SAN JOSÉ STATE UNIVERSITY

CMPE – 283 – Project 2

Spring – 2015

Large Scale Performance-Statistics gathering

Submitted to

Professor Simon Shim

Department of Computer Engineering

Submitted By - Team 14

Aravind Ganesh (010056956)

Ashutosh Chandane (010011443)

Sudhakar Kamanboina (010011417)

Neethi Srinivas Murthy (010027680)

Swetha Rathna Kempahanumaiah (009343477)

CONTENTS

Table of Contents

1. INTRODUCTION:

The importance of virtualization has always grown drastically with the growth of cloud infrastructure. Nearly every organization use virtualization for setting up the development and test environments. Simultaneously, the logs generated by such infrastructure is collected and monitored for performance purpose and mainly for balancing the loads across different VM if needed. Representation of such logs and analyzed data is also gaining importance for administration purpose specially when there is large scale metrics.

1.1. Goals:

As part of the project 1, a private cloud was created for maintaining VM statistics. As an extension of project 1, an administrator module for collecting and analysis of large scale metrics of VMs and VHosts using open source tools and display data in visual format is to be implemented.

1.2. Objectives:

The key objective of the project is to collect the real time performance statistics of the virtual machines. The data are also stored in a scalable database in order to support the continuous collection of performance statistics. This contains both the historical data and transactional data, which are consolidated and displayed for analytical purposes. The statistics are presented in the form of graph which helps in better analysis and understanding.

1.3. Needs:

In order to collect continuous performance data from each VM and its VHost, there's a need for a performance manager application which is running in each VM and logging required statistics. Also, storing of these millions of data in a structured format should also be taken care. For representation of the data, proper visualization library and tools should be identified.

2. BACKGROUND:

Through this project, performance analysis and metrics helps the administrator to manage the infrastructure and also give an insight into the cloud infrastructure. Continuing on the project 1 private cloud, an additional module with collects continuous statistical data of each of the VM and VHost is needed. To achieve this, certain agents and collectors are installed and configured on each of the VMs which collects and dumps logs to a centralized server. Since there is lot of data retrieved, say millions of data per VM, a structured format is required to maintaining this information. Using certain visualization tools, these VM data is presented in an analysis report based on which an administrator can perform auto migration of certain VMs if it reaches certain threshold.

3. REQUIREMENTS

3.1. Functional Requirements

1. Collect the performance statistics logs of the available hosts and virtual machine.
2. The statistics logs collected should include – Per Guest Level CPU, Memory, Threads, I/O, VMotion.
3. Collector Process that will Poll, Process and Archive collected data must be using Logstash tool.
4. Atleast one ‘grok’ pattern must be used in logstash to format log data.
5. The logs collected must be stored in noSQL database like ElasticSearch.
6. Presentation or Visualization tool like Kibana should be used for displaying statistical data in a chart format.
7. Ability to fetch a log record based on certain criteria like time range or search string.

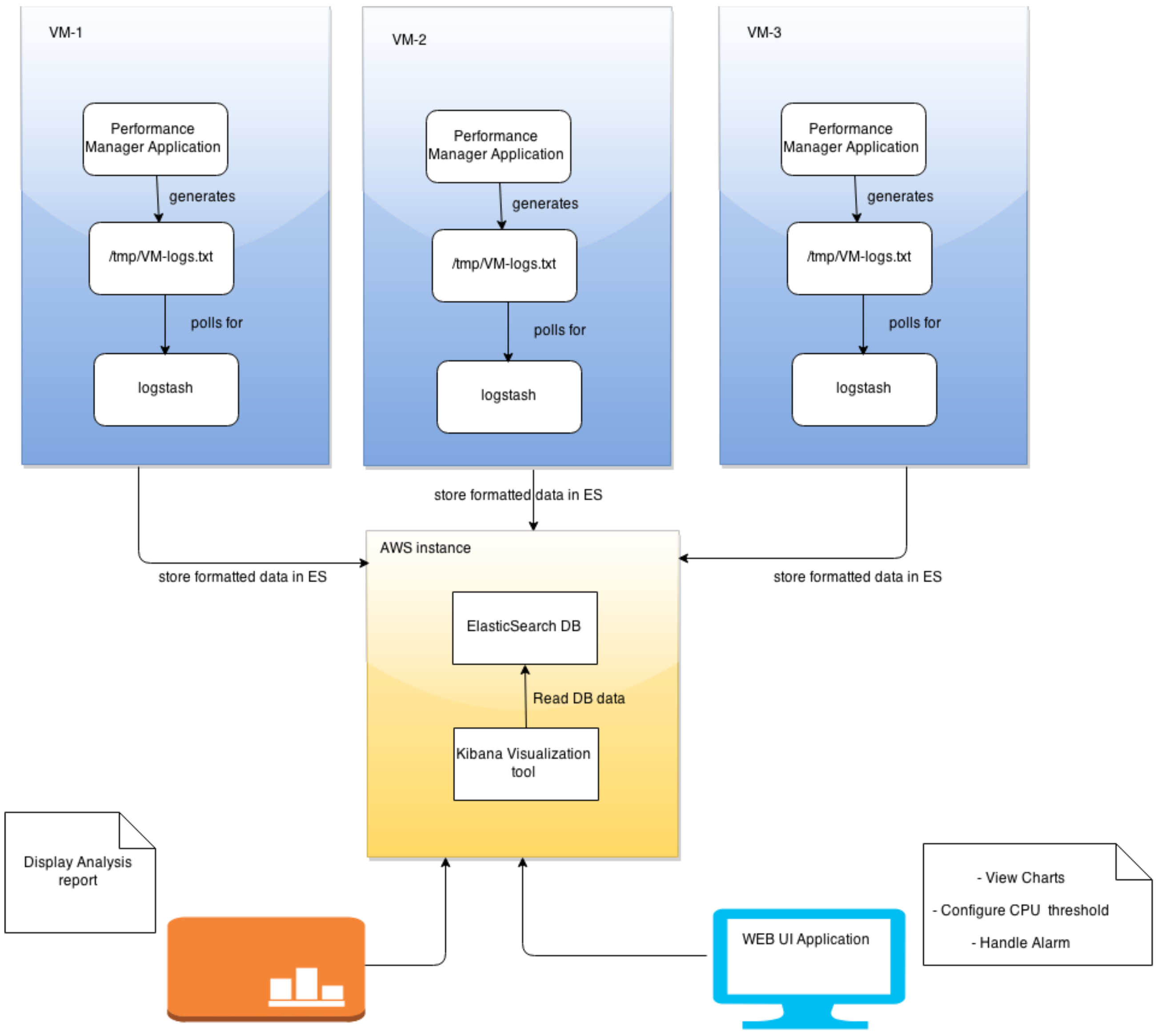
8. An alarm to watch a single metric that the user specifies. An email or a display popup needs to be performed once the alarm is raised.

3.2. Non-Functional Requirements

1. Agents and the collector can be started independently.
2. The database used should be able to manage huge unformatted data with a data range of 0.5 million records.
3. The data presentation should be simple and easily understandable.

4. DESIGN:

The aim of the project is collecting large scale statistical data and generate analysis reports which helps in rescheduling resources across different VHosts. In order to collect continuous logs of VM and VHosts, a java based application, performance manager program is implemented using the VMWare APIs. This application is executed on each of the VMs. This program generates an output file dumping the required data at an interval of 60seconds. The statistical parameters that are gathered are VM Name, CPU usage, Network statistics, Memory statistics and System statistics.   Below shows all the components involved:



The main components are described below:

Performance Manager Application:

This is a java application which implements log collection on each of the VM. This application is executed as a startup program when the VM boots up. Using the VI APIs like – "PerfEntityMetricBase" java class, the statistical values are retrieved on a periodic interval of

60seconds. This statistical data is dumped into a text file and stored in /tmp so that logstash can poll on it for update.

Logstash script:

Logstash event management tool is installed on each of the VMs and configured to poll on certain file update. The /tmp/VM-log.txt file that is generated by performance manager is polled by logstash for any update. So if any new data is appended by performance manager, then the logstash script reads through the entries and creates a formatted data with name, value pair and loads into an ElasticSearch database for analysis purpose. Currently we have installed and configured logstash in 3 VMs to support fault tolerance.

ElasticSearch DB:

The millions of data generated from the performance manager had to be stored in a distributed and easily manageable database. ElasticSearch was chosen for that. Using the AWS ElasticSearch instance, a centralized database of data is maintained.

Visualization Tool:

The vast amount of logs is to be displayed in the form of graphs for easier analysis. Kibana visualization tool is used for the creation of graphs. The data in ElasticSearch DB is used as the source of data for generating graphs. An AWS instance for ElasticSearch is used as part of the development setup. Kibana display features like displaying based on aggregated data setup or string search is used.

Web UI Application:

A web UI application is needed for maintaining alarms when a given metric reaches a threshold. When the VM statistic metric reaches that threshold, certain action will be performed on the UI like displaying a popup and sending an email.

5. IMPLEMENTATION

5.1. Environment

Each of the team have been assigned a separate data center. Our team data center (130.65.132.114) has two hosts created with VMWare ESXsi installed. The host ipaddress are: 130.65.133.22 & 130.65.133.23. The performance manger application is running as startup process on two VMs under host: 130.65.133.23. The same two VMs has logstash installed and configured such the collected data is send to Elasticsearch DB. An independent centralized server with ElasticSearch and Kibana is created using the AWS instance, so that all the VM's logstash script can dump its data to this centralized server. The same instance can be used for using Kibana visualization tool which reads the data from ElasticSearch and generates reports/charts per VM. The web UI application implemented for threshold configuration and alarm monitoring is run on a standalone machine which also connects to the AWS instance for VM statistical data.

5.2. Tools

Eclipse: Eclipse IDE is an open source tool, which is used for developing the java programs for the collection of performance metrics from hosts and virtual machines.

Logstash: Logstash is an open source tool, we have used it for collecting, managing and parsing the logs and storing the logs into ElasticSearch DB. The important configurations in the Logstash done for our project are:

- Input: Inputs are the mechanism for passing logs to Logstash. In our project, the file generated from Performance Manager Program should be the file path to be entered in 'file' field of input.

- Filter: Used for intermediate processing of data. Using the grok pattern, each line input from the /tmp/VM-log.txt is read and filtered into a structured format so that it can be stored in a proper format in DB.
- Output: Is the final phase of logstash pipeline. In our scenario, we need to store the statistical data in DB. So the output field is configured with ElasticSearch DB access information.

ElasticSearch DB: ElasticSearch DB is a nosql DB, which is used for storing the large amount of unstructured data generated in the form of logs.

Kibana Visualization tool: Kibana is a simple and flexible API, which is used to creating graphs and virtualizing the performance statistics. The data are pulled from the ElasticSearch database for each VM and displayed in the form of graph.

Stress testing tool:

Stress is a stability testing tool used for varying the CPU load in the virtual machines by either increasing or decreasing the load. This is used primarily to test the workability of the performance manager by varying the CPU load to maximum and minimum loads and obtaining different statistics.

5.3. Implementation Approach:

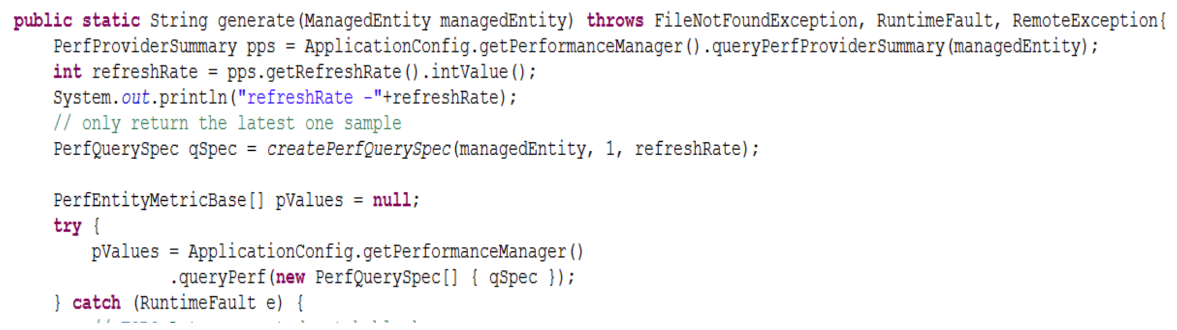
Performance Monitor:

In order to collect performance metrics, VMware java API are used and a log file is generated for each VM. The subsequent logs for VM and its VHost gets appended correspondingly as the performance metrics collector runs continuously on them. This log files is filtered to extract the

5 essential metrics of the VM namely, CPU usage, memory usage, power usage, network usage which is further dumped into the log text file.

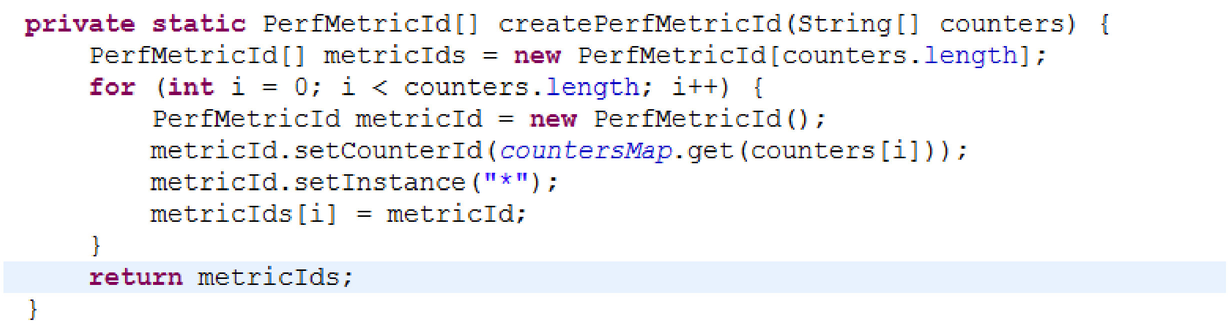
Steps followed to collect the metrics:

Performance manager is created for host and VM entity. To list the available intervals present, QueryPerfProviderSummary is used as shown below:



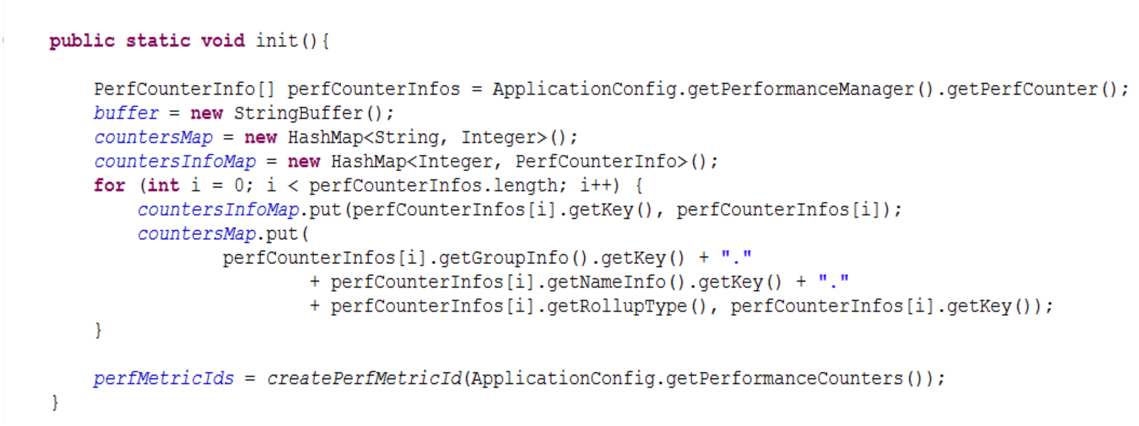
```
public static String generate(ManagedEntity managedEntity) throws FileNotFoundException, RuntimeFault, RemoteException{
    PerfProviderSummary pps = ApplicationConfig.getPerformanceManager().queryPerfProviderSummary(managedEntity);
    int refreshRate = pps.getRefreshRate().intValue();
    System.out.println("refreshRate -"+refreshRate);
    // only return the latest one sample
    PerfQuerySpec qSpec = createPerfQuerySpec(managedEntity, 1, refreshRate);

    PerfEntityMetricBase[] pValues = null;
    try {
        pValues = ApplicationConfig.getPerformanceManager()
                .queryPerf(new PerfQuerySpec[] { qSpec });
    } catch (RuntimeFault e) {
```

The list of metricIds are created of `PerfMetricId object type:`



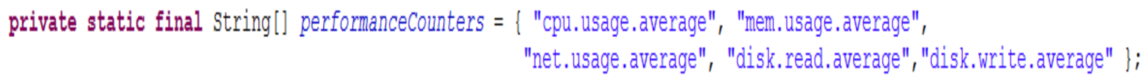
```
private static PerfMetricId[] createPerfMetricId(String[] counters) {
    PerfMetricId[] metricIds = new PerfMetricId[counters.length];
    for (int i = 0; i < counters.length; i++) {
        PerfMetricId metricId = new PerfMetricId();
        metricId.setCounterId(countersMap.get(counters[i]));
        metricId.setInstance("*");
        metricIds[i] = metricId;
    }
    return metricIds;
}
```

Below code snippet shows the key decriptions of each of the metrics.



```
public static void init(){

    PerfCounterInfo[] perfCounterInfos = ApplicationConfig.getPerformanceManager().getPerfCounter();
    buffer = new StringBuffer();
    countersMap = new HashMap<String, Integer>();
    countersInfoMap = new HashMap<Integer, PerfCounterInfo>();
    for (int i = 0; i < perfCounterInfos.length; i++) {
        countersInfoMap.put(perfCounterInfos[i].getKey(), perfCounterInfos[i]);
        countersMap.put(
                perfCounterInfos[i].getGroupInfo().getKey() + "."
                        + perfCounterInfos[i].getNameInfo().getKey() + "."
                        + perfCounterInfos[i].getRollupType(), perfCounterInfos[i].getKey());
    }

    perfMetricIds = createPerfMetricId(ApplicationConfig.getPerformanceCounters());
}
```

The filtered keys are:



```
private static final String[] performanceCounters = { "cpu.usage.average", "mem.usage.average",
                                                     "net.usage.average", "disk.read.average","disk.write.average" };
```

ElasticSearch DB support:

An amazon instance is created for ElasticSearch support. The public ipaddress is later used by all the other servers like – logstash script, Web UI backend program.

Kibana Visualization implementation:

An amazon instance is created for Kibana as well. Using the dashboard features of Kibana, different dashboards for Overall statistics dashboard, VHost statistics dashboard and VM statistics dashboard are generated. Chart features like –Pie chart, Line chart, Area chart, Vertical bar chart are used. Using these display charts, the statistics across VM and VHost are analysed for networking utilization, disk read/write utilization, CPU utilization and memory utilization. Aggregation data like maximum, unique count and average data are generated for VM and VHost is generated using Kibana's aggregation options.

Web UI Application for Alarm:

Using the code base of Project1, the alarm feature is implemented as an extension of project 1.

Some of the implementation changes done are:

- The Home page(listVM.jsp) is updated with an additional button as "Monitor"
- The web page (listVM.jsp) which displays all the VMs details, has an additional parameter displaying the alarm status.
- A new webpage (monitor.jsp) is created with options for configuring a new alarm, displaying VM & VHost statistics is added.
- Backend MySQL database is extended to support additional database table with alarm information.

6. ASSUMPTIONS:

1. ElasticSearch support large memory space to store all the log data.
2. Logstash take care of creating a format output data according the filters provided.
3. Memory space to support log collection of atleast 3 VMs is available in our setup.

7. LIMITATIONS:

The whole framework that we implemented has some limitations.

1. Currently only 3 VM and VHosts are monitored to collect logs for computational and storage capacity limitations.
2. Only 5 performance metrics like CPU usage, Memory consumption, Network usage etc are collected for each VM and rest of the metrics are not considered as part of this project.
3. The log collector tool used here, Logstash, is not compatible with the all of the plug-ins for various inputs, filters and outputs formats.
4. Kibana visualization tool doesn't expose APIs for setting an alarm based on certain user threshold value. Hence a separate Web UI application is support for alarm support.

8. FUTURE WORK AND ITS EXTENSION:

1. All VM performance metrics can be collected instead of only 5 parameters which can be graphically represented for analysis purpose.
2. Alarm support could be handled as part of Kibana dashboard if any APIs are available in the future, instead of having separate UI application to support that feature.
3. A mechanism to flush the DB data collected over years or months can be taken care too.

9. INDIVIDUAL CONTRIBUTION:

Name	**Contribution**
Aravind Ganesh	• Alarm support feature – Front end changes. • ElasticSearch configuration.
Ashutosh Chandane	• Kibana configuration and creating analysis charts. • Dashboard design for VM, VHost and Overall statistics. • “Start stop service for Log generation” feature implementation.
Neethi Srinivas Murthy	• Alarm support feature – Web UI Front end changes. • Web UI front end changes for display threshold charts. • Report (Testing Section)
Sudhakar Kamanboina	• Performance Manager log generation java application. • Logstash configuration in VMs. • Alarm feature support – Web Backend code implementation. • Overall integration and system level testing of all modules.
Swetha Rathna Kempahanumaiah	• Performance Manager Application design and logstash configuration. • Alarm support feature – Web UI CSS changes. • Stress testing • Writing Report

10. INSTALLATION AND EXECUTION MANUAL

Logstash installation:

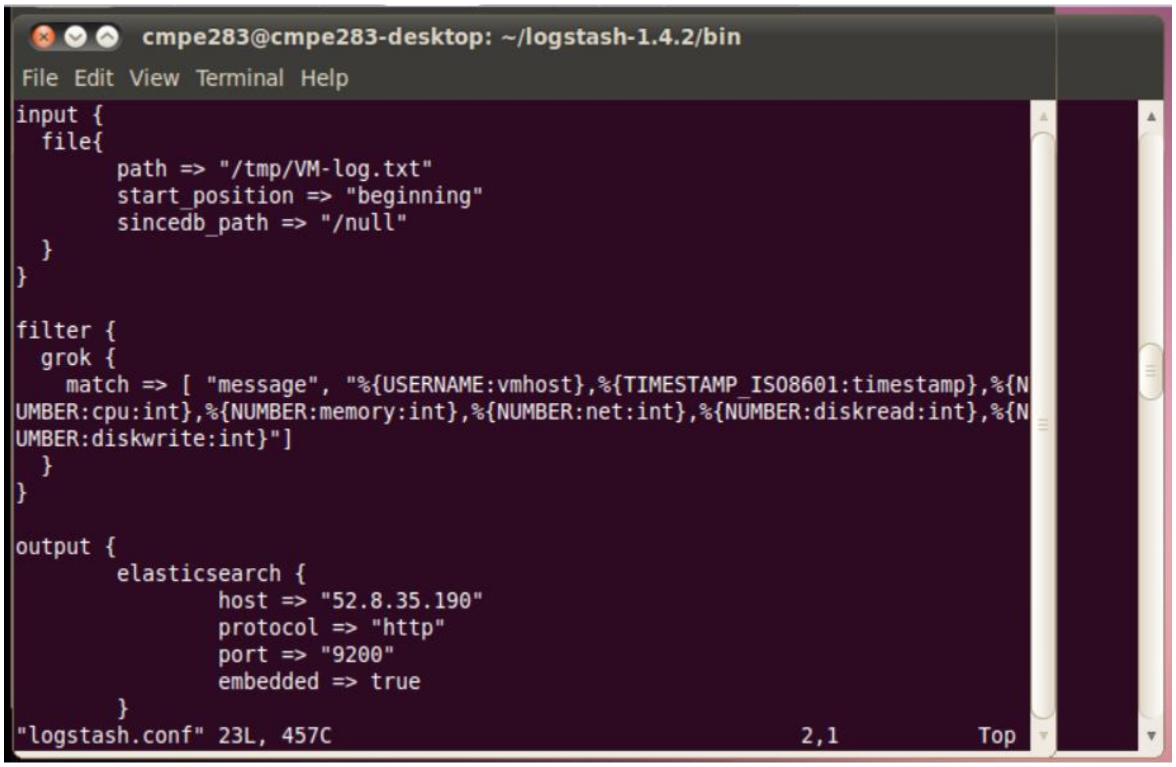
Logstash is used to collect and parse the logs and load it into the ElasticSearch DB. Logstash requires the installation of java in the system as a pre-requisite. Below commands are executed:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-installer
sudo update-java-alternatives -s java-7-oracle
```

Later, logstash is installed by executing the commands:

```
curl -O https://download.elasticsearch.org/logstash/logstash/logstash-1.4.2.tar.gz
tar zxvf logstash-1.4.2.tar.gz
cd logstash-1.4.2
```

Now, the logstash is configured to read from the input log file, filter the logs into a formatted data and later output into the Elasticsearch DB server. Below screenshot shows the logstash conf file configured:


cmpe283@cmpe283-desktop: ~/logstash-1.4.2/bin

File Edit View Terminal Help

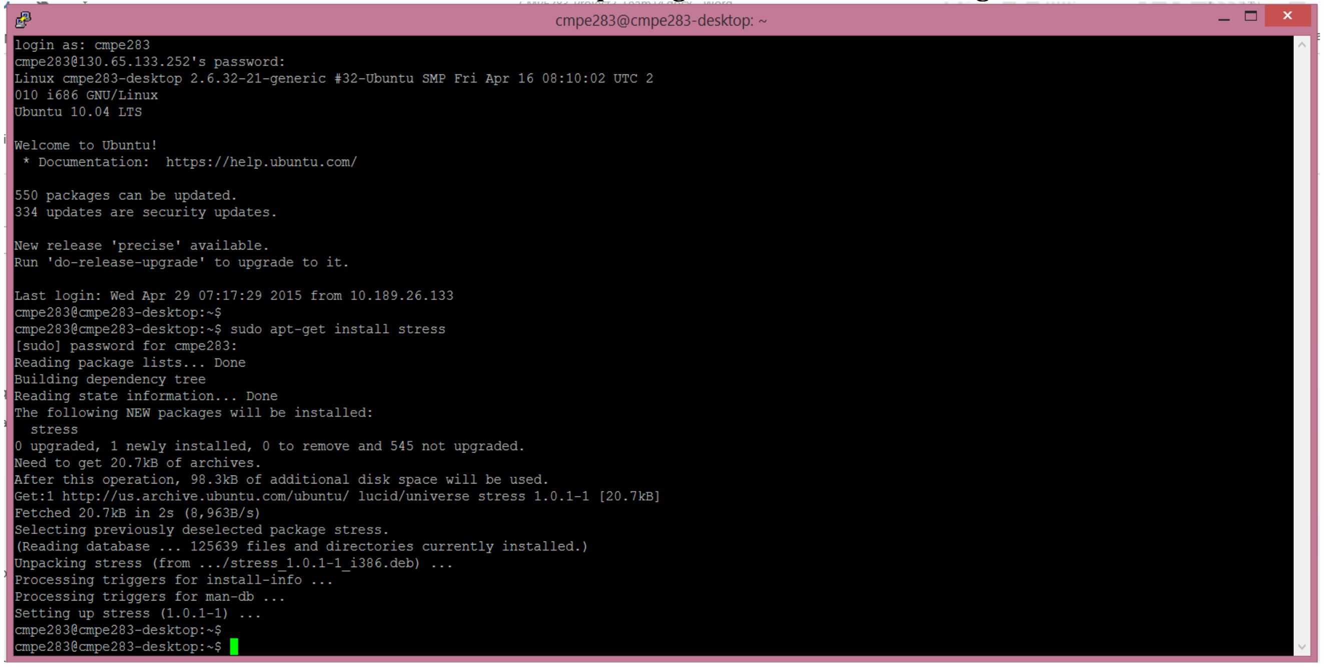
```
input {
  file{
        path => "/tmp/VM-log.txt"
        start_position => "beginning"
        sincedb_path => "/null"
  }
}

filter {
  grok {
    match => [ "message", "%{USERNAME:vmhost},%{TIMESTAMP_ISO8601:timestamp},%{N
UMBER:cpu:int},%{NUMBER:memory:int},%{NUMBER:net:int},%{NUMBER:diskread:int},%{N
UMBER:diskwrite:int}"]
  }
}

output {
        elasticsearch {
                host => "52.8.35.190"
                protocol => "http"
                port => "9200"
                embedded => true
        }
"logstash.conf" 23L, 457C                                     2,1           Top
```

ElasticSearch and Kibana installation:

An AWS instance with ElasticSearch & Kibana is created as part of this project.

Stress testing tool installation:

Stress tool is installed on each of the VM requiring the CPU stress testing.


cmpe283@cmpe283-desktop: ~

```
login as: cmpe283
cmpe283@130.65.133.252's password:
Linux cmpe283-desktop 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2
010 i686 GNU/Linux
Ubuntu 10.04 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

550 packages can be updated.
334 updates are security updates.

New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Apr 29 07:17:29 2015 from 10.189.26.133
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$ sudo apt-get install stress
[sudo] password for cmpe283:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 545 not upgraded.
Need to get 20.7kB of archives.
After this operation, 98.3kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ lucid/universe stress 1.0.1-1 [20.7kB]
Fetched 20.7kB in 2s (8,963B/s)
Selecting previously deselected package stress.
(Reading database ... 125639 files and directories currently installed.)
Unpacking stress (from .../stress_1.0.1-1_i386.deb) ...
Processing triggers for install-info ...
Processing triggers for man-db ...
Setting up stress (1.0.1-1) ...
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$
```

11. SYSTEM TESTING RESULTS:

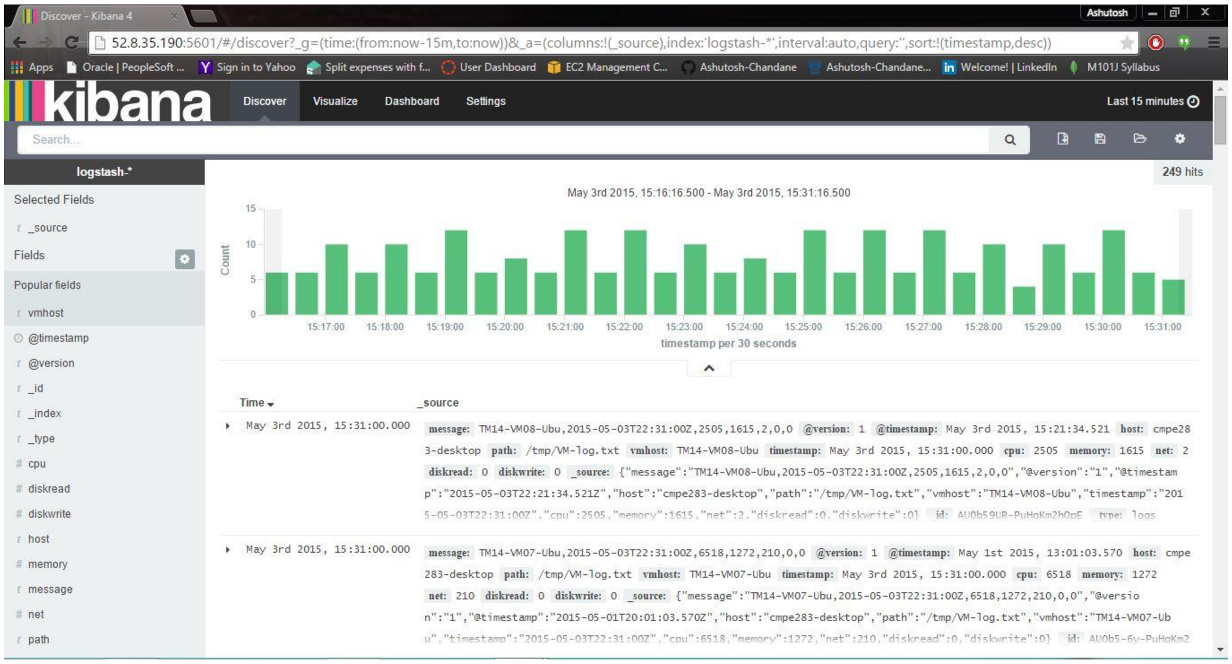
11.1 Test Cases:

Test Case Id	Test case Description	Expected Result	Actual Result	
			Pass	Fail
1	When Virtual machine is powered ON, it should return the Virtual Machine name, its corresponding host name and the performance statistics of both virtual machine and its corresponding host should be saved in a log file at specified location.	The Statistics of the vHosts and the VMs are successfully fetched.	Yes	
2	To check the connection with the ElasticSearch and Storage of Data in ElasticSearch. Run the command to execute the Logstash. Data from specified input path should be parsed to MongoDB.	Successfully connection should be established with ElasticSearch and data should be parsed from specified location and saved to ElasticSearch.	Yes	
3	Test remote connectivity with the ElasticSearch server for Aggregator so that the Aggregator program could remote connect.	Remote connection should is successful ElasticSearch tool.	Yes	

4	Test case to ensures that the aggregator is able is connect to ElasticSearch and insert aggregated statistics in database.	The data from ElasticSearch is aggregated and pushed.	Yes	
5	Visualization			
5.1	To test if the UI REST web services are able to connect with the deployed PHP scripts and get back the data.	There should be no error on loading the VMware DevOps dashboard and data should be fetched for the charts.	Yes	
5.2	To test data consistency between the ElasticSearch data and the data populated into visualization widgets.	The data points on the charts and table are correct an as expected.	Yes	
5.3	Test Case to check if the graph and data tables are auto refreshing after every fixed interval of time.	The values in the graphs are updated after a configured time interval.	Yes	

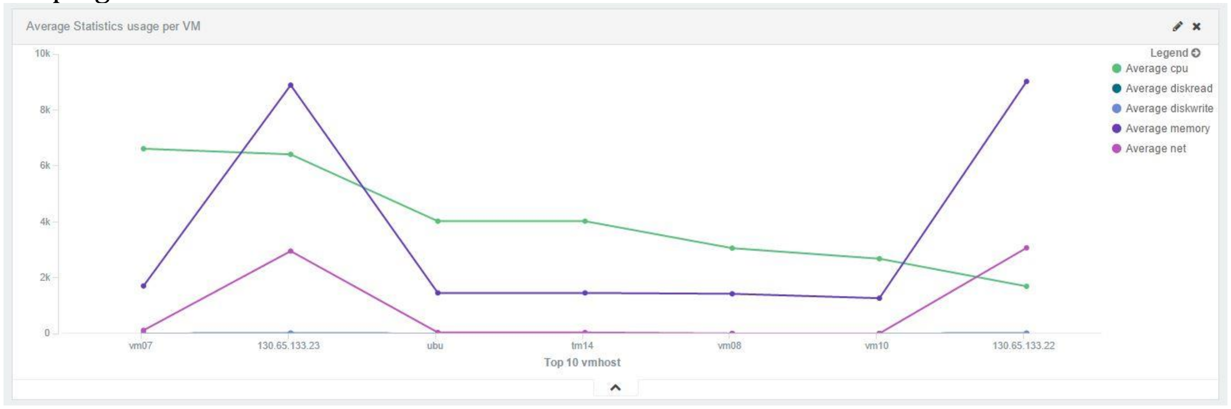
11.2 Test Results Screenshots:

Search results of a VM –

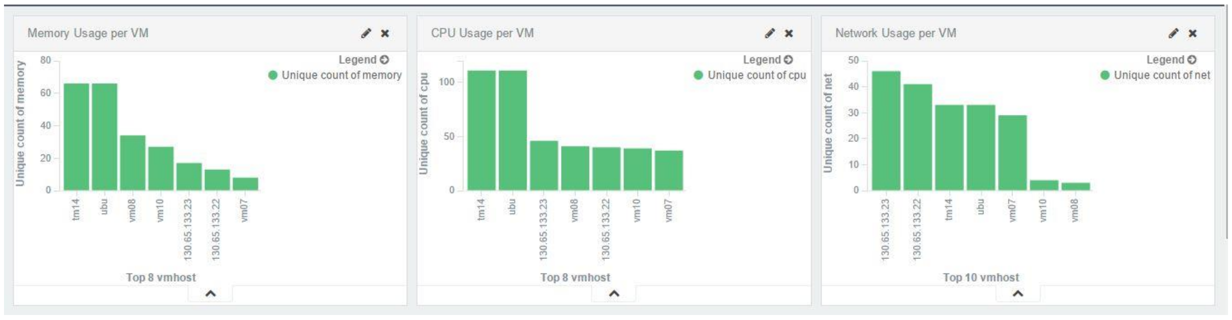


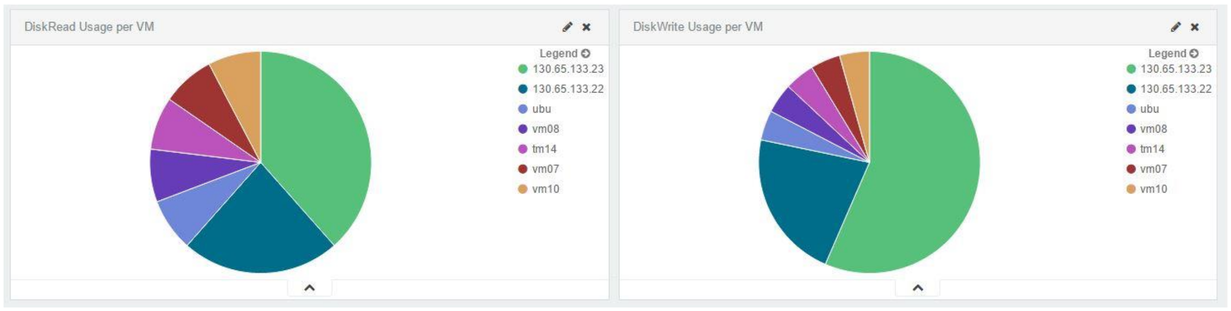
```
Time    _source
May 3rd 2015, 15:31:00.000   message: TM14-VM08-Ubu,2015-05-03T22:31:00Z,2505,1615,2,0,0 @version: 1 @timestamp: May 3rd 2015, 15:21:34.521 host: cmpe283-desktop path: /tmp/VM-log.txt vmhost: TM14-VM08-Ubu timestamp: May 3rd 2015, 15:31:00.000 cpu: 2505 memory: 1615 net: 2 diskread: 0 diskwrite: 0 _source: {"message":"TM14-VM08-Ubu,2015-05-03T22:31:00Z,2505,1615,2,0,0","@version":"1","@timestamp":"2015-05-03T22:21:34.521Z","host":"cmpe283-desktop","path":"/tmp/VM-log.txt","vmhost":"TM14-VM08-Ubu","timestamp":"2015-05-03T22:31:00Z","cpu":2505,"memory":1615,"net":2,"diskread":0,"diskwrite":0} _id: AU0b59UR-PuHqKm2hOpE _type: logs
May 3rd 2015, 15:31:00.000   message: TM14-VM07-Ubu,2015-05-03T22:31:00Z,6518,1272,210,0,0 @version: 1 @timestamp: May 1st 2015, 13:01:03.570 host: cmpe283-desktop path: /tmp/VM-log.txt vmhost: TM14-VM07-Ubu timestamp: May 3rd 2015, 15:31:00.000 cpu: 6518 memory: 1272 net: 210 diskread: 0 diskwrite: 0 _source: {"message":"TM14-VM07-Ubu,2015-05-03T22:31:00Z,6518,1272,210,0,0","@version":"1","@timestamp":"2015-05-01T20:01:03.570Z","host":"cmpe283-desktop","path":"/tmp/VM-log.txt","vmhost":"TM14-VM07-Ubu","timestamp":"2015-05-03T22:31:00Z","cpu":6518,"memory":1272,"net":210,"diskread":0,"diskwrite":0} _id: AU0b5-6y-PuHqKm2
```

Graph generated for Statistics of a VM -

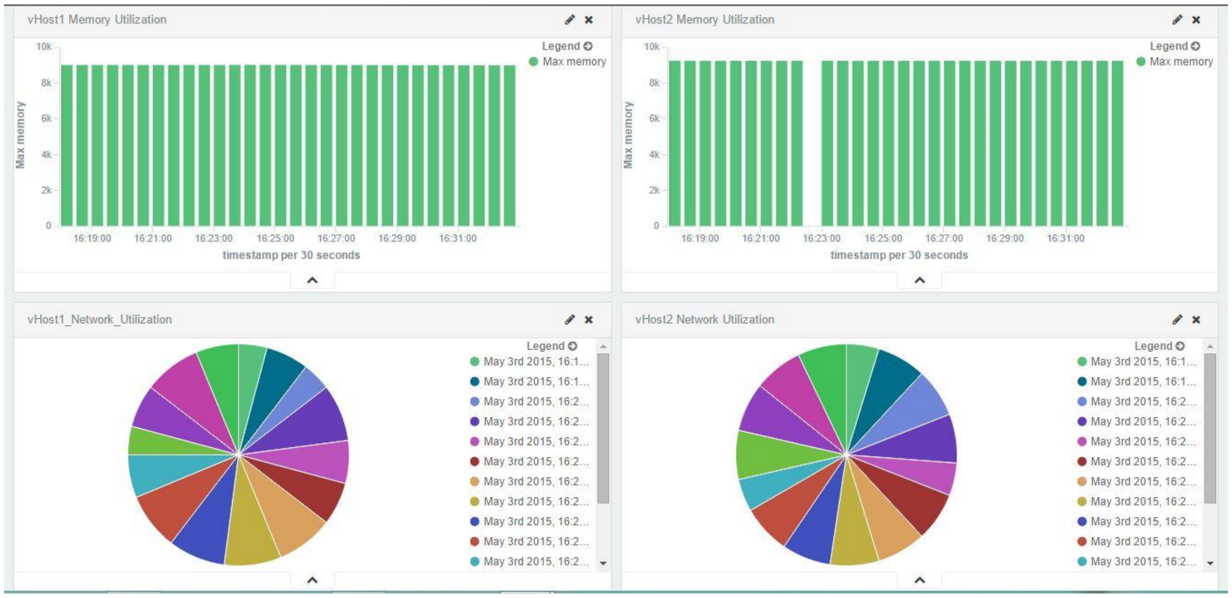


Graphs for comparison of different parameters -

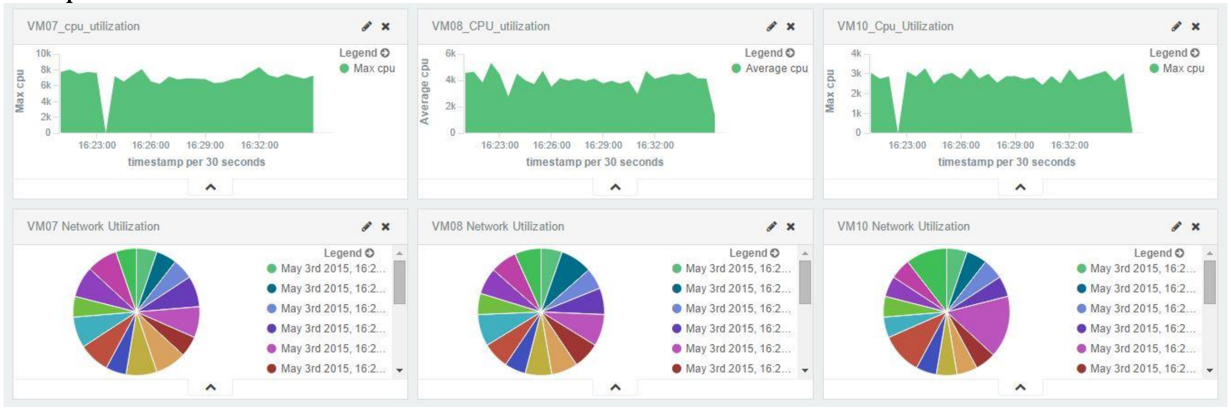




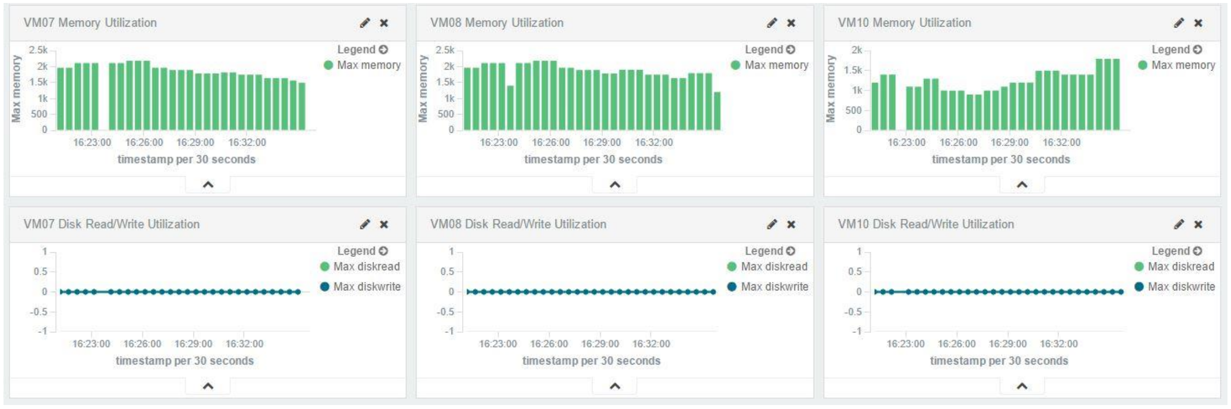
Comparison of vHost1 and vHost2 Memory utilization -



Comparison of different VMs CPU utilization -



Comparison of different VMs Memory utilization -



12. TESTING

12.1 Unit Testing:

Individual modules of the project like Logstash, ElasticSearch and Kibana were tested. First, Logstash was developed independent of the other modules and made sure they were working as expected. ElasticSearch and Kibana were tested in the same manner.

12.2 Integration Testing:

Different modules of the project like Logstash, ElasticSearch and Kibana were combined and tested together as a group.

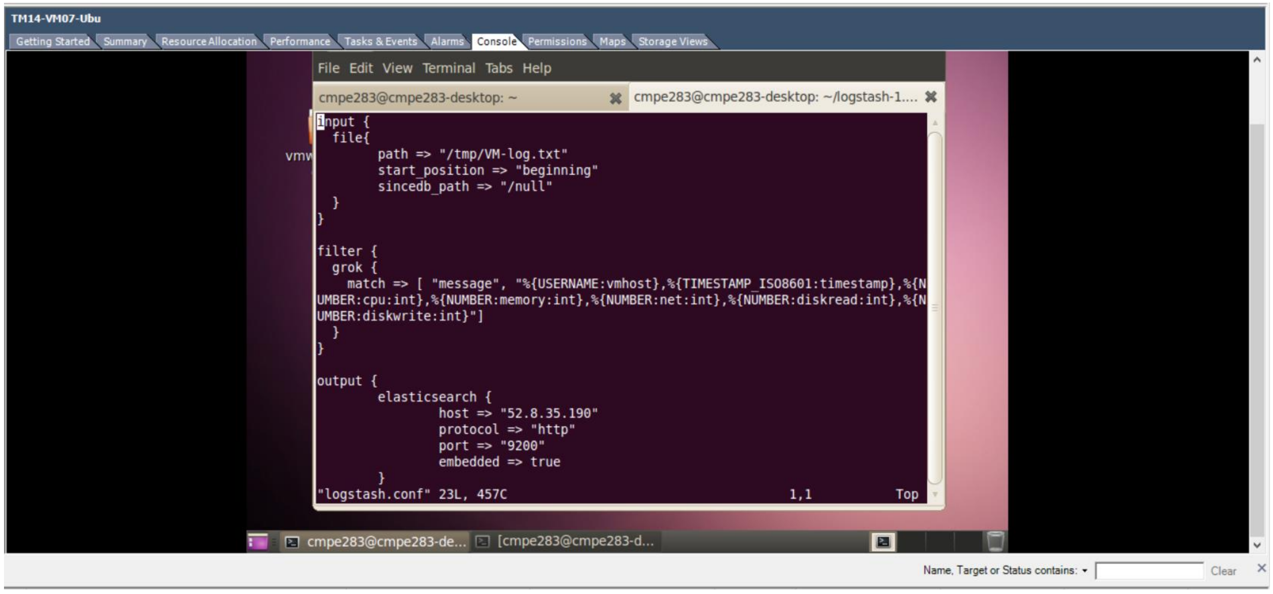
12.3 Individual Testing Contribution:

Name	Testing Contribution
Aravind Ganesh	Unit Testing
Ashutosh Chandane	Kibana Testing
Neethi Srinivas Murthy	Integration Testing
Sudhakar Kamanboina	System level Testing
Swetha Rathna Kempahanumaiah	Logstash and Stress testing scenarios

13. SCREENSHOTS:

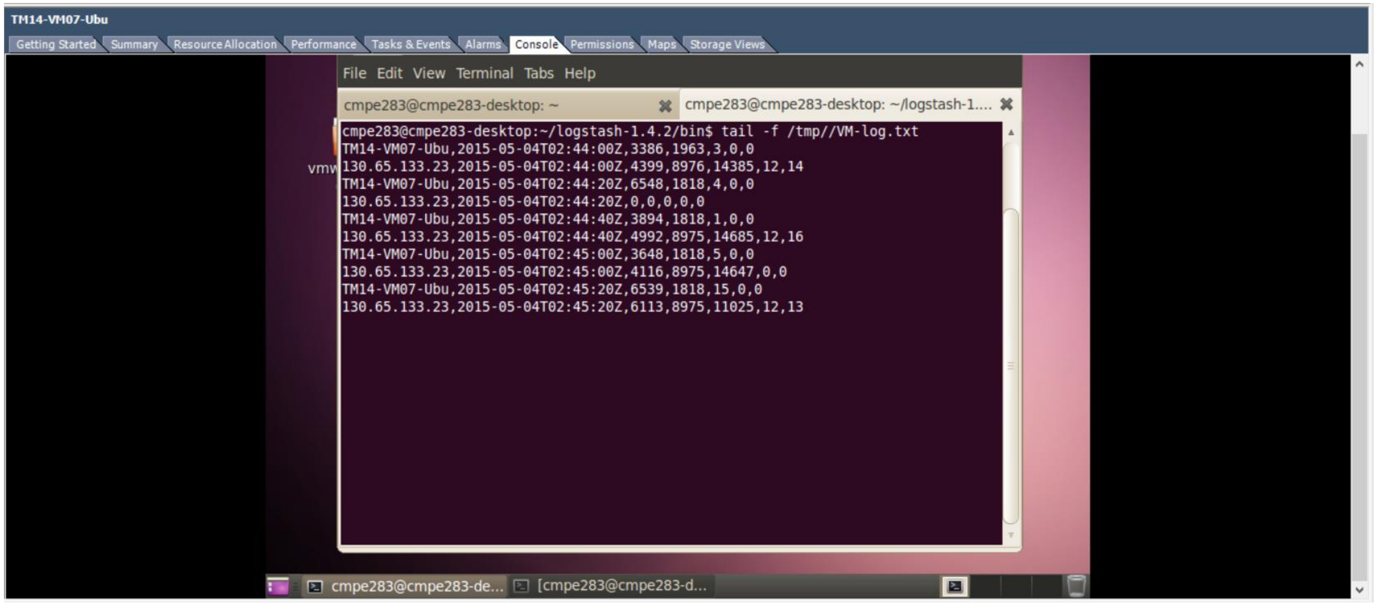
Logstash screenshots:

Logstash.conf file configuration in the VM:



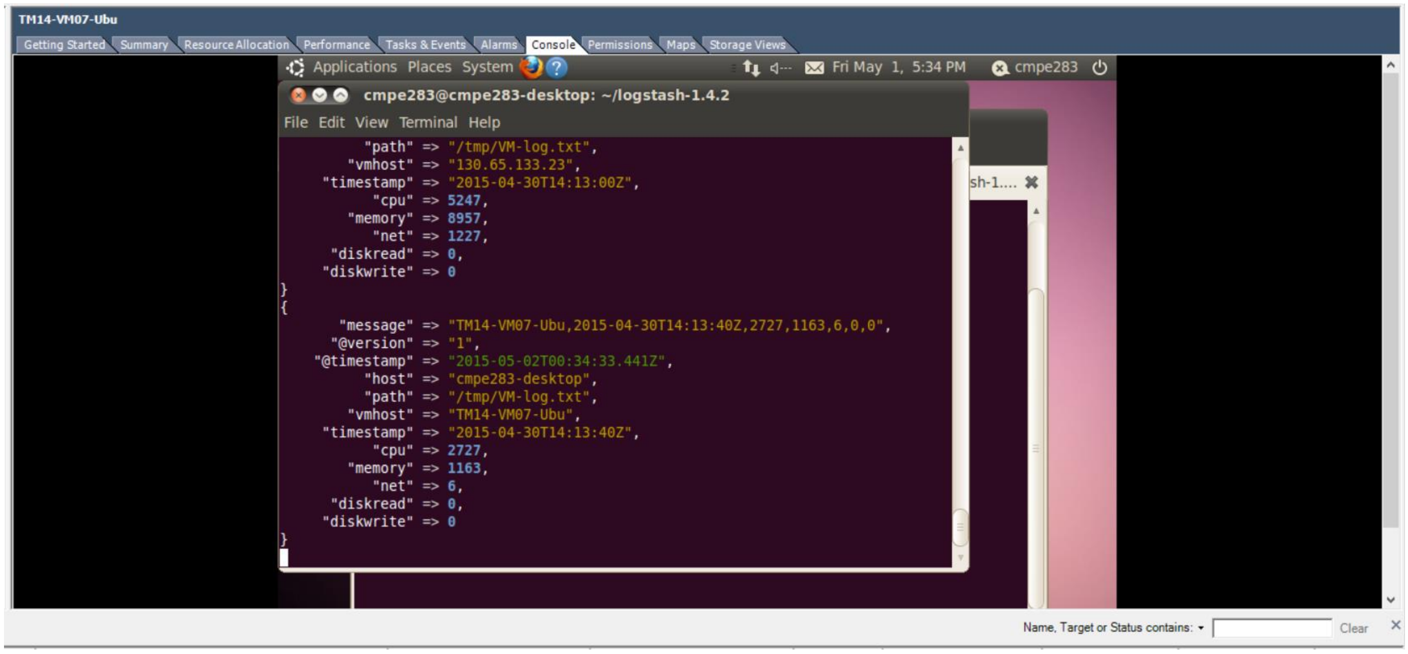
```
input {
  file{
        path => "/tmp/VM-log.txt"
        start_position => "beginning"
        sincedb_path => "/null"
  }
}

filter {
  grok {
    match => [ "message", "%{USERNAME:vmhost},%{TIMESTAMP_ISO8601:timestamp},%{N
UMBER:cpu:int},%{NUMBER:memory:int},%{NUMBER:net:int},%{NUMBER:diskread:int},%{N
UMBER:diskwrite:int}"]
  }
}

output {
        elasticsearch {
                host => "52.8.35.190"
                protocol => "http"
                port => "9200"
                embedded => true
        }
"logstash.conf" 23L, 457C                                     1,1           Top
```

Log file generated by the Performance Manager application with VM and VHost statistical information:



```
cmpe283@cmpe283-desktop:~/logstash-1.4.2/bin$ tail -f /tmp//VM-log.txt
TM14-VM07-Ubu,2015-05-04T02:44:00Z,3386,1963,3,0,0
130.65.133.23,2015-05-04T02:44:00Z,4399,8976,14385,12,14
TM14-VM07-Ubu,2015-05-04T02:44:20Z,6548,1818,4,0,0
130.65.133.23,2015-05-04T02:44:20Z,0,0,0,0,0
TM14-VM07-Ubu,2015-05-04T02:44:40Z,3894,1818,1,0,0
130.65.133.23,2015-05-04T02:44:40Z,4992,8975,14685,12,16
TM14-VM07-Ubu,2015-05-04T02:45:00Z,3648,1818,5,0,0
130.65.133.23,2015-05-04T02:45:00Z,4116,8975,14647,0,0
TM14-VM07-Ubu,2015-05-04T02:45:20Z,6539,1818,15,0,0
130.65.133.23,2015-05-04T02:45:20Z,6113,8975,11025,12,13
```

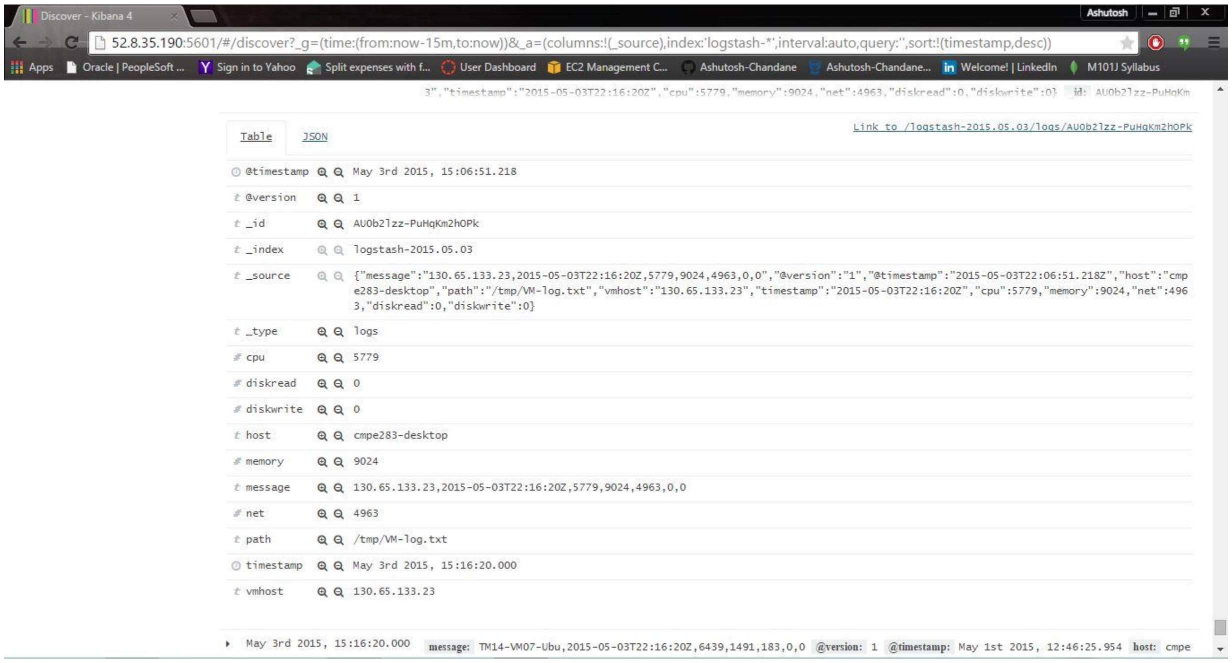
Logstash execution:



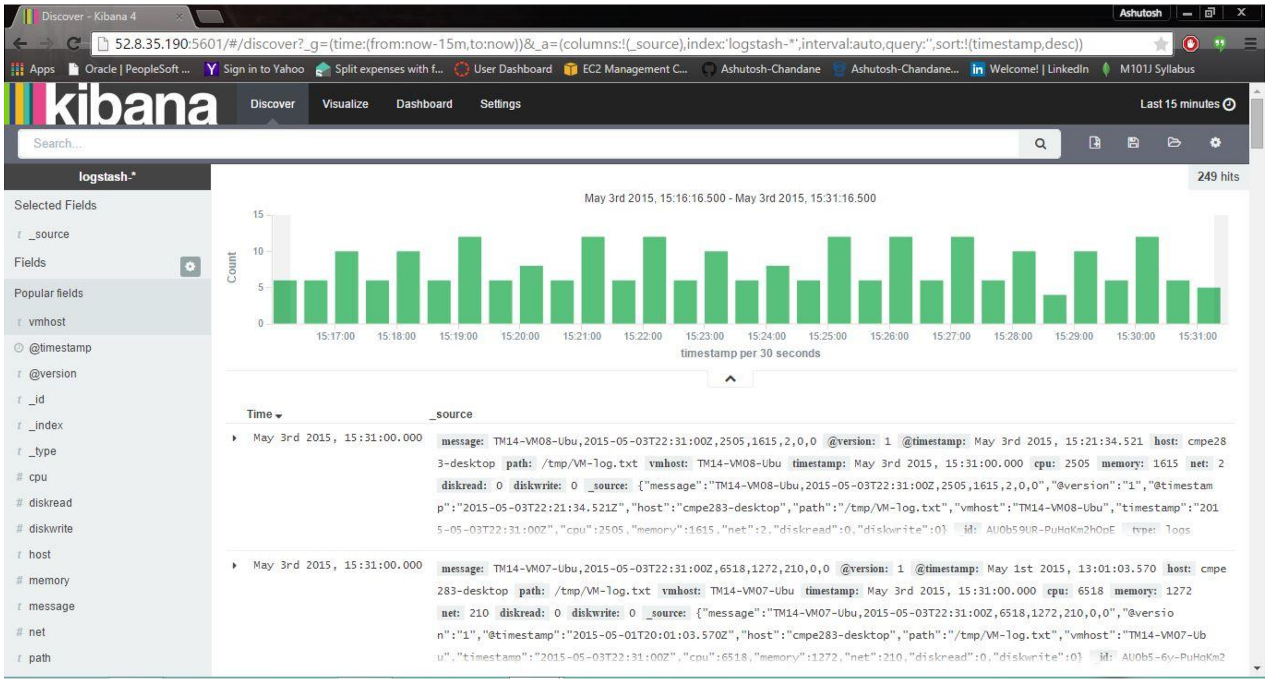
```
        "path" => "/tmp/VM-log.txt",
      "vmhost" => "130.65.133.23",
   "timestamp" => "2015-04-30T14:13:00Z",
         "cpu" => 5247,
      "memory" => 8957,
         "net" => 1227,
    "diskread" => 0,
   "diskwrite" => 0
}
{
     "message" => "TM14-VM07-Ubu,2015-04-30T14:13:40Z,2727,1163,6,0,0",
    "@version" => "1",
  "@timestamp" => "2015-05-02T00:34:33.441Z",
        "host" => "cmpe283-desktop",
        "path" => "/tmp/VM-log.txt",
      "vmhost" => "TM14-VM07-Ubu",
   "timestamp" => "2015-04-30T14:13:40Z",
         "cpu" => 2727,
      "memory" => 1163,
         "net" => 6,
    "diskread" => 0,
   "diskwrite" => 0
}
```

Kibana screenshots:

Log information collected in logstash from Kibana

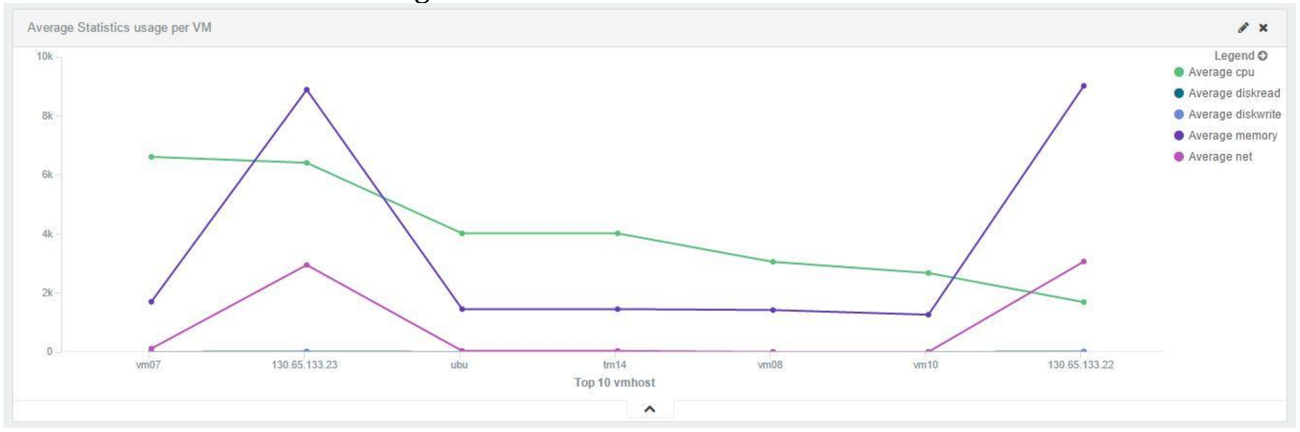


```
52.8.35.190:5601/#/discover?_g=(time:(from:now-15m,to:now))&_a=(columns:!(_source),index:'logstash-*',interval:auto,query:'',sort:!(timestamp,desc))

3","timestamp":"2015-05-03T22:16:20Z","cpu":5779,"memory":9024,"net":4963,"diskread":0,"diskwrite":0}  _id: AUOb2lzz-PuHqKm

Table   JSON                                         Link to /logstash-2015.05.03/logs/AUOb2lzz-PuHqKm2hOPk

@timestamp   May 3rd 2015, 15:06:51.218
@version     1
_id          AUOb2lzz-PuHqKm2hOPk
_index       logstash-2015.05.03
_source      {"message":"130.65.133.23,2015-05-03T22:16:20Z,5779,9024,4963,0,0","@version":"1","@timestamp":"2015-05-03T22:06:51.218Z","host":"cmp
             e283-desktop","path":"/tmp/VM-log.txt","vmhost":"130.65.133.23","timestamp":"2015-05-03T22:16:20Z","cpu":5779,"memory":9024,"net":496
             3,"diskread":0,"diskwrite":0}
_type        logs
cpu          5779
diskread     0
diskwrite    0
host         cmpe283-desktop
memory       9024
message      130.65.133.23,2015-05-03T22:16:20Z,5779,9024,4963,0,0
net          4963
path         /tmp/VM-log.txt
timestamp    May 3rd 2015, 15:16:20.000
vmhost       130.65.133.23

May 3rd 2015, 15:16:20.000   message: TM14-VM07-Ubu,2015-05-03T22:16:20Z,6439,1491,183,0,0  @version: 1  @timestamp: May 1st 2015, 12:46:25.954  host: cmpe
```

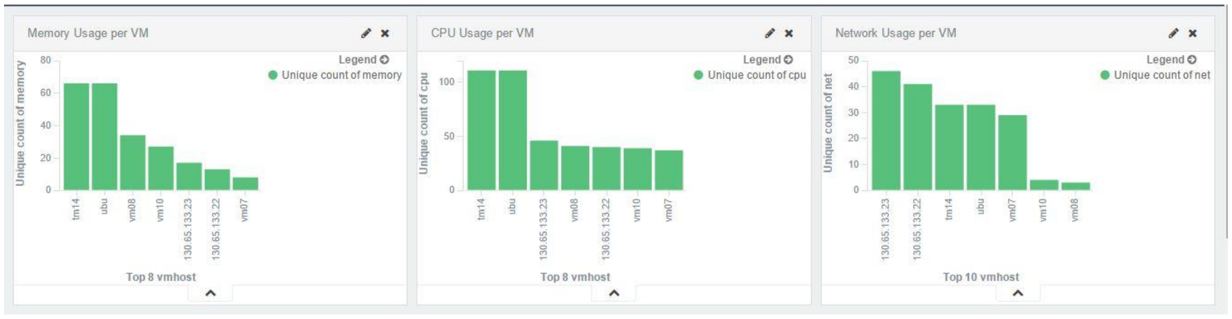


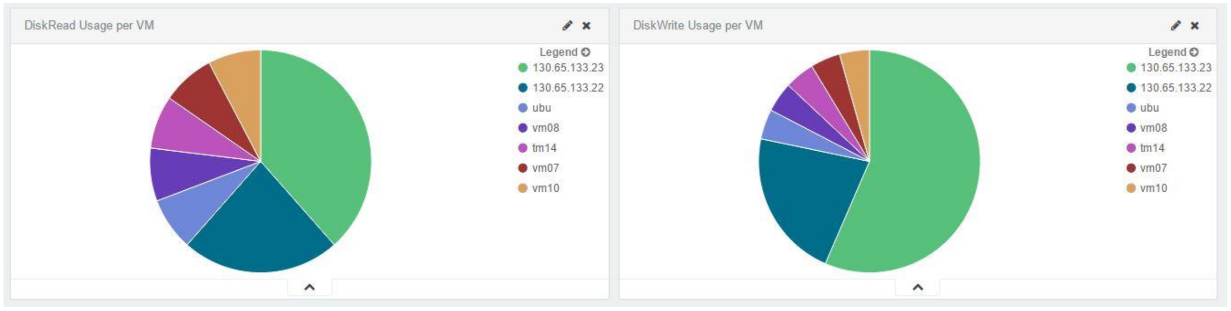
```
52.8.35.190:5601/#/discover?_g=(time:(from:now-15m,to:now))&_a=(columns:!(_source),index:'logstash-*',interval:auto,query:'',sort:!(timestamp,desc))

Time                         _source
May 3rd 2015, 15:31:00.000   message: TM14-VM08-Ubu,2015-05-03T22:31:00Z,2505,1615,2,0,0  @version: 1  @timestamp: May 3rd 2015, 15:21:34.521  host: cmpe28
                             3-desktop  path: /tmp/VM-log.txt  vmhost: TM14-VM08-Ubu  timestamp: May 3rd 2015, 15:31:00.000  cpu: 2505  memory: 1615  net: 2
                             diskread: 0  diskwrite: 0  _source: {"message":"TM14-VM08-Ubu,2015-05-03T22:31:00Z,2505,1615,2,0,0","@version":"1","@timestam
                             p":"2015-05-03T22:21:34.521Z","host":"cmpe283-desktop","path":"/tmp/VM-log.txt","vmhost":"TM14-VM08-Ubu","timestamp":"201
                             5-05-03T22:31:00Z","cpu":2505,"memory":1615,"net":2,"diskread":0,"diskwrite":0}  _id: AUOb59UR-PuHqKm2hOpE  _type: logs
May 3rd 2015, 15:31:00.000   message: TM14-VM07-Ubu,2015-05-03T22:31:00Z,6518,1272,210,0,0  @version: 1  @timestamp: May 1st 2015, 13:01:03.570  host: cmpe
                             283-desktop  path: /tmp/VM-log.txt  vmhost: TM14-VM07-Ubu  timestamp: May 3rd 2015, 15:31:00.000  cpu: 6518  memory: 1272
                             net: 210  diskread: 0  diskwrite: 0  _source: {"message":"TM14-VM07-Ubu,2015-05-03T22:31:00Z,6518,1272,210,0,0","@versio
                             n":"1","@timestamp":"2015-05-01T20:01:03.570Z","host":"cmpe283-desktop","path":"/tmp/VM-log.txt","vmhost":"TM14-VM07-Ub
                             u","timestamp":"2015-05-03T22:31:00Z","cpu":6518,"memory":1272,"net":210,"diskread":0,"diskwrite":0}  _id: AUOb5-6y-PuHqKm2
```

Statistics for all the VMs average metrics

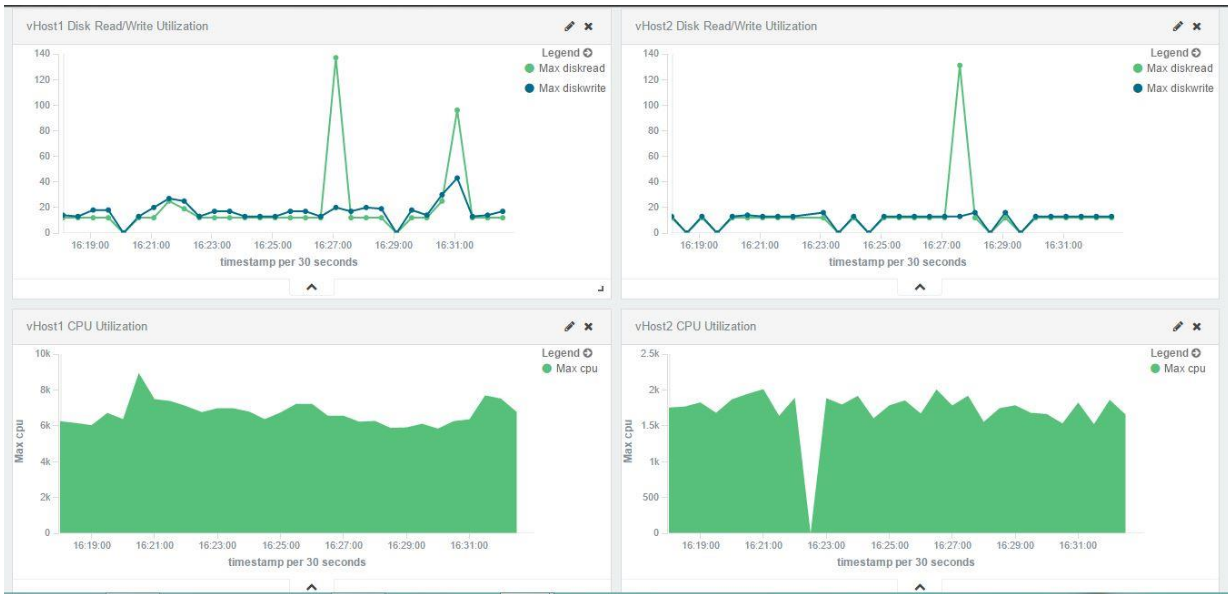


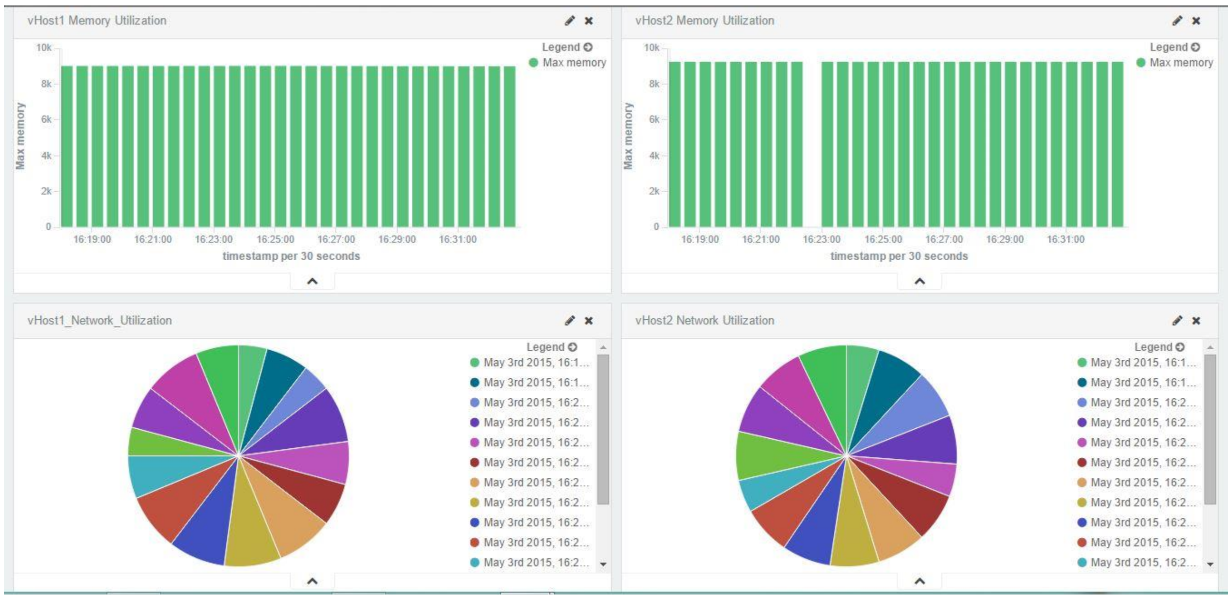
Per VM statistical data display



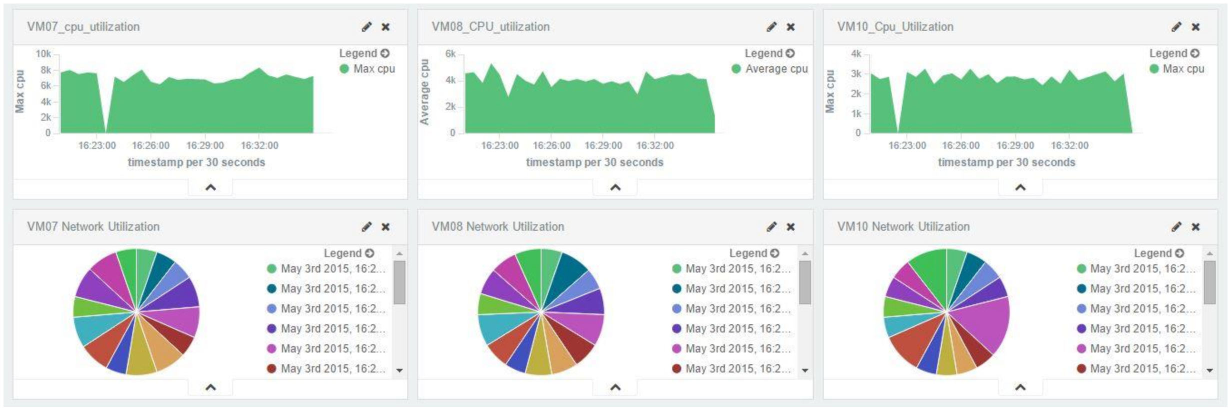


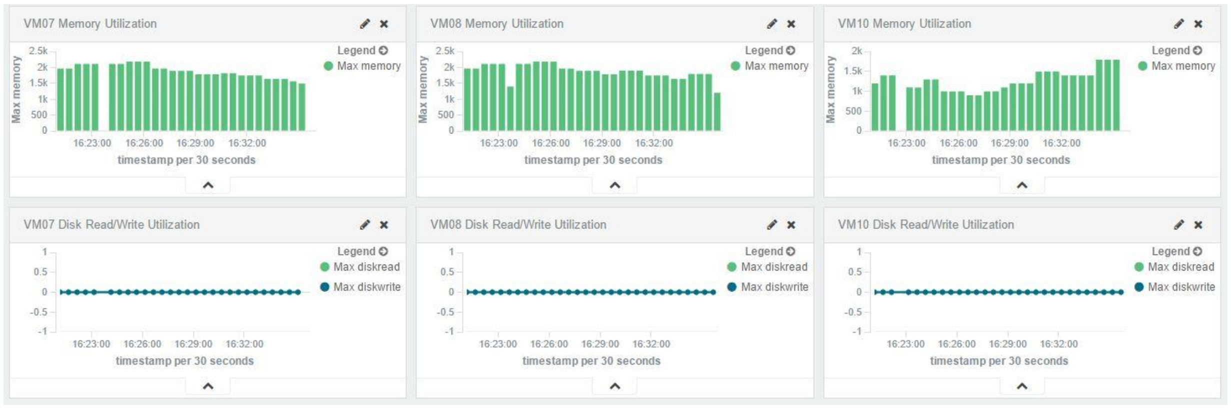
Comparison of different parameters of vHost1 and vHost2 -





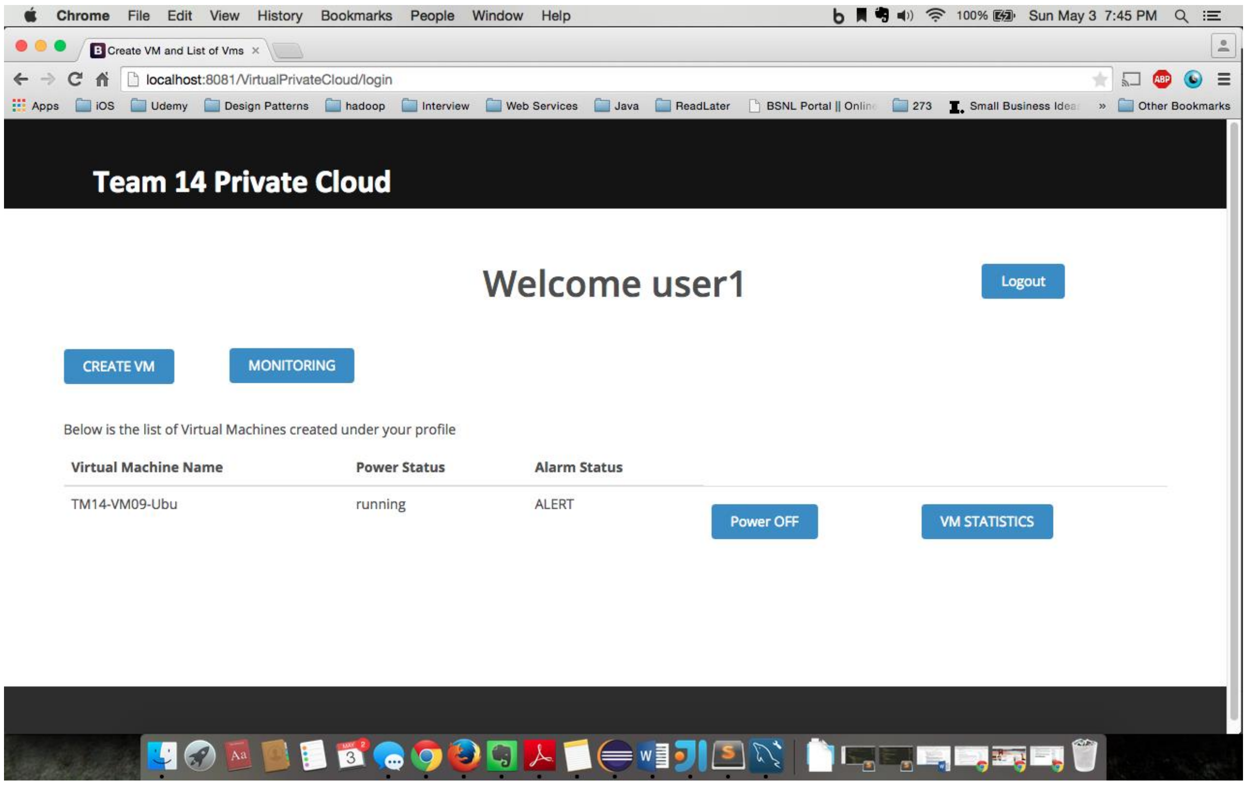
Aggregated data of all VMs:





Web UI application screenshots:

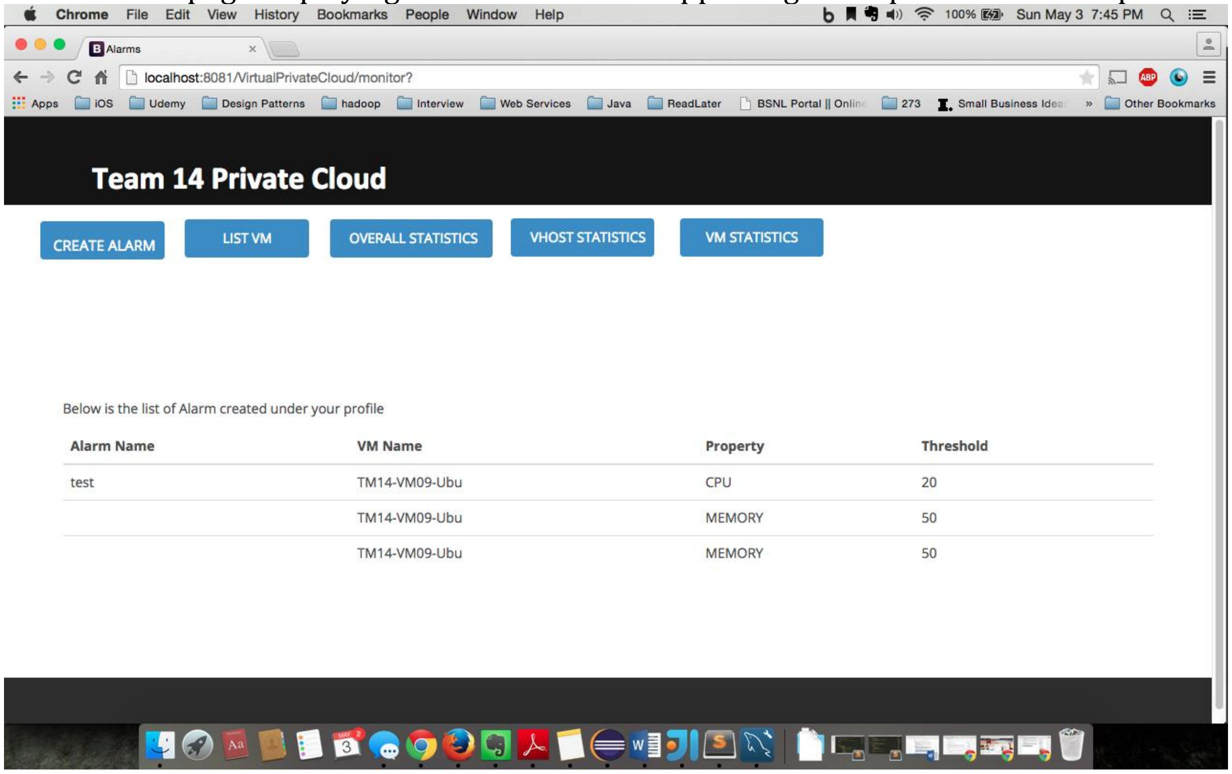
List VM web page displaying alarm status of each VM


Chrome File Edit View History Bookmarks People Window Help
Sun May 3 7:45 PM
Create VM and List of Vms
localhost:8081/VirtualPrivateCloud/login
Team 14 Private Cloud
Welcome user1
Logout
CREATE VM
MONITORING
Power OFF
VM STATISTICS

Below is the list of Virtual Machines created under your profile

Virtual Machine Name	Power Status	Alarm Status
TM14-VM09-Ubu	running	ALERT

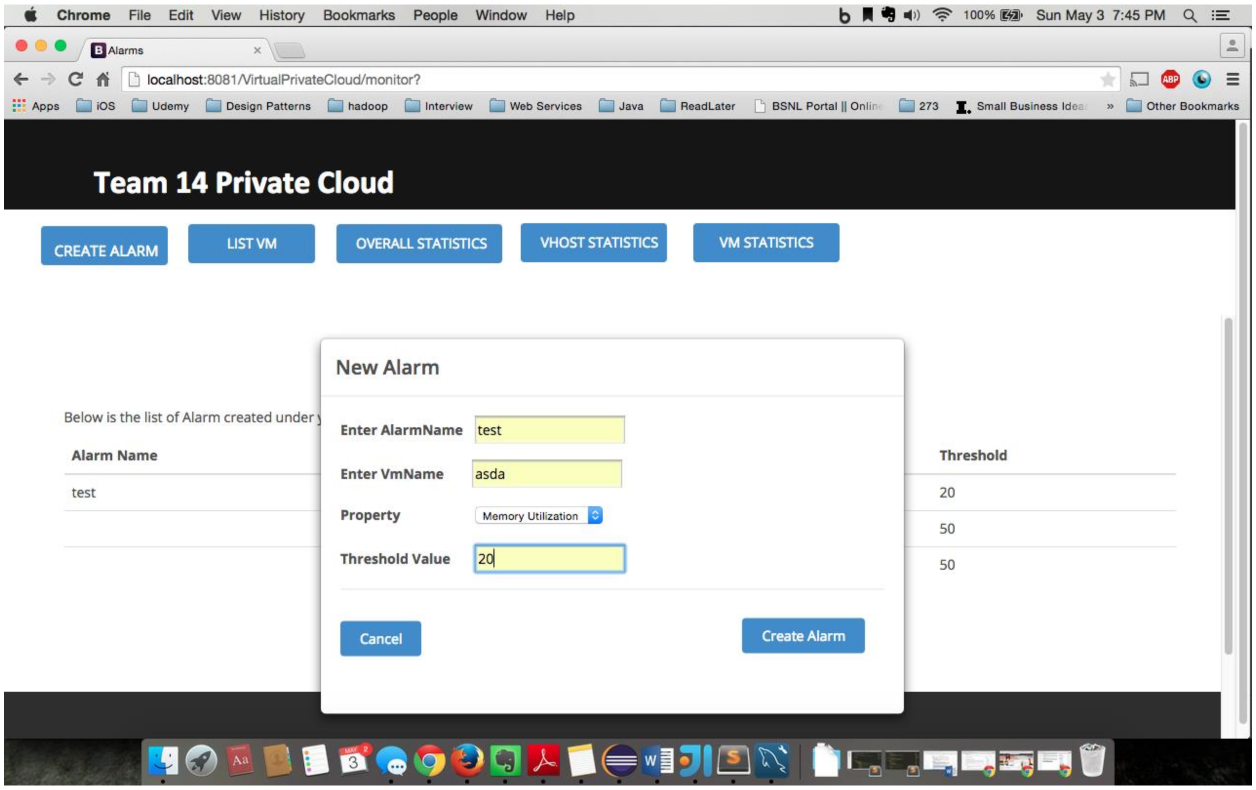
Monitor web page displaying list of alarms and supporting new options to view report



Below is the list of Alarm created under your profile

Alarm Name	VM Name	Property	Threshold
test	TM14-VM09-Ubu	CPU	20
	TM14-VM09-Ubu	MEMORY	50
	TM14-VM09-Ubu	MEMORY	50

Create Alarm window



Team 14 Private Cloud

CREATE ALARM | LIST VM | OVERALL STATISTICS | VHOST STATISTICS | VM STATISTICS

Below is the list of Alarm created under y

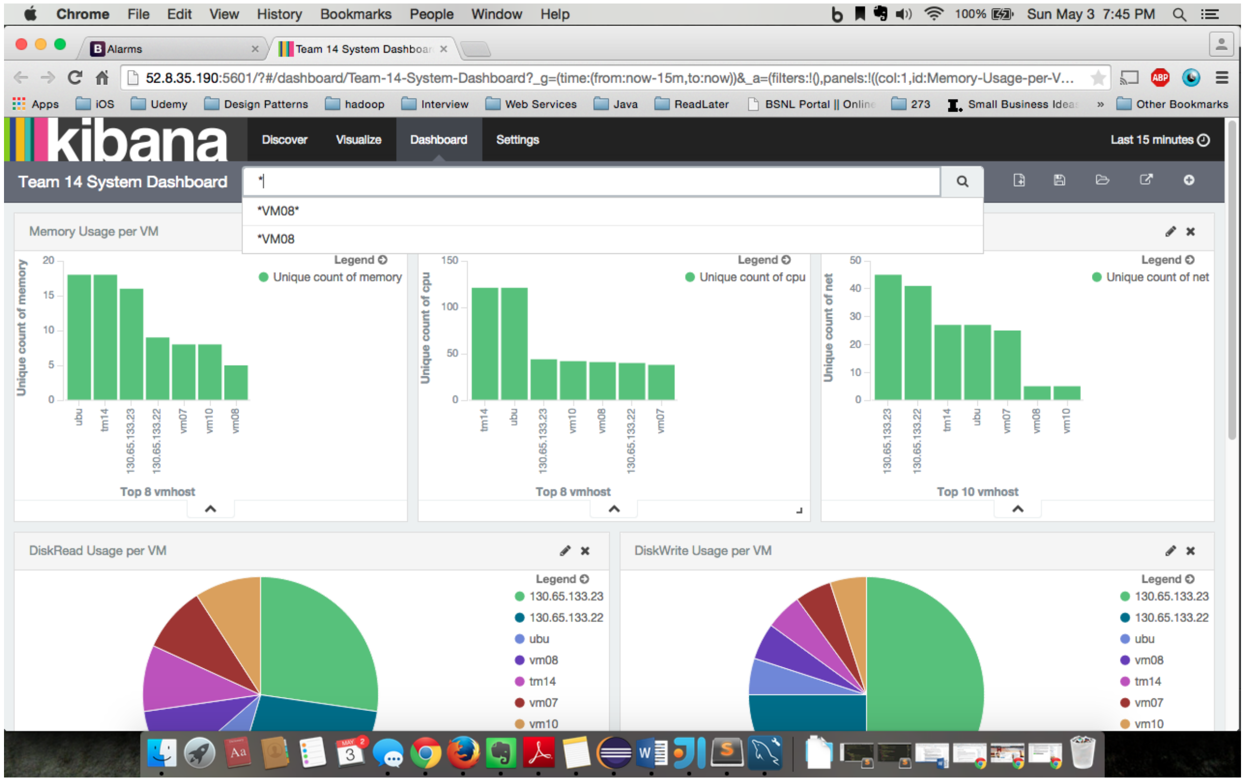
Alarm Name	Threshold
test	20
	50
	50

New Alarm

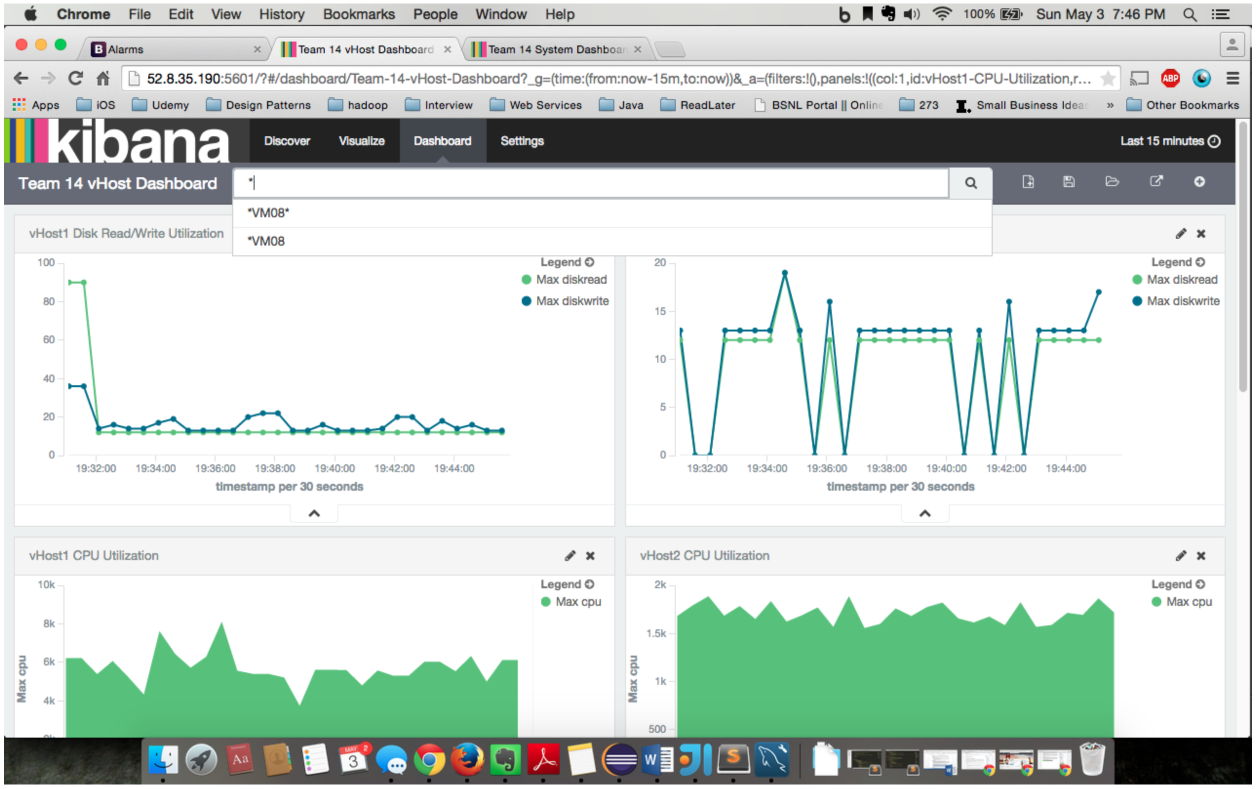
Enter AlarmName	test
Enter VmName	asda
Property	Memory Utilization
Threshold Value	20

Cancel | Create Alarm

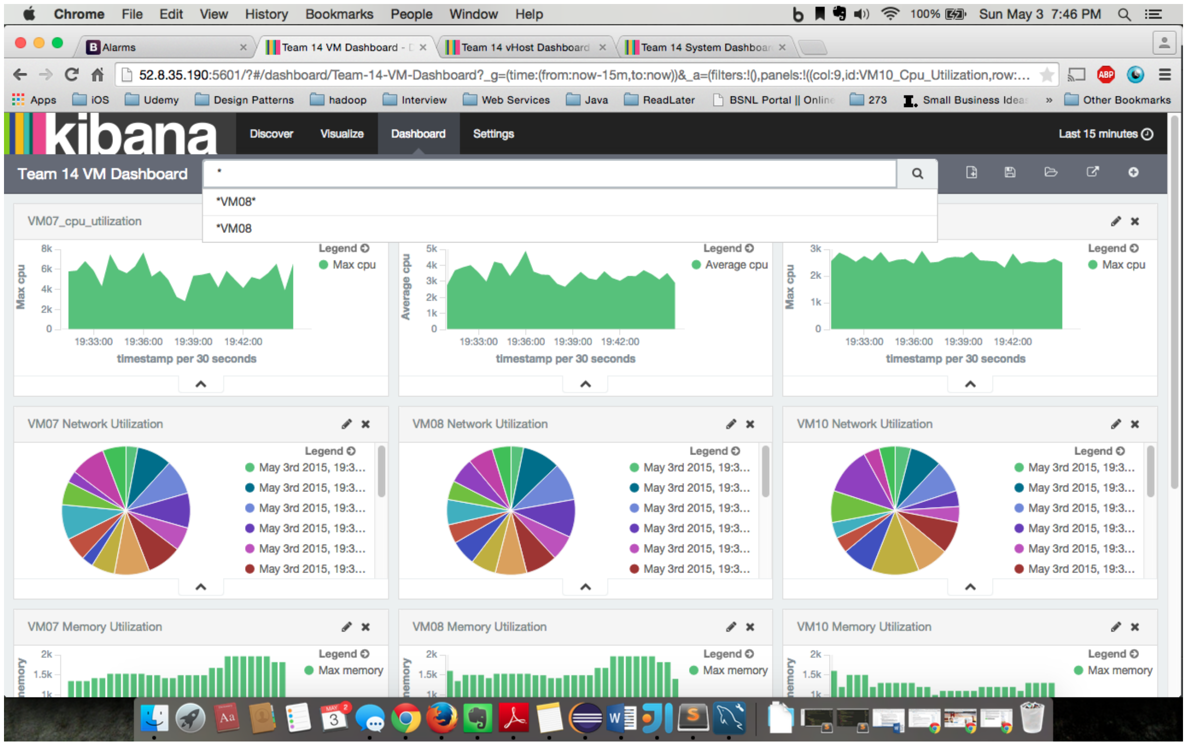
Overall Statistics implementation



VHost statistics displayed on "VHost Statistics" button

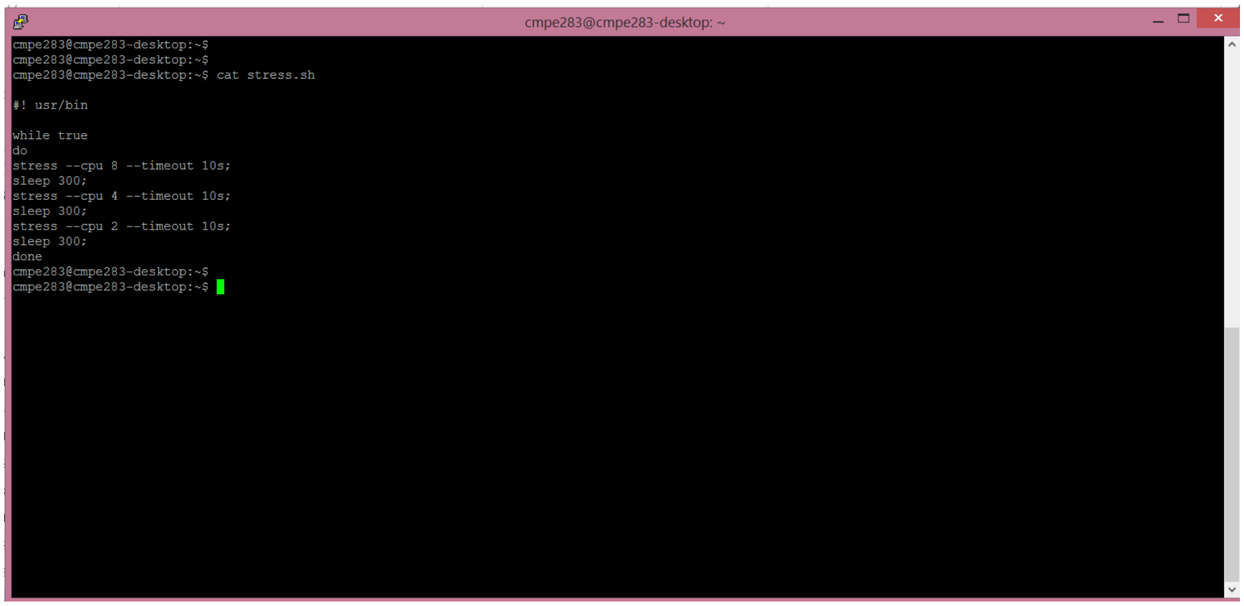


VM statistics displayed on "VM Statistics" button



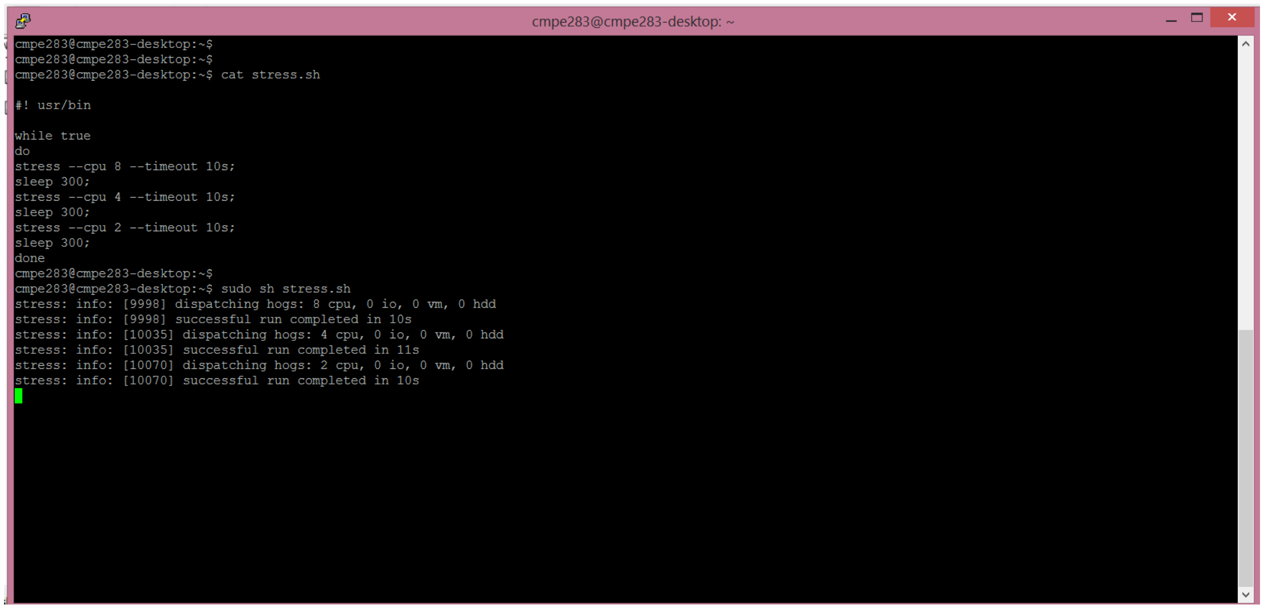
Stress testing with maximum CPU load:

Script to load the CPU on a periodic interval:

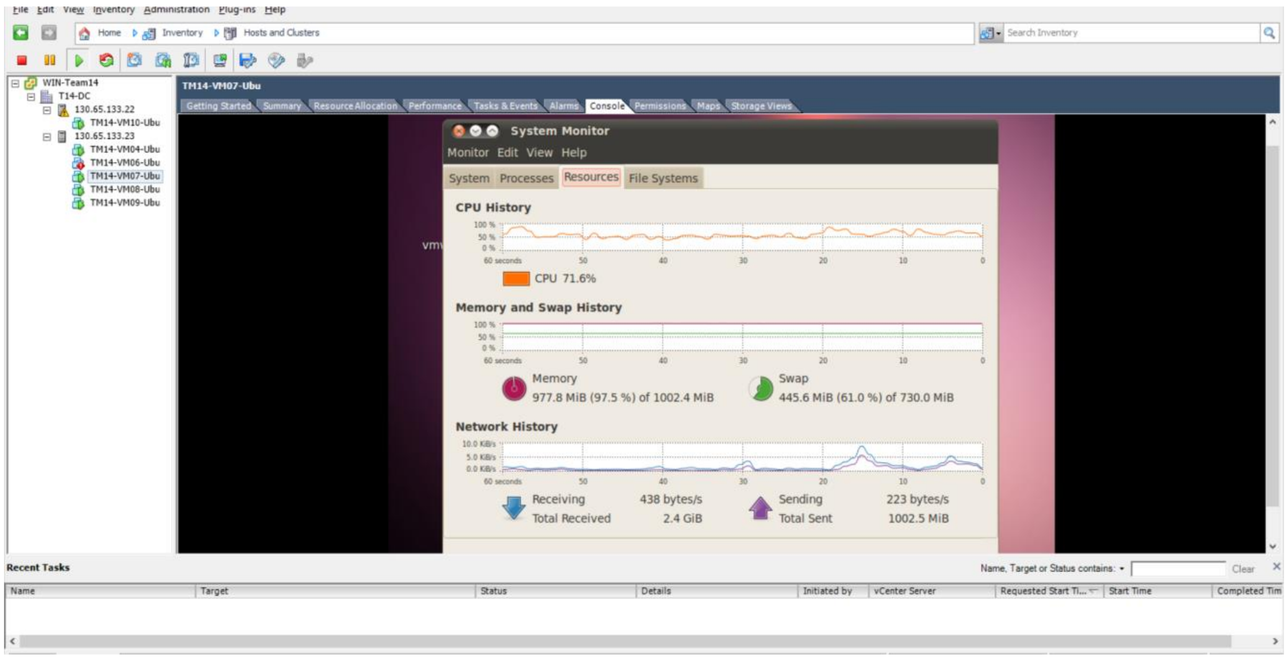

cmpe283@cmpe283-desktop: ~

```
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$ cat stress.sh

#! usr/bin

while true
do
stress --cpu 8 --timeout 10s;
sleep 300;
stress --cpu 4 --timeout 10s;
sleep 300;
stress --cpu 2 --timeout 10s;
sleep 300;
done
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$
```

Monitor CPU Utilization load in VM


cmpe283@cmpe283-desktop: ~

```
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$ cat stress.sh

#! usr/bin

while true
do
stress --cpu 8 --timeout 10s;
sleep 300;
stress --cpu 4 --timeout 10s;
sleep 300;
stress --cpu 2 --timeout 10s;
sleep 300;
done
cmpe283@cmpe283-desktop:~$
cmpe283@cmpe283-desktop:~$ sudo sh stress.sh
stress: info: [9998] dispatching hogs: 8 cpu, 0 io, 0 vm, 0 hdd
stress: info: [9998] successful run completed in 10s
stress: info: [10035] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: info: [10035] successful run completed in 11s
stress: info: [10070] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd
stress: info: [10070] successful run completed in 10s
```


File Edit View Inventory Administration Plug-ins Help
Home Inventory Hosts and Clusters
Search Inventory
WIN-Team14
T14-DC
130.65.133.22
TM14-VM10-Ubu
130.65.133.23
TM14-VM04-Ubu
TM14-VM06-Ubu
TM14-VM07-Ubu
TM14-VM08-Ubu
TM14-VM09-Ubu
TM14-VM07-Ubu
Getting Started Summary Resource Allocation Performance Tasks & Events Alarms Console Permissions Maps Storage Views
System Monitor
Monitor Edit View Help
System Processes Resources File Systems
CPU History
Memory and Swap History
Network History
Recent Tasks
Name Target Status Details Initiated by vCenter Server Requested Start Ti... Start Time Completed Tim

```
CPU 71.6%
Memory 977.8 MiB (97.5 %) of 1002.4 MiB
Swap 445.6 MiB (61.0 %) of 730.0 MiB
Receiving 438 bytes/s    Total Received 2.4 GiB
Sending 223 bytes/s      Total Sent 1002.5 MiB
```

14. CHALLENGES:

As part of this project, we got an in-depth knowledge of the VMWare architecture, a complete framework of logging mechanism in VM and VHosts, generating statistical reports and analysis of the same. In the middle all these learning, we faced few challenges too like:

1. Design proposal for maintaining a centralized ElasticSearch server such all other components could access it.
2. Identifying only the required five metric value from the list of performance metrics.
3. Configuring performance manager application as a startup application on VM bootup.
4. Configuring logstash filter grok pattern according to our input log file.
5. Using Kibana aggregation features which helps in generating graph based on time range of search string.
6. Exploring Kibana APIs to configure alarm on certain threshold value.
7. Implementation of alarm as an extension of project 1 code.

15. REFRENCES:

1. http://logstash.net
2. http://openjdk.java.net/projects/jdk6/
3. http://www.slashroot.in/logstash-tutorial-linux-central-logging-server
4. https://www.digitalocean.com/community/tutorials/how-to-use-kibana-dashboards-and-visualizations
5. https://www.vmware.com/support/pubs/sdk_pubs.html