Cloud Computing for Control Systems

CERN Openlab Summer Student Program

9/9/2011

ARSALAAN AHMED SHAIKH

CONTENTS

INTRODUCTION

The project *Cloud Computing for Control Systems* aimed to design and configure a complete private cloud infrastructure for control system applications. The aim of the project is to enable small scale teams to setup test environments for development and experimentation purposes. Cloud technology enables virtualization of multiple operating systems as independent software applications. Virtualization was introduced as early as the 1960's by IBM, enabling users to execute their applications simultaneous as well as independently.

Since its early days the technology has evolved, allowing the virtual instances to run in privileged mode and access the underlying hardware directly. Cloud computing has taken this technology to enable enterprise level deployments of virtual machines and better utilization of vast resources such as data centers.

The motivation behind the project is the wide user community at CERN, which comprises of development, test and operational teams. Virtualization managed by management software which is the cloud technology facilitates rapid development.

The commercial success of Amazon cloud and other such enterprise solutions has motivated the open source community, which in turn has provided solutions such as *Eucalyptus, OpenStack and OpenNebula.* Out of the three OpenNebula was selected as the cloud technology for this project as it fulfilled the requirement of supporting hypervisors for both linux and windows operating systems. In addition to that the user friendliness and easy to manage capabilities were also contributing factors in the decision.

The hardware used for the project is HP Proliant 380 G4 machines with 8 gigabytes of main memory, 500 gigabyte secondary storage. The machines were connected to high bandwidth network connections.

SYSTEM COMPONENTS

The project contains many components the four major components are *OpenNebula Cloud Management Toolkit, VMware ESX Hypervisor, OpenStack Image Registry and Delivery Service(Glance) and Instance Management Agent.*

OPENNEBULA CLOUD MANAGEMENT TOOLKIT

OpenNebula is an IaaS (Infrastructure as a Service) open source solution which allows building private, public and hybrid clouds. It has been designed to be able to integrate with any kind of network or storage system and supports the main types of hypervisors: KVM, VMware ESXi and XEN.

OpenNebula has five major components which are mentioned below.

OpenNebula daemon – Responsible for handling all incoming requests either from the command line interface or API.

Scheduler – Selects the best available host on which to deploy a specific virtual machine.

Information Manager – Collects resource availability and utilization information from hosts and instances respectively.

Transfer Manager – Responsible for image management; tasks such as cloning and deleting instances.

Virtual Machine Manager – It provides an interface to the underlying hypervisor. All operations that have to be performed on a virtual machine go through this interface.

Hook Manager – Responsible for executing virtual machine hook programs which are automatically triggered when the state of the virtual machine changes.

OpenNebula also has a database where it saves information.

With respect to the storage, OpenNebula works with three possibilities: shared - NFS (there is a shared data area accessible by OpenNebula server and computational nodes), non-shared - SSH (there is no shared area - live migrations cannot be used) and LVM (there must be a block device available in all nodes).

VMWARE ESX HYPERVISOR

VMware ESX is an enterprise-level computer virtualization product offered by VMware, Inc. ESX is a component of VMware's larger offering, VMware Infrastructure, and adds management and reliability services to the core server product. The original ESX is being replaced by ESXi.

VMware ESX and VMware ESXi are bare-metal embedded hypervisors that are VMware's enterprise software hypervisors for servers that run directly on server hardware without requiring an additional underlying operating system.

The basic server requires some form of persistent storage (typically an array of hard disk drives) for storing the hypervisor and support files. A smaller footprint variant, ESXi, does away with the first requirement by permitting placement of the hypervisor onto a dedicated compact storage device. Both variants support the services offered by VMware Infrastructure

IMAGE REGISTRY AND DELIVERY SERVICE (GLANCE)

Glance provides services for discovering, registering, and retrieving virtual machine images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image. Future releases will have the ability to convert virtual machine image from one format to another

VM images made available through Glance can be stored in a variety of locations from simple filesystems to object-storage systems like the OpenStack Swift project.

Glance has two main components, Glance API server is the main interface. It routes requests from clients to registries of image metadata and to its backend stores, which are the mechanisms by which Glance actually saves incoming virtual machine images. It supports *Swift, Filesystem, Amazon's S3*

service and HTTP. Glance Registry Server is the second component it is a web service that adheres to the Glance RESTful API for image metadata.

INSTANCE MANAGEMENT AGENT

Instance management agent is a software component developed in this project as additional management service that resides inside each of the ESXi hypervisors it clones a predefined number of images on the ESXi node. It is an enhancement to the *ssh* based distributed storage architecture used in OpenNebula, it provides fast deployment of instances. This component will be explained in detail later.

ARCHITECTURE PERFORMANCE EVALUATION

Opennebula supports different architectures for cloud computing configuration the main difference between the architectures is how a storage component is organized and the way it is accessed. There was a need to evaluate the performance of the architectures and optimize them to better accommodate the project needs.

PERFORMANCE METRICS

The architectures were judged on the following metrics.

- CPU Utilization (percentage)
- Overall virtual machine deployment time (minutes)
- Individual virtual machine deployment time (minutes)
- Disk Utilization (average block read and writes / second)

EVALUATION TESTS

To measure the performance certain tests were planned based on the serveral different factors. The test consists of *Baseline, Single Interval, Multi Interval, Single Burst and Multi Burst.*

In each of the tests mentioned above it was assumed that at time *'t0'* the OpenNebula frontend server receives 10 simultaneous virtual machine deployments request and each virtual machine to be deployed has the same size of 10 gigabytes. It was also assumed that during the deployment of these virtual machines no other activity takes place. A small JAVA program was written to simulate a queuing behavior for the test.

BASELINE TEST

In the baseline test no parallel deployments were allowed. The test was designed to simulate a queue like behavior where at most only one virtual machine can be deployed. In this test all the virtual machines were deployed on the same hypervisor. The objective was to establish criteria from which to analyze the performance the architecture under the other test.

SINGLE & MULTI INTERVAL TEST

In single interval the all virtual machines were deployed on a single ESXi hypervisor, hence the word *single* in the name of the test. The test simulates a queue but also allows a degree of concurrent deployments. As assumed the system receives 10 simultaneous deployment requests and test adds them to a queue. Each virtual machine is deployed after the lapse of a predetermined interval in time. In the case of the tests conducted the interval selected was of three minutes. Multi interval test was designed to mirror the functionality of single interval; the only difference between the two was that multi interval equally distributed the virtual machines on to three ESXi hypervisors.

SINGLE & MULTI BURST TEST

In burst test there is no queuing mechanism employed, all deployment requests are executed at the same instant they are received. In assumption that all 10 requests were received at t0 therefore all deployments commands were executed. The tests differentiate on the basis of the number of ESXi hypervisors involved, one in the case of Single Burst and three in the case of Multi Burst.

CLOUD ARCHITECTURE

As mentioned earlier one of the important objectives of this project was to not just setup a cloud infrastructure but to thoroughly study different architectures and their performance. Components in a cloud are loosely coupled and can be manipulated to fit different architectures suiting the existing infrastructure.

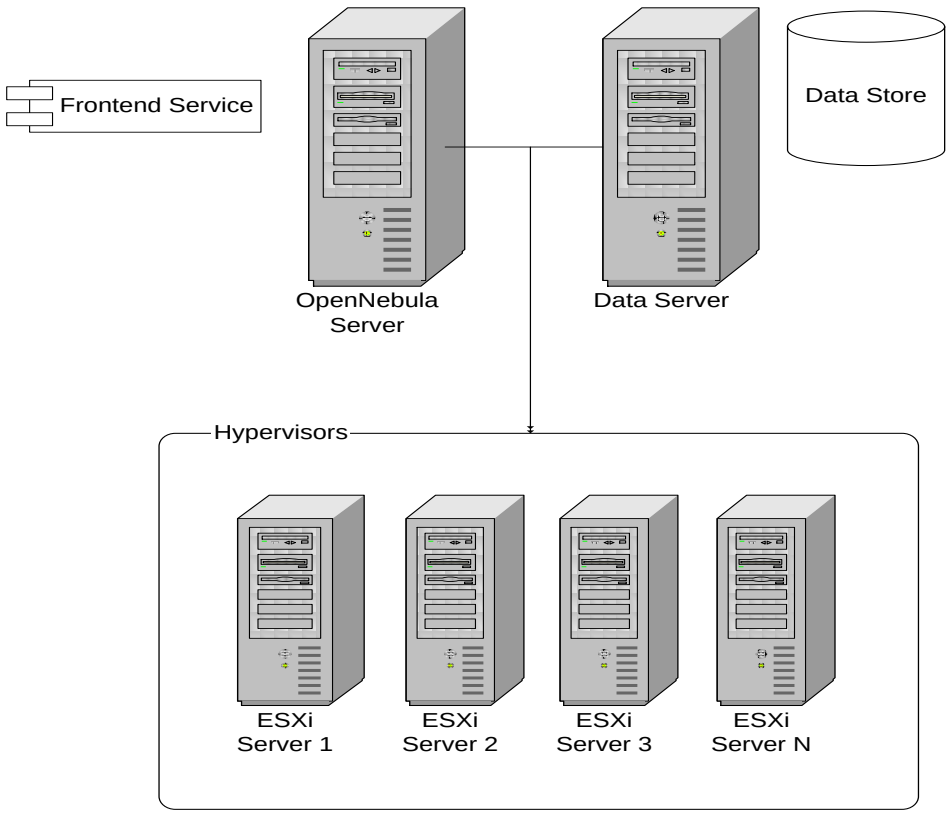
Under this project three basic architectures were implemented and experiments were conducted over them to understand their advantages and disadvantage and after analyzing the results a few components were developed to enhance the capability and performance according to pre existing hardware.

Following are the architectures that were implemented:

- NFS Based Central Storage Architecture (Frontend and Data Server on Separate Machines)
- NFS Based Central Storage Architecture (Frontend and Data Server on Same Machines)
- SSH Based Distributed Storage Architecture

NFS BASED CENTRAL SHARED STORAGE ARCHITECTURE (FRONTEND AND DATA SERVER ON SEPARATE MACHINES)

In this architecture a separate server is used for data storage, the data storage server is the main point of all communication, all the worker nodes as well as the frontend server must have the server drive mounted.



The data server holds the image repository, virtual instances data, OpenNebula database and all OpenNebula configuration files hence it becomes the focal point the architecture and also the central point of failure which is a disadvantage. But on the other hand by decoupling the storage from the frontend, the time to recover in the case of a server crash on the frontend is drastically reduced. Another frontend server can be up and running with same configuration, image repository and instance repository in less than 15 minutes. In addition to that the worker nodes only have to communicate with the data server therefore even if there is temporary failure at the frontend there will be no effect on the users who are using the deployed virtual machine.

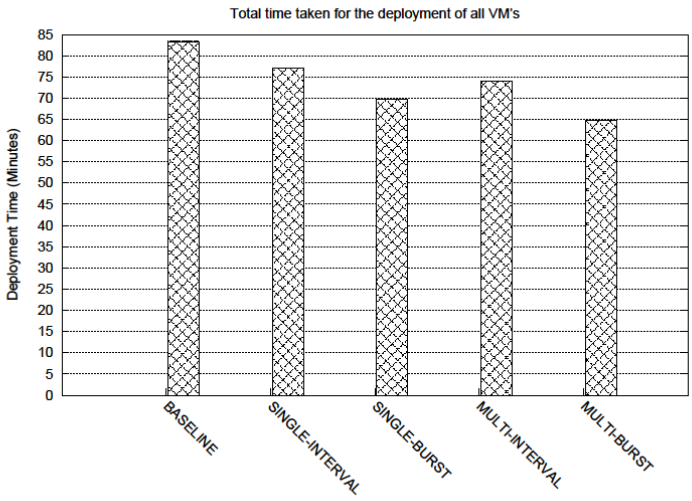
One drawback of having separate frontend and data servers is that communication between them takes place over the network which introduces a level of overhead. OpenNebula when deploying a virtual machine creates a clone of the base image. The base image resides in the image repository on the data server and the clone also must be copied on the data server. The frontend reads blocks of data over the network and then again sends the same the blocks back to the data server. This issue not only increases the deployment time but also utilizes unnecessary network bandwidth.

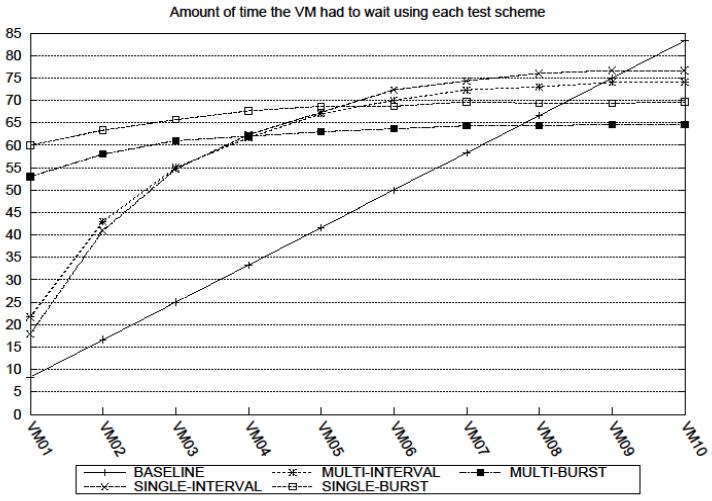
PERFORMANCE EVALUATION

On all of the tests performed on this architecture the total time taken to deploy all the 10 virtual machines does not differ to a great extent (maximum difference of 15 minutes). This test somewhat indicates that the entire tests performed the same, this is misleading the individual virtual machine deployment wait time must be viewed along with the total deployment time. The result shows that

Baseline performed poorly overall, but up to the seventh deployment it has the most VMs that are ready for usage while at the same instant of time intervals tests only 3 VMs and burst test have 0 VMs ready for usage.

None of the tests achieve minimum criteria of performance which was set at maximum 15 minutes of a single machine deployment apart from VM1 in baseline test.

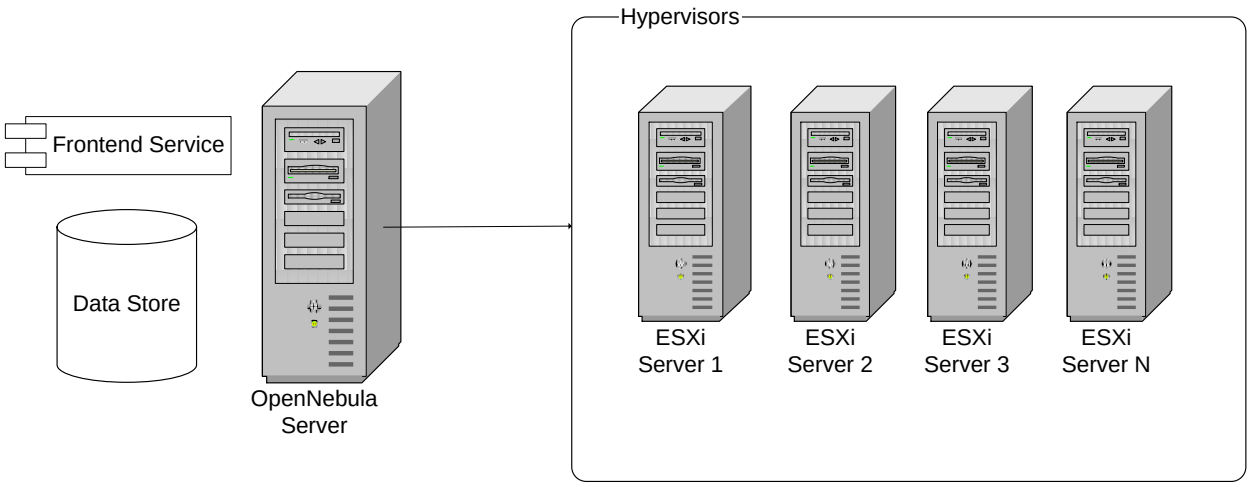




NFS BASED CENTRAL SHARED STORAGE (FRONTEND AND DATA SERVER ON SAME MACHINE)

In the second variant of the central storage architecture the frontend and data services reside on the same machine. The worker nodes access the OpenNebula server by mapping the NFS drive and using the NFS communication protocol.

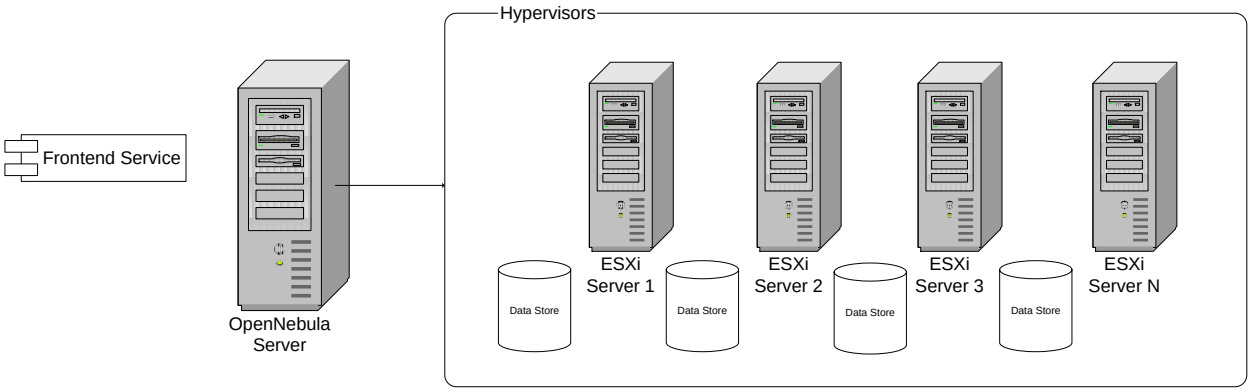
This architecture drastically increases the risk of single point of failure, as now the OpenNebula server becomes the focal point of communication. All the advantages gained in the previous architecture are lost. The time to recover the frontend server amplifies and if backup of the data repositories are not taken then there are chances of data loss. Any failure to the OpenNebula server will directly affect the users of the virtual machines.



Even though this architecture has serious problems, it does has some advantages the cloning process while deploying a virtual machine which is the most time consuming part is reduced due to no network overhead due to which the deployment time of virtual machine decreases, in addition to that there is no unnecessary bandwidth consumption.

SSH BASED DISTRIBUTED STORAGE

This architecture is designed to fully optimize the utilization of resources. The data store in the cloud infrastructure as has three components OpenNebula configuration files and database which do not require a lot storage capacity, Image repository which contains all the base images from which to clone the virtual machines, the image repository can be large depending on the number of different base images required and finally instance repository which contains the deployed virtual machines this component has the largest storage requirements.



In this architecture the image repository and configuration files remain on the frontend server but the instance repository is distributed over the network utilizing the local storage of the worker nodes. The architecture reduces the risk of central failure. If the OpenNebula server fails then worker nodes continue working and the users observe no affect.

The time required recovering or to setup a new OpenNebula server is less then that required by the second architecture but more than the first. The image repository is not updated regularly thus not requiring frequent backups, on the other hand configuration files must be backed up regularly.

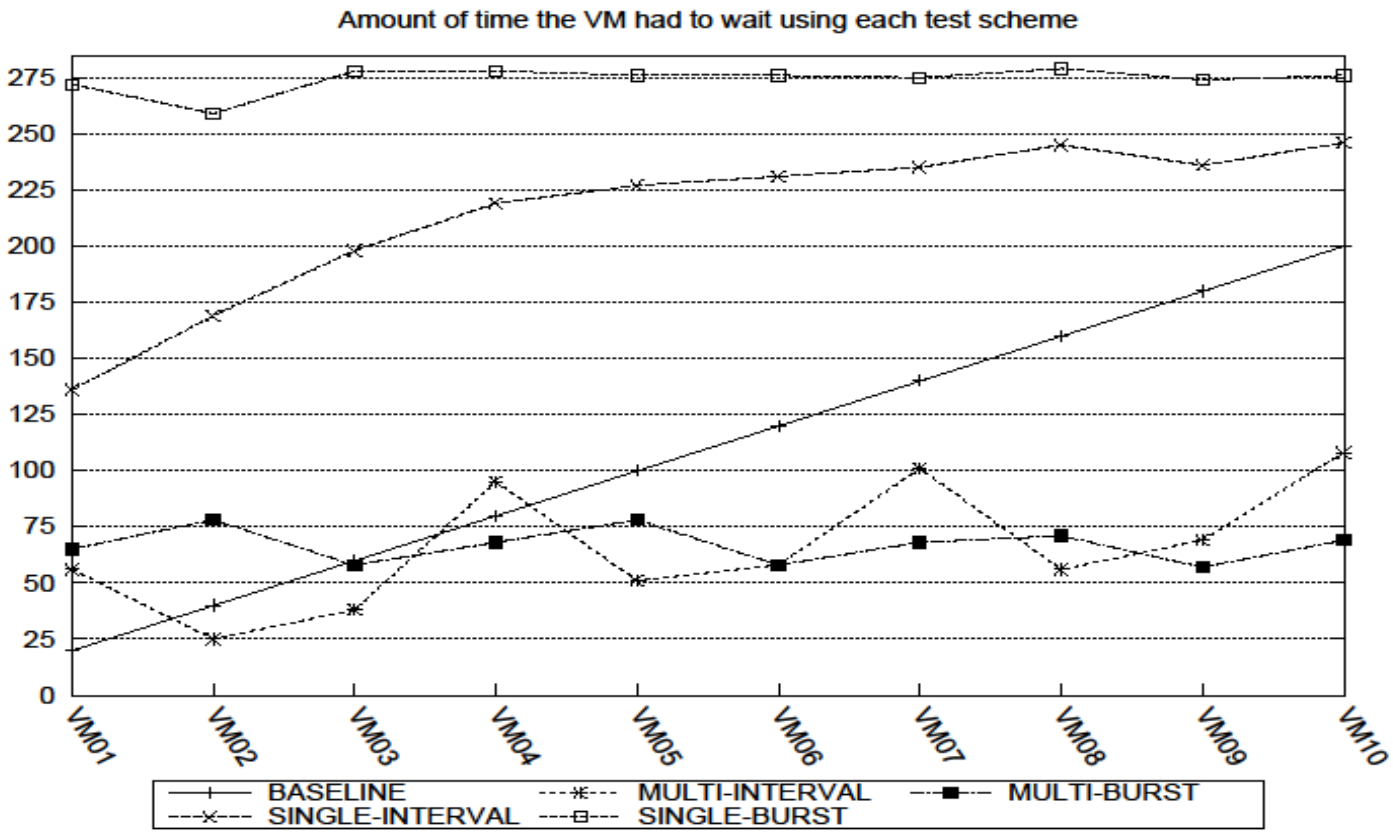
In terms of this project and hardware currently at disposal this is the ideal architecture as all worker nodes have a sufficiently large storage capacity.

PERFORMANCE EVALUATION

Theoretically this architecture is designed to perform the fasted in terms of deployment time as the disk I/O is distributed over several worker nodes as compared to all load on a central storage. But after the compilation of results, the performance measures were opposite. There was major increase in wait time for individual virtual machine and also total deployment time in some cases crossed four hours. These results were a major setback in the project as this was the desired architecture.

Investigating the causes for below par performance for this architecture two main contributing factors were established.

1. The *dropbear* SSH version used by VMware ESXi hypervisor is not particularly optimized for large size file transfer. Copying the same file on ESXi server through SSH takes more than twice the time required by NFS file transfer.
2. ESXi hypervisor uses vmware's virtual machine file system (vmfs), vmfs is optimized for virtual machine I/O. But the initial copying of large files requires more time.



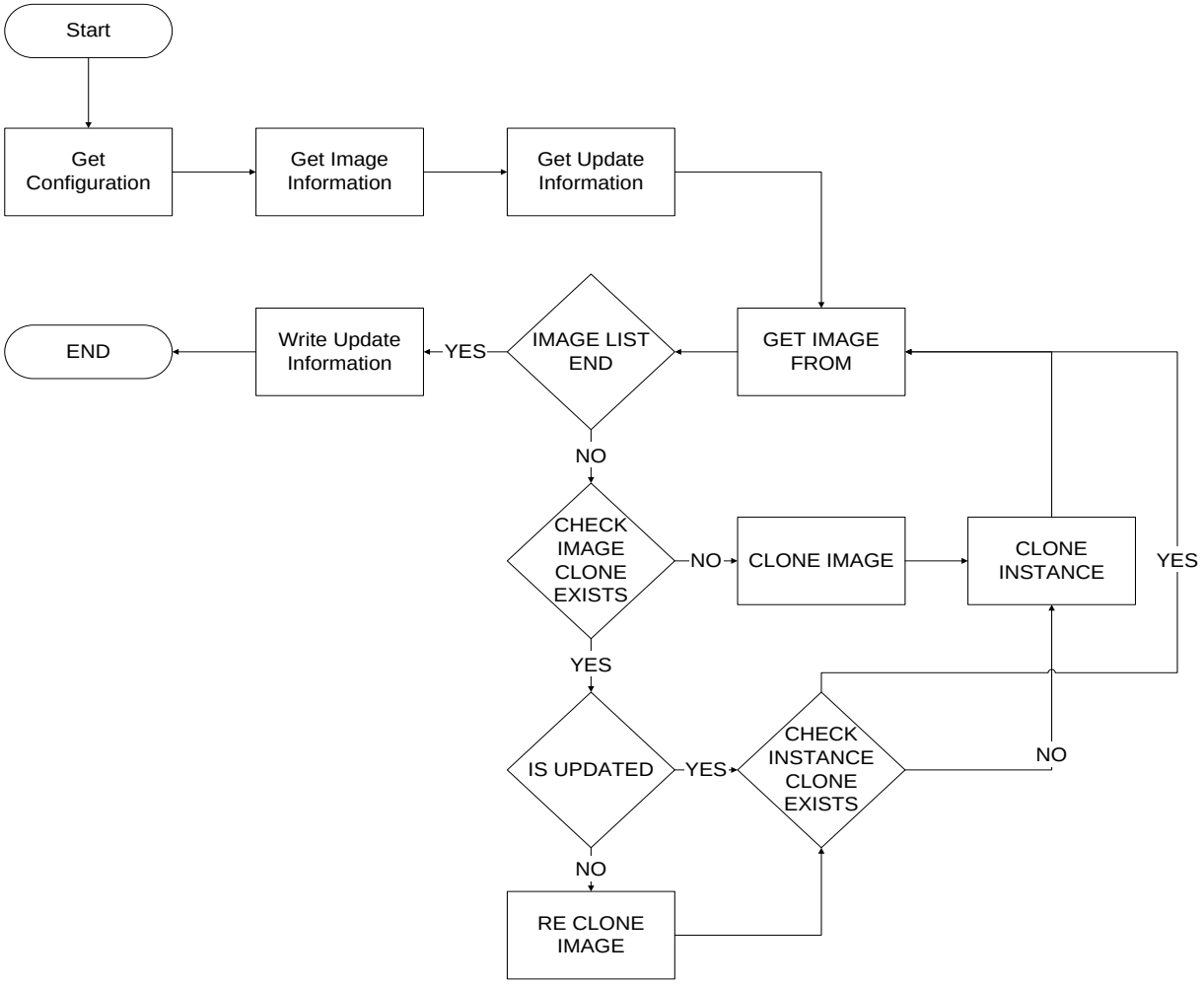
SSH BASED DISTRIBUTED STORAGE WITH INSTANCE MANAGEMENT AGENT ARCHITECTURE

After the performance evaluation of architecture 3, it is observed that even though it is best suited to the current needs and hardware available for the project. The performance is the worst and is not feasible for operational use. Therefore an optimized solution was design to enhance the performance of the architecture. The optimization required the development of a new component the *Instance Management Agent* and changes to the transfer component of OpenNebula.

INSTANCE MANAGEMENT AGENT

The agent will be installed in every worker node and its main role was create a local back of the most frequently used base images in their own local data store and create ready to use base image clones. But the agent required a more sophisticated image repository, which is why it was decided to use *OpenStack Glance* image repository.

The agent is executed on predefined interval (Cron jobs can be used for that purpose). Each agent has its own configuration file which holds information regarding *URI* of the Glance RESTful API and the number of pre cloned base images. The agent reads these configurations on every cycle of execution. It also maintains update information for all the base images. If the image source has been updated in the main repository then it synchronizes the local repository.



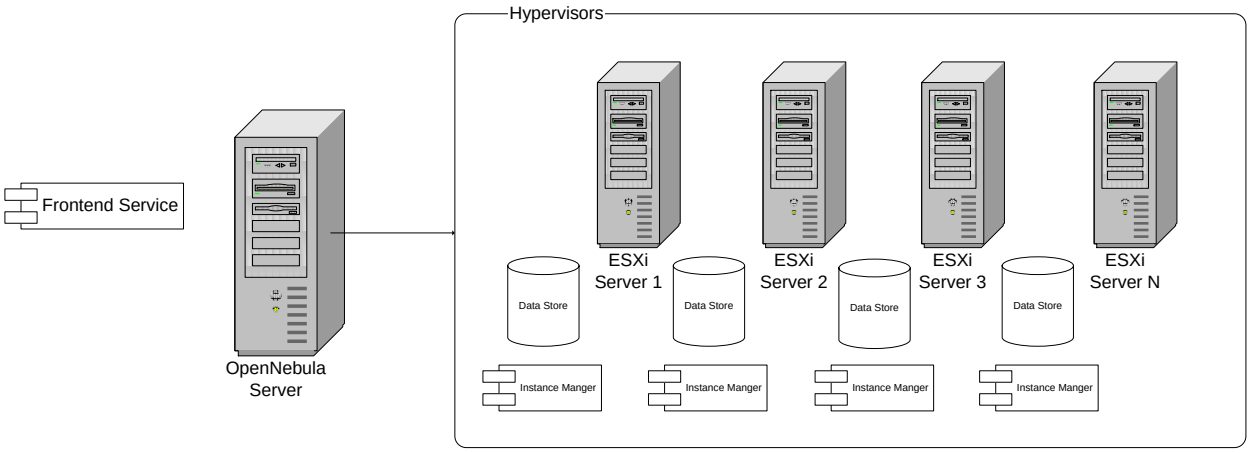
The agent then gets the list of base images from the main Glance repository using a webservice call, the image lists are checked if they exist in the local repository, if it is the updated image and then checks if the instance clones exist. If all the conditions satisfy it moves on to the next image otherwise takes necessary step. The above flow chart explains the tasks performed by the agent.

The OpenNebula transfer component for vmware has also been modified to check whether a clone already exists on the target worker node if not it reverts back to normal working of architecture 3.

ARCHITECTURAL BENEFITS

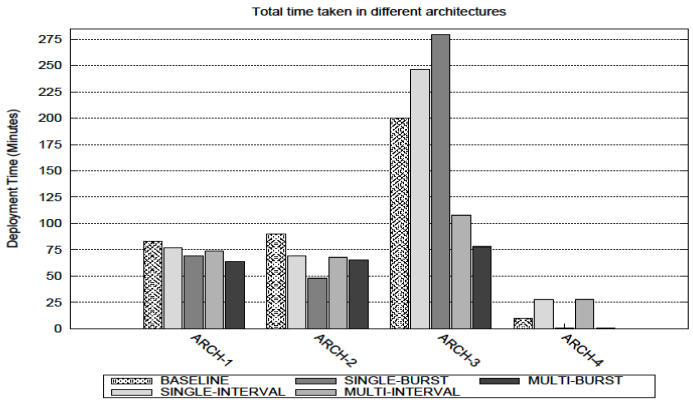
This architecture not only fits the requirements but also inherent reliability. The main image repository is backed up in local repositories on each of the worker nodes which reduces the probability of losing

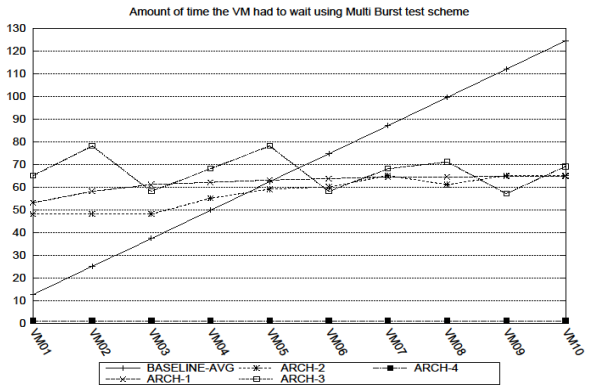
data and there is no single point of failure. If any of the worker nodes or frontend server fails it will not have any bearing on any other node in the system. The effect of failure is localized.



AGGREGATED EVALUATION

Comparing the performances of all the architectures with optimized solution shows increase in performance many folds, by pre cloning the virtual machines in a background process has not only improved on the previous minimum deployment time attained but also achieved our goal for deploying individual virtual machines under 15 minutes.





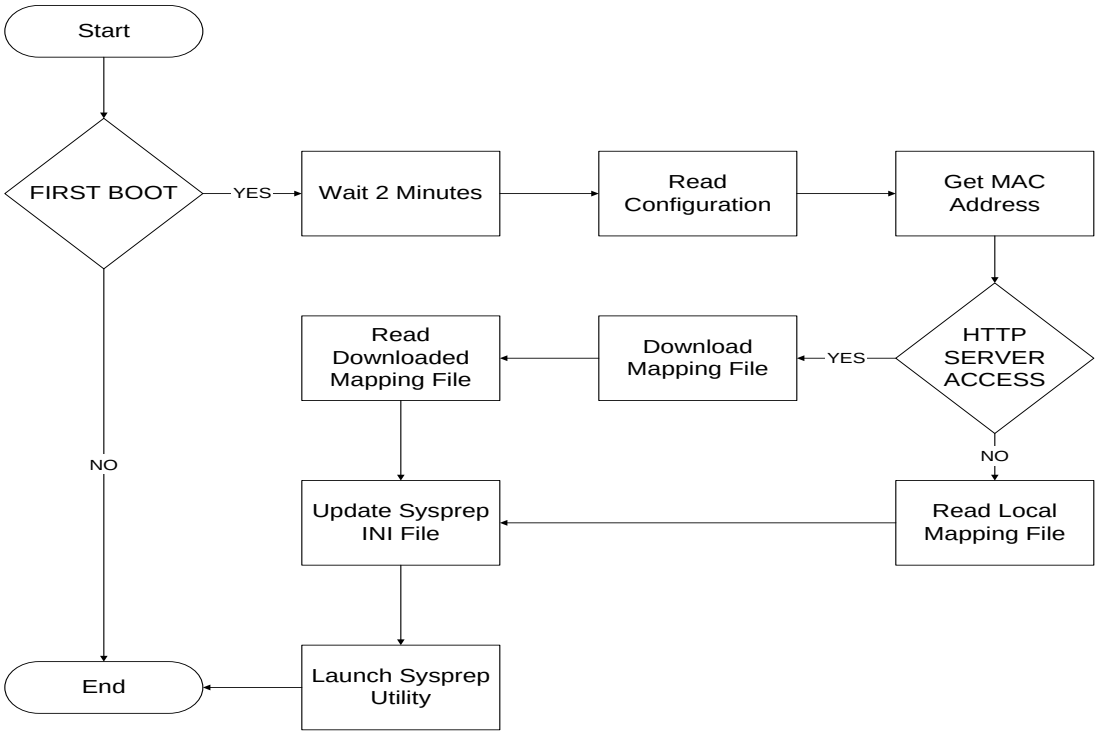
AUTOMATION

One of the major goals of this project was to make the whole process less human intensive as possible. Two areas needed the most focus, the initial setup of the private cloud management tool and the deployment process of a virtual machine.

The deployment process which starts with a user issuing a request, is pretty much stream lined the problem occurs with providing instance level contextualization; it generally involves leasing of recourses from the hypervisor such as MAC address, IP address and changing the host name. The ESXi and OpenNebula Frontend servers communicate to register the virtual machine and also provide it with a MAC address preconfigured in the virtual machine template file on OpenNebula while the IP address is automatically assigned by a DHCP server. What remains an issue is the hostname, especially in windows operating system even though the cloned instance has its own MAC and IP address it still has the same host name of the base image, therefore the instance is not accessible on the network and the user cannot access the it until a system administrator manually changes the host name of the virtual machine.

AUTOMATED INSTANCE LEVEL CONTEXTUALIZATION

A VB script was developed for automating the host name change which utilizes the system preparation utility of windows operating system. The script is embedded in the base image along with the utility and is automatically lunched the first time the deployed system boots. Following figure illustrates the flow of the process.



Every time the virtual machine is booted the script is launched by the OS. The script checks whether the instance is booted for the first time or the contextualization process was correctly executed by checking if the *sysprep* folder exists, if yes then it waits for the instance to complete other startup processes this is necessary as the script accesses an http server over the network and the Ethernet controller must be initialized. But as the there is no actual approach to verify network accessibility a failsafe mechanism is employed. After the wait state is over the script reads the configuration file which holds the URL of the http server where the mapping file is located, it consists of a key value pairs of MAC addresses and their corresponding hostnames. The configuration file also holds an optional parameter for host name prefix such as 'LABO-864-'; this prefix is added to the new host name. In the next step the script reads the MAC address assigned to the instance checks the availability of the http server and downloads the mapping file. If the server cannot be accessed then the failsafe mechanism is used the base image consists of a local mapping file. The script reads the mapping file for the correct host name and updates the *sysprep ini file* and launches the utility before exiting. The *sysprep* utility automatically reboots the system and runs a wizard to change the host name and add the instance to the CERN domain.

AUTOMATED CLOUD SETUP

The cloud infrastructure has many components as already mentioned above and configuring these components can take time. Therefore automating configuration was essential a bash script was developed to handle this process. The script has two steps.

CONFIGURING CENTRAL STORAGE

This step is optional depending on the architecture being used. The script creates necessary directories, users and sets correct accessibility rights. It installs *Network File System* service and automatically allows remote accessibility.

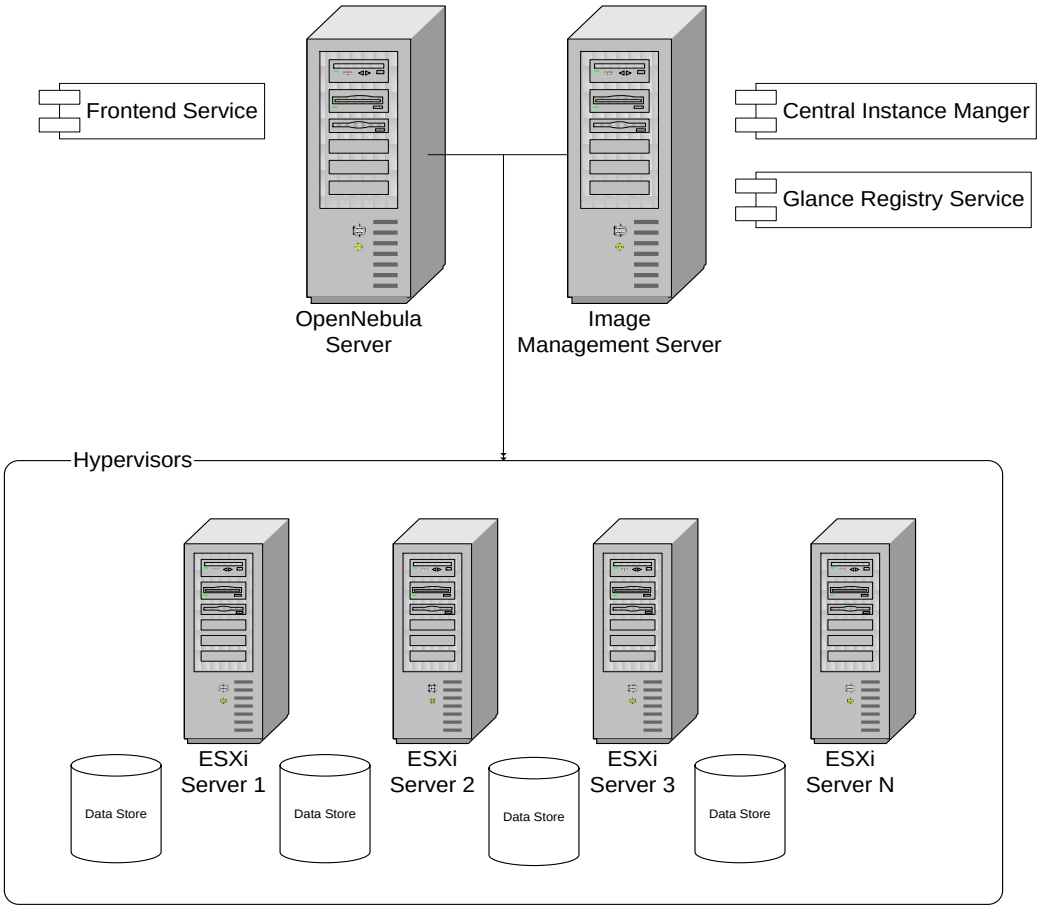
CONFIGURING OPENNEBULA FRONTEND

The setup installs the *OpenNebula* front end server, its dependencies and configuration and settings. This process has to sub steps *Initial Configuration* run as a privileged user and *Installation* run as *OpenNebula* user.

FUTURE ENHANCEMENT

The *Distributed Storage with Instance Management Agent Architecture* can further be improved the instance management agent has to be installed and managed on each *ESXi* server. This is achievable on a small scale cloud setup but does not scale when the cloud consists of hundreds of ESXi servers.

In the enhanced architecture individual agents are no longer required; they are replaced by an instance manager which handles all the deployment of image backups on every *ESXi* server. The enhancement in the architecture allows the design to become scalable.



Currently the Glance Image Repository service and OpenNebula image repository mechanism are completely decoupled and requires manual synchronization. In future the Glance registry can be integrated with OpenNebula.

ACHIEVEMENTS

The project has gained a lot from using open source software components and it was also necessary to give back to the community. During the course of the is project two errors were found in the current version of Opennebula VMWare Addon, which were fixed the Opennebula community are grateful of this contribution and have agreed to give CERN discount on the enterprise version OpennebulaPro.

Another significant achievement was automating instance level contextualization without which the project could not be deemed a success; this mechanism too can be made open to the open source community as this is an issue that many may be facing.

An alternative goal was to conduct scientific study on the usage and architectural design of private clouds. The data that has been collected is being utilized to write a research paper to be submitted in a conference.

This project not only utilized Opennebula as a cloud management tool but also created new components that overcame the problem of using distributed storage in open nebula. This new component can be used and integrated with other cloud management tools such as Open Stack and Eucalyptus.

DOCUMENTATION

Throughout the project important emphasis was given to documenting details of implementation, configuration guides and how problems were solved. The documents have been archived on an online repository for future reference.

The repository also contains each and every artifact produced in this project such

- Open Nebula configuration scripts
- Test document and observations
- Compiled graphs
- Contextualization Scripts
- Creating Windows XP base image
- Open Nebula configuration process
- Opennebula Operation Documentation

These artifacts will help anyone continuing to manage, enhance and studying the project.

CONCLUSION

Although two months has been a very short time, but this project and opportunity has enabled the exploration of many different aspects of cloud computing. Configuration of the system was easy part but designing the cloud according to the pre existing physical infrastructure required thorough evaluation of each architecture with the end result being a much optimized solution for small scale cloud and rapid deployment of virtual machines. Throughout the two months over five hundred virtual machines were deployed. The data gathered from these experiments has established the basis of research paper that will be published soon in upcoming conferences.