San José State
UNIVERSITY

Fall 2013

CMPE 283
Virtualization Technology – Section 01

LARGE SCALE STATISTICS & ANALYSIS

Submitted By:
Team 09
Manjunath Shivanna (008623368)
Ameya Patil (009293804)
Shriyansh Jain (009261590)
Harishwar Terupalli (008941777)
Rohan Pednekar (009317243)

Submitted To:
Professor Simon Shim
Computer Engineering Department
San Jose State University

Contents

1. Introduction

Virtualization is a increasingly efficient use of system resources which simplifies IT infrastructure with huge cost savings. It provides features such as scalability, automatic load balancing, portability, and increased efficiency of application deployment. Using virtualization technology, organizations deploy software's across enterprise without any conflicts, system changes, or any impact on stability or security.

Our project is to develop large-scale statistics gathering and analysis in scalable virtualized environments.

Collecting logs from large number of available virtual machines and analyzing them is a challenging task. The challenges lie in analysis of logs of individual as well as multi-tier virtual machines, using various tools to solve this problem and provide meaningful information about the performance to the users.

1.1 Goals

- Experiment with virtualized environment by managing, monitoring and testing virtual machines.
- Understanding need of gathering large-scale data from virtual machines.
- Analyzing collected information from host system, VMs, resource pool, compute resource and clusters.
- Represent this data in a graphical format using High Chart visualization tool.

1.2 Objectives

- Monitoring vHost and Virtual Machine performance.
- Gathering logs from virtual machines.
- Analyzing collected information from host system, VMs, resource pool, compute resource and clusters.
- Represent this data in a graphical format using High Charts visualization tool.

2. Background

Virtualization technology is rapidly growing in IT industry and this is becoming backbone of computation is small, medium scale as well as large scale organizations. Measuring performance and providing alternatives is a challenge of virtualization technology.

Prior to virtualization, only one operating system, application and the required hardware resources were provided to the users to access physical machines. If there were any issues related to performance or storage then customers or end users could use traditional tools to troubleshoot the issue. If they could not find it, IT could provide them with additional storage devices, network infrastructure, etc. This could temporarily solve the problem. But there was no permanent remedy for this issue.

Hence, there is a need to build a framework for large scale metrics data collection, analysis and display collected and analyzed information.

3. Requirements

3.1 Functional Requirements

- Bash Script for large-scale data (logs) collection form the virtual machines.
- Bash Script must collect logs, which will identify workloads from hosts.
- Agent process sitting at the logstash to collect and send monitoring data to the MongoDB Database.
- Java Module to process logs from MongoDB Database to MySQL Database.
- Processed data will be visualized by the analysis module.
- Visualization tool will be used to provide meaningful data representation to the users.

3.2 Non-Functional Requirements

- The System should be scalable for future requirements
- System should handle heavy workloads.

4. Data Source

4.1 Logs

Component: Bash Script
Location: /home/collector.sh
Purpose: Records System logs to a metric.log file

Component: metic.log
Location: /home/metric.log
Purpose: Records the system logs generated from the bash script

4.2 Performance Manager

We have used Logstash for storing the below mentioned metrics from virtual machines into MongoDB Database, using the bash script.

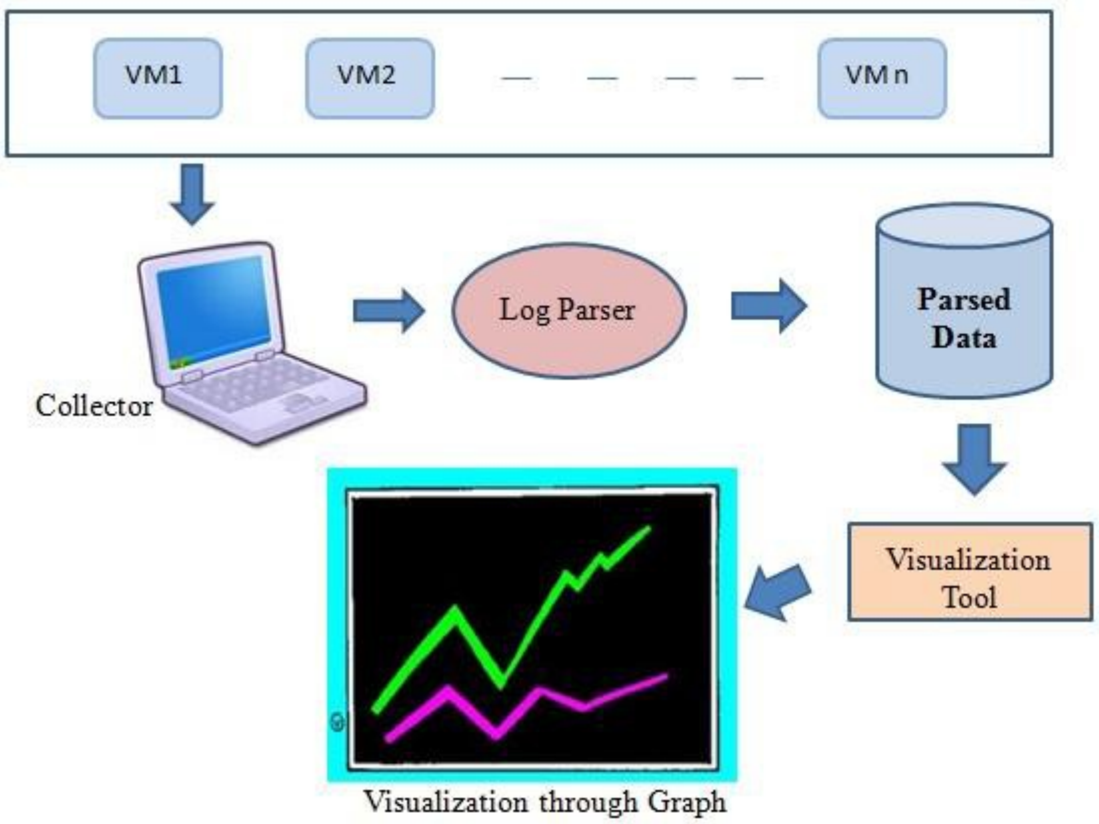
1. CPU Usage
2. Memory Usage
3. Disk Usage
4. Network Usage
5. Threads Running

5. Design

5.1 Architecture

Following components are involved in the logs collection and analysis tool :

1. Collector
2. Agent
3. Analysis Process
4. Visualization



Collector:
It collects system logs such as CPU usage, memory etc. from the virtual machines. It comprises of a bash script, which generated logs at regular interval. The logs generated needs to be stored to a database for further processing. We have used logstash tool for storing the logs to a Database.

Agent:
It is a logstash tool which takes the logs collected by the collector, converts the logs to a compatible format (JSON) and stored the data in a database. We have Used MongoDB Database for this purpose. Logstash pushes the logs to a database every 5 seconds it gets generated. It greatly increases the system performance by automating the logs storing process to a database.

Analysis:
It takes the data from the MongoDB Database and stores it a MySQL database for Visualization purpose. Data gets pushed to a MySQL Database every 5 minutes and every hour. We have written a Java Program to perform the data pushing task from NoSQL to MySQL Database.

Visualization Tool:
The data parsed to a MYSQL database can be visualized using analysis tool such as HighCharts, Google Charts. We are using HighCharts for visualization purpose, which gets the data in real time from the MYSQL database and displays the analysis. We have used JSP (Java Servlets Pages) for getting the data in real time over the server.

6. Workflow

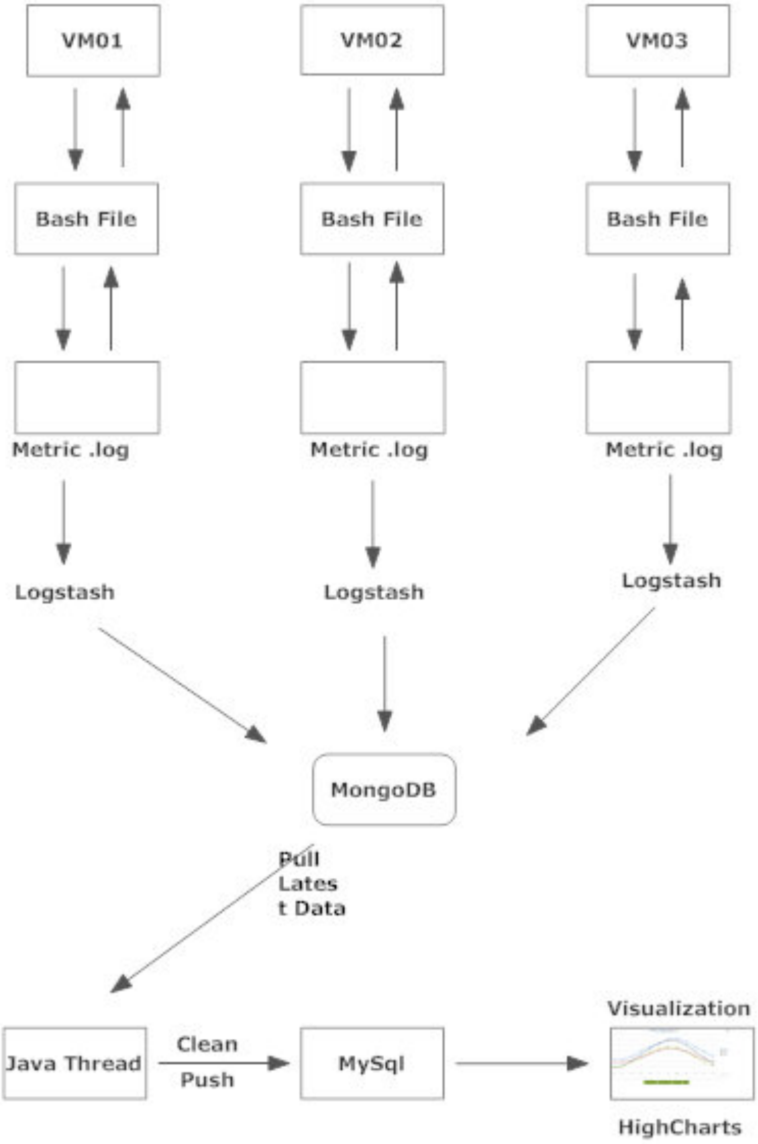
We are using Bash Script named collector.sh to gather the performance measurement metrics such as CPU usage, memory etc. which need to be running on a virtual machine.

Data Collection and Storage:

- The data being generated from the bash script gets stored to a metic.log file which gets stored to a MongoDB database every 5 seconds through logstash.
- Logstash converts the data gathered in a metric.log file to a JSON format and stored the data a NOSQL Database.
- Logstash automatically stores the data generated in a metric.log file through bash script, to a NOSQL database.
- Then, data from NOSQL database gets stored to SQL database every 5 minutes through a java program.

Visualization

- We are using HighCharts Visualization tool, JSP (Java Server Pages) for visualization.
- Data residing on MySQL is queried every 5mins, one hour and 24 hours to get data required to display in the form of graphs.
- Data is transferred from MySQL using Ajax call with the help of JSP.


VM01
VM02
VM03
Bash File
Bash File
Bash File
Metric .log
Metric .log
Metric .log
Logstash
Logstash
Logstash
MongoDB
Pull
Lates
t Data
Java Thread
Clean
Push
MySql
Visualization
HighCharts

7. Implementation

7.1 Environment

Cumulus 2 at 130.65.132.214 has three hosts:
Host 1 – 130.65.132.215
Host 2 – 130.65.132.216
Host 3 – 130.65.132.217
We are monitoring these vHosts which are clustered together and managed by vCluster_Team09

7.2 Tools and Technologies

- Tomcat Web Server 7.0.12
- VMWare vSphere Client
- MongoDB, MySQL
- High Charts, jQuery, Ajax
- JSP

VMware vSphere Client is used to connect to the datacenter, access the hosts from it and also the available virtual machines on these hosts.
Bash Script is used to generate system logs.
Eclipse is used for code management.
Operating System used is Ubuntu.

7.2.1 Track 1- Application Performance Management

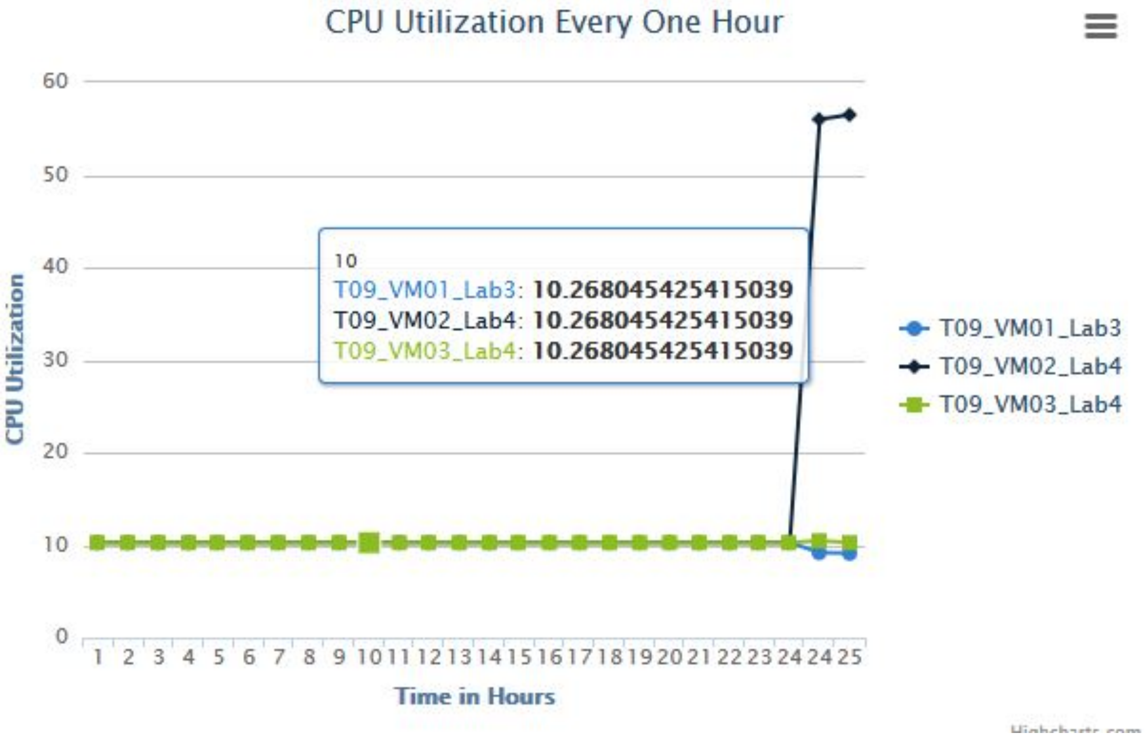
It is application to monitor performance and availability of software systems.
It is measured by evaluating performance of computational resources and measuring the performance as seen by the end users.

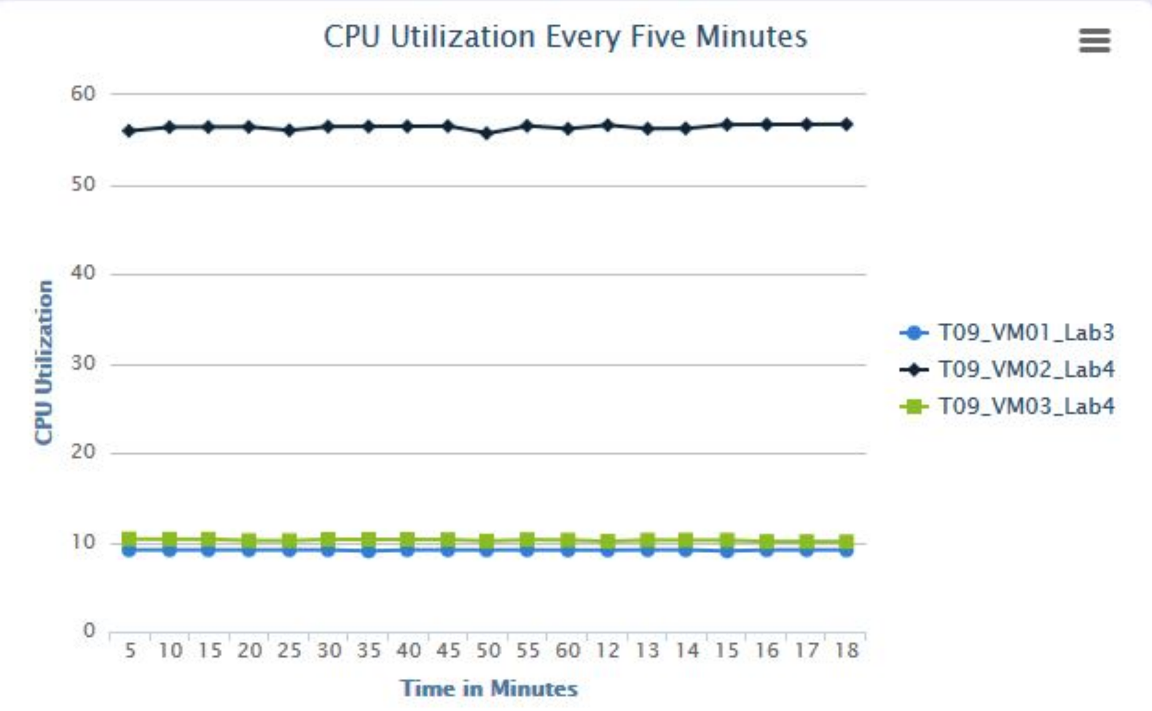
Framework:

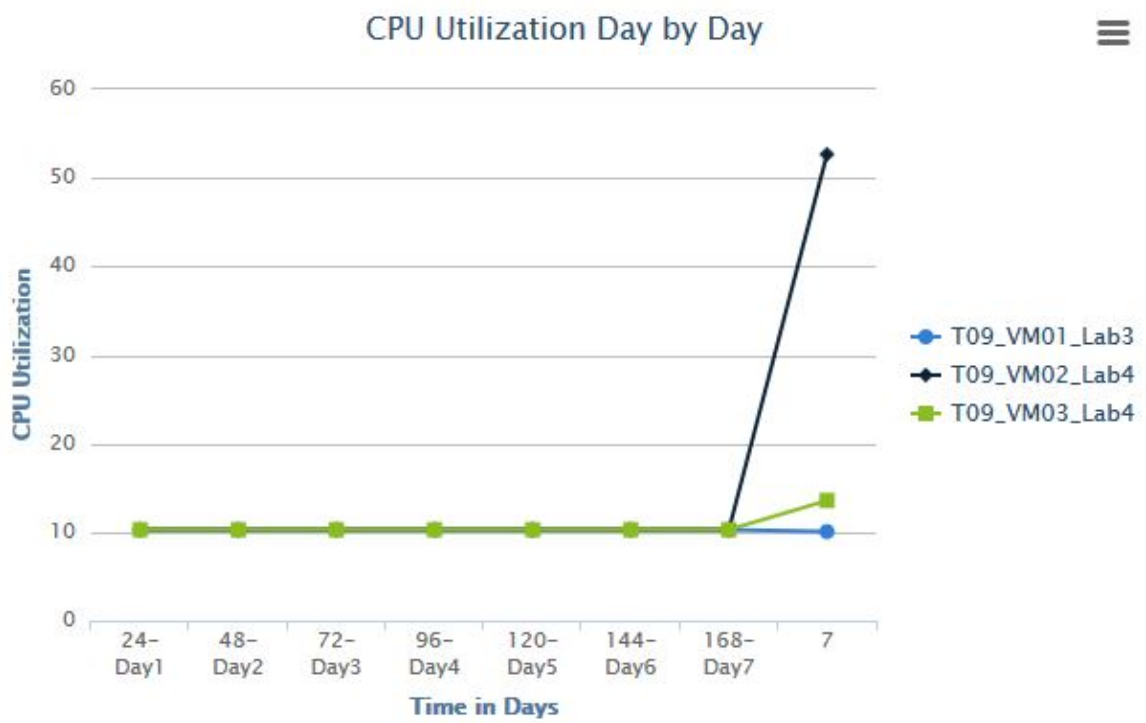
Five components

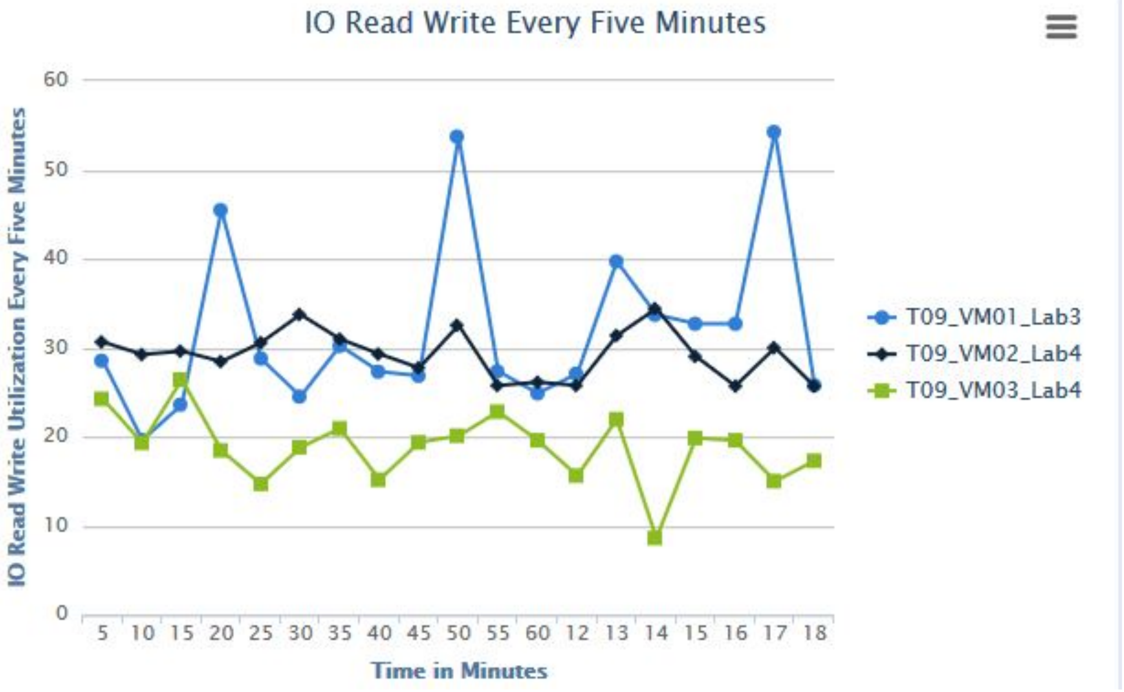
- End User Experience,
- Runtime Application Architecture,
- Business Transaction,
- Deep dive Component Monitoring,
- Analytics/Reporting

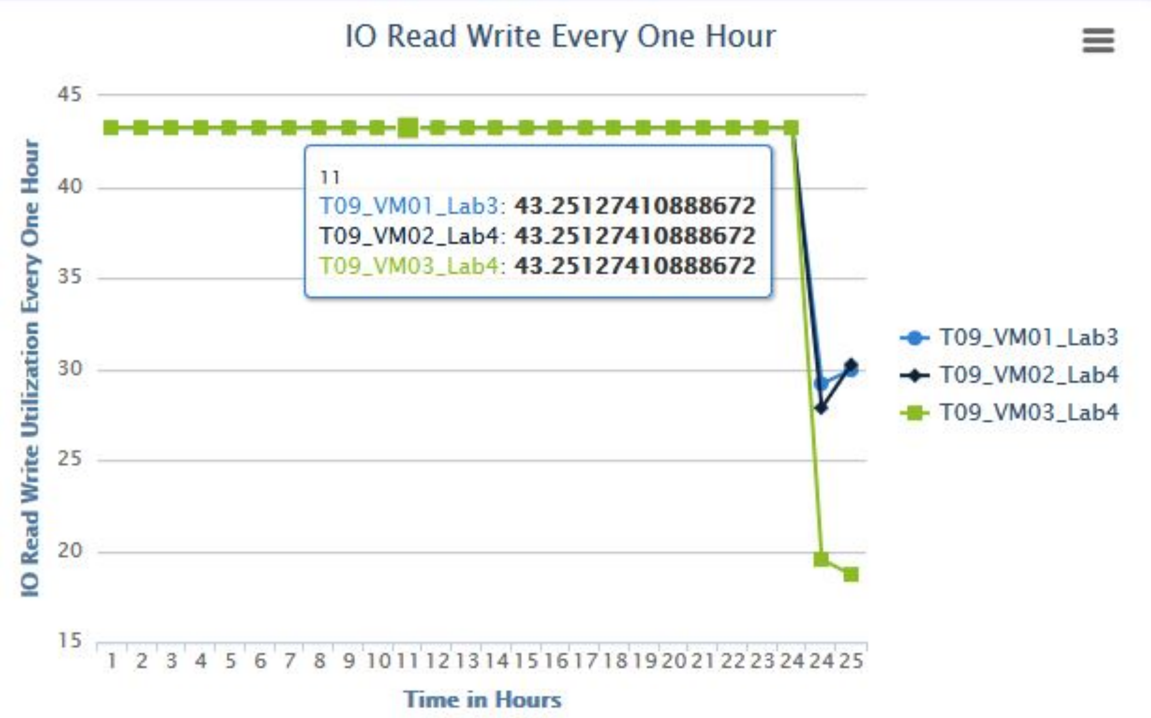
7.3 Screenshots

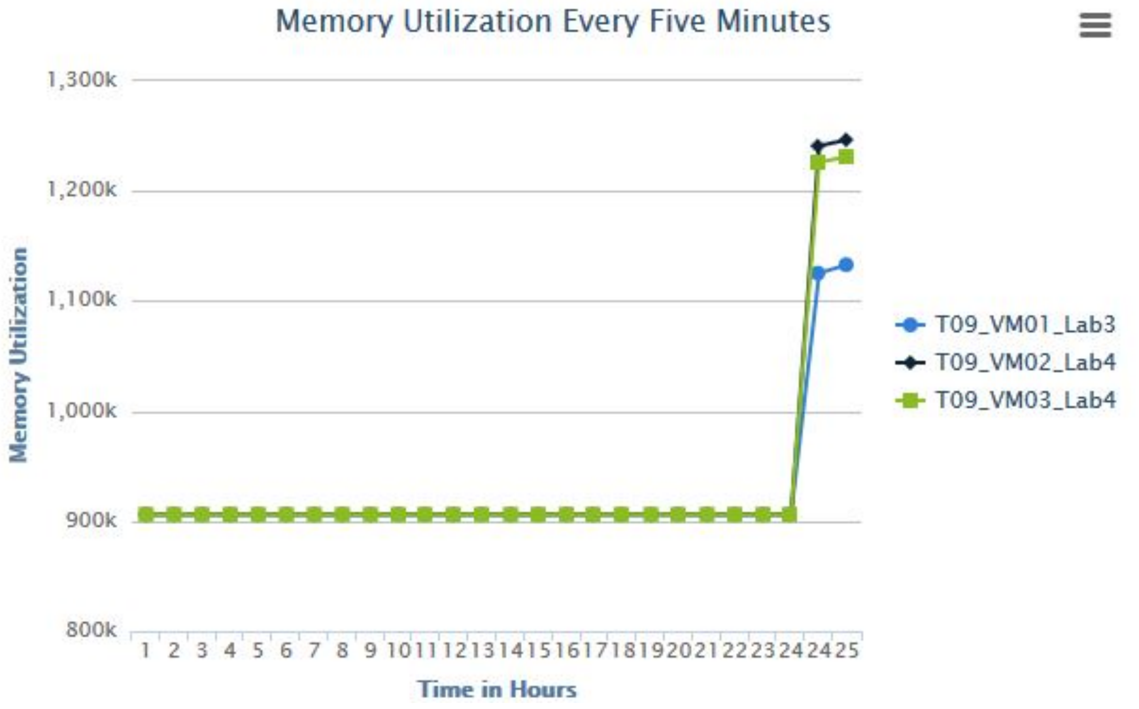


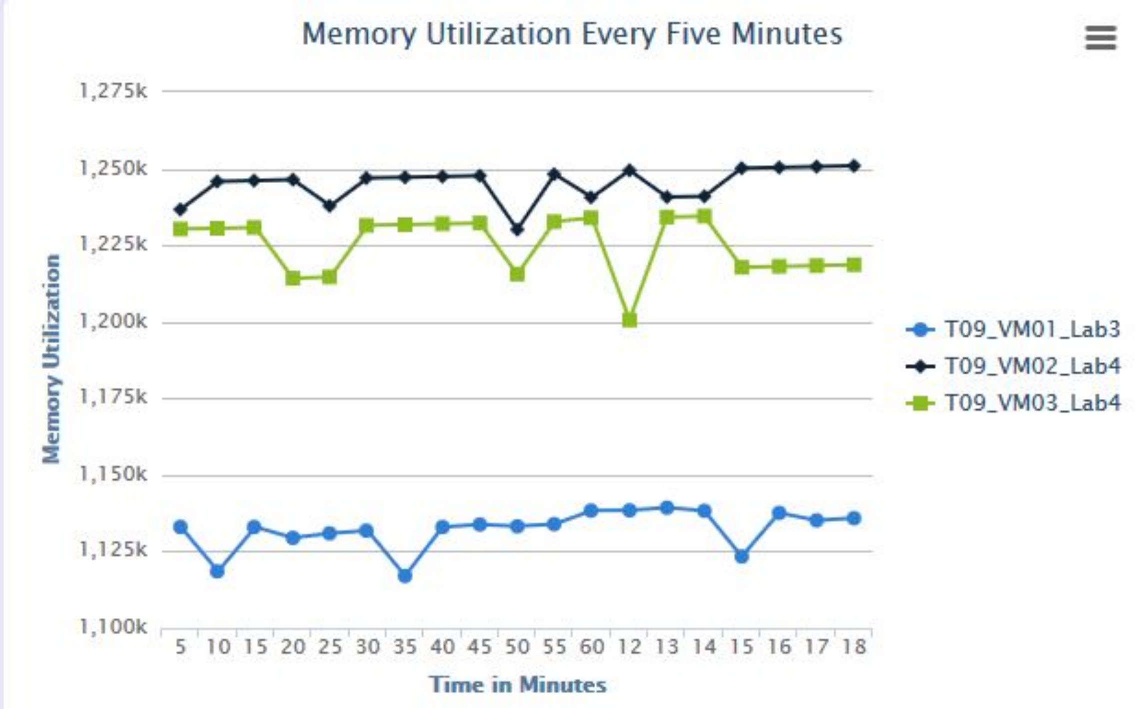


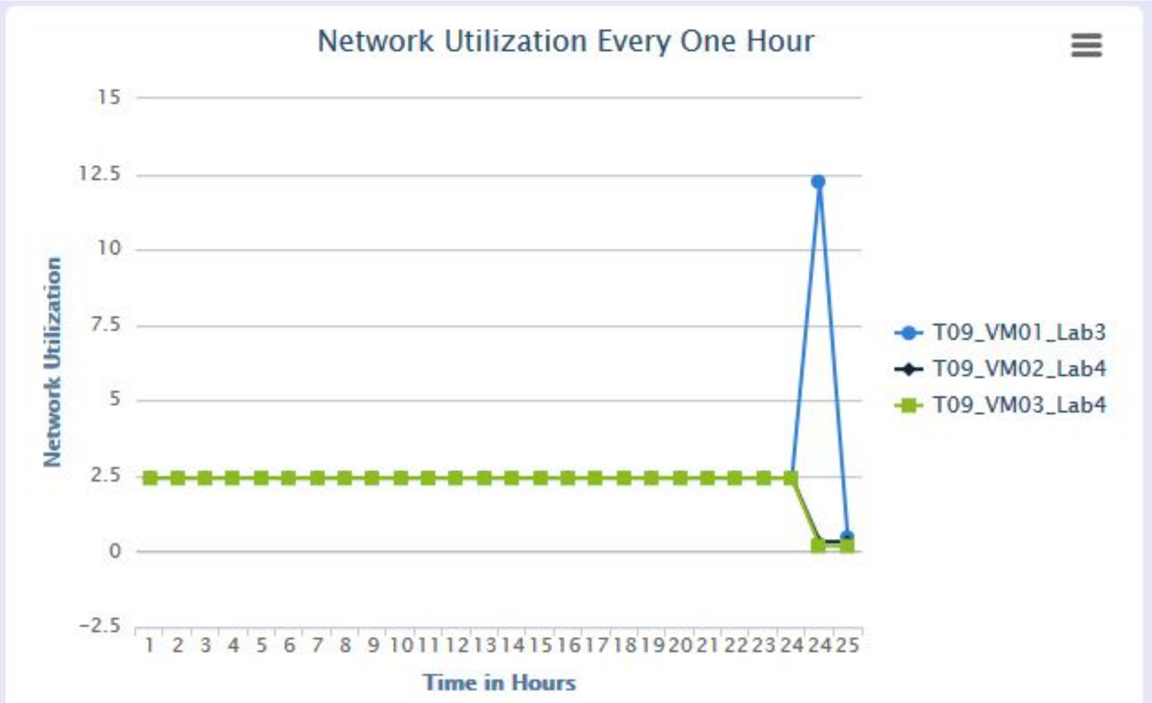

CPU Utilization Day by Day
CPU Utilization
60
50
40
30
20
10
0
24-Day1
48-Day2
72-Day3
96-Day4
120-Day5
144-Day6
168-Day7
7
Time in Days
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4


IO Read Write Every Five Minutes
IO Read Write Utilization Every Five Minutes
60
50
40
30
20
10
0
5
10
15
20
25
30
35
40
45
50
55
60
12
13
14
15
16
17
18
Time in Minutes
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4


IO Read Write Every One Hour
IO Read Write Utilization Every One Hour
45
40
35
30
25
20
15
11
T09_VM01_Lab3: 43.25127410888672
T09_VM02_Lab4: 43.25127410888672
T09_VM03_Lab4: 43.25127410888672
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 24 25
Time in Hours
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4


Memory Utilization Every Five Minutes
Memory Utilization
1,300k
1,200k
1,100k
1,000k
900k
800k
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 24 25
Time in Hours
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4


Memory Utilization Every Five Minutes
1,275k
1,250k
1,225k
1,200k
1,175k
1,150k
1,125k
1,100k
Memory Utilization
5 10 15 20 25 30 35 40 45 50 55 60 12 13 14 15 16 17 18
Time in Minutes
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4


Network Utilization Every One Hour
15
12.5
10
7.5
5
2.5
0
-2.5
Network Utilization
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 24 25
Time in Hours
T09_VM01_Lab3
T09_VM02_Lab4
T09_VM03_Lab4

8. Assumptions

Java Based Environment in Eclipse.
NoSQL and MySQL.
Requires 1 or 2 virtual machines to be installed in Data Center.
High Charts is a HTML5 visualization tool, Hence a browser is required.

9. Limitations

It is not centralized application. There is need to configure this application on every Virtual Machine.

10. Individual Contribution

Team Member Name	Contribution
Manjunath Shivanna	Project Planning DB Design Data processing from MySQL using JSP UI development using HighCharts
Shriyansh Jain	Project Planning DB Design Data pooling using log stash Data transfer from logstash to MongoDB
Ameya Patil	Project Planning DB Design Testing Documentation
Harishwar Terupalli	Project Planning DB Design Testing Data transfer from NoSQL to MySQL
Rohan Pednekar	Project Planning DB Design Statistics Collection from Virtual Machine Documentation

11. Future Work

Application can be developed on different environment.
We can add more number of performance and metrics and we can implement more robust parser for these data and analysis will be improved by this.
We can scale the whole application. We have developed this for 3 threads and 3 host. But we can scale it for more number of hosts and for each host using quad core we can maximize number of threads that are collecting data from database.
We can use multiple data store to store collected Performance metrics.

12. Testing

```
public class TestCases {

      @BeforeClass
      public static void setUpBeforeClass() throws Exception {
            System.out.println("Before");
      }

      @Test
      public void BasicTesting(){
            String SERVER_NAME = "130.65.132.214";
            String USER_NAME = "administrator";
            String PASSWORD = "12!@qwQW";
            String HOSTNAME = "130.65.132.216";
            String url = "https://" + SERVER_NAME +
"/sdk/vimService";
            DisplayHostCounter dhc = new DisplayHostCounter();
            dhc.testing(url,USER_NAME,PASSWORD,HOSTNAME);
            Assert.assertTrue(true);
      }

      @Test
      public void BasicTesting1(){
            String SERVER_NAME = "130.65.132.214";
            String USER_NAME = "admininstrator";
            String PASSWORD = "12!@qwQW";
            String HOSTNAME = "130.65.132.216";
            String url = "https://" + SERVER_NAME +
"/sdk/vimService";
            DisplayHostCounter dhc = new DisplayHostCounter();
            dhc.testing(url,USER_NAME,PASSWORD,HOSTNAME);
            Assert.assertFalse(false);
      }

      @Test
      public void BasicTesting2(){
            String SERVER_NAME = "130.65.132.214";
            String USER_NAME = "administrator";
            String PASSWORD = "12!@qwQW";
            String HOSTNAME = "130.65.132.216";
            String url = "https://" + SERVER_NAME +
"/sdk/vimService";
            DisplayHostCounter dhc = new DisplayHostCounter();
            dhc.testing(url,USER_NAME,PASSWORD,HOSTNAME);
            Assert.assertFalse(false);
      }

      @Test
      public void BasicTesting3(){
            String SERVER_NAME = "130.65.132.214";
            String USER_NAME = "administrator";
            String PASSWORD = "12!@qwQW";
            String HOSTNAME = "130.65.132.216";
            String url = "https://" + SERVER_NAME +
"/sdk/vimService";
            DisplayHostCounter dhc = new DisplayHostCounter();
            dhc.testing(url,USER_NAME,PASSWORD,HOSTNAME);
            Assert.assertFalse(false);
      }

      @Test
      public void BasicTesting4(){
            String SERVER_NAME = "130.65.132.214";
            String USER_NAME = "administrator";
            String PASSWORD = "12!@qwQW";
            String HOSTNAME = "130.65.132.216";
            String url = "htt://" + SERVER_NAME +
"/sdk/vimService";
            DisplayHostCounter dhc = new DisplayHostCounter();
            dhc.testing(url,USER_NAME,PASSWORD,HOSTNAME);
            Assert.assertFalse(false);
      }

      @Test
      public void PingTestPass()throws Exception{
            String ipAddress = "130.65.132.214";
            VMInitalizaler init = new VMInitalizaler();
            init.pingVM(ipAddress);
            Assert.assertTrue(true);
      }

      @Test
      public void PingTestFail()throws Exception{
            String ipAddress = "130.65.132.216";
            VMInitalizaler init = new VMInitalizaler();
            init.pingVM(ipAddress);
            Assert.assertFalse(false);
      }

      @Test
      public void CheckHostPass(){
            String hostIP = "https://130.65.132.214/sdk";
            VHostMHealthUpdateThread thread = new
VHostMHealthUpdateThread(hostIP);
            Assert.assertTrue(true);
      }

      @Test
      public void CheckHostFail(){
            String hostIP = "https://130.65.132.214/sdk";
            VHostMHealthUpdateThread thread = new
VHostMHealthUpdateThread(hostIP);
            Assert.assertFalse(false);
      }
```

```
	@AfterClass
	public static void tearDownAfterClass() throws Exception {
		System.out.println("After");
	}


}
```

13. Conclusion

Successfully analyzed large-scale data of CPU, disk, memory, system and network. This system can be used to solve many problems related to virtual machines.

14. References

[1] http://en.wikipedia.org/wiki/Application_performance_management
[2] http://www.vmware.com/pdf/esx_lun_security.pdf
[3] http://www.vmware.com/pdf/esx_lun_security.pdf
[4] http://www.mongodb.org/
[5] http://docs.mongodb.org/ecosystem/tutorial/getting-started-with-java-driver/
[6] http://vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/
[7] http://www.highcharts.com/