Phoenix: System for Implementing Private and Hybrid Cloud for OMIC Sciences Applications

Prahalad H.A.[1*], Asoke Talukder[2*], Shubhangi Pardeshi[3#], Sameer Tamsekar[4#], Hari Krishna R.[5#], Chandrashekar M.A.[6#], B. Niket[7#], Santhosh Gandham[8*]

[*]Geschickten Solutions, Bangalore, India

[#] Department of Computer Engineering, NITK, Surathkal, India

[1*]prahalad@geschickten.com, [2*]asoke.talukder@geschickten.com, [3]imshubhangi@gmail.com, [4]stamsekar@gmail.com, [5]harrykris85@gmail.com, [6]chandru.srgp@gmail.com, [7]bniketh@gmail.com, [8*]santhosh.gandham@geschickten.com,

Abstract— Computational Quantitative Biology applications like Genomics, Transcriptomics, Proteomics, Metabolomics and Systems Biology at large require high computing resources that include both processing and storage. Cloud computing provides dynamically scalable on-demand infrastructure in a virtualised environment for processor intensive and data/storage intensive applications. Users need not own this infrastructure; rather use them as and when needed by paying for these resources in pay-as-you-use model that are generally available as a service over the internet. In this paper we present Phoenix - a middleware system for *platform as a service* (PaaS). This paper describes Phoenix as a novel system for implementing GenomicsCloud – A Cloud computing solution designed specifically to solve OMIC sciences problems. It comprises of a vast pool of compute, storage and application infrastructure for processing the data generated by *next generation sequencers* (NGS).

Keywords – GenomicsCloud, Phoenix, Systems Biology, Computational Biology, OMIC Sciences, Cloud Computing, Virtualization, Next Generation Sequencing.

I. INTRODUCTION

Computational Quantitative Biology and applications in OMIC Sciences (Genomics, Transcriptomics, Proteomics, etc.) demand high computing, storage, and networking capabilities. The raw data generated by any experiment or in OMIC sciences often exceed terabytes of data and already challenging the computational infrastructure typically available in many laboratories. Furthermore, biological datasets are having exponential growth doubling every 18 months [1]. These data are used to understand the genetic structure of a species and how to engineer them to the benefit of mankind by either to cure some disease or engineering high-yield crops. The only long-term solution to the challenges posed by the massive data-sets being generated is to combine computational biology research with advances from Cloud Computing.

Systems Biology is another faculty of research where multiple datasets from variety of species are examined. Large-scale data integration in Systems Biology has catalysed identification of nearly all yeast mitochondrial proteins and many of their functional interactions, as well as how this knowledge has aided the search for new disease genes. The human candidate genes proposed can be tested back in the yeast, where cell-based assays can be performed in a high-throughput manner. All these types of research in Systems Biology and applications in OMIC Sciences demand High Performance Computing (HiPC) and high storage capacity [2].

Cloud Computing [3] is emerging as a new style of distributed computing that adapts to dynamically scalable system resource requirements. With Cloud Computing resources – platform, infrastructure and software are dynamically configured and offered to user's on-demand as Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS) and Software-as-a-Service (SaaS) respectively. All these can be combined to build a GenomicsCloud. GenomicsCloud consists of a vast pool of virtualized resources in a hybrid cloud environment with parallelized Computational Biology applications that can either scale-up or scale down on-demand.

Cloud Computing also offer some added advantages; through Cycle Scavenging or shared resources, it can create a "grid" (Grid Computing paradigm) from the unused resources in a network of participants (private or hybrid) nodes. Typically this technique uses desktop computer instruction cycles through virtualization that would otherwise be wasted at non peak-hours like night, during lunch, or even in the scattered seconds throughout the day when the computer is waiting for user input or slow devices [4]. Through this shared infrastructure, it improves energy efficiency, reduces hardware (compute, storage and network) infrastructure requirements, costs, floor space, carbon foot-print, and ultimately increases utilisation.

Because Cloud Computing uses shared resource facility – it is important that the Cloud is managed effectively and efficiently so that service level can be ensured. Phoenix is an attempt to introduce ease of implementing and administering any GenomicsCloud instance.

Also, resources in the cloud are distributed, which makes the management of the virtual resources available in cloud complex. Thus a Virtual Machine Manager (VMM) [5] is required to manage clusters with lesser efforts. The existing VMMs have support for limited hypervisors. Also most of these hypervisor tools have only command line interface – these requires high level of expertise to use them. For Computational Biologists a GUI (Graphical User Interface) based tool is desired. Also, a command based tool requires remote login into the remote system making is more prone to security attacks. The basic philosophy of Cloud is that it will

978-1-4244-7202-4/10/$26.00 ©2010 IEEE

be used over the internet making it necessary that the user-interface is Web-based. Therefore, to make a GenomicsCloud user friendly, it is recommended that an administrative tool for GenomicsCloud be Web-based tool that can be used through a Web browser. Web-based management also makes it possible to be used from even a Smartphone. In addition, the Web-based administrative tool must provide support for maximum and most popular hypervisors.

Phoenix is an Open-source tool developed at National Institute of Technology Karnataka at Surathkal by the Geschickten team and presented here to address all the challenges of a GenomicsCloud. It is designed to be hypervisor agnostic. It supports Virtual Box [6] hypervisor in addition to Xen [7], Kernel based Virtual Machine (KVM) [8] VMware [16] and Amazon Elastic Compute Cloud (EC2) [9]. It uses [14] as backend to manage virtual machines with many feature enhancements from security to rendering. Phoenix has a Web-based interface developed on top of generic messaging API (Application Programming Interface) of OpenNebula. Another novel approach for Phoenix is that is offers higher security through user authentication; also, any third party application can be developed quite easily on top of this generic messaging API of Phoenix.

In this paper we present the design and performance analysis of the GenomicsCloud management tool Phoenix. The paper is organized as following. In Section 2 we present the philosophy of Cloud Computing. In Section 3 we present GenomicsCloud. In Section 4 we provide related work including OpenNebula and Globus Nimbus. In Section 5 we present design, analysis and implementation of virtualization tool “Phoenix” to manage the cloud resources. Further, Section 6 describes functionalities of Phoenix. Section 7 presents the performance analysis results of Phoenix. We conclude the paper with Section 8.

II. Cloud Computing

Cloud Computing can be viewed as a distributed computing system providing compute, storage and network as a service, on-demand on a pay-as-you-use basis. The Cloud appears to users as a single autonomous system providing all the computing infrastructure user needs. It is built on servers having different levels of virtualization technologies. The services provided by cloud are accessible to clients connected via a network infrastructure that may be wired or wireless using thin clients like a browser or a Smartphone.

Many thinkers believe that Cloud Computing is poised to become the defining technology of the 21st century. It is claimed that Cloud Computing will become an essential tool in the world of research and enterprise alike. From aeronautics to life-sciences including finance and computer simulation. Cloud solutions such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) will make inroads into most areas.

As Cloud services need to be scalable, distributed and scale-up or scale-down on-demand, some of the desired characteristics of such an infrastructure are:

- Self-healing
- Self monitoring
- Resource registration and discovery
- Service level agreement definitions
- Automatic reconfiguration

Cloud Computing provides benefits of enormous infrastructure to the users or clients without having to worry about the actual implementation and administration. Since storage service is one of the important services provided in the Cloud, clients have access to multiple data centres from any system in the globe having access to the cluster. Thus Clouds provide more mobility to clients. One of the major advantages of Cloud Computing is that, it is highly automated and scalable. This means clients can add services as and when needed without cost incurred for additional hardware. To support this flexible environment complex cloud infrastructure is built and maintained transparently by Cloud providers in distributed manner. So, clients need not worry about data-centres or implementation and management of services.

Similar concepts were there in the past as well. These were paradigms like Utility Computing, Autonomic Computing and Grid Computing [4]. Cloud is an amalgamation of all these and can be defined as,

- Utility computing - The packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility such as electricity
- Autonomic computing - Computer systems capable of self-management
- Grid computing - A form of distributed computing whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers, acting in concert to perform very large tasks. These grids could well be Processor Grids, Data Grids, or a combination of both.

III. GenomicsCloud

GenomicsCloud is a Cloud Computing Infrastructure for Computational Biology and OMIC Sciences applications. The path-breaking human genome project gave rise to an exponential increase in the volume and diversification of data, including gene and protein data, nucleotide sequences and biomedical literature. Research projects in genetics labs around the world that are researching on genome sequencing, transcriptome studies, etc produce an ever-increasing amount of data; therefore, the area of computational biology now poses some of the biggest challenges in computer science and data mining such as data storage, visualization, modelling, and discovering new meaning out of this enormous data.

GenomicsCloud is a conglomeration of clusters and grids specifically designed for Computational Biology applications. The computing infrastructure can be a private grid-computing infrastructure (private cloud) formed either out of the desktop computers that uses the philosophy of CPU Scavenging or Cycle Scavenging or dedicated High Performance Computing (HiPC) systems along with public clouds (likes of Amazon and Google). GenomicsCloud includes any cluster as well the enterprise might have. In addition, GenomicsCloud has the

capability to extend over the public cloud infrastructure in a secured fashion. GenomicsCloud is managed through Phoenix (discussed in Section V & VI) makes it quite easy to manage the hypervisors in the private Cloud or even public Cloud.

Providing an infrastructure like Phoenix to manage PaaS and IaaS is not sufficient; therefore, GenomicsCloud also provides tools for parallelization. This includes both tightly-coupled parallelization and loosely-coupled parallelization mechanism. There are many OMIC sciences applications that are inherently serial; examples are Bruijen Graph [10] Assembly following a Sequencing experiment. For such applications a tightly-coupled cluster with parallel code is more suitable. On contrast, for Sequence analysis or Gene hunting applications loosely-coupled grids will produce better results with the help of algorithms like MapReduce [11].

IV. Related Work

In this section we present various Virtual Machine Manager Tools and the underlying philosophy used to manage virtual infrastructure in distributed system.

A. *Eucalyptus*

Eucalyptus [12] is an open source tool, which can be used to create in-premise private and hybrid cloud. The current interfaces to Eucalyptus are compatible with Amazon's EC2, Simple Storage Service (S3) [9], and Elastic Block Storage (EBS) interfaces [9], but the infrastructure is designed to support multiple client-side interfaces. It is based on commonly available UNIX tools and web services.

B. *Nimbus*

Nimbus is another Open Source toolkit [13] which is developed to build scientific Clouds. It is used to create IaaS type of clouds. The Nimbus contains a frontend Globus service and multiple workspace control agents on host resources for virtual machine deployment. Nimbus allows a client to deploy virtual machines (VMs) on physical resources to lease remote resources and configuring them to represent an environment as per the user requirements.

C. *OpenNebula*

OpenNebula is a VMM that enables the dynamic deployment and replacement of virtualized services. It is an open source virtualization infrastructure engine. It is used to deploy, monitor and control virtual machines. Open Nebula currently supports KVM and Xen hypervisors along with public cloud interface Amazon EC2 over command line interface.

OpenNebula system consists of one OpenNebula server called front-end and multiple number of cluster nodes on which virtual machines are deployed. Front-end communicates with cluster nodes using Secure Shell (SSH) protocol. OpenNebula currently supports NFS (Network File System) and SSH protocols to transfer virtual images from image repository. The front end has command line user interface (CLI) to create and manage virtual machine and cluster nodes in system cluster.

V. Phoenix – the Open Source Framework

Phoenix is a framework that is designed to manage a large cloud that comprise of Grids, Clusters, and Clouds; these could be either in a private space or in the public space. The Phoenix framework consists of two modules viz. Front-end for user interface built on top of OpenNebula core and OpenNebula drivers for interfacing different hypervisors which extended from core as shown in Fig. 1. We discuss these modules and OpenNebula in subsequent subsections.

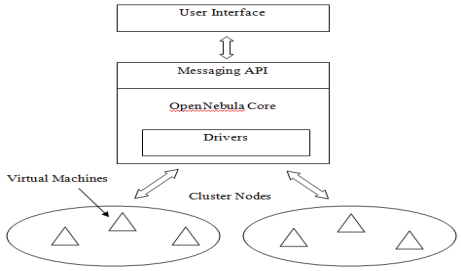

Fig. 1. Phoenix Architecture

The communication between drivers and core is performed using simple ASCII protocol, which simplifies development of new drivers.

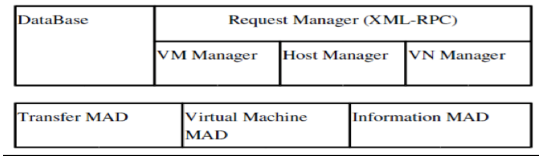

Fig. 2. OpenNebula Core

Following are the three Middleware Access Drivers (MAD) in OpenNebula:

- Transfer MAD: Transfer driver is used by transfer manager to transfer virtual machine images and checkpoint files.
- Information MAD: Information driver is used while deploying and monitoring virtual machine to gather information of physical resources.
- Virtual Machine MAD: It is used to deploy and control virtual machines.

VI. PHOENIX - ARCHITECTURE

Architecture of Phoenix core is shown in Fig. 2. The system is divided into three major components comprising static OpenNebula at the top, which is common for any hypervisor in use. The middle layer is the pluggable driver modules. Different Information and Virtual Machine drivers are required to interact with different hypervisors. This figure shows drivers for VirtualBox but similar type of drivers for other hypervisor may also exist when activated by the administrator. The lowest layer component is cluster node on which virtual Machine is deployed. The interaction of Cluster Node with VirtualBox hypervisor managed by `libvirtd` [15] daemon is shown in the diagram.

The command line utilities are available in OpenNebula to send commands to OpenNebula engine. These utilities are XML-RPC clients and calls XML-RPC methods exported by Request Manager Module in OpenNebula core exports; these are called by XML-RPC clients such as command line utilities provided by OpenNebula. All manager processes are created when OpenNebula daemon is started. These manager processes are responsible to call respective drivers for communication with underlying hypervisor on cluster nodes. The MAD layer in OpenNebula core provides uniform access to these drivers. To perform tasks such as deploying VM, gathering information etc., VMM and IM drivers issue libvirt based commands to execute on cluster node using SSH. On the cluster node, libvirtd daemon communicates with hypervisor. Libvirtd can be configured to use TLS, SSH, TCP or UNIX sockets for communicating with the hypervisor.

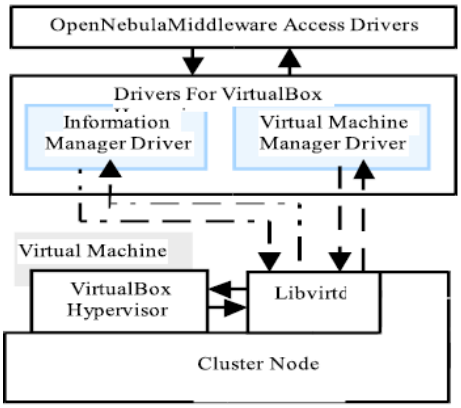

Fig. 3. Connector flow Diagram

VII. FUNCTIONALITIES OF PHOENIX

The Front-end of Phoenix provides web based user interface to create and manage the virtual machines. The major components of user interface are host management, virtual machine management (VMM), and virtual network management. In addition to these cluster component management, Phoenix provides facility to administer users. With the built-in access control mechanism users can view or manage virtual infrastructure.

This Web based management console is basically divided into four subsections viz. User Administration, Hosts Management, Virtual Machine Management and Virtual Network Management (Fig. 3).

A. User Administration

User administration is provided to users of Phoenix management console. It provides facilities such as new user registration (creation), login facility, changing user information, profile etc. It also provides user statistics such as login count, last login request time and IP address, Current request time and IP address.

B. Hosts Management

Hosts management provides addition of cluster nodes where administrator can deploy virtual machines. Summary information provides details such as current state, monitoring drivers, and supported actions such as hardware usage information, capacity of the hosts, memory usage, CPU usage, number of virtual machines running etc.

C. Virtual Machine Management

Virtual Machine management gives an option to create a virtual resource and deploy it on any of the available pool of hosts. Once it is deployed, it is continuously monitored and its status is shown on the summary page of Virtual Machines.

D. Virtual Network Management

In Phoenix, we can define certain networks, having a pool of available IP address and MAC address which could be assigned to a Virtual Machine deployed on a Host. Using these assigned addresses, Virtual Machines can make use of Network adapters of Hosts to communicate with each other and create a subnet among them.

VIII. CHARGING BILLING AND AUDIT TRAIL

All actions requested from the management console are logged as usage information. These are collective logs which are user specific. This usage information can be used to audit the usage. This can also be used to reconcile the charges user might have been asked to pay to a public cloud provider.

For GenomicsCloud hosted on top of Phoenix to be provided as a SaaS (Software as a Service), requires some billing and charging capability. There are basically two ways by which a service provider can charge the user for using the GenomicsCloud service. Charging is mainly divided into three categories, viz., one-time fee, recurring fee, and usage-based fee.

1. There could be a one-time fee during registration or provisioning of service that is similar to license fee.
2. User can pay a fixed amount (recurring) and rent the specific set of physical resources for a specific amount of time. E.g. User can rent 4 CPU machines with 2GHz processing power and 10G Hard disk space for a month. Phoenix provides the machine credentials which would be valid for a month and can be managed via Phoenix management console. This service can be charged for specific levels of authorization. E.g. Professional license will allow user to save the state of machine, migrate the machine to free physical resource, while Basic license will allow only start and stop actions.
3. Second way to charge the user is to charge the user based on their usage. This uses logging of usage details of every action user performed and charge it according to its use. This is useful for refund or SLA (Service Level Agreement) management.

IX. MAJOR FINDINGS

Phoenix tool was found to be better compared to other virtual machine managers such as OpenNebula, Eucalyptus and Globus Nimbus, based on factors like usability, flexibility and supportive to cloud infrastructures. Table 2 shows comparison of Phoenix with other VMMs. The graphical user

interface makes this tool user friendly and easy for VM management even for users who are not computer savvy like users from Biology background. Also additional support for VirtualBox overweights other VMM engines including OpenNebula.

TABLE 1. COMPARISON RESULTS

Category		Eucalyptus	GlobusNimbus	OpenNebula	Phoenix
Virtual Machines	Xen	Yes	Yes	Yes	Yes
	KVM	Yes	No	Yes	Yes
	VMWare	No	No	Yes	Yes
	VirtualBox	No	No	No	Yes
Cloud Interfaces Supported	Amazon EC2	Yes	Yes	Yes	Yes
Cloud Infrastructures	Simple(Public/Private)	Yes	Yes	Yes	Yes
	Hybrid	No	No	Yes	Yes
User Interfaces	Command Line Interface	Yes	Yes	Yes	Yes
	Graphical User Interface	Yes	No	No	Yes

TABLE 2. PERFORMANCE EVALUATION

Actions	Open Nebula (Sec)	Phoenix (Sec)	Difference (Seconds)	Increase
Host Creation (1)	0.1	0.104	0.004	4%
Host Deletion (2)	0.15	0.155	0.005	3.33%
VM Creation (3)	0.2	0.208	0.008	4%
VM Deletion (4)	0.19	0.196	0.006	3.15%
			Average Increase	3.62%

To test the performance of the tool we used following scenario. System specifications of the machines used are as follows: CPU: Dual Core 2 GHz, RAM: 2 GB, OS: Ubuntu Linux 8.10.

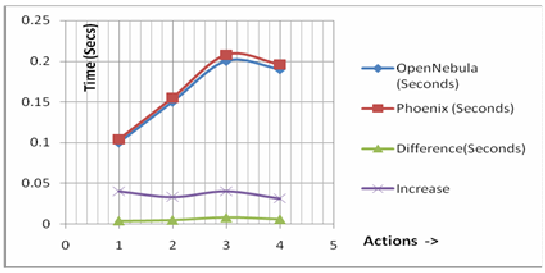

Fig. 4. Graph plotted using data mentioned in Table 3

Three machines connected in Ethernet LAN have been used in which one of them acted as an OpenNebula server and has Phoenix Web based management console, with messaging daemon always on run. Other two machines are configured as cluster nodes. We tested the delay introduced due to additional messaging API layer introduced. Results are presented in table 3 and Fig. 4.

X. FUTURE WORK

In addition to the features described in section VI, the following feature set currently under development will be available in the future releases.

1. Management of virtual machine templates on the server
2. Web-based VNC interface for managing virtual machines
3. Robust Access control and multi-user management
4. Phoenix as GenomicsCloud administration framework with Quality of Service (QoS) for guaranteed service level at real-time.

XI. CONCLUSION

In this paper we presented challenges involved in Computational Biology applications where enormous computing power and storage is required. In this context, we presented virtual infrastructure management framework Phoenix that aims to extend existing command-line based management tools into a Web-based tool with security added. This is an Open-source framework built above OpenNebula to support VirtualBox and other standard hypervisors. The drivers developed are seamlessly integrated with current codebase with the Web based graphical front-end. The Phoenix framework is to manage GenomicsCloud, the Cloud Computing environment aligned to specific needs of OMIC sciences applications. Performance results of Phoenix indicate that the overhead induced is very less compared to functionalities introduced in this ecosystem.

REFERENCES

[1] GenBank Statistics: http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

[2] Fabiana Perocchi, Eugenio Mancera and Lars M. Steinmetz, Systematic screens for human disease genes, from yeast to human and back, Mol. BioSyst., 2008, 4, pp 18–29

[3] Brian Hayes, Cloud computing, Communications of the ACM, v.51 n.7, July 2008

[4] Wikipedia, The Free Encyclopedia: www.wikipedia.org

[5] James E. Smith, Ravi Nair, The Architecture of Virtual Machines, Computer, vol. 38, no. 5, pp. 32-38, May, 2005.

[6] VirtualBox – [URL]. Http://www.virtualbox.org, access on May. 2008

[7] Paul Barham , Boris Dragovic , Keir Fraser , Steven Hand , Tim Harris , Alex Ho , Rolf Neugebauer , Ian Pratt , Andrew Warfield, Xen and the art of virtualization, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA

[8] Kernel Virtual Machine [URL]. http://www.linux-kvm.org

[9] Amazon Web Services [URL]. http://aws.amazon.com/, access on May. 2008

[10] Daniel R. Zerbino, Ewan Birney, Velvet: Algorithms for de novo short read assembly using de Bruijn graphs

[11] Jeffrey Dean and Sanjay Ghemawat, MapReduce: Simpli ed Data Processing on Large Clusters, Open System Design and Implementation (OSDI 2004).

[12] Eucalyptus [URL]. http://www.eucalyptus.com/ , access on May2008

[13] Globus Nimbus [URL]. http://workspace.globus.org/, access on May 2008

[14] OpenNebula Project [URL]. http://www.opennebula.org/, access on May. 2008

[15] Lizhe Wang, Jie Tao, Marcel Kunze, Dharminder Rattu, Alvaro Canales Castellanos, The Cumulus Project: Build a Scientific Cloud for a Data Center, Cloud Computing and Applications-08, October 23rd and 28th, 2008

[16] VMware: www.vmware.com