce [12].

There are many definitions of cloud computing. The first definition comes from the work of Mell and Grance [1] and is a popular working definition of cloud computing from the National Institute of Standards and Technology, U.S. Department of Commerce. Their definition focuses on computing resources which can be accessed from anywhere and may be provisioned online. It also specifies five characteristics of cloud computing (i.e. on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service), three service models (i.e. Software as a Service, Platform as a Service and Infrastructure as a Service) and four deployment methods (i.e. private cloud, community cloud, public cloud and hybrid cloud). Most of the other definitions do not mention deployment methods. In contrast to other definitions, this one does not explicitly mention virtualization as a key technology.

According to this definition, the five key characteristics of cloud services are:

- on demand self service: a consumer can acquire computing power unilaterally and automatically without requiring any human intervention by the service provider;
- broad network access: the computational capabilities are accessible on the Internet in accordance with standard mechanisms;
- resource pooling: users have the impression that the available resources are unlimited and can be purchased in any amount and at any time. Resources of the service provider come together to serve a variety of consumers, according to a multi-tenant model. The physical and virtual resources are dynamically assigned and reassigned to consumer, based on his/her requests without having any control or knowledge of the exact location of the resources assigned to him/her;
- rapid elasticity: resources are able to be quickly allocated and elastically;
- measured services: the cloud systems automatically control and optimize resource use by evaluating appropriate parameters (e.g, storage, processing power, bandwidth and active user accounts).

Buyya et al. [13] supply a definition with the vision that cloud computing will become the fifth utility and that cloud computing is the next generation data centre. This definition extensively highlights the importance of Service Level Agreement (SLA) and the lack of market oriented resource management (e.g. violation of SLAs,

automatic allocation of resources to fulfil SLAs, etc.) in cloud environments.

Grossman [14] offers a general, working definition of cloud computing, which highlights that cloud computing environments should behave like local data centres.

Vaquero et al. [15] collected 22 excerpts from previous works and fused these into a single definition by studying the common properties of cloud computing. This definition emphasizes the importance of Service Level Agreements (SLA) in order to increase confidence in the cloud environment and defines virtualization as the key enabler of cloud computing.

In the paper [16], Marston et al. explicitly state that cloud computing lowers costs by offsetting capital costs with operational ones. Moreover, they give an interesting business perspective to cloud computing which is missing in almost all the other related publications.

The definition taken from [17] by Intel, offers a particular perspective on cloud computing, from the vantage point of a major hardware manufacturer with a commercial interest in building clouds. It describes three main aspects that help realize cloud computing: *federation* (ease of moving data and services within the cloud), *automation* (cloud services should be provisioned with no human intervention) and *client-awareness* (cloud-based applications can take advantage of the capabilities of the end-point device). The definition also makes explicit reference to security, by evidencing that only authenticated devices may connect to the cloud environment.

Hill et al. [18] give a recent definition of cloud computing that presents cloud computing as a utility.

Finally, the Open Grid Forum (OGF) reports a cloud definition in [19]. Specifically, cloud is a data processing infrastructure providing an extensive integrated client interface permitting information discovery, data management, job management, logging, accounting and monitoring. Most clouds are enclosed inside a single administrative domain, do not provision communication between different administrative domains, and provide limited security.

Thus, it is possible to summarize the main advantages of cloud computing model:

- Service-oriented: computing services are all offered as end-products to customers. Therefore, the administration and maintenance of these devices fall under the responsibility of the cloud vendor.
- Massive scalability: the allocation of resources is flexible and elastic, allowing an efficient management of spikes in computational requirements.
- On-demand resources: resources can be added in real time, sometimes in an automated fashion (e.g. through a process that monitors CPU load and automatically adds more CPU resources once a usage threshold is reached).
- Virtualization: this technology enables the sharing of computer hardware amongst multiple users in a common resource pool, thanks to the abstraction of physical servers and of storage devices. This is, in particular, advantageous when using heterogeneous computing resources which need to be presented as one single platform or when different operating systems are to be run on the same hardware.

2.1. Service models. Cloud services can be classified into three main models:

- *Infrastructure as a Service* (IaaS): this service model is offered in a computing infrastructure that includes servers (typically virtualized) with specific computational capability and/or storage. The user controls all the storage resources, operating systems and applications deployed to, while he/she has limited control over the network settings. An example is Amazon's Elastic Compute Cloud (EC2), which allows the user to create virtual machines and manage them, and Amazon Simple Storage Service (S3), which allows to store and access data, through a web-service interface.
- *Platform as a Service* (PaaS): it allows the development, installation and execution of user-developed applications. Applications must be created using programming languages, libraries, services and tools supported by the provider that constitute the development platform provided as a service. An example is Google Apps Engine, which allows to develop applications in Java and Python, provides for both languages the SDK and uses a plugin for the Eclipse development environment.
- *Software as a Service* (SaaS): customers can use the applications provided by the cloud provider infrastructure. The applications are accessible through a specific interface. Customers do not manage the cloud infrastructure or network components, servers, operating systems or storage. In some cases, it is possible to manage specific configurations of the application.

2.2. Delivery models. Cloud services can be made available to users in different ways. In the following, a brief description of the delivery models is presented:

1. *Public Cloud*: public cloud services are offered by vendors who provide the users/customers the hardware and software resources of their data centers. Examples of public clouds are Amazon[1] (Amazon EC2 provides computational services, while Amazon S3 storage services); Google Apps (which provides software services like Gmail, Google Docs or Google Calendar, and development platform like Google App Engine); and Microsoft Azure[2], that is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. It provides both PaaS and IaaS services and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.
2. *Private Cloud*: private cloud is configured by a user or by an organization for their exclusive use. Services are supplied by computers that are in the domain of the organization. To install a private cloud, several commercial and free tools are available (e.g. OpenStack[3], Eucalyptus[4], Open Nebula[5], Terracotta[6] and VMWare Cloud[7]).
3. *Community Cloud*: it is an infrastructure on which are installed cloud services shared by a community or by a set of individuals, companies and organizations that share a common purpose and that have the same needs. The cloud can be managed by the community itself or by a third-party (typically a cloud service provider).
4. *Hybrid Cloud*: the cloud infrastructure is made up of two or more different clouds using different delivery models, which, while remaining separate entities, are connected by proprietary or standard technology that enables the portability of data and applications.

3. Cloud Computing in Healthcare and Biomedicine. Recent studies indicated that cloud computing can improve healthcare services and benefit biomedical research [10, 12, 20], by offering new possibilities.

An important driver for the adoption of cloud technology in the healthcare is to lower the cost of healthcare delivery. This cost has grown to such huge proportions that governments face serious funding issues. The recognition that technology can improve patient care while reducing costs has meant that governments are willing to push the traditionally slow healthcare industry to a faster pace of adoption. In [21], the authors pointed out that, based on Azures pricing model, the cost of their cloud-based telemedicine service is based on the use of CPU resources, which is charged by USD $ 0.12 per hour, and the use of database, which is charged by USD $9.99 per GB per month. Dudley and his colleagues [22] demonstrated that cloud computing is a viable and cheaper technology that enables large-scale integration and analysis for studies in genomic medicine. They compared computational and economic characteristics of a cloud-based service, with a local institutional cluster.

Another important driver is the big data growth in healthcare and bioinformatics [23]. As the amount of digital information increases, the ability to manage this data becomes a growing problem. This data holds the keys to future clinical advances, but often remains inaccessible to researchers. Cloud Computing can be the enabling factor for data sharing and integration at a large scale. In the paper [24], high performance computing (HPC) solutions in bioinformatics, Big Data analysis paradigms for computational biology, and the issues that are still open in the biomedical and healthcare fields are discussed. Specifically, the authors pointed out that cloud computing addresses Big Data storage and analysis issues in many fields of bioinformatics, thanks to the virtualization that avoid to move too big data.

Moreover, in the specific field of telemedicine, in order to support efficient monitoring and automated analysis of larger patient populations, it is essential to have an infrastructure that provides high throughput, high volume storage and reliable communication. *Horizontal scalability* (i.e. the ability for a system to easily expand

[1] hhttps://aws.amazon.com
[2] https://azure.microsoft.com
[3] https://www.openstack.org
[4] https://www.eucalyptus.com
[5] http://opennebula.org
[6] http://terracotta.org
[7] http://www.vmware.com/

its resource pool to accomodate heavier load) and *geographic scalability* (i.e. the ability to maintain performance, usefulness or usability regardless of expansion from concentration in a local area to a more distributed geographic pattern) represent two requirements that cloud computing can fulfill [25].

In the field of medical imaging, thanks to high-resolution imaging instruments, the volume of data will reach petabytes. Therefore, it is evident that the cloud computing model will provide a potential contribution to satisfy computational needs related to the reconstruction and analysis of medical images and to allow a wide sharing of imaging data, as well as advanced remote analysis.

In the following sections, the main cloud-based applications proposed in the fields of bioinformatics, molecular modeling, medical imaging, telemedicine and medical informatics are illustrated and discussed.

3.1. Cloud-based Bioinformatics Solutions. The cloud computing represents a cost-effective solution for the problems of storing and processing data in the context of bioinformatics. Thanks to the progress of high-throughput sequencing technologies, there was an exponential growth of biological data. Therefore, classical computational infrastructure for data processing have become ineffective and difficult to maintain [26, 27].

The traditional bioinformatics analysis involves downloading of public datasets (e.g. NCBI, Ensembl), installing software locally and analysis in-house. By entering the data and software in the cloud and providing them as a service, it is possible to get a level of integration that improves the analysis and the storage of bioinformatics big-data. In particular, as a result of this unprecedented growth of data, the provision of data as a service (*Data as a Service*, DaaS) is of extreme importance. DaaS provides a data storage in a dynamic virtual space hosted by the cloud and allows to have updated data that are accessible from a wide range of connected devices on the web. An example is represented by the DaaS of Amazon Web Services (AWS) [28], which provides a centralized repository of public data sets, including archives of GenBank, Ensembl, 1000 Genomes Project, Unigene, Influenza Virus. AWS public dataset[8] contains not only data related to the field of biology but also to other scientific fields, such as astronomy, chemistry, climate, economy, etc. All datasets are provided as public AWS services and so can be easily integrated into cloud-based applications.

In the following subsections, examples of SaaS, PaaS and IaaS for several tasks in bioinformatics domain are presented.

3.1.1. Bioinformatics Tools deployed as SaaS. In recent years, there have been several efforts to develop cloud-based tools to execute different bioinformatics tasks [29], e.g. genomic applications[9] [10] [30, 31], sequences alignment [32], gene expression analysis[11] [12] [13] [33, 34, 35]. Other examples of SaaS bioinformatics tools are reported in the following.

CloudBurst[14] is a new parallel read-mapping algorithm optimized for mapping next-generation sequence (NGS) data to the human genome and other reference genomes, for use in a variety of biological analyses including SNP discovery, genotyping and personal genomics [36]. CloudBurst uses the open-source Hadoop implementation of MapReduce to parallelize execution using multiple compute nodes.

PeakRanger[15] [37] is a software package for the analysis in Chromatine ImmunoPrecipitation Sequencing (ChIP-seq) technique. This technique is related to NGS and allows to investigate the interactions between proteins and DNA. Specifically, PeakRanger is a peak caller software package that can be run in a parallel cloud computing environment to obtain extremely high performance on very large data sets.

VAT (Variant Annotation Tool)[16] [38] has been developed to functionally annotate variants from multiple personal genomes at the transcript level as well as to obtain summary statistics across genes and individuals. VAT also allows visualization of the effects of different variants, integrates allele frequencies and genotype data from the underlying individuals and facilitates comparative analysis between different groups of individuals.

[8] http://aws.amazon.com/publicdatasets
[9] http://cloudaligner.sourceforge.net
[10] http://bowtie-bio.sf.net/myrna
[11] http://fx.gmi.ac.k
[12] http://bowtie-bio.sourceforge.net/myrna/index.shtml
[13] http://tinyurl.com/yunbedownload
[14] http://sourceforge.net/projects/cloudburst-bio/
[15] http://www. modencode.org/software/ranger/
[16] http://vat.gersteinlab.org

VAT can either be run through a command-line interface or as a web application. Finally, in order to enable on-demand access and to minimize unnecessary transfers of large data files, VAT can be run as a virtual machine in a cloud-computing environment.

STORMSeq (Scalable Tools for Open-source Read Mapping) [39], is a graphical interface cloud computing solution that performs read mapping, read cleaning and variant calling and annotation with persoanl genome data. At present, STORMSeq costs approximately 2 dollars and 510 hours to process a full exome sequence and 30 dollars and 38 days to process a whole genome sequence. The authors provide this open-access and open source resource[17] as a user-friendly interface in Amazon EC2.

In [40], the authors propose an efficient Cloud-based Epistasis cOmputing (**eCEO**) model for large-scale epistatic interaction in genome-wide association study (GWAS). Given a large number of combinations of SNPs (Single-nucleotide polymorphism), eCEO model is able to distribute them to balance the load across the processing nodes. Moreover, eCEO model can efficiently process each combination of SNPs to determine the significance of its association with the phenotype. Authors have implemented and evaluated eCEO model on their own cluster of more than 40 nodes. The experiment results demonstrate that the eCEO model is computationally efficient, flexible, scalable and practical. In addition, authors have also deployed the eCEO model on the Amazon Elastic Compute Cloud[18].

Cloud4SNP [41] is a novel Cloud-based bioinformatics tool for the parallel preprocessing and statistical analysis of pharmacogenomics SNP DMET microarray data. It is a Cloud-based version of DMET- Analyzer [42], that has been implemented on the Cloud using the Data Mining Cloud Framework [43], a software environment for the design and execution of knowledge discovery workflows on the Cloud [44]. It allows to statistically test the significance of the presence of SNPs in two classes of samples using the well known Fisher test. Cloud4SNP uses data parallelism and employs an optimization technique to avoid the execution of useless Fisher tests, through the filtering of probes with similar SNPs distributions.

ProteoCloud [45] is a freely available, full-featured cloud-based platform to perform computationally intensive, exhaustive searches using five different peptide identification algorithms. ProteoCloud[19] is entirely open source, and is built around an easy to use and cross-platform software client with a rich graphical user interface. This client allows full control of the number of cloud instances to initiate and of the spectra to assign for identification. It also enables the user to track progress, and to visualize and interpret the results in detail. Table 3.1 lists the main SaaS bioinformatics applications and their essential features.

3.1.2. Bioinformatics Platforms deployed as PaaS. Currently, the most used platform (PaaS) for bioinformatics applications is **Galaxy Cloud**[20], which is a Galaxy cloud-based platform for the analysis of data at a large scale. It allows anyone to run a private Galaxy installation on the Cloud exactly replicating functionality of the main site[21], but without the need to share computing resources with other users. With Galaxy Cloud, unlike software service solutions, the users can customize their deployment as well as retain complete control over their instances and associated data; the analysis can also be moved to other cloud providers or local resources, avoiding concerns about dependence on a single vendor. Currently, a public Galaxy Cloud deployment is provided on the popular Amazon Web Services (AWS) cloud; however, it is compatible with Eucalyptus and other clouds [46]. **CloudMan**[22] [47] enables individual bioinformatics researchers to easily deploy, customize, and share their entire cloud analysis environment, including data, tools, and configurations.

In [48], a modular and scalable framework called **Eoulsan**[23], based on the Hadoop implementation of the MapReduce algorithm dedicated to high-throughput sequencing data analysis, is presented. Eoulsan allows users to easily set up a cloud computing cluster and automate the analysis of several samples at once using various software solutions available. Tests with Amazon Web Services demonstrated that the computation cost is linear with the number of instances booked, as the running time with the increasing amounts of data. Eoulsan is implemented in Java, supported on Linux systems and distributed under the LGPL License.

[17] http://www.stormseq.org
[18] http://www.comp.nus.edu.sg/ wangzk/eCEO.html
[19] http://proteocloud.googlecode.com
[20] http://galaxy.psu.edu
[21] Galaxy is an open source, web-based platform for data intensive biomedical research; http://usegalaxy.org/
[22] http://cloudman.irb.hr
[23] http://transcriptome.ens.fr/eoulsan/

TABLE 3.1
SaaS Bioinformatics Applications

Name	Available	Domain	Task/Data	Cloud Type
Cloud Aligner [30]	YES	Genomics (Genome resquencing, short-read aligner)	A map/reduce based application for mapping short reads generated by the next-generation sequencing machine	Public Cloud (AWS)
CloudBlast [32]	YES	Genomics	Massive sequence alignment	Public Cloud (AWS)
Fx [33]	YES	Genomics	RNA-Seq analysis tool	Public Cloud (AWS)
Myrna [34]	YES	Genomics (RNA-sequencing)	Calculation of differential gene expression in large RNA-seq datasets	Public Cloud (AWS)
YunBe [35]	YES	Transcriptomics	Gene set analysis tool for biomarker identification	Public Cloud (AWS)
CloudBurst [36]	YES	Genomics (Genome resquencing, short-read aligner)	Parallel Mapping of NGS data	Public Cloud (AWS)
PeakRanger [37]	YES	Genomics (Sequencing)	Peak caller for ChIP-Seq data	Binaries for an Amazon EC2 image will be available
VAT [38]	YES	Genomics	Framework to functionally annotate variants in personal genomes	Public Cloud (AWS)
StormSeq [39]	YES	Genomics (Sequencing)	Parallel Processing Personal Genomics Data	Public Cloud (AWS)
eCEO [40]	YES	Genomics (Sequencing)	Epistasis Computing Model of SNPs data	Public Cloud (AWS)
Cloud4SNP [41]	No	Genomic	Microarray data analysis	Private Cloud (Microsoft Azure)
ProteoCloud [45]	YES	Proteomics (Mass spectrometry)	Tools for identification of very large sets of spectra using five different peptide identification algorithms	Public Cloud (AWS)

The Table 3.2 summarizes the bioinformatics platforms (PaaS) introduced so far by evidencing the application domain, the type of cloud (public, private, etc.) and the information relative to possible availability.

3.1.3. Bioinformatics Virtual Machines deployed as IaaS. Bionimbus[24] [49] is an open source cloud-computing platform used by a variety of projects to process genomics and phenotypic data. It is based primarily upon OpenStack, which manages on-demand virtual machines that provide the required computational

[24]http://bionimbus.opensciencedatacloud.org

TABLE 3.2
PaaS Bioinformatics Applications

Name	Avail	Domain	Task	Cloud Type	O.S.	Programming Language
GalaxyCloud [46]	YES	Bioinformatics	Cloud-scale Galaxy for large-scale data analysis	Public Cloud (AWS)	Not available	Not available
CloudMan [47]	YES	Bioinformatics	Platform for tool, data, and analysis distribution.	Public Cloud (Nectar)	*NIX	Python
Eoulsan [48]	YES	Genomics	High throughput sequencing analyses	Public Cloud (AWS)	Linux	Java

resources, and GlusterFS, which is a high-performance clustered file system. Bionimbus also includes Tukey, which is a portal, and associated middleware that provides a single entry point and a single sign on for the various Bionimbus resources; and Yates, which automates the installation, configuration, and maintenance of the software infrastructure required. Bionimbus is required to operate under a set of guidelines developed by NIST called FISMA. These requirements served as a guide to improve the security of Bionimbus and to generate the associated documentation. The authors modified the OpenFISMA application to help automate this process, e.g. with their modifications, security scans are performed automatically at regular intervals and the results recorded.

Cloud Virtual Resource, **CloVR**[25] [50] is a new desktop application for push- button automated sequence analysis that can utilize cloud computing resources. CloVR is implemented as a single portable virtual machine (VM) that provides several automated analysis pipelines for microbial genomics, whole genome and metagenome sequence analysis. In addition CloVR supports use of remote cloud computing resources to improve performance for large-scale sequence processing. Supported clouds include the commercial Amazon Elastic Compute Cloud and the academic platforms DIAG [51] and Magellan [52] To provide security and help ensure data privacy, each remote cluster of CloVR VMs uses a unique, randomly generated authentication key. This key is used to enable secure data transfer between instances with Secure Shell (SSH) both within the cloud and over the Internet and between the local client VM and master cloud CloVR VMs.

Cloud BioLinux [53][26] is a publicly accessible Virtual Machine (VM) that enables scientists to quickly provision on-demand infrastructures for high-performance bioinformatics computing using cloud platforms. Users have instant access to a range of pre-configured command line and graphical software applications, including a full-featured desktop interface, documentation and over 135 bioinformatics packages for applications including sequence alignment, clustering, assembly, display, editing, and phylogeny. Cloud BioLinux is available through the Amazon EC2 cloud. For higher security, instead of providing a password through the EC2 launch instance wizard, a user can take the additional step of setting up Secure Shell (SSH) keys, which is fully described in the documentation. Besides the Amazon EC2 cloud, the authors started instances of Cloud BioLinux on a private Eucalyptus cloud and demonstrated access to the bioinformatics tools interface through a remote connection to EC2 instances from a local desktop computer.

3.1.4. Cloud-based Solutions for Molecular Modeling. Different molecular modeling tasks, such as molecular dynamics simulations, quantum mechanical calculations or 3D virtual library construction, are very

[25]http://clovr.org
[26]http://cloudbiolinux.org

TABLE 3.3
Bioinformatics IaaS

Name	Avail.	Domain	Task	Cloud Type	Operating System	Programming Language
CloudBio-Linux [53]	YES	Bioinformatics	A publicly virtual machine for high performance bioinformatics computing	Public (AWS) and private Cloud (Eucalyptus)	Linux, Windows, Mac OSX	Python
CloVR [50]	YES	Genomics	A virtual machine for automated and portable sequence analysis	Cloud Middleware EC2, Eucalyptus, Nimbus	Platform independent	Not available
Bionimbus [49]	YES	Genomics	Cloud-based system for genomic data	Private cloud (OpenStack)	Not available	Not available

computationally intensive tasks and generate or process huge amount of data. Therefore, cloud computing model guarantees for these types of molecular modeling tasks scalability, reliability, and lower cost [54]. The use of cloud computing for molecular modeling applications is still in its infancy, because in order to use cloud computing, many technical tasks must be performed by the user such as configuring the compute nodes, installing the required software, and launching, monitoring and terminating the computation [55].

One of the most sophisticated SaaS for cloud computing are provided by commercial providers such as CycleCloud[27], which offers clusters with preinstalled applications that are widely used in the area of bioinformatics, proteomics, and computational chemistry. The most interesting from a molecular modeling perspective are Gromacs, to perform molecular dynamics (MD) simulations, and ROCS to perform shape-based screening of compounds.

Recently announced AceCloud[28] is a commercial on-demand GPU cloud computing service. The optimization of MD simulation packages for running on GPUs, such as Amber, Gromacs, Lammps, and NAMD is reported to result in large performance increases compared to calculations run on CPUs [56].

The Rosetta software suite[29] facilitates molecular modeling tasks such as prediction of protein and RNA 3D structures, protein-ligand and protein-protein docking, antibody modeling, and enzyme design. Because of the large search spaces of these problems, finding good solutions is highly computationally demanding. To meet this challenge a Seattle biomedical software company Insilicos announced the formation of RosettaATCloud LLC, a joint venture with the University of Washington, that developed Rosetta@cloud, which offers the Rosetta software suite as a cloud-based service hosted on Amazon Web Services.

AutoDockCloud [57] is a workflow system that enables distributed screening on a cloud platform using the molecular docking program AutoDock. AutoDockCloud is based on the open source framework Apache Hadoop, which implements the MapReduce paradigm for distributed computing. In addition to the docking procedure itself, preparation steps, such as generating docking parameters and ligand input files, can be automated and distributed by AutoDockCloud.

[27]http://www.cyclecomputing.com/ cyclecloud/applications
[28]http://www.acellera.com/products/acecloud/
[29]https://www.rosettacommons.org

3.2. Cloud-based Medical Informatics Solutions. A major challenge for medical informatics concerns the storage of heterogeneous data into a single searchable database for clinical purposes or research. The cloud allows greater sharing and integration of Electronic Health Record (EHR), also allowing a reduction in costs related to hardware, software, network and staff. Following these considerations, it is expected a considerable increase in the use of the cloud for storing personal health information online [58].

Many studies have proposed different cloud-based frameworks in an attempt to improve the EHR. Among these, Chen et al. [59] have proposed a new system of access control in cloud computing environment that allows to securely access the patient's medical record even in multi-user system.

Doukas et al. [60] have presented the implementation of a mobile system that allows data storage of e-health, updating and recovery with cloud computing.

In addition to the works presented so far as result of academic research, there are also several commercial systems made by companies that provide services of medical records.

An example is the Microsoft HealthVault[30] platform developed by Microsoft for managing health information of its members. The system helps people to collect, store and share their medical information with family members and health care providers. In addition, it also allows the management of fitness and diet connecting with third-party applications.

HealthVault gives particular attention to privacy and security of health information users. In the following, some security mechanisms are listed:

- Authentication: users can access the system through various mechanisms such as Facebook, Windows Live ID;
- Authorization: The system asks the user's permission before enabling any data access by an application;
- User Control: Users are able to control how their data have to be shared, explicitly granting or revoking consent for third-party applications to access their account. Users are also able to close the account at any time or to remove specific information.
- Auditing: a log that records the data access is built-in into the system and available to users.
- Source of data: the data in HealthVault contain the source of the data and applications and one can use this information to determine how to treat information coming from different sources. The digital signature also allows users to optionally verify the integrity of the data and their source.

Fusion [61] is a cloud-based experimental platform that aims to manage in a simple and secure way, health information of patients allowing them to share between operators in the healthcare system. The key features are low cost and large scale. With regard to data protection, the Fusion architecture includes mechanisms to enable the security and privacy and traceability of access to stored medical data. Specifically, since all data residing in the cloud are encrypted, and the Fusion Store by itself cannot decrypt the data, the security and privacy of data are guaranteed. Different health care professionals can use Fusion to store patient information and share it securely. Fusion also supports the integration of proprietary systems of health information management and provides integrated data sharing mechanisms.

Several hospitals and healthcare organizations have been adopted cloud computing solution. On July 2011 Chelsea and Westminster Hospital set a cloud computing system to manage and store their EHRs. With this system, patients have full control over who has access to their health records. However the Cloud provider company must prove the security of its system. To solve this problem the Cloud provider company has implemented a security mechanism in which users have to pass multiple ID checkpoints to access the database [83].

Another example of Cloud Computing and e-Health services is found at the "Bambino Gesu" Italian Hospital. This hospital, placed in Rome, is famous for being one of the largest research and treatment centers in the field of pediatrics. Since hospital people are using Cloud, they have experienced advantages such as: better collaboration between the medical staff, better connection with patients and more free time for the IT group [62].

Cloudera has been working with Mount Sinai School of Medicine clinical and academic faculty to develop new methods and systems for analyzing data in a variety of important areas including genomics and multiscale biology.

[30] https://www.healthvault.com/

Recently, Apple has announced new Health app that collects all health and fitness data (heart rate, calories burned, blood sugar, cholesterol) in order to give to the user a clear and current overview of his/her health. It is possible also to create an emergency card with important health information, for example, the blood type or allergies.

The Google project, called Baseline Study, will collect anonymous genetic and molecular information from 175 people – and later thousands more – to create what the company hopes will be the fullest picture of what a healthy human being should be. Baseline will amass a much larger and broader set of new data. The hope is that this will help researchers detect killers such as heart disease and cancer far earlier, pushing medicine more toward prevention rather than the treatment of illness. Google has already built one of the world's largest networks of computers and data centers to serve online-search results quickly.

3.3. Cloud-based Medical Imaging Solutions. Many studies have demonstrated the utility of MapReduce to solve the problems of medical imaging on a large scale in a cloud computing environment. For example, Meng et al. [63] have developed a fast and scalable technique for image reconstruction of 4D CT (Four Dimension Computer Tomography) cone beam with MapReduce in a cloud computing environment.

Avila-Garcia et al. [64] proposed a framework based on cloud for the analysis of images in order to diagnose colorectal cancer.

Silva et al. [65] proposed a system based on a set of DICOM routers interconnected through an infrastructure of public cloud to support the exchange of medical images between institutions.

In the field of radiology, Patel [66] highlighted the main advantages of the use of cloud-based SaaS, PaaS and IaaS applications. Specifically, SaaS allows access to three-dimensional visualization and quantitative analysis tools from any computer with a web client and internet access. This permits to solve license problems and the need of powerful local workstation. iNtuition UNLIMITED[31] cloud-based service by TeraRecon, Inc is an example of commercial product for radiologists. Moreover, the sharing of images between radiologist of different institutions and the increasing storage capacities needed bring to adoption of cloud-based solutions. Commercial vendors such as Insite One and Pacs Drive offer cloud based imaging archiving and storing solutions.

Security and privacy of patients images should be ensured in order to avoid unauthourized access. Kagadis et al. [67] discuss the need to adopt specific methods of ensuring data security including encrypting all data transferred over the internet using SSL (Secure Sockets Layer), and giving only an authorized user access to the system.

3.4. Cloud-based Telemedicine Solutions. In many countries, increasing longevity and declining fertility rates have led to the aging of the population and, consequently, the need for medical care is increased. In addition, there is a large population that does not receive good health care services because they live in rural communities.

Telemedicine, through the provision of remote monitoring services and medical assistance, allows to reach people living in rural areas and to reduce the healthcare costs. Moreover, the use of cloud computing could allow to delivery better telemedicine services. Specifically, the cloud can help provide effective treatment and care of patients due to its benefits such as on-demand access anywhere, anytime, low costs and high elasticity.

In this subsection, a description of cloud-based telemedicine applications are illustrated, highlighting the advantages and limits of the proposed approaches. Table 3.4 listed the features of the cited telemedicine applications. The major number of reported studies focus on telecardiology [68].

Pandey et al. in [25] propose the development of an autonomic cloud environment that collects people's health data and stores them to a cloud-based information repository and facilitates analysis on the data using software services hosted in the cloud. The scheme by Pandey et al. integrates mobile and Cloud technologies with electrocardiogram (ECG) sensors to enable remote monitoring of patients with heart-related problems such as cardiac arrhythmias. The patients connect the sensors to their body and then run an app on a mobile device. The app connects to the sensors via Bluetooth. The app will then periodically upload data to the Cloud. The user can then download graphs from the Cloud which represent the users health states. The scheme also implements middleware in the Cloud. There are web services for users to analyse their ECG, draw graphs, etc. The current scheme does not handle security issues, even if the authors propose and discuss the use of a trusted

[31] http://www.terarecon.com/cloud/

third party-certification authority to implement a Public Key Infrastructure (PKI) based authentication and authorization system.

Also in the paper [69], security and privacy issues are not discussed. The authors propose a cloud-based system for monitoring and real-time analysis of electrocardiographic signals (ECG). The system has been designed to be accessible from anywhere using a smartphone, a tablet, desktop or laptop. The ECG signals can be collected by wearable ECG devices and transmitted via Bluetooth to the client, which then transmits to the cloud. In particular, the web application has been developed using the Spring framework[32]. The final application was deployed to a server in the Amazon Cloud using Amazon Web Services (AWS). The transmission of data between the client and the server has been done by using HTTPS (Hypertext Transfer Protocol Secure). The server, upon receiving the data, controls the quality of the signals and, if necessary, applies appropriate pre-processing and enhancement techniques. Subsequently, the system extracts information related to heart rate variability (RR interval[33]). The system provides an interface for the patient and for the physician. The patient may display the ECG trace and the extracted parameters. The physician can visualize these data, and according to their value, he/she could make a diagnosis that is sent to the patient.

In [21], Hsieh et al. describe the implementation of a cloud computing based telemedicine application, which can upload 12-lead ECG reading to a cloud service from which they can be visualized on a variety of network connected devices, both mobile and fixed. This cloud application supports a variety of clinically used 12-lead ECG devices, such as Philips XML-ECG, HP compatible SCP-ECG, and Mortara DICOM-ECG by first transcoding the data from a vendor-specific format to an XML format in a public cloud computing platform, Microsoft Azure. To safeguard the ECG data in the cloud, they have designed a security mechanism to prevent them from theft and protect the patients privacy, including authentication for the use of Web roles and Worker roles, data encryption during message communications among roles, and ECG file encryption and verification while ECG report are retrieved in storage account and database. Moreover, ECG files are transmitted via secure sockets layer (SSL) based HTTP (HTTPS) where ECG files are protected by certificate based encryption and verification instead of plain text based HTTP.

Body Sensor Network (BSN), with their huge amount of gathered sensor data and their limited processing power, can be empowered by exploiting a Cloud computing infrastructure to realize an integrated platform that provides [70]:

- the ability to utilize heterogeneous sensors;
- scalability of data storage;
- scalability of processing power for different kinds of data analysis;
- global access to the processing and storage infrastructure;
- easy sharing of results;
- pay- as-you-go pricing for using BSN services;

Fortino [71] introduced the BodyCloud architecture which enables the management and monitoring of body sensor data via the Cloud. It provides functionality for receiving and managing sensor data in a seamless way from a body sensor network (BSN). BodyCloud also comprises of a scalable framework that allows support for multiple data streams required for running concurrent applications. Many challenges (interoperability, heterogeneity, security, data validation and consistency) strictly related to BSNs are discussed in [72].

In [73] a cloud computing based Voice over IP (VoIP) service for diabetic patients self care management, is presented. The patients subscribing the cloud service received VoIP calls with pre-recorded voice messages as self-care reminders. In this study, the participants obtained better glycemic control than the patients without subscribing the service.

In [74] a cloud-based intelligent system for real-time monitoring of users with diabetes (Cloud Based Intelligent Health Care Service, CBIHCS), has been presented. CBIHCS allows the monitoring of different vital parameters such as blood pressure, glucose and ECG. The collected data are sent via bluetooth to mobile devices which then transmit them to the cloud. Measured signals and patients information are stored on the cloud. Considering the analysis of the data, a PCA (Principal Component Analysis) is applied for the selection

[32] http://projects.spring.io/spring-framework/

[33] the time elapsing between two consecutive R waves in the electrocardiogram or the interval from the peak of one QRS complex to the peak of the next as shown in the electrocardiogram

of attributes, and classification techniques, such as KNN and Naïve Bayes have been used to determine the state of health of the users. Furthermore, the authors have implemented statistical prediction techniques to determine the patterns of resource use and have proposed a simple heuristic for a dynamic elastic infrastructure. In particular, the authors highlight that in the development of the system, the standard Web Service Resource Framework (WSRF) has been adopted in order to facilitate the deployment of the final system on a cloud, like Amazon EC2. They carried out two types of experiments: the aim of the first experimentation was to assess the classifier's performances and to check the proper functioning of the system; in the second experimentation a study on dynamic allocation of resources was performed. Moreover the authors have been addressed security challenges in a cloud hosted health care application: they have implemented security mechanisms at multiple levels and provide role based access control to ensure the protection of critical medical data of patients. They propose to integrate the use of symmetric cryptosystems for authentication and role based access control (RBAC) mechanisms for authorization.

In Kim et al. [75], another cloud-based system for vital signs monitoring is presented. The framework is composed of four distinct modules: a module to receive vital sign data using standardized messaging methods; a module to transform these data into a standardized schema; a module to evaluate the health state using biosignal data; and a service-oriented architecture (SOA) component module that allows users to access medical services with standardized messaging methods. Security and privacy issues have not been discussed, but a new concept of HaaS Healthcare as Service has been introduced: healthcare systems are defined as services in order to secure interoperability.

The paper [76] presents an expert diagnosis system based on cloud computing. It classifies a users fitness level based on supervised machine learning techniques. This system is able to learn and make customized diagnoses according to the users physiological data, such as age, gender, and body mass index (BMI). In addition, an elastic algorithm based on Poisson distribution is presented to allocate computation resources dynamically. The paper does not present any detailed information about cloud implementation.

In [77], a new mobile app, Dental Calendar, combined with cloud services specific for dental care, has been created. This new system would remind patients about every scheduled apointment, and help them take pictures of thier own oral cavity parts that require dental treatment and send them to dentists along with a symptom description. All this information would be sent to dentists through cloud services. The authors propose a general system's architecture, but they do not discuss it in detail.

Finally, Thilakanathan et al. [78] addressed the issues of privacy and security in the domain of mobile telecare and Cloud computing. They demonstrated a telecare application that will allow doctors to remotely monitor patients via the Cloud. They use this system as a basis to showcase a model that will allow patients to share their health information with other doctors, nurses or medical professionals in a secure and confidential manner. The key features of the model include the ability to handle large data sizes and efficient user revocation. The authors also addressed the problems of achieving efficient user revocation, especially when considering large data sizes. For this, they defined a secure data sharing model and protocol, and demonstrated the feasibility of the protocol through a prototype which combines smartphone, Bluetooth and Cloud computing technologies.

4. Cloud challenges in healthcare and biomedicine. Clinical data must be hosted on publicly accessible servers according to privacy and security rules, such as the Health Insurance Portability and Accountability Act (HIPAA). The EHRs are exposed to possible abuse and require security measures based on the identity management, access control, policy integration and compliance management.

Typical entities in a cloud based health record system are patients, hospital staff, such as doctors, nurses, pharmacies, and laboratory staff, insurance companies, and the cloud service providers. Due to the distributed architecture of the cloud, the patient EHRs are stored at and shared among many third-party providers. Therefore, the data is susceptible to unauthorized access and attacks. Specifically, the paper [79] claims that storing huge volumes of patients' sensitive medical data in third-party cloud storage is susceptible to loss, leakage or theft. The privacy risk of cloud environment includes the failure of mechanisms for separating storage, memory, routing, and even reputation between different tenants of the shared infrastructure. The centralized storage and shared tenancy of physical storage space means the cloud users are at higher risk of disclosure of their sensitive data to unwanted parties.

Threats to the data privacy in the cloud include spoofing identity, tampering with the data, repudiation,

TABLE 3.4
Cloud-based telemedicine systems

References	Application Domain	Task/Data	Cloud	Security and Privacy
[25]	Telecardiology	ECG monitoring	Public Cloud (Amazon Web Services)	No
[69]	Telecardiology	ECG real-time monitoring	Public Cloud (Amazon Web Services)	No
[21]	Telecardiology	ECG monitoring	Public Cloud (Microsoft Azure)	Yes
[71]	Body Sensor Network (BSN)	Management and Monitoring of body sensor data	Not available	Yes
[73]	Teleconsulting	Diabetes	Not available	No
[74]	Diabetes	Monitoring of different vital parameters	Public Cloud (Amazon Web Services)	Yes
[75]	Telemonitoring	VMonitoring of vital parameters	Not available	Not available
[76]	Wellness	Expert system for health fitness level classification	Not available	Not available
[77]	Dental care	Appointment reminders and transmission of oral cavity pictures	Not available	Not available

and information disclosure. In spoofing identity attack, the attacker pretends to be a valid user whereas data tampering involves malicious alterations and modification of the content. Repudiation threats are concerned with the users who deny after performing an activity with the data. Information disclosure is the exposure of information to the entities having no right to access information. The same threats prevail for the health data stored and transmitted on the third-party cloud servers.

Therefore, confidentiality and integrity of the stored health data are the most important challenges elevated by the healthcare and biomedicine cloud-based systems.

A secure protection scheme will be necessary to protect the sensitive information of the medical record. There is considerable work on protecting data from privacy and security attacks. NIST [80] has developed guidelines to help consumers to protect their data in the Cloud. The paper [81] evidences that using cryptographic storage significantly enhances security of the data. The paper discusses the main mechanisms to be adopted in order to guarantee and satisfy the previous cited issues. Specifically, the authors present and discuss the utility of cryptographic and non-cryptographic approaches. The cryptographic approaches to mitigate the privacy risks utilize certain encryption schemes and cryptographic primitives. Conversely, non-cryptographic approaches mainly use a policy based authorization infrastructures that allow the data objects to have access control policies. Particularly, in the public cloud environment operated by the commercial service providers and shared by several other customers, data privacy and security are the most important requirements.

Abbas et al. [82] summarized the security and privacy requirements for cloud-based applications in healthcare and biomedicine fields in the following way:

1. Integrity: it is needed to ensure that the health data captured by a system or provided to any entity is true representation of the intended information and has not been modified in any way.
2. Confidentiality: the health data of patients is kept completely undisclosed to the unauthorized entities.
3. Authenticity: the entity requesting access is authentic. In the healthcare systems, the information provided by the healthcare providers and the identities of the entities using such information must be

verified.

4. Accountability: an obligation to be responsible in light of the agreed upon expectations. The patients or the entities nominated by the patients should monitor the use of their health information, whenever that is accessed at hospitals, pharmacies, insurance companies etc.
5. Audit: it is needed to ensure that all the healthcare data is secure and all the data access activities in the e-Health cloud are being monitored.
6. Non-Repudiation: repudiation threats are concerned with the users who deny after performing an activity with the data. For instance, in the healthcare scenario neither the patients nor the doctors can deny after misappropriating the health data.
7. Anonymity: it refers to the state where a particular subject cannot be identified. For instance, identities of the patients can be made anonymous when they store their health data on the cloud so that the cloud servers could not learn about the identity.
8. Unlinkability: it refers to the use of resources or items of interest multiple times by a user without other users or subjects being able to interlink the usage of these resources. More specifically, the information obtained from different flows of the health data should not be sufficient to establish linkability by the unauthorized entities.

Finally, in the cloud, physical storages could be widely distributed across multiple jurisdictions, each of which may have different laws regarding data security, privacy, usage, and intellectual property. For example, the US Health Insurance Portability and Accountability Act (HIPAA) restricts companies from disclosing personal health data to nonaffiliated third parties. Similarly, the Canadian Personal Information Protection and Electronic Documents Act (PIPEDA) limits the powers of organizations to collect, use, or disclose personal information in the course of commercial activities. However, a provider may, without notice to a user, move the users information from jurisdiction to jurisdiction. Data in the cloud may have more than one legal location at the same time, with differing legal consequences.

In summary, security and privacy of data are the main challenges to be faced when deploying healthcare and biomedicine applications on the cloud. However, while in the healthcare community this awareness starts to be recognized by the various actors involved in data production, collection and decision making, as demonstrated by several initiatives and approaches developed, in biomedicine applications, based on the integrated analysis of biological (mainly omics) data and clinical data, there is not yet a clear indication on how security and privacy issues are to be faced. Most probably, the solutions that start to be adopted in healthcare could be adapted to biological and clinical data used in such applications.

5. Conclusions. Applications and services in health care and biomedicine pose quite demanding requirements. The fulfillment of these requirements could results in an improvement in the provision of services to patients in health care, as well as an increase in knowledge in the biomedical field. As shown and discussed in this paper, in order to meet these requirements, the use of the cloud computing model is required. Although supercomputing or Grid Computing can provide the computational power and the storage required in biomedical applications such as medical imaging or electronic medical record, the elasticity in providing such resources, a clear definition of Service Level Agreement, and the possibility for the customer to use a pay-per-use model, make the Cloud more suitable to support those applications.

Naturally, the adoption of this technology with its benefits will determine a reduction of costs and the possibility of also providing new services. Different comparative analysis demonstrate the potentials of cloud technology in reducing cost of IT organization: in the case of cloud adoption, institutions and laboratories are free from the expense and having to install and maintain applications locally [83]. Knaus et al. [84] focus on the financial aspects of grid and cloud computing. The authors discuss and compare the costs of a local cluster maintained by a department to a comparable cloud offer. Specifically, they list all direct costs and estimate all indirect costs and, with the help of a simulation study they estimate the costs for typical research projects in biomedicine. In their case, they conclude that using a cloud is a more viable option.

Also in the paper [85], the authors analyzed some economic and practical aspects of exploiting cloud computing for the in silico drug discovery and they highlight the advantages for small-medium laboratories working in the field of biotechnology, which typically do not have the possibility to invest a sufficient amount of time and money in creating and maintaining an in-house ICT infrastructure that suits the processing of the

large amount of bioinformatics data that are nowadays produced.

However, it is important to emphasize that the use of cloud in these fields is featured still by a number of open issues, such as the security and privacy, that require a rapid and efficient solution.

6. Acknowledgments. This work has been partially funded by the *PON DICET- INMOTO-ORCHESTRA (PON04a2-D)* project, funded by the Italian Ministry of Research and Education (MIUR).

REFERENCES

[1] P. Mell and T. Grance, *The NIST definition of cloud computing*, 2011.
[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, *A view of cloud computing*, Commun ACM, 53(4):50-58, 2010.
[3] A.M.H. Kuo, *Opportunities and Challenges of Cloud Computing to Improve Health Care Services*, J Med Internet Res., 13(3):e67, 2013.
[4] E.J.Schweitzer, *Reconciliation of the cloud computing model with US federal electronic health record regulations*, J Am Med Inform Assoc., 2011, doi: 10.1136/amiajnl-2011-000162.amiajnl-2011-000162.
[5] J.Haughton, *Year of the underdog: Cloud-based EHRs*, Health Manag Technol,32(1):9, 2011.
[6] J. Kabachinski, *What's the forecast for cloud computing in healthcare?*, Biomed Instrum Technol., 45(2):14650, 2011.
[7] Y. Lin, C. Yu and Y. Lin, *Enabling Large-Scale Biomedical Analysis in the Cloud*, BioMed Research International, 2013, http://dx.doi.org/10.1155/2013/185679.
[8] C. Lin, S. S Abdul, D. L. Clinciu, J. Scholl, X. Jin, H. Lu, S. S. Chen, U. Iqbal, M. J. Heineck and Y. Li, *Empowering village doctors and enhancing rural healthcare using cloud computing in a rural area of mainland China*, Computer methods and programs in biomedicine, 113:585592, 2014.
[9] J. Ekanayake, T. Gunarathne and J. Qiu, *Cloud Technologies for Bioinformatics Applications*, IEEE Transactions on parallel and distributed systems, 22(6):998-1011, 2011.
[10] S. P. Ahuja, S. Mani and J. Zambrano *A Survey of the State of Cloud Computing in Healthcare*, Network and Communication Technologies, 1(2):12–19, 2012.
[11] M. Cannataro, *Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare*, Medical Information Science Reference Hershey, PA: IGI Global Press; 2009.
[12] M. J. A. Eugster, M. Schmid, H. Binder and M. Schmidberger, *Grid and Cloud Computing Methods in Biomedical Research*, Methods of Information in Medicine 1:62–64, 2013.
[13] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, *Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility*, Future Generation Computer Systems, 25:599–616, 2009.
[14] R. L. Grossman, *The case for cloud computing*, IT Professionals, 11:23–27, 2009.
[15] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, *A break in the clouds: Towards a cloud definition*, ACM SIGCOMM Computer Communication Review, 39:50–55, 2009.
[16] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang and A. Ghalsasi, *Cloud computing - the business perspective*, Decision Support Systems, 51:176-189, 2011.
[17] Intel, *Intel's Vision of the Ongoing Shift to Cloud Computing*, (2010), http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/cloud-computing-intel-cloud-2015-vision.pdf.
[18] R. Hill, L. Hirsch, P. Lake and S. Moshiri, *Guide to Cloud Computing*, 2013.
[19] OGF Production Grid Infrastructure, *GFD.181 Glossary of Acronyms and Terms*, 20 March 2011.
[20] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester and P. Reynolds, *Cloud computing: A new business paradigm for biomedical information sharing*, Journal of Biomedical Informatics, 43:342–353, 2010.
[21] J. Hsieh and M. W. Hsu, *A cloud computing based 12-lead ECG telemedicine service*, BMC Medical Informatics and Decision Making, 12:77, 2012.
[22] J. T. Dudley, Y. Pouliot, J. R. Chen, A. A. Morgan and A. J. Butte, *Translational Bioinformatics in the cloud: an affordable alternative*, Genome Medicine, 2(51), 2010.
[23] Greene C. S, Tan J., Ung M., Moore J. H. and Cheng C., *Big data bioinformatics*, Journal of Cell Physiology 229(12):1896-1900, 2014.
[24] I. Merelli, H. Prez-Snchez, S. Gesing and D. DAgostino, *Managing, Analysing, and Integrating Big Data in Medical Bioinformatics: Open Problems and Future Perspectives*, BioMed Research International, 2014, http://dx.doi.org/10.1155/2014/134023.
[25] S. Pandey, W. Voorsluys, S. Niu, A. doker and R. Buyya, *An autonomic cloud environment for hosting ECG data analysis services*, Future Generation Computer Systems, 28:147–154, 2012.
[26] E. E Schadt, M. D. Linderman, J. Sorenson, L. Lee and G. P. Nolan, *Cloud and heterogeneous computing solutions exist today for the emerging big data problems in biology*, Nature Reviews Genetics, 12(3):224, 2011.
[27] R. L. Grossmann and K. P. White, *A vision for a biomedical cloud*, Journal of Internal Medicine 271(2):122–130, 2011.
[28] V. A. Fusaro, P. Patil, E. Gafni, D. P. Wall and P. J. Tonellato, *Biomedical Cloud Computing With Amazon Web Services*, Plos Computational Biology, 7(8), 2011.
[29] L. Dai, X. Gao, Y. Guo, J. Xiao and Z. Zhang, *Bioinformatics clouds for big data manipulation*, Biology Direct, 7(43), 2012.

[30] T. Nguyen, W. Shi and D. Ruden, *CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping*, BMC Research Notes, 4(171), 2011.
[31] B. Langmead, M. C. Schatz, J. Lin, M. Pop and S. L. Salzberg, *Searching for SNPs with cloud computing*, Genome Biology, 10:R134, 2009.
[32] A. Matsunaga, M. Tsugawa and J. Fortes, *CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications*, eScience, 2008. eScience '08.
[33] D. Hong, *FX: an RNA-Seq analysis tool on the cloud*, Bioinformatics, 28(5):721–723, 2012.
[34] B. Langmead, K. D Hansen and J. T. Leek, *Cloud-scale RNA-sequencing differential expression analysis with Myrna*, Genome Biology 11(R83), 2010.
[35] L. Zhang, S. Gu, B. Wang, Y. Liu, Y. and F.Azuaje, *Gene set analysis in the cloud*, Bioinformatics 28(2):294–295, 2012.
[36] M. C. Schatz, *CloudBurst: highly sensitive read mapping with MapReduce*, Bioinformatics, 25(11):1363–1369, 2009.
[37] X. Feng, R. Grossman and L. Stein, *PeakRanger: A cloud-enabled peak caller for ChIP-seq data*, BMC Bioinformatics, 12(139), 2011.
[38] L. Habegger,S. Balasubramanian, D. Z. Chen, E. Khurana, A. Sboner, A. Harmanci, J. Rozowsky, D. Clarke, M. Snyder and M. Gerstein, *VAT: a computational framework to functionally annotate variants in personal genomes within a cloud-computing environment*, Bioinformatics, 28(17):2267–2269, 2012.
[39] K. J. Karczewski, G. H. Fernald, A. R. Martin, M. Snyder, N. P. Tatonetti and J. T. Dudley, *STORMSeq: An Open-Source, User-Friendly Pipeline for Processing Personal Genomics Data in the Cloud*, PlosOne, 9(1), (2014).
[40] Z. Wang, Y. Wang, K. L. Tan, L. Wong and D. Agrawal, *eCEO: an efficient Cloud Epistasis cOmputing model in genome-wide association study*, Bioinformatics, 27(8):1045–1051, 2011.
[41] G. Agapito, M. Cannataro, P. H. Guzzi, F.Marozzo, D. Talia and P. Trunfio, *Cloud4SNP: Distributed Analysis of SNP Microarray Data on the Cloud*, In Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics (BCB'13).
[42] P. H. Guzzi, G. Agapito, M. T. Di Martino, M. Arbitrio, P. Tagliaferrri, P. Tassone and M. Cannataro, *DMET-analyzer: automatic analysis of affymetrix DMET data*, BMC Bioinformatics, 13:258, 2012.
[43] F. Marozzo, D. Talia and P. Trunfio, *A cloud framework for big data analytics workflows on Azure*, In Proc. of the 2012 High Performance Computing Workshop, 2012.
[44] F. Marozzo, D. Talia and P. Trunfio, *Using clouds for scalable knowledge discovery applications*, In Euro-Par Workshops, pages 220227, Rhodes Island, Greece, 2012.
[45] T. Muth, J. Peters, J. Blackburn, E. Rapp and L. Martens*ProteoCloud: A full-featured open source proteomics cloud computing pipeline*, Journal of Proteomics, 88:104–108, 2013.
[46] E. Afgan, D. Baker, N. Coraor, H. Goto, I.M. Paul, K. D. Makova, A. Nekrutenko and J. Taylor, *Harnessing cloud computing with Galaxy Cloud*, Nature Biotechnology, 29(11):972–974, 2011.
[47] E. Afgan, B. Chapman, and J. Taylor, *CloudMan as a platform for tool, data and analysis distribution*, BMC Bioinformatics 13(315), (2012).
[48] L. Jourdren, M. Bernard, M. A. Dillies and S. Le Crom, *Eoulsan: a cloud computing-based framework facilitating high throughput sequencing analyses*, Bioinformatics, 11(28):1542–1543, 2012.
[49] A. P.Heath, M. Greenway, R. Powell, J. Spring, R. Suarez, D. Hanley, C. Bandlamudi, M. E. McNerney, K. P. White and R. L. Grossman, *Bionimbus: a cloud for managing, analyzing and sharing large genomics datasets*, International Journal of American Medical Informatics Association, 2014, doi:10.1136/amiajnl-2013-002155.
[50] S. V. Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangala, R.Riley, C. Arze, J.R. White, O. White and W. F. Fricke, *CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing*, BMC Bioinformatics, 12(356), 2011.
[51] *Data Intensive Academic Grid*, http://diagcomputing.org/.
[52] *Magellan: Argonnes DOE Cloud Computing*, http://magellan.alcf.anl.gov/.
[53] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field and K. E. Nelson, *Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community*, Bioinformatics, 13(42), 2012.
[54] J. P. Ebejer, S. Fullea, G. M. Morris and P. W. Finn, *The emerging role of cloud computing in molecular modelling*, Journal of Molecular Graphics and Modelling, 44:177–187, 2013.
[55] A. Wong and A. M.Goscinski, *The Design and Implementation of the VMD Plugin for NAMD Simulations on the Amazon Cloud*, International Journal of Cloud Computing and Services Science (IJ-CLOSER), 1(4):155–171, 2012.
[56] M. S.Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand and A. L. Beberg, *Accelerating molecular dynamic simulation on graphics processing units*, Journal of Computational Chemistry, 30:864–872, 2009.
[57] S. R. Ellingson and J. Baudry, *High-throughput virtual molecular docking with AutoDockCloud*, Concurrency and Computation: Practice and Experience, 26(4):907–916, 2012.
[58] G. Fernandez-Cardenosa, I. de la Torre-Dez, M. Lpez-Coronado, and J. Rodrigues, *Analysis of Cloud-Based Solutions on EHRs Systems in Different Scenarios*, Journal of Medical Systems, 36:3777-3782, 2012.
[59] T. S. Chen, C. H. Liu, T. L. Chen, C. S. Chen, J. G. Bau and T. C. Lin, *Secure Dynamic Access Control Scheme of PHR in Cloud Computing*, Journal of Medical systems, 36(6):4005-4020, 2012.
[60] C. Doukas, T. Pliakas, and I. Maglogiannis, *Mobile healthcare information management utilizing Cloud Computing and Android OS*, Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1037-1040, 2010.
[61] S. Basu, A. H. Karp, J. Li, J. Pruyne, J. Rolia, S. Singhal, J. Suermondt and R. Swaminathan, *Fusion: Managing Healthcare records at cloud scale*, Computer, 45(11):42–49, 2012.
[62] L. L. Bosch-Andersen, *Hospital uses cloud computing to improve patient care and reduce costs*,

http://www.microsoft.eu/2011/04/15/hospital-uses-cloud-computing-to-improve-patient-care-and-reduce-costs/.
[63] B. Meng, G. Pratx and L. Xing, *Ultrafast and scalable cone-beam CT reconstruction using MapReduce in a cloud computing environment*, Medical Physics, 38(12):6603–6609, 2011.
[64] M. S. Avila-Garcia, A. E. Trefethen, M. Brady, F. Gleeson, F. and D. Goodman, *Lowering the Barriers to Cancer Imaging*, Fourth IEEE International Conference on eScience (2008).
[65] L. A. Bastio Silva, C. Costa and J. L. Oliveira, *DICOM relay over the cloud*, International Journal of Computer Assisted Radiology and Surgery, 8(3):323–33, 2013.
[66] R. P. Patel, *Cloud computing and virtualization technology in radiology*, Clinical Radiology, 67:1095–1100, 2012.
[67] G. C. Kagadis, C. Kloukinas, K. Moore, J. Philbin, P. Papadimitroulas, C. Alexakos, P. G. Nagy, D. Visvikis and W. R. Hendee, *Cloud computing in medical imaging*, Medical Physics, 40, 2013, doi: 10.1118/1.4811272.
[68] J. Hsieh, A. Li and C. Yang, *Mobile, Cloud, and Big Data Computing: Contributions, Challenges, and New Directions in Telecardiology*, International Journal of Environmental Research and Public Health, 10:6131-6153, 2013 doi:10.3390/ijerph10116131.
[69] H. Xia, I. Asif and X. Zhao, *Cloud-ECG for real time ECG monitoring and analysis*, Computer Methods and Programs in Biomedicine, 110:253–259, 2013.
[70] G. Fortino and M. Pathan, *Integration of Cloud computing and body sensor networks*, Future Generation Computer Systems 35:57–61, 2014.
[71] G. Fortino, D. Parisi, V. Pirrone and G. Di Fatta, *BodyCloud: A SaaS approach for community Body Sensor Networks*, Future Generation Computer Systems, 35:62–79, 2014.
[72] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski and R. Jafari, *Enabling effective programming and flexible management of efficient body sensor network applications*, IEEE Trans. Hum.Mach. Syst. 43(1):115133, 2013.
[73] J. D. Piette, M. O. Mendoza-Averales, M. Ganser, M. Mohamed, N. Marine and S. Krishnan, *A preliminary study of a cloud-computing model for chronic illness self-care support in an underdeveloped country*, American Journal of Preventive Medicine, 40(6), (2011).
[74] P. D. Kaur and Chana, I., *Cloud based intelligent system for delivering health care as a service*, Computer methods and programs in biomedicine, 113:346-359, 2014.
[75] T. W. Kim and H. C. Kim, *A healthcare system as a service in the context of vital signs: Proposing a framework for realizing a model*, Computers and Mathematics with Application, 64:1324–1332, 2012.
[76] K. C. Tseng and C. C. Wu, *An Expert Fitness Diagnosis System Based on Elastic Cloud Computing*, The Scientific World Journal, 2014, http://dx.doi.org/10.1155/2014/981207.
[77] Y. Lin, K. L. Peng, J. Chen, J. Y. Tsai, Y. C. Tseng, J. R. Yang and M. H. Chen, *Improvements in dental care using a new mobile app with cloud services*, Journal of the Formosan Medical Association, 2014, DOI: http://dx.doi.org/10.1016/j.jfma.2014.02.009.
[78] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo and L. Alem, *A platform for secure monitoring and sharing of gengene health data in the Cloud*, Future Generation Computer Systems, 35:102–113, 2014.
[79] M. Johnson, *Data hemorrhages in the health-care sector*, Financial Cryptography and Data Security, pp. 71–89, 2009.
[80] *Guidelines on security and privacy in public cloud computing. National Institute of Standards and Technology (NIST)*, U.S. Department of Commerce. Special Publication, 800-144, http://csrc.nist.gov/publications/nistpubs/800- 144/SP800-144.pdf.
[81] S. Kamara and K. Lauter, *Cryptographic cloud storage*, Financial Cryptography and Data Security, 2010, pp. 136-149.
[82] A. Abbas and S. U. Khan, *A Review on the State-of-the-Art Privacy Preserving Approaches in the e-Health Clouds*, IEEE Journal of Biomedical and Health Informatics, 18(4):1431–1441, 2014.
[83] N. Sultan, *Making use of cloud computing for healthcare provision: Opportunities and challenges*, International Journal of Information Management, 34:177–184, 2014.
[84] J. Knaus, S. Hieke, H. Binder and G. Schwarzer, *Costs of cloud computing for a biometry department: a case study*, Methods Inf Med, 52(1):7279, 2013.
[85] D. D'Agostino, A. Clematis, A. Quarati, D. Cesini, F. Chiappori, L. Milanesi and I. Merelli, *Cloud Infrastructures for In Silico Drug Discovery: Economic and Practical Aspects*, Hindawi Publishing Corporation BioMed Research International, 2013, http://dx.doi.org/10.1155/2013/138012.

Edited by: Jesus Carretero
Received: September 16, 2014
Accepted: January 21, 2015

DOI 10.12694/scpe.v16i1.1058
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 19–35. http://www.scpe.org

SELECTED APPROACHES AND FRAMEWORKS TO CARRY OUT GENOMIC DATA PROVISION AND ANALYSIS ON THE CLOUD

PHILIP C. CHURCH AND ANDRZEJ M. GOSCINSKI*

Abstract. While High Performance Computing clouds allow researchers to process large amounts of genomic data, complex resource and software configuration tasks must be carried out beforehand. The current trend exposes applications and data as services, simplifying access to clouds. This paper examines commonly used cloud-based genomic analysis services, introduces the approach of exposing data as services and proposes two new solutions (HPCaaS and Uncinus) which aim to automate service development, deployment process and data provision. By comparing and contrasting these solutions, we identify key mechanisms of service creation, execution and data access required to support non-computing specialists employing clouds.

Key words: Bio-informatics, Software as a Service, HPC

AMS subject classifications. 68M14, 92C37

1. Introduction. High Performance Computing (HPC) has enabled new discoveries in disciplines such as biology and medicine by providing computer facilities to perform complex algorithms and process big data within reasonable time frames. However, HPC requires powerful and expensive computational hardware, data storage, advanced middleware, and sophisticated discipline oriented applications. Due to their high initial purchase and maintenance costs, HPC resources are only affordable by rich institutions. Furthermore, these resources are shared by many researchers, which leads to long waiting times for application execution. Thus, many researchers cannot access HPC infrastructures when needed; they often scale down their applications to reduce waiting times.

In response to these problems, public cloud vendors such as Amazons Elastic Compute Cloud (EC2) [1] have started to provide solutions specifically designed for running HPC applications. These clouds provide the ability to scale on demand as the users requirements change, accelerating the discovery of new knowledge in various fields of research. Thus, discipline specialists now have access to on-demand, scalable and pay-as-you-go HPC facilities that can provide users faster turnaround times on their experiments.

However, while these clouds alleviate the costs of procuring required IT resources, the cost and time of learning how to prepare a HPC cloud and its applications remain a problem to many users [2]. Thus, if discipline scientists want to use HPC clouds for scientific discovery, they must also become system administrators and computer specialists performing time consuming resource management and software configuration activities [3]. Once the HPC cloud has been set up, additional time must be spent transferring data from local machines to the cloud before analysis can be carried out. This process differs depending on the area of discovery; when working with genomic data, users take advantage of both publicly available and privately collected data. As a result, a significant amount of the scientists time is spent to build an understanding of the HPC cloud, acquiring system management skills and managing data instead of conducting research on these HPC clouds.

Discipline specialists are used to accessing software tools through user friendly discipline oriented interfaces. Therefore, we propose to use the lessons learnt from using software tools and hide from a discipline specialist all the operations that require administrator and computing expert knowledge and skill to exploit clouds. This paper examines currently used solutions in the area of bioinformatics to carry out research in the cloud, as well as our work on building frameworks to simplify development and deployment of sequential and HPC application exposed as services to be executed in a Software as a Service (SaaS) cloud.

In terms of genomic data, the current approach taken in the bioinformatics area is that data is shared through public data servers (also known as biological databases). Three geographically separated databases (American, European and Asian) store genomic data which is mirrored on a daily basis; this is supplemented by smaller public databases which store specialized data (for example cancer data [4]). Research groups will often have a local server in which they maintain copies of the data they are interested in (public and privately

*School of Information Technology, Faculty of Science and Technology, Deakin University, Geelong VIC 3127, Australia (philip.church@research.deakin.edu.au, ang@deakin.edu.au)

collected) as well as analysis tools [5]. Since data is stored not only on private data servers but mainly on public data servers, there is a natural opportunity to expose them as Data as a Service (DaaS).

By comparing and contrasting these solutions, we identify the cloud access procedures that are carried out by discipline specialists, and cloud service components that are commonly provided by developers but hidden from these specialists. We demonstrate that building frameworks that automate these cloud service components opens the way to build SaaS and DaaS clouds that support genomic data analysis and make them easy to use by discipline, non-computing specialists.

In brief, the contributions of this paper are as follows;

- A review of current cloud service approches to analyse genomic data.
- An extention of the Uncinus framework to publish and access genomic databases.
- The identification of key components required by cloud services to support non-computing specialists.

The rest of this paper is as follows. Section 2 describes current approaches to carry out genomic research on the cloud. Sections 3 presents developed by us two cloud service solutions (HPCynergy and Uncinus). Section 4 presents the path towards DaaS in biology and describes an extension to the Uncinus framework to support current genomic data resources. Section 5 compares and contrasts cloud solutions using a range of criteria. Section 6 describes the key components required by cloud services to support non-computing specialists. Finally, section 7 presents the conclusion and future work.

2. Current Approches. The use of HPC clouds to support genomic analysis is of great interest to the bioinformatics community. HPC clouds promise the ability to access HPC resources cheaply and on-demand. While any software can run on IaaS clouds, software exposed as a service can be accessed by a majority of researchers working in the area of bioinformatics and therefore it is inherently more useful.

Popular tools used to carry out genomic research on the cloud include: Cloud BioLinux, Galaxy and the Tuxedo Suite.

2.1. Cloud BioLinux. One of the early attempts to simplify the deployment and execution of bioinformatics software on the cloud was Cloud BioLinux [6]. Cloud BioLinux is a virtual machine (VM) configured for high-performance bioinformatics using cloud platforms. At time of writing [7], over 135 bioinformatics tools have been deployed and configured on the virtual machine. As applications have been taken from a large number of sources, there is no standard interface mechanisms, instead users access and execute installed applications through unique graphical interfaces or command line. However documentation of each installed application is made available through a centralized website.

Cloud BioLinux is compatible with IaaS clouds that implement the Amazon model, such as Amazon EC2 [1], OpenStack [8] and Eucalyptus [9]. Alternatively the Cloud BioLinux VM can be executed on a desktop computer using virtualization software such as Virtualbox [10]. The operation to deploy Cloud BioLinux on Amazon EC2 is shown in Fig. 2.1.

1. First, the user selects and configures the Cloud BioLinux VM. This process requires that the user create or select a key pair (a security credential), which will be used by the user to securely connect to the VM instance after it is running. The user must also create or select a security group, which defines firewall rules for the VM instance. Lastly, they specify the number of machines in which to deploy Cloud BioLinux. Running VMs are shown in the Amazon EC2 management console.

2. Once the VM has been launched, the user can access Cloud BioLinux. Users can access the Cloud BioLinux command line through a SSH client (secure shell) and the security keys generated in step 1. Alternately they can use a virtual desktop client to access Cloud BioLinux graphically. Once connected, users transfer files to Cloud BioLinux from a local or remote server before running applications.

3. Once a user has finished their analysis, they can terminate their Cloud BioLinux virtual machine. For this purpose, in the Management Console, they locate the instance(s) in the list of instances on the Instances page, and confirm to terminate the instance(s).

Currently Cloud BioLinux does not provide HPC support; instead each user runs distinct VMs that encapsulate an operating system, analysis software and data. With this approach, users receive distinct and pre-defined computational resources, since each user initializes their own Cloud BioLinux server. While Hadoop/MapReduce [11] [12] is available on Cloud BioLinux, computation is limited to the single virtual machine. Future work will allow end-users to easily provision Hadoop clusters on any cloud. This will facilitate

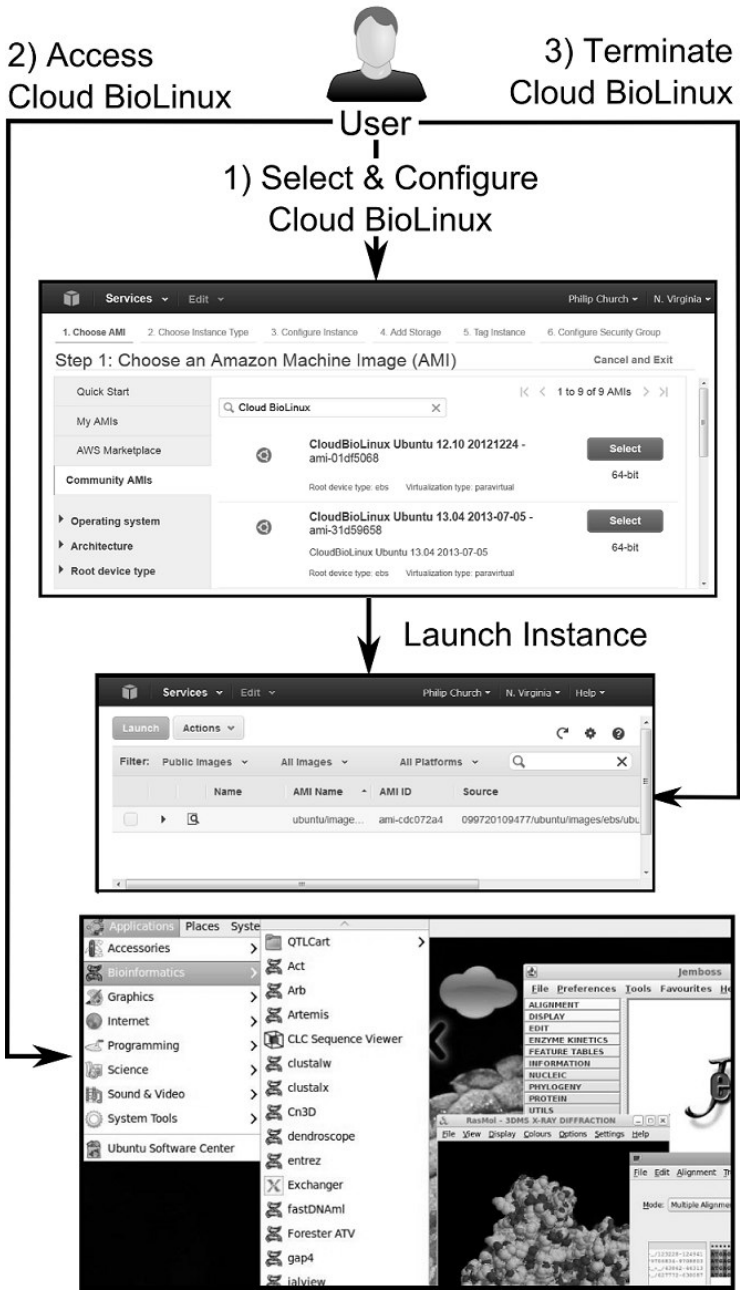


FIG. 2.1. *Cloud BioLinux Deployment Process*

running large-scale bioinformatics data processing pipelines.

2.2. Galaxy. Galaxy [13] is an web-based framework for genomic research. Due to the flexibility and easy-to-use nature of Galaxy, it has become a popular method for carrying out genomic analysis. Galaxy is deployed on a server and accessed through a web browser. Using provided web interfaces, users can share data and applications (as tools) and execute tools as workflows (see Fig. 2.2). Users using Galaxy first upload data from local or public data servers (biological databases) using a range of Galaxy tools. Uploaded data is transfered via FTP and stored on the Galaxy server. Tools which provide analysis capabilities are then used to

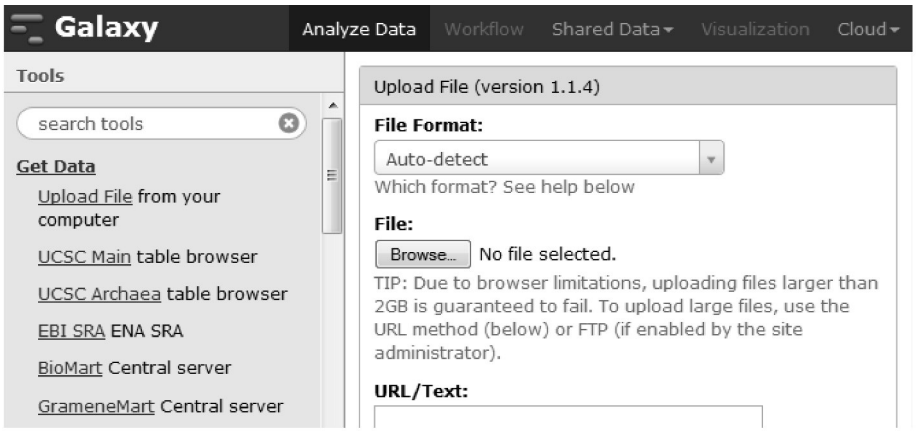


FIG. 2.2. *Galaxy Framework Interface*

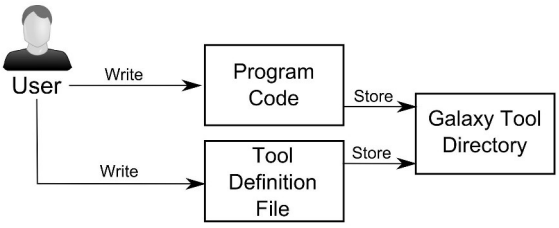


FIG. 2.3. *Galaxy Tool Deployment Process*

process data uploaded to the Galaxy server. Galaxy ensures that this analysis is reproducible by automatically generating metadata for each analysis step. Galaxys metadata includes: input datasets, tools used, parameter values, and output datasets. Galaxy also provides a workflow system which facilitates analysis repeatability, allowing users to carry out analysis based on stored metadata.

Adding a tool to Galaxy requires the user to provide an application to execute and a graphical interface (see Fig. 2.3). Applications can be written in a number of different programming languages as long as the Galaxy server has a valid interpreter installed; common languages include Python and Perl. Regardless of the language utilized, code is placed in the tools directory of the Galaxy installation. Each tool in this directory must be exposed through a graphical interface. To simplify this process, Galaxy users define a GUI for their tool using a Tool Definition File (TDF). A TDF is written in XML and consists of four parts; input controls, output data, test inputs (for validation purposes) and user help.

A key feature of Galaxy is workflow generation, where users can link together installed tools into a single pipeline. Through a pipeline Galaxy can run multiple tools concurrently, reducing the time taken for analysis. To simplify the construction of these pipelines, Galaxy provides an editor (see Fig. 2.4) from which users access a graphical interface for creating and modifying workflows. A user drags and drops tools onto the workflow canvas and configures each step in the workflow. Users can add tags to a workflow, annotating each step of the workflow. Workflows are run in Galaxys analysis workspace; like all tools executed in Galaxy, Galaxy automatically generates history items and provenance information for each tool executed via a workflow.

Galaxy supports the use of the Amazon EC2 cloud through the CloudMan service [14] (see Fig. 2.5). CloudMan automates the process of constructing a cluster using Amazon EC2 resources. Through a web interface, users must first specify their AWS Secret Key ID, and Secret Key. Once these details have been validated, a user can select the size of the cluster required and the location of a persistent data volume. CloudMan utilizes a modified Cloud BioLinux image which contains Sun Grid Engine (SGE) [15]. This image is automatically deployed on the Amazon EC2 cloud as both the head node and workers. Data stores are then linked to the

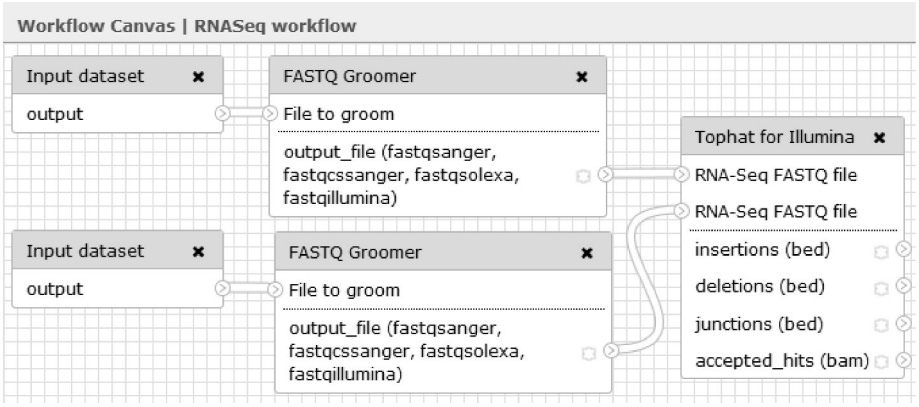


FIG. 2.4. *Galaxy Workflow Deployment Interface*

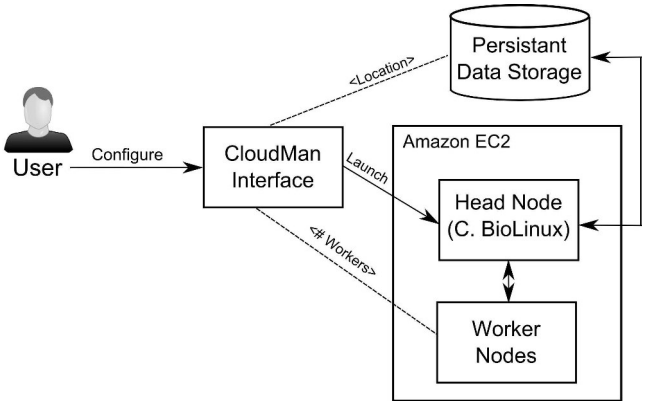


FIG. 2.5. *Galaxy Cloud Deployment Process*

constructed cluster. Once deployed, users can access Cloud BioLinux tools through command line and Galaxy through framework interfaces (via a web browser). By providing user with standard Galaxy interfaces, the cloud is made transparent to the user, in other words Galaxy running on the cloud is indistinguishable from Galaxy running on a local server.

2.3. Tuxedo Suite. The Tuxedo suite brings together commonly used bioinformatics tools for analyzing genomic data generated from high throughput sequencing machines. Some of these tools provide support for HPC platforms in order to reduce the time taken to process genomic data. Two HPC enabled tools in this suite, Crossbow and Myrna, have been made available as cloud services.

Crossbow [16] is a scalable application for genome sequencing aimed at performing alignment between the small fragments produced by the current generation of high-throughput sequencing machines. Crossbow has been integrated into Amazon EC2 using a map and reduce strategy [12]. A graphical interface (see Fig. 2.6) allows users to deploy Crossbow on the Amazon EC2 cloud. This interface requires user provide their AWS Secret Key ID and Secret Key to access their Amazon Account, input any optional arguments accepted by Crossbow and select the number and type of cloud instances they wish to use.

The operation of Crossbow is shown in Fig. 2.7, where a virtual cluster is first created on the Amazon EC2 cloud. Sequence fragments are then uploaded from the users desktop to Amazon storage (S3). Next crossbow code is uploaded to the master/head node, compiled and run. Output data is compressed and sent back to the user.

Myrna [17] is cloud-scale software developed for the purpose of analysing RNA-seq data. Using this software, it is possible to identify the number and type of genes present in a biological sample. To accomplish this, Myrna

FIG. 2.6. *Crossbow Service Interface*

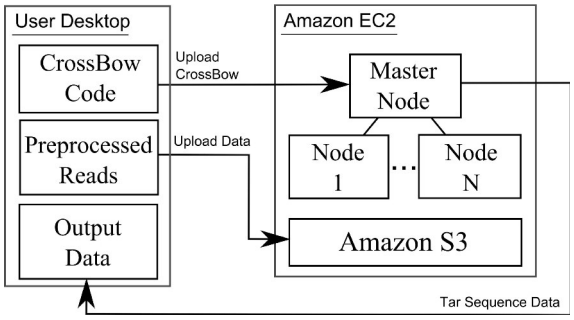


FIG. 2.7. *Crossbow Cloud Workflow*

aligns sequence data, normalizes and carries out statistical modeling in a single computational pipeline. Like Crossbow, Myrna also runs on Amazon EC2 using a map reduce strategy. During execution, each step in the Myrna pipeline is mapped and reduced. A map stage takes a stream of input data, analyses and returns results in the form of a stream. A sort/huffle phase is carried out that sorts data according to data similarity. Lastly, the reduce stage performs computation on data. Myrna is accessed as a service through a web interface in the same manner as Crossbow. Through this interface, users provide their Amazon ID and Secret Key and configure software settings. Upon execution of Myrna as a service, a virtual cluster on Amazon EC2 is created before deploying and executing Myrna.

3. Proposed Approches For Bioinformatic Services. Popular approaches to carry out bioinformatics on the clouds utilize a combination of packaged virtual machines and graphical interfaces to expose applications

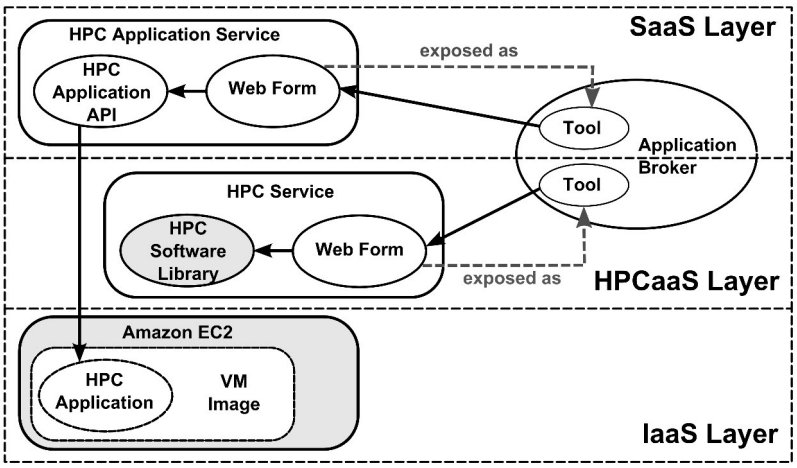


FIG. 3.1. *Implementation overview of HPC cloud framework*

as services. In this way, tools such as Galaxy and Tuxedo Suite (and to a lesser extent Cloud BioLinux) allows users to take advantage of scalability and clouds without understanding cloud architecture. While these solutions are simpler to use, they are difficult to develop and are not flexible (users limited to the services made available). In response we present two approaches for exposing bioinformatics applications as services, HPCaaS and Uncinus. HPCaaS brings targeted cloud computing to Galaxy, while Uncinus supports the development and deployment of services through a cloud middleware.

Our frameworks offer these features through HPC-based publicly accessible virtual machine (IaaS) abstraction, automating the process of exposing HPC biology and discipline applications as services, making these services visible through a broker, and evocable through user friendly discipline oriented interfaces.

3.1. HPCaaS: Galaxy Plugin. We have implemented a Galaxy plugin that allows users to deploy software packages on the EC2 cloud. This plugin was developed by integrating three components: (i) the EC2 service (public IaaS cloud) for providing HPC infrastructure, (ii) a HPC service software library for accessing high level HPC resources obtained from a IaaS cloud, and (iii) the Galaxy software application used as a web-based platform for exposing and accessing HPC application services.

The overview of the prototype design, demonstrating the relationships among the Amazon EC2 service, the HPC software library, and the Galaxy software application is shown in Fig. 3.1. Also shown in Fig. 3.1 is our view of the cloud service stack and where different cloud services would be found. At the bottom (IaaS layer), Amazon EC2 provides cloud infrastructure services. Supported HPC applications are installed in virtual machines and their images were saved and stored in EC2.

In the middle (HPCaaS layer), a HPC software library [18] was used to expose and access Amazon EC2 services in order to provide users a higher level of HPC services such as constructing and managing computer clusters. By using a specific feature of the Galaxy software application, a web-form of the HPC service was generated to be used as a simple but effective interface for users to access the HPC service. Finally, such HPC service was exposed as a tool in a Galaxy server.

On the top (SaaS layer), a HPC application service for a supported HPC application was developed as follows. First, an API of the HPC application was constructed. This API acts as a program stub for its corresponding HPC application, which had been installed in a virtual machine and had been stored in the Amazon EC2 service. Second, a web-form of the HPC application service was generated using the Galaxy application software. Finally, such HPC application service was exposed as a tool in a Galaxy server. It should be noted that each HPC application service would access the HPC services in the HPCaaS layer and the HPC application installed and stored in a VM image at the bottom IaaS layer through its web-form.

Using the HPCaaS Galaxy plugin, users can expose existing virtual machines stored on the Amazon cloud. For each VM, HPCaaS's Tool Definition File (TDF) needs to be modified. In this process, the location of the

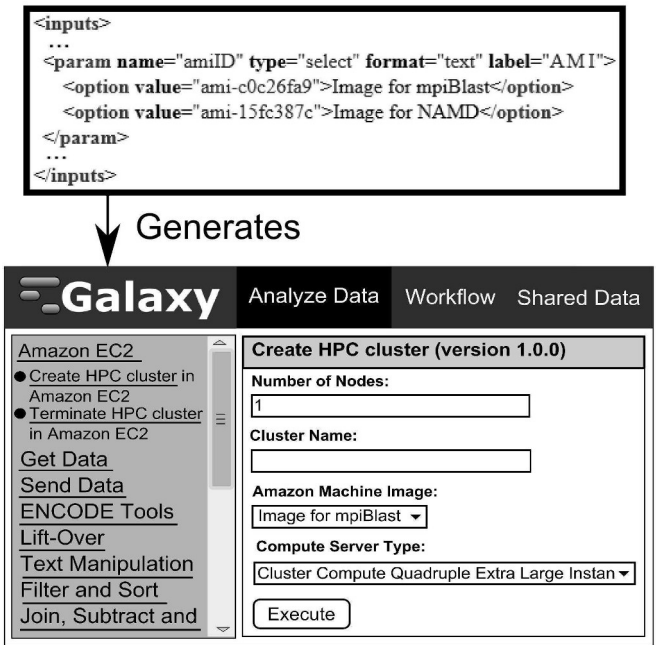


FIG. 3.2. *HPCaaS Galaxy Deployment Interface*

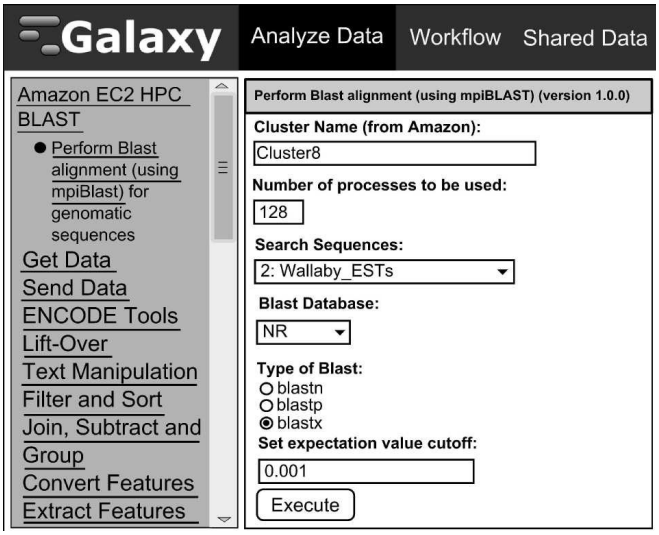


FIG. 3.3. *HPCaaS mpiBLAST service Interface*

Amazon Machine Image (defined as an ID given by the cloud provider) is added to the dropdown menu of the graphical interface (see Fig. 3.2). Lastly, an interface for the VM is generated; this is performed by defining a TDF in the same manner as other Galaxy tools.

The steps taken to utilize this interface is similar to that of the Tuxedo suite (see section 2.3), in that users can create a virtual cluster in the cloud by specifying the required number and type of cloud resources. HPCaaS will automatically construct a virtual cluster with the specified resources, each node consisting of a copy of the virtual machine image. However, HPCaaS is more flexible then the Tuxedo Suite as it allows multiple VM to be exposed through the same interface. In this way, the time taken to build HPC cloud services is reduced.

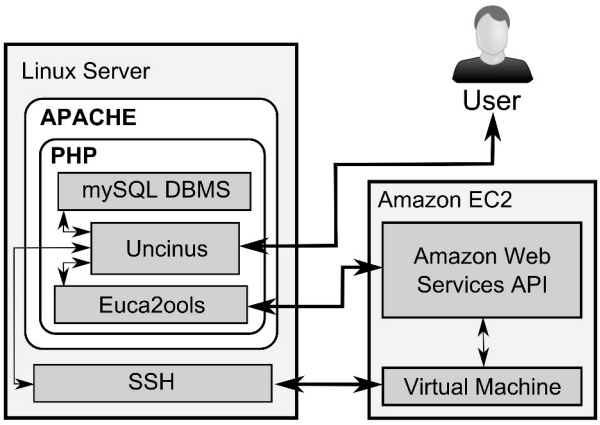


FIG. 3.4. *Uncinus Server Overview*

Once the virtual machine has been deployed on cloud resources, it is made available as a Galaxy tool. An interface for mpiBLAST [19], a commonly used distributed sequence alignment tool, is shown in Fig. 3.3. It consists of six input controls;

- Cluster Name: the name given to the cloud cluster deployed using HPCaaS.
- Number of processes: The number of instances of mpiBLAST that should be run. This also effects the number of fragments that the mpiBLAST database is divided into.
- Search Sequences: the file containing the sequence fragments in which to align.
- The database type: the database in which to carry out BLAST.
- The type of blast: the version of BLAST to run. This option depends on the type of data being analyzed
- The expectation cutoff: a text control which allows a user to choose a value in which to filter results.

The mpiBLAST interface is defined such that it mirrors similar tools such as the Tuxedo Suite and those provided through Galaxy. In this way, a user can access services in a uniform manner regardless of where the software package is being run.

3.2. Uncinus. Uncinus is an environment which takes advantage of existing computer and software resources. Users are provided with a web based interface in which they can share and access a range of cloud, resource and software services. Through use of a broker, Uncinus supports a modular approach to deploying applications on HPC clusters and IaaS clouds, users constucting services (consisting of software and hardware) on demand [20]. These on-demand services are made possible through automated application deployment and job submission provided by the Uncinus middleware.

Deployment of Uncinus on a Linux server is shown in Fig. 3.4. Both end-users and developers access Uncinus through a series of web pages (HTML/PHP) hosted by an APACHE server. Through these web pages, users can access services to publish applications and request compute resources. To provide these services, Uncinus interacts with a mySQL database (to store user credentials and service deployment information) and Euca2ools [9] (to interact with Amazon like clouds). Through euca2ools, users can access the Amazon Web Service API and thereby request cloud resources. Once cloud resources are deployed SSH is used to interact directly with virtual machines.

Users accessing Uncinus for the first time must create a user account. Through this account creation process, a user provides a unique username, password and cloud provider account details. For security reasons, cloud provider account details must be provided in the form of access and secret keys. Once logged in, users can access the components that make up Uncinus (mySQL DBMS, cloud middleware and Euca2ools) through graphical interfaces, shown in Fig. 3.5. The mySQL database is used to store end-user credentials and information about deployed applications/resources. Cloud Deployment Middleware sits above the IaaS cloud and provides a number of services including secure data transfer, cloud resource allocation, HPC environment setup, job submission systems and automated application deployment. Communication between the middleware services

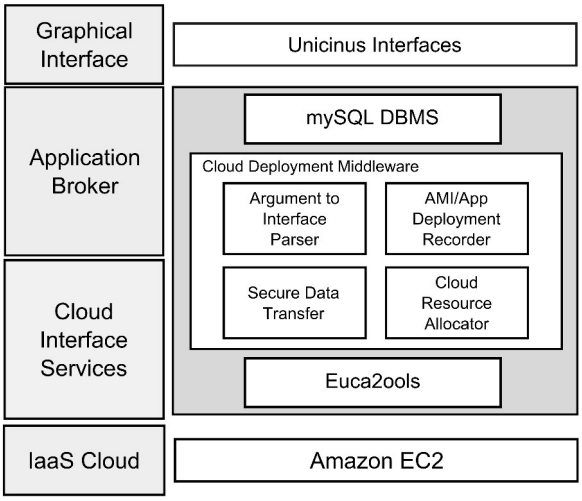


FIG. 3.5. *Uncinus Software Overview*

and the cloud is performed through Euca2ools.

Users access Uncinus through a series of web interfaces that expose the Cloud Deployment Middleware. Unlike other commonly used cloud solutions (described in section 2) which require construction of virtual machines, Uncinus creates services at the application level allowing users with limited cloud experience to develop software services. During the process of exposing an application, users specify a variety of attributes [21] including the service name, software files, installation scripts, software I/O and hardware requirements.

Fig. 3.6 presents the attributes required to deploy mpiBLAST as a service. Publication of this service begins by assigning the service a name (Application Name). Next, a compressed file containing the mpiBLAST source code was uploaded to the broker (Files). An installation script (Install Script) was provided to specify that the openMPI service be loaded as a prerequisite before compiling mpiBLAST. The services interface (Arguments) is specified by defining the type of data required. For this service, a control to upload sequence data and three text controls that determine the type of database and BLAST to be run were specified. The commands to execute mpiBLAST were provided (Running script) in the form of a shell script that formats the selected database and executes mpiBLAST over the selected number of processes. The output of the mpiBLAST service (Result) is a file called blast_results.txt which is to be downloadable by the user. Lastly, requirements for running mpiBLAST are provided; Linux (Operating System), a cluster with 8 nodes each with 8 core running at 2.66 GHz (CPU) and 2 GB of RAM per node (RAM).

The published mpiBLAST service was deployed using the interface shown in Fig. 3.7. Users begin by selecting from available computational resources (cloud and non-cloud) and from a list of published services. When this job is submitted, Uncinus carries out automated resource selection (using published attributes such as CPU and RAM) to identify how to deploy services on the selected resources. This resource selection process calculates a distance metric based on the service requirements and resource specifications.

Once the mpiBLAST service was deployed, a web interface is automatically generated using the published attributes. The mpiBLAST service interface presented to the user is shown in Fig. 3.8. Through this interface, users can: specify mpiBLAST argument, upload sequence data, execute mpiBLAST, download results, and terminate the job (freeing all cloud resources). Also provided through this interface is a tool for transferring files to and from the cloud; uploaded files are mirrored across each node in the cluster.

Through the current implementation of Uncinus, users can take advantage of published software and cloud resources, accessing resources without the need to carry out complex configuration tasks such as setting up a virtual cluster. However in the example above, users are still required to upload sequence data from their local machine. By exposing data as a service, Uncinus can be extended to allow software services to take advantage

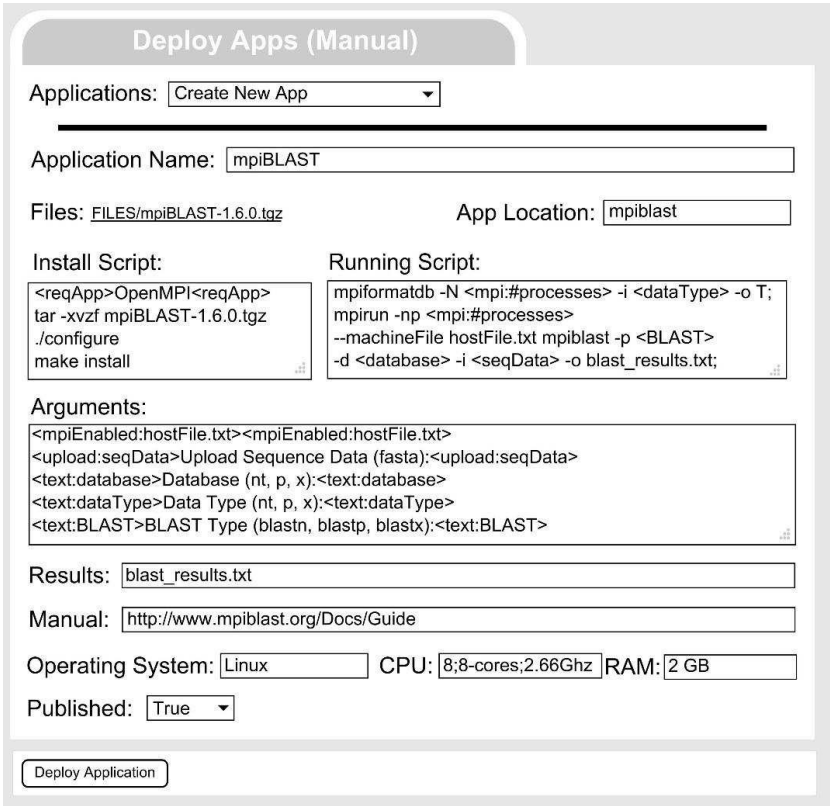


FIG. 3.6. *Uncinus mpiBLAST Publication*

of publically available data.

4. Moving Toward Data as a Service. Data as a Service is an area not highly explored in the area of genomics. The only commercial example is Amazon which provides two DaaS services: S3 which provides users with extendable storage and the Elastic Block Storage (EBS) which provides users the ability to create virtual hard-drives. Through these two services, Amazon provides access to genomic data from the 1000 genome project. An academic attempt at DaaS is being led by the University of Chicago [26], who have recently announced the creation of a petabtye scale open science data cloud which will allow researchers access to manage, analyze and share their data.

4.1. Toward DaaS in Biology. Currently the approach taken in the bioinformatics area is that data is shared through public data servers (also known as biological databases). Three primary databases (NCBI [22], EBI [24] and NGI [23]) store DNA sequence data and are mirrored on a daily basis, this is supplemented by many smaller databases which target a subset of data (for example specific diseases [4] and collection platforms [25]). Research groups working the area will often have a local server in which they maintain copies of the data they are interested in (public and privately collected) as well as analysis tools (as seen in solutions such as Galaxy) [5]. Data stored on these local servers exist in the form of human readable ASCII files which are often not indexed or easily searchable.

While these methodologies are satisfactory for small amounts of data, when analysing the amount of genomic data generated by current sequencing machines, there arises a number of issues.

- Database Discovery: Smaller specialized databases can be difficult to find, currently discovery of new databases is reliant on paper publication and word-of-mouth.
- Lack of Data Management: Research groups maintain local data servers which store data. Data is often stored in a human readable ASCII format which takes up more disk space compared to binary formats.

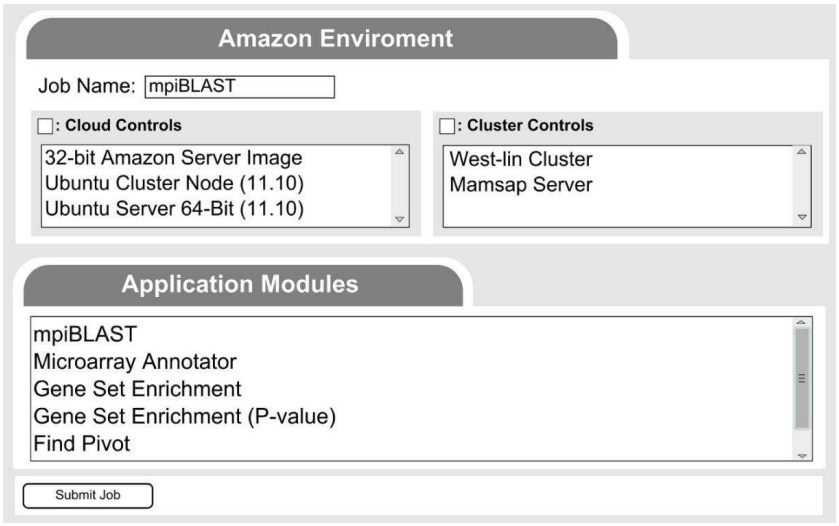


FIG. 3.7. *Uncinus Service Development Interface*

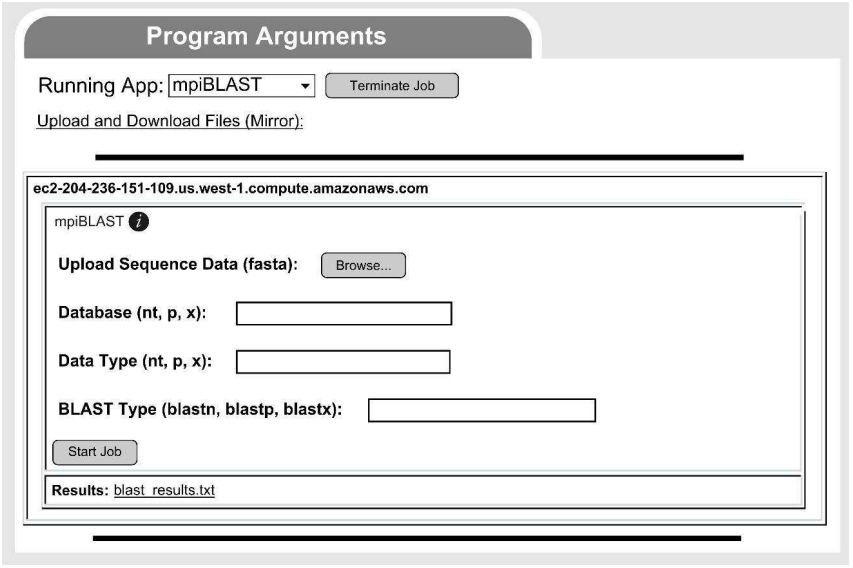


FIG. 3.8. *Uncinus mpiBLAST Service Interface*

Data is difficult to find as there are limited search/discovery mechanisms implemented on these local data servers.

- Database Reliability: Local data servers often have minimal/no backup or redundancy (due to the cost of storage). These systems are inflexible, which jeopardizes availability and reliability.
- Data Transfer: In context of utilizing cloud resources, most cloud tools require that data is moved from public databases to a user's local machine and then back to the cloud. This method is wasteful as data is transferred to a local machine where it is not needed.

Since data is stored not only on private data servers but mainly on public data servers, there is a natural opportunity to expose them as Data as a Service. The data could be stored on any of the server without the user involvement; the data attributes could be stored on a data broker, which becomes a major tool used by other user who wish to search for specific data. The data copies, for reliability and availability, could be stored

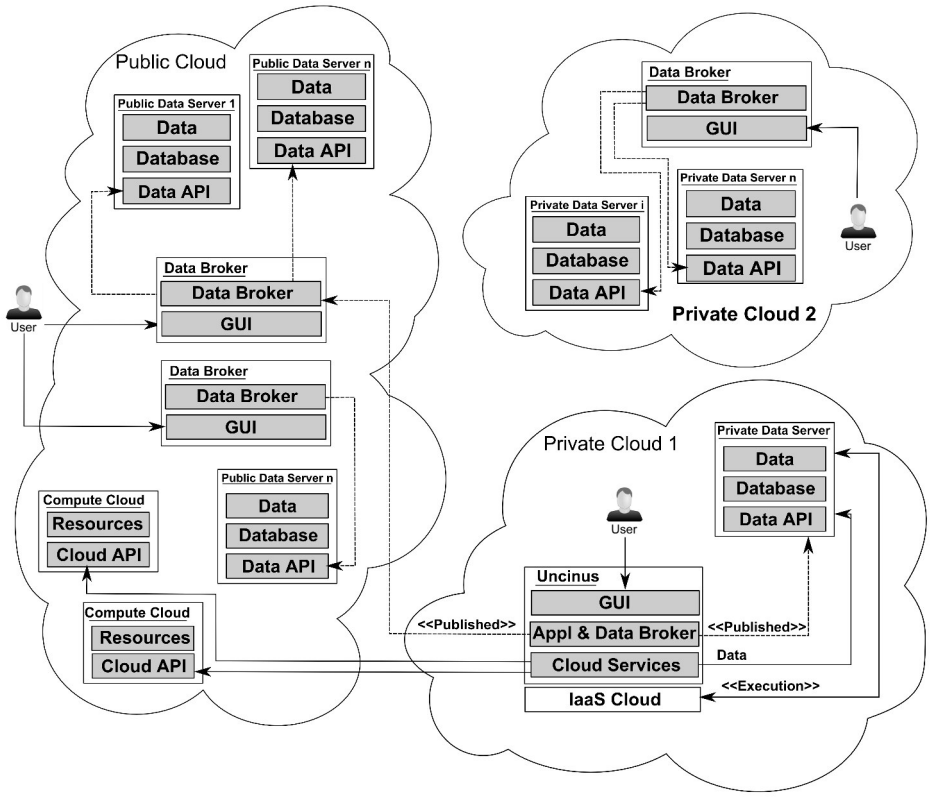


FIG. 4.1. *DaaS Overview*

on a number of servers; their location could be hidden from the user. The selection of a copy of the requested data could be carried out by the broker taking into consideration user location and money limits. A user oriented interface could offer an easy access to the data broker that supports data selection for the application of interest. Data are managed by cloud service providers; that relieves discipline specialists from many activities of computing nature. As shown in Fig. 4.1, the user can carry out discovery on their local resources, within the research group/laboratory private cloud, and/or a public or hybrid cloud. The last option lowers the costs of resources, supports computations on demand, and allows cooperation to be exploited widely.

4.2. DaaS in Uncinus. In this proposed solution we extend the Uncinus application broker to publish data severs. Users specify the location of the server as well as methods to download files (GET), upload files (PUT) and get directory listings (LIST). In this way Uncinus can support a range of public biological databases by abstracting their unique API. Public data brokers are also deployed which users can access to discover databases and share their own data. The data broker will also be able to upload data to published biological databases. By linking the Uncinus broker to a public broker, Uncinus users will be able to access private data servers and public biological databases. Through the proposed solution we are convinced that in many cases the performance of discovery in biology and medicine could be dramatically improved if data sharing (as described above) is applied widely.

5. Comparison of Solutions. Our paper investigates a number of popular bioinformatics SaaS cloud solutions. These solutions differ in how they are developed, deployed and used. We compare and contrast these solutions through the use of six criteria; 1) usability/access, 2) development, 3) deployment, 4) service discovery 5) cloud utilization and 6) data intergration. Each platform and their adherence to the five criteria are shown in Table 5.1.

The first criterion focuses on how the presented solutions are accessed and used. In general, all of the

TABLE 5.1
Popular solutions compared to defined criteria.

	Access	Develop.	Deploy.	Discovery	Utilization	Data
BioLinux						Amazon
Tuxedo						Amazon
Galaxy						Tools
HPCaaS						Tools
Uncinus						Data Broker

bioinformatics cloud software described in this paper allows users to take advantage of scalability and clouds without understanding cloud architecture. Through graphical interfaces, users can access HPC clouds without needing to carry out the time consuming task of setting up middleware to create a virtual cluster in the cloud. Cloud BioLinux exposes applications through an OS interface, while Tuxedo Suite, Galaxy, HPCaaS and Uncinus provide web based controls.

The second criterion focuses on comparing the effort taken to develop services. In general, development of bioinformatics cloud services is difficult and time consuming, requiring skills in genomics, HPC computing, programming and cloud computing. This is seen in Cloud BioLinux and HPCaaS, which depend on the creation and packing of a VM through the use of command line tools. In the case of HPCaaS, users must also modify the HPCaaS tool to recognize the VM. The Tuxedo Suite required the development of graphical interfaces and software API to communicate with the cloud application. Recognizing the difficulty of cloud service development, we have seen a gradual shift from individual services to frameworks such as Galaxy and Uncinus, which attempt to simplify service development. Galaxy makes the development process easier by simplifying construction of graphical interfaces. Uncinus takes this one step further and allows users to define attributes, which are used to generate graphical interfaces and better utilize resources.

The third criterion examines how solutions are deployed on the cloud; currently this ranges from difficult to deploy to no deployment required. Traditionally, HPC cloud solutions have been offered at an IaaS level, which is targeted at computing experts. Early bioinformatics solutions which made use of these IaaS cloud platforms (such as Cloud BioLinux) are difficult to deploy. Other solutions attempt to simplify deployment through automation of this deployment process. Galaxy, HPCaaS and Uncinus as frameworks, provide functions to deploy services on clouds. The Tuxedo Suite automates all aspects of deployment; users require only their Amazon ID and Secret Key.

The fourth criterion examines how these services are discovered by users. Individual services such as Cloud BioLinux and the Tuxedo Suite do not have any inbuilt form of discovery. The Tuxedo Suite consists of a small number of tools and therefore has no need for discovery mechanisms; as these suites get larger, the lack of service discovery can limit their use. Cloud BioLinux contains hundreds of different applications; however service discovery is not supported. Each application provided by Cloud BioLinux has different access methods. Frameworks such as Galaxy, HPCaaS, and Uncinus, which also make available large amounts of tools, use service repositories. By centralizing and indexing available services, users can find software easier.

The fifth criterion examines how well these solutions take advantage of the cloud during execution. The Tuxedo Suite is built directly for the cloud; this solution takes advantage of cloud scalability to increase performance. CloudBioLinux and Galaxy (via CloudMan) are based around duplicating the environment, and unless parallelism is built into the hosted tools, can only ensure a consistent level of performance (each user having their own instance of Galaxy or Cloud BioLinux). Our approaches aimed to expose individual distributed applications like the Tuxedo Suite while taking advantage of the framework features to simplify service development. The first approach, HPCaaS, extended Galaxy based analysis to the cloud through a HPC software library that exposed Amazon EC2 services. Using HPCaaS, users could access VM stored on the EC2 cloud and execute distributed applications through Galaxy interfaces. Uncinus focused on service level development and deployment on clouds. Instead of developing a VM, users could publish installation steps which would be automatically carried out.

The sixth criterion examines how well these solutions are integrated with data. While all package can be

integrated with data, some are more complex to set-up then others. Packages that use Amazon EC2 can use genomic data (such as data from the 1000 genome project) stored using Amazon Cloud Storage (S3) and the Elastic Block Store (EBS). Some variants of CloudBioLinux will automatically mount the virtual EBS drives, and Tuxedo Suite (while not providing native access to this data) could be configured to use the genomic data stored on Amazon S3. Galaxy provides tools which can be used to discover and download genomic data which is stored on the Galaxy Server. HPCaaS (as a Galaxy tool) can also utilize the data downloaded in this way. The method proposed by Uncinus uses a data broker, allowing for discovery of data across multiple up-to-date genomic databases, as a result Uncinus is able to access a wider range of data when compared to the Amazon based solutions. Furthermore Uncinus stores only the link and description of the data, instead of the actual data as seen in the Galaxy approach, this minimizes the need for local storage and prevents unnecessary transfer between the genomic database, the Uncinus server and the computational resources.

In conclusion, the cloud solutions presented in this paper fall under two categories, stand-alone services and frameworks. Individual solutions include Cloud BioLinux and the Tuxedo Suite, while frameworks include Galaxy, HPCaaS, and Uncinus. The benefits of utilizing individual services are that they are often built for and can fully utilize the cloud, in addition they often have minimal to no deployment requirements. On the other hand, individual services are difficult to develop (requiring specialized GUI and API) and are difficult to discover. For a user looking to carry out research on the cloud, picking the right service from hundreds is difficult. Popular frameworks solve this problem with large repositories of services managed by brokers. This model can identify and therefore reduce the overlap in the types of cloud software services being made available. Additionally, frameworks simplify development by automating key steps in the service development process such as for example construction of graphical interfaces. While these features clearly benefit discipline researchers, popular frameworks (Galaxy) do not take full advantage of distributed and cloud computing platforms. Our solutions attempted to fill this gap to some success, proposing frameworks which provided access to clouds in a manner similar to individual services.

6. Discussion. While originally developed for business, cloud and service computing can be used for research. Cloud resources on-demand can enhance local resources, reducing the turn-around time of analysis and/or allowing for bigger problems to be solved. However clouds do not fit all applications; depending on the research being carried out, HPC hardware may be required. Fortunately, a range of cloud providers now offer HPC cloud solutions. Services built on top of these HPC cloud solutions could offer researchers access to HPC on demand without the need to understand and carry out complex deployment methods. It is by combining service and cloud computing technologies that solutions have been devised to simplify genomics analysis. These solutions fall into two categories; stand-alone services which are built from the ground up to utilize the cloud, and cloud frameworks which simplify the development of cloud services.

Despite the advantages of frameworks such as Galaxy, the bioinformatics cloud software service area is still dominated by individual, isolated applications (as seen in the Tuxedo Suite and Cloud BioLinux). Development of these cloud services are time consuming and often repetitive. A review of these approaches shows that the most utilized operations during IaaS application deployment fall into four categories: cloud security, resource allocation, application deployment, and data transfer (see Table 6.1). When deploying HPC applications, operations in these categories must be repeated for each computational node utilized. For this reason, steps such as installation and data transfer (particular in large data scenarios) can greatly increase the time spent to utilize cloud resources. In addition, a user wishing to provide services through the cloud (a service provider) is required to provide a further layer of abstraction on top of the IaaS cloud, exposing the deployed cloud application through a graphical interface.

TABLE 6.1
Most Utilized Operations during IaaS deployment.

Cloud Security	Creating security keys and groups
Application Deployment	Installing software applications
Data Transfer	Transfer files to and from the cloud
Resource Allocation	Launch and Terminate Instances

TABLE 6.2
Components required for SaaS development.

Software API	Allows execution of cloud applications
Graphical Interface	Allows communication with the API
Service Publication (Optional)	Stores and allows users to find service

Once software has been deployed on the cloud, it is possible to expose it as a service. Software services can be discovered and utilized by a larger number of discipline specialists but require development of a software API and graphical interface which are then (optionally) published (see Table 6.2). A software API defines the instructions used to run the application. A graphical interface allows users to execute and pass application arguments to the API. These services can be published; cloud application, API and interfaces stored and indexed through a broker. Service publication supports users, allowing then to discover services that fulfill their goals.

We propose that future cloud solutions should look to automate as much of these operations as possible in order to make service development as simple as possible. When building clouds aimed at carrying out research, cloud specific operations should be logically separated from data processing and analysis.

7. Conclusion. In conclusion, this paper presents a survey of approaches currently used in genomic analysis. Trends in the area show a move from IaaS clouds to SaaS frameworks. Early cloud solutions (Cloud BioLinux) packaged software tools into a single virtual machine. Due to the difficulty of maintaining this type of monolithic solution, developers moved to individual cloud services (Tuxedo Suite). More recently, frameworks such as Galaxy, which provide usability similar to cloud services while simplifying development, have become popular. However Galaxy's support for clouds is still primitive, reliant on mirroring the current state of the Galaxy server.

As a response we developed and presented two SaaS cloud frameworks that draw upon concepts from SaaS clouds, Galaxy and bioinformatics cloud software, in order to solve known cloud usability issues (the difficultly of cloud development and flexibility of individual tools). These frameworks addressed cloud usability by providing researchers the tools to access the cloud and run distributed applications. Users with a background in programming, system administration and cloud computing can develop HPC software and publish VM resources. Users with knowledge of the software applications (but limited programming skills) can define required attributes through the service publication interfaces and become SaaS providers. Lastly, users with a background in biology and minimal computing knowledge can access the applications and resources (published by the broker) that are required to perform analysis. For such users, clouds are made completely transparent and HPC applications are exposed as services. In this way discipline specialists with different levels of computing expertise can take advantage of cloud resources. By incorperating the ability to publish data resources, we further simplfy the genomic analsis process, users will be able to access up-to-date genomic data while removing the need for locally maintained genomic databases.

Through this successful integration of cloud and service computing, the analysis, interpretation and computation of genomic mammalian data was made easier, to be carried out by non-computing discipline specialists, and cheaper. By comparing current approaches and our cloud frameworks we identify the cloud access procedures that are commonly carried out by users, and cloud service components that are commonly provided by developers. Operations carried out by users during IaaS application deployment fall into four groups: cloud security, resource allocation, application deployment, and data transfer. They require a computing expert to be carried out; discipline specialists should be relieved from learning clouds and acquiring skills to carry out these operations. Instead they should concentrate on their discipline problems using clouds, in particular SaaS clouds. The major components provided by cloud services fall into three categories: Software API, Graphical Interface and Service Publication. The development of these components should be carried out to satisfy discipline specialists requirements to allow them to directly benefit from SaaS clouds.

Future work should aim to automate these procedures in such a way that clouds are made transparent to the user. Research into graphical methods to create virtual machines could widen the use of IaaS clouds, while natural language could be applied to service development to automatically generate graphical interfaces from software manuals and source code. A number of technical improvments could also be made: to take advantage

of exisiting software repositories such as Galaxy, there is a need to standardize interface mechanisms, there is also a need to enable seamless application scalability to improve cloud performance.

REFERENCES

[1] Amazon, *Amazon Elastic Compute Cloud: Getting Started Guide*, LLC AWS Amazon, 25 (2010).
[2] Goscinski A, Brock M, Church P, *High Performance Computing Clouds.*, CRC, Taylor & Francis group, June 2011.
[3] Yelick K, Coghlan S, Draney B, Canon RS, *The Magellan Report on Cloud Computing for Science*, U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), December 2011.
[4] D. Fenstermacher, C. Street, T. McSherry, et. al, *The Cancer Biomedical Informatics Grid* , in Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the, 2005, pp. 743-746.
[5] L. Dai, X. Gao, Y. Guo, J. Xiao, and Z. Zhang, *Bioinformatics clouds for big data manipulation.*, Biology Direct, vol. 7, p. 43, 2012.
[6] Cloud Research Group, *Cloud BioLinux: pre-configured and on-demand high performance computing for the genomics community*, 2010.
[7] Krampis K, Booth T, Chapman B, Tiwari B, Bicak M, Field D, Nelson K, *Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community.*, BMC Bioinformatics 2012, 13(1):42.
[8] OpenStack, *Open source software for building private and public clouds.*, http://www.openstack.org/
[9] Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D, *The Eucalyptus Open-Source Cloud-Computing System*, Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE Computer Society 2009.
[10] Oracle VirtualBox, http://www.virtualbox.org
[11] Dean J, Ghemawat S, *MapReduce: simplified data processing on large clusters*, Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6; San Francisco, CA. 1251264: USENIX Association 2004: 10-10.
[12] Apache Hadoop, http://hadoop.apache.org/
[13] Goecks J, Nekrutenko A, Taylor J, Team TG, *Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences*, Genome Biol 2010, 11(8):R86.
[14] Afgan E, Baker D, Coraor N, Chapman B, Nekrutenko A, Taylor J, *Galaxy CloudMan: delivering cloud compute clusters*, BMC Bioinformatics 2010, 11(Suppl 12):S4.
[15] Gentzsch W, *Sun Grid Engine: Towards Creating a Compute Power Grid*, Proceedings of the 1st International Symposium on Cluster Computing and the Grid. IEEE Computer Society 2001.
[16] Langmead B, Schatz M, Lin J, Pop M, Salzberg S, *Searching for SNPs with cloud computing*, Genome Biol 2009, 10(11):R134.
[17] Langmead B, Hansen K, Leek J, *Cloud-scale RNA-sequencing differential expression analysis with Myrna*, Genome Biol 2010, 11(8):R83.
[18] Wong AKL, Goscinski AM, *A unified framework for the deployment, exposure and access of HPC applications as services in clouds*, Future Generation Computer Systems 2013, 29(6):1333-1344.
[19] Darling A, Carey L, Feng W, *The Design, Implementation, and Evaluation of mpiBLAST.*, ClusterWorld 2003: 2002.
[20] Church P, Wong A, Brock M, Goscinski A, *Toward Exposing and Accessing HPC Applications in a SaaS Cloud.*, Web Services (ICWS), 2012 IEEE 19th International Conference on: 24-29 June 2012 2012. 692-699.
[21] Brock M, Goscinski A, *Attributed Publication and Selection for Web Service-Based Distributed Systems*, Services - I, 2009 World Conference on: 6-10 July 2009 2009. 732-739.
[22] National Center for Bio-informatics, http://www.ncbi.nlm.nih.gov/
[23] National Genomics Institute, http://www.nig.ac.jp/
[24] European Bioinformatics Institute, http://www.ebi.ac.uk/
[25] A. Brazma, H. Parkinson, U. Sarkans, et. al, *ArrayExpress–a public repository for microarray gene expression data at the EBI.*, Nucl. Acids Res., vol. 31, pp. 68-71, January 1, 2003 2003.
[26] Open Science Data Cloud - University of Chicago, https://www.ci.uchicago.edu/research-centers/open-science-data-cloud

Edited by: Jesus Carretero
Received: September 8, 2014
Accepted: January 21, 2015

DOI 10.12694/scpe.v16i1.1059
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 37–51. http://www.scpe.org

ON THE FORMALIZATION OF ZSYNTAX WITH APPLICATIONS IN MOLECULAR BIOLOGY

SOHAIB AHMAD, OSMAN HASAN, UMAIR SIDDIQUE*

Abstract. Recent progress in nanotechnology and optical imaging offers promising features to develop effective drugs to cure chronic diseases, such as cancer and malaria. However, qualitative characterization of biological organisms (such as molecules) is the foremost requirement to identify key drug targets. One of the most widely used approaches in this domain is molecular pathways, which offers a systematic way to represent and analyse complex biological systems. Traditionally, such pathways are analysed using paper-and-pencil based proofs and simulations. However, these methods cannot ascertain accurate analysis, which is a serious drawback given the safety-critical nature of the applications of molecular pathways. To overcome these limitations, we recently proposed to formally reason about molecular pathways within the sound core of a theorem prover. As a first step towards this direction, we formally expressed three logical operators and four inference rules of Zsyntax, which is a deduction language for molecular pathways. In the current paper, we extend this formalization by verifying a couple of behavioural properties of Zsyntax based deduction using the HOL4 theorem prover and developing a biologist friendly graphical user interface. Moreover, we demonstrate the utilization of our work by presenting the formal analysis of cancer related molecular pathway, i.e., TP53 degradation and metabolic pathway known as Glycolysis.

Key words: Molecular biology, logical deduction

AMS subject classifications. 68Q25, 92C40, 03D05

1. Introduction.

Molecular biology is extensively used to construct models of biological processes in the form of networks or pathways, such as protein-protein interaction networks and signalling pathways. The analysis of these biological networks, usually referred to as biological regulatory networks (BRNs) or gene regulatory networks (GRNs) [10], is based on the principles of molecular biology to understand the dynamics of complex living organisms. Moreover, the analysis of molecular pathways plays a vital role in investigating the treatment of various human infectious diseases and future drug design targets. For example, the analysis of BRNs has been recently used to predict treatment decisions for sepsis patients [15].

Traditionally, the molecular biology based analysis is carried out by biologists in the form of wet-lab experiments (e.g. [7, 13]). These experiments, despite being very slow and expensive, do not ensure accurate results due to the inability to accurately characterize the complex biological processes in an experimental setting. Other alternatives for deducing molecular reactions include paper-and-pencil proof methods (e.g. using Boolean modelling [27] or kinetic logic [28]) or computer-based techniques (e.g. [29]) for analysing molecular biology problems. The manual proofs become quite tedious for large systems, where the calculation of unknown parameters takes several hundred proof steps, and are thus prone to human errors. The computer-based methods consist of graph-theoretic techniques [21], Petri nets [11] and model checking [3]. These approaches have shown very promising results in many applications of molecular biology (e.g. [8, 14]). However, these methods are not generic and hence have been used to describe some specific areas of molecular biology only [4]. Moreover, the inherent state-space explosion problem of model checking [20] limits the scope of this success only to systems where the biological entities can acquire a small set of possible levels.

Theorem proving [12] can be considered as the second most widely used formal method. It does not suffer from the state-space explosion problem of model checking and has also been advocated for conducting molecular biology based analysis [30]. The main idea behind theorem proving is to construct a computer based mathematical model of the given system and then verify the properties of interest using deductive reasoning. The foremost requirement for conducting the theorem proving based analysis of any system is to formalize the mathematical or logical foundations required to model and analyse that system in an appropriate logic. There have been several attempts to formalize the foundations of molecular biology. For example, the earliest axiomatization even dates back to 1937 [31] and other efforts related to the formalization of biology are presented in [32, 25]. Recent formalizations, based on K-Calculus [6] and π-Calculus [22, 23, 24], also include some formal

*School of Electrical Engineering and Computer Science (SEECS), NUST, Islamabad, Pakistan ({11mseesahmad, osman.hasan, umair.siddique}@seecs.nust.edu.pk).

reasoning support for biological systems. But the understanding and utilization of these techniques is very cumbersome for a working biologist as highlighted by Fontana in [9].

In order to develop a biologist friendly formal deduction framework for reasoning about molecular reactions, we propose to formalize the Zsyntax [4] language in higher-order logic. Zsyntax is a formal language that supports modelling and logical deductions about any biological process. The main strength of Zsyntax is its biologist-centred nature as its operators and inference rules have been designed in such a way that they are understandable by the biologists. Traditionally, logical deductions about biological processes, expressed in Zsyntax, were done manually based on the paper-and-pencil based approach. This limits the usage of Zsyntax to smaller problems and also makes the deduction process error-prone due to the human involvement. As a first step towards overcoming this limitation, we formalized the logical operators and inference rules of Zsyntax in higher-order logic [2]. In the current paper, we build upon these formal definitions to verify a couple of key behavioural properties of Zsyntax based molecular pathways using the HOL4 theorem prover. The formal verification of these properties raises the confidence level in our definitions of Zsyntax operators and inference rules, which have complex interrelationships. Moreover, these formally verified properties can be used to facilitate the formal reasoning about chemical reactions at the molecular level. In order to illustrate the usefulness and effectiveness of our formalization for analysing real-world problems in molecular biology, we present the formal analysis of a cancer related molecular pathway involving TP53 degradation and a metabolic pathway known as Glycolysis. As a part of this work, we have also developed a user-friendly graphical user interface (GUI) for conducting the Zsyntax based formal analysis of molecular pathways. The distinguishing features of this tool includes its biologist friendliness and dedicated proof tactic that allows the users to perform automated reasoning about molecular pathways.

Our current framework handles static reactions but it can be further extended to study the reaction kinetics [4] due to the flexibility of Zsyntax. The main motivation behind using higher-order-logic theorem proving in our work is to be able to leverage upon the high expressiveness of higher-order logic and thus reason about differential equations and probabilistic properties, which form an integral part of reaction kinetics. However, the scope of the current paper is on the formalization of Zsyntax based deduction calculus for molecular pathways but this formalization can later be extended to support reaction kinetics as well because it is done in a higher-order-logic theorem prover.

The rest of the paper is organized as follows: Section 2 provides an introduction to Zsyntax and the HOL4 theorem prover. The higher-order-logic formalization of Zsyntax operators and inference rules using HOL4 are described in Sect. 3. This is followed by the descriptions of the behavioural properties of Zsyntax along with their formal proof sketches in Sect. 4 and 5, respectively. We provide a brief overview of the GUI developed as a part of this work in Sect. 6. In order to demonstrate the use of our formalization, we present two case studies on the TP53 degradation and glycolytic pathway in Sect. 7. We conclude the paper in Sect. 8 while highlighting some interesting potential applications of our work.

2. Preliminaries.

2.1. Zsyntax. Zsyntax [4] exploits the analogy between biological processes and logical deduction. Some of the key features of Zsyntax are: 1) the ability to express molecular reactions in a mathematical way; 2) heuristic nature, i.e., if the conclusion of a reaction is known, then one can deduce the missing data from the initialization data; 3) computer implementable semantics. Zsyntax consists of the following three operators:
Z-Interaction: The interaction of two molecules is expressed by the Z-Interaction ($*$) operator. In biological reactions, Z-interaction is not associative.
Z-Conjunction: The aggregate of same or different molecules (not necessarily interacting with each other) is formed using the Z-Conjunction (&) operator. Z-Conjunction is fully associative.
Z-Conditional: A path from A to B under the condition C is expressed using the Z-Conditional ($\rightarrow$) operator as: $A \rightarrow B$ if there is a C that allows it.

Zsyntax supports four inference rules, given in Table 2.1, that play a vital role in deducing the outcomes of biological reactions:

Besides the regular formulas that can be derived based on the above mentioned operators and inference rules, Zsyntax also makes use of *Empirically Valid Formulae* (EVF). These EVFs basically represent the non-logical axioms of molecular biology and are assumed to be validated empirically in the lab.

TABLE 2.1
Zsyntax Inference Rules

Inference Rules	**Definition**
Elimination of Z-conditional(→E)	`if C ⊢ (A → B) and (D ⊢ A) then (C&D ⊢ B)`
Introduction of Z-conditional(→I)	`C&A ⊢ B then C ⊢ (A → B)`
Elimination of Z-conjunction(&E)	`C ⊢ (A&B) then (C ⊢ A) and (C ⊢ B)`
Introduction of Z-conjunction(&I)	`(C ⊢ A) and (D ⊢ B) then (C&D) ⊢ (A&B)`

It has been shown that any biological reaction and its final outcome can be derived by using the above mentioned three operators and four inference rules [4]. For example, consider a scenario in which three molecules A, B and C react with each other to yield another molecule Z. This can be represented as a Zsyntax theorem as follows:

$$\texttt{A \& B \& C} \vdash \texttt{Z}$$

The Z-Conjunction operator & is used to represent the given aggregate of molecules and then the inference rules from Table 1 are applied on these molecules along with some EVFs (chemical reactions verified in laboratories) to obtain the final product Z. For the above example, these EVFs could be:

$$\texttt{A * B} \rightarrow \texttt{X} \text{ and } \texttt{X * C} \rightarrow \texttt{Z}$$

meaning that A will react with B to yield X and X in return will react with C to yield the final product Z.

The main contribution of our paper is the formal verification of the Zsyntax based deduction method based on the higher-order-logic formalization of the above-mentioned operators and inference rules using the HOL4 theorem prover. This work will in turn facilitate the derivation of biological reactions within the sound core of HOL4.

2.2. HOL4 Theorem Prover. HOL4 is an interactive theorem prover developed at the University of Cambridge, UK, for conducting proofs in higher-order logic. It utilizes the simple type theory of Church [5] along with Hindley-Milner polymorphism [17] to implement higher-order logic. HOL4 has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

In order to ensure secure theorem proving, the logic in the HOL4 system is represented in the strongly-typed functional programming language ML [19]. An ML abstract data type is used to represent higher-order logic theorems and the only way to interact with the theorem prover is by executing ML procedures that operate on values of these data types. The HOL4 core consists of only 5 basic axioms and 8 primitive inference rules, which are implemented as ML functions. Soundness is assured as every new theorem must be verified by applying these basic axioms and primitive inference rules or any other previously verified theorems/inference rules.

A HOL4 theory is a collection of valid HOL4 types, constants, axioms and theorems, and is usually stored as a file in computers. Users can reload a HOL4 theory in the HOL4 system and utilize the corresponding definitions and theorems right away. Various mathematical concepts have been formalized and saved as HOL4 theories by the HOL4 users. We utilize the HOL4 theories of Booleans, arithmetics and lists extensively in our work. Table 2.2 provides the mathematical interpretations of some HOL4 symbols and functions frequently used in this paper.

3. Formalization of Zsyntax. We modelled the molecules as variables of arbitrary data types (α) in our formalization of Zsyntax [2]. A list of molecules (α *list*) represents the Z-Interaction or a molecular reaction among the elements of the list. The Z-Conjunction operator forms a collection of non-reacting molecules and can now be formalized as a list of list of molecules (α *list list*). This data type allows us to apply the Z-Conjunction operator between individual molecules (a list with a single element) or multiple interacting molecules (a list with multiple elements). The Z-Conditional operator is used to update the status of molecules, i.e., generate a new set of molecules based on the available EVFs (wet-lab verified reactions). Each EVF is modelled in our formalization as a pair (α *list* # α *list list*) where the first element is a list of molecules (α *list*) indicating the reacting molecules and the second element is a list of list of molecules (α *list list*) indicating the resulting set of

TABLE 2.2
HOL4 Symbols and Functions

HOL4 Symbol	Standard Symbol	Meaning
$\wedge$	*and*	Logical *and*
$\vee$	*or*	Logical *or*
$\neg$	*not*	Logical *negation*
::	*cons*	Adds a new element to a list
++	*append*	Joins two lists together
`HD L`	*head*	Head element of list L
`TL L`	*tail*	Tail of list L
`EL n L`	*element*	n^{th} element of list L
`MEM a L`	*member*	True if a is a member of list L
`LENGTH L`	*length*	Length of list L
`FST`	fst (a, b) = a	First component of a pair
`SND`	snd (a, b) = b	Second component of a pair
`SUC n`	$n + 1$	Successor of a *num*

molecules after the reaction between the molecules of the first element of the pair has taken place. A collection of EVFs is represented as a list of EVFs ((α *list* # α *list list*)*list*) in our formalization.

The elimination of Z-Conditional rule is the same as the elimination of implication rule (Modus Ponens) in propositional logic and thus it can be directly handled by the HOL4 simplification and rewriting rules. Similarly, the introduction of Z-Conditional rule can also be inferred from the rules of propositional logic and can be handled by the HOL4 system without the introduction of a new inference rule. The elimination of the Z-Conjunction rule allows us to infer the presence of a single molecule from an aggregate of inferred molecules. This rule is usually applied at the end of the reaction to check if the desired molecule has been obtained. Based on our data types, described above, this rule can be formalized in HOL4 by returning a particular molecule from a list of molecules:

DEFINITION 3.1. Elimination of Z-Conjunction Rule

```
⊢ ∀ L m.  z_conj_elim L m = if MEM m L then [m] else L
```

The function `z_conj_elim` has the data type (α *list* $\rightarrow$ α $\rightarrow$ α *list*). The above function returns the given element as a single element in a list if it is a member of the given list. Otherwise, it returns the argument list as it is.

The introduction of Z-Conjunction rule along with Z-Interaction allows us to perform a reaction between any of the available molecules during the experiment. Based on our data types, this rule is equivalent to the append operation of lists.

DEFINITION 3.2. Intro of Z-Conjunction and Z-Interaction

```
⊢ ∀ L m n.  z_conj_int L m n = FLAT [EL m L; EL n L]::L
```

The above definition has the data type (α *list list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ α *list list*). The HOL4 functions `FLAT` and `EL` are used to flatten a list of list to a single list and return a particular element of a list, respectively. Thus, the function `z_conj_int` takes a list `L` and appends the list of two of its elements `m` and `n` on its head.

Based on the laws of stoichiometry [4], the reacting molecules using Z-Conjunction operator have to be deleted from the aggregate of molecules. The following function represents this behaviour in our formalization:

DEFINITION 3.3. Reactants Deletion

```
⊢ ∀ L m n.  z_del L m n = if m > n
                then del (del L m) n
                else del (del L n) m
```

Here the function `del L m` deletes the element at index `m` of the list L and returns the updated list as follows:

Definition 3.4. Element Deletion

```
⊢ ∀ L. del L 0 = TL L ∧
  ∀ L n.  del L (n + 1) = HD L::del (TL L) n
```

Thus, the function `z_del L m n` deletes the m^{th} and n^{th} elements of the given list `L`. We delete the higher indexed element before the lower one in order to make sure that the first element deletion does not effect the index of the second element that is required to be deleted. The above data types and definitions can be used to formalize any molecular pathway (which is expressible using Zsyntax) and reason about its correctness within the sound core of the HOL4 theorem prover.

4. Formal Reasoning support for Zsyntax. Our main objective is to develop a framework that accepts a list of initial molecules and possible EVFs and allows the user to formally deduce the final outcomes of the corresponding biological experiment within the sound core of HOL4.

In this regard, we first develop a function that compares a particular combination of molecules with all the EVFs and upon finding a match introduces the newly formed molecule in the initial list and deletes the consumed instances.

Definition 4.1. EVF Matching

```
⊢ ∀ L E m n.  z_EVF L E 0 m n =
    if FST (EL 0 E) = HD L
      then (T,z_del (TL L ++ SND (EL 0 E)) m n
      else (F,TL L) ∧
  ∀ L E p m n.
z_EVF L E (p + 1) m n =
    if FST (EL (p + 1) E) = HD L
      then (T,z_del (TL ++ SND (EL (p + 1) E)) m n
      else z_EVF L E p m n
```

Here, HD and TL returns head and tail of a list respectively. Similarly, FST and SND returns first and second element of a pair respectively. The data type of the function `z_EVF` is: (α *list list* $\rightarrow$ (α *list*#α *list list*) *list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *bool* # α *list list*). The function `LENGTH` returns the length of a list. The function `z_EVF` takes a list of molecules `L` and recursively checks its head, or the top most element, against all elements of the EVF list `E`. If there is no match then the function returns a pair with its first element being false (`F`), indicating that no match occurred, and the second element equal to the tail of the input list `L`. Otherwise, if a match is found then the function replaces the head of list `L` with the second element of the EVF pair and deletes the matched elements from the initial list as these elements have already been consumed. This modified list is then returned along with a true (`T`) value, which acts as a flag to indicate an element replacement.

Next, in order to deduce the final outcome of the experiment, we have to call the function `z_EVF` recursively by placing all the possible combinations of the given molecules at the head of list `L` one by one. This can be done as follows:

Definition 4.2. Recursive Function to model the argument n in Def. 4.1

```
⊢ ∀ L E m.  z_recur1 L E m 0 =
    z_EVF (z_conj_int L m 0) E (LENGTH E - 1) m 0 ∧
  ∀ L E m n.
z_recur1 L E m (n + 1) =
    if FST (z_EVF (z_conj_int L M (n + 1)) E (LENGTH E - 1) m (n + 1)) ⇔ T
      then z_EVF (z_conj_int L m (n + 1)) E (LENGTH E - 1) m (n + 1)
      else z_recur1 L E m n
```

Definition 4.3. Recursive Function to model the augment m in Def. 4.1

```
⊢ ∀ L E n.z_recur2 L E 0 n =
    if FST (z_recur1 L E 0 n) ⇔ T
      then (T,SND (z_recur1 L E 0 n))
```

```
      else (F,SND (z_recur1 L E 0 n)) ∧
  ∀ L E m n.
z_recur2 L E (m + 1) n =
    if FST (z_recur1 L E (m + 1) n) ⇔ T
      then (T,SND (z_recur1 L E (m + 1) n))
      else z_recur2 L E m (LENGTH L - 1)
```

Both the above functions have the same data type, i.e, (α *list list* $\rightarrow$ (α *list* #α *list list*) *list* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *num* $\rightarrow$ *bool* #α *list list*). The function `z_recur1` calls the function `z_EVF` after appending the combination of molecules indexed by variables `m` and `n` using the introduction of Z-Conjunction rule. Thus, the new combination is always placed on the top of the list, which is passed as the variable `L` to the function `z_EVF`. Moreover, we provide the length of the EVF list (`LENGTH E - 1`) as the variable `p` of the function `z_EVF` so that the new combination is compared to all the entries of the EVF list. It is also important to note that the function `z_recur1` terminates as soon as a match in the EVF list is found. This function also returns a flag indicating the status of this match as the first element of its output pair. The second function `z_recur2` checks this flag and if it is found to be true (meaning that a match in the EVF list is found) then it terminates by returning the output list of the function `z_recur1`. Otherwise, it continues with all the remaining values of the variable `m` recursively. In the first case, i.e., when a match is found, the above two functions have to be called all over again with the new list. These iterations will continue until there is no match found in the execution of functions `z_recur1` and `z_recur2`. This overall behaviour can be modelled by the following recursive function:

Definition 4.4. Final Recursion Function for Zsyntax

```
⊢ ∀ L E m n.
z_deduction_recur L E m n 0 = (T,L) ∧
  ∀ L E m n q.
z_deduction_recur L E m n (q + 1) =
    if FST (z_recur2 L E m n) ⇔ T
      then z_deduction_recur
        (SND (z_recur2 L E m n)) E (LENGTH (SND (z_recur2 L E m n)) - 1)
        (LENGTH (SND (z_recur2 L E m n)) - 1) q
      else (T,SND (z_recur2 L E (LENGTH L - 1) (LENGTH L - 1)))
```

The function `z_deduction_rec` also has the same data type as the above mentioned recursion functions. The variable of recursion in this function should be assigned a value that is greater than the total number of EVFs so that the application of none of the EVF is missed to guarantee correct operation. Similarly, the variables `m` and `n` above should be assigned the values of `(LENGTH L - 1)` to ensure that all the combinations of the list `L` are checked against the elements of the list of EVFs. Thus, the final deduction function for Zsyntax can be expressed in HOL4 as follows:

Definition 4.5. Final Deduction Function for Zsyntax

```
⊢ ∀ L E. z_deduction L E =
    SND (z_deduction_recur L E (LENGTH L - 1) (LENGTH L - 1) LENGTH E)
```

The type of the function `z_deduction` is (α *list list* $\rightarrow$ (α *list* # α *list list*) *list* $\rightarrow$ α *list list*). It accepts the initial list of molecules and the list of valid EVFs and returns a list of final outcomes of the experiment under the given conditions. Next, the output of the function `z_deduction` has to be checked if the desired molecule is present in the list. This can be done by using the elimination of the Z-Conjunction rule given in Definition 1.

The formal definitions, presented in this section, allow us to recursively check all the possible combinations of the initial molecules against the first elements of given EVFs. In case of a match, the corresponding EVF can be applied and the process can restart again to find other possible matches from the new list of molecules. This process terminates when no more molecules are found to be reacting with each other and at this point we will have the list of post-reaction molecules. The desired result can then be obtained from these molecules using the elimination of Z-Conjunction rule. The main benefit of the development, presented in this section, is that it facilitates automated reasoning about the molecular biological experiments within the sound core of a theorem, which will be demonstrated later.

5. Formal Verification of Zsyntax Properties. In order to ensure the correctness and soundness of our definitions, we use them to verify a couple of properties representing the most important characteristics of molecular reactions. The first property deals with the case when there is no combination of reacting molecules in the list of molecules and in this case we verify that after the Zsyntax based experiment execution both the pre and post-experiment lists of molecules are the same. The second property captures the behaviour of the scenario when the given list of molecules contains only one set of reacting molecules and in this case we verify that after the Zsyntax based experiment execution the post-experiment list of molecules contains the product of the reacting molecules minus its reactants along with the remaining molecules provided initially. We represent these scenarios as formally specified properties in higher-order logic using our formal definitions, given in the previous section. These properties are then formally verified in HOL4.

5.1. Scenario 1: No Reaction. We verify the following theorem for the first scenario:

Theorem 1.

```
⊢ ∀ E L.
        ∼(NULL E) ∧ ∼(NULL L) ∧
        (∀ a m n.  MEM a E ∧ m < LENGTH L ∧ n < LENGTH L
              ⇒ ∼MEM (FST a) [HD (z_conj_int L m n)])
        ⇒ z_deduction L E = L
```

The variables `E` and `L` represent the lists of EVFs and molecules, respectively. The first two assumptions ensure that both of these lists have to be non-empty, which are the pre-conditions for a molecular reaction to take place. The next conjunct in the assumption list of Theorem 1 represents the formalization of the no-reaction-possibility condition as according to this condition no first element of any pair in the list of EVFs `E` is a member of the head of the list formed by the function `z_conj_int`, which picks the elements corresponding to the two given indices (that range over the complete length of the list of molecules `L`) and appends them as a flattened single element on the given list `L`. This constraint is quantified for all variables `a`, `m` and `n` and thus ensures that no combination of molecules in the list `L` matches any one of the first elements of the EVF list `E`. Thus, under this constraint, no reaction can take place for the given lists `L` and `E`. The conclusion of Theorem 1 represents the scenario that the output of our formalization of Zsyntax based reaction would not make any change in the given molecule list `L` and thus verifies that under the no-reaction-possibility condition our formalization also did not update the molecule list.

The verification of this theorem is interactively done by ensuring the no-update scenario for all molecule manipulation functions, i.e., `z_EVF`, `z_recur1`,
`z_recur2` and `z_deduction_recur`, under the no-reaction-possibility condition [1]. For example, the corresponding theorem for `z_EVF` function is as follows:

Theorem 2.

```
⊢ ∀ E L m n P.
        ∼(NULL E) ∧ ∼(NULL L) ∧ m < LENGTH L ∧ n < LENGTH L ∧
        P < LENGTH E ∧ (∀ a.  MEM a E ⇒ ∼MEM (FST a) [HD L])
        ⇒ z_EVF L E P m n = (F,TL L)
```

The assumptions of above theorem ensure that both lists `L` and `E` are not empty and the arguments of the function `z_EVF` are bounded by the `LENGTH` of `L` and `E`. The last conjunct in the assumption list models the no-reaction-possibility condition in the context of the function `z_EVF`. The conclusion of the theorem states that no update takes place under the given conditions by ensuring that the function `z_EVF` returns a pair with the first element being `F` (False), representing no match, and the second element being equal to `TL L`, which is actually equal to the original list `L` since an element was appended on head of `L` by the parent function.

5.2. Scenario 2: Single Reaction. The second scenario complements the first scenario and, caters for the case when a reaction is possible and we verify that the molecules list is indeed updated based on the outcomes of that reaction. In order to be able to track the reaction and the corresponding update, we limit ourselves to only one reaction in this scenario but since we verify a generic theorem (universally quantified) for all possibilities our result can be extended to cater for multiple reactions as well. The theorem corresponding to this scenario 2 is as follows:

Theorem 3.

```
⊢ ∀ E L z m' n'.
      ∼NULL E ∧ ∼NULL (SND (EL z E)) ∧ 1 < LENGTH L ∧ m' < LENGTH L ∧
      n' < LENGTH L ∧ z < LENGTH E ∧ ALL_DISTINCT (L ++ SND (EL z E)) ∧
      (∀ a b.  a ≠ b ⇒ FST (EL a E) ≠ FST (EL b E)) ∧ m' ≠ n' ∧
      (∀ K m n.  m < LENGTH K ∧ n < LENGTH K ∧
      (∀ j.  MEM j K ⇒ MEM j L ∨ ∃ q.  MEM q E ∧ MEM j (SND q)) ⇒
        if (EL m K = EL m' L) ∧ (EL n K = EL n' L)
          then HD (z_conj_int K m n) = FST (EL z E)
          else ∀ a.  MEM a E ⇒ FST a ≠ HD (z_conj_int K m n))
      ⇒ z_deduction L E = z_del (L ++ SND (EL z E)) m' n'
```

The first two assumptions ensure that neither the list `E`, i.e., the list of EVFs, nor the second element of the pair at index z of the list `E` is empty. Similarly, the third assumption ensures that the list `L`, i.e., the list of initial molecules, contains at least two elements. These constraints ensure that we can have at least one reaction with the resultant being available at index `z` of the EVF list. The next four assumptions ensure that the indices `m'` and `n'` are distinct and these along with the index `z` fall within the range of elements of their respective lists of molecules `L` or EVFs `E`. According to the next assumption, i.e., `ALL DISTINCT (L ++ SND (EL z E))`, all elements of the list `L` and the resulting molecules of the EVF at index `z` are distinct, i.e., no molecule can be found two or more times in the initial list `L` or the post-reaction list `E`. The next assumption, i.e., `(∀ a b. a ≠ b ⇒ FST (EL a E) ≠ FST (EL b E))`, guarantees that all first elements of the pairs in list `E` are also distinct. Note that this is different from the previous condition since the list `E` contains pairs as elements and the uniqueness of the pairs does not ensure the uniqueness of its first elements. The final condition models the presence of only one pair of reactants scenario. According to the assumptions of this implication condition, the variable `K` is used to represent a list that only has elements from list `L` or the second elements of the pairs in list `E`. Thus, it models the molecules list in a live experiment. Moreover, the variables `m` and `n` represent the indices of the list `K` and thus they must have a value less than the total elements in the list `K` (since the first element is indexed `0` in the HOL4 formalization of lists). Now, if the indices `m` and `n` become equal to `m'` and `n'`, respectively, then the head element of the `z_conj_int K m n` would be equal to `FST` of `EL z E`. Otherwise, for all other values of indices `m` and `n`, no combination of molecules obtained by `HD(Z_conj_int K m n)` would be equal to the first element of any pair of the list `E`. Thus, the if case ensures that the variables `m'` and `n'` point to the reacting molecules in the list of molecules `L` and the variable `z` points to their corresponding resultant molecule in the EVF list. Moreover, the else case ensures that there is only one set of reacting molecules in the list `L`. The conclusion of the theorem formally describes the scenario when the resulting element, available at the location `z` of the EVF list, is appended to the list of molecules while the elements available at the indices `m'` and `n'` of `L` are removed during the execution of the function `z_deduction` on the given lists `L` and `E`.

The proof of Theorem 3 is based on verifying sub-goals corresponding to this scenario for all the sub-functions, i.e., `z_EVF`, `z_recur1`, `z_recur2` and `z_deduction_recur`. The formal reasoning for all of these proofs involved various properties of the `del` function for a list element and some of the key theorems developed for this purpose in our development are given in Table 5.1 and more details can be found in [1].

The formalization described in this section consumed about 500 man hours and approximately 2000 lines of HOL4 code, mainly due to the undecidable nature of higher-order logic. However, this effort raises the confidence level on the correctness of our formalization of Zsyntax. This fact distinguishes our work from all the other formal methods based techniques used in the context of BRNs, where the deduction rules are applied without being formally checked. Moreover, our formally verified theorems can also be used in the formal analysis of molecular pathways. The assumptions of these theorems provide very useful insights about the constraints under which a reaction or no reaction would take place. To the best of our knowledge, this is the first time that properties, like Theorems 1 and 3, about a molecular pathway experiment have been formally verified. Thus, the identification of these properties and their formal verification both constitute contributions of this paper.

As part of this work, we also developed a simplifier `Z_SYNTAX_SIMP` [1] that simplifies the proof with a single iteration of the function `z_deduction_recur` and works very efficiently with the proofs involving our functions. The proof steps can be completely automated and the proof can be done in one step as well. However, we have kept the reasoning process manual purposefully as this way users can observe the status of the reaction

TABLE 5.1
Formally verified Properties of the `del` *Function*

Signature	Theorem
del_ASSOC_THM	$\vdash \forall$ L E m. m $<$ LENGTH L $\Rightarrow$ del (L ++ E) m = del L m ++ E
del_LENGTH_THM	$\vdash \forall$ L E m. m $<$ LENGTH L $\Rightarrow$ LENGTH (del L m) = LENGTH L $-$ 1
del_EL_THM	$\vdash \forall$ L m n. m $<$ n $\wedge$ n $<$ LENGTH L $\wedge$ 1 $<$ LENGTH L $\Rightarrow$ EL m L = EL m (del L n)
del_DISTINCT_THM	$\vdash \forall$ L n. n $<$ LENGTH L $\wedge$ ALL_DISTINCT L $\Rightarrow$ ALL_DISTINCT (del L n)
del_MEM_THM	$\vdash \forall$ L a m. m $<$ LENGTH L $\wedge$ MEM a (del L m) $\Rightarrow$ MEM a L
del_NOT_MEM_THM	$\vdash \forall$ L m. ALL_DISTINCT L $\wedge$ m $<$ LENGTH L $\Rightarrow\sim$ MEM (EL m L) (del L m)

at every iteration, which is a very useful feature to get an insight of what is happening inside a reaction. Each application of `Z_SYNTAX_SIMP` on the reaction, depicted in Fig. 7.1, would result in moving from a state n to $n+1$.

6. Graphical User Interface. The interest of using formal methods in the domains of bio-informatics and system biology has been rapidly growing since the last decade. However, there is a notable gap between biologists, who prefer wet lab experiments, and computer scientists, who can provide efficient tools to perform computer experiments with a unified framework. This fact has somewhat limited the usage of formal methods by the bio-informatics and pharmaceutical industry in research and development. We believe that one of the major reasons for the above-mentioned gap is the unavailability of biologists friendly formal tools. The idea on which our GUI functions is that the user needs to provide the initial list of molecules and possible EVFs. Consequently, the verification relating the two inputs can be done automatically. The GUI, developed in C#, is capable of initiating a HOL4 session in the command line and automatically converting user provided data into the language supported by HOL4. These translated texts are then utilized to form a HOL 4 theorem, which is then passed to HOL4 via the command line and the verification process automatically starts using the formal definitions and tactics developed in our work. Figure 6.1 provides a snapshot of our GUI while handling the example of Glycolysis pathway, which will be described in more detail in the next section. Many real-world

FIG. 6.1. *Graphical User Interface*

case studies have been automatically verified, including the examples that are described in the next section, by our GUI. Upon the successful verification of a theorem the GUI returns a message that the given pathway has been successfully deduced. The current version of our GUI is available for download at [1] and we are working to further enhance its usability by providing graphical insights about the verification.

7. Case Studies. In the section, we present the utilization of our Zsyntax formalization and its corresponding properties by two real-world cases studies: 1) Reaction involving TP53 degradation. 2) Molecular pathway of Glycolysis which is responsible for the production of Pyruvate and also provides energy for cell growth.

7.1. TP53 Degradation. TP53 gene encodes p53 protein, which plays a crucial role in regulating the cell cycle of multicellular organisms and, thus, functions as a tumour suppressor for preventing cancer [4]. We utilize the proposed framework to analyse the TP53 degradation reaction [4]. The regulatory loop leading to the TP53 degradation is illustrated in Fig. 7.1, where the dark coloured circles denote the list of molecules given by biologists and the hollow circles depict the molecules that do not interact with any other molecule in a particular step of the reaction.

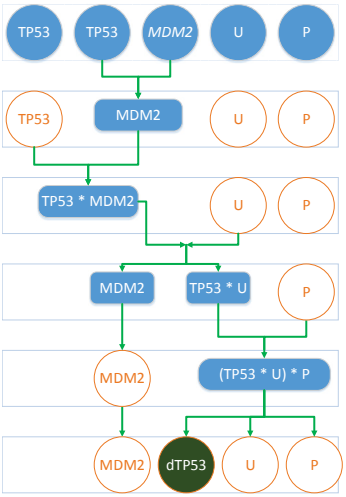


FIG. 7.1. *Reaction Representing Degradation of TP53*

A theorem representing the regulatory loop involving MDM2, *MDM2* and TP53 and leading to TP53 degradation [4], can be described as follows:

$$\texttt{TP53 \& TP53 \& MDM2 \& U \& P} \vdash \texttt{d(TP53)}$$

In HOL4, this biological reaction can be written as the following theorem using our reported framework:

```
⊢ DISTINCT [TP53;dTP53;P;U;iMDM2;MDM2] ⟹
z_conj_elim (z_deduction [[TP53];[TP53];[iMDM2];[U];[P]]
                         [([TP53;MDM2],[[MDM2]]);
                          ([MDM2;TP53],[[TP53;MDM2]]);
                          ([TP53;MDM2;U],[[TP53;U];[MDM2]]);
                          ([TP53;U;P],[[dTP53];[U];[P]])]) [dTP53]
=[[dTP53]]
```

The `DISTINCT` function used in the assumption of the above theorem ensures that all the molecules are distinct. The first list argument of the function `z_deduction` is the initial aggregate (IA) of molecules that are available for reaction and the second list argument of the function `z_deduction` represents the valid EVFs for this reaction. Thus, the function `z_deduction` would deduce the final list of molecules under these particular conditions. The function `z_conj_elim` will return the molecule `dTP53` if it is present in the post-reaction list of molecules, as previously described.

Figure 7.1 shows the step-wise reaction leading to d(TP53). The blue-coloured circles show the chemical interactions and green colour represents the desired product in the pathway, whereas each rectangle shows the total number of molecules in the reaction at a given time. It is obvious from the figure that whenever a reaction yields a product, the reactants get consumed (no longer remain in the list) hence satisfying the stoichiometry of a reaction.

The script of TP53 Degradation Theorem is given below:

```
e(RW_TAC std_ss[DISTINCT,UNIQ_MEM,MEM]);
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(Z_SYNTAX_SIMP z_EVF_def );;
e(REWRITE_TAC [MEM,z_conj_elim_def]);
```

It is quite evident from the proof script of Theorem 1 that its proof steps can be completely automated and the proof can be done in one step as well. However, we have kept the reasoning process manual purposefully as this way the users can observe the status of the reaction at every iteration. For example, each application of `Z_SYNTAX_SIMP` on the reaction, depicted in Fig. 7.1, would result in moving from a state n to $n + 1$.

Note that we can also record the time required to complete each iteration. For example, in case of above examples, the time for each step in the above case-study is given in Table 7.1.

TABLE 7.1
Runtime per Iteration

Iteration	Duration (Seconds)
$1 \rightarrow 2$	95.969
$2 \rightarrow 3$	51.829
$3 \rightarrow 4$	27.127
$4 \rightarrow 5$	28.062
$5 \rightarrow 6$	0.2130

7.2. Molecular Pathway of Glycolysis. Glycolysis is a metabolic pathway which converts glucose into pyruvate and energy released in this process is used for the cell growth [18]. In molecular biology literature, Glycolysis is a determined sequence of ten enzyme-catalyzed reactions. Formation of Glucose-6-phosphate, Fructose-1,6-bisphosphate (F1,6P), etc. are intermediate steps in glycolysis [18]. In order to deduce the complete Glycolysis pathway, it is convenient to divide the pathway in three blocks as shown in Fig. 7.2. All the abbreviations used in Fig. 7.2 are described in Table 7.5. The dark coloured circles represent enzymes and each of them participates in the reaction at a certain point and only helps in the formation of new molecules. It is important to note that enzymes do not get consumed, which means that they retain their state after the completion of the reaction.

The first phase of the pathway leads to formation of Fructose-1,6-bisphosphate (F1,6P). The theorem representing the reaction of the glycolytic pathway leading from D-Glucose to F1,6P [4] can be described in classical Zsyntax format as follows:

```
Glc & HK & GPI & PFK & ATP & ATP ⊢ F1,6P
```

We can formally encode this theorem in HOL4 as follows:

```
⊢ DISTINCT [Glc; HK; GPI; PFK; ATP; ADP; G6P; F6P; F16P] ⟹
(z_conj_elim (z_deduction [[Glc];[HK];[GPI];[PFK];[ATP];[ATP]]
                          [([Glc;HK],[[HK;Glc]]);
                           ([HK;Glc;ATP],[[HK];[G6P];[ADP]]);
                           ([G6P;GPI],[[F6P];[GPI]]);
                           ([F6P;PFK],[[PFK;F6P]]);
                           ([PFK;F6P;ATP],[[PFK];[F16P];[ADP]])] ) [F16P]
= [[F16P]]
```

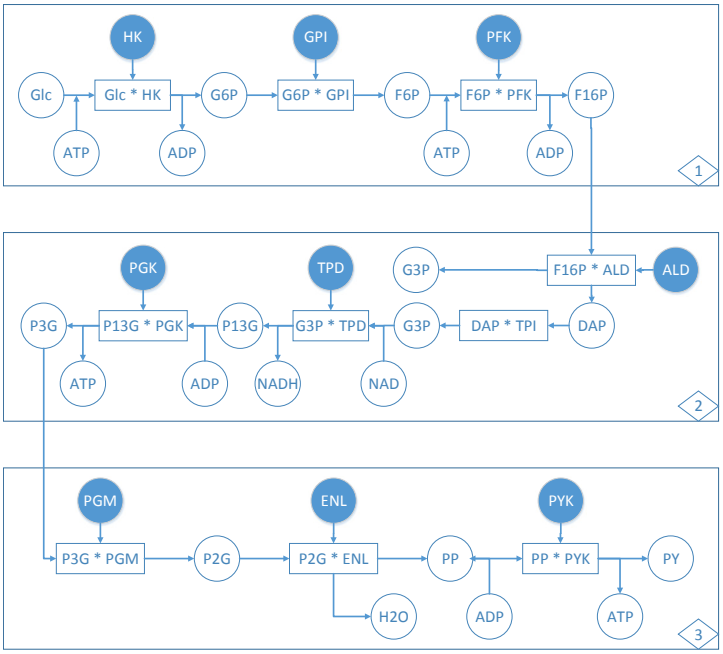


FIG. 7.2. *Formation of Pyruvate through Glycolysis*

where the first list argument of the function `z_deduction` is the initial aggregate (IA) of molecules that are available for reaction and the second list represents the valid EVFs for this reaction. The EVFs mentioned in the form of pairs and involving the molecules (G6P, F6P, etc.) are obtained from wet lab experiments, as reported in [4]. The `DISTINCT` function used above makes sure that all molecules (from initial aggregate and EVFs) used in this theorem are indeed distinct. Thus, the function `z_deduction` would deduce the final list of molecules under these particular conditions. The function `z_conj_elim` returns the molecule `F1,6P` if it is present in the post-reaction list of molecules. The verification time required for each iteration step is given in Table 7.2.

TABLE 7.2
Runtime per Iteration

Iteration	Duration (Seconds)
$1 \rightarrow 2$	11.996
$2 \rightarrow 3$	7.376
$3 \rightarrow 4$	12.964
$4 \rightarrow 5$	12.756
$5 \rightarrow 6$	9.240
$6 \rightarrow 7$	0.048

Fructose-1,6-bisphosphate formed in the above reaction further reacts with different enzymes to form 3-Phosphoglycerate, which completes the second phase of Glycolysis pathway. Note that during this phase two molecules of G3P are formed and each molecule follows the same path. For the sake of simplicity, we only provided one NAD molecule so that only one molecule can complete the pathway to form one Pyruvate molecule. After the successful deduction of first phase of Glycolysis, we can model the second phase as follow:

$$\texttt{F16P \& ADP \& ALD \& TPI \& NAD \& TPD \& PGK} \vdash \texttt{P3G}$$

which can be formalized in HOL4 as follows:

```
⊢ DISTINCT [F16P; ALD; TPI; NAD; TPD; PGK; G3P;
            DAP; P13G; P3G; NADH; ADP; ATP] ⟹
(z_conj_elim (z_deduction [[F16P];[ADP];[ALD];[TPI];[NAD];[TPD];[PGK]]
                          [([F16P;ALD],[[G3P];[DAP]]);
                           ([DAP;TPI],[[G3P];[TPI]]);
                           ([G3P;TPD],[[TPD;G3P]]);
                           ([TPD;G3P;NAD],[[P13G];[TPD];[NADH]]);
                           ([P13G;PGK],[[PGK;P13G]]);
                           ([PGK;P13G;ADP],[[PGK];[P3G];[ATP]])] ) [P3G]
= [[P3G]]
```

The time required for each iteration of the second phase of Glycolysis is given in Table 7.3.

TABLE 7.3
Runtime per Iteration

Iteration	Duration (Seconds)
$1 \rightarrow 2$	24.940
$2 \rightarrow 3$	20.532
$3 \rightarrow 4$	22.676
$4 \rightarrow 5$	16.804
$5 \rightarrow 6$	20.780
$6 \rightarrow 7$	17.120
$7 \rightarrow 8$	0.188

3-Phosphoglycerate (P3G) formed in the above step further reacts with other enzymes to yield the final product of Glycolysis pathway, i.e., Pyruvate. A theorem representing the conversion of 3-Phosphoglycerate (P3G) to Pyruvate (PY) can be described as follows:

```
P3G & ADP & PGM & ENL & PYK ⊢ PY
```

Consequently, the corresponding HOL4 expression is given as follows:

```
⊢ DISTINCT [PGM; ENL; PYK; P3G; P2G; H2O; PP; PY; ADP] ⟹
(z_conj_elim (z_deduction [P3G];[ADP];[PGM];[ENL];[PYK]]
                          [([P3G;PGM],[[P2G];[PGM]]);
                           ([P2G;ENL],[[H2O];[PP];[ENL]]);
                           ([PP;PYK],[[PYK;PP]]);
                           ([PYK;PP;ADP],[[PYK];[PY];[ATP]])] ) [PY]
= [[PY]]
```

Similar to the above steps, the timing statistics for the above theorem are given in Table 7.4.

TABLE 7.4
Runtime per Iteration

Iteration	Duration (Seconds)
$1 \rightarrow 2$	5.860
$2 \rightarrow 3$	5.676
$3 \rightarrow 4$	9.384
$4 \rightarrow 5$	7.784
$5 \rightarrow 6$	0.044

This completes the description of the utilization of our work to formally reason about two real-world case studies, i.e., TP53 degradation and molecular pathway of Glycolysis.

Our HOL4 proof script is available for download [1], and thus can be used for further developments and analysis of different molecular pathways. It is important to note that formalizing Zsyntax and then verifying its

TABLE 7.5
Abbreviations of molecules used in Glycolysis

Abbr.	**Biological Entity**	**Abbr.**	**Biological Entity**
Glc	Glucose	PYK	Pyruvate Kinase
HK	Hexokinase	G6P	Glucose-6-phosphate
ATP	Energy	F6P	Fructose-6-phosphate
GPI	Phosphoglucomutase	F16P	Fructose-1,6-diphosphate
PFK	Phosphofructokinase	G3P	Glyceraldehyde-3-Phosphate
ALD	Aldolase	DAP	Dihydroxyacetonephosphate
TPI	Triosephosphateisomerase	P13G	1,3-biphosphoglycerate
TPD	Triosephosphate Dehydrogenase	P2G	2-phosphoglycerate
PGK	Phosphoglycerokinase	PP	Phophoenolpyruvate
PGM	Phosphoglyceromutase	PY	Pyruvate
ENL	Enolase	P3G	3-Phosphoglycerate

properties was a very tedious effort. However, it took around 10 lines of code to formally define and verify each theorem related to the above case studies in HOL4, which clearly illustrates the usefulness of our foundational work. We have shown that our formalization is capable of modelling molecular reactions using Zsyntax inference rules, i.e., given a set of possible EVFs, our formalism can derive a final aggregate **B** from an initial aggregate **A** automatically. In case of a failure to deduce **B**, the proposed method still provides the biologist with all the intermediate steps so that one can examine the reaction in detail and figure out the possible cause of failure. The evident benefit of our reasoning approach is its automatic nature as the user does not need to think about the proof steps and which EVFs to apply where. However, the most useful benefit of the proposed approach is its accuracy as the theorems are being verified in a formal way using a sound theorem prover. Thus, there is no risk of human error or wrong application of EVFs. Finally, due to the computer-based analysis, the proposed approach is much more scalable than the paper-and-pencil based analysis presented in [4].

8. Conclusions. Most of the existing formal verification research related to molecular biology has been focussed on using model checking. As a complementary approach, the primary focus of the current paper is on using a theorem prover for reasoning about molecular pathways. The main strength of this approach, compared to existing model checking related work, is that the underlying methods and deduction rules can also be formally verified besides the verification of a particular molecular pathway case. Leveraging upon this strength, we formally verified two key behavioural properties of molecular pathways based on the Zsyntax language, which presents a deduction style formalism for molecular biology in the most biologist-centred way. Besides ensuring the correctness of our formalization of the Zsyntax operators and inference rules, the formally verified properties also play a vital role in reasoning about molecular pathways in the sound core of a theorem prover. The practical utilization and effectiveness of the proposed development has been shown by presenting the automatic analysis of reaction involving TP53 degradation and metabolic pathway known as Glycolysis.

The proposed work opens the doors to many new directions of research. Firstly, we are enhancing our GUI to add more biologist friendly features in it. Moreover, we are also targeting some larger case studies, such as Dysregulation of the cell cycle pathway during tumour progression [16] and Fanconi Anemia/Breast Cancer (FA/BRCA) pathway [26]. Another interesting future direction is to leverage upon the high expressiveness of higher-order-logic and utilize calculus and differential theoretic reasoning to add reaction kinetics support in our formalism.

REFERENCES

[1] S. AHMAD. Formal Reasoning about Molecular Pathways - HOL Proof Script. `http://save.seecs.nust.edu.pk/projects/holsyntax/holzsyntax.html`, 2014.

[2] S. AHMAD, O. HASAN, AND U. SIDDIQUE. Towards Formal Reasoning about Molecular Pathways in HOL. In *International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 378–383. IEEE, 2014.

[3] C. Baier and J. Katoen. *Principles of Model Checking*. MIT Press, 2008.
[4] G. Boniolo, M. D'Agostino, and P. Di Fiore. Zsyntax: a Formal Language for Molecular Biology with Projected Applications in Text Mining and Biological Prediction. *PloS ONE*, 5(3):e9511–1–e9511–12, 2010.
[5] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.
[6] V. Danos and C. Laneve. Formal Molecular Biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
[7] N.H. Hunt et al. Immunopathogenesis of Cerebral Malaria. *International Journal for Parasitology*, 36(5):569–582, 2006.
[8] L. Trilling Fab. Corblin, E. Fanchon. Applications of a Formal Approach to Decipher Discrete Genetic Networks. *BMC Bioinformatics*, 11(1):385, 2010.
[9] W. Fontana. Systems Biology, Models, and Concurrency. *SIGPLAN Notices*, 43(1):1–2, January 2008.
[10] F. Cassez et al. G.Bernot. Semantics of Biological Regulatory Networks. *Electronic Notes Theoretical Computer Science*, 180(3):3–14, 2007.
[11] Peter J. E. Goss and J. Peccoud. Quantitative Modeling of Stochastic Systems in Molecular Biology by using Stochastic Petri Nets. *Proceedings of the National Academy of Sciences*, 95(12):6750–6755, 1998.
[12] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.
[13] K. Hirayama. Genetic Factors Associated with Development of Cerebral Malaria and Fibrotic Schistosomiasis. *Korean J. Parasitol*, 40(4):165–172, Dec 2002.
[14] M. Magnin L. Paulevé and O. Roux. Abstract Interpretation of Dynamics of Biological Regulatory Networks. *Electronic Notes Theoretical Computer Science*, 272(0):43–56, 2011.
[15] C.J. Langmead. Generalized Queries and Bayesian Statistical Model Checking in Dynamic Bayesian Networks: Application to Personalized Medicine. In *Proc. International Conference on Computational Systems Bioinformatics*, pages 201–212, 2009.
[16] R. Maglietta, V. Liuzzi, E. Cattaneo, E. Laczko, A. Piepoli, A. Panza, M. Carella, O. Palumbo, T. Staiano, F. Buffoli, A. Andriulli, G. Marra, and N. Ancona. Molecular Pathways Undergoing Dramatic Transcriptomic Changes During Tumor Development in the Human Colon. *BMC Cancer*, 12(1):608, 2012.
[17] R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1977.
[18] D. Nelson. *Lehninger Principles of Biochemistry*. W.H. Freeman, New York, 2008.
[19] L.C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1996.
[20] R. Pelánek. Fighting State Space Explosion: Review and Evaluation. In *Formal Methods for Industrial Critical Systems*, volume 5596 of *Lecture Notes in Computer Science*, pages 37–52. Springer, 2008.
[21] J. Pospchal and V. Kvasnika. Reaction Graphs and a Construction of Reaction Networks. *Theoretica Chimica Acta*, 76(6):423–435, 1990.
[22] A. Regev and E. Shapiro. Cells as Computation. *Nature*, 419:343, 2002.
[23] A. Regev and E. Shapiro. The π-Calculus as an Abstraction for Biomolecular Systems. In *Modelling in Molecular Biology*, Natural Computing Series, pages 219–266. Springer, 2004.
[24] A. Regev, W. Silverman, and E. Y. Shapiro. Representation and Simulation of Biochemical Processes Using the pi-Calculus Process Algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.
[25] M. Rizzotti and A. Zanardo. Axiomatization of Genetics. 1. Biological Meaning. *Journal of Theoretical Biology*, 118(1):61–71, 1986.
[26] A. Rodríguez, D. Sosa, L. Torres, B. Molina, S. Frías, and L. Mendoza. A Boolean Network Model of the FA/BRCA Pathway. *Bioinformatics*, 28(6):858–866, 2012.
[27] L. Thomas and R. d' Ari. *Biological Feedback*. CRC Press, USA, 1990.
[28] R. Thomas. *Kinetic Logic: A Boolean Approach to the Analysis of Complex Regulatory Systems*, volume 29 Lecture Notes in Biomathematics. Springer-Verlag, 1979.
[29] J.J Tyson, C.A. Nagy, and B. Novak. The Dynamics of Cell Cycle Regulation. *Bioessays*, 24(12):1095–1109, 2002.
[30] O. Wolkenhauer, D. Shibata, and M.D. Mesarovic. The Role of Theorem Proving in Systems Biology. *Journal of Theoretical Biology*, 300(0):57–61, 2012.
[31] J.H. Woodger, A. Tarski, and W.F. Floyd. *The Axiomatic Method in Biology*. The University Press, 1937.
[32] A. Zanardo and M. Rizzotti. Axiomatization of Genetics 2. Formal Development. *Journal of Theoretical Biology*, 118(2):145–152, 1986.

Edited by: Jesus Carretero
Received: September 14, 2014
Accepted: October 24, 2014

DOI 10.12694/scpe.v16i1.1060
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 53–69. http://www.scpe.org

ACCELERATING COMPARATIVE GENOMICS WORKFLOWS IN A DISTRIBUTED ENVIRONMENT WITH OPTIMIZED DATA PARTITIONING AND WORKFLOW FUSION

OLIVIA CHOUDHURY, NICHOLAS L. HAZEKAMP, DOUGLAS THAIN, SCOTT J. EMRICH*

Abstract. The advent of next generation sequencing technology has generated massive amounts of biological data at unprecendented rates. Comparative genomics applications often require compute-intensive tools for subsequent analysis of high throughput data. Although cloud computing infrastructure plays an important role in this respect, the pressure from such computationally expensive tasks can be further alleviated using efficient data partitioning and workflow fusion. Here, we implement a workflow-based model for parallelizing the data-intensive tasks of genome alignment and variant calling with BWA and GATK's HaplotypeCaller. We explore three different approaches of partitioning data, **granularity-based**, **individual-based**, and **alignment-based** , and how each affect the run time. We observe **granularity-based partitioning** for BWA and **alignment-based partitioning** for HaplotypeCaller to be the optimal choices for the pipeline. We further discuss the methods and impact of workflow fusion on performance by considering different levels of fusion and how it affects our results. We identify the various open problems encountered, such as understanding the extent of parallelism, using heterogenous environments without a shared file system, and determining the granularity of inputs, and provide insights into addressing them. Finally, we report significant performance improvements, from 12 days to under 2 hours while running the BWA-GATK pipeline using partitioning and fusion.

Key words: genome alignment, variant calling, workflow fusion, data partitioning, performance

AMS subject classifications. 68M14, 92D20

1. Introduction.

Next generation sequencing (NGS) [35] techniques have had widespread impact in fields such as molecular medicine, evolution, human migration, DNA forensics, and agriculture by inexpensively generating Giga base-pairs (Gbp) per machine day [29]. As sequencing throughput increases, the major research bottlenecks are the time and computational resources demanded for managing, deciphering and analyzing such large-scale biological data. Genome alignment and variant calling are the two most important stages of comparative genomics applications, which often require weeks or months for downstream analysis of NGS data. Thus, developing high throughput workflows for such data-intensive applications is an important and challenging problem in bioinformatics.

In the last few years, many alignment and variant discovery tools have implemented sophisticated algorithms to reduce computational resource requirement and run time for analyzing massive biological data [17, 15, 27, 20, 24, 13]. The extent of parallelism that can be achieved by adopting the optimization techniques focused on multicore servers is often limited. For instance, Burrows Wheeler Aligner (BWA) [19] is one of the fastest and most accurate alignment tools currently available. However, for our test data of 100-fold coverage of 50 oak individuals' genomes the multithreading option provided by BWA did not significantly reduce run time. Similarly, GATK's SNP and indel calling walker HaplotypeCaller [8] generates more accurate results compared to the other variant calling tools, but often requires weeks or months to analyze high coverage NGS data. For example, HaplotypeCaller required 12 days to determine variants for our test data, and the in-built options for runtime improvement on a multicore machine also did not considerably reduce run time.

As the underlying algorithms of both tools support coarse-grained parallelism, the trade off between accuracy and speedup can be mitigated by creating a parallelizable workflow [14]. Such a workflow can harness resources from distributed systems by dividing the workload into independent tasks and executing the tasks parallely. In this regard, one of the key factors responsible for achieving a considerable speedup is efficient partitioning of data at the preliminary stage of the workflow. As BWA and HaplotypeCaller do not provide any method of data partitioning, we explored different ways of doing so, some specific to a particular tool. For each tool, we identified the best partitioning technique and tested the same for the entire pipeline. We also discussed several open problems that are likely to be encountered while developing such data-intensive applications, particularly understanding the scope of parallelism within an entire pipeline, finding the optimal partitioning value for parallelism, and using a heterogeneous environment without a shared file system.

*Corresponding Author. Email: semrich@nd.edu. All authors are affiliated with the Department of Computer Science and Engineering at the University of Notre Dame, Notre Dame, IN, USA.

For BWA, we partitioned the data following **granularity-based** and **individual-based** approaches. For the former, we split the query file based on a predetermined optimal granularity value whereas for the latter, we split it based on individual names. We observed that times taken for the alignment procedure in both approaches were similar, but there was an additional cost incurred in the splitting step of the **individual-based** approach. Here, each read from the query had to be compared with the barcode information, which ended up being time consuming for approximately 140 million reads. Thus, **granularity-based** approach of data decomposition was efficient for parallelized execution of BWA.

The second stage of the pipeline involved variant calling using HaplotypeCaller. Similar to BWA, we tested **granularity-based** and **individual-based** approaches for the input data containing alignment information in BAM format [20]. In addition to that, we adopted a new method, named **alignment-based partitioning**. Here, we first split the reference file into multiple bins, each containing unique contigs. We then split the concatenated SAM output of BWA into smaller files, each having alignment information pertaining to a reference bin. We then run HaplotypeCaller for a pair of sorted BAM file and its corresponding reference bin. Contrary to the other two approaches, the **alignment-based** approach caused a significant run time improvement for HaplotypeCaller as the search space was considerably reduced while preserving all required data in smaller reference files. The overall run time of the pipeline for genome mapping and variant calling, which earlier took approximately 12 days, was now drastically reduced to 2 hours.

Makeflow was used as the workflow description language and execution engine. Makeflow allows for the description of static DAG workflows. Work Queue was used as the batch execution engine. It implements the master-worker framework. With Work Queue a determined number of workers can be initialized and used, allowing for caching between jobs on the worker.

After exploring the parallelism within each component of the pipeline, we merged the pipeline into a single workflow to explore the effects of workflow fusion. We define workflow fusion as the process of merging two sequential workflows into a single workflow that takes advantage of caching, elimination of choke points, and higher throughput using stacked partitioning methods. Here, we also explored how different partitioning methods perform within a single fused workflow, and discuss how workflow fusion can apply in data-intensive pipelines often found in bioinformatics.

2. Methods.

2.1. Overview of Genome Alignment and BWA. The millions of short reads generated by next generation sequencing techniques are usually compared to the genome of a model organism, known as the reference genome, for further biological analysis. One accurate means of genome comparison is genome alignment. In Fig. 2.1, after aligning sequences A and B, the vertical bars denote the positions of matches whereas the hyphens represent insertions or deletions, commonly known as indels. There are two major categories of mapping algorithms. The first implements the Burrows Wheeler Transform (BWT) [3] for data compression, and the second method is based on hashing to speed up alignments. Depending on the genome size, the entire alignment procedure may take several days to compute similarity between two given sequences. For instance, short read alignment tools like MAQ [22] and SOAP [26] typically require more than 5 CPU-months and 3 CPU-years, respectively, for aligning 140 billion base pairs (Bbp) [18].

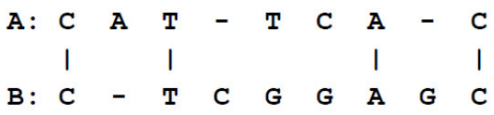


FIG. 2.1. *Genome alignment between sequences A and B showing matches, mismatches, and indels*

One of the widely used alignment tools, Burrows Wheeler Aligner (BWA) employs the Burrows Wheeler Transform (BWT) algorithm to align short queries with low error rate as well as long queries with higher error rates. BWA is light-weight, supports paired-end mapping, gapped alignment, and various file formats, like ILLUMINA [4] and ABI SOLiD [12]. The default output format for all its algorithms is SAM (Sequence

Alignment Map), which can further be analyzed using the open-source SAMtools package and others to visualize the alignments and call variants [20].

2.2. Overview of Variant Calling and HaplotypeCaller. Variants are allelic sequences that differ from the reference sequence by a single base pair or a comparatively longer interval. In Fig. 2.1, at the fifth position of the aligned sequence there occurs a mismatch or single nucleotide polymorphism (SNP). SNPs can account for the variation in phenotypic traits and disease susceptibility among different individuals. They can also serve as markers in genome wide association studies (GWAS) and help in identifying genes associated with various traits or diseases. The performance of a variant calling tool is largely dependent on the quality and coverage of sequencing data. Popular variant calling tools like SAMtools and MAQ use a Bayesian statistical model to evaluate the posterior probability of genotypes.

GATK employs similar, yet more sophisticated Bayesian algorithms. For our experiments we used GATK's HaplotypeCaller walker because of its high sensitivity. HaplotypeCaller functions by indexing the input set and creating walkers to locate variations between the query and reference. Once a difference is detected, it performs local assemblies to fill gaps or correct mistakes. Though it generates accurate results, it is less frequently used as it takes a long time check more variants, both indels and SNPs, for a given sequence. To remedy the slower performance, we opted for parallel execution of its subtasks.

2.3. Framework of the parallelized pipeline. For our active projects, BWA (version 0.7.5) and GATK's (version 2.5-2) SNP and indel calling walker HaplotypeCaller produced the best biological results. As seen in Table 4.5, the method of genome alignment, preparation of inputs for subsequent variant calling, and determination of variants using these tools in a sequential order took 12 days to complete. Many newly developed variant detection tools, including GATK's HaplotypeCaller improve genotype calling by increasing runtime [33], which is a major bottleneck.

A possible approach to address this problem is to develop efficient algorithms or data structures to reduce the search space during the computationally expensive task of alignment or variant calling [21]. Considering the massive size of NGS data and the considerably high computational resource requirement for their analysis, parallelizing the tools using a workflow-based model that can harness resources from clusters, clouds, and grids is a more tractable solution. We conceptually divided up genome alignment and variant discovery, and used the Makeflow [38] language to define each subtask along with its dependencies. Makeflow can employ various systems, including multi-core machines, batch systems like the Sun Grid Engine (SGE) [9] and Condor Pool [28] to execute the individual tasks. For our experiments, we enabled Makeflow to use the Work Queue [2] master-worker framework as an alternative scalable framework to process data across clusters, clouds, and grids. The Work Queue framework also handles all data transfers and can cache files when running jobs remotely.

We identified that a key feature for assuring improved run time during parallelization is efficient partitioning of data. We adopted different approaches of data partitioning for BWA and HaplotypeCaller and compared and contrasted each combination using Work Queue-derived resources.

- For **granularity-based partitioning** in BWA, we tested the workflow by varying the number of partitions and the size of each partition. We determined the optimal granularity value or partition size to be the one for which run time was the lowest. For our query data containing approximately 140 million reads, splitting it into smaller chunks such that each contained 200000 reads proved to be the optimal choice. The splitting resulted in the generation of 715 smaller files, which were then used for running BWA.
- For **individual-based partitioning** in BWA, we used coarser granularity by dividing the pooled data into multiple files, each corresponding to an individual. As our data comprised genomic sequences of 50 individuals, we divided it into 50 files based on individual names.

Fig. 2.2 illustrates the workflow of these approaches when applied to BWA. The split function refers to the creation of smaller query files based on a granularity value (for **granularity-based partitioning**) or individual names (for **individual-based partitioning**). In both cases, we assigned the task of running BWA on each pair of small query data and reference data to a work queue worker. At the completion of alignment phase, we added read group and platform information to each BWA output (SAM file) and compressed them into their sorted and indexed binary versions (BAM) to be compatible with HaplotypeCaller.

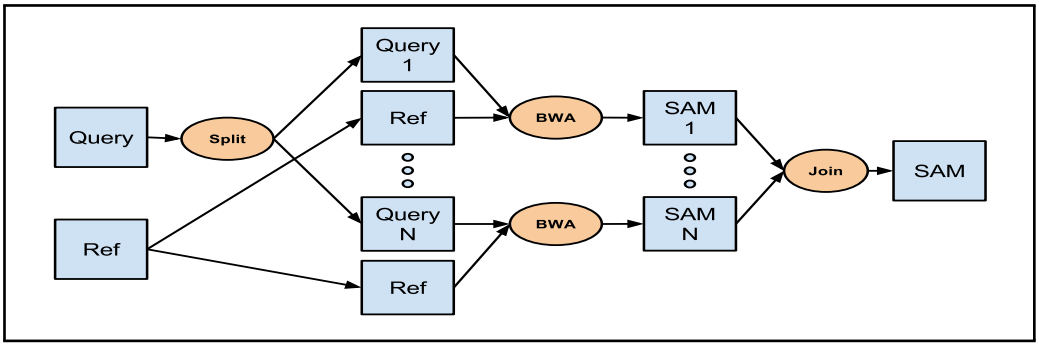


FIG. 2.2. *Framework of* ***granularity-based*** *and* ***individual-based*** *data partitioning approaches in BWA. N=715 for* ***granularity-based*** *and N=50 for* ***individual-based****. At the end, the output files (SAM format) of BWA were joined to form a single file containing all the alignment information.*

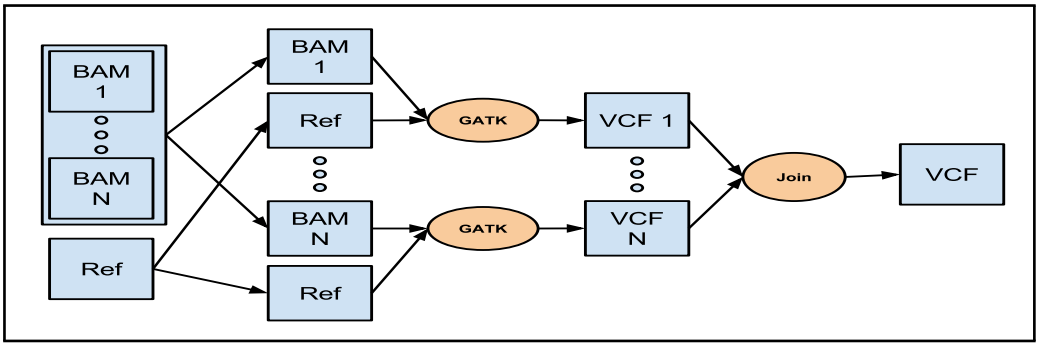


FIG. 2.3. *Framework of* ***granularity-based*** *and* ***individual-based*** *data partitioning approaches in HaplotypeCaller. N=715 for* ***granularity-based*** *and N=50 for* ***individual-based****. The reference file and each sorted and indexed BAM file were sent to a worker for executing GATK's HaplotypeCaller. Outputs of HaplotypeCaller in VCF format were joined to create a single output file.*

In the next phase of the pipeline, we tested three ways of partitioning mapped data for variant calling using HaplotypeCaller.

- For **granularity-based partitioning**, similar to BWA, we split the aligned data for HaplotypeCaller on the basis of a determined optimal granularity value.
- For **individual-based partitioning**, we split the aligned data based on individual names, resulting in 50 smaller files.
- For the new approach, called **alignment-based partitioning**, we partition the SAM file based on alignment information. We first split the reference file into multiple smaller files, each having a fixed number of unique contigs. For each small reference file, we split the pooled SAM file such that the smaller SAM file would contain alignment information of reads corresponding to the contigs of that particular reference subset.

Fig. 2.3 depicts the workflow for executing parallelized HaplotypeCaller using **granularity-based** and **individual-based** approaches. Fig. 2.4 shows **alignment-based** data partitioning implemented in HaplotypeCaller. For each approach, we executed HaplotypeCaller for a pair of smaller sorted BAM file and the reference sequence on a work queue worker. The output of HaplotypeCaller contained variant information in VCF format [5]. In the final step of the pipeline, we concatenated individual VCF files into a single VCF file containing variants for the entire population, with which we began the analysis. As we will discuss in Section 3, **granularity-based** and **alignment-based** data partitioning were the most efficient approaches for running the parallelized pipeline for BWA and HaplotypeCaller, respectively.

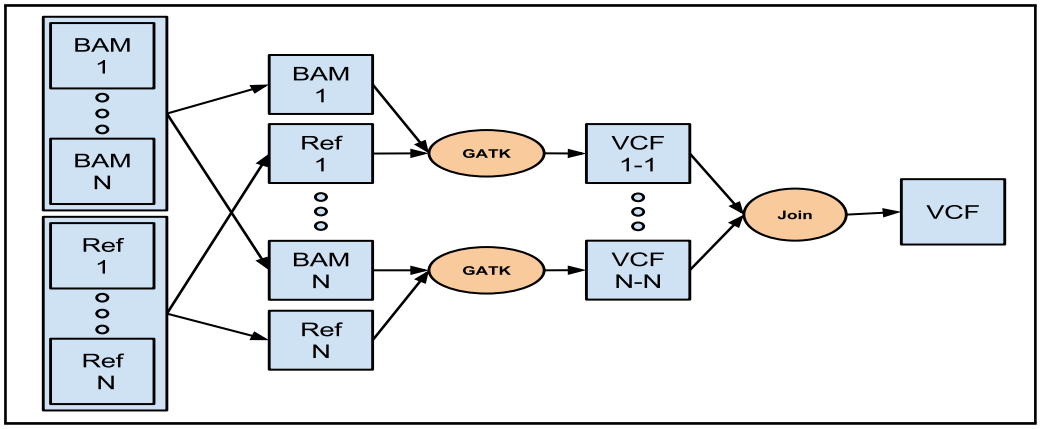


FIG. 2.4. *Framework of* ***alignment-based*** *data partitioning approach in HaplotypeCaller. The reference file was split into bins and the SAM file was split based on the contigs in the bins to which the reads aligned. Each pair of smaller reference bin and its corresponding BAM file were then sent to a worker to run GATK's HaplotypeCaller.*

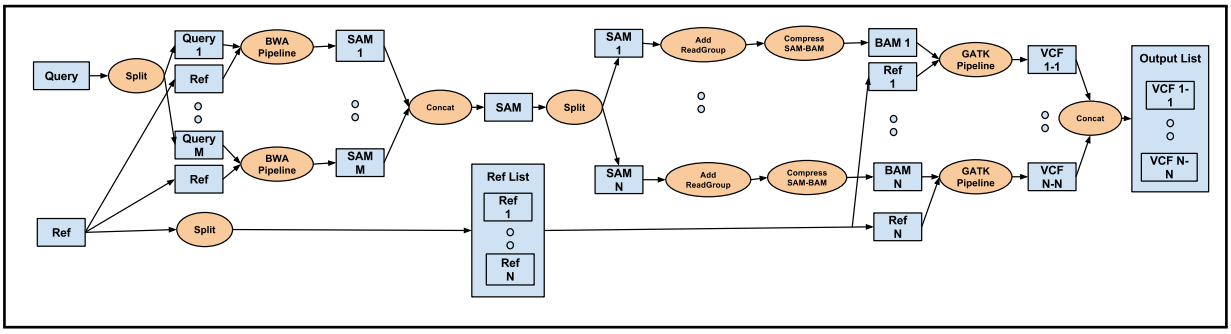


FIG. 2.5. *Framework of the optimized pipeline incorporating* ***granularity-based*** *BWA and* ***alignment-based*** *HaplotypeCaller. It also includes the intermediate stages of adding read groups to SAM files and converting them to their sorted and binary formats.*

2.4. Makeflow and Work Queue. Makeflow is a workflow engine that allows for the expression of static DAG (Directed Acyclic Graphs) workflows. Makeflow uses a straight-forward syntax for describing each task within a workflow. This syntax requires a list of all input files, a list of all output files, and the command to run on these input files. Makeflow requires that all files, both explicit in the command and implicit requirements, are listed. Requiring all files to be mentioned allows the task to be sent to any platform that supports the executable. When the makeflow is executed, only tasks whose required input files exist are sent for execution.

Work Queue is a master-worker job execution engine that is used for this project. With the Makeflow that describes the workflow, execution is started by creating and broadcasting a Work Queue master. This master then waits for worker and distributes jobs as available to keep the workers busy. Workers are created either at a local machine, or by submitting multiple workers at once to a batch system such as Condor or SGE. Once these workers connect, jobs are sent and the files required are transported and cached. This strategy is key as it allows for a file to be sent only once to each worker and then reused for every subsequent job sent to it.

Within a single workflow, the benefits of file reuse are evident, but as soon as the workflow ends the caches are cleared for security and efficient use, as leaving files on remote systems is to be avoided. Therefore, to utilize the cache the workflow must be fused into one workflow, where caches are used consistently within a single workflow.

3. Workflow Fusion. We define workflow fusion as the idea of using information about the software and the computing systems, which are involved in the execution of a workflow, to accelerate a set of sequential but interrelated workflows. These cross workflow optimizations fuse the previously independent workflows into

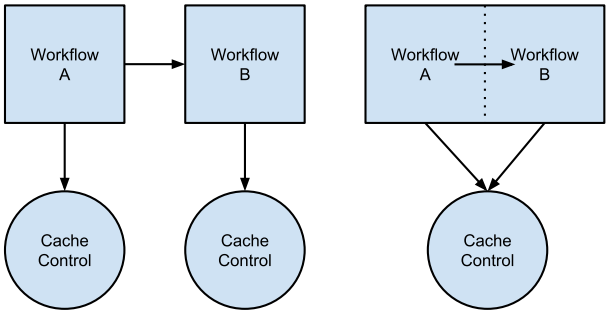


FIG. 3.1. *The figure illustrates how two merged workflows, A and B, utilize the same cache controller. In practice this is done using Work Queue, where all files in a Makeflow are aggressively cached. When separate, the two workflows' caches are independent and can not be used between each other. However, when the control of that cache is shared, previously transferred files can be utilized.*

one. Though this causes less flexibility in how the workflows are used, the improvements can be worthwhile. As we look at ways in which workflow fusion can benefit performance, it can be noted that different methods require different levels of knowledge of the workflows and difficulty in the depth of fusion. We explored full workflow caching, which was done automatically by concatenating makeflow, choke elimination, which required understanding how two workflows can be efficiently partitioned, and full fusion, in which we manually merged two divergent partitioning techniques for acceleration.

3.1. Full Workflow Caching. A basic approach of workflow fusion is full workflow caching. This method relies on a single manager of tasks and data that allows for better data management. Although Work Queue as a system aggressively caches all files that are sent to a worker, no caching occurs between independent workflows. By fusing the workflows, we take full advantage of previously sent data that are useful for multiple steps such as reference information. Fig. 3.1 shows the organization of the cache controller, and how merging them unifies the controller. This is the most basic form of workflow fusion, as all that is needed is to merge two makeflow files together and check that the references and input and output files are consistent to maintain the logical flow of the newly created workflow.

In this form of workflow fusion, we rely on the workflow execution engine for the performance improvement. This method is the easiest to implement, and its benefits can be seen at all levels of workflow fusion, as illustrated in Fig. 3.2. This benefit however, is negated if files are modified between workflows. For example, the base reference is split during the alignment-based partitioning for GATK, and is no longer recognized by Work Queue as a cached file. Originally, the **alignment-based** GATK benefitted by limiting the data that needed to be transferred [32], but with full workflow caching each new smaller piece of the reference is sent and the benefit of the cached reference is lost. Thus, combining two workflows presents challenges when the previously used partitioning scheme may no longer be the optimal solution of the fused workflow.

3.2. Elimination of Choke Points. A choke point is simply a location in the workflow that requires all threads of computation to converge. Similar to a barrier operation in traditional parallel computing, choke points hinder computation by stopping all concurrency until the slowest serial task completes. The effort of eliminating these choke points is the second method of workflow fusion. When eliminating choke points, an effort is made to alleviate traffic where data converges on a single node and is redistributed in some form. Although caching can limit some of the outgoing traffic from later partitioning steps, there is still a lot traffic that is incurred sending files in and out of these choke points.

Because choke point elimination strives to take advantage of files already distributed, it is important to understand how to convert the results of the previous workflow into the inputs of the next workflow. Such transformations are crucial, since they can be applied directly to the data. A clear example of a transformation is the sorting of aligned results prior to starting variant calling. Prior to fusion, all of these processes were sequentially completed, but performing a transform as part of the prior step allows avoiding expensive traffic

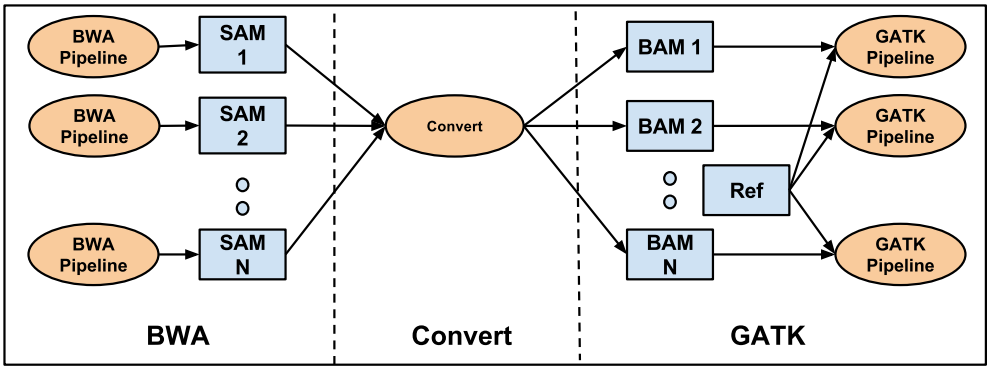


FIG. 3.2. *Framework of the Cache Fused Concept. The pipeline comprises the BWA step, intermediate conversion steps for adding read groups, converting SAM files to their sorted and indexed formats, and the GATK step. The reference used by GATK is the same as that used in BWA, allowing for it to be cached at the worker and not sent as additional traffic.*

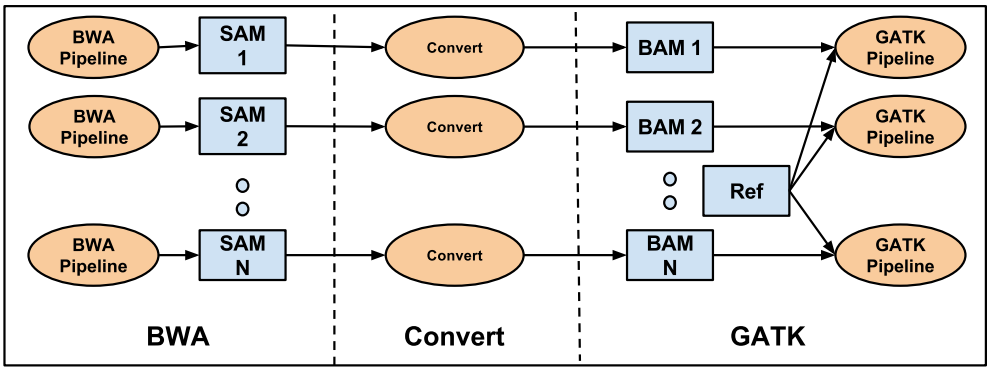


FIG. 3.3. *Framework of the Choke Elimination Concept. As can be seen here, the removal of intermediate data choke points allows computational threads to progress further through the workflow without needing to wait for slower threads to finish and communicate with the master. This lowers the amount of data the master is required to deliver at once and allows it the transfers to be offset based on when they arrive.*

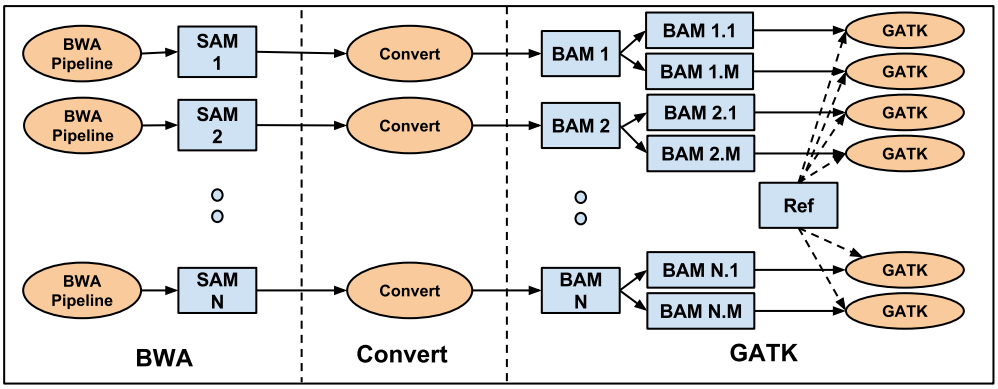


FIG. 3.4. *Framework of the Merged Partition Concept. As can be seen here, the conversion step appears prior to the mapping, but could go either before or after depending on what state the inputs need to be. It is also useful to note how, though as simple map is used here, any process that relates two partitioning methods could be added as long as it does not require coalescing the full data set.*

to converge the data on one system.

This is done most clearly by utilizing the partitions of the previous workflow as the partitions of the subsequent workflow, as shown in Fig. 3.3. Using established partitioning eliminates intermediate partition transfers, and allows independent partitions to continue executing without waiting for slower branches. The cost of this approach is that both workflows must be able to utilize this partition with acceptable performance. If the two different workflows do not share a common suitable partitioning scheme, we can utilize the information about the workflows to potentially find a more appropriate hybrid method.

3.3. Merged Partitioning. Finding a hybrid partitioning scheme requires knowledge about how different workflows interact and therefore how they could be optimally partitioned. Unfortunately, it is not always obvious how to do such divisions. For example, alignment-based partitioning is best for GATK but this requires performing alignments using BWA. One compromise is to maintain BWA partitions utilizing **individual-based** partitioing in BWA, and then partioning each individual utilizing the **alignment-based**. Although this is a basic method to merge partitioning schemes, the idea can be extend to any method that relates one set of inputs to another, as shown in Fig. 3.4.

When successful, applying these ideas to a set of workflows alleviates the time and stress on the system, while allowing faster machines to process more without having to wait and proceed in lock step with remaining tasks.

3.4. Issues that may arise with workflow fusion. Though performance is often improved by melding multiple workflows together, there are several pitfalls. The first is that as the workflows become more intertwined, debugging becomes more difficult to monitor, understand, and fix. Undetected failures in different states produce incorrect output, which could have occurred at any of the fused processes. This issue can be exacerbated when workflows operate at different stages concurrently. For this reason a clear understanding of what errors may occur within the different processes and how to handle them is paramount. The second pitfall is some methods of partitioning, or combinations of partitioning methods, may not accelerate the workflow or be feasible due to dependencies (e.g., alignment-based GATK drawn from BWA output). When this occurs more time must be spent understanding and writing appropriate transformations.

4. Results. For our experiment, we aligned and detected variants for 50 *Quercus rubra* (northern red oak) individuals' ILLUMINA HiSeq RAD [30] data with over 100-fold coverage. The reference sequence for this non-model species contained approximately 400000 individual loci. The system used for analysis consisted of a cluster of 26 machines, each with 8 cores and 32 GB RAM. This allowed us to use up to 208 (26×8) workers in the cluster. To illustrate our base run time, BWA (single end) required about 4 hours to complete the alignment procedure sequentially, and GATK's HaplotypeCaller required 12 days to detect SNPs and indels. Run times for BWA does not include the time for indexing the reference, which can be done once and re-used for each alignment task. Next we will discuss different techniques explored for partitioning data to find an optimal solution.

4.1. Partition. Prior to running the experiments, it was important to understand the impact the selected partitioning method would have on the overall runtime.

The performance of the algorithm with each partitioning method is important, but if several methods perform similarly within a process the deciding factor is the impact of the method used on the data organization. As such, we developed several parsing scripts in Perl that allowed us to partition the data using a common language for comparison. To find an accurate time for each, the scripts were run 5 times with the reported run time being the average of those runs.

First we explored the two methods that were discussed for BWA. Granularity-based partitioning performed well, by only requiring 13:23 minutes to parse the entire data set into 715 partitions. This was expected, as the partitioning done here was superficial and only relied on the number of reads that were to be placed in each file. The second method, **individual-based** was expected to be considerably slower as it needed to map its barcodes to individual names for subsequent splitting. However, **individual-based** partitioning required only 18:39 minutes to create 50 partitions. A comparison of results can be seen in Table 4.1.

Next, the different partitioning methods that were used in GATK were explored. Granularity-based approach required 4 hours 52 minutes to divide the aligned data from BWA up to 2000 partitions. **Individual-**

TABLE 4.1

Comparison of runtimes for different approaches of data partitioning in BWA. Due to lower splitting time, ***granularity-based partitioning*** *was the best approach for BWA. However, the two approaches' similar times make them good candidates for the pipeline.*

BWA Runtime (DD:HH:MM:SS)				
Partitioning	**Split**	**Align**	**Concat**	**Total**
Individual	18:39	17:58	22:26	59:23
Granular	13:23	21:11	21:10	**55:44**

TABLE 4.2

Granularity-based *BWA requires less time when more workers are in use. For the test data set (Data 1), due to the overhead of additional data transfer, the system could not scale well beyond 20 workers. This should not be considered as a shortcoming of the framework, which achieved higher speedup and efficiency with more workers when tested on a larger query data set (Data 2) comprising 60 individuals.*

Runtime (HH:MM:SS)						
Workers	**Data 1**			**Data 2**		
	Time	**Speedup**	**Efficiency**	**Time**	**Speedup**	**Efficiency**
2	3:01:57	1.94	0.97	8:46:13	1.80	0.90
5	1:14:30	4.77	0.95	3:33:45	4.45	0.89
10	1:08:42	5.19	0.51	1:48:10	8.78	0.87
20	59:00	5.98	0.29	55:34	17.23	0.86
50	57:06	6.19	0.12	33:17	28.27	0.57
100	56:00	6.30	0.06	20:52	47.41	0.47

based required 27:35 minutes and produced 50 partitions, for 50 individuals. The following partitioning approach was based on **alignment-based partitioning** that was discussed earlier. **Alignment-based** splits the reference and entire SAM query file into 10 partitions based on alignment information. This method required 15:41 minutes to perform. A comparison of results can be seen in Table 4.3.

4.2. Tool Improvement. Once a splitting script was created implementing each of the above-mentioned partitioning schemes, BWA and HaplotypeCaller were executed using 100 workers. Table 4.1 presents the runtimes for BWA using different data partitioning approaches. For BWA, the alignment and concatenation stages of **granularity-based** and **individual-based** approaches required similar time. The step of splitting the query in **individual-based partitioning** was a bottleneck due to the higher look up time to match an individual's name from barcode information. The **granularity-based** approach did not involve this searching step and hence was more efficient (Table 4.1: shown in bold font). For HaplotypeCaller, preparation of input comprised the times for splitting the concatenated SAM file based on each approach, adding read group information, and converting SAM to sorted and indexed BAM formats. Although there was an additional cost in splitting the SAM file for **alignment-based partitioning**, the overall runtime was significantly lower (Table 4.3: shown in bold font) than the other two approaches.

After inferring **granularity-based** approach to be the best for BWA, we tested its run time behavior from 2 workers up to 100 workers. Fig. 4.1 shows how the framework performed with increased number of workers. As more workers were used, the increase in performance diminished rapidly. For the test data set and machine configuration, we achieved the best performance with 100 workers, after which adding more workers did not improve run time. However, when compared to 20 workers, the 50 and 100 worker iterations had only a minor performance improvement, showing the system did not scale linearly. This was due to the additional time incurred in data transfer, particularly in sending all the query files, reference and index files to each worker, which caused an overhead as we added more workers. Fig. 4.1 supports this fact as we can see the amount of data sent and the time required for it increased with more workers. Fig. 4.3 presents a histogram for running 715 tasks in each stage (bwa aln and samse) of BWA. It gives an overview of the overall behavior of **granularity-based** BWA tasks when allowed to use 100 workers. Similarly, Fig. 4.4 provides a visualization of the run

TABLE 4.3

Comparison of runtimes for different approaches of data partitioning using GATK's HaplotypeCaller. Due to overall shorter coupling time ***alignment-based partitioning*** *was the most optimal choice. The HaplotypeCaller phase was also fastest due to reduced search space by limiting the number of loci in the reference at each partition. The quick HaplotypeCaller performance for granularity, similarly, comes from limited loci available during variant calling.*

GATK Runtime (DDd HH:MM:SS)					
Partitioning	**Coupling**		**Variant Calling**	**Concat**	**Total**
	Split	**Other**			
Individual	27:35	3:59:10	2d 5:18:23	9:00	2d 9:54:08
Granular	4:52:34	7:06:25	41:51	5:30	12:46:20
Alignment	15:41	1:03:47	24:36	4:00	**1:48:04**

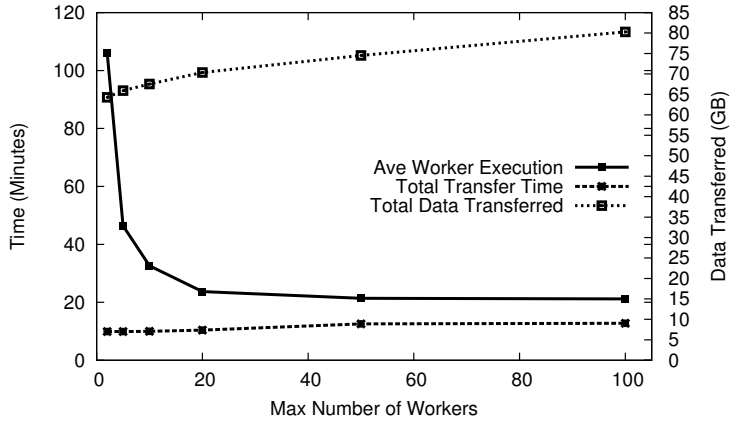


FIG. 4.1. *Runtime of* ***granularity-based*** *BWA for increasing number of workers. The time accounts for the parallelized alignment step of BWA, and does not consider the times for splitting the query sequence or concatenating the outputs. The run time gradually decreases with more workers. For large number of workers the overhead of data transfer hinders further improvement.*

time behavior of the same approach for 100 workers. The number of tasks submitted and completed followed a similar pattern until the former reached a plateau denoting submission of all tasks while there were some tasks yet to be complete. The number of tasks running followed a steady line due to the use of a dedicated cluster. We also tested the framework for another data set (Data 2 in Table 4.2) having a larger unique query data. Fig. 4.2 shows the transfer of necessary files from the master to the workers over the network. This feature is particularly useful in an environment without a shared file system, as in our case. The transferred shared data can be cached at the workers to be re-used by the workers.

As seen in Table 4.3, for HaplotypeCaller, the **alignment-based** approach was most efficient. While splitting the SAM file into smaller SAM files, based on the contigs to which the reads aligned, we optimized the method of looking up the reference sequence to detect variants. Instead of searching the entire reference file, we searched through a small section of the reference sequence that was guaranteed to contain information for all the alignments in the small SAM file. During splitting of the SAM file we also ignored the entries that did not align to contigs, reducing the number of elements to process. The other two approaches did not have the scope of reducing the search space during variant detection. Also, the overhead of sending the entire reference file to a worker for each task was responsible for increased run times in **individual-based** and **granularity-based** approaches of HaplotypeCaller.

4.3. Pipeline Improvement. Based on the aforementioned results, we generated an optimized pipeline for genome alignment and variant calling combining **granularity-based** BWA and **alignment-based** HaplotypeCaller, as shown in Fig. 2.5. Fig. 4.3 presents the average time taken by a task running BWA and Haplotyper. This time represents the generalized functioning of the pipeline and is independent of the underly-

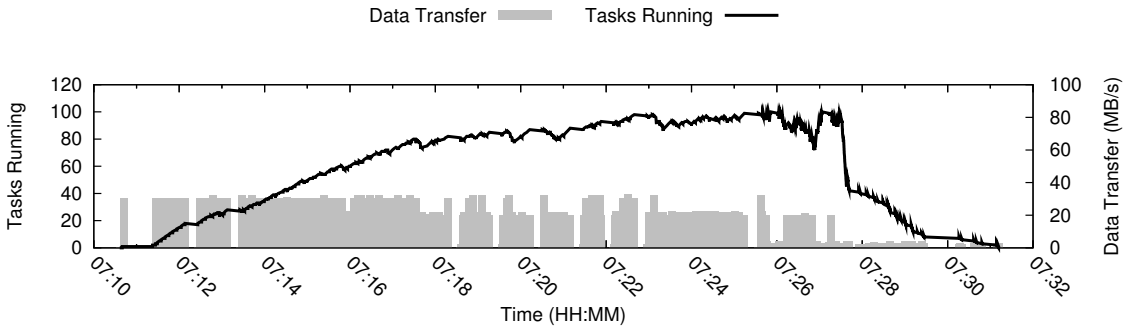


FIG. 4.2. *Mechanism of data transfer for an environment without a shared file system. The thick line denotes the number of tasks executing during the given timeline for* ***granularity-based*** *BWA. The gray bars show data transfer from the master to worker nodes. The left axis measures the number of tasks running whereas the right axis measures the rate of data transfer in MB/s.*

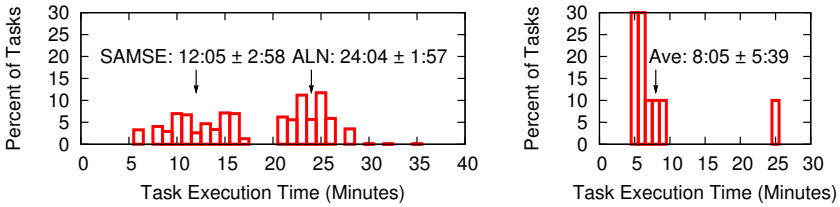


FIG. 4.3. *The histogram for BWA (left) is bimodal due to the alignment (aln) and generation of reads in SAM format (samse) steps. These times were measured for 1430 tasks, where 715 tasks were attributed to each process.The histogram for GATK (right) only contains 10 tasks, which are tightly grouped, except for a single heavy outlier.*

ing batch system. On the other hand, Table 4.4 provides the runtimes specific to our pipeline which employed 100 work queue workers in a heterogeneous, distributed system. It shows the breakdown of run times for individual components of the pipeline. The efficient data partitioning followed by parallelization of tasks reduced the runtimes of each phase, resulting in an overall runtime improvement from approximately 12 days to under 3 hours.

4.4. Workflow Fusion. Table 4.5 compares the times when running the individual tools sequentially to that when run in our optimized pipeline as well as the workflow fusion-based pipeline.

The first method that we performed was a full cache fused workflow. This was done by concatenating the Makeflow files for each workflow together. Following this, all intermediate conversion processes were added, and the fused makeflow was run utilizing Work Queue as before. We were able to acheive a result of 2 hours and 42 minutes, as shown in Table 4.5. This is not surprising as it is close to the seperate disjoint pieces of the workflow, and used **individual-based** BWA followed by **alignment-based** HaplotypeCaller. This result verifies concept, and does not hinder the performance of the workflow.

The second method implemented was the choke point elimination by matching partitioning types. For this we utilized the makeflows written using **individual-based partitioning** for both BWA and GATK. For the intermediate processes, the concatenation of BWA and splitting of GATK were ommitted and the intermediate steps of adding read groups and converting to sorted and indexed binary formats were applied to each individual. This method completed in almost 2 and a half days,(Table 4.5) which is on the same scale as the BWA and GATK seperate makeflow using **individual-based partitioning**. **Individual-based paritioning** was used, as **granularity-based** is conceptually the same from BWA to GATK, but split into different partitions due to the different organization of FASTQ and SAM files. This implementation improved the coupling of the workflow by eliminating the join-split sequence used in the pipeline and cache fused. However, the results are much slower, due to GATK's performance on this partition method. The intermediate steps while faster, are dominated by GATK's slower perfomance using **individual-based partitioning**.

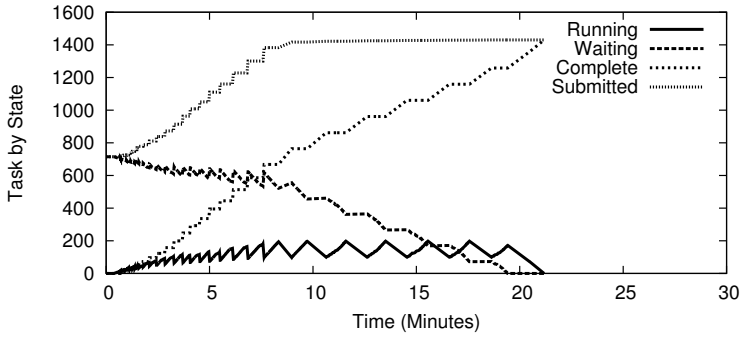


FIG. 4.4. *Run time behavior of* ***granularity-based*** *BWA using 100 workers. The lowest line represents the currently running tasks, the jagged appearance is due to the manner in which tasks are distributed using Work Queue. While distributing tasks, finished tasks wait to be collected, they are then collected and tasks are sent out. This waiting causes the jaggedness.*

TABLE 4.4

Run times of individual steps of the optimized pipeline using 100 workers. For BWA, the query data is split based on ***granularity-based partitioning****. The outputs of BWA, in SAM format, are then concatenated. This is followed by the coupling stage which includes splitting the reference and concatenated SAM file based on* ***alignment-based partitioning****, adding read groups (RGs), and converting SAMs to their sorted, indexed binary versions (BAMs). GATK's HaplotypeCaller runs on the BAM files and reference bins to generate VCF files, which are finally concatenated to contain variant information for the entire dataset. It is important to note that conversion from SAM to BAM is the most expensive step. We address this problem using Workflow Fusion.*

Runtime (MM:SS)		
BWA	**Split Query**	13:23
	Parallelized BWA	21:11
	Concat SAM	21:10
Coupling	**Split Ref and SAM**	15:41
	Add RGs	11:10
	SAM to BAM	52:37
GATK	**Parallelized GATK**	24:36
	Concat VCF	4:00
Total		2:43:48

The final method implemented was the full partition merging of BWA and GATK. Beginning with the makeflow for **individual-based** BWA, the concatenation step was removed, and each individual was then split using the alignment-based method. Following the splitting for **alignment-based**, each new split had the intermediate processes performed. This added a significant amount of tasks, with only a small amount of additional overhead, as the inputs were smaller. This method was able to complete in 1 hour and 55 minutes (Table 4.5). This was an improvement over the best results we were able to obtain utilizing the seperate makeflows, and was a result of caching and higher number of available tasks. Specifically, nodes that took longer now did not slow down the workflow. This higher level of granularity and better partitioning scheme selection allowed for runtime improvement. The performance of the entire pipeline was improved by increasing the concurrency and allowing coupling steps to be completed more quickly, as seen in Fig. 4.5. In Fig. 4.6, the significant differences between the coupling steps of cache fused and partition fused are evident.

The interesting information that arises when using different workflow fusion techniques is the intermediate steps. The three different methods that were outlined, do not necessarily apply well to all workflows. This current workflow is a perfect example, because the overall runtime for choke elimination is considerably slower than the pipeline described without fusion, the intermediate steps were faster. This can be seen in Table 4.6. It is also interesting to note that additional splitting step for partition fused did not slow down the entire execution more than it increased performance. However, when choke elimination was applied in conjuction with partition

TABLE 4.5

Comparison of run times for sequential, parallel, and workflow fusion execution of the pipeline. CF stands for Cache Fused, CE stands for Choke Elimination, and PF stands for Partition Fused. The run time for the parallelized execution represents the times taken by the optimized pipeline for 100 workers. The intermediate coupling column describes the time spent transforming data from BWA for GATK. The fused workflows were run using 100 workers, as can be seen by the conversion times workflow fusion mitigates conversion and spends more time in computation.

Runtime (HH:MM:SS)				
	BWA	**Coupling**	**GATK**	**Total**
Sequential	4:04:00	6:50:41	12d 00:00:00	12d10:54:41
Parallel	55:44	1:23:28	24:36	2:43:48
CF	2:42:49			2:42:49
CF + CE	2d 8:43:46			2d 8:43:46
CF + CE + PF	1:55:37			1:55:37

TABLE 4.6

Comparison of time spent in the coupling stages between BWA and GATK using different levels of workflow fusion. It is interesting to note that time spent in partition fused splitting the data further increased the remaining intermediate steps enough to mitigate the cost.

Coupling Runtime (HH:MM:SS)				
	Concat	**Split**	**Convert**	**Total**
CF	9:11	15:41	1:42:09	2:07:01
CF + CE	-	-	1:07:00	1:07:00
CF + CE + PF	-	18:52	39:36	58:28

fusion the resultant workflow performed well and aided the effect of partition fusion.

While applying these different optimization methods, we can clearly see that these methods can be applied in a cumulative manner to the workflow. In fact, the three methods that were outlined have a hierarchy, with the base method of cache fusion is built on by choke elimination, and choke elimination is used with partition fusion. This set of methods is a preliminary look at methods of workflow fusion, and could likely be expanded into different operations that could be either cumulative, as the methods presented, or mutually exclusive from one or more methods due to workflow structure. Exploring workflow fusion with other workflow pipelines would facilitate better understanding of workflow fusion methods.

Using this set of experiments, we were able to outline and show a set of operations for workflow fusion and how they applied to a genomic pipeline. We were able to achieve a 40% improvement in performance from the base pipeline. This improvement warrants further exploration of workflow fusion, and understanding how different operations used in fusion affect the performance of a process. Though this concept was applied to a genomic pipeline in this paper, the concepts discussed are applicable to different sectors of research.

5. Open Problems. Understanding the extent of parallelism: One of the most important factors while developing a parallel application is understanding if the underlying algorithm exhibits potential for parallelism. In each step of this pipeline, individual tasks were independent, without one task having to wait for the completion of another, but the data decomposition was critical. In this context, the commands written in the Makeflow script represented the order of execution of tasks as a directed acyclic graph (DAG) [11] but additional work was needed to do it in the best manner. As shown in our results, adding more workers does not guarantee reduced run time. Resource contention, lack of sufficient tasks to exploit the availability of all workers, few tasks with abnormally long run times, network traffic, and higher data transfer times often cause an overhead, especially in such data-intensive parallel applications.

Using a heterogeneous environment without shared file system: The procedure of data transfer while executing tasks in a distributed environment becomes challenging in the absence of a shared file system. The use of work queue workers mitigates this problem as they solely depend on the files transferred by the masters (Fig. 4.2), as specified in the makeflow rules. For even larger data-intensive applications that require various clusters to scale up their performance, a hierarchical implementation of work queue comprising

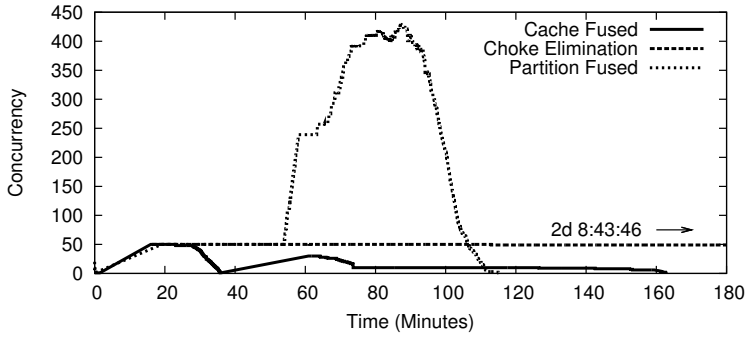


FIG. 4.5. *Comparison of available concurrency for Partition Fused, Choke Elimination, and Cache Fused workflows. Available concurrency consists of all tasks that are ready to be run and either waiting or running. As can be seen by the significant size of the Partition Fused line, the increase in partition promotes a significantly higher level of concurrency. Also, though Choke Elimination had more available concurrency, the partitioning method used caused slower performance.*

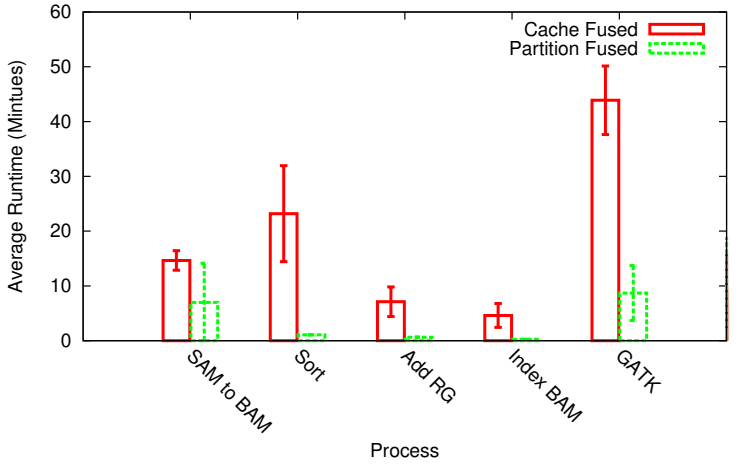


FIG. 4.6. *Comparison of average runtimes for the set of coupling processes in cache fused and partition fused workflows. Clearly, the time spent both in intermediate steps as well as GATK itself was longer per node in cache fused than in partition fused. It is important to note that the number of partitions was 10 for cache fused and 500 for partition fused.*

a multi-slot worker and a foreman [1] can reduce the overhead of data transfer, causing an improvement in performance. For our application, because the shared data (reference file) was comparatively smaller than unique data (sequence/BAM), we did not benefit from these features, but others may.

Determining the granularity value: One of the important challenges in developing parallel programs is determining the optimal granularity to balance the times spent on communication, particularly the overhead due to file transfer, and computation. Earlier studies [31] have shown that a significant improvement in performance can be achieved if the number of tasks and size of each task can be optimized. This trade off is based on the underlying algorithm as well as the configuration of the machine on which the program runs. For a heterogeneous system like the Condor pool, care should be taken while determining the optimal value of granularity; however we have shown our framework is consistent under load and will use the maximum parallelism available.

6. Related Work. Genome mapping and assembly have traditionally been refa- ctored using MPI (Message Passing Interface) [6] or MapReduce [7] frameworks. ClustalW-MPI [23] is a version of the multiple sequence aligner ClustalW [37] that uses MPI to run on distributed memory systems. CloudBurst [36] is an early application that implements the seed-and-extend genome mapping algorithm on a Hadoop [10] based MapReduce framework. Although CloudBurst can align millions of reads in a reasonable time, it takes longer

to process billions of reads when compared to a multi-core version of another cloud-based variant calling tool Crossbow [16]. Further, CloudBurst identifies all possible alignments in a read, which is often not a requirement in current comparative genomics projects. Crossbow employs Hadoop to parallelize Bowtie for alignment and SOAPsnp [25] for variant calling. It does not natively support gapped alignment. SEAL [34] is another scalable aligner based on BWA and Picard that uses the MapReduce framework for parallelization of tasks.

Although the MPI framework allows execution of parallel programs, it requires complicated software development for refactoring tools. With Hadoop's MapReduce-based parallelization, it is not easy to tune the parameters for granularity, as was required in our experiments. Hadoop implements its own method of mapping tasks and reducing them on completion. Also, our application was capable of dynamically scaling up or down the workers as needed, which is difficult to implement in Hadoop's MapReduce-based framework. Finally, unlike Hadoop, our developed framework could harness resources from a heterogeneous system. Our system supported these features by implementing the Makeflow language and Work Queue master-worker framework to generate a parallelized workflow for comparative genomics applications. In this paper, we also analyzed the effect of different data partitioning approaches on the runtime of mapping and variant calling tools. Although we considered BWA and HaplotypeCaller as test cases, the approaches of data partitioning presented in this paper are generic and can be implemented to improve the performance of other bioinformatics tools. Particularly, the tools whose underlying algorithm or data structure is similar to our test tools, can potentially benefit from our proposed approach of optimized data partitioning. We observed that our optimized pipeline could further benefit from workflow fusion which merges diverse sequential workflows for optimization. We explained the impact of caching, eliminating choke points, and stacked partitioning methods on performance while implementing workflow fusion. Again, this idea is highly applicable in developing parallelized workflows for any data-intensive application. Moreover, the discussion on the possible barriers and the solutions to overcome them is useful for adapting parallelized workflows on ad hoc clouds. This in turn can aid the development of efficient bioinformatics applications that can harness resources from a heterogeneous, distributed environment.

7. Conclusion. We significantly improved the performance of a variant discovery bioinformatics pipeline by developing a new workflow to run parallely on a heterogeneous, distributed system. We identified that one of the important factors to enable efficient parallelism is optimized partitioning of data. In this regard, we discussed different techniques of data decomposition, namely **granularity-based**, **individual-based**, and **alignment-based partitioning**. We considered BWA for genome alignment and GATK's HaplotypeCaller for SNP and indel calling as our test tools. We tested both the tools for all the approaches of data partitioning and concluded that **granularity-based partitioning** was the best approach for BWA. Unlike **individual-based partitioning**, this method did not require the overhead of comparing each read in the query with the barcode information to find a matching individual name. For HaplotypeCaller, **alignment-based** approach was proved to be the most efficient technique. By splitting the reference file and the SAM file accordingly, it reduced the search space to a considerable extent, which in turn lowered the runtime from days to hours. After selecting the optimized data partitioning approach for each tool, we combined them into a pipeline framework that could efficiently analyze NGS data in a substantially reduced time. For the test dataset comprising genomic data of 50 oak individuals, our optimized pipeline required approximately 4 hours when run parallely with optimized number of workers, in this case 100 workers, as opposed to 12 days, when the tools were run sequentially. Following our exploration of the pipeline, we applied the concepts discussed in workflow fusion to our optimized workflow. After applying all of the concepts, we were able to achieve an execution time of just under 2 hours. We further discussed the potential challenges that are likely to be encountered while parallelizing such data-intensive applications and proposed a solution to each.

Acknowledgment. The authors would like to thank Jeanne Romero-Severson for providing the Illumina data used to test our framework. This work was supported in part by National Institutes of Health/National Institute for Allergy and Infectious Diseases (grant number HHSN272200900039C) to SJE and National Science Foundation grant SI2-SSE (number 1148330) to DT.

REFERENCES

[1] M. Albrecht, D. Rajan, and D. Thain, *Making work queue cluster-friendly for data intensive scientific applications*, in Cluster Computing (CLUSTER), 2013 IEEE International Conference on, IEEE, 2013, pp. 1–8.

[2] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, and D. Thain, *Work queue+ python: A framework for scalable scientific ensemble applications*, in Workshop on Python for High Performance and Scientific Computing at SC11, 2011.

[3] M. Burrows and D. Wheeler, *A block-sorting lossless data compression algorithm*, (1994).

[4] P. J. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, *The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants*, Nucleic acids research, 38 (2010), pp. 1767–1771.

[5] O. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, et al., *The variant call format and vcftools*, Bioinformatics, 27 (2011), pp. 2156–2158.

[6] A. Darling, L. Carey, and W.-C. Feng, *The design, implementation, and evaluation of mpiBLAST*, Proceedings of ClusterWorld, 2003 (2003).

[7] J. Dean and S. Ghemawat, *Mapreduce: simplified data processing on large clusters*, Communications of the ACM, 51 (2008), pp. 107–113.

[8] M. DePristo, E. Banks, R. Poplin, K. Garimella, J. Maguire, C. Hartl, A. Philippakis, G. del Angel, M. Rivas, M. Hanna, et al., *A framework for variation discovery and genotyping using next-generation DNA sequencing data*, Nature genetics, 43 (2011), pp. 491–498.

[9] W. Gentzsch, *Sun grid engine: Towards creating a compute power grid*, in Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, IEEE, 2001, pp. 35–36.

[10] A. Hadoop, *Hadoop*, 2009.

[11] S. Handley, *On the use of a directed acyclic graph to represent a population of computer programs*, in Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, IEEE, 1994, pp. 154–159.

[12] O. Harismendy, P. C. Ng, R. L. Strausberg, X. Wang, T. B. Stockwell, K. Y. Beeson, N. J. Schork, S. S. Murray, E. J. Topol, S. Levy, et al., *Evaluation of next generation sequencing platforms for population targeted sequencing studies*, Genome Biol, 10 (2009), p. R32.

[13] D. C. Koboldt, K. Chen, T. Wylie, D. E. Larson, M. D. McLellan, E. R. Mardis, G. M. Weinstock, R. K. Wilson, and L. Ding, *Varscan: variant detection in massively parallel sequencing of individual and pooled samples*, Bioinformatics, 25 (2009), pp. 2283–2285.

[14] I. Lanc, P. Bui, D. Thain, and S. Emrich, *Adapting bioinformatics applications for heterogeneous systems: a case study*, Concurrency and Computation: Practice and Experience, (2012).

[15] B. Langmead and S. L. Salzberg, *Fast gapped-read alignment with bowtie 2*, Nature methods, 9 (2012), pp. 357–359.

[16] B. Langmead, M. Schatz, J. Lin, M. Pop, and S. Salzberg, *Searching for SNPs with cloud computing*, Genome Biol, 10 (2009), p. R134.

[17] B. Langmead, C. Trapnell, M. Pop, S. L. Salzberg, et al., *Ultrafast and memory-efficient alignment of short dna sequences to the human genome*, Genome Biol, 10 (2009), p. R25.

[18] T. J. Ley, E. R. Mardis, L. Ding, B. Fulton, M. D. McLellan, K. Chen, D. Dooling, B. H. Dunford-Shore, S. McGrath, M. Hickenbotham, et al., *Dna sequencing of a cytogenetically normal acute myeloid leukaemia genome*, Nature, 456 (2008), pp. 66–72.

[19] H. Li and R. Durbin, *Fast and accurate short read alignment with burrows–wheeler transform*, Bioinformatics, 25 (2009), pp. 1754–1760.

[20] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, et al., *The sequence alignment/map format and SAMtools*, Bioinformatics, 25 (2009), pp. 2078–2079.

[21] H. Li and N. Homer, *A survey of sequence alignment algorithms for next-generation sequencing*, Briefings in bioinformatics, 11 (2010), pp. 473–483.

[22] H. Li, J. Ruan, and R. Durbin, *Mapping short dna sequencing reads and calling variants using mapping quality scores*, Genome research, 18 (2008), pp. 1851–1858.

[23] K.-B. Li, *ClustalW-MPI: Clustalw analysis using distributed and parallel computing*, Bioinformatics, 19 (2003), pp. 1585–1586.

[24] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang, *SNP detection for massively parallel whole-genome resequencing*, Genome research, 19 (2009), pp. 1124–1132.

[25] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang, *Snp detection for massively parallel whole-genome resequencing*, Genome research, 19 (2009), pp. 1124–1132.

[26] R. Li, Y. Li, K. Kristiansen, and J. Wang, *Soap: short oligonucleotide alignment program*, Bioinformatics, 24 (2008), pp. 713–714.

[27] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang, *Soap2: an improved ultrafast tool for short read alignment*, Bioinformatics, 25 (2009), pp. 1966–1967.

[28] M. Litzkow, M. Livny, and M. Mutka, *Condor-a hunter of idle workstations*, in Distributed Computing Systems, 1988., 8th International Conference on, IEEE, 1988, pp. 104–111.

[29] M. Metzker, *Sequencing technologiesthe next generation*, Nature Reviews Genetics, 11 (2009), pp. 31–46.

[30] M. R. Miller, J. P. Dunham, A. Amores, W. A. Cresko, and E. A. Johnson, *Rapid and cost-effective polymorphism identification and genotyping using restriction site associated dna (rad) markers*, Genome Research, 17 (2007), pp. 240–248.

[31] C. Moretti, J. Bulosan, D. Thain, and P. J. Flynn, *All-pairs: An abstraction for data-intensive cloud computing*, in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, IEEE, 2008, pp. 1–11.

[32] O. Choudhury, N. Hazekamp, D. Thain, and S. Emrich, *Accelerating Comparative Genomics Workflows in a Distributed Environment with Optimized Data Partitioning*, C4Bio Workshop at CCGrid, 2014
[33] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song, *Genotype and snp calling from next-generation sequencing data*, Nature Reviews Genetics, 12 (2011), pp. 443–451.
[34] L. Pireddu, S. Leo, and G. Zanetti, *SEAL: a distributed short read mapping and duplicate removal tool*, Bioinformatics, 27 (2011), pp. 2159–2160.
[35] J. Reis-Filho, *Next-generation sequencing*, Breast Cancer Res, 11 (2009), p. S12.
[36] M. Schatz, *Cloudburst: highly sensitive read mapping with mapreduce*, Bioinformatics, 25 (2009), pp. 1363–1369.
[37] J. D. Thompson, D. G. Higgins, and T. J. Gibson, *Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*, Nucleic acids research, 22 (1994), pp. 4673–4680.
[38] L. Yu, C. Moretti, A. Thrasher, S. Emrich, K. Judd, and D. Thain, *Harnessing parallelism in multicore clusters with the all-pairs, wavefront, and makeflow abstractions*, Cluster Computing, 13 (2010), pp. 243–256.

Edited by: Jesus Carretero
Received: August 31, 2014
Accepted: December 1, 2014

DOI 10.12694/scpe.v16i1.1061
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 71–84. http://www.scpe.org

PERFORMANCE COMPARISON AND TUNING OF VIRTUAL MACHINES FOR SEQUENCE ALIGNMENT SOFTWARE

ZACHARY J. ESTRADA, FEI DENG, ZACHARY STEPHENS, CUONG PHAM,
ZBIGNIEW KALBARCZYK, AND RAVISHANKAR K. IYER*

Abstract. We explore the performance cost of virtualisation for the fast growing application domain of genomics. Traditionally, scientific applications have been considered too high-performance to pay the performance cost of virtualisation. However, as the demand for computing power for genomics is ever-increasing, the cloud can become an attractive way to meet the scaling challenge presented by Next-Generation Sequencing (NGS). We seek to explore the feasibility of running an NGS pipeline in a cloud, and in doing so consider two prevalent short-read sequence alignment programs, BWA and Novoalign. We executed those applications in three separate open-source system virtualisation solutions: the KVM hypervisor, the Xen para-virtualised hypervisor, and Linux Containers. We compare the runtime in each environment against the runtime of the same system without virtualisation and measure the relative performance of each hypervisor. We investigate and reduce as much as possible any overhead, presenting tuning suggestions for cloud implementers and users. Overall, we find that the overhead introduced by virtualisation can be reduced to low single-digit percentages, a cost we believe to be more than acceptable, especially given that two of the three solutions, Xen and Containers, exhibit near-zero overhead.

Key words: Sequence alignment, virtual machines, performance

AMS subject classifications. 68M14, 92D20, 68M20

1. Introduction. The booming throughput of Next-Generation Sequencing (NGS) platforms imposes a progressively larger burden on genomics computing infrastructures as that throughput can cause genomic data to be generated faster than they can be consumed. For smaller sequencing facilities without a dedicated IT staff, that growing requirement for data centre resources may move beyond the skill set of non-IT professionals. In such cases, the effort of managing and processing information could inhibit researchers and clinicians from making innovations in their own fields of expertise. Even for larger institutions, unpredictable demand and high ownership costs can make the traditional enterprise IT model undesirable. Cloud computing is becoming a natural solution to the problem of expanding computational needs, thanks to its low-cost, on-demand nature and offloaded management [27, 25, 6]. For the bioinformatician or clinician, cloud computing can offer a convenient, scalable alternative to traditional methods of managing computational resources. In particular, Infrastructure-as-a-Service (IaaS) cloud resources offer users nearly complete control over their computing environment, allowing them to install an operating system of their choosing and run their applications natively.

The key enabling technology for IaaS cloud computing is server virtualisation, in which one partitions a single computer's resources into a set of virtual machines (VMs). The effort to virtualise systems has mainly been fuelled by growing enterprise IT environments, where the primary benefits of virtualisation (isolation, flexibility and density) help these environments scale more efficiently while consolidating costs. Cloud computing further exploits those benefits at a larger scale and uses virtualisation to abstract computing resources as a utility model.

The benefits of virtualisation do not come without cost. Fundamentally, some overhead is expected from having an additional layer between the hardware and software. There are many approaches to system virtualisation, including instruction emulation, para-virtualisation and hardware virtualisation (further details are included in Sect. 2), offering a range of management features and performance costs to applications. Furthermore, those costs are affected by the configuration of each virtualisation technology, as well as the specific application being run. As will be demonstrated throughout this work, many aspects of a system's configuration can impact virtualisation performance overhead in subtle ways. Thus, understanding the trade-offs and differences between virtualisation technologies is important when one is deciding whether or not to move to the cloud, which cloud platform to choose, or how to manage a virtualised data centre.

In this paper, we investigate three of the most popular open-source system virtualisation technologies: Linux Kernel Virtual Machines (KVM) [15], Xen [3] and Linux Containers (LXC) [1]. While the kernel features that

*Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA, ({zestrad2, feideng2, zstephe2, pham9, kalbarcz, rkiyer}@illinois.edu)

enable LXC are not a Virtual Machine Manager (VMM) or hypervisor in the strictest sense, LXC is increasingly considered as a potential replacement for virtualisation, and for simplicity, we refer to all three technologies as *VMMs*. This article extends our previous work on measuring the performance of sequence alignment applications in a virtualised environment [7]. In that work, we conducted synthetic benchmarks and measured the performance overhead of virtualisation for the Burrows-Wheeler Aligner (BWA) and Novoalign sequence aligners. We found that VM performance was not consistent across applications and that VMs could have up to 20% overhead for our sample workloads. In this work, we describe the results of performing the same measurements on an additional class of hardware (a workstation computer with a different generation processor) and move beyond default configurations to determine the source of the previously observed performance overhead. Additionally, we tuned the VMMs to achieve the minimum possible overhead for the applications studied. We found that after proper tuning and setup, the overhead for each VMM can be reduced to a nearly insignificant value, with Xen and LXC exhibiting near-zero overhead and KVM showing 4% overhead for BWA and 2% for Novoalign. The results were demonstrated to be relatively stable across both machine types considered.

2. Background.

2.1. Genomics Workloads. The most resource and time intensive step in many Next-Generation Sequencing (NGS) pipelines is often the alignment step, where short reads are mapped to a reference sequence. In collaboration with a major genomics institution[1] we have studied a production variant calling pipeline and observed that alignment can take roughly 30% of the total execution time of the entire workflow. Therefore, when one is looking to reduce costs or gain mobility by moving tasks to a cloud, it is important to consider the effect of doing so on this critical component. It can be expected that various alignment programs may utilise computing resources in a similar way, so we run two production-grade alignment packages to test this hypothesis and derive conclusions that we intend to be representative of alignment programs in general. One of the programs chosen is the Burrows-Wheeler Aligner [17] (BWA) and the other is Novoalign [2]. We chose BWA as it is a popular open-source package and Novoalign as it is widely considered by many in the genomics community to be the most accurate alignment program available [16].

2.2. Kernel-based Virtual Machines. Hardware assisted virtualisation (HAV) technologies provided by different CPU vendors (e.g. VT-x from Intel or SVM from AMD) add extensions to the x86_64 instruction set to readily support Virtual Machines. The Linux Kernel Virtual Machine (KVM) hypervisor utilises HAV and serves as a good example of a hardware-assisted virtualisation platform.

In addition to being a demonstrative example of a VMM built using HAV, KVM is a valuable use case as it is a natural choice for Linux VMs since it is included with the upstream Linux kernel. Furthermore, KVM is widely considered the *de facto* standard for the open-source Infrastructure-as-a-Service (IaaS) cloud management platform OpenStack[2] and the virtualisation management platform libvirt.

2.3. Xen. Xen is considered a full Type I hypervisor, even the 'host' OS that has direct access to the hardware and manages the other VMs runs inside a special VM called 'Domain 0.' Xen helped popularise the concept of para-virtualisation, where a modified version of the operating system is run to avoid the unnecessary redundancy and overhead that comes with providing a full system (e.g., having to virtualise x86 instructions that are difficult to virtualise) to the guest OS [3].

Xen is an important use case because the most widely used public cloud IaaS platform, Amazon EC2, is built on top of a modified version of Xen. While the results presented here may not be representative of performance on EC2, Xen is still a popular hypervisor.

2.4. Linux Containers. Linux Containers (LXC) represent a different virtualisation paradigm that offers less isolation than the other two VMMs introduced, i.e. all containers share the same operating system kernel [26]. The advantage of sharing an OS kernel means that there is no device emulation and no duplicated effort of running an OS on top of another OS. This method is not virtualisation in the usual sense, but containers can be used for many of the same reasons one would use virtual machines. The isolation provided by LXC is stronger than usual OS process isolation, thanks to the use of `cgroups` [22] and kernel namespaces [14].

[1] The institution wishes to remain anonymous

[2] `http://openstack.org`

Kernel namespaces allow one to have separate domains for certain kernel objects (i.e. processes, users, network interfaces and mounts) and `cgroups` allow one to allocate and partition resources amongst different processes.

Container-based virtualisation is gaining popularity and acceptance, as evidenced by the successful open-source project Docker [12].

3. Methodology.

3.1. Performance Measurements in Virtual Machines. Since a virtual machine does not have complete control over the hardware, some care should be taken to ensure accurate measurements. One particular problem for performance measurements is unstable timekeeping inside VMs [4]. It is easy to see that since a VM time-shares with the host operating system, there will be more clock drift than expected for a physical server. This drift tends to be unpredictable as well because it is dependent on the state of a live system (e.g. how applications running on the hypervisor and other VMs use the system). As a simple example, consider a VMM implemented using Intel's VT-x HAV: when a virtual machine attempts a certain action (e.g. execute a privileged instruction), it induces a *VM Exit* event. Control is then transferred to the hypervisor, which performs some task related to the action and returns control to the VM. During this switch, the VM is not running and a gap could be introduced in the VM's virtual timestamp counter (TSC). While there are well established techniques for minimising this clock drift (e.g. a para-virtualised clock synchronised for KVM, **kvm-clock**), precise timing measurements in VMs are still difficult to obtain. For precise measurements, it is considered best practice to use a stable reference outside of the VM [20]. While the timing measurements used in this work are of a large enough scale for these effects to be considered insignificant, we use a 3rd party server to measure timestamps so that the same clock is used for all measurements. For higher resolution timing, one could use the hypervisor and a trapping mechanism (e.g. hypercall) as opposed to an external server.

3.2. Operating System Considerations. When benchmarking applications that use a large amount of input data from the disk, it is important to be mindful of how these machines cache this data. If one is using virtual machine images that are layered on top of an existing filesystem (as could be expected in a cloud environment utilising the KVM hypervisor), then both the VM operating system and host operating system could be caching data, depending on how the VMM is configured. Therefore, it is important to clear the buffer cache between all data samples to ensure that each experiment starts in a similar state (one could also reboot the machines to ensure a consistent state, but our experimentation has shown this to be excessive). In our experiments, virtual machines were restarted and the buffer cache was cleared before each sample.

Most modern multi-processor architectures have their memory arranged in a Non-Uniform Memory Access (NUMA) configuration. In a NUMA architecture, different processors/cores communicate with different regions of memory at different speeds, with fastest access to 'local' memory and slower access to 'non-local' memory. For optimal performance, it is clear that a process should only be scheduled to run on a CPU that has local access to the process's memory. Most modern OS schedulers are aware of NUMA, but one can still direct the OS to only run a process on a certain set of CPUs by setting the process to have affinity for a set of CPUS or 'pinning' it to those CPUs (e.g. by using the `taskset` command in Linux) [19]. Similarly, most VMMs offer the ability to pin vCPUs to certain physical CPUs. In this study, pinning of virtual machines' vCPUs was considered in performance tuning to avoid negative effects from non-local memory accesses.

4. Experimental Set-up.

4.1. Hardware and Operating System. The experiments were performed on two classes of machines, a Dell PowerEdge R720 server with dual-socket 8 core Intel Xeon E5-2660 'Sandy Bridge' 2.20GHz CPUs (3.0GHz Turbo boost) with 20MiB of cache and a homebuilt workstation computer with dual-socket 6 core Intel Xeon E5645 'Westmere' 2.40GHz CPUs (2.67 GHz Turbo boost) with 12MiB cache. Since multiple machines were available, experiments were repeated on two identical Dell R720 machines to rule out machine-specific bias and there was no discernible difference between the two (only one workstation class machine was available, so we could not perform the same validation). The Dell servers have 128GiB of DDR3-1333MHz memory and 8 1TiB ST91000640SS 6.0Gb/s SAS drives arranged in a RAID 1+0 array. The RAID controller used (PERC H710P Mini) has a 1024MiB battery backed cache, with the disk array set to the default 'write back' mode. This server configuration can be considered a typical enterprise class large-memory server one could expect to find

TABLE 4.1
Hardware specifications

Machine	System Name	CPU	Memory	Storage
Machine 1	Dell PowerEdge R720	Dual-socket Intel Xeon E5-2660	128GiB DDR3-1333MHz	8x1TiB 6.0Gb/s SAS RAID 1+0
Machine 2	'Homebuilt' workstation	Dual-socket Intel Xeon E5645	32GiB DDR3-1333MHz	1TiB 6.0Gb/s SATA

in a data centre or research cluster. The workstation has 32GiB of DDR3-1333Mhz memory with a single 1TiB ST1000DM003 6.0Gb/s SATA hard drive. Throughout this work, the Sandy Bridge Dell servers will be referred to as 'Machine 1' and the homebuilt Westmere workstation as 'Machine 2.' The hardware specifications of each machine are summarised in Table 1.

The host operating system used was Ubuntu 12.04 LTS (kernel version 3.5.0-45-generic). This platform was chosen as it is nearly ubiquitous in IaaS cloud environments and VM images containing this operating system are available by default from most, if not all, major IaaS public cloud providers.

4.2. Virtual Machine Configuration. The libvirt[3] API and toolchain (e.g. the `virsh` utility) were used to manage VMs for all platforms. The VMs started as OpenStack instances based on the 12.04 LTS Ubuntu cloud image,[4] and were afterwards the exported to be run separately outside of OpenStack. Both of the CPUs in the different machines support Intel VT-x HAV technology with VT-d and they also have support for Extended Page Tables (EPT). Unless stated otherwise, each VM was allocated 8GiB of memory and 4 vCPUs.

For each VMM, we first ran experiments with default configurations and then attempted to obtain maximum performance (shortest run time for the alignment applications) by tuning VMM settings for improved throughput. In the VMM configurations specified as 'default,' limited performance tuning was performed, i.e. ensuring all supported cpu features were available to VMs' vCPUs, using para-virtualised drivers wherever possible (e.g. `virtio` for KVM and the para-virtualised version of Xen).

4.3. Simulated Read Data. In order to ensure that differences observed were not caused by variance in input data, all of the results presented below used an identical copy of the input dataset. The input data consisted of paired-end 75bp reads sampled uniformly from human chromosome 1 (UCSC hg19[5]) with 2x coverage using the ART Illumina read simulator [11]. The simulated FASTQ files contained approximately 3 million reads with an average Phred score of 22 and mean fragment length of 200bp. We will refer to this dataset as **chr1** throughout the remainder of this work.

Since we simulate paired-end short reads, BWA was run with the `aln` command twice (to align each half of the read) and the `sampe` command took the output of each `aln` step to perform the final mapping. For both BWA and Novoalign, we indexed the reference *a priori* and both aligners were configured to output the alignment in SAM format [18]. Note that we only investigate the performance characteristics of the alignment utilities and aside from generic comments when comparing these tools, the quality of the output is beyond the scope of this work.

5. Results.

5.1. Default Settings. All of the results presented in this Subsection are for VMs and hypervisors with default settings from Ubuntu 12.04, as described in Sect. 4.2. The measurements reflect what one can expect for 'out-of-the-box' performance. Furthermore, the default performance values provide a useful baseline for tuning, enabling one to see what improvements can be made later on (cf. Sect. 5.2).

[3] `http://libvirt.org/`
[4] `http://cloud-images.ubuntu.com/precise/`
[5] http://hgdownload.soe.ucsc.edu/goldenPath/hg19/chromosomes/chr1.fa.gz

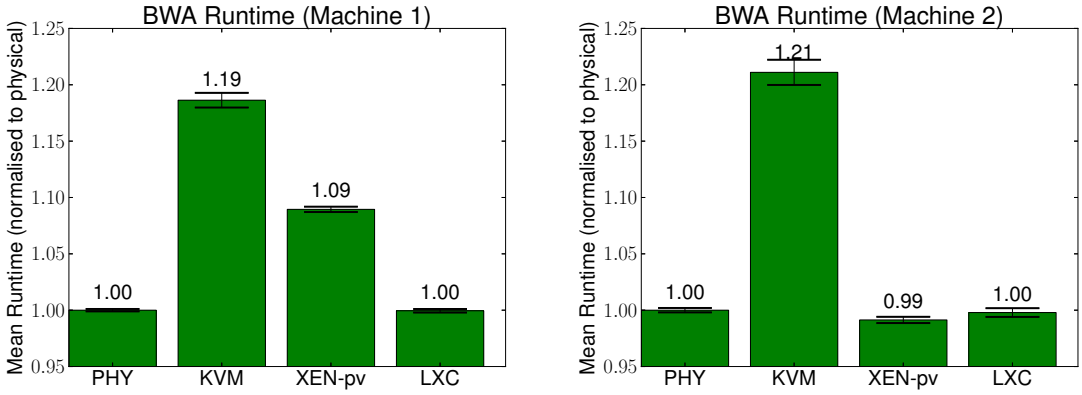


FIG. 5.1. *Results from 30 samples (10 on Machine 2) of running BWA on simulated paired-end Chromosome 1 reads on one virtual machine with the hypervisors in default configuration for Ubuntu 12.04. The x-axis indicates the hypervisor, where PHY = physical server, KVM = Kernel Virtual Machine, XEN-pv = Paravirtualised Xen and LXC = Linux Container. The error bars indicate the 95% confidence interval.* ***(a)*** *Machine 1 (Sandy Bridge R720), for which the physical server took a mean of 1955.49 seconds to complete the alignment.* ***(b)*** *Machine 2 (Westemere workstation), for which PHY took a mean of 2460.09 seconds to complete. (Note that even though XEN-pv has a normalised runtime of 0.99, the confidence intervals of PHY and XEN-pv overlap.)*

5.1.1. BWA. The results of running BWA on Machine 1 and all four execution environments (physical server, Kernel Virtual Machine, para-virtualised Xen and Linux Containers) are shown in Fig. 5.1.a. (Note that only one virtual machine was running at a time.) In this test, a paired end alignment with BWA was run 30 times on the **chr1** dataset, and a mean of the run time was obtained. In all cases, the application (BWA) was run single-threaded. Since many sequencing pipelines involve large amounts of data, it is not uncommon for users to run alignment and variant calling applications single-threaded and parallelise across datasets (e.g., to parallelise across chromosomes when mapping an entire genome). Here, we see that LXC's performance closely matches that of the physical server, but Xen and KVM have significant overhead. We reiterate that these results are consistent across runs on both of the servers with Machine 1's hardware configuration. However, when considering the performance on Machine 2 (cf. Fig. 5.1.b), one can see a stark difference from Machine 1. In particular, Xen exhibits a much larger overhead for Machine 1. This difference will be investigated in Sect. 5.2.2.

5.1.2. Novoalign. Novoalign was run on the same input data as BWA was, and the mean runtime and confidence intervals were obtained (cf. Fig. 5.2). Again, the application executions were single threaded computations with one VM running at a time. Note that the scales on the y-axis of the plots in Fig. 5.2 are significantly smaller than those of BWA. One can see that for Machine 1, KVM is now on par with the physical server, and Xen exhibits very low overhead.

5.2. Improvements from Tuning. Default configurations are generally designed to balance reasonable performance with power efficiency and resource sharing for a wide variety of machines. However, optimum performance can usually be obtained only by tuning various parameters and system behaviour, which are often highly specific to the system of interest. In this Subsection, we describe how we tuned each hypervisor to obtain performance as close as possible to that of running on a physical machine. For KVM, we followed best practices when it comes to tuning for memory and I/O throughput. In Xen, we found that the version has a strong impact, and we attribute that impact to vCPU scheduling improvements.

5.2.1. KVM Tuning. In all of the above runtimes for both BWA and Novoalign, KVM demonstrated the largest measured performance overhead: roughly 20% for both machines when running BWA. Since KVM is widely considered the *de facto* choice for Linux virtualisation, that high performance cost warranted further investigation. We start with recommended tuning practices and explaining the effect they have on the applica-

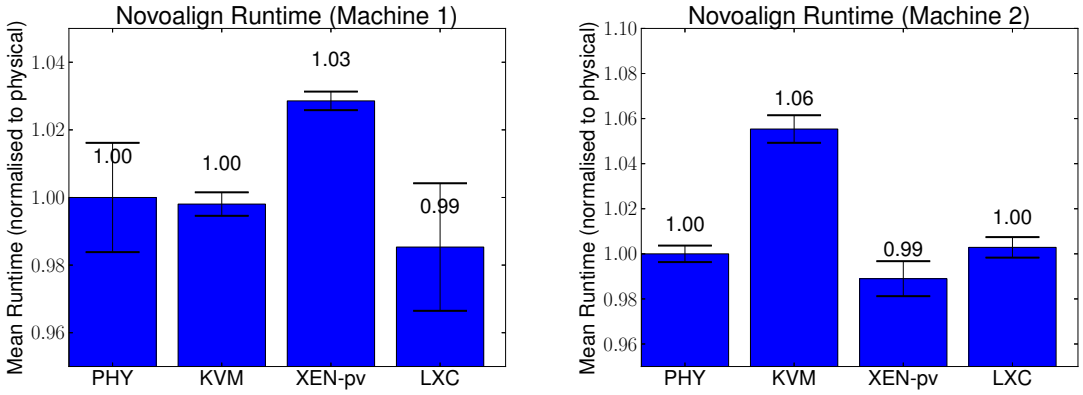


FIG. 5.2. *Results from 30 samples (10 on Machine 2) of running Novoalign on simulated paired-end Chromosome 1 reads on one virtual machine with the hypervisors in the default configuration for Ubuntu 12.04. The error bars indicate the 95% confidence interval.* ***(a)*** *Machine 1 (Sandy Bridge R720), for which the physical server (PHY) took a mean of 4533 seconds to complete the alignment.* ***(b)*** *Machine 2 (Westemere workstation), for which PHY took a mean of 5736 seconds to complete.*

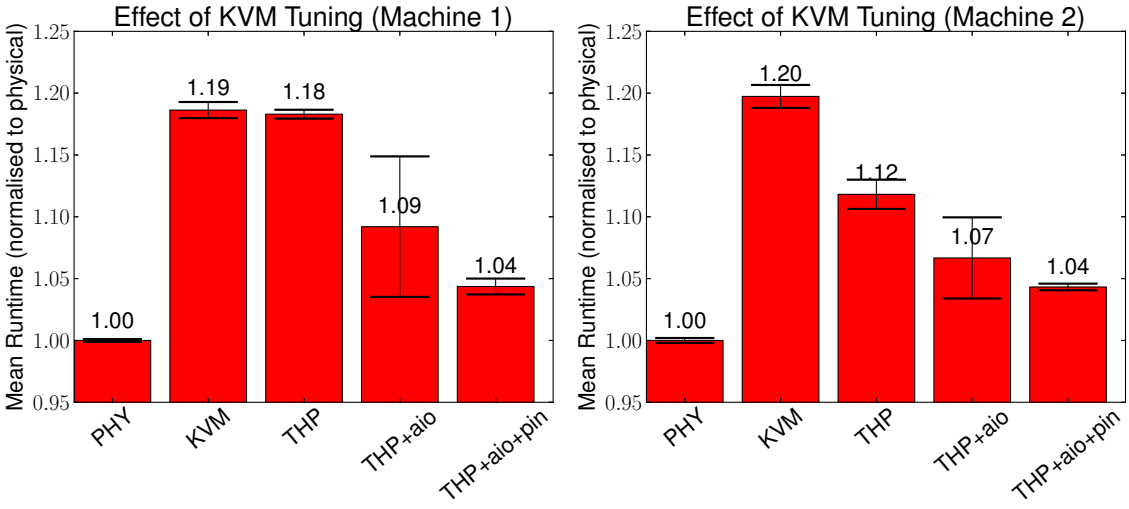
tions studied [29]. When tuning, we chose to focus on using BWA, since it had a shorter run time (allowing for more efficient experimentation), but it also consistently presented a high overhead with KVM. The results of KVM tuning are summarised in Fig. 5.3. As both sequence aligners used in this work are memory-intensive applications (in that they operate on relatively large in-memory datasets), improving memory performance should have a positive effect on application performance. In that respect, we investigate different virtual memory and NUMA configurations. Since BWA and Novoalign can work with large files, we also investigate parameters for tuning disk performance.

One way to improve memory performance is to reduce the number and cost of page faults by using 'huge' pages, or memory pages larger than the system's default. Traditionally, huge pages would need to be reserved by the operating system at boot time, but *transparent huge pages* (THP) can be dynamically assigned to anonymous memory regions (e.g., heap and stack space). `libvirt` supports the use of THP for KVM guest memory, and all data points designated with 'THP' (cf. Fig. 5.3) had THP enabled.

Though the para-virtualised `virtio` drivers were used with raw VM image file-backed virtual disks, there was still room to tune disk I/O parameters. The `libvirt` API defaults to using QEMU worker threads for asynchronous I/O (AIO). Another option is to use native Linux asynchronous I/O.[6] Experiments with native AI/O are designated by 'aio' in Fig. 5.3. We see that native AI/O is helpful in reducing the mean time on both machines. We also tested different caching strategies, but we found the default cache='none' (chosen by OpenStack when creating the VM image), which bypasses the host OS's buffer cache, to be the best-performing strategy.

As mentioned in Sect. 3, modern CPUs (sockets) have asymmetric memory access times to different memory regions. That asymmetric memory speed is also referred to as *Non-Uniform Memory Access* (as defined previously), and, in keeping with standard terminology, we will refer to a specific region of memory (e.g., memory associated with CPU socket 0) as a *NUMA node*. In our experiments, we observed that in the default configuration, roughly half of a VM's pages resided on one NUMA node while the other half resided on the other node. (Both types of machines we used had two CPU sockets and therefore two NUMA nodes.) In order to prevent processors from accessing memory non-local to their NUMA node, one can restrict virtual CPUs to a set of physical CPUs. In Fig. 5.3, the 'pin' data points refer to runs in which vCPUs were manually pinned to specific NUMA nodes.

[6] `http://www.linux-kvm.org/page/Virtio/Block/Latency`



(a) (b)

FIG. 5.3. *Results from 10 samples of running BWA on the* ***chr1*** *dataset using the KVM hypervisor with various tuning options (cf. 5.2.1 for more details). The error bars indicate the 95% confidence interval. 'PHY' indicates the physical server and 'KVM' indicates KVM with a 'default' configuration. 'THP' means that the hypervisor had Transparent Huge Pages enabled, 'aio' indicates that the VM was using Linux native asynchronous I/O and 'pin' indicates that the VM was pinned to CPUs on the same NUMA node.* ***(a)*** *Machine 1 (Sandy Bridge R720), for which the physical server (PHY) took a mean of 1955 seconds to complete the alignment.* ***(b)*** *Machine 2 (Westemere workstation), for which PHY took a mean of 2460 seconds to complete.*

5.2.2. Xen Version. When examining the performance of each VMM in its respective default configuration, we can see that relative VMM runtime completion is not consistent across machines (cf. Fig. 5.1 and 5.2). Specifically, Xen-pv exhibits a runtime that is (statistically) equivalent to that of the physical server for Machine 2, but the runtime for Machine 1 exhibits approximately a 9% overhead.

To further investigate those differences, we ran the low-level CacheBench benchmark both in and out of the Xen VMs to characterise the memory performance of each machine [23]. The results are displayed in Fig. 5.4. Inspection of Figs. 5.4.a and 5.4.b reveals that Machine 1 exhibits a performance degradation of roughly 10%, whereas Machine 2 has nearly identical memory performance under Xen 4.1. That disparity is also reflected in Figs. 5.1 and 5.2, which show that the Xen VM experienced performance overhead in Machine 1 (also close to 10%), but not in Machine 2.

While investigating the cause for the disparity in Xen performance across machine types, we experimented with different versions of Xen. After upgrading to Xen 4.4, we reran CacheBench, and the results for Machine 1 demonstrated no overhead. The CacheBench experiments were repeated for all minor versions of Xen, and it was confirmed that the performance on Machine 1 changed with version 4.3; the results are presented in Fig. 5.5. Contrasting the axes with those of Fig. 5.4, it is apparent that Xen 4.3 now matches the physical server performance (as Machine 2 did in Figs. 5.4.c and 5.4.d).

The 4.3 version of Xen included many performance enhancements, but the most notable is the more NUMA-aware scheduler it introduced.[7] Since forcing appropriate NUMA placement results in large improvements to applications running in KVM, it is not surprising that a more NUMA-aware scheduler would be helpful for a two-CPU-socket machine running Xen.

5.2.3. Overall Runtime after Tuning. Fig. 5.6 shows the relative (to the physical server) performance of BWA and Novoalign on all platforms after tuning. The tuning included NUMA node pinning for the physical server and LXC. As can be seen, performance ordering is now consistent across both machine types, and all VMs have single-digit percent overheads.

[7] `http://wiki.xenproject.org/wiki/Xen_Project_Release_Features`

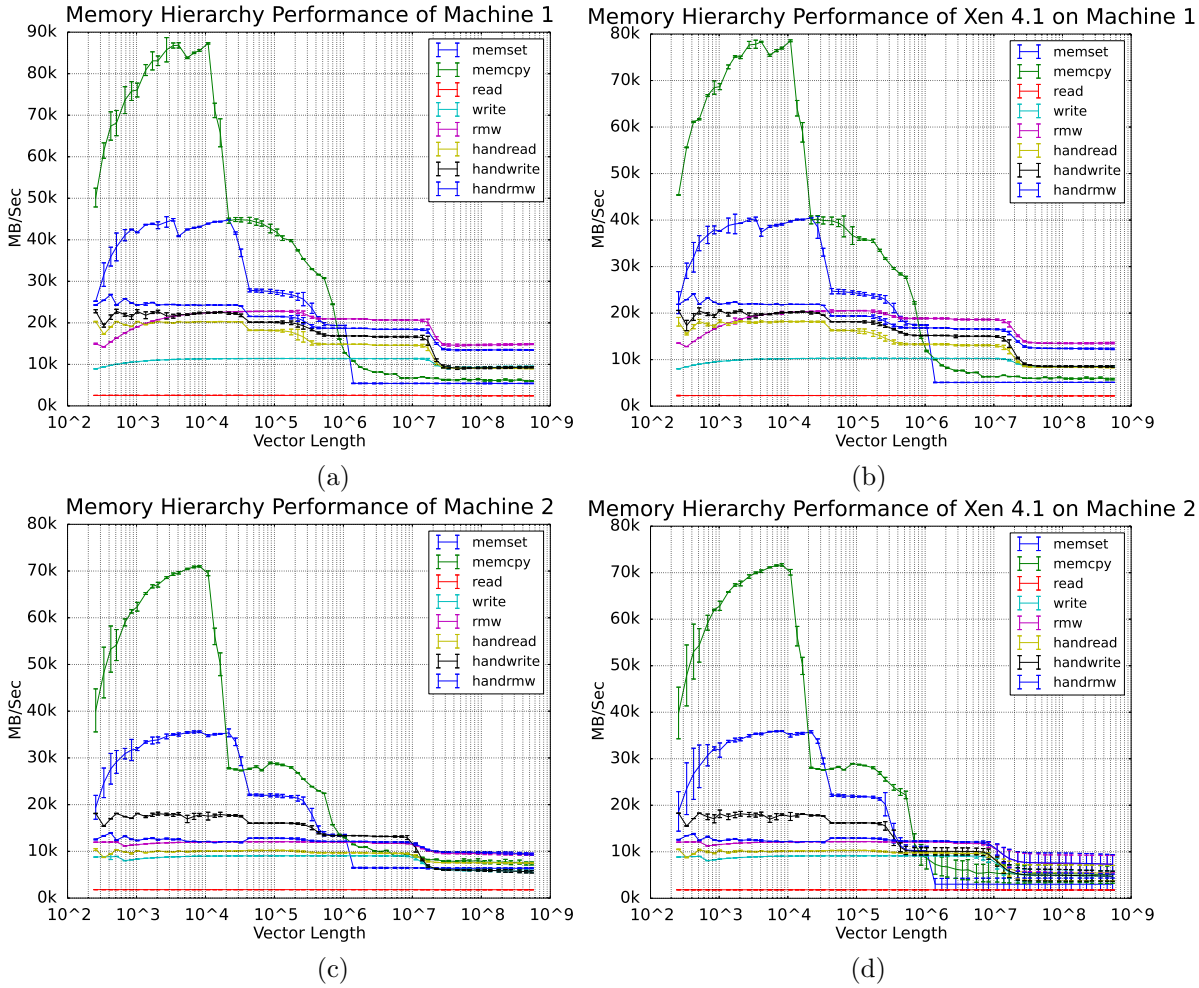


FIG. 5.4. *Results from CacheBench of llcbench run on Machines 1 (Sandy Bridge) and 2 (Westmere). Figs.* ***(a)*** *and* ***(b)*** *are from Machine 1, and Figs.* ***(c)*** *and* ***(d)*** *are from Machine 2. While* ***(a)*** *and* ***(b)*** *may appear to be identical, note the difference in y-axis scales, representing a roughly 10% performance difference. Error bars indicate the 95% confidence interval calculated over 30 independent runs of the benchmark.*

5.3. Execution Statistics. To better understand the characteristics of each application, we ran collectl[8] during the execution of a single instance of each application on Machine 1 (physical). Collectl is a utility that collects OS statistics about a running system, such as CPU load and memory usage. We chose to measure CPU, memory and disk usage over time. The outputs from collectl during the execution of each application are graphed in Figs. 5.7.a and 5.7.b. The execution pattern in Fig. 5.7.a is consistent with the execution of BWA commands. As can be seen in Fig. 5.7.b, the first `aln` command finished at roughly 40% of the total runtime and the SAM output is generated when `sampe` starts at roughly 90%, initiating the heavy disk activity. Fig. 5.7.b shows that Novoalign reads the input files all at once and performs its operations in-memory.

6. Analysis and Discussion. Analysing the data presented earlier, we make the following observations:

6.1. CPU pinning can be effective for managing resource contention. One surprising observation was that while modern OS schedulers (the Linux task scheduler schedules KVM vCPUs) are NUMA-aware and should attempt to keep processes running on the same CPU to avoid cache thrashing, we were still able to gain a considerable performance increase by manually placing vCPUs/processes. It should be noted that

[8] `http://collectl.sourceforge.net/`

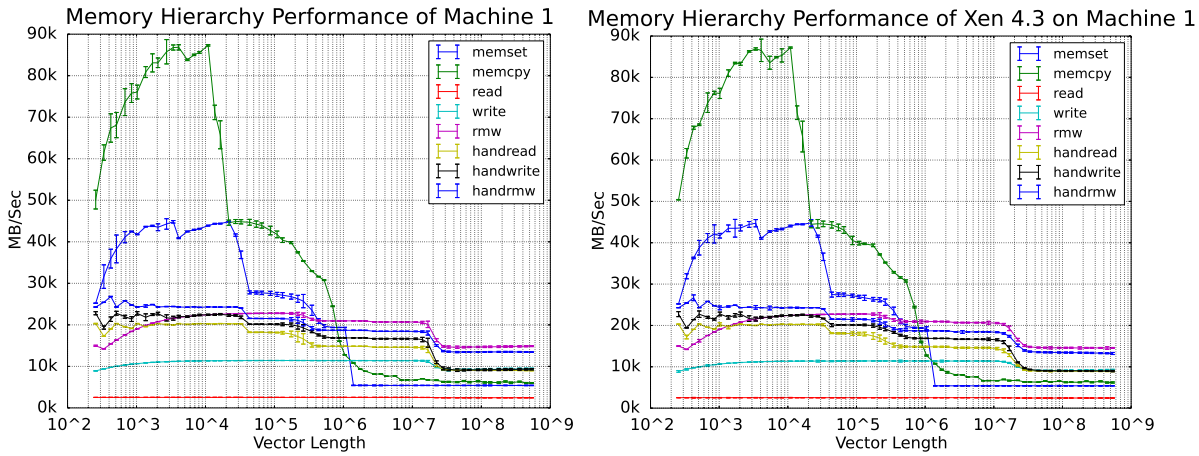


FIG. 5.5. *Results from CacheBench of llcbench run on Machine 1 with Xen 4.3.* ***(a)*** *shows the physical server and* ***(b)*** *shows llcbench run under Xen 4.3, which alleviates the overhead seen in Figs. 5.4.a and 5.4.b. Error bars indicate the 95% confidence interval calculated over 30 independent runs of the benchmark*

.

the machines were not maximally utilised in our studies, and more intensive workloads would likely exacerbate the performance loss of poorly placed processes. For the proper Xen versions, Xen did quite well in matching physical machine performance, and we attribute that to a more NUMA-aware scheduler.

6.2. Despite having a similar objective, alignment programs can utilise system resources in very different ways. Figs. 5.7.a and 5.7.b allow us to compare how the two alignment applications differ in their use of system resources. BWA periodically reads from the disk, while Novoalign reads all of its input in one shot. However, we expect that behaviour to change with datasets larger than system memory, and further experimentation is required. The applications' differing behaviour provides evidence contrary to our hypothesis in Sect. 2.1 that alignment programs should utilise resources similarly. Therefore, users and system integrators should be careful to observe how their chosen application performs in a given environment.

6.3. Novoalign is efficient with system resources. Based on our observation of how the applications use resources, while Novoalign takes more than twice the time BWA does to finish execution on both systems, Novoalign is more efficient with system resources. One must remember that though the objective of these applications is the same, their outputs from the same input data are different. Fig. 5.7.a shows that BWA has a more bursty read/write pattern. It could fill caches and cause the application to block on I/O bandwidth. In Fig. 5.7.b, Novoalign can be seen to make many small writes (as is also evident from the constant steep increase of the buffer cache size). In addition to the battery-backed write cache on Machine 1's RAID controllers, modern filesystems (e.g., the ext4 [21] filesystem used on both machines) have a delayed allocation cache strategy, and the many small writes that Novoalign performs would likely be more amenable to this strategy than to bursts of larger writes.

6.4. Performance for a given VMM is dependent on the application being executed. In Fig. 5.1.a, one can see that KVM performs the worst for BWA on Machine 1, but in Fig. 5.2.a, Xen performs worse for Novoalign (although the difference is small). We note that those results are for an 'improper' version of Xen before performance enhancements. If one observes similar behaviour where application performance is different for different VMMs, it may be indicative of an issue like the one we faced with Xen 4.1 on Machine 1. Also, this observation reveals that the behaviour of one application on a particular VMM may not be fully indicative of how another application would behave on the same VMM, or how that same application would perform on a different VMM.

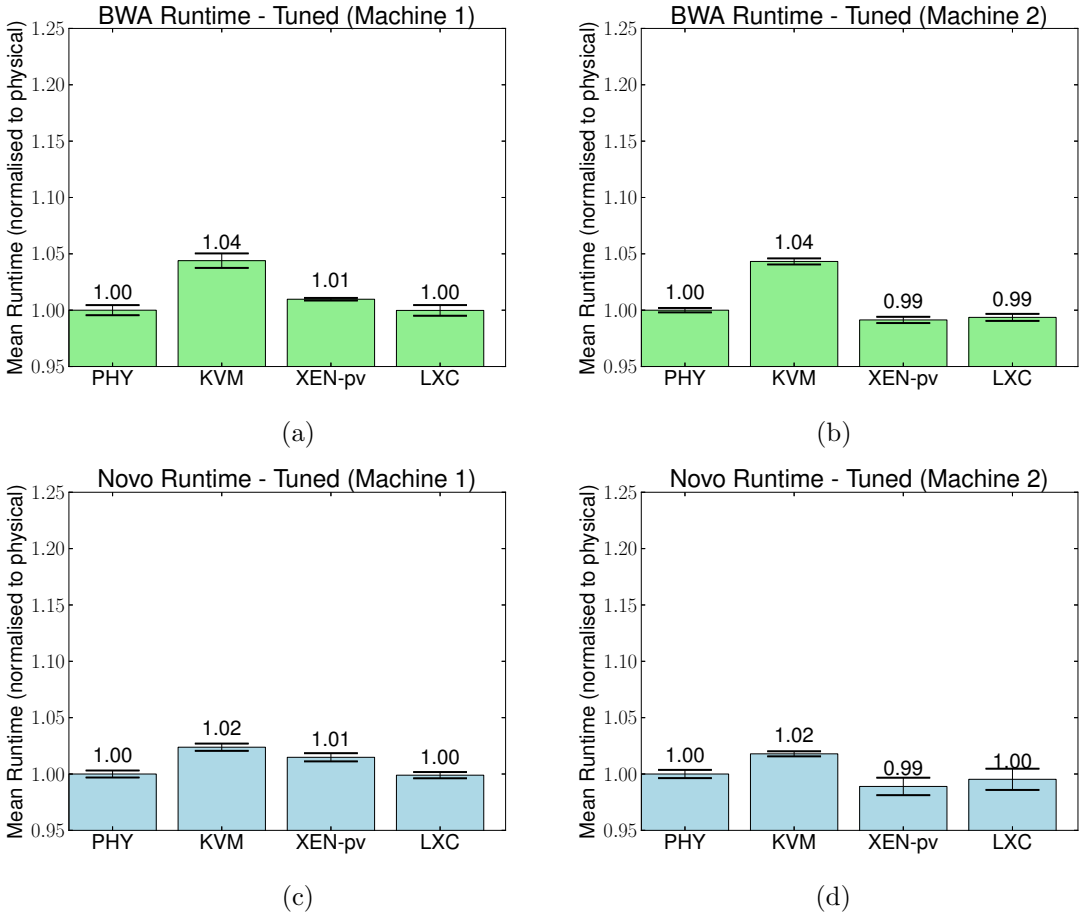


FIG. 5.6. *Results from 10 samples of running BWA on simulated paired-end Chromosome 1 reads with the hypervisors tuned for performance (see 5.2 for more details). The values indicate performance relative to a physical machine running on identical hardware with NUMA-aware CPU pinning. The error bars indicate the 95% confidence interval.* ***(a)*** *Machine 1 (Sandy Bridge R720), with a physical server runtime of 1954.93* ***(b)*** *Machine 2 (Westemere workstation), with a physical server runtime of 2460.09.* ***(c)*** *and* ***(d)*** *show data for the same systems as* ***(a)*** *and* ***(b)*** *(respectively), but with Novoalign instead of BWA. For Novoalign, Machine 1 had a physical server runtime of 4212.19, and Machine 2 had a runtime of 5714.91. While KVM had no measurable overhead in the default configuration with Novoalign, when both PHY and KVM were tuned for NUMA, we observed a drop in the relative performance for KVM, even though it completes in 94% of the time of the original PHY measurement.*

6.5. After tuning, performance ordering is the same across machine types. Keeping in mind the above observation and considering Figs. 5.1.a and 5.2.a, we see that in one case KVM is the slowest, and in the other case, Xen is the slowest. After VMMs were tuned, performance ranking became consistent across applications and machine types. It would take a more detailed study to confirm the generality of this observation.

6.6. The extent to which huge pages are beneficial is machine-dependent. In Fig. 5.3, one can see that Machine 2 benefits noticeably more than Machine 1 from using THP. Machine 1 has a larger cache size than Machine 2, so TLB misses are likely less expensive on Machine 1, as page table entries are more likely to reside in a lower-level cache on Machine 1.

6.7. CPU-bound applications run at near-baremetal speeds in virtualised environments. This was an expected result, as virtual machines should run unprivileged operations at native speeds. This is helpful information for bioinformaticians whose workloads may be largely CPU-bound. Furthermore, when considering performance, it appears that as long as a task behaves in that way, the choice of VMM is less important.

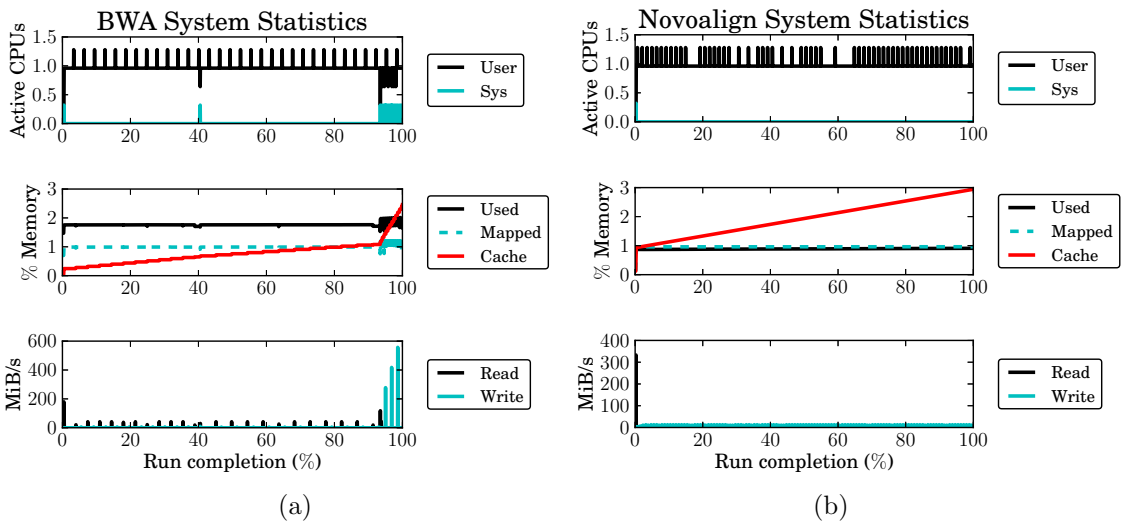


FIG. 5.7. *Output from running collectl during a single-threaded execution of BWA (a) and Novoalign (b) on Machine 1. CPU, memory and disk statistics were sampled and collected once per second. The CPU data were normalised to the number of available processing threads the system had, and 'User' refers to the time the CPU spends in userspace, 'System' refers to the time the CPU spends performing OS tasks. For memory, 'Used' refers to the percentage of system memory used by applications; 'Cache' corresponds to the amount used by the buffer cache (cleared immediately before each run), and 'Mapped' corresponds to the amount of memory used for memory-mapped files. The last plot in each set corresponds to disk I/O and shows the read/write throughput of the disks at each point in time. We remind the reader that the disk controller used on Machine 1 has an external 1GiB battery-backed cache.*

As discussed in 6.2, Novoalign and BWA have different disk access patterns. We believe that Novoalign's disk access pattern contributes to its having a smaller virtualisation performance cost than BWA. That access pattern causes Novoalign to become CPU/memory-bound after application start-up, and to be overall less sensitive to VMM hardware emulation overhead.

6.8. Linux Containers have near-zero performance overhead. Based on our data, it is clear that if one wishes to build a cloud computing environment with performance as the paramount concern, a cloud built using Linux Containers would be the best choice amongst the options presented here. That conclusion was expected, but the intention of performing the experiments with LXC was to demonstrate and quantify the performance improvement for a bioinformatics workload. But, anyone choosing to adopt a container-based solution should be aware that there are certain drawbacks. We tested with a newer version of libvirt (1.1.1), and the LXC driver still did not support vCPU pinning without manual modification of `cgroups`. Also, some cloud adopters see the ability to choose one's entire software stack (including the OS kernel, especially if the users are interested in a particular version or tuning parameters) as an advantage of cloud computing, and it is impossible to do so when using LXC unless one has control over the host OS (which is not likely in a cloud scenario).

6.9. Paravirtualised Xen has extremely low overhead. After the appropriate version of Xen was chosen, it did exhibit an extremely small overhead. Xen has nearly the same performance as LXC, but with a much more robust isolation method.

7. Related Work. Cloud computing services have been available for quite some time now, and as these technologies have matured, more users are porting their workloads to cloud environments. Many studies have focused on quantifying the impacts of cloud computing's underlining technologies, particularly virtualisation, to user applications. The results of these studies play an important role in guiding users to selecting or building the most suitable cloud for their workload.

Many independent studies have been conducted to quantitatively compare the performance among open source hypervisors [5, 28] and among commercial hypervisors [31, 24]. The authors of study [5] compared the

KVM and Xen hypervisors in three aspects, namely overall performance, performance isolation and scalability (i.e., the ability to add virtual machines without losing performance). Based on standard benchmarks, such as Linux kernel compilation and IOzone, they did not reach a general conclusion as to whether KVM or Xen is better because the performance results are application-dependent. In the performance isolation test, KVM outperformed Xen. Meanwhile, Xen was demonstrated to be more scalable than KVM. Coming from a commercial landscape, both studies [24] and [31] compared the performance of XenEnterprise and VMWare ESX by using a set of industry-standard performance benchmarks, such as SPECcpu2000Integer, Passmark, Netperf and SPECjbb2005. The two studies presented an interesting competition between the two popular commercial offerings of hypervisors. In the first study [24], VMWare ESX 3.0.1 noticeably outperformed XenEnterprise 3.0.3 in majority of the tests. Interestingly, the second study [31] showed that XenEnterprise version 3.2 matched VMWare ESX 3.0.1 in most tests, with some exceptions where Xen performed better.

The traditional High Performance Computing (HPC) community has expressed their interest in cloud computing through many studies comparing HPC workloads on different types of VMMs. The common perception is that the expected performance overhead of $\sim$10% is acceptable for these applications when not running tightly coupled MPI-style computations [9]. To put that number in perspective, however, for a sequencing workload that could take e.g. 1000 CPU Hours to complete, 10% would be equivalent to roughly 4 CPU days. The authors in [13] ran tightly coupled MPI-based applications and observed significantly degraded performance (6-20x) on EC2. In a study closer to what was done in this work, the author in [10] used WCD EST, an EST assembler (contrasted with the whole genome assemblers used here) to measure the performance of EC2 for scientific workloads. The author focused on scalability and variability, and showed that EC2 can achieve acceptable scalability. This work also highlighted many qualitative benefits of moving to the cloud vs. participating in a typical shared computing cluster.

A recent study investigated the potential use of container-based virtualisation in HPC [30]. However, that study focused on HPC benchmarks, whereas the work presented in this paper evaluates actual application performance. That work also demonstrated that LXC and a physical server have near-identical performance, but found that Xen was slower than LXC and the physical server by roughly a factor of 2. It is difficult to determine the cause of the stark difference between their results and those presented in this paper, but the version of Xen was similar (4.1.2 in the other work and 4.1.5 as we initially tested here). As presented in Sect. 5.2.2, future versions of Xen may yield significantly better results. Additionally, the operating system used in that study was Ubuntu 10.04 and potentially had less efficient para-virtualisation drivers.

In a similar effort, study [8] presented a performance comparison of KVM and Docker [12], which is built using Linux Containers. This study demonstrated that Docker's performance is equal or better than KVM in most tests. In addition, the authors showed that there are opportunities for performance tuning in both KVM and Docker. Our study again confirmed this conclusion in the domain of the genomics software.

Our study focuses on comparing three popular open-sourced VMMs, which cover all three major virtualisation technologies, namely Type I hypervisors, Type II hypervisors and light-weight containers. Furthermore, we specifically tuned these virtualisation platforms for genomic workloads, and showed that this is a worthwhile effort.

8. Conclusions. The results presented here are useful for genomics researchers and clinicians who are considering adopting a cloud based solution, particularly those who elect to explore the option of building a private cloud. As expected, when workloads became more CPU intensive and less I/O bound, performance inside a VM is practically indistinguishable from a physical server. We have demonstrated that Novoalign falls strongly into this class of software and is amenable to virtualisation. After proper tuning, BWA essentially exhibits similar behaviour. This means that these applications are ideal for use in a cloud setting, but one must either build a private cloud or work with the cloud provider to ensure proper tuning has taken place as an 'out-of-the-box' solution may fall very short of the full potential of the system. From a performance perspective, We believe that the small overhead introduced by virtualisation leaves IaaS cloud computing as a viable solution for genomics workloads that have traditionally been run inside data centres.

Acknowledgements. The authors are grateful for insightful and helpful discussions with Steven Lumetta, Victor Jongeneel, Liudmila Mainzer, Gloria Rendon and Arjun Athreya. The authors also thank Jenny Applequist for her assistance in preparing this article. This material is based on research sponsored in part by the

National Science Foundation, in part by the Air Force Research Laboratory and the Air Force Office of Scientific Research, under agreement number FA8750-11-2-0084, in part a grant from Infosys technologies, and in part by a Faculty Award from the IBM Corporation. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Portions reprinted, with permission, from [7] ©2014 IEEE.

REFERENCES

[1] *LXC container driver*. Online, `http://libvirt.org/drvlxc.html`, 2014.
[2] *Novocraft technologies*. Online, `http://novocraft.com`, 2014.
[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, *Xen and the art of virtualization*, ACM SIGOPS Operating Systems Review, 37 (2003), pp. 164–177.
[4] T. Broomhead, L. Cremean, J. Ridoux, and D. Veitch, *Virtualize everything but time.*, in OSDI, vol. 10, 2010, pp. 1–6.
[5] T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, *Quantitative comparison of xen and kvm*, Xen Summit, Boston, MA, USA, (2008), pp. 1–2.
[6] J. T. Dudley, Y. Pouliot, R. Chen, A. A. Morgan, and A. J. Butte, *Translational bioinformatics in the cloud: an affordable alternative*, Genome medicine, 2 (2010), p. 51.
[7] Z. Estrada, Z. Stephens, C. Pham, Z. Kalbarczyk, and R. Iyer, *A performance evaluation of sequence alignment software in virtualized environments*, in Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, May 2014, pp. 730–737.
[8] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, *An updated performance comparison of virtual machines and linux containers*, technology, 28, p. 32.
[9] M. Fenn, M. A. Murphy, and S. Goasguen, *A study of a kvm-based cluster for grid computing*, in Proceedings of the 47th Annual Southeast Regional Conference, ACM, 2009, p. 34.
[10] S. Hazelhurst, *Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud*, in Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology, ACM, 2008, pp. 94–103.
[11] W. Huang, L. Li, J. R. Myers, and G. T. Marth, *ART: a next-generation sequencing read simulator*, Bioinformatics, 28 (2012), pp. 593–594.
[12] S. Hykers, *What is docker?* Online, `https://www.docker.com/whatisdocker/`, 2013.
[13] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, *Performance analysis of high performance computing applications on the Amazon web services cloud*, in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE, 2010, pp. 159–168.
[14] M. Kerrisk, *Namespaces in operation, part 1: namespaces overview*. Online, `http://lwn.net/Articles/531114/`, 2013.
[15] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, *kvm: the linux virtual machine monitor*, in Proceedings of the Linux Symposium, vol. 1, 2007, pp. 225–230.
[16] H. Li, *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM*, arXiv preprint arXiv:1303.3997, (2013).
[17] H. Li and R. Durbin, *Fast and accurate short read alignment with burrows–wheeler transform*, Bioinformatics, 25 (2009), pp. 1754–1760.
[18] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, et al., *The sequence alignment/map format and samtools*, Bioinformatics, 25 (2009), pp. 2078–2079.
[19] R. Love, *CPU affinity*, Linux Journal, (2003).
[20] P. Luszczek, E. Meek, S. Moore, D. Terpstra, V. M. Weaver, and J. Dongarra, *Evaluation of the HPC challenge benchmarks in virtualized environments*, in Euro-Par 2011: Parallel Processing Workshops, Springer, 2012, pp. 436–445.
[21] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, *The new ext4 filesystem: current status and future plans*, in Proceedings of the Linux Symposium, vol. 2, Citeseer, 2007, pp. 21–33.
[22] P. Menage, *CGROUPS*. Online, `https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt`, 2006.
[23] P. Mucci, *Llcbench(low-level characterization benchmarks) home page*, World Wide Web, http://icl.cs.utk.edu/projects/llcbench/(July 2000), (2000).
[24] P. Muditha Perera and C. Keppitiyagama, *A performance comparison of hypervisors*. Online, `http://www.vmware.com/pdf/hypervisor_performance.pdf`, 2007.
[25] X. Qiu, J. Ekanayake, S. Beason, T. Gunarathne, G. Fox, R. Barga, and D. Gannon, *Cloud technologies for bioinformatics applications*, in Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, ACM, 2009, p. 6.
[26] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, *Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors*, in ACM SIGOPS Operating Systems Review, vol. 41, ACM, 2007, pp. 275–287.
[27] L. D. Stein et al., *The case for cloud computing in genome informatics*, Genome Biol, 11 (2010), p. 207.
[28] A. Theurer, K. Rister, O. Krieger, R. Harper, and S. Dobbelstein, *Virtual scalability: Charting the performance of linux in a virtual world*, in Linux Symposium, 2006, p. 393.
[29] M. Wagner, *KVM PERFORMANCE IMPROVEMENTS AND OPTIMIZATIONS*. Online, `http://www.linux-kvm.org/`

wiki/images/5/59/Kvm-forum-2011-performance-improvements-optimizations-D.pdf, 2011.
[30] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, *Performance evaluation of container-based virtualization for high performance computing environments*, in Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, IEEE, 2013, pp. 233–240.
[31] A. XenSource, *Performance comparison of commercial hypervisors*, 2007.

Edited by: Jesus Carretero
Received: September 1, 2014
Accepted: January 21, 2015

DOI 10.12694/scpe.v16i1.1062
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 85–101. http://www.scpe.org

EXTENDING XNAT TOWARDS A CLOUD-BASED QUALITY ASSESSMENT PLATFORM FOR RETINAL OPTICAL COHERENCE TOMOGRAPHIES*

CHRISTOPH JANSEN† MAXIMILIAN BEIER† MICHAEL WITT† JIE WU‡ AND DAGMAR KREFTING†

Abstract. Neuroscientific research is increasingly based on image analysis methods. Among them, optical coherence tomography (OCT) allows for non-invasive visualisation of anatomical structures on a micrometer scale. The platform presented in this paper is designed for automatic quality assessment of retinal OCTs. It extends the image management platform XNAT by services to calculate and store quality measures. It is also extensible regarding new quality measure algorithms, allowing the developer to upload, compile and test code for the system's architecture. The processing tools are provided as a cloud-based service employing OpenStack. Different approaches using fully equipped virtual machines and Docker containers are investigated and compared regarding security and performance aspects.

Key words: medical imaging, cloud, SaaS, IaaS, XNAT, OCT, OpenStack, Docker

AMS subject classifications. 68M14, 68M20, 68U10, 68U35

1. Introduction.

Modern neuroscience research is increasingly using imaging techniques such as magnetic resonance imaging (MRI) and computed tomography (CT). Among these techniques, optical coherence tomography (OCT) utilizes the characteristic refractivity of different tissue for non-invasive visualisation of anatomical structures on micrometer scale. Becoming a standard diagnostic tool in ophthalmology, it is of rising interest for investigation of multiple sclerosis, as correlation between retinal changes and the patient's symptoms could be found [1, 2].

1.1. Image Quality.

Virtually all data processing relies on sufficient data quality, but few tools inherently validate the quality of the input data. In today's complex image processing, as for example the FreeSurfer pipeline for segmentation of anatomical brain MRI, it is difficult to manually evaluate in advance, if a certain image's quality is sufficient for the envisioned processing [3]. Information about input data requirements are often only given in the software documentation and it is left to the user to use appropriate input data [4]. Today this issue is addressed by offering a visual inspection and interactive correction of intermediate results. This is a feasible approach if there are few datasets, but with increasing numbers of datasets and patients in clinical research, manual processing of the images becomes hardly manageable. Furthermore, the evaluation of the appropriateness of an image for a certain analysis method may require a high level of expertise and experience. Therefore, automatic in-advance quality validation, as known from commercial online photo print services, would help the researcher to avoid quality-related result errors. In this paper, a platform specifically designed for *Quality Management for Retinal Optical Coherence Tomography* (QMROCT) is presented. At least for the most commonly used type of retinal OCT scan, the so-called ring scan, quality criteria are specified, known as OSCAR-IB criteria [5]. The criteria are categorized into the groups "(O) obvious problems, (S) poor signal strength, (C) centration of scan, (A) algorithm failure, (R) retinal pathology other than MS related, (I) illumination and (B) beam placement". Some of these features – for example the signal strength – are easily evaluated automatically, while others, such as the correct position, might require more complex image analysis including certain segmentation and image registration tasks itself.

1.2. Research Platform.

Medical image data management for research requires further functionalities than classical picture archiving and communication systems (PACS) used in daily care. Additional features are for example the grouping of data into different clinical trials, storage and management of non-DICOM data and fine-grained access control [6]. An increasingly used open source platform supporting such features is the "Extensible Neuroimaging Archive Toolkit" (XNAT) [7]. In conjunction with several collaborative groups

* The presented work is supported by the German Federal Ministry of Education and Research (Somnonetz-01EZ1132), the German Federal Ministry of Economic Affairs and Energy (Macchiato), the Berlin Institute for Applied Research (QM ROCT), and the ITEA2-programme (EASI-CLOUDS).

†University of Applied Sciences Berlin Germany, Email: c.jansen@student.htw-berlin.de
‡TMF e.V. Berlin, Germany

around the world, the INCF Task Force has also started work on several tools to ease and eventually automate the practice of data sharing in field of neuroimaging [8]. The goal is to allow researchers to easily share raw, processed, and derived neuroimaging data and improving the reproducibility of neuroimaging studies.

1.3. Cloud infrastructures for medical imaging. Medical image processing methods for post-processing and analysis to study the normal and pathological brain structure and function often have high demands on computing power and memory. The resulting requirements on data and computing availability and reliability are specifically addressed by cloud infrastructures. Cloud computing is 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 [14]. Common cloud platforms are Microsoft Azure [10], Amazon EC2 [11], Google Cloud Platform [12] and the open source platform OpenStack [13]. Recently, research projects have been started to implement cloud-based neuroimaging applications. The German part of the EASI-CLOUDS [20] project implements a cloud-based neuroradiologic analysis platform to process compute-intensive clinical examinations. The project focuses on Service Oriented Architecture (SOA) and Service Level Agreement (SLA) management for several applications. The NeCTAR group is working on providing cloud-based image analysis and processing software packages [21] for research communities via remotely accessible user-interfaces. This project focuses on processing large image data sets in a cloud environment. To control the processing, a workflow management is integrated and used as the user interface. To improve performance in large medical image access and processing, a cloud-based multi-agent system is built for image retrieval by Alonso-Calvo and others [22]. Here, single images are divided so that the resultant sub-images are stored and managed separately by different agents distributed within the cloud. Da Mota and co-workers [23] are using Microsoft Azure to apply machine learning algorithms to functional Magnetic Resonance Imaging (fMRI) data in a MapReduce approach. They are mainly focusing on algorithms, as well as efficient and stable processing. R & D Cloud CEIB [24] is a bioimaging project developing a modular cloud software for classification purposes. This team also extends XNAT for their needs in bioimaging by using its generic DICOM and XML data types. Furthermore, many solutions developed for grid-based neuroimage processing can be adopted to cloud infrastructures. For example, web-based Grid portals such as the Charité Grid portal [15] and the WS-PGRADE portal [16] provide components and services (e.g. Grid Security Infrastructure (GSI) [17] based authentication and authorization, workflow management, pseudonymisation service especially for patient data and secure data transfer) which can also be used in cloud-based neuroimage processing. As an additional feature for applications with user interaction, the visualisation of intermediate results can also be integrated into neuroimage applications [19].

2. Requirements. The overall goal of the presented platform is to facilitate state-of-the-art image quality assessment. This implies dynamically growing image data and quality measures. New images are created by the scientists performing clinical trials. The image data is assumed to be acquired in intermittent bursts, whenever a clinical trial is in its patient recruiting phase. In contrast, new quality measures are expected to be developed continuously but in a much larger time-scale. From the scientist's perspective, the system must provide the following features:

1. Whenever an image is stored in the platform, the currently available quality measures are calculated.
2. The calculated quality measures are accessible in the context of the image.
3. For manual inspection, a visual representation of the calculated quality features is accessible in the context of the images.
4. Whenever a new quality measure is integrated into the system, it is automatically calculated for all stored images.
5. Whenever a quality measure is updated, the new version is calculated for all images, and the former values are archived.

2.1. Users. Two abstract types of users are identified: The *clinical researcher*, who manages his or her medical images and studies within the platform; and the *computer scientist*, who develops new automated quality measure methods and integrates them into the platform. All interfaces of the platform to the clinical researcher should be integrated into his or her daily working environment, whereas the algorithm developer

must not have access to the medical data stored within the system, except authorized test data. It is possible to subdivide the two types further, for example by setting up different projects for different medical teams in the research platform. The development process of the algorithms themselves is not part of the platform. Developers prefer to use their own personalized environment for code development, in our case Matlab. It implies that an interface to integrate new methods must be provided for the developer. In particular, the developer needs to be able to control the reliability of the code on the platform, so he or she must be able to run the code on the platform with test data and confirm the intended behaviour of the integrated software.

2.2. Resources. Regarding resource consumption, we expect constantly growing storage and strongly varying computing power requirements. Whenever new data or new processing methods are integrated into the platform, image processing is necessary. Each new image will be analysed with all currently available methods and each new algorithm will be applied to all currently available images. Such a scenario with strongly varying resource requirements might strongly benefit from scalable computing resources, as provided by cloud infrastructures. As the processing of the individual images and algorithms are independent from each other, they can be executed simultaneously in a distributed environment. As a possible drawback, data transfer might be a significant factor for the total processing time in distributed environments, as image processing is in most cases data-intensive. So besides scalable computing resources, broadband network capabilities might also be crucial for performance earnings.

2.3. Security. There are several security requirements regarding the envisioned platform, from medical data protection to malicious code execution.

Medical data protection. As medical images are stored and processed, personal data protection must be ensured. While the retina of the eye is known to be unique in every human individual, the OCT data implicitly contain reidentification potential. On the other hand, reidentification based on the OCT data is only possible, if the attacker already has access to the subject's data. In contrast to e.g. reconstructed 3D-MRT of a head, the identification features of the OCT are not observable by everyone. The envisioned platform will be used by a clinical trials unit of the university clinics, which implies, that the data is specifically acquired for study purposes and not for patient care. The image metadata do not contain the patient's name but a pseudonym that is managed locally within the clinical trials unit. The only identifying information found in the metadata is the patient's birthday, which is automatically changed to the year of birth in a preprocessing step before entering the platform. We assume that strong data protection measures like file-based encryption are not required in the current setting.

Confidentiality of the research topic. As neuroscience research is a very active and competitive field, scientists are very careful in exposing their ongoing research projects and their data in advance. So weakness in the platform security might discourage usage.

Availability of the system. As the system is envisioned to be integrated into the scientist's daily working environment, it must not interfere with other tasks the user is performing within the system. In particular, temporary load peaks on memory and the CPUs due to image processing may slow down other applications.

Integrity of the system. As image processing methods are subject to current research, the system is explicitly laid out for integration of new methods without any need to alter the system itself. It implies that the system must be open in the sense that arbitrary code can be executed in the system. This is a high security risk, as malicious or unstable code may affect the system's integrity and stability. Code may delete or modify important system data or serve as an information leak. So security measures against these system vulnerabilities must be taken.

3. Methods. This section explains the components of the research infrastructure and the applied security measures.

3.1. Components. The two main components in the system are the research platform and the cloud infrastructure. To keep the different concerns of the two user groups – clinical researcher and computer scientist – well separated, we decided to integrate separated components for code and job management. As a design principle, the components are loosely coupled and use standard tools and protocols wherever possible. The system encompasses different aspects of services: The underlying resources are provided as infrastructure as a service (IaaS), the deployment of new algorithms is managed by platform as a service (PaaS) and the actual

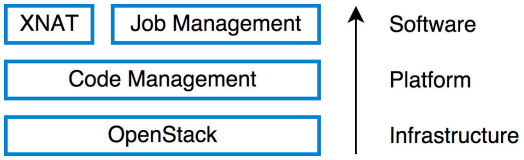


FIG. 3.1. *QMROCT components and their service categories.*

quality assessment is implemented as software as a service (SaaS). Different components of the system contribute to the different service models (cf. Fig. 3.1).

3.1.1. Infrastructure as a Service.

OpenStack. OpenStack is an Infrastructure-as-a-Service platform. It allows for on-demand creation and utilization of virtual servers, storage and network. It incorporates the management in form of multiple APIs with overall user and role management. OpenStack provides both command line tools as well as a web interface to ease the use of the REST API, that allows for managing users, projects, virtual servers, network and storage. Cloud infrastructure components dedicated for computing purposes (so called compute nodes) provide the resources (CPU and memory) to execute virtual environments that are tailored for their computation task. Load can be balanced throughout the system and therefore peaks in the required computational power can be handled. The system scales horizontally by adding additional compute nodes which will enhance the systems total memory capacity and processing power. A crucial part in this scenario, where computing resources can be created on demand and required data storage capabilities are assigned on runtime, is the network infrastructure, which is responsible for this high dynamic processing. All compute nodes are connected through a management IPv4 network, which enables communication with the control components. Everything regarding dynamic storage and computation is handled in virtual networks on top of a physical tunneling network. It connects the compute nodes with a dedicated network node. The network node creates internal bridges that will virtually route traffic through the physical network and deliver it to the virtual environment's network port that is handled entirely inside the compute node. For securing the virtual networks, Linux iptables are used. The current prototype uses OpenStack *Juno* on Ubuntu 14.04 servers. All management services are installed on the control node, except the compute services and the network service, that run on dedicated servers. To date, one control node, one network node and four compute nodes are provided. OpenStack provides two built-in mechanisms for load balancing. The Nova Scheduler handles hardware resources provided by the connected compute nodes and uses filter rules to decide which node is chosen to run a virtual server. The second mechanism is the Heat API, integrated into OpenStack since the Havana release[1]. Heat is able to run a stack of one or more virtual servers for a given application, based on a configuration file. If the load on one virtual server is too high, Heat can start a second machine to split up the load. Since OpenStack is designed for huge corporate clouds, both mechanisms assume that the infrastructure provides unlimited resources. In particular in private clouds, as employed in the presented platform, resources are often limited. A request to start a new virtual server fails and produces an error message, if there are not enough physical resources available. Virtual computation instances that can be launched by the cloud customer are referred to as *virtual servers* in this manuscript. We distinguish regular *virtual machines* from *Docker containers*, which can both be used as virtual servers in OpenStack. For testing new features, a test environment using *Devstack* is employed [38]. Devstack provides an install script for deployment of a complete OpenStack cloud infrastructure on a single physical or virtual machine. A Vagrant-based automated install script for virtual Devstack instances with Docker integration has been created for easy reproducibility and is available on GitHub [39, 40].

Docker. The employment of full virtual machines has considerable performance drawbacks in the given use case, as is shown in the results in Sect. 4.4. Successors of that approach are Docker [27] based Linux containers. They have been shown to perform better than other virtualisation concepts by Estrada and others as well as Felter and coworkers [28, 29]. Docker is a fairly new technology – the first release was March 2013 – but is based on the much older concept of operating system-level virtualisation. This special type of virtualisation is

[1] `https://wiki.openstack.org/wiki/Heat`

TABLE 3.1
Structure of the CMS API.

Method	Path	Description
GET	/files	Lists all files
GET	.../matlabs	Lists all Matlab files
POST	.../matlabs	Adds a new Matlab file
GET	.../file.m?compile	Compiles the specific Matlab file
DELETE	.../file.m	Deletes the specific file
GET	.../dicoms	Lists all DICOM files
GET	.../executables	Lists all executables
GET	.../binary?execute	Executes the specific binary

possible in Linux systems providing the Kernel features *control groups* and *namespaces* [30]. Both features are part of the Linux Kernel since 2008 and can be regarded as stable [31, 32]. Docker uses the term *container* for isolated environments, i.e. a control group with a namespace. A container has its own process tree, network interface (and so an own IP, routing table and iptables rules), file system and resources, such as memory, CPU and I/O. Docker adds some useful features to Linux containers: Dockerfiles holding a sequence of instructions to assemble a container image, versioning of images, a REST API, and in particular a driver for OpenStack.

Since the Havana release, OpenStack provides a hypervisor driver for Docker that is expected to be fully integrated into Kilo, the next major OpenStack version [34]. With this driver, Nova treats Docker containers like virtual machines, making it possible to use them in the same way as regular virtual machines.

3.1.2. Platform as a Service.

Code Management Service. The Code management service (CMS) is the main interface for the algorithm developer to upload, test and provide their Matlab code as stand-alone executables. The graphical user interface is implemented as a single page web application which is loosely coupled to a RESTful web service in the back end. This API offers basic file management, such as to read, delete or create files, as well as to compile and execute them. The whole functionality of the code management service is given in Table 3.1. The services are implemented as a PHP application that runs on an Apache Webserver, secured by password based authentication and HTTPS.

3.1.3. Software as a Service. The execution of the quality assessment analysis is managed by different services and components within the system.

XNAT. The main entry point for the clinical researcher is XNAT, where he or she can upload image data and start the quality assessment analysis. XNAT is implemented as a Java Enterprise web application, employing PostgreSQL as database system. The experiments presented in this paper are carried out on an XNAT 1.6.2.1 instance, using Apache Tomcat 7, Oracle JDK 1.6 and PostgreSQL 9.1 on a Debian 6.0 server instance. The data structures are designed to support typical research collaborations: researchers, projects, subjects, experiments etc. Data may be shared between different projects, and users can be assigned to the project with fine grained access rights. While especially supporting DICOM images and reports, all kinds of data types can be stored as freely definable key-value pairs (where the key is a name and the value a number), which is used in this project to implement the different quality management parameters. New parameters can be added at any time.

Besides the graphical user interface, a built-in DICOM interface and a comprehensive REST API [25] are provided. XNAT provides a pipeline engine, that can execute external applications and shell scripts. It may pass all required parameters to the application to allow for download the data, process it and upload the results back to XNAT. The pipeline engine is used within the presented infrastructure to initiate cloud-based processing. However, due to certain limitations of the built-in pipeline engine, in particular sequential processing of pipelines, the actual monitoring and scheduling of the cloud job is taken over by a dedicated cloud-aware job management service.

Image Upload. To ease the image upload for the clinical researcher, a dedicated image gateway is installed within the clinics. It receives images from the OCT scanner, blurs some patient related metadata of the OCT

images to minimize the reidentification potential, compresses the images and sends them via the REST interface to the external XNAT instance. Once the images are received by XNAT, it extracts the data and stores them in the prearchive. Two further steps are automatically executed: (1) The images must be marked as "ready" for archiving, before they are moved from the prearchive into the archive; (2) Once arrived at the archive, the images can be processed.

Job Management Service. The execution of the cloud-based image analysis methods is carried out by the job management service (JMS). It launches virtual servers depending on hardware availability, initiate job execution and terminate the servers when the job is finished. A particular feature is resource-aware scheduling. Because the built-in OpenStack tools do not provide functionality to resolve issues with limited resources, a lightweight Python-based cloud scheduler has been developed. It provides a REST API to enqueue incoming jobs and processes them whenever there are free resources. The scheduler is also able to limit the amount of virtual servers running simultaneously and to reuse them several times for different jobs if needed.

QMROCT VM image and Docker container. The quality assessment software itself is delivered within specifically configured virtual servers. In particular, the Matlab Compiler Runtime 2013b and the currently provided Matlab executables themselves are installed. We would like to emphasise, that the usage of Matlab standalone applications, i.e. compiled Matlab code, allows to deploy the code multiple times even in public clouds without licensing issues, as the free-of-charge Matlab Compiler Runtime is used. Furthermore, SSH is used to access the server. The QMROCT VM image is based on the official Ubuntu 14.04 cloud image [36]. It incorporates the Cloud-Init package [37], providing a lot of functionality to OpenStack, like resizing the image partition and inserting a public key for SSH authentication. On the downside, this results in a slower boot process for the virtual machine.

The QMROCT Docker container is based on a standard Ubuntu 12.04 image. It also contains the Matlab Compiler Runtime and executables, but not the Cloud-Init package, because a Docker container does not need features like repartitioning. Therefore the Docker container only runs an SSH daemon. A public key for authentication cannot be copied to the instance on start-up without Cloud-Init. To overcome this issue Docker is configured to initialize a key, when building the container image.

3.2. Security measures. The security assessment of the system follows the so-called *IT-Grundschutz* (IT basic protection) methodology of the Federal Office for Information Security (BSI Standard 100-2) [41], which interprets the requirements of the ISO Standards of the ISO 2700x family [42]. The basic idea of IT-Grundschutz is, that most IT systems can be modelled as a combination of well-known building blocks like servers, clients, network components and applications. For each of these building blocks, there exist typical security risks and organizational and technical security recommendations, that are collected in the *IT-Grundschutz* catalogues. The first step in such a security assessment is the analysis and documentation of the addressed business processes, encompassing the involved data, human actors and applications. The information processed is then categorized in different *protection levels*, depending on the possible harm due to loss of confidentiality, integrity or availability of the respective information. Now the related IT components are identified and are mapped on the existing building blocks. A subsequent basic security check compares the target and current state of the system regarding the security recommendations. A work plan is then defined to efficiently implement higher levels of information security.

4. Results. The resulting quality assessment platform provides a separated environment for clinical researchers and code developers. It is analysed in terms of functionality, usability, security and performance.

4.1. Functionality. Both the OCT quality assessment initialized by the clinical researcher as well as the integration of new image processing methods by the computer scientist were successfully implemented. An overview of the architecture is given in Figure 4.1. The so-called QMROCT Gateway is the central component to link XNAT to OpenStack through the JMS service. It also provides the code developer interface. The system's functionality is described along the two user scenarios, the quality assessment and the code upload.

4.1.1. Quality Assessment. The process of quality assessment on new images is shown in Figure 4.2. The clinical researcher uploads new images to XNAT. XNAT can be configured to trigger the quality assessment pipeline automatically, whenever new images have been uploaded, or to start the pipeline only on user interaction. XNAT's built-in pipeline system calls a small program, which sends the corresponding XNAT session

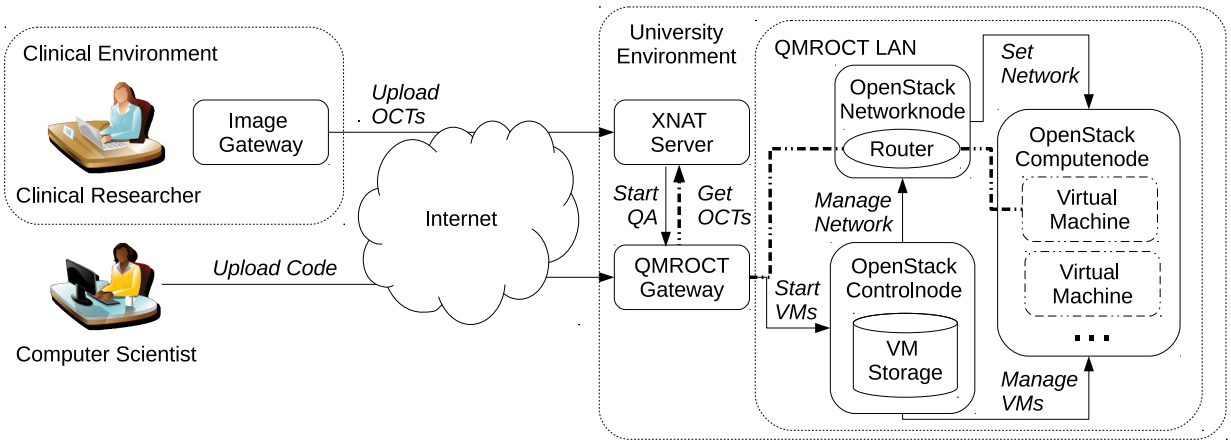


FIG. 4.1. *Architecture of the QMROCT infrastructure.*

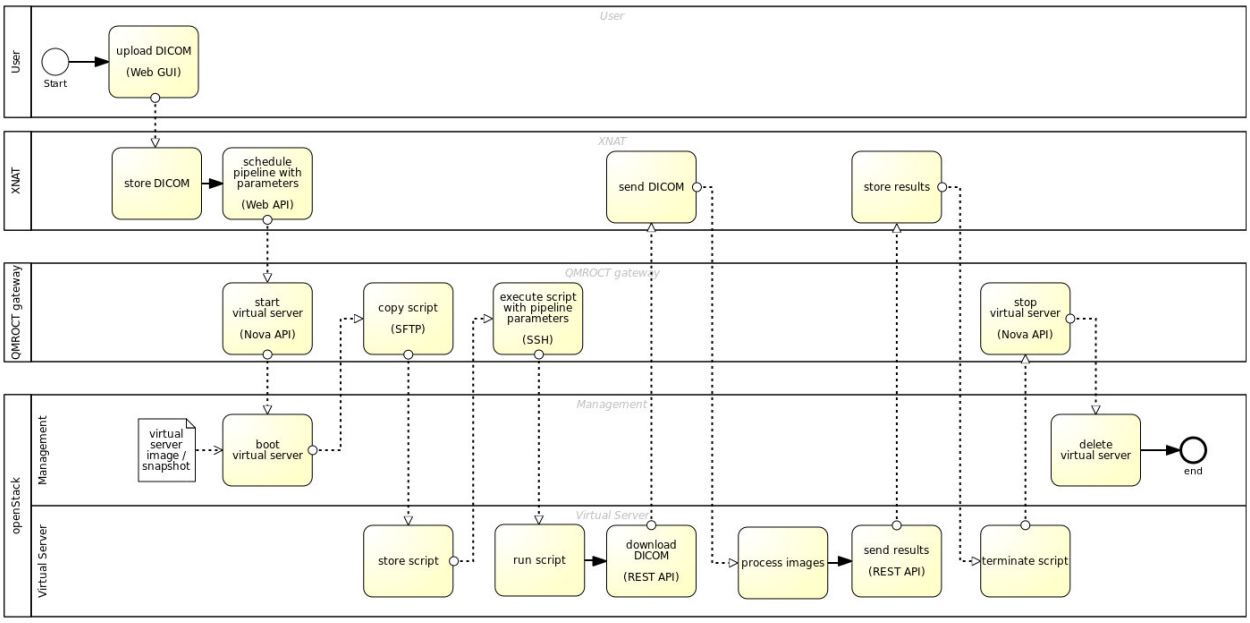


FIG. 4.2. *Process of the clinical researcher scenario: Quality assessment of new OCT images.*

parameters as JSON object to the JMS. The JMS handles the further job processing with the resources provided by OpenStack. After launching a virtual server, the QMROCT shell script is copied to the server and is executed with the given pipeline parameters via the SSH interface. The QMROCT shell script then connects to the XNAT server specified in the pipeline parameters and downloads the OCT images. These images are processed with the MATLAB executables present on the virtual server; and the results encompassing the numeric *Key Quality Indicators* and a DICOM result image are uploaded to the corresponding XNAT image session. When the processing is finished, the script terminates and the JMS shuts down the virtual server.

4.1.2. Code Upload. The process of uploading and testing code is shown in Figure 4.3. The software developer uploads new Matlab scripts to the CMS using the web interface, that is shown in Figure 4.4. Single

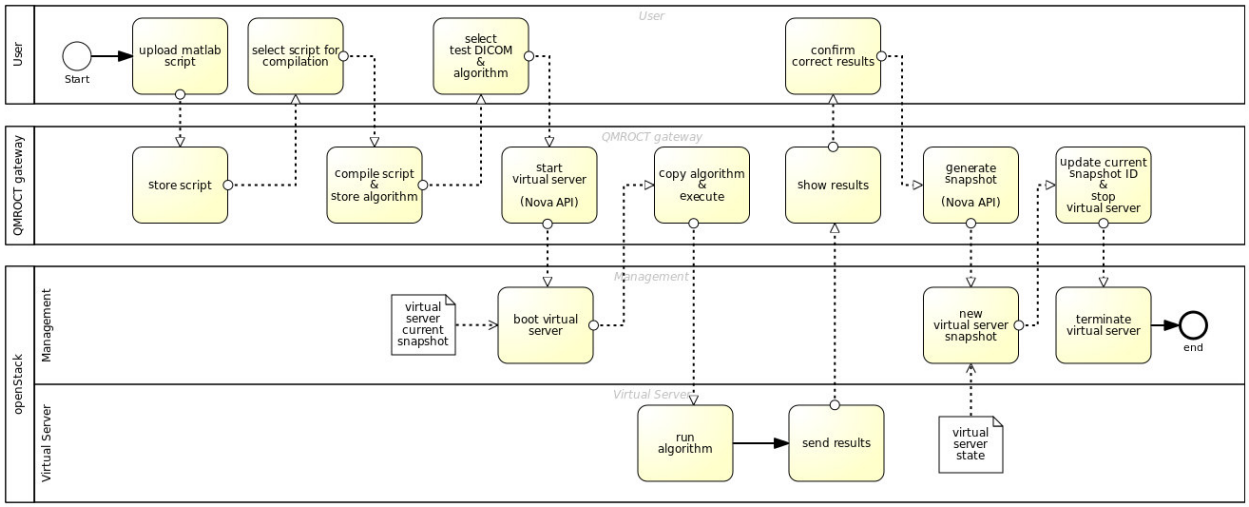


FIG. 4.3. *Process of the algorithm developer scenario: Compilation and testing of new Matlab code.*

scripts or zip-archived folders containing script collections are supported. These scripts are then stored in a local Git repository for version control. He or she may then select a script for compilation. When successfully compiled, the executables are added to the executable archive on the gateway, and a Matlab-generated run script for the new executable is shown in the list of available methods. Every algorithm needs to be tested before it can be used for quality assessment: The user selects a run script, chooses – depending on the method – one or more test images and further parameters and starts the execution manually. Then the new method is executed on the server and results or possible errors are displayed in the user interface. Key quality indicators are shown in the user-interface, the DICOM result image can be downloaded following the given link. When no errors occurred and the results are identical to those in the local environment, the developer may confirm the results, and the new method is integrated into the QMROCT virtual server.

4.2. Usability. The user interaction within the QMROCT platform is designed to be very simple. Within XNAT, the cloud-based processing is completely transparent and appears to be a standard XNAT pipeline. The clinical researcher accesses the cloud services and resources within his or her daily working environment. The only step that users need to do is selecting the image data to be processed and then submit the job, which will automatically start in the background. No use of command-line based tools, no manual installation and configuration of software clients or even complicated input of technical configuration data are needed by the clinical researcher.

With the CMS, several scripts can be uploaded simultaneously by simply dragging and dropping them onto the upload area. Timestamps are added to the scripts and executables to give the user an overview of the status of the uploaded code. The only convention, the developer has to follow, is to add a specified signature line to the Matlab script that includes the name of the function and – based on a mutually agreed list – the type of every parameter. The signature line is parsed by the upload tool to offer the user – after he or she selected an executable – a list of all available parameters, pre-filled with data based on an educated guess, e.g. a list of test DICOM images or the default DICOM dictionary file. The results of the code execution will be displayed directly in the web interface and can be compared to the local reference data. If a developer deems an algorithm mature, he or she can approve it for production usage. The whole process is completely independent from XNAT and does not interfere with the work of the clinical researchers.

Regarding the extensibility of the platform for the system developers, the strict service oriented architecture (SOA) offers an easy-to-use solution to add new services into the system, for example further user interfaces or additional cloud services.

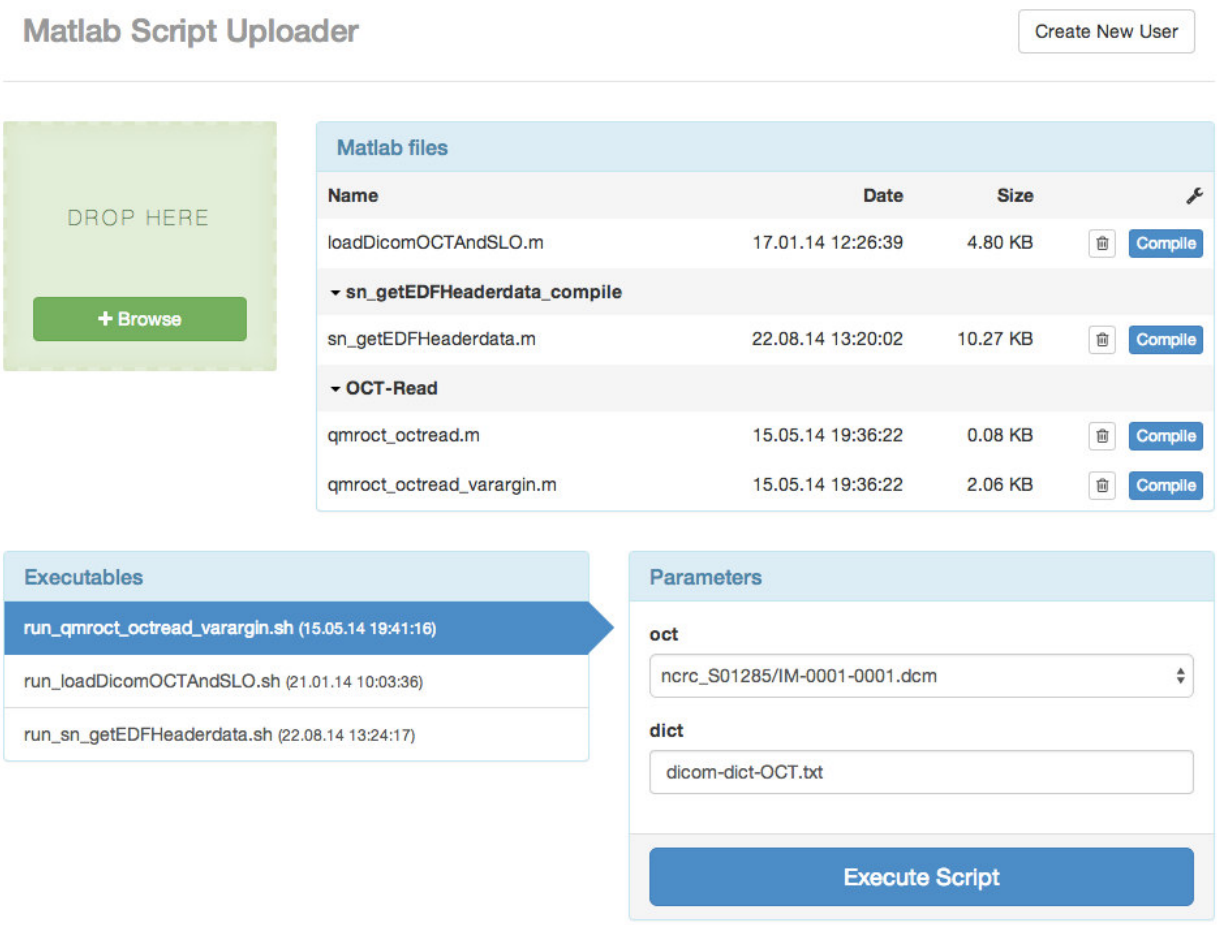


FIG. 4.4. *CMS Web Interface to upload and test new Matlab code.*

4.3. Security. Following the IT-Grundschutz methodology, the processes and involved systems are identified as described in this manuscript (cf. Sect. 3.2). The different user roles identified within this system are shown in Table 4.1. The processed information includes possible metadata, code, and user activities. The confidentiality level is medium, rather than high, as the image data are pseudonymised. The same holds for the availability level and the integrity level. To mitigate the consequences of a system breakdown, all image data processed within the system are also stored locally in the researcher's PACS and the source code is also available in the developer's local environment. So typical security measures are supposed to be sufficient for the system.

4.3.1. Confidentiality. As mentioned before, all communication through the internet is encrypted via TLS, encompassing user and machine access. Within the OpenStack infrastructure, no component is directly connected to the internet. In particular, the virtual servers run in their private network and may connect to the internet via a virtual router on the network node and a subsequent router on the QMROCT gateway. The so-called "floating IP" address, which is assigned to a virtual server, can only be accessed from the QMROCT gateway. An OpenStack security group restricts this access to allow only SSH. Any virtual machine or container can initialize an internet communication with XNAT, but they cannot be called from outside. Due to usability reasons, PKI-based user authentication and authorisation is not employed. As cloud infrastructures are designed for multi-tenant use rather than collaborative resource sharing, the separation between different users is much

stronger than in typical academic Grid infrastructures, lowering the risk of information leaks. All code is executed within the virtual environments which are always instantiated for one specific task and terminated afterwards, so no information about former activities is stored there. From a security perspective, the Docker containers behave similar to virtual machines, as they shield different instances from each other. A container can not manipulate another, because it only knows about it if declared explicitly a priori, i.e. by linking containers. As of now there is no known way for a process to break out of a Docker container, but a potential risk is posed by the Docker daemon as it has to run with root privileges. There is work in progress to change namespaces to run a container as an unprivileged user [33], lowering the mentioned risk. On XNAT and OpenStack, the built-in password-based authentication and role-based access control methods are used (cf. Table 4.1).

4.3.2. Integrity. The risk of malicious manipulation of data can be regarded as comparatively low. As data transfer is encrypted, man-in-the-middle attacks are difficult, and all user activity on the different components – the XNAT and the CMS/JMS services – are logged. A higher risk might be the unintended assignment of image data and results to a wrong subject. As the virtual servers processing the image data only know the parameters of the assigned pipeline, it has no information about other resources from XNAT. Therefore it is very unlikely, that QM results are stored along the wrong subject or image data. In case a mismatch would occur, it can be identified by the overview image that is also provided as result. The code, the results and the virtual machine snapshots are under version control, so in case of undesired modification, a former version can be reconstructed.

4.3.3. Availability. Main risks for availability are network interruptions and server failures. The production network infrastructures of the participating institutions are employed, leading to very few network problems. To avoid availability issues because of the code execution – may it be due to high CPU load or due to malicious or unstable code – all processing steps are transferred to the cloud environment. In addition, the use of virtual servers increases the stability of applications, since the system's configuration is largely independent from the host system of the cloud provider. In case of a successful attack on a virtual server, it is hardly possible to get access to the host system, so there is very low risk that the platform and application availability is affected. To account for possible data-loss, the XNAT platform is integrated into the institution's backup system, while the code stored on the QMROCT gateway is duplicated in the QMROCT VM and Docker images.

4.4. Performance. Performance tests have been carried along the typical clinical researcher's use-case of the infrastructure: At the end of an examination day, the images are uploaded to the infrastructure and are processed. The tests are performed with 12 OCT image sets. Each image set consists of an optical overview image and a tomographic scan of the retina. The overview images have always a size of 580 kB, the tomographies' sizes range from 14 MB to 27 MB; adding up to 294 MB for the full dataset.

Currently, three quality assessment methods are implemented within the infrastructure, encompassing poor signal strength and centration of the scan. We would like to emphasise that the algorithms are not developed by the authors but by the project partners (cf. 6). Method 1 and method 3 each provide a result image visualising the regarding quality indicators, method 1 and method 2 provide key image indicators in form of key-value pairs. The result images' sizes range from 700 KB to 2 MB, the key quality indicators are written in a text file, encompassing few bytes. Each full processing from pseudonymisation of the image to quality assessment result storage back to XNAT has been performed ten times. As the quality assessment pipeline could not be started remotely via the REST API, the tests are subdivided into the upload process and the quality assessment process.

Image Upload Process. The image upload process encompasses five different steps (cf. 3.1.3). The duration of the different steps are given in Figure 4.5.

The whole image upload process requires less than 2 minutes in all runs, with an average runtime of (94 $\pm$ 6) seconds. The main contributions are the data compression and transfer. Interestingly, decompression on the XNAT server is much faster. The data transfer took in average (65 $\pm$ 6) seconds, giving a data transfer rate about 4.33 MB/s. Compared to other modalities, OCT images are small, so data transfer from the clinics to the quality assessment infrastructure will not be a severe performance issue. But with an estimated data transfer of about 350 GB per day, the current setup does not support medical imaging methods that generate data in a magnitude of terabytes.

TABLE 4.1
Defined user roles with their permission in the QMROCT infrastructure.

Role	Rights	Data Access
End User		
Clinical Researcher	Can upload images, execute analysis jobs and search images via XNAT web portal	Images inside her or his project and analysis results
Computer Scientist	Can upload new image analysis algorithms via gateway server	Source code and compiled code on the QMROCT gateway
Administration		
XNAT Administrator	Can change configuration of XNAT and activate/deactivate XNAT and user access to fix bugs or improve system safety	Log files of the application, configuration data, data of portal users, no access to application data
Server Administrator	Can update and reboot the server and services	Information about users, processes and images stored on the file system
Application Administrator	Can change application configuration, activate/deactivate components of the application	Log files of the application, configuration data, no user data
Cloud Administrator	Can change cloud components and user management and deactivate/start cloud services to fix bugs and improve system safety	Log files of component configuration data, data of user management, no access to application data and configuration data (image data) of cloud services
Development		
JMS/CMS Developer	Can change code and job management services and user interface to fix bugs or improve performance of the application	Data necessary for the execution of application functionality, possibly log files, access to user data or image data stored within the service
XNAT Developer	Can change the application code to fix bugs or improve performance of the application	Log files of the application, configuration files, possibly data contents related to application performance, no personal data access
Cloud Service Developer	Can change services and update libraries to fix bugs or improve performance of the services	Log files of the service, configuration files, possibly data contents related to application performance, no personal data access, boot access to hardware

Quality Assessment Process. The quality assessment process is realised using two OpenStack compute nodes with Intel(R) Core2 Duo (3 GHz) CPUs. One compute node is equipped with 8 GB RAM, the other one with 4 GB RAM. The VMs described in paragraph 3.1.3 are instantiated with 2 GB of RAM. The JMS is configured to allow a maximum of four VMs at a time. We would like to emphasise that overloading the compute nodes is possible, but the current bottleneck is the provision of floating IPs, which may fail if more than four IPs are requested at the same time. The overall runtime of the quality assessment process is shown in Figure 4.6.

The whole quality assessment process requires in all runs less than 16 minutes, with an average runtime

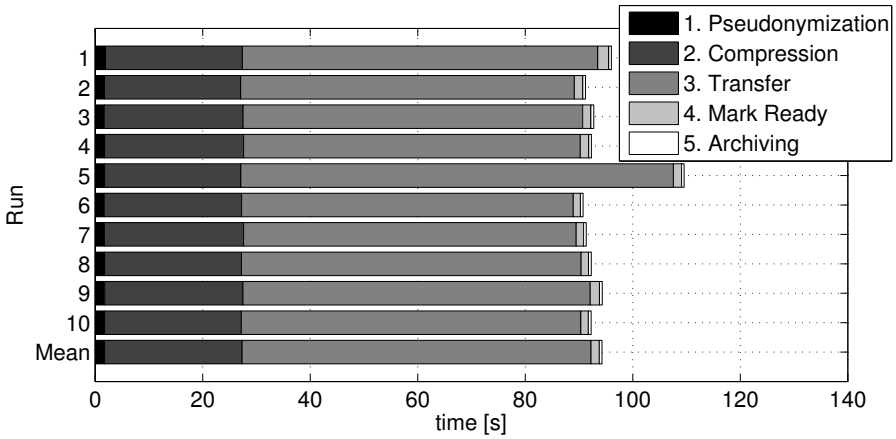


FIG. 4.5. *Runtime of the image upload process of 12 image sets with a total size of 294 MB.*

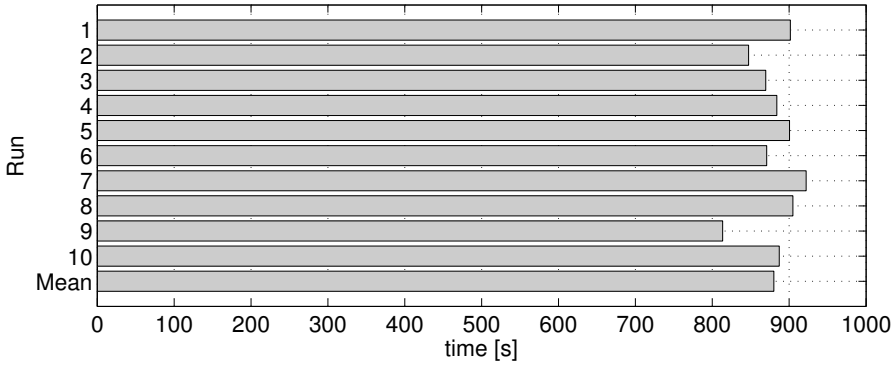


FIG. 4.6. *Runtime of the quality assessment process of 12 image sets with a total size of 294 MB and three image processing methods.*

of 14.6 minutes, or (880 ± 30) seconds. So the whole image pipeline from pseudonymisation to result storage requires less than 20 minutes for all 12 images.

To get a closer look into the execution times of the different processing steps, Figure 4.7 shows the runtimes of all individual quality assessments.

The quality assessment for a single image set requires always less than 13 minutes, in average about 5 minutes or ((280 ± 46) s). Compared to the average runtime of the complete set of 12 image sets, the distributed system provides a speed-up of factor 3.1 compared to sequential processing. A maximum limit of 4 is given by the current settings of the scheduler. The processing itself with an average of (190 ± 34) s takes the main part of the overall runtime, followed by the instantiation of the VM ((76 ± 18) s). The data transfer within the system requires an average of (12 ± 5) s for each individual image set. Figure 4.8 shows the mean runtimes for the individual image sets.

The differences in runtime might on one hand be caused by image properties, like the image size, which would effect the data transfer and possibly the execution times of the quality assessment methods; or intrinsic image features like the signal-to-noise-ratio, which might affect the execution times of one or more quality assessment methods. Table 4.2 shows the different image sizes of the processed images.

As can be seen by comparison with Figure 4.8, there is no obvious correlation between the image size and the data transfer time. On the other hand the runtime might be influenced by the availability of cloud resources. In particular, we see an increasing prolongation of the runtime for images 2, 3, and 4. We assume, that this is caused by the simultaneous start of the first four virtual machines. Later, due to differences in the execution times of the quality assessments, the resource allocation is desynchronized for the four concurrent processes, so

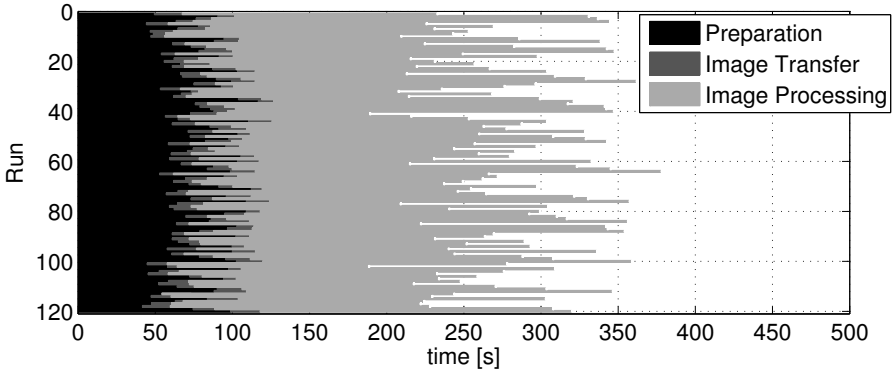


FIG. 4.7. *Runtimes of the individual quality assessments for single image sets.*

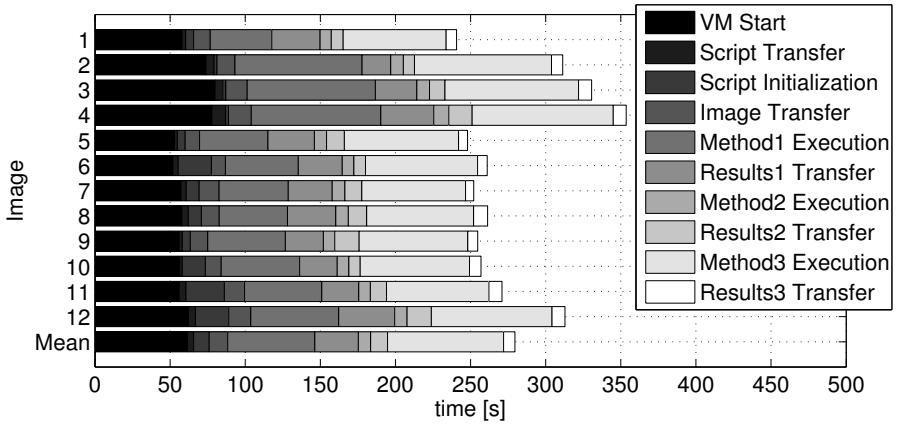


FIG. 4.8. *Average runtimes for the quality assessment of the different image sets.*

VM instantiation and data transfer is better spread in time.

Docker containers. While performance optimisation of the algorithms is up to the algorithm developers, on the system side the starting of the virtual machines takes the main part of the management overhead. This performance could be significantly increased by using Docker containers. To compare the performance of Docker containers and virtual machines, additional tests have been realised. These performance tests have been carried out within a single virtual Ubuntu 14.04 Devstack machine equipped with 8 GB RAM and four virtual CPUs assigned. The OpenStack version used with Devstack is *Icehouse*, the last release before *Juno*. For each test, ten strongly simplified data processing pipelines are scheduled at once. They are then processed in sequential or in parallel mode. Parallel mode schedules the pipelines up to a fixed number of two simultaneously running jobs. As the focus of this test was on the mangement overhead, no data transfer nor image processing was included, but a test script writing a string to a file was employed. For each pipeline a virtual server is booted and a small test script, which only writes a string to a file, is copied to this server and then executed. For every test the total time for processing all ten pipelines and the time each virtual server is running have been measured. The virtual server runtime is measured from booting the server until the test pipeline script has been executed and terminates. Every test is carried out five times. The results are given in Figures 4.9 and 4.10. We would like to emphasise, that the results cannot be compared directly with the performance tests above. As the Docker support in OpenStack is still in active development, we performed these tests in a test environment.

As can be seen in Figure 4.9, the system scales well, but not linearly. While the average total time for a sequential execution of ten runs with regular virtual machines is about 47 minutes, parallel execution reduces the execution time to 27 minutes (about 60%). Using Docker, the average total time is close to 5 minutes

TABLE 4.2
Image sizes of the tomographic images.

Image	1	2	3	4	5	6	7	8	9	10	11	12
Size [MB]	27	23	23	23	27	27	23	14	23	27	23	23

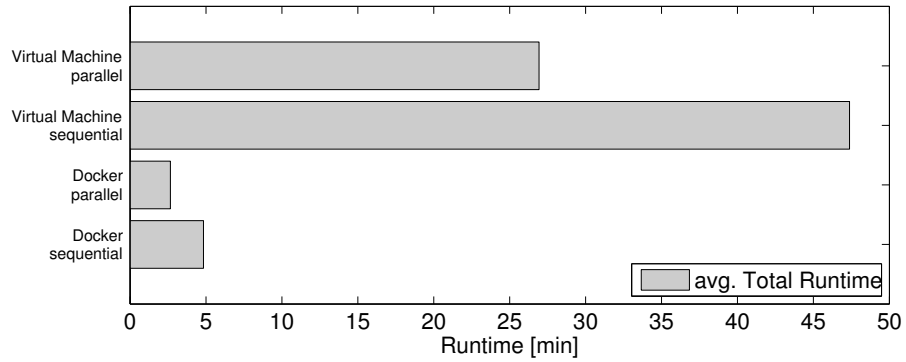


FIG. 4.9. *Performance tests comparing the average total time of executing 10 pipelines in virtual machines and Docker containers, each in sequential and in parallel mode.*

in sequential and about 2.6 minutes in parallel mode, requiring about 10% of the time with respect to the respective tests using virtual machines.

The average runtime for each virtual server is shown in Figure 4.10. For both, virtual machine and Docker container, the runtime is slightly longer in parallel mode compared to sequential execution. This measurement correlates with the results from Figure 4.9, where the parallel execution requires slightly more than 50% of the sequential execution. It can be explained by the inherent resource sharing: The parallel execution uses more computing capacities, which results in a slightly slower execution of an individual run.

5. Conclusion and Outlook. The envisioned quality assessment platform for OCT images has been set up successfully. The system is currently used by the clinical researchers from the local university clinics participating in the QMROCT project. In order to increase functionality and usability and to minimize specific security problems of the neuroimage processing, we have integrated existing and newly developed platforms and components to the QMROCT environment. Both the image processing pipeline as well as the test execution of new code requires only few interaction steps with the user: The quality assessment requires the data and pipeline selection, the code testing requires code and data selection and optional parameter settings, the results are automatically transferred to the respective target component and user interface. The QMROCT platform has been analysed with respect to IT security and many recommendations from the *IT-Grundschutz* catalogue have been implemented. However, as the specific applications cannot be modelled with standard building blocks, and as new security vulnerabilities in underlying services and tools are reported regularly, system security is a continuous process. Anyhow, the most vulnerable step regarding patient data protection, the automatic upload of image data from the clinical environment is not yet fully employed and is ongoing work.

According to the performance results, the quality assessment of a typical examination day can be processed within 20 minutes. Currently, three quality assessment methods are available. However, as new methods are continuously integrated, the processing time will eventually increase. On the infrastructure, the instantiation of the virtual machines takes the largest share of the management overhead. The employment of Docker containers significantly reduces the management overhead to about 10% with respect to virtual machines. Reuse of virtual servers in large-scale image processing is envisioned, but is to be examined regarding confidentiality, as it would violate the disposable-virtual-server approach, that currently wipes any image-specific information from the cloud infrastructure directly after job execution. The integration of Docker into the production environment is currently realised, so full production-like performance tests need to prove these findings. Furthermore, the employment of about 40 computers from student's labs as OpenStack compute nodes is envisioned, allowing

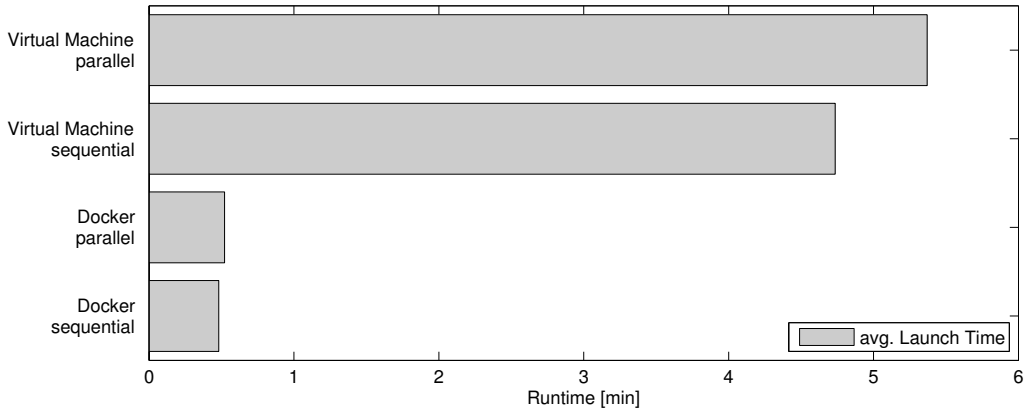


FIG. 4.10. *Average launch time of each virtual server during the performance tests.*

for faster processing of large data sets. While the JMS is capable of handling limited resources and common errors, a descent workflow manager monitoring the jobs to give detailed information about process status, cloud related errors or wrong usage of resources is to be implemented. Many workflow management tools like Pegasus [43], MOTEUR [44], Taverna [45] or WS-PGRADE were developed for grid infrastructures and are continuously supporting more and more cloud infrastructures. OpenStack itself recently published a new workflow management service called Mistral [46], that we will evaluate. We would like to emphasise that the QMROCT platform – except the XNAT extensions for the quality measure data – is not research-domain specific. In particular, other science gateways might be connected to the JMS and CMS. Currently, further neuroimage and biosignal processing scenarios are implemented using the same infrastructure.

6. Acknowledgements. We would like to thank all participants of the QMROCT project, in particular Frank Haußer and Inge Beckers from Beuth University of Applied Sciences Berlin for leading the quality assessment algorithm development and Alexander Brandt, Timm Oberwahrenbrock, Hannah Zimmermann and Ella Kadas from the Neuroscience Research Center of Charité - Universitätsmedizin Berlin for providing the OCT images and medical expertise.

7. Source Code. The developed components are open source projects and available on the QMROCT GitHub page:

1. Pipeline-Scheduler [35] (GNU General Public License)
2. Vagrant Box including Devstack with Docker integration [40] (MIT License)
3. Docker Container including Matlab and an SSH daemon [47] (MIT License)

REFERENCES

[1] A. PETZOLD ET AL., *Optical coherence tomography in multiple sclerosis: a systematic review and meta-analysis*, Lancet Neurology, Sep. 9, 2010. Accessed: Sep. 11, 2014. Available: http://www.thelancet.com/journals/laneur/article/PIIS1474-4422%2810%2970168-X/abstract

[2] A. U. BRANDT ET AL., *Primary retinal pathology in multiple sclerosis as detected by optical coherence tomography*, Brain, Feb., 2013. Accessed: Sep. 11, 2014. Available: http://brain.oxfordjournals.org/content/134/11/e193.extract

[3] B. FISCHL, *FreeSurfer*, Jan. 1, 2012. Accessed: Sep. 10, 2014. Available: http://www.sciencedirect.com/science/article/pii/S1053811912000389

[4] *FreeSurfer Beginners Guide*. Accessed: Sep. 10, 2014. Available: http://freesurfer.net/fswiki/FreeSurferBeginnersGuide

[5] P. TEWARIE ET AL., *The OSCAR-IB consensus criteria for retinal OCT quality assessment*, Apr. 19, 2012. Accessed: Sep. 11, 2014. Available: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0034823

[6] *DICOM*. Accessed: Feb. 9, 2014. Available: http://medical.nema.org/standard.html

[7] D. S. MARCUS ET AL., *The extensible neuroimaging archive toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data*, Neuroinformatics, 2007, Springer. Accessed: Sep. 11, 2014. Available: http://link.springer.com/article/10.1385/NI%3A5%3A1%3A11

[8] J.-B. Poline et al., *Data sharing in neuroimaging research*, Apr. 5, 2012, Frontiers in Neuroinformatics. Accessed: Sep. 10, 2014. Available: http://journal.frontiersin.org/Journal/10.3389/fninf.2012.00009/abstract
[9] S. Seifert et al., *Semantic annotation of medical images*, Mar. 11, 2010, Medical Imaging 2010: Advanced PACS-based Imaging Informatics and Therapeutic Applications. Accessed: Sep. 10, 2014. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=748216
[10] Microsoft *Azure*. Accessed: Sep. 10, 2014. Available: http://azure.microsoft.com
[11] Amazon, *AWS, Amazon elastic compute cloud (EC2)*. Accessed: Sep. 10, 2014. Available: http://aws.amazon.com/
[12] Google, *Google Cloud Platform*. Accessed: Sep. 10, 2014. Available: https://cloud.google.com/
[13] OpenStack Foundation, *OpenStack*. Accessed: Sep. 10, 2014. Available: http://www.openstack.org/
[14] P. Mell and T. Grance, *The NIST definition of cloud computing*. Accessed: Sep. 10, 2014. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[15] J. Wu et al., *The Charité Grid Portal: User-friendly and Secure Access to Grid-based Resources and Services*, Oct. 12, 2012, Journal of Grid Computing, Springer. Accessed: Sep. 10, 2014. Available: http://link.springer.com/article/10.1007%2Fs10723-012-9234-3
[16] P. Kacsuk, *P-GRADE portal family for Grid infrastructures*, Mar. 10, 2011, Concurrency and Computation: Practice and Experience. Accessed: Sep. 10, 2014. Available: http://onlinelibrary.wiley.com/doi/10.1002/cpe.1654/abstract
[17] I. Foster et al., *A security architecture for computational grids*, 1998, CCS '98 Proceedings of the 5th ACM conference on Computer and communications security, ACM Press. Accessed: Sep. 10, 2014. Available: http://portal.acm.org/citation.cfm?doid=288090.288111
[18] I. Dinov et al., *Neuroimaging Study Designs, Computational Analyses and Data Provenance Using the LONI Pipeline*, September 28, 2010. Accessed: Sep. 10, 2014. Available: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0013070
[19] R. Siewert et al., *Web-based Interactive Visualization in a Grid-enabled Neuroimaging Application Using HTML5*, 2012, IOS Press. Accessed: Sep. 11, 2014. Available: http://ebooks.iospress.nl/publication/21428
[20] C. Fiehe et al., *Building a Medical Research Cloud in the EASI-CLOUDS Project*, Jun., 2014, Science Gateways (IWSG) 6th International Workshop, IEEE. Accessed: Sep. 10, 2014. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6882066&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6881322%2F6882053%2F06882066.pdf%3Farnumber%3D6882066
[21] T. Bednarz et al., *Cloud-based image analysis and processing toolbox for biomedical application*, 2012. Accessed: Sep. 10, 2014. Available: http://www.ci.uchicago.edu/escience2012/pdf/escience2012_submission_189.pdf
[22] R. Alonso-Calvo et al., *Cloud computing service for managing large medical image data-sets using balanced collaborative agents*, 2011, Advances in Intelligent and Soft Computing, Springer Berlin Heidelberg. Accessed: Sep. 10, 2014. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-19875-5_34
[23] B. Da Mota et al., *Machine learning patterns for neuroimaging-genetic studies in the cloud*, Apr 8, 2014, Frontiers in Neuroinformatics. Accessed: Sep. 11, 2014. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3986524/
[24] J.M. Salinas et al., *R & D Cloud CEIB: Management System and Knowledge Extraction for Bioimaging in the Cloud*, 2012, Advances in Intelligent and Soft Computing, Springer. Accessed: Sep. 11, 2014. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-28765-7_39
[25] L. Richardson et al., *RESTful web services*, 2007, O'Reilly.
[26] OpenStack Foundation, *OpenStack Havanna*. Accessed: Sep. 11, 2014. Available: https://wiki.openstack.org/wiki/Heat
[27] Docker, Inc. Accessed: Sep. 12, 2014. Available: https://www.docker.com/
[28] Z.J. Estrada et al., *A Performance Evaluation of Sequence Alignment Software in Virtualized Environments*, May, 2014, 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Accessed: Sep. 11, 2014. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6846525&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6846525
[29] W. Felter et al., *An Updated Performance Comparison of Virtual Machines and Linux Containers*, July 21, 2014, IBM Research Report. Accessed: Sep 10, 2014. Available: http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf
[30] J. Petazzoni, *Lightweight Virtualization with Linux Containers*, June 7, 2013, The 5th China Cloud Computing Conference. Accessed: Sep. 10, 2014. Available: http://www.ciecloud.org/2013/subject/07-track06-Jerome%20Petazzoni.pdf
[31] *Linux Kernel 2.6.24*. Accessed: Sep. 12, 2014. Available: http://kernelnewbies.org/Linux_2_6_24
[32] *Linux Kernel 2.6.26*. Accessed: Sep. 12, 2014. Available: http://kernelnewbies.org/Linux_2_6_26
[33] *LXC 1.0.5*. Accessed: Sep. 12, 2014. Available: https://linuxcontainers.org/news/
[34] OpenStack Foundation, *Docker, OpenStack Wiki*. Accessed: Aug. 30, 2014. Available: https://wiki.openstack.org/w/index.php?title=Docker&oldid=61664
[35] QMROCT, *Pipeline Scheduler*. Accessed: Sep. 10, 2014. Available: https://github.com/QMROCT/pipeline-scheduler
[36] Ubuntu Documentation Team, *Ubuntu Enterprise Cloud*. Accessed: Sep. 10, 2014. Available: http://help.ubuntu.com/community/UEC/Images
[37] OpenStack Foundation, *OpenStack Linux image requirements*. Accessed: Sep. 10, 2014. Available: http://docs.openstack.org/image-guide/content/ch_openstack_images.html
[38] OpenStack Foundation, *Devstack*. Accessed: Sep. 10, 2014. Available: http://devstack.org/
[39] HashiCorp, *Vagrant*. Accessed: Sep. 10, 2014. Available: http://www.vagrantup.com/
[40] QMROCT, *Vagrant based Devstack with Docker*. Accessed: Aug. 30, 2014. Available: https://github.com/QMROCT/vagrant-devstack-docker
[41] Federal Office for Information Security, *BSI-Standards*. Accessed: Sep. 10, 2014. Available: https://www.bsi.bund.

de/EN/Publications/BSIStandards/BSIStandards_node.html
[42] *ISO/IEC 27001:2013, Information technology - Security techniques - Information security management systems - Requirements*. Accessed: Sep. 10, 2014. Available: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54534
[43] K. Lee et al., *Adaptive workflow processing and execution in pegasus*, Nov., 2009, Concurrency and Computation: Practice and Experience. Accessed: Sep. 10, 2014. Available: http://onlinelibrary.wiley.com/doi/10.1002/cpe.1446/abstract
[44] T. Glatard et al., *Flexible and Efficient Workflow Deployment of Data-Intensive Applications On Grids With MOTEUR*, Aug., 2008, International Journal of High Performance Computing Applications. Accessed: Sep. 10, 2014. Available: http://hpc.sagepub.com/content/22/3/347
[45] K. Wolstencroft et al., *The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud*, May 2, 2013, Nucleic Acids Research. Accessed: Sep. 10, 2014. Available: http://nar.oxfordjournals.org/content/41/W1/W557
[46] OpenStack Foundation. *Mistral*, Accessed: Sep. 10, 2014. Available: https://wiki.openstack.org/wiki/Mistral
[47] QMROCT, *Docker container with Matlab and SSH*, Accessed: Sep. 10, 2014. Available: https://github.com/QMROCT/matlab-docker

Edited by: Jesus Carretero
Received: September 15, 2014
Accepted: January 21, 2015

DOI 10.12694/scpe.v16i1.1063
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 16, Number 1, pp. 103–120. http://www.scpe.org

A TOOL FOR MANAGING THE X1.V1 PLATFORM ON THE CLOUD

EMANUEL MARZINI, PAOLO MORI*, SERGIO DI BONA, DAVIDE GUERRI, MARCO LETTERE†, AND LAURA RICCI‡

Abstract. The X1.V1 platform is a service oriented architecture, deployed on a set of Virtual Machines, that integrates services for managing fundamental activities in the ehealth scenario, such as Patient Identification, Clinical Document Repository, Prescription, and many others. The efficient provisioning of such services to citizens and to health professionals requires the adoption of increasingly powerful computing systems, in order to ensure that the growing number of service requests are served in acceptable time, even in case of computational peaks. Porting the X1.V1 platform on the Cloud could address the problem of the computational requirement mutability, since Cloud elasticity allows the reallocation of resources to Virtual Machines when necessary. This paper proposes a framework for managing the execution of the X1.V1 platform on the Cloud. This framework enables an easy, quick, and secure management of the Cloud resources allocation and reallocation to X1.V1 Virtual Machines, in order to enhance the platform performances, optimize resource utilization and, consequently, reduce the whole services cost. The design of the framework is focused on security aspects as well, because unauthorized accesses could lead to serious inefficiencies of the ehealth services. Finally, besides describing the architecture design and implementation of the X1.V1 Cloud manager framework, this paper also presents a set of experimental results which confirm the validity of the proposed approach to solve the X1.V1 platform performance issues in distinct reference scenarios.

Key words: Health records, Cloud computing

AMS subject classifications. 68M14, 97M60

1. Introduction. The spreading of the Electronic Health Record (EHR) is changing the vision of the European health systems, moving from a vertical approach to solutions able to organize and manage entire healthcare processes based on the integration of care settings. In order to fully support this view, these infrastructures need to be oriented to support care policies expressed as effective clinical services to the citizen/patient, both from a management and an optimization of resources standpoint. In fact, care and assistance needs are changing today as a consequence of the social, economic and technological advancements occurred in the last decade. Healthcare spending has strongly increased in all industrial countries, and in Europe it will potentially climb up to about 15% of the GDP (Gross Domestic Product) by 2020. This growth is largely driven by the increase in demand due to the average life expectancy and, as consequence, to the number of people affected by chronic diseases which becomes more expressed with increasing age. This new situation has three different but interconnected perspectives:

1. the clinical services managers are challenged to deliver effective healthcare services at reasonable costs;
2. the clinicians are interested to apply and improve the effectiveness of clinical guidelines, increasing the emphasis on evidence-based medicine;
3. the citizens are increasingly demanding high quality and continuity of care, they are more aware of clinical risk management and are claiming more efficient healthcare services, with reduction of waiting times and a better utilization of resources.

In an increasingly digital economy, in order to face these challenges, the healthcare cannot continue to be a service where the user is passive. This trend is reflected in EHR infrastructures which are becoming increasingly arranged and enriched so that the citizen/patient can be more involved in her/his own care, gradually moving away from a passive role. At the same time, in order to guarantee continuity of care across different care settings it is necessary to provide healthcare operators with an integrated set of services and facilities for defining, monitoring, evaluating and fine-tuning personalized care programmes.

In order to make this transition technically and economically viable it is necessary to consider a definitive move towards new organizational models able to govern seamless integration across government, community and private information systems (EHRs, Personal Health Records PHRs, Telecare centres). Such models must be supported by distributed architectures where users, systems, applications, and devices are able to collaborate.

*Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy (paolo.mori@iit.cnr.it).
†Dedalus SpA, Livorno, Italy (name.surname@dedalus.eu).
‡Dipartimento di Informatica, Università di Pisa, Pisa, Italy (laura.ricci@unipi.it).

Service oriented architectures (SOAs), based upon largely accepted standards, allow heterogeneous systems that support different care environments interacting and combining personal, social and environmental issues that affect the citizen/patient's healthcare processes.

This scenario will progressively push the healthcare organizations to adopt more and more powerful computing systems to cope with the increasing workload related to the management of the new ehealth services. Moreover, huge repositories are required for managing the very large and continuously increasing number of clinical documents. In addition, as a consequence of the availability of more and more complex services, the computational requirements can change significantly from one service to another, or they can even change during time. For example, the computational requirement of the prescription process, starting from the General Practitioner (GP) that prescribes a drug or a treatment up to the drug delivery or administration, is likely to be high during daytime, i.e., during working hours of the GPs, but most likely it is low during the night when only emergency doctors use this service. As a counter example, services related to audit and control processes are more likely to run during the night. By adopting Cloud-based paradigms, healthcare organizations as well as complex systems could receive significant benefits from several points of view. In particular, one of the essential characteristics of a Cloud system is the elasticity [14], that is defined by NIST as: "Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time" [20]. Cloud elasticity is a crucial feature in our scenario, since it can help healthcare organizations to cope with the mutability of the computational and storage requirement of the ehealth systems and services. With the main goal of validating the approach proposed in this paper, we have taken into account a commercial platform, distributed by Dedalus SpA[1] (an Italian company which is national leader in healthcare software), aimed at guaranteeing the interoperability among the different interacting actors of the healthcare scenario. The X1.V1 platform is a service oriented architecture, deployed on a set of Virtual Machines (VMs), that integrates services for managing fundamental activities in the ehealth scenario, such as Patient Identification, Clinical Document Repository, Operator Profiling, Access control and authorization, as well as Prescription and Reporting processes.

This paper presents a Cloud Management System, named X1.V1 Cloud Manager, which allows the execution and the management of the X1.V1 platform on a Cloud system through a simple graphic interface and is able to change the allocation of the Cloud resources to the X1.V1 services in just one click in order to promptly react to a computational peak, or according to a predefined scheduling to face a forthcoming expected overload. These activities, in fact cannot be performed manually since it would be difficult, time consuming and error prone for the system administrator. The X1.V1 Cloud manager is able to manage the VMs that composes the X1.V1 platform and to change the overall resources allocation following some predefined configurations. Each configuration represents the optimal allocation of the Cloud resources to obtain the best performances of the platform in a specific usage scenario. Switching from one configuration to another has a great impact on the X1.V1 platform performance, and switching to the wrong configuration could even result in a denial of service in some critical situations. Hence, the access to the X1.V1 Cloud Manager must be protected to avoid unauthorized and inappropriate reallocations of Cloud resources. In particular, at least two distinct access levels must be defined: one that allows the system administrator to select a subset of the existing configurations (e.g., the ones that do not involve additional costs for the customer), and another that allows the system administrator to switch to any configuration, and to upload customized configuration to face with unusual or emergency scenarios. Summarizing, the main feature of the X1.V1 Cloud manager is the easy exploitation of the Cloud elasticity, while providing adequate security to protect the processes of the ehealth services.

This paper is organized as follows: Section 2 gives a brief overview of the X1.V1 platform, Section 3 describes the main reasons for executing the X1.V1 on the Cloud, and Section 4 describes some related works. Section 5 describes the main features of the X1.V1 Cloud Manager, Section 6 gives some implementation details and describes a set of experiments, and Section 7 concludes the paper.

2. The X1.V1 platform. In order to support the interoperability among the different care environments (hospitals, primary care, homecare) and to provide the healthcare operators with mechanisms for implementing

[1] http://www.dedalus.eu/

integrated care pathways, starting from 2005 Dedalus defined, designed and implemented the interoperability platform, namely X1.V1 [18]. In particular, X1.V1 was designed to address the requirements of both national or regional healthcare service models, when the territorial organization is divided into self-regulatory administrative regions or municipalities. From a functional point of view, the X1.V1 platform is the enabling tool for implementing the EHR at regional and national level and the EPR (Electronic Patient Record), by supporting healthcare and clinical processes at enterprise level. Moreover X1.V1 could be used for supporting pathologys networks, second-opinion services and decision making processes with data collection and performance indicator elaborations. The X1.V1 platform is currently adopted by several Italian healthcare organizations, such as the Italian Regions Abruzzo, Umbria and Marche, for the EHR management. From a technological point of view, it implements a federative architecture, based on the standard IHE (Integrating Healthcare Enterprises) XDS (Cross Enterprises Document Sharing) integration profile. In particular, the X1.V1 platform follows a Service oriented approach by virtualizing the following components:

- Single-Sign-On (SSO) module, for managing authorization and authentication functions;
- Master Patient Index (MPI) module, for supporting the unique identification of patients among the different care environments;
- Master Code Index (MCI) module, for shared management of coding and terminologies;
- Document repositories, able to store all the managed clinical documents produced in the different care environments;
- Document Registry, able to store the relevant metadata of the documents and to provide the index of the documents stored in the repositories;
- Enterprise Service Bus (ESB) and a Workflow engine, for the orchestration of the clinical processes implemented through the platform.

The main information systems interoperating with the platform are the EMRs (Electronic Medical Records) of the General Practitioners, the Hospital Clinical Information Systems (HCIS), including EMRs, first aid and operating theatres applications, the Laboratory and Radiology Information systems, as well as non-clinical systems, such as booking systems, ADT. Moreover the platform is able to interoperate with the Pharmaceutical information systems and end-users/citizens viewing interfaces. The platform is able to support among the others the following activities:

- The eDocuments shared management, such as the publication and the accesses of the Patient Summary, the Patient Care Coordination Report, the Medical reports produced by the HCIS;
- the ePrescription processes;
- integrated care pathways related to predefined care models (such as Chronic Care model);
- the eBooking of health services;
- the events related to health episodes.

The use of the X1.V1 solution therefore needs the deployment and the management of a very complicated healthcare ecosystem, that requires an heterogeneous, time dependent and scalable allocation of resources.

3. Executing the X1.V1 Platform on the Cloud. The increasing popularity of Cloud systems [20] [30] is mainly due to the possibility to exploit (virtualized) resources when users actually need them for executing (parts of) their processes. Distinct Cloud service models have been defined [15], depending on the kind of resources that are provided. The framework proposed in this paper exploits the Infrastructure as a Service (IaaS) model, where the resources provided to users are computational infrastructures consisting of Virtual Machines (VMs) connected by virtual networks. When requesting VMs, users can choose the most proper network configuration, VM features (e.g., CPU count and memory space), VM images (i.e., the operating system they need for their application) and they can install and run on their VMs the applications they need. Once requested, the virtual computational infrastructure is available quickly and the number of machines and/or their features can be increased or decreased by the users on demand during the computation and according to their needs. Cloud users are charged by Cloud providers for resource usage, depending on the number of machines, on their features, on the computational time used, etc..

Public IaaS Cloud facilities are currently provided by many big companies such as Amazon[2] and Microsoft[3],

[2] http://aws.amazon.com/ec2/
[3] http://www.microsoft.com/windowsazure/features/compute/

and a number of frameworks, such as OpenNebula[4], Eucalyptus[5] or OpenStack[6], are available to deploy Cloud systems in users' data centres (Private Clouds) [31] [33].

The X1.V1 platform provides a set of services concerning several aspects of the ehealth scenario, such as Patient Identification, Clinical Document Repository, User Profiling, Prescription and others, and these services are already deployed on a set of VMs. Hence, executing the X1.V1 platform on the Cloud is straightforward. Allocating a static set of Cloud resources to each of these VMs could result in low performances as well as in a waste of resources, since the computational load of the services is variable over time. We report here two reference examples taken from the normal operation of the X1.V1 platform.

Example 1: The X1.V1 platform computes some statistics during the night, while the other services are likely to be unused. This is a very heavy task, and sometimes the results are not ready before the day after when daily services should start again. Hence, the load of the VMs involved in this task is very high during the night, while it is negligible during the day. Consequently, during the night hours the VMs that perform this task should have a large number of resources, while the VMs running the other services (such as the Booking or the Prescription one) should have a reduced number of resources.

Example 2: The Prescription service is used mainly by GPs, Pharmacists and Specialists to perform the ePrescription process. In fact, the GPs exploit the service to store the drugs and diagnostic exams prescriptions for their patients, and the Pharmacists and Specialists use the same service to retrieve such prescriptions and to update them according to drugs provided or exams executed. The Prescription service is likely to receive a large number of service requests during the working hours of GPs, Pharmacists and Specialists. In particular, there could be some computational peaks, i.e. a very large number of requests that are issued in a short period of time. Consequently, since each request is issued by a person that is waiting for the answer, in order to serve each request within a reasonable response time, the resources allocated to the VM(s) running the Prescription service should be increased during the working hours of GPs, Pharmacists and Specialists.

The previous examples point out that the usual behaviour of the X1.V1 platform from the point of view of resource requirements is known. In other words, during the daily operations there are predictable variations of the computational load of the X1.V1 VMs. Hence, the execution of the X1.V1 platform on a IaaS Cloud would greatly benefit from the Cloud elasticity in order to grant a given quality of service, i.e., to ensure an upper bound to response time of the provided services [4]. As a matter of fact, the framework could increase the resources assigned to a given VM, and hence to the services deployed on it, when a computational peak is expected for this VM. Symmetrically, the framework could reduce the resources assigned to other VMs when a low computational load is expected for these VMs. Performing the management and the update of the allocation of the Cloud resources to the X1.V1 VMs manually is not convenient, because it requires the knowledge of the allocation of the services to the VMs, it would be time consuming and error prone for the system administrator. For this reason, this paper proposes a framework for the management of the allocation of the Cloud resources to the VMs that compose the X1.V1 platform.

Besides the benefits previously described, the adoption of Cloud computing to perform critical tasks introduces also security issues. These security issues are described by the European Network and Information Security Agency (ENISA) in the report "Cloud Computing. Benefits, Risks and Recommendations for Information Security" [10], and by the Cloud Security Alliance (CSA) in the reports "The Notorius Nine. Cloud Computing Top Threats in 2013" [8] and "Security Guidance for Critical Areas of Focus in Cloud Computing" [7].

4. Related Work. A number of works in the literature, such as the ones in [2], [6], [16], and [25] describe some approaches for the design and implementation of several health related services (such as the EHR or health monitoring systems) on the Cloud. Instead, the focus of this paper is different from these works because we take for granted that the X1.V1 platform is already designed for being executed on the Cloud, and we focus on the design, implementation and validation of a tool which eases the exploitation of the Cloud elasticity, in order to allow the X1.V1 platform to take maximum benefit from the Cloud environment.

The authors of [3], instead, propose a tool, called Elastack, which is an automated monitoring and adaptive system, generic enough to be applied to existing IaaS frameworks, and intended to provide elasticity to these

[4]http://opennebula.org
[5]http://www.eucalyptus.com
[6]http://www.openstack.org

frameworks. As a matter of fact, the authors state that the elasticity support of the currently available Cloud frameworks is quite immature, and they define their own tool to support elasticity in IaaS Clouds. Their framework is based on the Serpentine middleware [19]. Elastack reacts to workload peaks by creating new instances of VMs, and these instances receive some tasks to be performed. These new instances are then destroyed when they are no longer necessary.

Roy et al. [26] describe a framework for efficient autoscaling in the Cloud, aiming at guarantee a given Quality of Service (QoS) level. Their framework exploits a look-ahead resource allocation algorithm based on model-predictive techniques for workload forecasting which allocates and deallocates VMs to the application trying to optimize the amount of used resources while respecting the promised QoS and minimizing the operational cost.

The approach proposed in [1] defines two adaptive controllers which use both reactive and proactive controls to dynamically change the number of VMs allocated to a service based on the current and on the predicted future load of that service.

The previous approaches differ from the one proposed in this paper because they resolve performance issues simply by allocating new VMs or deallocating existing ones, and this requires that multiple instances of the applications deployed on the VMs can be run in parallel. The X1.V1 Cloud Manager, instead, does not simply increase the number of the VMs of the system to react to a computation peak. In fact, the X1.V1 Cloud Manager defines a set of configurations exploiting at best the Cloud resources assigned to the X1.V1 VMs in some specific scenarios (such as the ones described in the two reference examples in Section 3). Hence, the X1.V1 Cloud Manager could resolve a temporary performance issue due to the overloading of some services even without increasing the total amount of resources assigned to the X1.V1 platform, but simply choosing a different allocation of the same resources to the VMs of the X1.V1 framework.

CloudScale [29] is a prediction driven system that automates fine-grained elastic resource scaling for Cloud infrastructures. CloudScale does not assume any prior knowledge about the applications that are running on the Cloud, and it exploits an online resource demand predictor, described in [13], that is application agnostic and that achieves a good prediction accuracy for a range of real world applications. CloudScale also defines two complementary handling schemes to be executed when the resource demand predictor underestimates the resource requirement, to update the resource allocation thus avoiding the violation of the previously defined service level objectives.

The main difference between the X1.V1 Cloud Manager and the approaches proposed in [26], [1], and [29] is that they address two different problems: the X1.V1 Cloud Manager assumes a deep knowledge of the application that is running on the Cloud, the X1.V1 platform (because the configurations that describe the resource allocation are defined according to the usual application behaviour), while the other approaches do not assume any knowledge of the applications that are running on the Cloud, and for this reason they are focused on predicting the future resources requirement of these applications in order to promptly update the resources allocation.

CloudScale[7] is also a research project funded under the European FP7 programme. The main aim of the project, introduced in [5], is to provide an engineering approach for building scalable cloud applications and services based on the state of the art cloud technologies and software. The CloudScale approach will make cloud systems scalable by design, i.e. they will fully exploit the cloud elasticity during their operations to provide scalability, while minimizing the amount of computational resources used. One of the reference scenario addressed by CloudScale will be the EHR one. Since [5] is a work-in-progress paper, it simply provides an overall description of the Cloudscale approach, and it does not allow us to make a comparison with our approach.

A set of further works concerning other approaches to exploit Cloud elasticity are described in [12].

Finally, the difference between the X1.V1 Cloud Manager and the graphical tools provided by the main Cloud frameworks (e.g., Sunstone and Horizon) is that the X1.V1 Cloud Manager allows to transparently manage a number of distinct Cloud providers, which hosts the X1.V1 VMs, and the configurations are customized for the computational requirements of the X1.V1 framework in several scenarios.

5. X1.V1 Cloud Manager. The X1.V1 Cloud manager is a framework meant to enable an easy, quick, and secure management of the resources allocated to the X1.V1 platform in the Cloud environment, in order to

[7] www.cloudscale-project.eu

enhance the platform performances, optimize resource utilization and, consequently, to reduce the whole cost of the computation. In particular, the main aim of the X1.V1 Cloud Manager is to enable the system administrator to easily deploy and manage the VMs that build up the X1.V1 platform in the Cloud environment, in order to quickly react to computational peaks by properly updating the resources allocation to the X1.V1 VMs. In fact, the X1.V1 Cloud Manager allows the system administrator to define a set of configurations, where each configuration defines in details the resources to be allocated to each VM of the system (e.g., the number of virtual CPUs or the size of the RAM memory) and other configuration details (e.g., the Cloud Provider on which each VM will be deployed). In general, each configuration defines the best resource allocation to obtain a good performance in a specific scenario, such as the GPs' labs opening hours one. With reference to the previous example, the system administrator could define the following configurations:

Normal: allocates the same amount of resources to all the VMs;

Interactive: allocates more resources to the VMs hosting the interactive services, i.e., the services which require interactions with people (e.g., doctors, patients), such as the Prescription service, reducing the resources allocated to the statistic service and or to other services;

Night: allocates a small amount of resources to the VMs that implement the interactive services, and reserves high computational power and large memory space for the VMs responsible to perform statistical analysis or data warehousing on the available data;

Power: allocates high computational power and a large memory space to all the VMs of the X1.V1 platform.

Many other configurations could be defined by the system administrator, to handle other possible situations where a different allocation of the resources to the X1.V1 VMs could lead to better performances.

The X1.V1 Cloud Manager allows the management of distinct Cloud providers at the same time, even running distinct Cloud softwares, such as OpenStack or OpenNebula. Hence, a configuration file could state that some VMs of the X1.V1 platform can be deployed on a provider running OpenNebula and other VMs on another one running OpenStack. Switching from a configuration to another, besides changing the allocation of the resources to the VMs, could even migrate some of them from a Cloud provider to another.

Depending on the Cloud provider and on the related agreement, some configurations could result in additional costs. As an example, we could define the *Normal*, the *Interactive*, and the *Night* configurations in a way such that they require the same amount of resources, i.e., they define three different ways of partitioning these resources, while the *Power* configuration requires more resources than the others. We also suppose that the *Power* configuration uses more resources than the ones agreed with the Cloud provider (or available in the user data center in case of Private Cloud); hence the *Power* configuration could lead to additional cost for the Cloud user. Hence, in order to reduce the cost of the computation, during the normal operation of the X1.V1 platform the first three configurations should be used, while the *Power* configuration should be exploited only when it is really necessary to guarantee a good service level. The X1.V1 Cloud manager represents the configurations previously described through a OVF [9] file (with extensions).

When the system administrator selects a configuration, the X1.V1 Cloud Manager performs all the operations required to update the VMs according to the new configuration, such as changing the amount of memory or the number of virtual CPUs allocated to the VMs or migrating the VMs from one Cloud Provider to another.

The X1.V1 Cloud Manager includes a monitor component that collects information from the running VMs, such as the CPU workload, the amount of memory used and the network traffic. This information is elaborated in order to provide aggregated data that could help the system administrator to decide whether the current configuration is resulting in good performance or a more suitable configuration can be chosen.

We plan, as a future work, to provide the X1.V1 Cloud Manager of a further component which automatically decides whether, in a given scenario, a new configuration should be adopted, i.e., whether another resource allocation would result in better performance of the X1.V1 platform. This component will include a resource demand predictor that will exploit the information collected by the monitor system in order to perform the configuration switch before the actual change of the load of the X1.V1 VMs and only if the time spent to perform the configuration switch is paid by better performance.

The X1.V1 Cloud Manager has been designed to provide an adequate security level for the processes and data involved in the X1.V1 platform. In particular, this paper is focused on the security of the X1.V1 Cloud Manager, in order to prevent unauthorized users to change the current configuration. As a matter of fact,

adopting the wrong configuration could lead to a significant degradation of the performance of some X1.V1 services, thus causing a denial of service. Instead, this paper does not cover the security issues specific of the X1.V1 platform because it assumes that the X1.V1 platform already provides an adequate level of security.

The security support of the X1.V1 Cloud Manager includes the authentication and the authorization phase. The authentication phase exploits the LDAP certificate-based authentication method [28], [32], which provides an adequate security level for the critical applications such as the X1.V1 platform. Each entry of the LDAP directory represents a X1.V1 administrator, i.e., a subject who has some rights concerning the management of the X1.V1 platform on the Cloud, and include a set of attributes that are paired with the corresponding X1.V1 administrator, that will be used in the authorization phase for evaluating the access control policy. The *role* is an example such attributes. Other attributes (static or even dynamic, i.e., that change over time) could be introduced in order to define complex access control policies. The authorization component is aimed at checking that the user that authenticated herself/himself on the Dashboard really holds the right to perform the operation she/he requests, because distinct sets of rights are assigned to distinct (classes of) users. To design both the architecture of the authorization system and the security policy language, we rely on the XACML standard [23] developed by the OASIS consortium, because it is well known, widespread, and supported both by academic (and, consequently, free and open source) such as WSO2 Balana, and by commercial tools, such as the Axiomatics authorization framework [11]. The XACML standard naturally allows to implement the Role Based Access Control (RBAC) model [27], [22], i.e., to define a set of roles and to pair a set of rights to each of the roles. In our scenario we could have (at least) two roles, ADMIN and SUPERADMIN. The role ADMIN will have the right to select a subset of the existing configurations only, i.e., the ones that do not require to pay an additional fee to the Cloud Providers, while the role SUPERADMIN will have the right to select any of the existing configurations and even to load a new and customized configuration written for the specific situation she/he is coping with. For what concerns the interactions with the Cloud Providers, instead, we exploit at best the security mechanisms they provide for the authentication on the Cloud platform and for securing the communications between the X1.V1 Cloud Manager and the Cloud platform.

5.1. Architecture. The architecture of X1.V1 Cloud Manager is shown in Figure 5.1, and consists of the following components.

Web Dashboard. It is a web application providing a graphical interface which allows to easily manage the configuration of the cloud services that build up the X1.V1 platform. All the configurations stored in the repository are listed in the dashboard, and the administrators can switch from the current configuration to another one in just one click, i.e., simply pushing the related button, provided that they hold the proper right. Authorized administrators could also upload customized configurations to deal with special or emergency situations. The Switch Configuration Command (SCC) is sent to the Configuration Engine. This interface also allows to monitor the resource usage of the VMs of the X1.V1 cloud system, properly represented through a set of graphs. Figure 5.2 shows the dashboard interface that allows the configuration switch, while Figure 5.3 shows the dashboard page that represents the resource usage of each VM.

Configuration Engine. This component is the core of X1.V1 Cloud Manager. This module receives the Switch Configuration Commands (SCC) from the dashboard, processes and executes them on the Cloud Providers that host the final services. Each SCC requires to change the Cloud resource allocation according to the related configuration file. The directives listed in the configuration file need to be mapped into a set of specific commands to be sent to the Cloud Providers to perform the required resource reallocation. For this reason, a specific driver for each Cloud Provider has been included in the architecture, to translate the SCC into a set of commands that can be executed by the Cloud Provider. Finally, the Configuration Engine embeds the Policy Enforcement Point (PEP) that triggers the authorization process.

Security Manager. It provides a cross support for the authentication and authorization of users accessing the X1.V1 Cloud Manager. The authentication system is based on a LDAP server that is invoked by the web dashboard. After a successful login, the authentication system also returns a set of user attributes, such as the user ID and Role, that are necessary to perform the authorization process. The authorization process is triggered by the Policy Enforcement Point embedded in the Configuration Engine, which invokes the authorization system to check the authorization policies every time it receives a SCC. The access control policies are written in XACML language, a widely used standard developed by OASIS consortium. In the reference scenario, the policy exploits

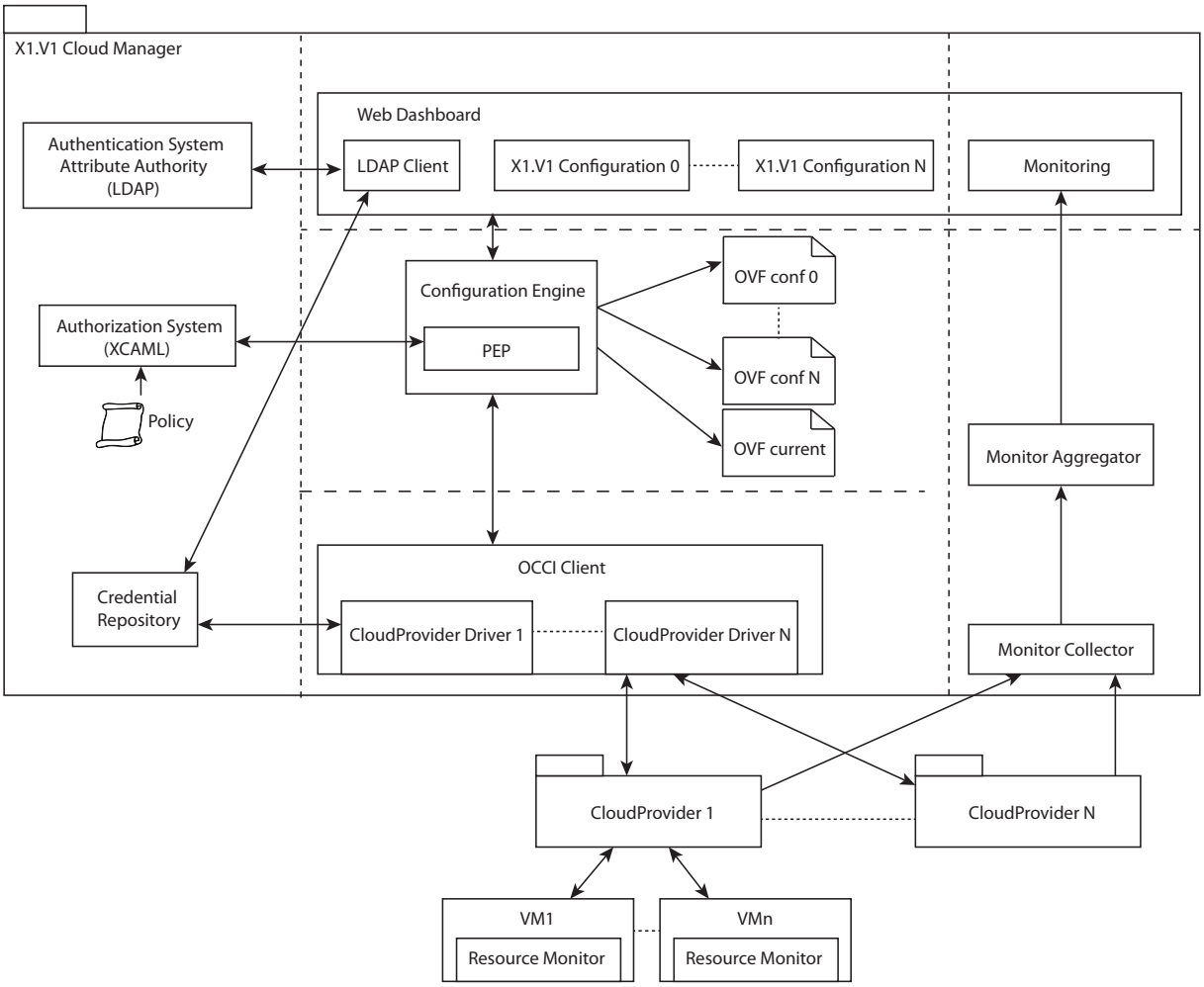


FIG. 5.1. *Framework Architecture*

the Role Based Access Control (RBAC) model, where a set of roles are defined and a set of rights is paired to each of these roles. One (or more) role is assigned to each user, thus determining the users rights. However, the XACML language is very expressive, and allows to write more complex security policies than the pure RBAC one. As a matter of fact, besides the role, other attributes can be stored in the LDAP server and exploited in the access control policy for the authorization decision process.

Driver for Cloud providers. The drivers are in charge of the communications with the Cloud Providers, i.e., of customizing the commands defined by the Configuration Engine for the specific Cloud Provider these commands are to be sent. Hence, a specific driver must be instantiated for each distinct Cloud provider exploited for the execution of the X1.V1 VMs. In general, a driver could also be instantiated to exploit a Public Cloud, such as Amazon EC2, for the execution of some of the VMs. The drivers also manage the channels used for the request/response to the Cloud Providers and the channels used for the monitoring communications (data and alert). The channel for the request/response is a simple OCCI interface common to all the Cloud drivers present in the system. The drivers also manage the security of the interactions with the Cloud Providers. In particular, a set of local users is statically defined on each Cloud Provider, and each driver exploits the right user to interact with each Cloud Provider. Secure communications must be exploited with the Cloud Providers that support them.

Monitoring System. For each of the VMs running on the Cloud Providers, the monitoring system collects a set of usage parameters, such as the CPU and RAM utilization and network statistics. The monitoring system also allows to set some alarms when a metric exceeds a threshold. The alarms have two level of emergency: WARNING and FAILURE. When a VM notifies the violation of a threshold the system forwards the notification to the dashboard, in a way such that the administrator can change the current configuration. The monitoring information is sent from the Cloud Provider to a collector module present in the Cloud Manager. This module collects both the statistics and the alarms coming from the Cloud Provider, and it passes these data to a Monitoring aggregator that splits the hardware statistics and the alarms. Finally, the monitoring information are displayed in the monitoring section of the Web dashboard in a proper graphical format, that also provides a clear visualization of the alarms (Figure 5.3).

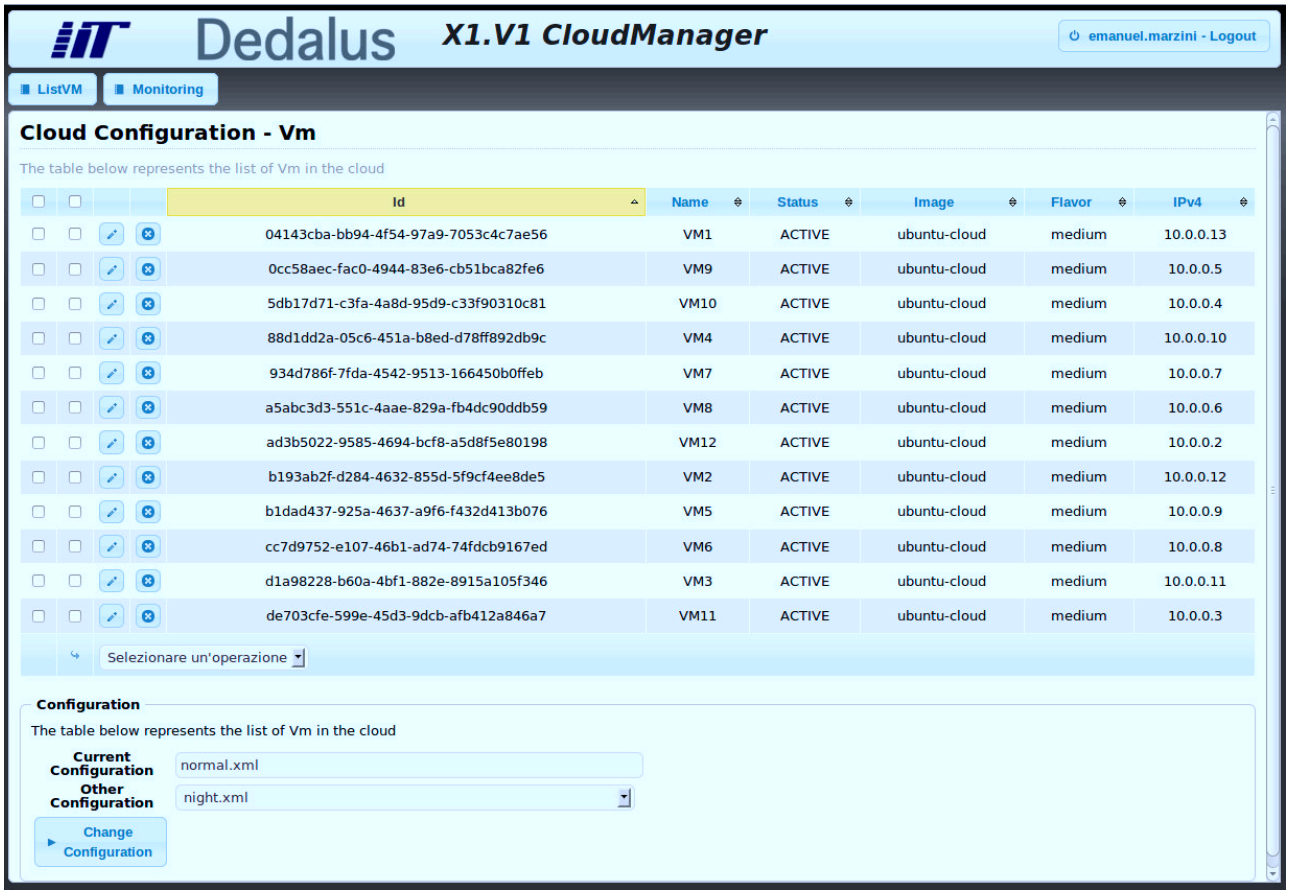


FIG. 5.2. *Dashboard Configuration Switch page*

6. Prototype. This section describes the main features of the prototype we developed, that implements most of the functionalities of the architecture described above, and presents a set of experimental results.

6.1. Implementation Details. The X1.V1 Cloud Manager core is mainly developed in Java EE under Apache Tomcat web server. The Web dashboard uses the typical Java EE tools like a JSP and Servlet. Ajax and the Highcharts library have been used to display graphs.

The authentication system exploits the certificate-based authentication method provided by the LDAP protocol. In particular, we exploited LDAP over SSL (LDAPS), which allows to use a secure channel to communicate the credential in the authentication phase instead of a plain one. We added custom attributes in posixAccount to retrieve credentials to communicate with OpenStack. Once retrieved, these values are used

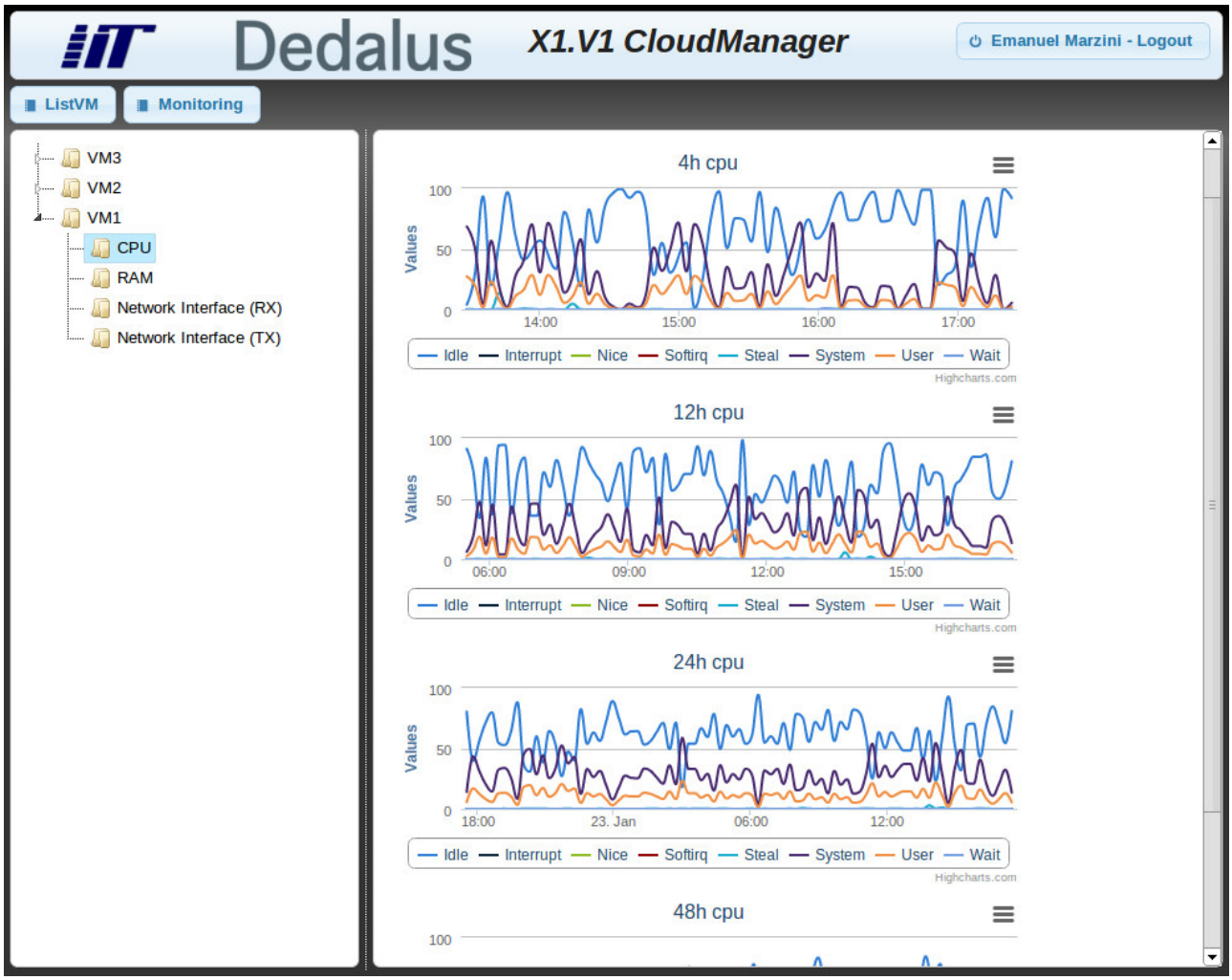


FIG. 5.3. *Dashboard Monitoring page*

once for the initial authentication on OpenStack, and they are not saved anywhere. Our implementation expoits the OpenLDAP[8] software to set up both the LDAP server and the client within the dashboard.

The authorization system is based on the WSO2 Balana[9] XACML engine, which is an open source software for evaluating access control policies written in XACML v3.0. An alternative XACML engine that could be adopted is the one released by SUN[10]. However, SUN's engine supports XACML v2.0 only, it is no longer maintained and it is considerably slower than the WSO2 Balana engine. Another XACML engine is the one developed by Axiomatics, which is a commercial product[11]. In particular, a significant advantage is that Axiomatics provides an authorization framework covering all the phases of the authorization process, such as policy writing, enforcing and tuning.

The communications among the Configuration Engine and the XACML authorization system follows the SAML protocol [21] and the opensaml library is exploited.

The current version of our prototype implements one Cloud Provider driver only, i.e., the OpenStack one,

[8]http://www.openldap.org/
[9]http://xacmlinfo.org/category/balana/
[10]http://sunxacml.sourceforge.net
[11]http://www.axiomatics.com/solutions/products.html

because OpenStack is a widely used open source software for setting up private Clouds, and it is the one we used for our experiments. OpenStack does not require to shutdown the VMs to change their resource allocation. Hence, the X1.V1 services are simply suspended when a configuration switch is performed, and they are available again right after the resource reallocation. For the Cloud Provider installation we chose the OpenStack Havana release with the fundamental modules installed on two distinct nodes: the Controller Node and the Compute Nodes. The Controller Node hosts Keystone (Security manager), Glance (Image manager) and Nova (VM manager), and it manages all the request from the X1.V1 Cloud Manager. The Compute Nodes host only the compute agent for Nova and the network module. These nodes receive the requests from the Controller Node and manage the execution of VMs. In our prototype the network communications are managed through nova-network without the specific module Neutron.

As far as concerns the authentication on the Cloud providers, OpenStack provides a token system to allow interactions through its API calls. When the user successfully performs the initial authentication phase, Keystone assigns a temporary token, and this token can be used by the OpenStack driver to send commands to the Cloud provider. The token expiration is setted to 24h, but this parameter is configurable by OpenStack. The initial authentication on each Cloud provider is executed when the user performs the authentication on the X1.V1 Cloud Manager. In particular, a PKI infrastructure is used to authenticate on OpenStack, and the required key is stored in a password protected Keystore, that is opened by the user at authentication time. This credential allows the OpenStack driver to perform the initial authentication phase on the Cloud provider and to obtain the authentication token required to invoke the subsequent commands to manage the Cloud resource allocation. For this reason we have a secure channel to communicate among X1.V1 Cloud Manager and OpenStack Cloud provider.

The monitoring system exploits Collectd[12] to retrieve the usage information from the VMs. There is a client on each VM in the cloud that retrieve the statistics of CPU usage and RAM utilization. These data are sent to a collector daemon in each cluster that redirect the statistics to X1.V1 Cloud Manager. The data collected are analyzed and visualized on the web dashboard.

6.2. Experimental Results. In order to validate the proposed framework, we performed a set of experiments to evaluate the time required to switch from one configuration to another varying the number of VMs in the system. In particular, we measure the time from the moment when we press the switch configuration button on the X1.V1 Cloud Manager interface to the moment when all the VMs of the system are again in active state. The aim of these experiments is to evaluate whether the proposed framework could be successfully adopted to increase the performances of the X1.V1 platform on the Cloud, e.g., by addressing the issues described by the two reference examples presented in Section 3.

In general, we can say that the porting of the X1.V1 platform on the Cloud through the proposed framework is beneficial when the time required to perform the switch between two configurations is low enough to be compensated by the better system performance due to the new configuration. In particular, to address the issue described in *Example 1*, the time required to switch to a configuration which allocates more resources to the VM(s) that performs the statistics should be compensated by a faster execution of the statistics procedure. In other words, the time T_N required to carry out the statistic task exploiting the new configuration should be lower than the sum of time T_I required to carry out the same task exploiting the initial configuration and twice the time T_S required to perform the configuration switch ($T_N + 2T_S < T_I$). We consider twice the time required to perform the configuration switch ($2T_S$) because we take into account also the time to restore the initial configuration, since the new configuration could be inadequate for the daily operation of the system, supposing that the two configuration switches take the same time. To address the issue described in *Example 2*, instead, the time required to switch to a configuration which allocates more resource to the VM(s) running the Prescription service should be reasonably low in order to give a prompt response to users (GPs, Pharmacists, Specialists and, overall, the patients) that are waiting for service completion.

The physical testbed where we performed the first set of experiments consists of one machine equipped with a Intel i7 2600 3.4 Ghz CPU with 8 cores and 8GB of RAM memory which hosts the Controller Node, and one machine equipped with an AMD Opteron 6380 2.5 Ghz CPU with 16 cores and 32 GB of RAM memory, which

[12] http://collectd.org/

hosts the compute node. The operating system installed on these machines is Linux Ubuntu 12.04 LTS (Precise Pangolin), with 3.11.0-26-generic 64bit kernel. The OpenStack version used is Havana with the components Keystone (Security manager), Glance (Image manager) and Nova (VM manager). The Controller node hosts the following modules: keystone, glance-registry, glance-api, nova-api, nova-cert, nova-consoleauth, nova-scheduler, nova-conductor, nova-novncproxy. The Compute nodes host nova-compute and nova-network.

The allocation of the resources to the VMs is defined by the following three VM flavors:

- *Small* (1 VCPU, 1GB RAM)
- *Medium* (1 VCPU, 1,5GB RAM)
- *Large* (1 VCPU, 2GB RAM)

Although our architecture states that the configuration files follow the OVF standard, we found out that the current release of OpenStack does not provide a stable support for OVF yet. Hence, the current implementation adopts a custom format to represent the configuration files, which will be replaced by a customization of the OVF format as soon as possible. The custom format used for the experiments has a XML structure with the essential information for each VM.

To perform our tests we defined four distinct configurations: *Normal, Interactive, Night*, and *Power* which differ one from another for the flavors choosen for each VM of the system. In fact, in the *Normal* configuration the flavor of all the VMs is *Medium*, while in the *Power* configuration, all the VMs are *Large*. Moreover, supposing to have *N* VMs, in the *Interactive* (respectively, *Night*) configuration, the flavor of the first *N/2* VMs is *Large* (respectively, *Small*) and the flavor of the remaining VMs is *Small* (respectively, *Large*). Table 6.1 reports these configurations in case of 4 VMs.

Table 6.1
Configurations in case of 4 VMs

	Normal	Interactive	Night	Power
VM1	Medium	Large	Small	Large
VM2	Medium	Large	Small	Large
VM3	Medium	Small	Large	Large
VM4	Medium	Small	Large	Large

The full amount of resources required by the *Normal, Interactive*, and *Night* configurations is the same (4 VCPUs and 6GB of RAM memory in the example of Table 6.1), while the *Power* configuration requires more resources (4 VCPUs and 8GB of RAM memory in the same example). Figures 6.1 and 6.2 show the time required to switch from one of the four configurations to another for each couple of them, varying the total number of VMs of the system. The VM image we used for this set of experiments is the Ubuntu Cloud image, which is "a pre-installed disk image that has been customized by Ubuntu engineering to run on cloud-platforms such as OpenStack" equipped with an Ubuntu Server 12.04 LTS (Precise Pangolin)[13]. The Ubuntu Cloud image size is 248MB.

The results in Figures 6.1 and 6.2 clearly show that the time required to switch from one configuration to another depends on the number of VMs whose flavor is changed. In particular, Figure 6.1 shows the time required to switch from a configuration to another in those cases where this switch requires to change the flavor of half of the VMs of the system. As an example, in the case of 8 VMs the switch from the *Interactive* to the *Power* configuration requires to change the flavor of 4 VMs from *Small* to *Large*. Figure 6.2, instead, shows the time required to switch from a configuration to another in those cases where this switch requires to change the flavor of all the VMs of the system. As an example, in the case of 8 VMs the switch from the *Normal* to the *Power* configuration requires to change the flavor of 8 VMs from *Medium* to *Large*.

For any number of VMs, we notice that the time to perform the configuration switchs requiring to change the flavor of half of the VMs (Figure 6.1) is considerably less than the time to perform the configuration switches that require to change the flavor of all the VMs (Figure 6.2). For example, in the case of 8 VMs, the time to switch from the *Interactive* configuration to the *Power* one is about 20 seconds, while the time to switch

[13] https://cloud-images.ubuntu.com/precise

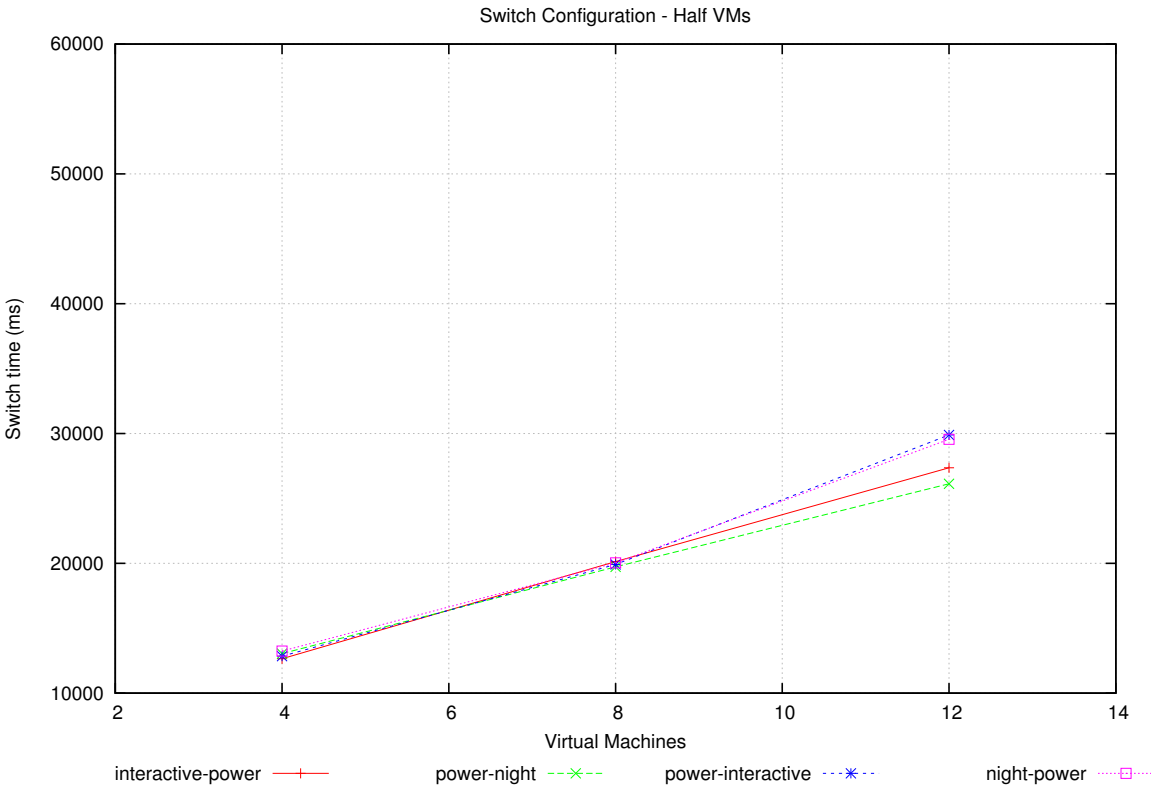


FIG. 6.1. *Time to perform a switch involving half of the VMs (Ubuntu Cloud image)*

from the *Normal* to *Power* one is about 31 seconds. Moreover, we also notice that increasing the number of VMs, the time required to perform any kind of switch increases. In particular, we notice that, doubling the number of VMs whose flavor is changed (either doubling the number of the VMs of the system or performing a configuration switch that require to change the flavor of all the VMs instead of half of them), the time required to perform the whole switch procedure is considerably higher, although the switch procedure does not take twice the time. For example, the time to perform the switch from the *Normal* configuration to the *Power* one in case of 4 VMs is about 19 seconds, and in case of 8 VMs is about 31 seconds. We estimated that, on average, the time required to change the flavor of N VMs is 156% the time required to change the flavor of $N/2$ VMs.

Furthermore, we notice that the maximum switch time in case of 4 VMs is about 20 seconds, while in case 12 VMs is about 47 seconds. This means that switching from a configuration that is not suitable to perform a given task to one that is more suitable for this task is convenient when the second configuration allows to save at least 20 seconds in case of 4 VMs and 47 seconds in case of 12 VMs. However, if the second configuration is not suitable for the normal operation of the system, we should consider twice the switch time i.e., respectively, 40 and 94 seconds, because we need to restore the initial configuration.

The results in Figure 6.3 show the time to perform a configuration switch exploiting another VM image, i.e., Ubuntu 12.04 Desktop image, whose size is 1.4GB. The figure shows the time required to perform both the configuration switches that involve half the VMs of the system and all the VMs of the system. We notice that the results in Figure 6.3 confirm the comments we made for Figures 6.1 and 6.2, and that, in the worst case, the time required for a configuration switch is about 20 seconds on 4 VMs and less than 50 seconds on 12 VMs. Since the previous set of experiments has been executed exploiting one compute node only, no VM migration occurred.

Hence, we performed another set of experiments on a different testbed, which consists of four machines. This testbed includes one machine equipped with a Intel i7 2600 3.4 Ghz CPU with 8 cores and 8GB of RAM

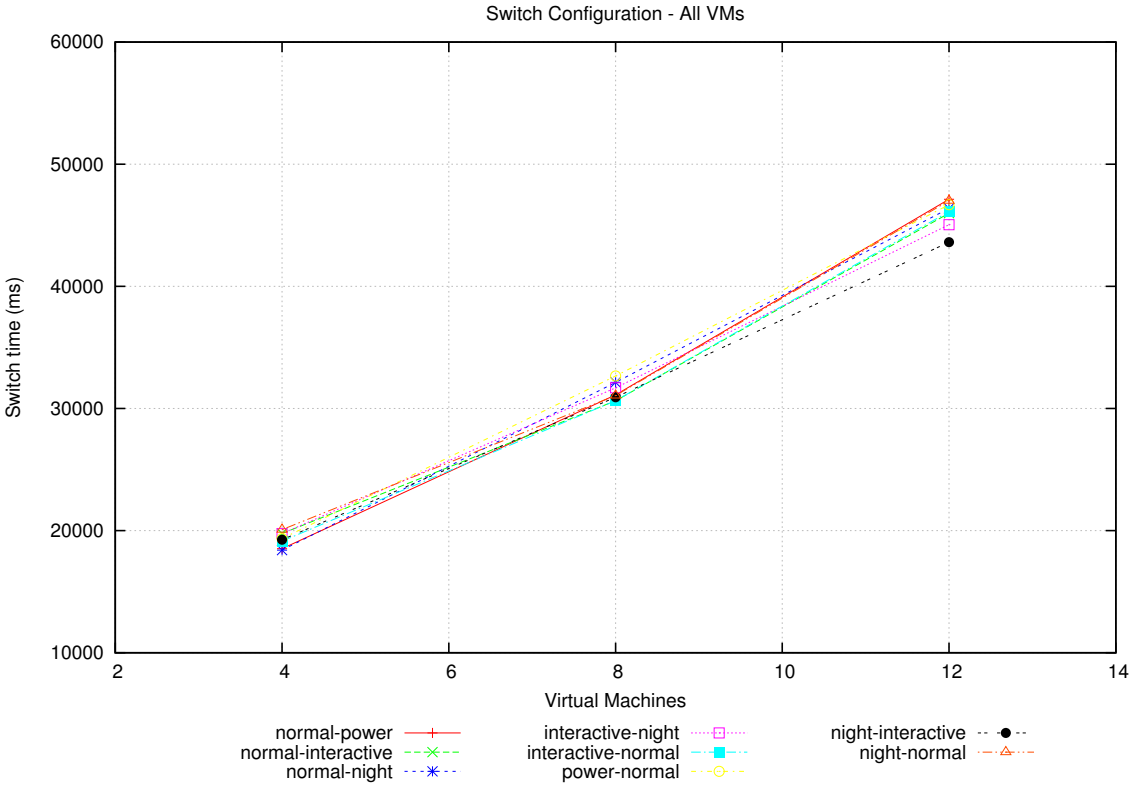


FIG. 6.2. *Time to perform a switch involving all of the VMs (Ubuntu Cloud image)*

memory which hosts the Controller Node, and three machines which are exploited as compute nodes: the first machine (A) is equipped with an AMD Opteron 6380 2.5 Ghz CPU with 16 cores and 32 GB of RAM memory, the second machine (B) is equipped with an Intel i5 750 2,66 Ghz CPU with 4 core and 8 GB of RAM memory, and the third machine (C) is equipped with Intel i5 760 2,8 Ghz CPU with 4 core and 8 GB of RAM memory. The operating system installed on all these machines is Linux Ubuntu 12.04 LTS (Precise Pangolin), with 3.11.0-26-generic 64bit kernel. These experiments also consider a different allocation of the resources for the three VM flavors:

- *Small* (1 VCPU, 1GB RAM)
- *Medium* (2 VCPU, 1,5GB RAM)
- *Large* (4 VCPU, 3GB RAM)

Moreover, the VM image we exploited for all the VMs is Ubuntu Server 12.04 LTS (Precise Pangolin), and the image size is 3GB.

Figure 6.4 shows the time required to perform the configuration switch from the *Interactive* to the *Power* configuration, which changes the flavor of half the VMs of the system, and the time required to switch from the *Normal* configuration to the *Power* one, which involves all the VMs of the system. When the system consists of 4 VMs, the time to switch from the *Interactive* to the *Power* configuration is about 17 seconds, while the time to switch from the *Normal* to the *Power* configuration is about 28 seconds and the VMs, that before the switch were running on machine A, did not migrate on other machines. The results for 4 VMS are slightly greater with respect to the ones obtained from the previous set of experiments. We think that this could be due to the fact that the flavour used in the second set of experiments exploits a larger set of resources (VCPUs and RAM memory).

Instead, when the system consists of 8 VMs, the switch from the *Interactive* to the *Power* configuration, on average, takes about 156 seconds. This time is considerably larger with respect to the one in Figure 6.3,

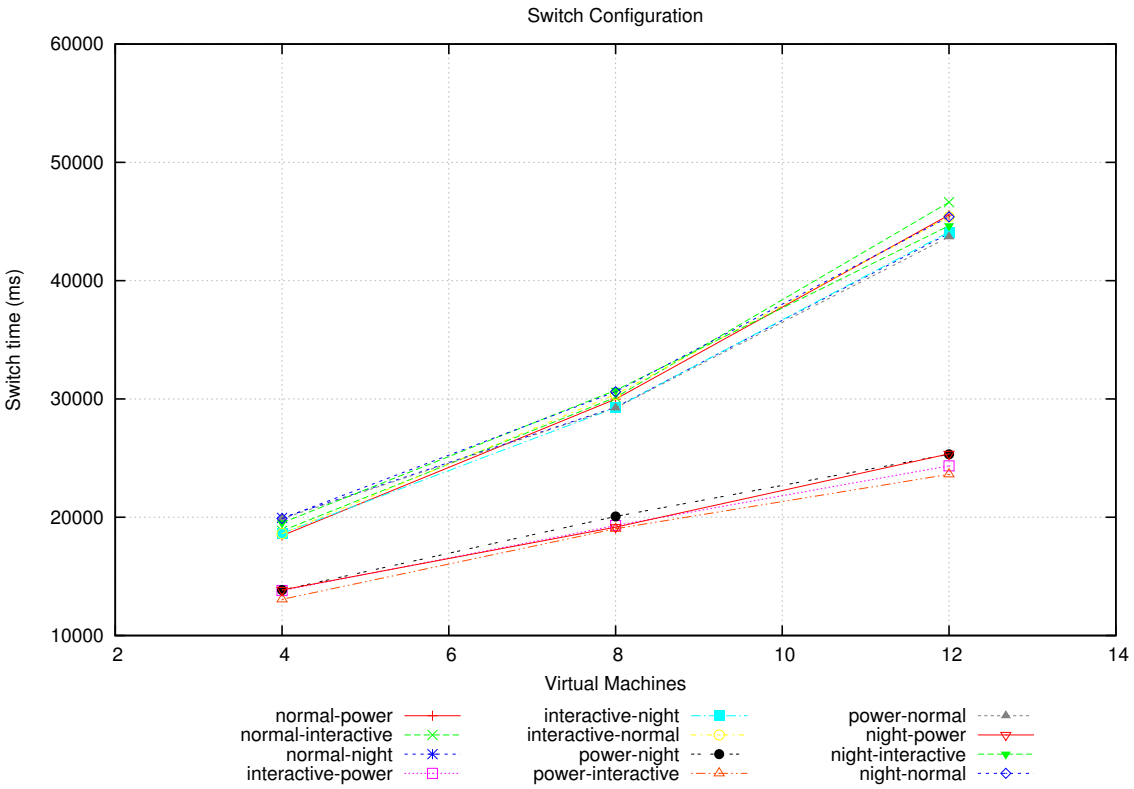


FIG. 6.3. *Time to perform a switch (Ubuntu Desktop image)*

and the main reason is that, on average, about 2 VMs (2,15) migrated from one compute node to another. In particular, taking into account one specific experiment, before the switch 5 VMs were running on machine A, 2 VMs were running on machine B, and 1 VM was running on machine C. The switch caused the migration of 3 VMs, and after the switch all the VMs were running on machine A. The time required for this specific experiment was 222 seconds. In another experiment, 6 VMs were allocated on machine A and 2 VMs were allocated on machine B before the switch. The switch involved the migration of 2 VMs, and took about 137 seconds. After the switch all the VMs were allocated on machine A. In a further experiment, before the switch all the VMs were running on machine A, and after the switch all the VMs were still running on machine A. In this case, the time required to perform the switch was about 32 seconds. As far as concerns the switch from the *Normal* to the *Power* configuration, the average execution time is almost the same, i.e., about 158 seconds, and the average number of migrated VMs is about 2 (2,21). Finally, the results of the experiments performed with 12 VMs show that the *Interactive* to *Power* configuration switch requires, on average, 358 seconds and involves the migration of 5 VMs, while changing the configuration from *Normal* to *Power* requires 378 seconds and involves the migration of 6 VMs.

These results show that, in case of migration, the time required to perform the configuration switch mainly depends on the number of VMs that are migrated, and the number of VMs whose flavor is changed is not important in this case. In turn, the number of VMs that are migrated from one physical machine to another depends on the original allocation of the VMs to the physical machines, and on the allocation policy of OpenStack. This explains, for instance, why the time required for the configuration switch in case of 8 VMs is almost the same in the case where the flavor of 4 VMS is changed and in the case where the flavor of 8 VMs is changed, i.e., because the average number of migrated VMs is almost the same. Moreover, we performed a further set of experiments on 4 VMs exploiting the same VM image described previously and the following flavors:

- *Small* (1 VCPU, 1GB RAM)

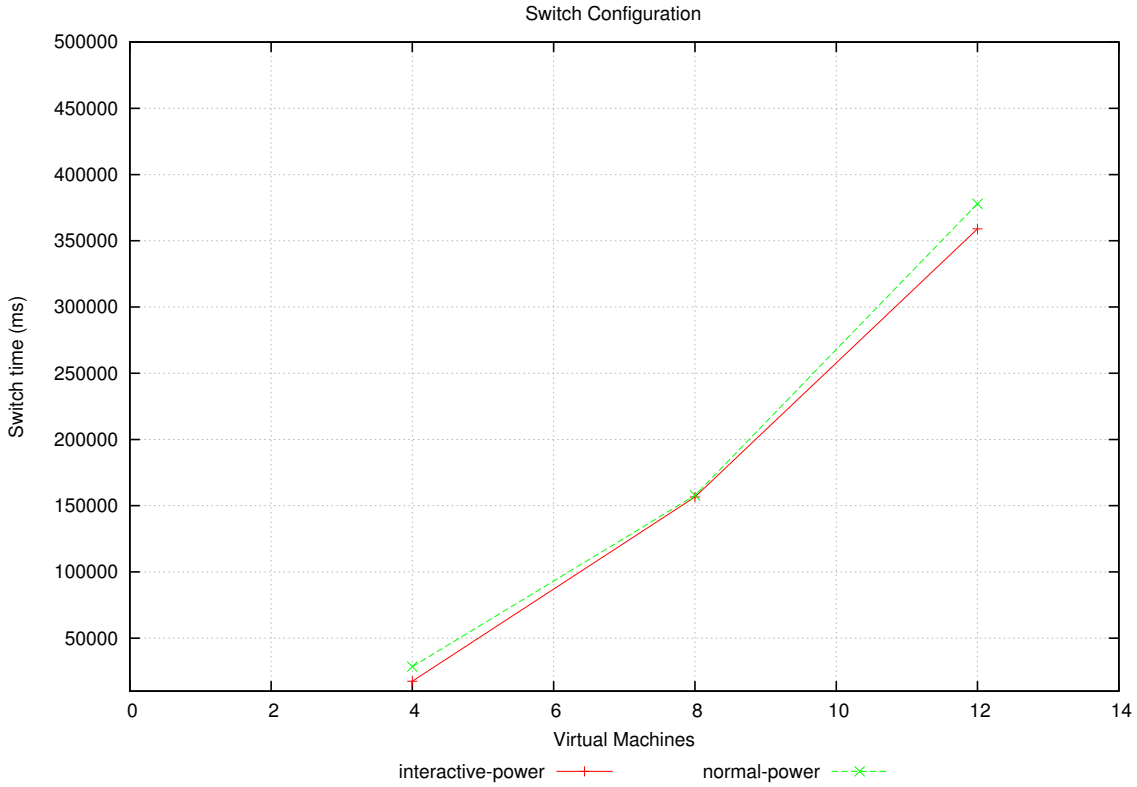


FIG. 6.4. *Time to perform a switch (Ubuntu Server image)*

- *Medium* (4 VCPU, 4GB RAM)
- *Large* (6 VCPU, 6GB RAM)

The average switching time was 148 seconds and, on average, the number of migrated VMs was 2 (1,9). This result confirms that the switching time mainly depends on the number of VMs that are migrated. As a matter of fact, the switching time of this experiment is almost the same as the one obtained in the previous set of experiments with 8 VMs (156 seconds) because the number of VMs that migrated is almost the same (about 2) in the two experiments.

7. Conclusion. This paper proposed a tool, the X1.V1 Cloud Manager, which optimizes the resource utilization for the execution of the X1.V1 platform on the Cloud by easing the exploitation of the Cloud elasticity. In fact, the X1.V1 Cloud Manager allows to change the allocation of the Cloud resources to the VMs according to some predefined configurations, in order to reduce the response time of the X1.V1 services in case of computational peaks, to reduce the completion time of some heavy computational tasks, and to save resources when the overall load of the platform is low. Since the resource requirements of the X1.V1 platform is known, the configuration switch to be performed during the daily operations can be planned in advance. However, some configuration switches can be manually initiated by the administrators to resolve unexpected computational peaks. Moreover, as a future work, we plan to integrate a further component in our framework in order to deal with unpredictable variation of the computational load of the VMs, i.e., to automatically detect, or even predict, when a new configuration should be adopted to achieve better performance.

The X1.V1 Cloud Manager security support includes proper authentication and authorization mechanisms, in order to avoid that unauthorized subjects change the Cloud resource allocation, since this could reduce the performance of the X1.V1 platform, increase the cost of the computation, and could even result in a Denial of Service. The paper presented two sets of experimental results which prove that, when no VM is migrated

from one compute node to another, the time required to perform a configuration switch is quite low, even in case of a large number of VMs. When some VMs are migrated from one compute node to another, instead, the configuration switch time increases considerably. Since the VMs migration affects the configuration switch time, which depends on the size of the VMs that are migrated, the X1.V1 Cloud manager will be configured in a way such that only the VMs with small images are migrated. Hence, the VMs storing the EHR data are not be migrated, while the VMs implementing some of the services provided by the X1.V1 platform could be migrated.

As ongoing work, we are deploying our framework in the main data center of the Dedalus SpA company (which is located in Avellino, Italy) in order to perform some simulations in a real scenario, trying also to evaluate the benefits resulting from changing configuration in some specific load distribution.

As future work, we plan to study the adoption of an advanced authorization model, the Usage Control Model [24] to regulate the usage of the X1.V1 VMs, as described in [34], [17]. As an example, the usage control authorization system could restore the previous (or a standard) configuration when the administrator that updated the resource allocation with a new configuration loses the right required to perform this switch. This policy could be adopted when the new configuration involves additional costs for the user.

Acknowledgment. This work was partially supported by the Italian project: "Piattaforme Cloud Sicure per applicazioni critiche: il caso del Fascicolo Sanitario Elettronico (PICs): progetto POR CRO FSE 2007-2013 Asse IV Capitale Umano" funded by Regione Toscana.

REFERENCES

[1] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.

[2] A. Bahga and V.K. Madisetti. A cloud-based approach for interoperable electronic health records (EHRs). *IEEE Journal of Biomedical and Health Informatics,*, 17(5):894–906, 2013.

[3] L. Beernaert, M. Matos, R. Vilaça, and R. Oliveira. Automatic elasticity in openstack. In *Proceedings of the Workshop on Secure and Dependable Middleware for Cloud Monitoring and Management (SDMCMM 12)*, pages 2:1–2:6. ACM, 2012.

[4] S. Di Bona. Benefits of cloud compunting in EHR implementation. In *Proceedings of the Global Forum 2012 : SHAPING A CONNECTED DIGITAL FUTURE, Stockholm (Sweden), November 12th-13th, 2012*. Dedalus SpA, 2012.

[5] G. Brataas, E. Stav, S. Lehrig, S. Becker, G. Kopčak, and D. Huljenic. Cloudscale: Scalability management for cloud systems. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 335–338, New York, NY, USA, 2013. ACM.

[6] Y. Chen, J. Lu, and J. Jan. A secure EHR system based on hybrid clouds. *Journal of Medical Systems*, 36(5):3375–3384, 2012.

[7] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v3.0, 2011.

[8] Cloud Security Alliance. The notorious nine. cloud computing top threats in 2013, February 2013.

[9] DMTF. Open Virtualization Format Specification. Version: 2.1.0. Distributed Management Task Force, 2013.

[10] ENISA. Cloud computing - benefits, risk and recommendations for information security, 2009.

[11] F. Frisch. The identity & access management (r)evolution. Federation and attribute based access control. Axiomatics AB, 2014.

[12] G. Galante and L.C.E. de Bona. A survey on cloud computing elasticity. In *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*, pages 263–270, 2012.

[13] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *Proceedings of the 2010 International Conference on Network and Service Management (CNSM 10)*, pages 9–16, 2010.

[14] N.R. Herbst, S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, pages 23–27. USENIX, 2013.

[15] C.N. Hoefer and G. Karagiannis. Taxonomy of cloud computing services. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1345–1350, Dec 2010.

[16] M. Kuo, Y. Guo, and T. Sahama. Cloud computing for healthcare research information sharing. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 12)*, pages 889–894, 2012.

[17] A. Lazouski, G. Mancini, F. Martinelli, and P. Mori. Usage control in cloud systems. In *Proceedings of the 7th International Conference for Internet Technology and Secured Transactions (ICITST12)*, pages 202–207, 2012.

[18] S. La Manna. The vision of dedalus for interoperability innovation in the ehealth sector. In *Proceedings of the Global Forum 2013 : DRIVING THE DIGITAL FUTURE, Opportunities for Citizens and Businesses, Trieste (Italy), October 28th-29th, 2013*. Dedalus SpA, 2013.

[19] M. Matos, Jr. A. Correia, J. Pereira, and R. Oliveira. Serpentine: Adaptive middleware for complex heterogeneous distributed systems. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 08)*, pages 2219–2223. ACM, 2008.

[20] P. Mell and T. Grance. The NIST definition of cloud computing. recommendation of the national institute of standards and technology, 2011.
[21] OASIS. SAML 2.0 profile of XACML version 2.0, August 2010.
[22] OASIS. XACML v3.0 core and hierarchical role based access control (RBAC) profile version 1.0, March 2010.
[23] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0, January 2013.
[24] J. Park and R. Sandhu. The $UCON_{ABC}$ usage control model. *ACM Transactions on Information and System Security*, 7:128–174, 2004.
[25] B.E. Reddy, T.V.S. Kumar, and G. Ramu. An efficient cloud framework for health care monitoring system. In *Proceedings of the International Symposium on Cloud and Services Computing (ISCOS 2012)*, pages 113–117, 2012.
[26] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. *2013 IEEE Sixth International Conference on Cloud Computing*, 0:500–507, 2011.
[27] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
[28] J. Sermersheim. Lightweight directory access protocol (LDAP): The protocol. IETF Request for Comments: 4511, 2006.
[29] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC 11)*, pages 5:1–5:14. ACM, 2011.
[30] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: Towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, December 2008.
[31] G. von Laszewski, J. Diaz, W. Fugang, and G.C. Fox. Comparison of multiple cloud frameworks. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD 2012)*, pages 734–741, 2012.
[32] M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan. Authentication methods for LDAP. IETF Request for Comments: 2829, 2000.
[33] X. Wen, G. Gu, Q. Li, Y. Gao, and X. Zhang. Comparison of open-source cloud management platforms: Openstack and opennebula. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2457–2461, 2012.
[34] X. Zhang, M. Nakae, M.J. Covington, and R. Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Transactions on Information and System Security*, 11(1):3:1–3:36, 2008.

Edited by: Jesus Carretero
Received: August 31, 2014
Accepted: January 21, 2015

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2_ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.