|  |
| --- |
|  |
| Dynamically Switch Provision Clusters on Academic Cloud |
| School of Informatics and Computing  Indiana University, Bloomington  Spring 2012 |
|  |
| **B534 Course Project**  **Group 14** |
|  |

Group Members

Kailash Nagarkoti

Ramesh Prasad

|  |
| --- |
|  |

Table of Contents

[Review History 1](#_Toc341981393)

[Problem Statement 2](#_Toc341981394)

[Introduction 2](#_Toc341981395)

[Requirements 2](#_Toc341981396)

[System Architecture 2](#_Toc341981397)

[Dynamic Provisioning System 2](#_Toc341981398)

[Workflow – Dynamic Provisioning System 3](#_Toc341981399)

[Resource Monitoring System 3](#_Toc341981400)

[Message Broker 3](#_Toc341981401)

[Monitoring Daemon 4](#_Toc341981402)

[Monitoring UI 4](#_Toc341981403)

[Workflow – Resource Monitoring 5](#_Toc341981404)

[Experiment Setup 6](#_Toc341981405)

[Provisioning Bare Metal Mode 6](#_Toc341981406)

[Provisioning Virtual Machine Mode 6](#_Toc341981407)

[System Screenshots 6](#_Toc341981408)

[Resources monitor displaying stats for Baremetal nodes - 7](#_Toc341981409)

[Resource Monitor displaying stats for VM nodes - 9](#_Toc341981410)

[9](#_Toc341981411)

[Acknowledgement 11](#_Toc341981412)

[References 11](#_Toc341981413)

# Review History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date Created** | **Modified By** | **Section Modified** | **Reviewer** |
| Version 1.0 | 25-NOV-2012 | Kailash | Initial Draft | Ramesh |
| Version 1.1 | 28-NOV-2012 | Ramesh | Implementation | Kailash |

# Problem Statement

The goal of this project is to port our previous projects (PageRank program & Resource Monitoring System) on Bare Metal clusters and Virtual Machine Set up on these nodes and we achieve this by using the provided Future Grid Dynamic Provisioning infrastructure to switch between Bare Metal and Virtual Machine Environments. The focus is to understand the low level architectural differences of the distributed environment in these two environments and also observe and study the performance of our MPI PageRank program.

# Introduction

Dynamic Provisioning Environment is a networked computing environment where server computing instances or Virtual Machines is provisioned from a centralized application. Thus it provides the ability and possibility to use on-demand resources in a shared academic Cloud environment; user of this system can simply send a request with specifying their needs to the resource manager to obtain different kinds of computing resources, where the requested computing resources are deployed or instantiated on-the-fly. For example, users of FutureGrid can easily obtain a set of Bare Metal machines from Torque resource manager or boot up a set of Virtual Machines.

In this project we use the Dynamic Provisioning system built by Future grid admin and support team. It helps us to switch between Bare Metal and Virtual Machine compute cluster environments by utilizing XCat, Moab and Torque job scheduler. We implement this by writing job scripts to run our MPI PageRank and Resource Monitoring system programs.

# Requirements

For this project, we need to have a Dynamic Provisioning System to switch between Bare Metal nodes and Virtual Machine instances. We need to use our Resource Monitoring program that monitors the CPU and memory utilization in a distributed environment. The system should support resource monitoring of Bare Metal nodes and VM. Monitoring information needs to be collected and aggregated through the message broker and needs to be summarized to display the overall CPU and memory utilization percentages using graphs. All this needs to be done automatically by writing PBS scripts.

# System Architecture

## Dynamic Provisioning System

The architecture of the provisioning system we use to perform our experiment is as shown in figure 1. This system uses Xcat, Moab and Torque job scheduler to facilitate switching between Bare Metal and Virtual Machine compute cluster environments.

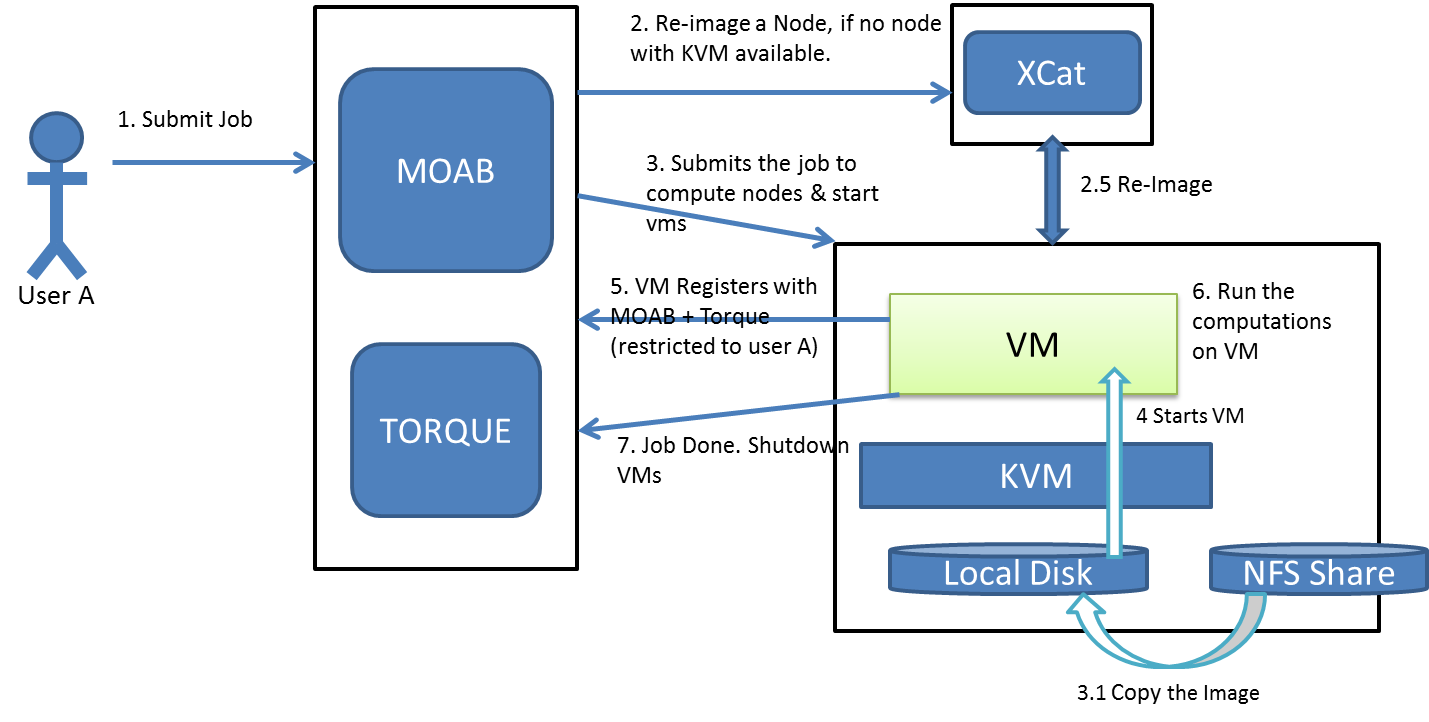


Figure 1: Dynamic Provisioning System

XCat - XCat is an open source distributed computing management and provisioning tool. It allows us to provision Operating Systems on physical and virtual machines. It also allows us to remotely manage systems by providing remote console and distributed shell support.

Moab – It intelligently optimizes workload throughput and increases productivity.

To utilize this whole system we write separate scripts for bare metal nodes and Virtual Machines. We submit it to the PBS manager.

## Workflow – Dynamic Provisioning System

The workflow for the Dynamic Provisioning System is very straightforward after logging into future grid.

We just need to submit our Bare metal or the Virtual Machines scripts to the queue manager. The script contains all the necessary commands to work on desired environment. The output and analysis will be further discussed in the Resource Monitoring workflow.

## Resource Monitoring System

Our monitoring system consists of three major components: a Message Broker, Monitoring Daemons which run on the compute nodes in the distributed network and a client side Monitoring The GUI retrieves and displays the resource utilization statistics.

### Message Broker

A Message Broker is the middleware which performs message storage, message validation, message transformation and message routing between the producer application and client consumer application. It mediates communication amongst applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, effectively implementing [decoupling](http://en.wikipedia.org/wiki/Decoupling#Software_Development). Examples of message brokers include the NaradaBrokering [4] and activeMQ [5]. In our project, we have used the activeMQ message broker service.

### Monitoring Daemon

Monitoring daemon is a background process which runs on each compute node in the distributed system which captures and publishes the system resource utilization information to the Message Broker. In our case we capture only the CPU and Memory utilization. Our Monitoring daemon is responsible for a Producer is an typical publish/subscribe MVC approach for message passing.

### Monitoring UI

Monitoring UI is a client side application which connects to the Message Broker and listens to the messages on the specific topic which are published by the Monitoring Daemon. The Monitoring UI client summarizes the gathered CPU and Memory utilization from all compute nodes and represents them in a graphical pattern on the client UI when user selects the summarized output. The monitoring UI creates a consumer to receive the messages from the broker.

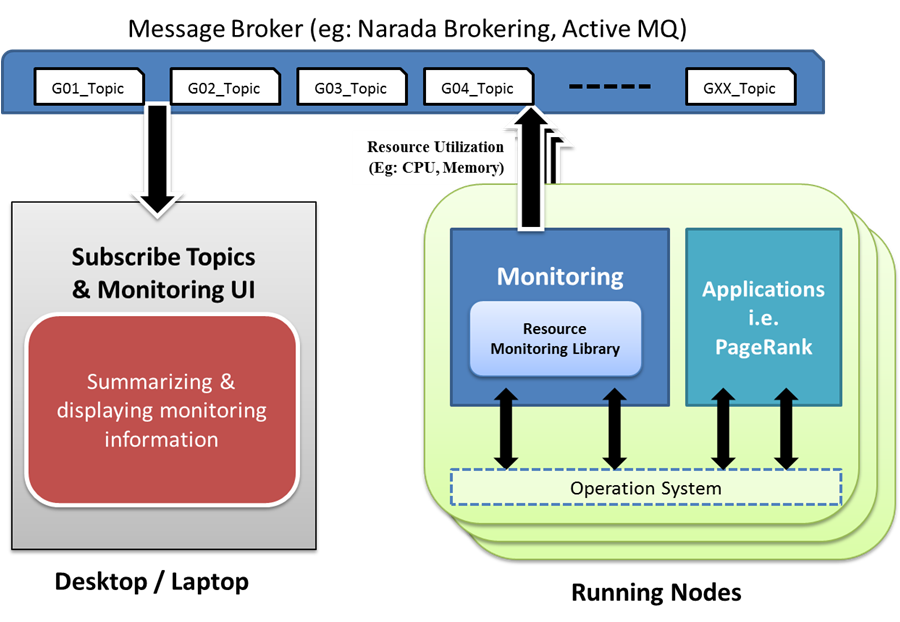


Figure 1. Architectural overview [6]

## Workflow – Resource Monitoring

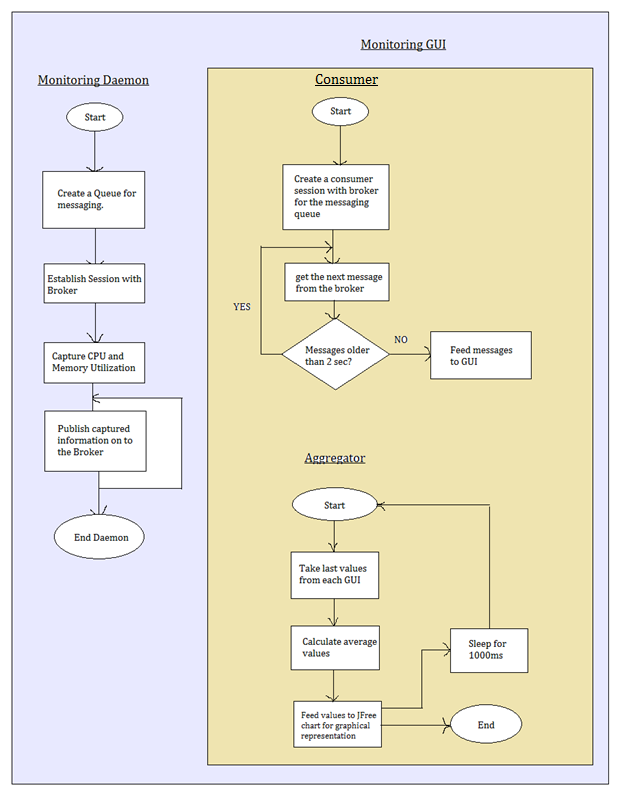


Figure 2. Workflow diagram

# Experiment Setup

To implement this project we have written two scripts to run the program on Bare Metal nodes and Virtual Machines. By using Dynamic Provisioning system we are able to switch between these two environments. We specify the nodes to run the program and number of processors to use. Also we use variety of PBS options to manage our programs such as error logs, email notifications and shutting down our programs.

## Provisioning Bare Metal Mode

In the bare metal scripts we specify the specific queue allotted to perform our experiments and specify the nodes and commands for error logging and for generating output file.

<<< JUST FEW LINES FROM SCRIPTS with short description – may be copy paste from the script..>>>

## Provisioning Virtual Machine Mode

To obtain a VM cluster, we write job script to start the virtual machines in the reserved nodes. The detailed steps are as below:

1. Stage and start the VMs - /user/local/bin/start\_vms
2. Wait till the VMs become reachable - /user/local/bin/start\_vms
3. End job - /user/local/bin/start\_vms

# System Screenshots

These screen shots show performance of our MPI page rank program on Future grid - Bare metal nodes and Virtual machines instances.

Replace with new screen shots for BM and VM

Discussion

<<< project 4 gui results >>>

## Resources monitor displaying stats for Baremetal nodes -

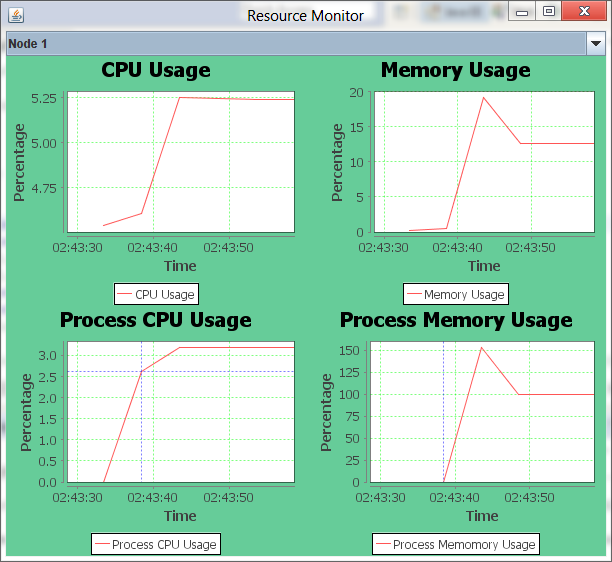


Figure 2: screenshot depicting start of mpi process

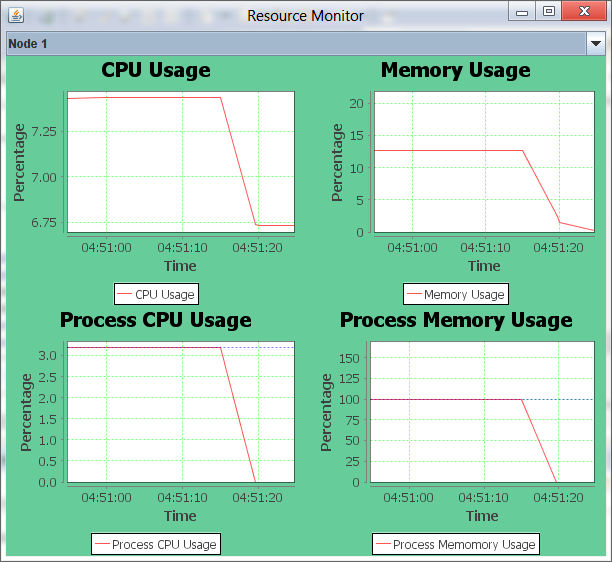


Figure 3: screenshot depicting end of mpi process

## Resource Monitor displaying stats for VM nodes -

# 

Figure 4: screenshot depicting start of mpi process.

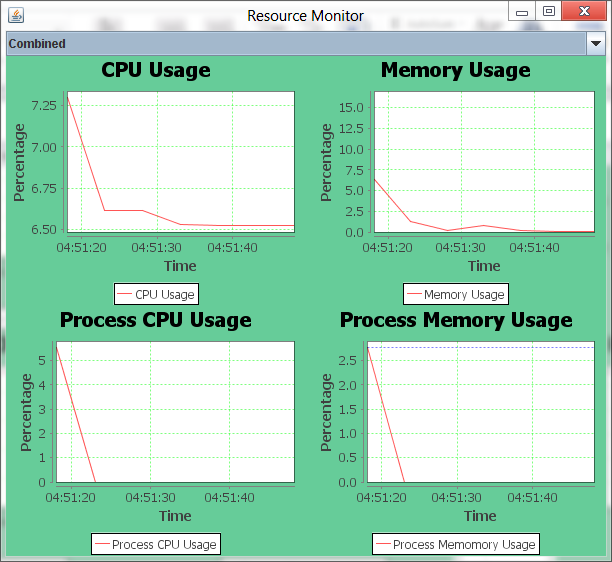


Figure 5: screenshot depicting end of mpi process.

# Acknowledgement

As the group members of this project we would sincerely like to thank Prof. Judy Qui, Associate Instructors Stephen Wu and Ila Jogaikar, for guiding us throughout the course of the project.

# References

1. Ganglia - <http://ganglia.sourceforge.net/>
2. Nagios - <http://www.nagios.org/>
3. Amazon Cloudwatch - <http://aws.amazon.com/cloudwatch/>
4. Narada Brokering - <http://fr.wikipedia.org/wiki/NaradaBrokering>
5. JMS - <http://en.wikipedia.org/wiki/Java_Message_Service>
6. Sigar Resource monitoring API - <http://www.hyperic.com/products/sigar>
7. <http://sourceforge.net/projects/sigar/>
8. JFreeChart, <http://www.jfree.org/jfreechart/>
9. ActiveMQ - <http://en.wikipedia.org/wiki/Apache_ActiveMQ>
10. ActiveMQ, <http://activemq.apache.org/>
11. <http://www.opsview.com/de/node/138>
12. Project #3 Resource Monitoring System Document
13. Futuregrid <https://portal.futuregrid.org/>
14. PBS Script <http://rcc.its.psu.edu/user_guides/system_utilities/pbs/>
15. <http://portal.futuregrid.org>
16. <https://portal.futuregrid.org/tutorials/euca-hadoop>
17. <http://mpj-express.org/docs/guides/linuxguide.pdf>
18. <http://en.wikipedia.org/wiki/PageRank>
19. Qsub: <http://www.clusterresources.com/torquedocs21/commands/qsub.shtml#I>