**Virtual Machine Metrics Collection and Analysis**

**in Scalable Virtualized Environment**



**CMPE 283-VIRTUALISATION**

**Fall 2013**

**Project 2-Report**

**By**

**(Team 14)**

**Advisor: Dr.SimonShim**

**Jayadev Akkiraju (008732035)**

**Prudhvi Chandra (009268610)**

**Mintu Abraham (008782098)**

**Sourabh Bhosale (008654646)**

**Geetha Anne (008741057)**

**Table of Contents**

Introduction 4

Objective 4

Background 4

Requirement 5

Functional Requirement 5

Non-Functional Requirement 5

Design 6

Implementation & Approach 9

Assumptions&Limitations…………………………………………………………………………………….....18

Security Configuration 18

Installation and Execution manual 25

Testing 27

Future Work 31

Individual Contribution 31

Conclusion 31

References 32

**INTRODUCTION:**

## Objective

The objective of this project is to:

* Experiment with the virtualized environment by managing , monitoring and testing virtual machines
* Collect and analyze the logs from virtual machines and ESXis to a remote client open source tools such as scribe, log4j, logstash etc.
* Understand the need of gathering and analysis for large data.
* Visualize the collected data using the tools such as Vaadin UI, Chart4J, Graph.js, Google visualization tool, graphite etc.

**BACKGROUND:**

System Logs contain error messages, operational data and usage information which can help to manage applications and servers. There are many log analyzer tools such as scribe, graylog2, log4j, logstash etc. which read and parse log files, extract and aggregate information to generate reports on the application. With the content of the logs, use of it varies e.g. Web-server log can be used to study traffic patterns; network log can be used to optimize network performance or detect malicious intrusion; security logs can provide information about breaches, attacks etc. Some of the objectives of log analysis are:

* Compliance with security policies and audit
* System troubleshooting
* Security incident response

This project primarily focuses on the performance logs of virtual machines. The system will collect data such as CPU usage, memory usage from each virtual machine on hourly basis. Utilizing this information, graphs are generated which can help to understand the workload, resource utilization etc.

**REQUIREMENTS:**

## Functional Requirement

* System should be able to collect system data to identify workloads on system elements (jobs, hosts, guests etc.)
* Agents will collect the configured metrics and commit as local record tables on each host, for a timed interval of some seconds.
* Pollers poll the agents and get record tables back via suitable Transport (TCP/ UDP).
* System should be able to save the collected data in noSQL database such as Cassandra or MongoDB.
* After collection is successful, system will archive data records and notify the engines.
* These engines will then analyze and reduce data, and create second set of tables that has analyzed values
* System should be able to generate abstract views of the system processing collected system data
* System should be able to show system data in the form of graph.
* Hourly Analyzed Record Rollup Process will kick in every hour for rolling up data into 24 sets.
* Daily Analyzed Records Rollup. 24 hour Process will kick in for rolling up data from the 24 sets into daily sets.
* Purging and Cleanup are performed by the designed system.

## Non-Functional Requirement

* Ease Of Use: The Proposed system should be easily utilized by software professionals
* Security: System needs to be secure enough.
* Understandability: Any beginner should easily understand the system
* Maintainability: System should be maintainable

**DESIGN ,ARCHITECHTURE and COMPONENTS WORKFLOWS:**

The VSphereVcenter client design consists of cluster, resource pool, vHost and Virtual machines.Below given is the diagram showing the basic design of a typical VCenter.

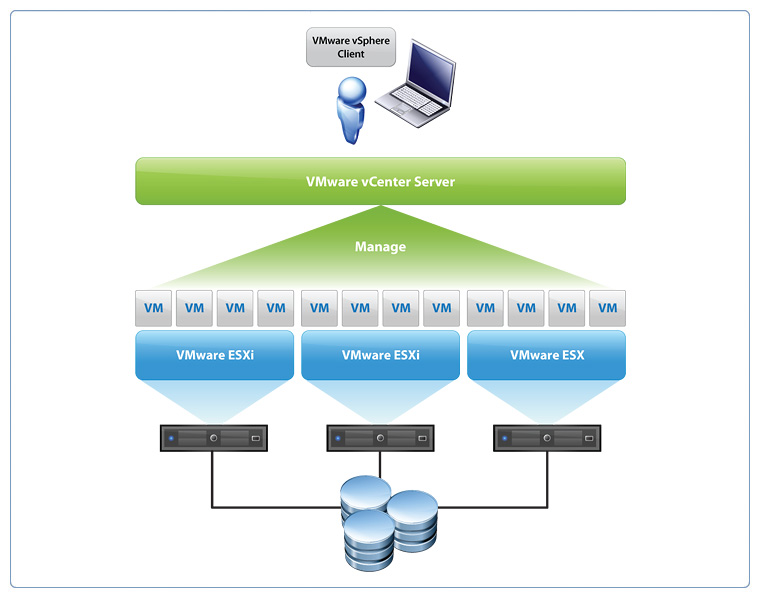


Figure : VCenter VSphere Representation

The System is designed incorporating various tools such as log4j for log parsing and graphite for graphing/reporting. With the help of VI Java APIs, system will fetch data related to VM using log4j .This data is stored in MongoDB database. This data will be used for real time graphing.

**ARCHITECHTURE OF THE SYSTEM:**



Figure: High level system design

The above diagram shows the proposed design of the required system. The Virtual machine metrics like CPU, Memory etc. are collected for a timed interval by a collector tool. Later these system data are parsed according to our requirement and stored in a NOSQL database. After the storage this data is visualized.

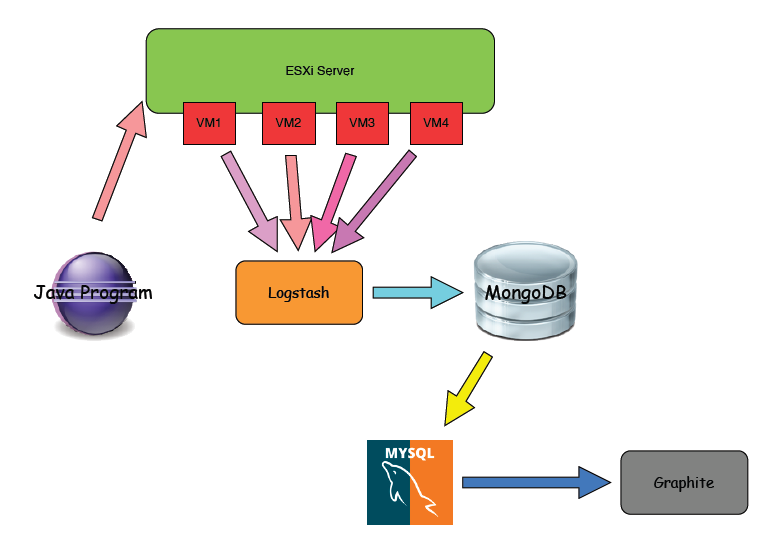
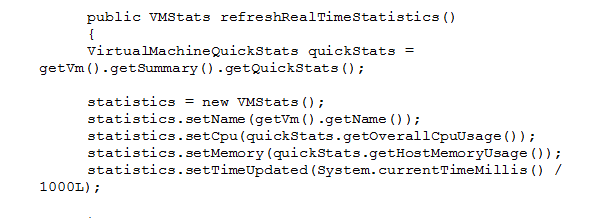
****

Figure: System Architecture

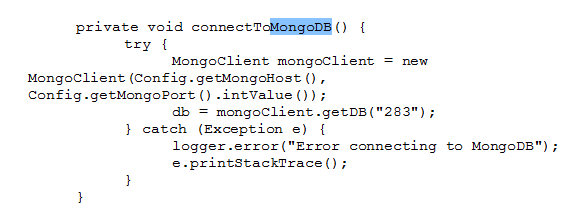
A collection of system data (metrics, logs) to identify workloads on system elements (jobs, hosts, guests etc) as you have done in project 1 and store them in noSQL Database such as Cassandra or Mongo DB is done. Host Monitoring is performed and collection of metrics like Per Guest Level CPU, Memory, Threads, I/O, Locks, Network Statistics etc.Applying higher-level processing (average per 5 mins intervals, etc) on collected system data to generate abstract views of the system.MYSQL was used to store them. The outcome of the above steps is presented and visualized in a simple manner.

**COMPONENT WORKFLOWS:**

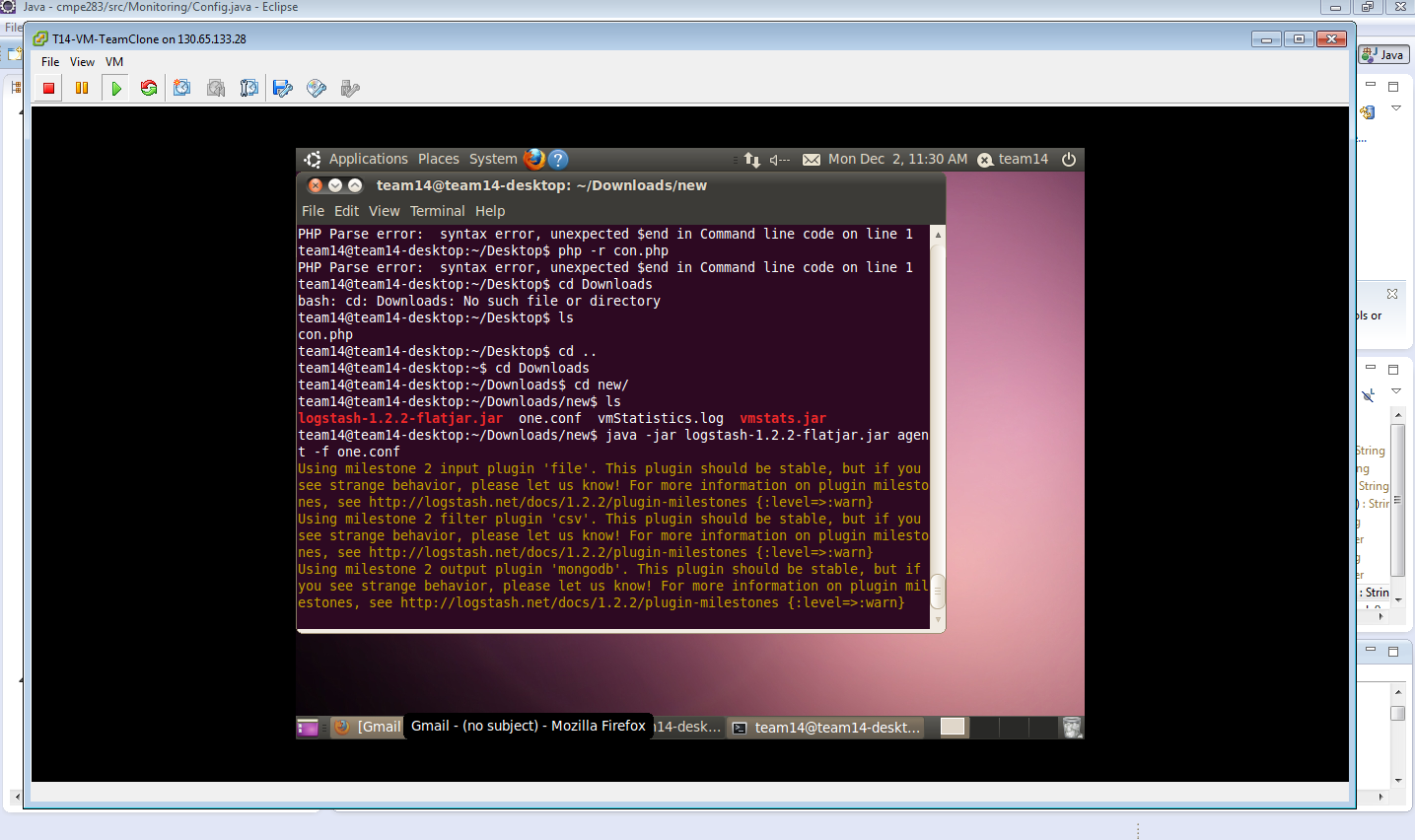
1. First step of the project work flow is to log into the Vsphere Vcenter Client.
2. Maintain atleast one healthy ESXi host and a minimum of three Virtual Machines intialised and powered On. Make sure the VMs are all monitored and healthy.
3. Perform the Security configurations on the ESXi host and also on all the Virtual Machines
4. Install logstash on all the Virtual Machines which helps in gathering of metrics like CPU Usage, Memory, I/O like disk read , disk usage, Network statistics etc.

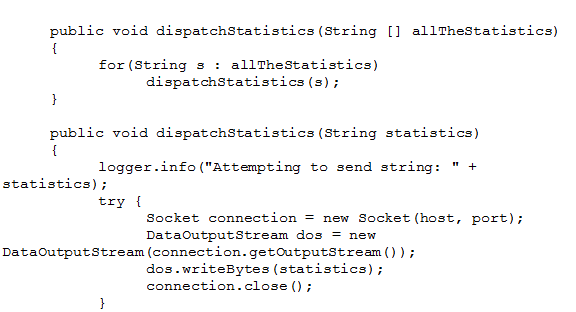


1. Install and start the MongoDB server on one of the Virtual machines which acts as the Collector Process



1. The MongoDb server on this collector Vm is made to be up and running all the time of execution.
2. MYSQL server is also started on the same Virtual Machine and made sure to be up and running.
3. Supervisor module in the run on the webserver and it helps to get the connection established.
4. Logstash module collects the metrics from all virtual machines and automatically pushes them into the MongoDB database.





1. Log4j.tar has one file named log4j.properties which helps in collecting the logs .This is also in cooperated into the workspace of the project.
2. From the MongoDB the logs are pushed and stored into MYSQL
3. Graphite’s carbon tool is installed and made sure the graphite server is running.
4. The parsed data is retrieved from MYSQL and are visualized using Graphite tool.
5. Graphs representing the VM statistics at a certain time interval like hourly, minute basis are shown and visualized.

**AGENT PROCESS :**

**COLLECTOR PROCESS:**

**ANALYSIS MODULE:**

**SUPERVISOR MODULE TO CONTROL/MANAGE AGENTS:**

**IMPLEMENTATION AND APRROACH:**

The system is implemented using VI java APIs, graphite and mongoDB. Eclipse IDE is used with VI Java APIs to provide an interface to VMware virtual infrastructure management framework. All the operations such as collecting logs, showing graphs etc. are performed through VI java SDK without any human interference. Java is used as a scripting language for writing this prototype. . Logstash is an open source tool for managing events and logs.

The mongodb schema used in the system is represented using JSON as follows:-

{

"\_id": ObjectId('518fhjbbdnshywb284ys'),

"name": "T14\_VM\_prudhvi",

"cpu": 47,

"memory": 819,

"diskReadAverage": 32,

"diskWriteAverage": 21,

"diskTotalLantency": 671,

"netUsageAverage": 12,

"datastoreReadAverage": 12,

"datastoreWriteAverage": 27,

"netBytesRxAverage": 532,

"netBytesTxAverage": 321,

"time": 1360172041

}

**Virtual Machines and Resource Pools:**

The code of this project contains methods that determine the available virtual machines in the resource pool. Available VMs and resource pools are defined with the help of getVmsForMonitoring and getAvailableResourcePools to monitor.

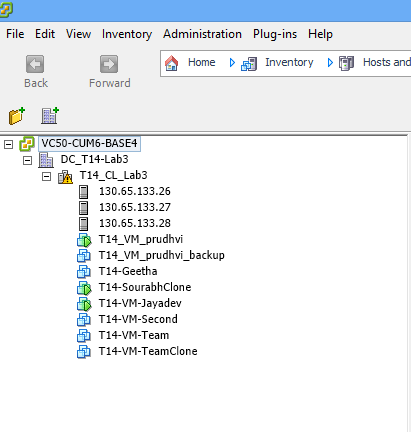


Figure : Available VMs & Resource Pools

**Statistics:**

The statistics such as CPU, Memory, Disk Read & write etc. are collected the following methods:

timer**.**schedule**(new** RealTimeStatisticsTask**()** **,** 0**,** 10 **\*** 1000**);**

timer**.**schedule**(new** FifteenMinuteStatisticsAggregationTask**()** **,** 0**,** 15 **\*** 60 **\*** 1000**);**

timer**.**schedule**(new** HourlyStatisticsAggregationTask**(),** 0**,** 60 **\*** 60 **\*** 1000**);**

timer**.**schedule**(new** DailyStatisticsAggregationTask**(),** 0**,** 24 **\*** 60 **\*** 60 **\*** 1000**);**

These performance statistics are collected on fifteen minutes, hourly, daily and real time basis. For example, the aggregated statistics are collected with the help of following **getAllAggregatedStatistics** method:

**public** String**[]** **getAllAggregatedStatistics(**DB db**,** Long delta**,** String label**)** **{**

String **[]** allFields **=** **{**"cpu"**,**

"memory"**,**

"diskUsageAverage"**,**

“diskReadAverage"**,**

"diskWriteAverage"**,**

"netUsageAverage"**,**

"netBytesRxAverage"**,**

"netBytesTxAverage"**};**

List**<**String**>** allMessages **=** **new** ArrayList**<**String**>();**

**for(**String field **:** allFields**){**

Iterable**<**DBObject**>** results **=** aggregateStatistics**(**db**,** field**,** delta**);**

**for(**DBObject result **:** results**){**

logger**.**info**(**field **+** ": (average: " **+** result**.**get**(**"average"**)** **+** ")"**);**

allMessages**.**add**(**"283.aggregated." **+** getVm**().**getName**()** **+** "." **+** label **+** "." **+** field **+** " " **+** result**.**get**(**"average"**)** **+** " " **+** System**.**currentTimeMillis**()** **/** 1000L **+** "\n"**);**

**}**

**}**

String **[]** allStatistics **=** **new** String**[**allMessages**.**size**()];**

**return** allMessages**.**toArray**(**allStatistics**);**

**}**

Similarly, real time statstics are collected with the following **getAllRealTimeStatistics** method:

**public** String**[]** **getAllRealTimeStatistics()** **{**

String **[]** allStats **=** **{**

statistics**.**realTimeCpuStats**(),**

statistics**.**realTimeMemoryStats**(),**

statistics**.**realTimeDiskUsageAverageStats**(),**

statistics**.**realTimeDiskReadAverageStats**(),**

statistics**.**realTimeDiskWriteAverageStats**(),**

statistics**.**realTimeNetUsageAverageStats**(),**

statistics**.**realTimeNetBytesRxAverageStats**(),**

statistics**.**realTimeNetBytesTxAverageStats**()};**

**return** allStats**;**

**}**

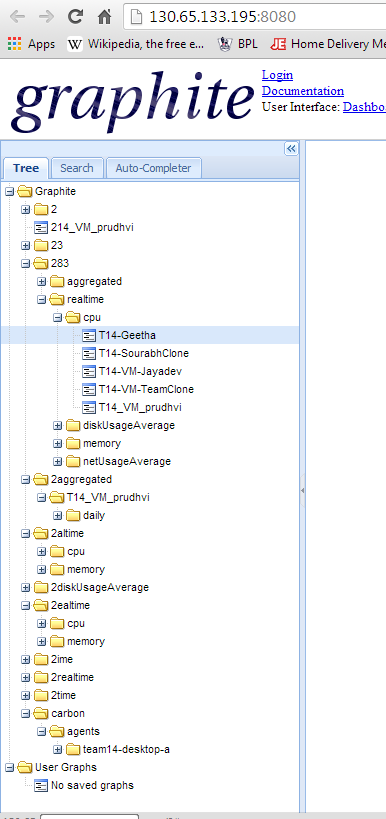


Figure : Statistics in the Graphite display

**Storing and Saving the obtained Statistics:**

These statistics are then saved into the database. The code snippet for it is:

**public** **boolean** **saveStatistics(**DB db**)**

**{**

DBCollection collection **=** db**.**getCollection**(**getVm**().**getName**());**

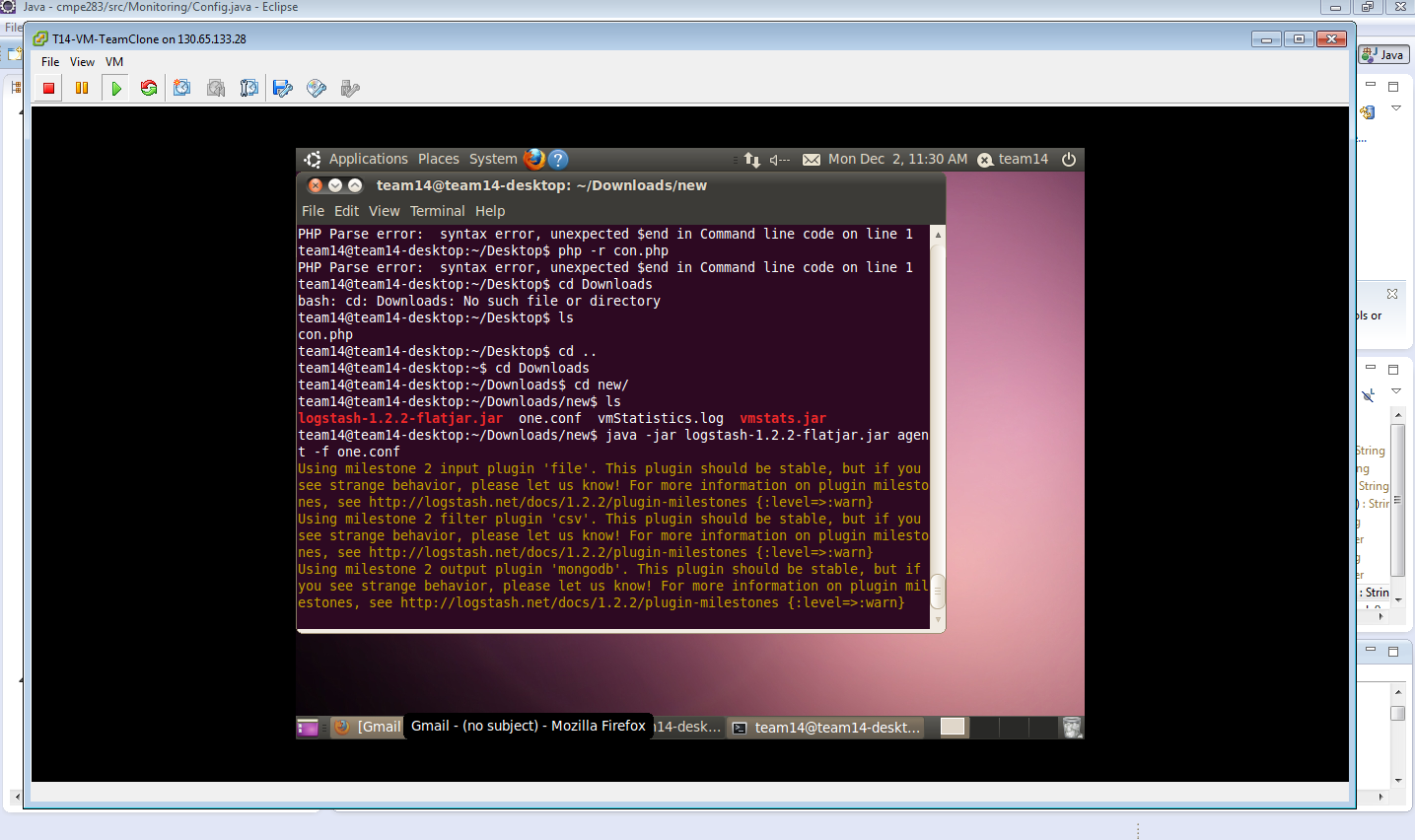
collection**.**insert**(**statistics**.**getMongoDoc**());**

**return** **true;**

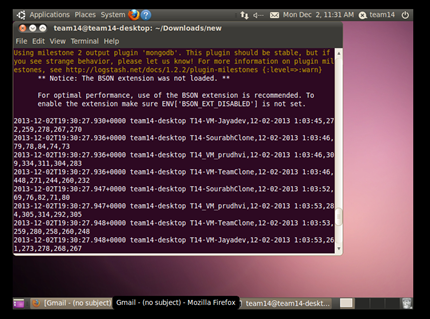
**}**

**Logstash Agent** will collect and parse the logs. Logstash will collect and ship logs for theVMs as well as from the VMware ESXi Host.

**Screenshot 1: Log stash showing the running logs from all the VMs**

****

**Screenshot 2: Logstash showing the generated logs**

****

**Graphs:**

The real time and aggregated graphs are plotted with the help of logs using ‘**Graphite’.** The real time CPU usage of all the three VMs is as:



Figure : Real time CPU usage of the VM

The real time memory usage of the VMs is as:

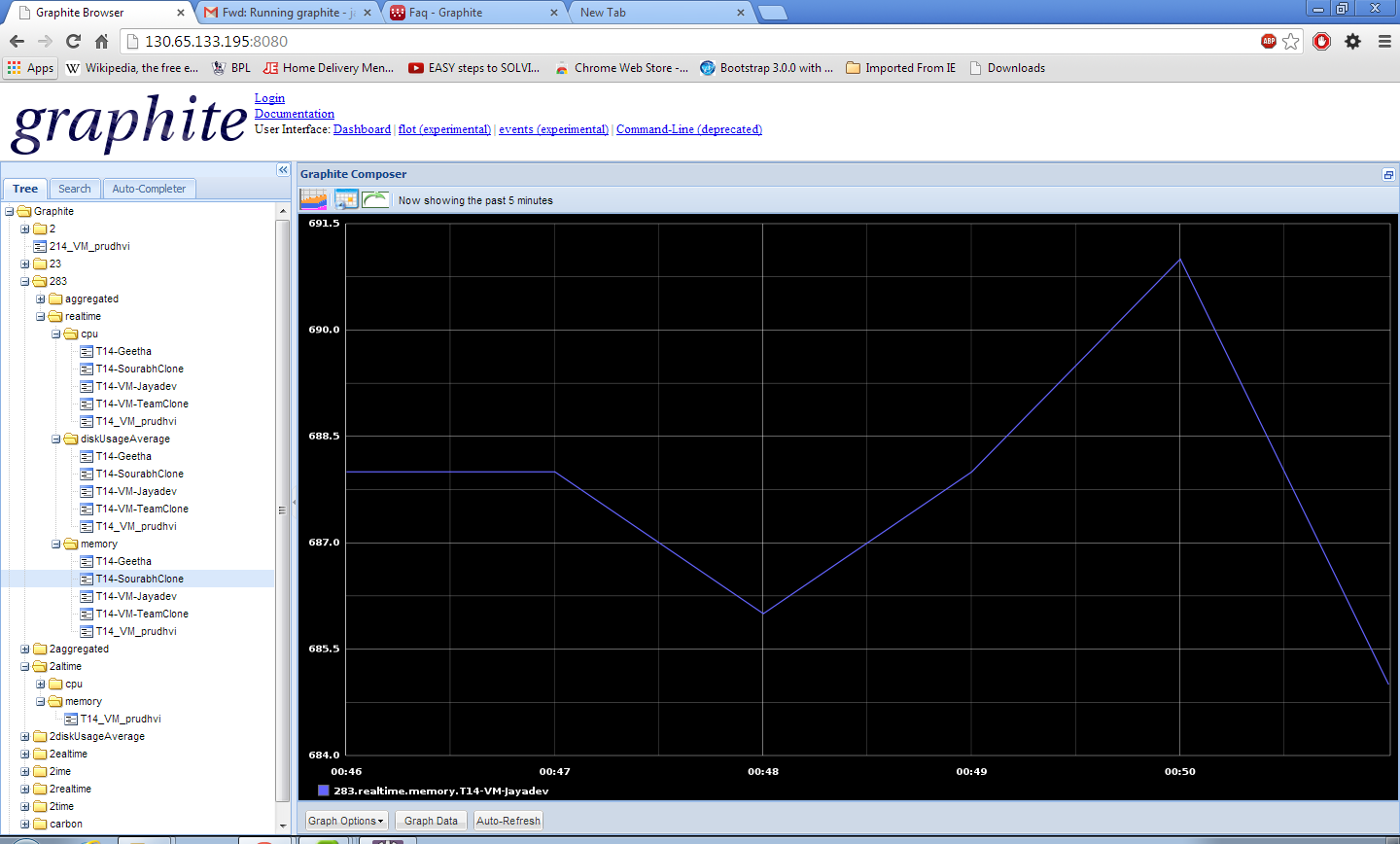


Figure: Real time Memory usage of VM

The overall graph presentation for VMs is as:

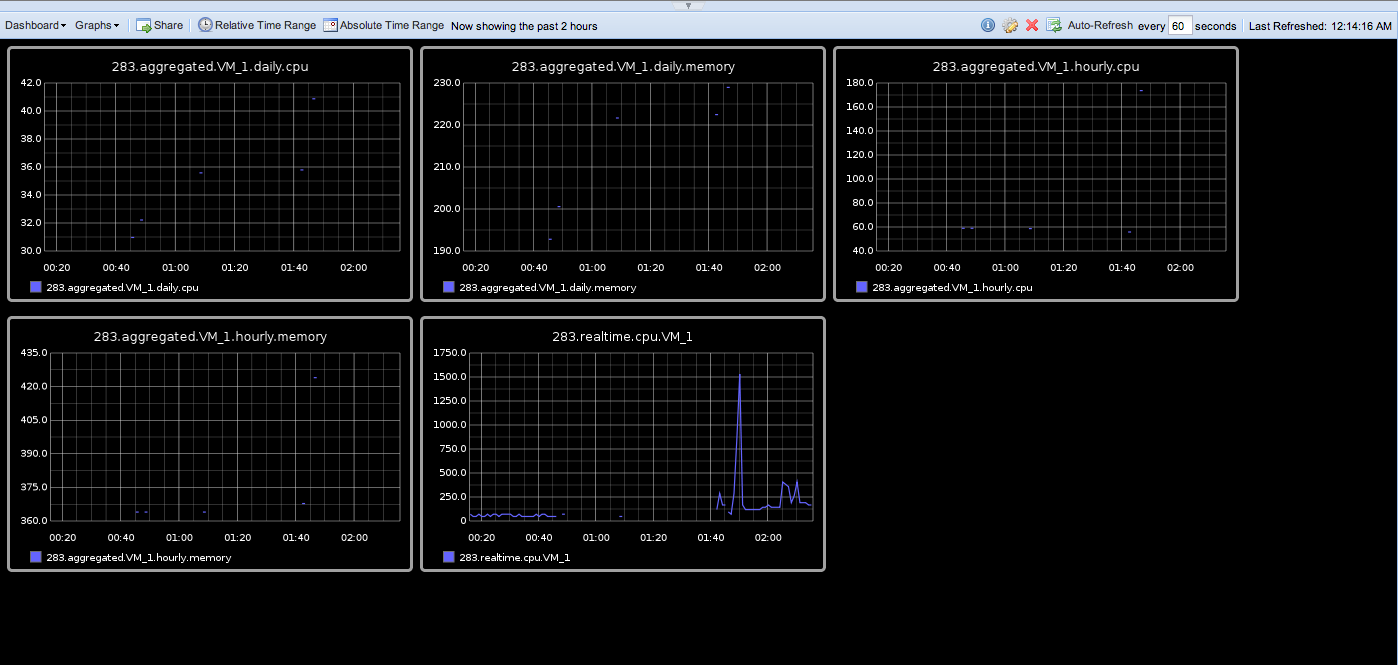


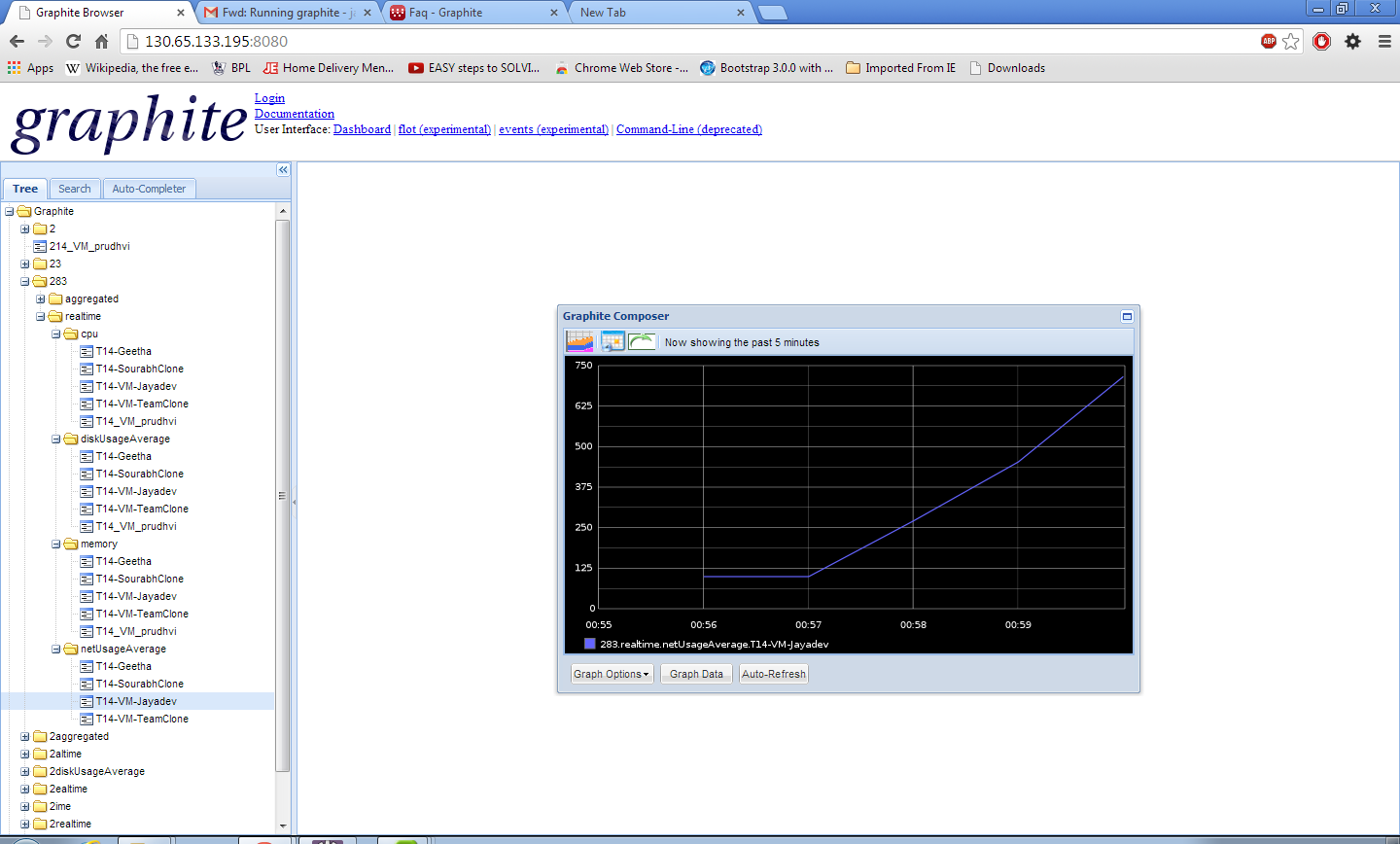
Figure : Overall graph representation

The IO statistics of the VMS is as:

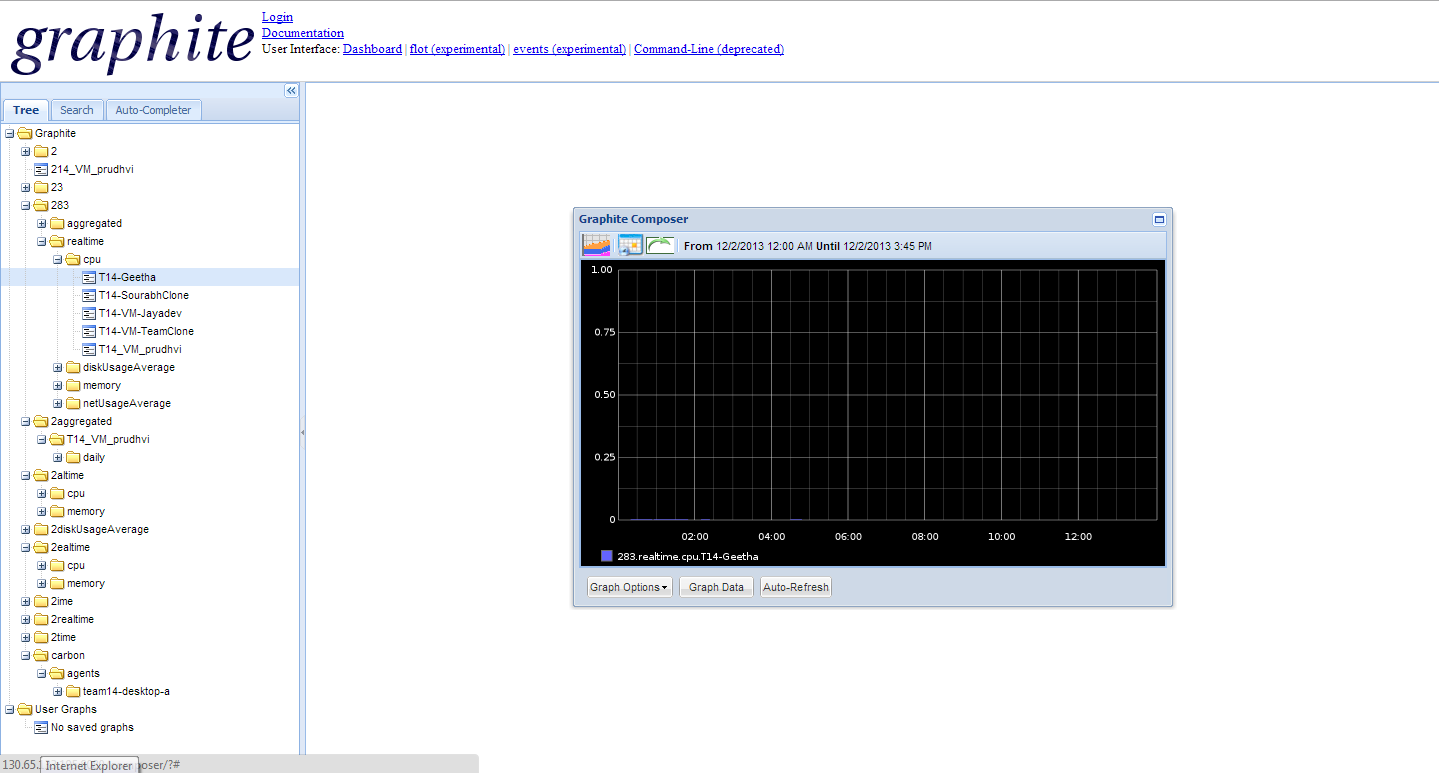


Figure: IO statistics of VM\_1

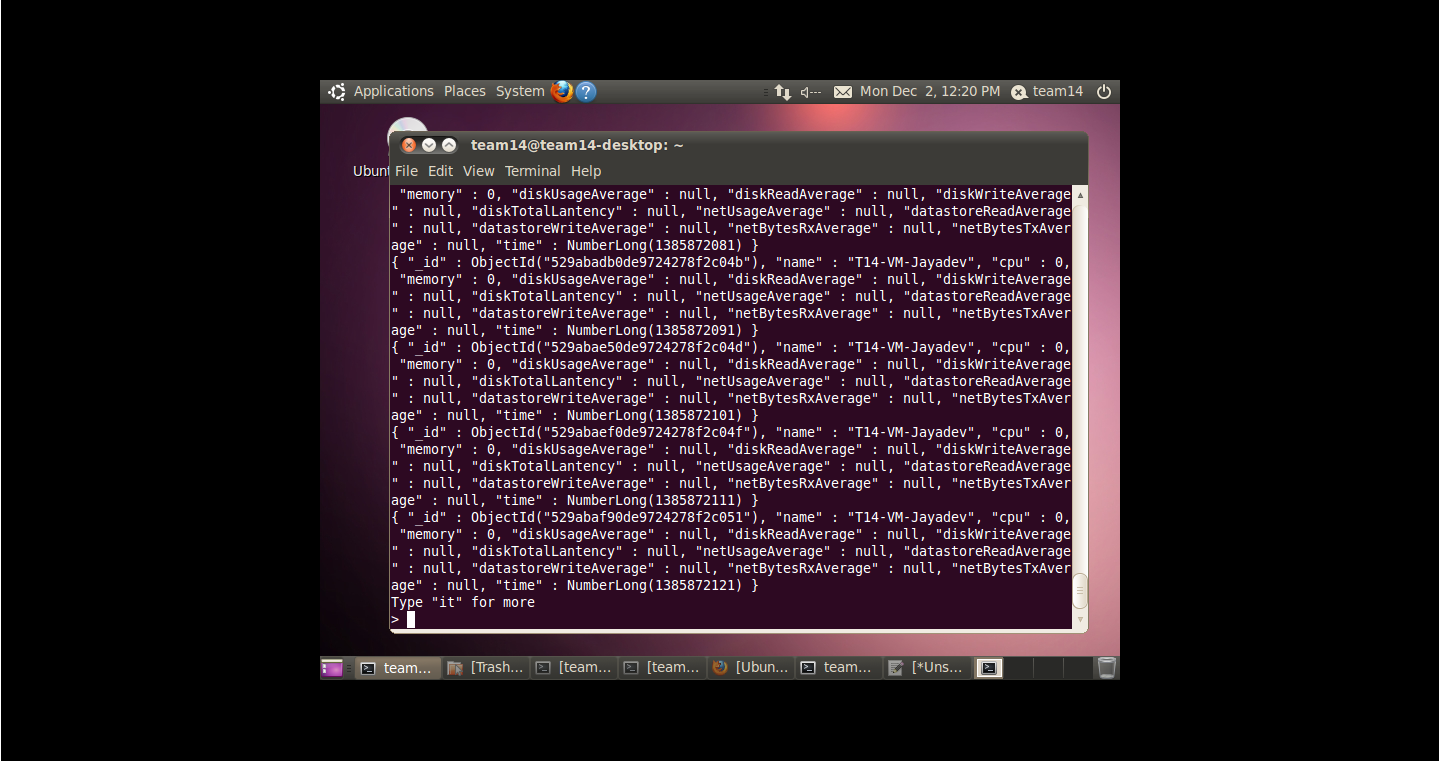
Network Statistics of the VM



Graph when a VM is powered off



**Screenshot: Mongodb instance showing the logs being inserted**

****

**ASSUMPTIONS AND LIMITIATIONS**

**Assumptions:**

A setup of five virtual machines and two hosts are expected. In those Vm’s set a virtual machine as a log collector and Log server. Log collector collects data from the rest of the virtual machines and the hosts. A log collector tool is expected to collect the logs and a nosql database is to be used to store those logs. The log values may not change for some particular intervals. In order to avoid redundant values in the database table purging of repetitive rows is to be done and pushed into Mysql database which will be connected to a visualization tool. A visualization tool is expected to show the performance metrics of the logs.

**Limitations:**

The whole framework that we implemented has some limitations.

1. The implemented framework currently collects just a few specified logs like CPU usage, Network usage, Memory consumption, etc. Any addition to the performance metrics collection will recoding.

2. The main tool used here, logstash, does not have the all of the plug-ins for many different inputs, filters and outputs.

3. it is essential that the logserver be on a linux system since, open-source tools, like Logstash, Elastic Search can only be used on a Linux based system.

4. The data we are graphing is not entirely real-time. There's transmissions delays and other delays taken during extraction of data or performance metrics, which is about 20seconds.

5. It is not possible to calculate the data in graphite itself.

We have installed logstash which collects the logs generated by an executable jar and pushes it into Mongo db which is a Nosql database to store those logs in realtime. There is a centralized VM which has Mongo db,Mysql, carbon and graphite installed on it. Carbon and graphite are the visualization tool software which monitoring the mysql database results and generates graphs accordingly.

**SECURITY CONFIGURATION:**

There are three layers of Security namely: virtualization layer, virtual machine and virtual networking layer.

**VIRTUAL MACHINE SECURITY:**

**Firewall:** In VM layer, **firewall** is the most obvious option for security.

**Resource reservation:** We can protect this layer further by setting up resource reservation and limits on the hosts. For example, through the detailed resource controls available in ESXi,,a virtual machine can be configured so that it always receives at least 10 percent of the host’s CPU resources, but never more than 20 percent. Resource reservations and limits protect virtual machines from performance degradation that would result if another virtual machine consumed excessive shared hardware resources. For example, if one of the virtual machines on a host is incapacitated by a denial-of-service (DoS) attack, a resource limit on that machine prevents the attack from taking up so much of the hardware resources that the other virtual machines are also affected. Similarly, a resource reservation on each of the virtual machines ensures that, in the event of high resource demands by the virtual machine targeted by the DoS attack, all the other virtual machines still have enough resources to operate.

**Virtual Networking Layer:**

**Firewall:** In a virtual machine environment, we can plan layout for firewalls between different components; physical machines such as vCenter Server systems and ESXi hosts.

One virtual machine and another—for example, between a virtual machine acting as an external Web server and a virtual machine connected to internal network.

**Approach:**

**Lockdown Mode:**

To increase the security of ESXi hosts, enable lockdown mode. When lockdown mode is enabled, no users other than vpxuser have authentication permissions, nor can they perform operations against the host directly. Lockdown mode forces all operations to be performed through vCenter Server. When a host is in lockdown mode, you cannot run vSphere CLI commands from an administration server, from a script, or from vMA against the host. External software or management tools might not be able to retrieve or modify information from the ESXi host.

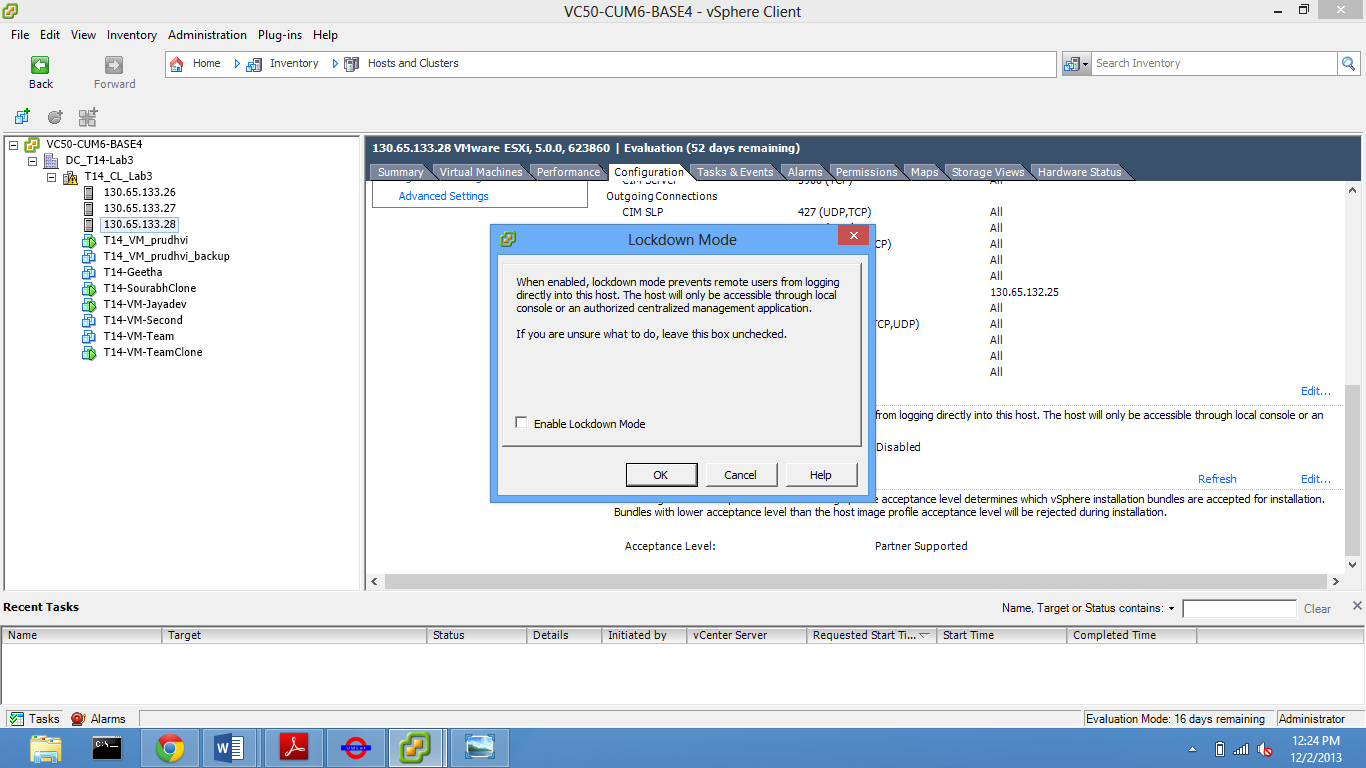


Figure :Lock down Mode

**Resource Reservation for VM:**

Resource reservation on each of the virtual machines ensures that, in the event of high resource demands by the virtual machine targeted by the DoS attack, all the other virtual machines still have enough resources to operate.

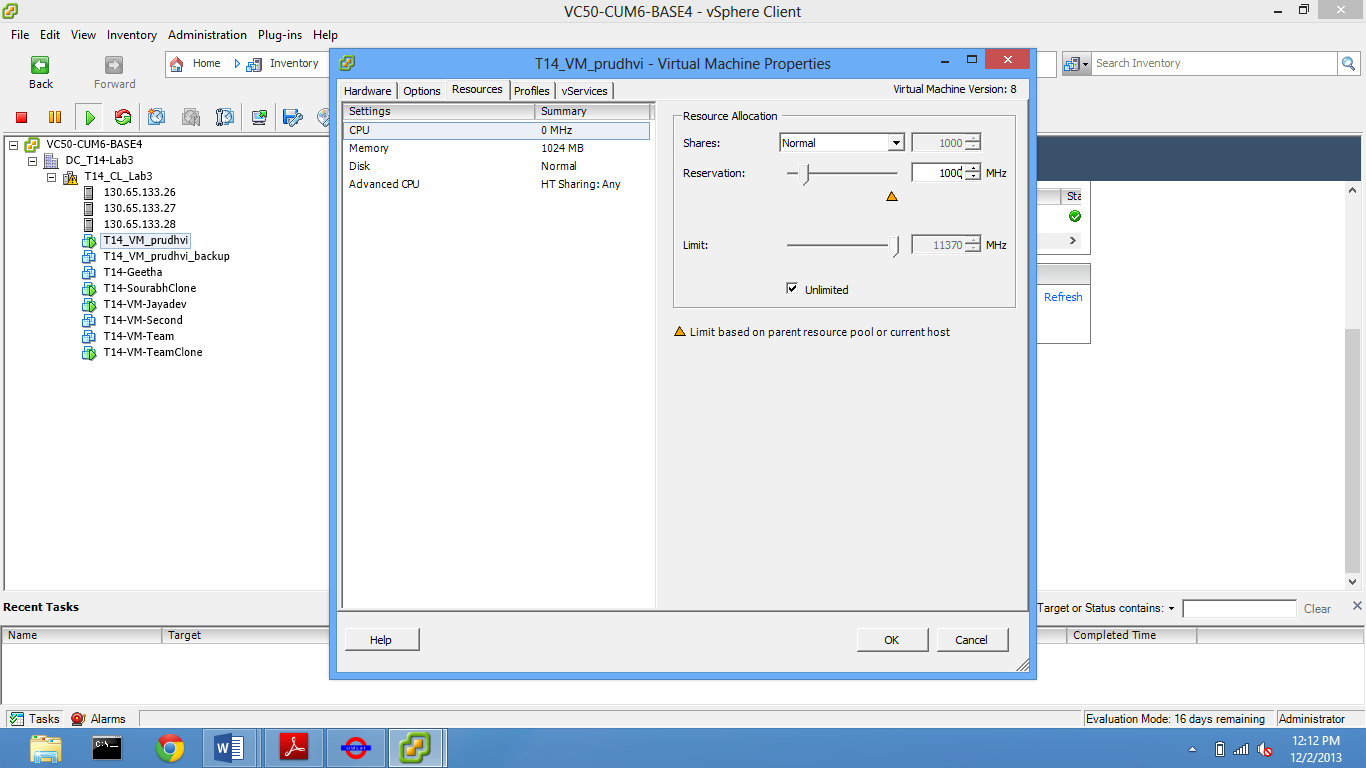


Figure: CPU Reservation



Figure: Memory Reservation

**VSphere Client security setting :**

When vShere Client under Firewall section is enabled to only certain list of IP address, this provides security to the access of that particular host or VM’s installed on it. It gives privacy as an advantage and provides safety and security to the content on that particular host’s VM. In the screens below, we can see that 130.65.133.195 host gets disconnected because the IP which was accessing it earlier is not listed in the allowed IP list.

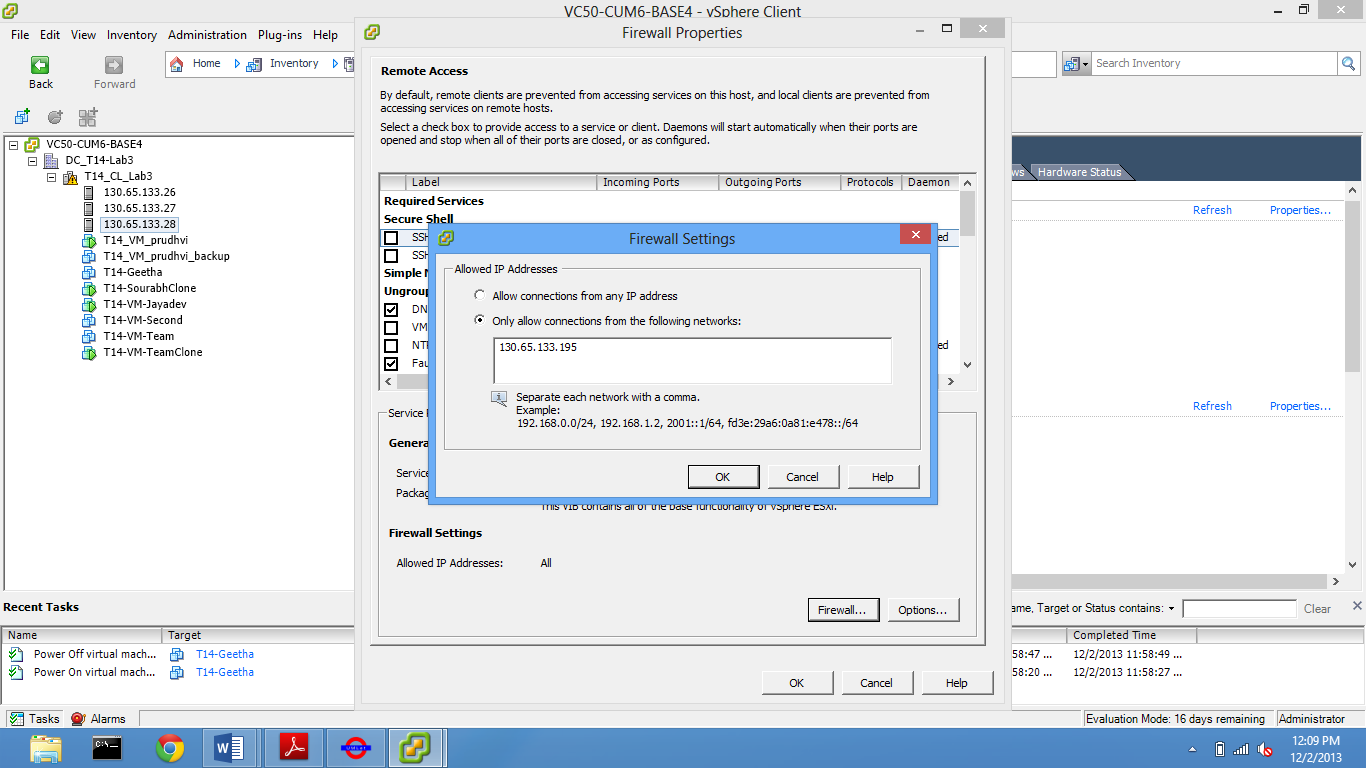


Figure: vSphere Client security setting

**Network Time Protocol Security:**

There are lots of security vulnerabilities which create a threat to the computers and the data saved on them if NTP Daemon is not running. A large number of services on modern corporate network require time to be synchronized within network or with absolute time and may fail if there are any problems with time synchronization. For example the following require time precision-

1. Expired software licenses/keys: ~weeks
2. Directory Synchronization: ~minutes
3. Collaboration services: ~minutes
4. Scheduling services: ~minutes
5. Authentication services (for example Kerberos): ~minutes
6. Logging: ~seconds
7. Production: ~milliseconds

So the security setting under firewall section – “NTP client” if is unchecked, stops the running NTP Daemon and causes the data a threat.

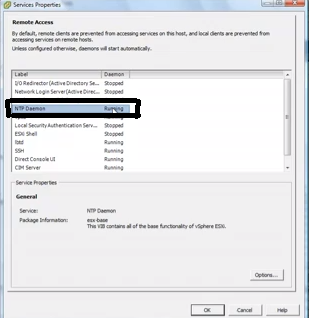


Figure:Network Time Protocol Security

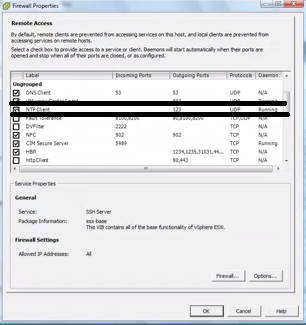


Figure:Network Time Protocol Security

NTP Daemon Stopped.

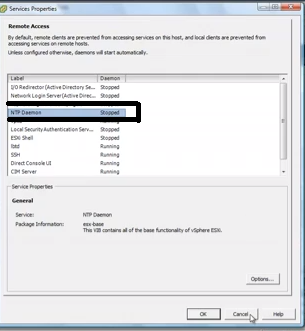


Figure:Network Time Protocol Security

**INSTALLATION AND EXECUTION MANUAL:**

**Installation:**

Steps to install each component as mentioned below:

* **MongoDb:**

1. Download the latest version of MongoDB from <http://www.mongodb.org/>
2. Follow the installation steps provided at <http://docs.mongodb.org/manual/tutorial/install-mongodb-on-ubuntu/> .
3. Make sure the MongoDb is been configured.
4. To start the MongoDb is done by issueing the following command.

sudo service mongodb start

* **Graphite:**

1. Check on the requirement for installing Graphite <http://graphite.wikidot.com/installation>
2. Get the package from <https://launchpad.net/graphite/+download/>
3. Install all the package (whisper, Carbon and graphite webapp)
4. Install carbon

**Execution:**

1. Import the src folder into Eclipse.

2. Add the VI JAVA and MONGODB jars to the jars folder in the project library.

3. Add log4j.properties files which is required for logger class and add it to workspace.

4. Add log4j.jar to the jar files.

5. Run mongodb on your mongohost which is your collector vm

6. Run “sudo service mongodb start “

7. cd /opt/graphite && sudo ./bin/carbon-cache.py –debug start

8. cd /opt/graphite && sudo ./bin/run-graphite-devel-server.py /opt/grahite

9. Run the supervisor.java file in the package.

10. Graphite visualization tool will be running on the server connected through 8080 port.

11. Run the graphite server with carbonhost ip followed by :8080 (130.65.133.195:8080)

**TESTING:**

Manual testing is performed for this system following ‘**bottom up approach’.** After integrating all these functionalities together**, integration testing** is performed manually following **end to end system testing**. Both **white box** testing is performed trying to cover all the **branches** and **conditions** as well **as black box** testing is also performed. The test cases are as follows:

| **Test Case Id** | **Test case Description** | **Test case Status** | **Expected Result** | **Actual Result** | | | Input and Output Files |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Pass** | | **Fail** |
| **1** | **VMStats.java** |  |  |  | |  |  |
| 1.1 | **Positive Test case:**  Test case ensures that it returns the VM name, CPU usage of the VM, network, storage stats and the time stamp. And make sure that the VM is Powered ON.  Verification Screen shot:  Screen shot performance of the VM from Vshpere Client |  | The current VM statistics is appended to the database with the time stamps appended with that. | Yes | |  | Input:  Screen shot from Vsphere:    Output:  Snap From DB (MongoDb)    Snap from UI: Graphite: |
| 1.2 | Test case ensures that it returns the VM name, CPU usage of the VM, network, storage stats and the time stamp when all the VM is powered off on the ESXI server |  | The CPU usage, memory should be No data to report for the particular instance of time. | Yes | |  | Input:  Screenshot from Vsphere:    Output:  Snap from UI: Graphite: |
| 1.3 | Test case ensures that the change in CPU usage, network, storage stats and the memory is collected when many VM are powered in on the same time. |  | The current VM statistics is appended to the database with the time stamps appended with that separately for each VM and the output is showed as the data collected for each VM separately. | Yes | |  | Input:  Screen shot from Vsphere:    Output:  Snap From DB (MongoDb)    Snap from UI: Graphite: |
| 1.4 | Data load:  After particular time period test case ensures that the Database is uploaded with current data. |  | MongoDB collections are updated with latest statistic taken from the VM | Yes | |  | Input:  Screen shot from Vsphere:    Output:  Snap From DB (MongoDb) |
| **2.0** | **Carbon.Java** |  |  |  | |  |  |
| 2.1 | Positive Test case:  Test case ensures that the data from the database is correctly pushed to the visualization using graphite graph. |  | The current data displayed on graph must match with database (MongoDb) | Yes | |  | Input:  Snap From DB (MongoDb)    Output:  Snap from UI: Graphite: |
| 2.2 | Negative Test case:  Test case ensures that the no data should be reported on Graphite graph when all the VM are switched off. |  | When the VM is switched off No information about VM gets loaded in MongoDb and displayed in graph. | Yes | |  | Input:  Screenshot from Vsphere:    Output:  Snap from UI: Graphite: |
| 2.3 | Test case ensures that all the data are pulled and displayed correctly when many VM are running at the same time |  | The Graphite graph must show the chart for each VM’s in different color | Yes | |  | Snap from UI: Graphite: |
| 2.4 | Test case ensures that the Statics of VM is thrown out every 15 minutes, Every 1 hour. |  | When the timestamp on the graph changes the respective chat on the time interval is been showed off. | Yes | |  | Snap from UI: Graphite: |
| 2.5 | Using JUNIT:  Test case ensures that when Null value is passed to the port number and host it should throw out an error in Junit. |  | When Junit run is performed on the file it should throw up a error as Null parameter is passes. | Yes | |  | Carbontest.java:    Junit Output ScreenShot: |
| 3.0 | Logger.java |  |  |  | |  |  |
| 3.1 | Test case ensures that the Log file has been updated when any change is been made |  | The log file has to be updated with current changes on it. | Yes | |  | Input:  Snap From DB (MongoDb)    Snap from UI: Graphite:    Output:  Log File: |
| **4.0** | **Supervisor.java** |  |  |  | | |  |
| 4.1 | Test case ensures that the data is been dispatched to correctly as per the request from the Grapite. The Data’s that should be showed up in graph chat are CPU usage, memory, packets sent, storage etc. |  | When a single VM is Powered ON the stats must be stored in MongoDb, and respective data must be sent to grapite chat by carbon.java module | Yes |  | | Input:  Screen shot from Vsphere:    Output:  Snap From DB (MongoDb)    Snap from UI: Graphite: |
| 4.2 | Test case ensures that the data is been dispatched to correctly as per the request from the Grapite. The Data’s that should be showed up in graph chat are CPU usage, memory, packets sent, storage etc. When multiple VM’s are switched ON |  | When a multiple VM is Powered ON the stats must be stored in MongoDb on respective name of the VM, and respective data must be sent to grapite chat by carbon.java module | Yes |  | | Snap from UI: Graphite: |
| 4.3 | Test case ensures that the data is been dispatched to correctly as per the request from the Grapite. The Data’s that should be showed up in graph chat are CPU usage, memory, packets sent, storage etc. When VM’s are switched OFF. |  | No date must be showed or thrown out to graph | Yes |  | | Input:  Screenshot from Vsphere:    Output:  Snap from UI: Graphite: |
| **5.0** | **Visualization – Graphite chat** |  |  |  |  | |  |
| 5.1 | Test case ensures that the chat displays the correct value from the Database. |  | The Stats shown up in graph must match with the value from the DB. | Yes |  | | Input:  Snap From DB (MongoDb)    Output:  Snap from UI: Graphite: |
| 5.2 | Test case ensures that when stats of multiple VM are displayed it should be differentiated with different colors. |  | The Stats if each VM is shown up with different color. | Yes |  | | Snap from UI: Graphite: |

**SCREENSHOTS WITH ANNOTATIONS**

**Screenshot1: CPU Usage graph of the VMs**

****

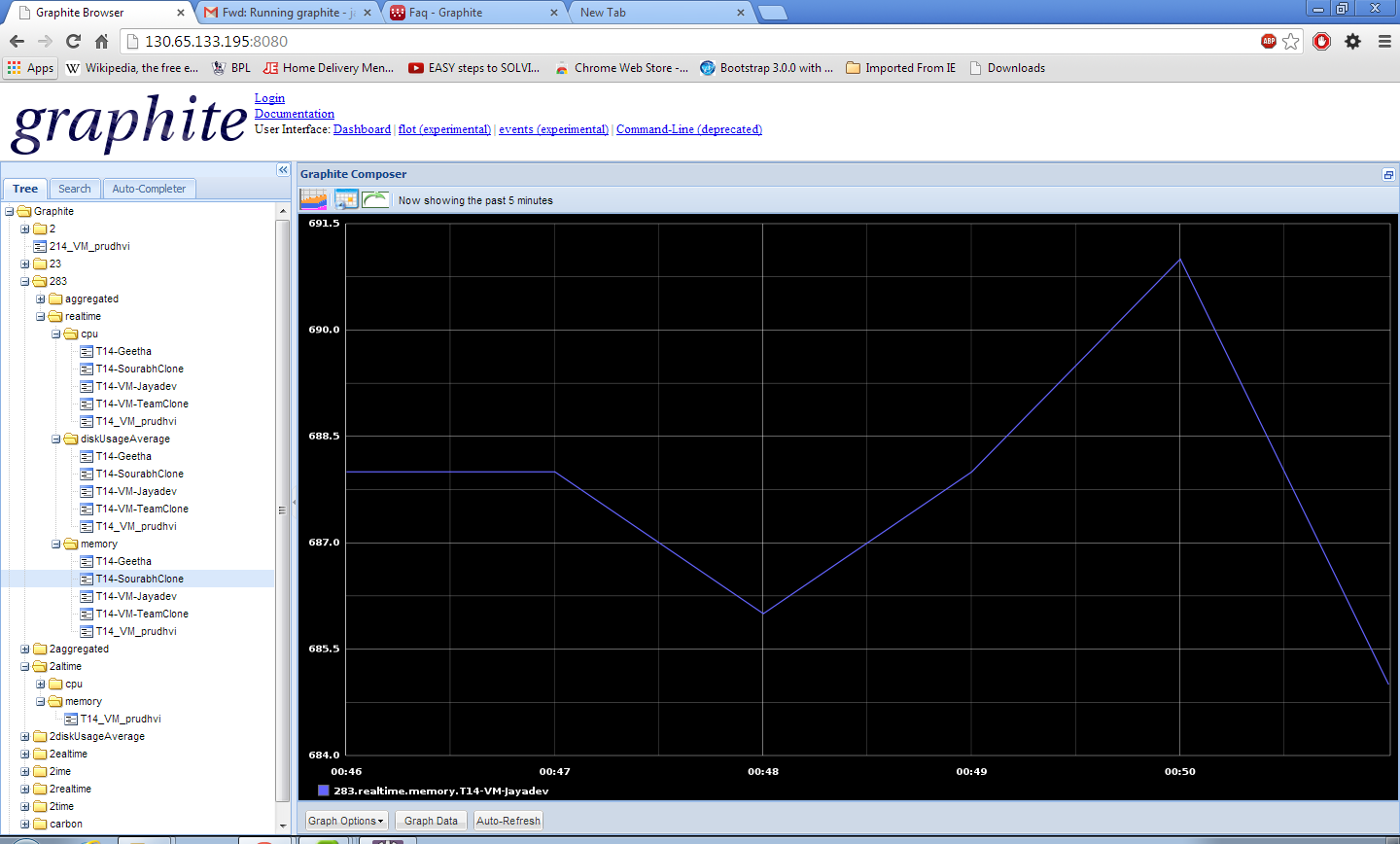
**Screenshot2:**

****

**Screenshot3: MemoryUsage graph of the VM\_jayadev**

****

**Screenshot 4: MemoryUsage graph of the VM\_jayadev in different scenario**

****

**Screenshot5: Disk Usage graph of all the VMs**

****

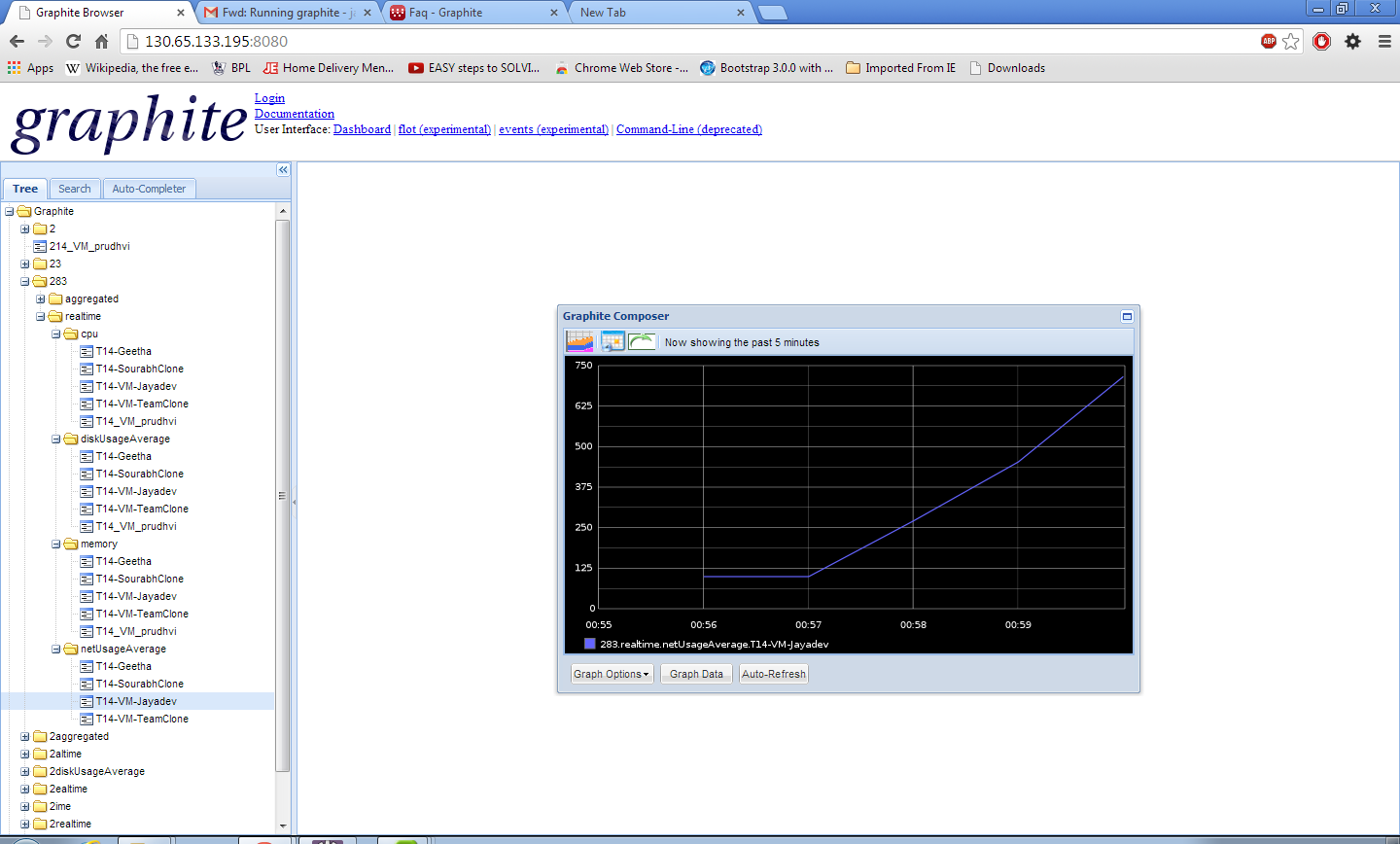
**Screenshot 6: Io-DiskUsage statistics Of VM-Sourabh**

****

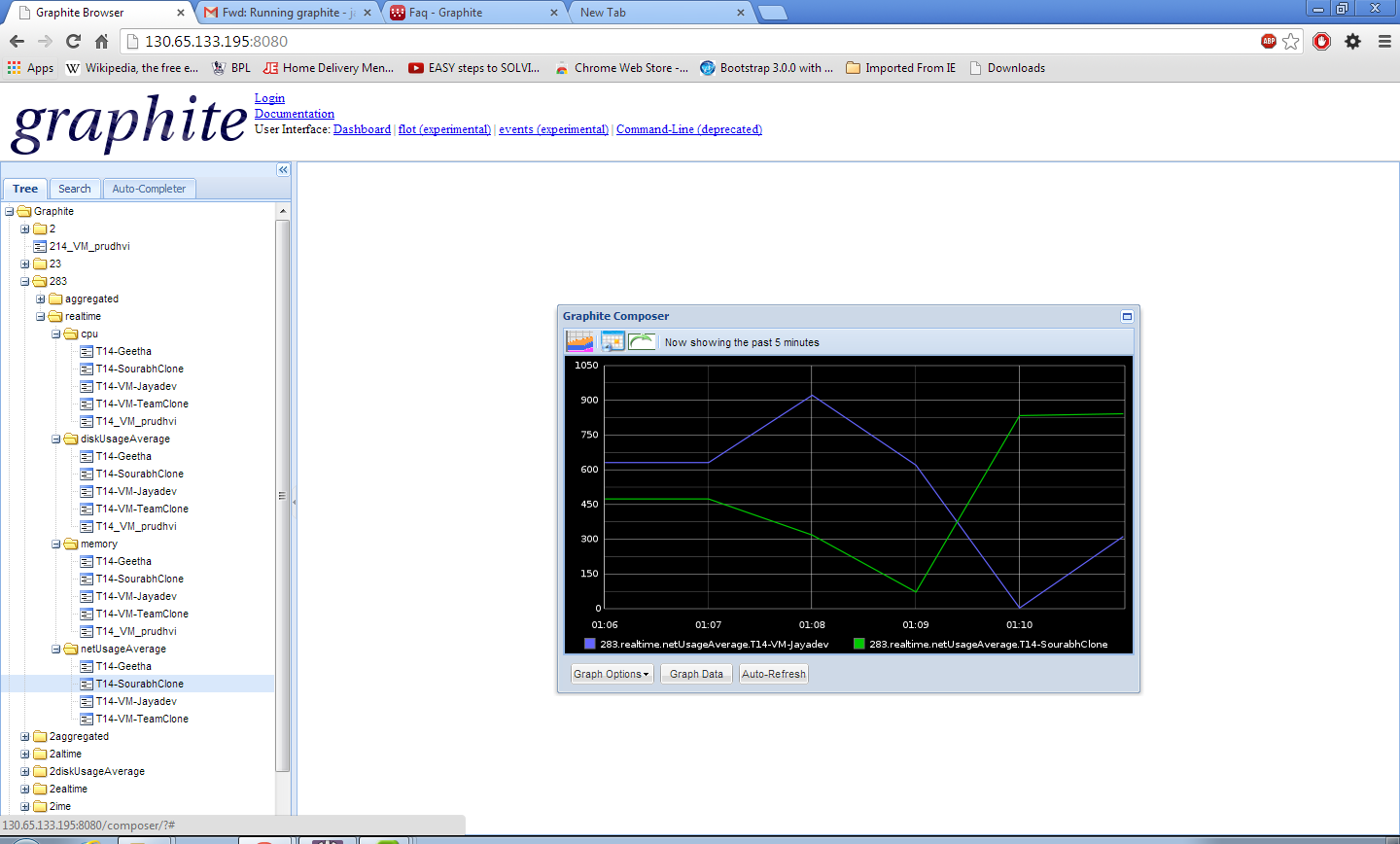
**Screenshot7: Io-DiskUsage statistics Of VM\_jayadev**

****

**Screenshot8: Network Usage graph of the VM\_jayadev**

****

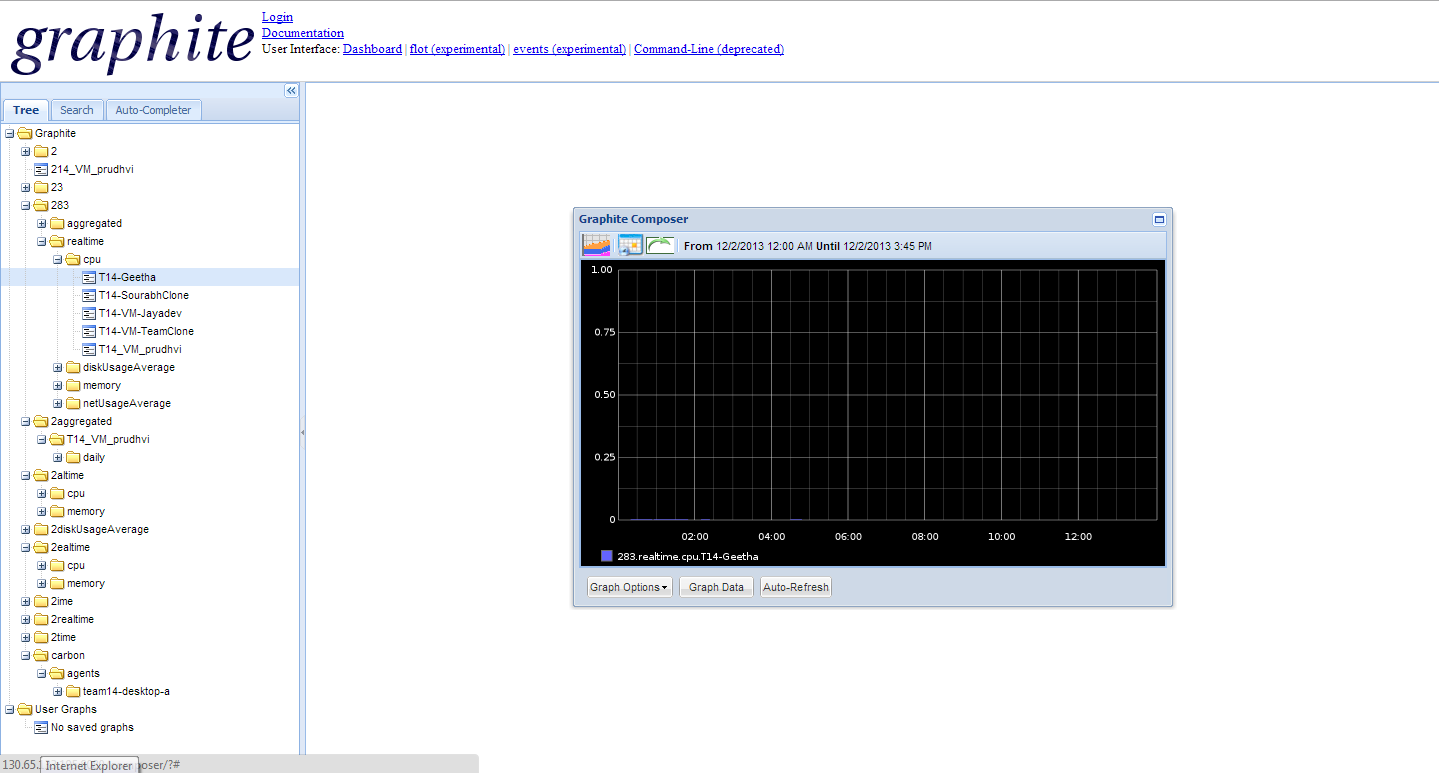
**Screenshot9: Network Usage graph of the VM\_jayadev, sourabh**

****

**Screenshot 10: Disk read graphs of VMs**

****

**Screenshot 10: VM switched off**

****

**FUTURE WORK:**

We plan to connect and integrate Logstash and NoSQL database and later connect with Graphite so that we can have a unified ,even more efficient logging and metrics interface.

**INDIVIDUAL CONTRIBUTION:**

|  |  |
| --- | --- |
| **Team Member** | **Contribution** |
| Jaidev Akkiraju | Log Collecting, Metric Collecting, Visualization, Presentation, MongoDB Aggregation |
| Prudhvi Chandra | Log collecting, Metric Collecting, visualization |
| Mintu Abraham | Security configuration, Presentation |
| Geetha Anne | Report, Database, Testing |
| Sourabh Bhosale | Report, Database, Security configuration |

**CONCLUSION:**

The following two concepts are concluded at the end of this project:

**LESSONS LEARNT:**

1.This project helped to get a better understanding of how to use VI java APIs to manipulate and parse logs as well as report graphs using Graphite.

2.The real time and aggregate graphs for memory, CPU and network usage is plotted periodically for each VM.

3. We gained good knowledge in Mongodb data storage and retrieval techniques.

4.We learned how to integrateMongoDB with the VI JAVA for the log storage and retrieval.

5.We faced the difficulty in pulling the parsed log data from mongodb and store it in the MYSQL.But later progression in the projeect helped us clearly understand that.

**CHALLENGES:**

1. We found the Installation of Graphite on the Ubuntu12.0.4 was a tricky process. It gave us many kernel level errors which we successfully rectified.
2. Initializing MongoDB at first was generating the mongd.lock which made us to consistently remove the lock and repair the database.

# References

1. Virtual Infrastructure VMware API reference documentation:

<http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/ReferenceGuide/>

2. VMware Infrastructure Architecture:

<http://www.vmware.com/pdf/vi_architecture_wp.pdf>

3. Programming in VI JAVA API helpful book:

VSphere Web services SDK Programming Guide

4. ESXi and VCenter Server 5 Documentation:

<http://pubs.vmware.com/vsphere-50/index.jsp#com.vmware.vsphere.doc_50/GUID-553E2EBD-6D19-4873-98FD-265B3A92F1F0.html>

5. Links helpful for Testing related information:

1.<http://stackoverflow.com/questions/520064/what-is-unit-test-integration-test-smoke-test-regression-test>

2. <http://www.typemock.com/unit-tests-integration-tests>

6.Jin S.; Retrieved from: http://www.doublecloud.org/2011/11/what-roles-does-a-user-have-in-vsphere/ on 2013, 11th May.

7.<http://eatingsecurity.blogspot.com/2013/04/home-lab-part-2-vmware-esxi-security.html>, Retrieved on 2013, 11th May

8.<http://daveshackleford.com/?p=658>, Retrieved on 2013, 11th May

9.[http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-security- guide.pdf](http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-security-%20%20guide.pdf), Retrieved on 2013, 11th May

