Assignment 7x : COL733

Monitoring Dashboard for Baadal

Group 20

Authors	Rachit Arora	Vaibhav Bhagee	Ankush Phulia	Kabir Chhabra
Entry No.	2014CS50292	2014CS50297	2014CS50279	2013CS50287

A. Introduction

Baadal is a cloud orchestration software for academic and scientific environments. It is based on open technologies like Linux, KVM, libvirt, OpenVSwitch, Apache, MySQL, and web2py.
Workflows are implemented expected in academic and research environments, where a user or student can request a VM, a faculty supervisor can approve it, and an administrator can commission it.

Our module is a humble attempt to add a statistics monitoring dashboard for an administrator to gauge the resource requirements and consumption of the Baadal system with respect to the hosts as well as the VMs. The dashboard provides information relating to CPU and memory usage, Disk Read load, Disk Write load, Network read load and the Network Write load.

Finally this information is exposed to the user using web2py views with an intuitive and colorful user interface optimized for ease of use. The module has seamlessly integrated into the pre-existing Baadal repository.

B. Installation

We present the required steps to install Baadal on a system running Ubuntu 14.04 with the Monitoring Dashboard Module developed by our group, while using the institute proxy connection.

1. Install git and clone the repository for Baadal.
2. Replace the Baadal web2py code folders for views,models,modules and controllers by our code submission.

```
cd ~/baadal2.0/baadalinstallation/baadal
rm -rf views models modules controllers
cd ~/our-folder
cp -r views models modules controllers ~/baadal2.0/baadalinstallation/baadal/
```

3. Export proxy settings before trying to build the Baadal project code. Ensure that python-paramiko dependency is installed.

```
export http_proxy=proxy62.iitd.ernet.in:3128
export https_proxy=proxy62.iitd.ernet.in:3128
cd ~/baadal2.0/baadaltesting/devbox
```

4. Make the devbox code with a super user access and all environment variables.

```
sudo -E make devbox
```

5. It is possible that the installation throws up errors due to inability of dependencies to get resolved without reboot. It is important to reboot and repeat steps 3,4 every time such errors occur.

6. The makefile finally installs an Apache server and runs web2py website at `http://localhost/baadal` .

```
sudo mkdir /opt/hadoop/
cd /opt/hadoop
wget http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-1.2.0.tar.gz
tar -xzf hadoop-1.2.0.tar.gz
mv hadoop-1.2.0 hadoop
chown -R hadoop /opt/hadoop
```

7. The scheduler needs to be run.

```
cd /home/www-data/web2py
su www-data -c "python web2py.py -K baadal:vm_task,baadal:vm_sanity,baadal:hos
t_task,baadal:vm_rrd,baadal:snapshot_task &"
```

8. It is now possible to login as **admin** with password **baadal**. Post this new VMs can be requested and approved within the portal.

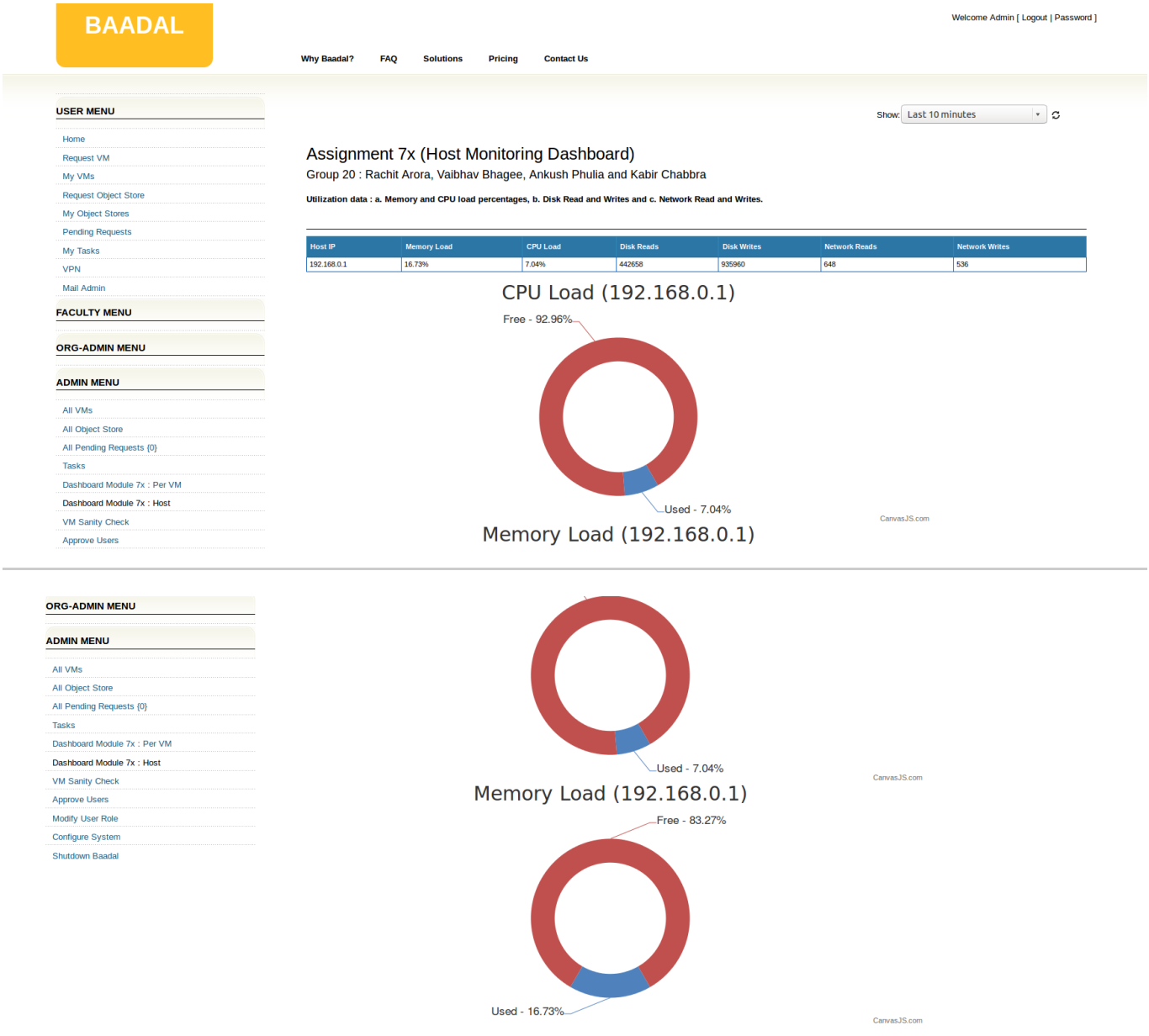
9. The monitoring dashboard module can be accessed from within the portal left pane as shown.

C. Monitoring

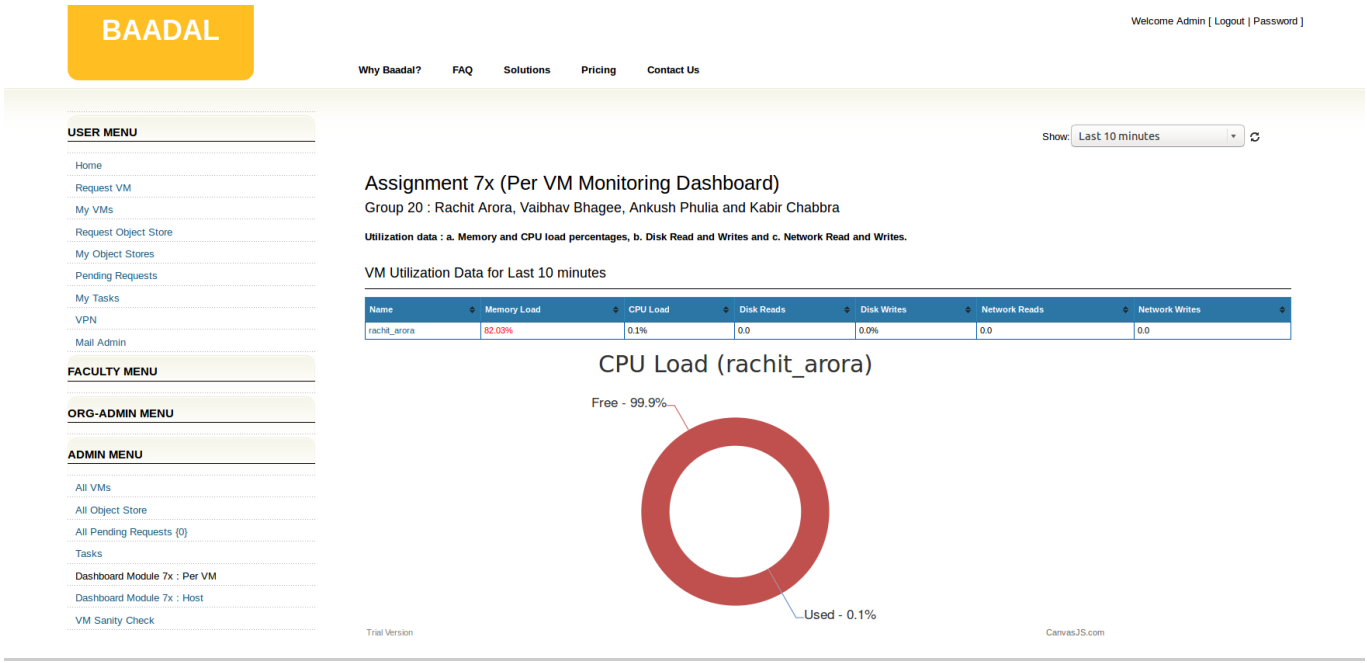
1. A variety of information is made available by the dashboard for tracking the machine workload for the Baadal host as well as all other running VMs. The linked statistics made available belong to the below categories:

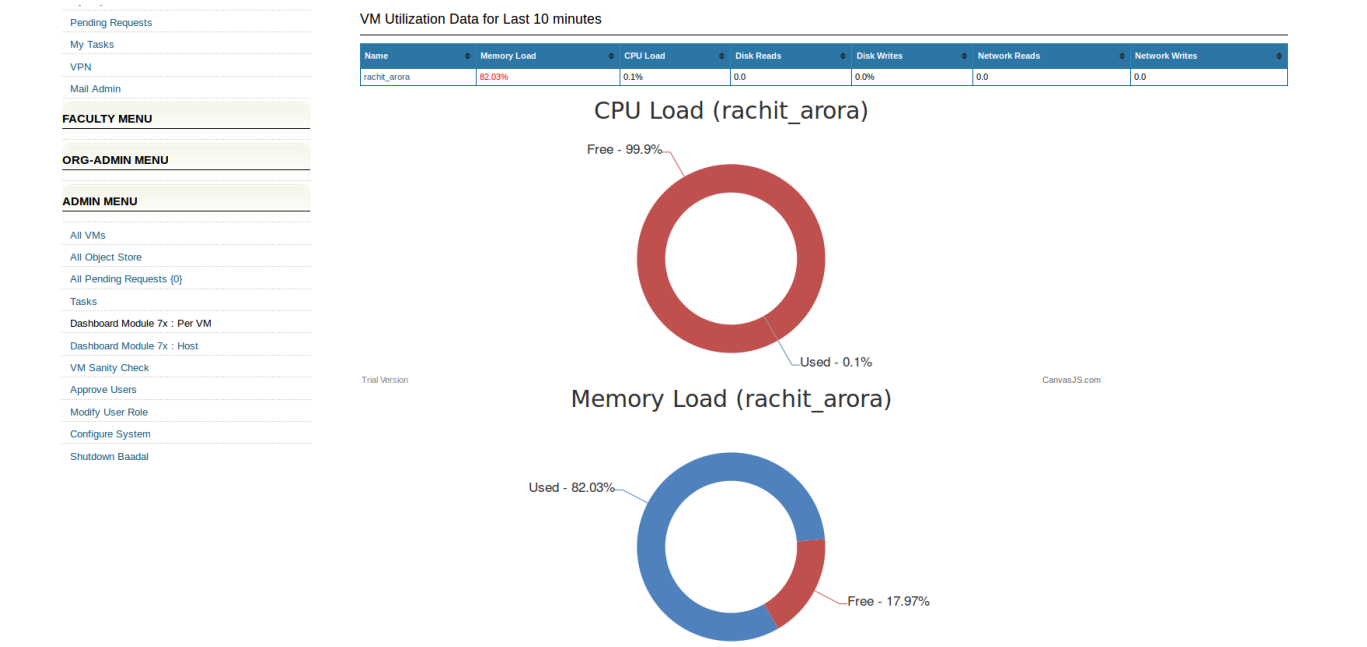
    - Memory Usage Percentage
    - CPU Usage Percentage
    - Disk Reads
    - Disk Writes
    - Network Reads
    - Network Writes

2. Monitoring for host reveals the above statistics for the host.



3. Monitoring per VM lists all the present VMs with their state and their usage/loads in a neat table and charts.





4. The dashboard allows changing the period for the statistics in a simple dropdown. It can allow viewing data collected recently, to viewing statistics averaged over months.

5. VMs with excessive memory or CPU usage (over 80%) get flagged in red so that more resources can be provisioned by the admin.

D. Design and Implementation

- The dashboard functionality is added to Baadal using web2py which is the same framework used by Baadal allowing seamless integration.
- web2py makes use of models, views and controllers for defining webpages and underlying logic efficiently.
- The data collection is accomplished for cpu, memory, network and disk statistics using the `modules/vm_utilization.py` .
- This information collected periodically is stored in a Round Robin Database.
- The database provides a lookup function `rrd_database_lookup()` for performing queries and return statistics for varying time durations.
- Relevant changes for monitoring were also made to some the `models/admin_model.py` file and the `controllers/admin.py` , to enable admin so that it can fetch the required data and store it properly.
- Finally views have been added (html files) to allow for visualizing the data. Javascript charts have been used to make a more readable UI for the dashboard

Models

- Added code that allows for accessing the data present in the rdd database.
- Time period for the data is taken as an input from the user before retrieving the information.
- Time period goes from fairly instantaneous (few minutes) to several months/years for monitoring VM and host performance.
- The functions `get_vm_util_data()` and `get_host_util_data()` in `admin_model.py` are used to collect the above mentioned information.

Controller

- Primarily used to convey the data in the models file logic to the html UI files in views.

Views

- Added views which receive and display the monitoring data appropriately.
- Tables present the data for all VMs in a list.
- A more intuitive interface is also used in the form of doughnut graphs for usage statistics.
- The dynamic behaviour of web2py allows detecting and communicating situations where the host or a VM is under extreme resource stress.
- Javascript Document Object Model allows for the neat animations and hover over behaviour.

E. References

- Baadal Repository at IITD PL-OS GitHub group
- W3Schools
- Web2Py documentation

- Baadal documentation on Moodle course page

F. Code Addition - Changelog

```
diff -ur baadalorig/controllers/admin.py modbaadal/controllers/admin.py
--- baadalorig/controllers/admin.py     2017-11-16 23:23:01.955670428 +0530
+++ modbaadal/controllers/admin.py      2017-11-16 23:37:46.751652287 +0530
@@ -93,12 +93,14 @@
 @check_moderator
 @handle_exception
 def hosts_vms():
-    form = get_util_period_form(submit_form=False)
+    form = get_util_period_form()
     util_period = VM_UTIL_10_MINS
     form.vars.util_period = util_period
+    if form.accepts(request.vars, session, keepvalues=True):
+        util_period = int(form.vars.util_period)
     host_util_data = get_host_util_data(util_period)
     hostvmlist = get_vm_groupby_hosts()
-    return dict(hostvmlist = hostvmlist, host_util_data = dumps(host_util_data), ut
il_form=form)
+    return dict(hostvmlist = hostvmlist, host_util_data = dumps(host_util_data), ho
st_util_data_val = host_util_data.values(), util_form=form)


 @check_moderator
@@ -546,7 +548,7 @@
 def vm_utilization():

     form = get_util_period_form()
-    util_period = VM_UTIL_24_HOURS
+    util_period = VM_UTIL_10_MINS
     form.vars.util_period = util_period

     if form.accepts(request.vars, session, keepvalues=True):
diff -ur baadalorig/models/admin_model.py modbaadal/models/admin_model.py
--- baadalorig/models/admin_model.py    2017-11-16 23:23:01.955670428 +0530
+++ modbaadal/models/admin_model.py     2017-11-16 23:37:46.751652287 +0530
@@ -10,7 +10,6 @@
     from gluon import db, request, session
     from applications.baadal.models import *  # @UnusedWildImport
 ###############################################################################
-from container_create import listallcontainerswithnodes
 from dhcp_helper import create_dhcp_entry, remove_dhcp_entry, \
     create_dhcp_bulk_entry
 from helper import IS_MAC_ADDRESS, get_ips_in_range, generate_random_mac
@@ -22,7 +21,7 @@
 from vm_helper import launch_existing_vm_image, get_vm_image_location, \
     get_extra_disk_location
 from vm_utilization import fetch_rrd_data, VM_UTIL_24_HOURS, VM_UTIL_ONE_WEEK, \
-    VM_UTIL_ONE_MNTH, VM_UTIL_ONE_YEAR, VM_UTIL_10_MINS, VM_UTIL_THREE_MNTH
+    VM_UTIL_ONE_MNTH, VM_UTIL_ONE_YEAR, VM_UTIL_10_MINS, VM_UTIL_THREE_MNTH, VM_UTI
L_1_MIN


 def get_manage_template_form(req_type):
@@ -816,7 +815,8 @@

 def get_util_period_form(submit_form=True):

-    _dict = {VM_UTIL_10_MINS : 'Last 10 minutes' ,
+    _dict = {VM_UTIL_1_MIN : 'Last 5 minutes',
+             VM_UTIL_10_MINS : 'Last 10 minutes' ,
              VM_UTIL_24_HOURS : 'Last 24 hours' ,
              VM_UTIL_ONE_WEEK : 'Last One Week',
              VM_UTIL_ONE_MNTH : 'Last One Month',
@@ -842,10 +842,10 @@
                     'vm_name' : vm.vm_name,
                     'memory'  : round(((util_result[0]/(vm.RAM * 1024*1024))*100), 2
),
                     'cpu'     : cpu_percent,
-                    'diskr'   : round(util_result[2], 2),
-                    'diskw'   : round(util_result[3], 2),
-                    'nwr'     : round(util_result[4], 2),
-                    'nww'     : round(util_result[5], 2)}
+                    'diskr'   : round(util_result[2], 0),
+                    'diskw'   : round(util_result[3], 0),
```

```
+                    'nwr'     : round(util_result[4], 0),
+                    'nww'     : round(util_result[5], 0)}
          vmlist.append(element)
      return vmlist

@@ -861,8 +861,13 @@
          mem_util=(util_result[0]/float(total_mem_kb))*100
          cpu_percent = round((float(util_result[1])*100)/(float(int(host_info.CPUs)*
5*60*1000000000)),2)

-         element = {'Memory' : str(round(mem_util,2)) + "%",
-                    'CPU'    : str(cpu_percent) + "%"}
+         element = {'IP' : host_info.host_ip.private_ip,
+                    'memory' : str(round(mem_util,2)),
+                    'cpu'    : str(cpu_percent),
+                    'diskr' : str(int(util_result[2])),
+                    'diskw' : str(int(util_result[3])),
+                    'nwr' : str(int(util_result[4])),
+                    'nww' : str(int(util_result[5]))}
          logger.debug(element)
          host_util_dict[host_info.id] = element

@@ -1225,6 +1230,7 @@

 def get_node_container_list():

-
+    from container_create import listallcontainerswithnodes
+
      return listallcontainerswithnodes()

diff -ur baadalorig/models/menu.py modbaadal/models/menu.py
--- baadalorig/models/menu.py   2017-11-16 23:23:01.955670428 +0530
+++ modbaadal/models/menu.py    2017-11-16 23:37:46.751652287 +0530
@@ -78,9 +78,9 @@
              response.admin_menu.insert(1, (T('All VM''s'), False, URL('admin','list
_all_vm')))

          if vm_enabled:
-             response.admin_menu.extend([(T('VM Utilization'), False, URL('admin','v
m_utilization')),
-                                         (T('VM Sanity Check'), False, URL('admin','
sanity_check')),
-                                         (T('Host and VMs'), False, URL('admin','hos
ts_vms'))])
+             response.admin_menu.extend([(T('Dashboard Module 7x : Per VM'), False,
URL('admin','vm_utilization')),
+                                         (T('Dashboard Module 7x : Host'), False, UR
L('admin','hosts_vms')),
+                                         (T('VM Sanity Check'), False, URL('admin','
sanity_check'))])
          if docker_enabled:
              response.admin_menu.extend([(T('Nodes and Containers'), False, URL('adm
in','nodes_containers')),
                                          (T('Container Sanity Check'), False, URL('a
dmin','cont_sanity_check'))])
@@ -128,4 +128,4 @@
          (T('Register Organization'), False, URL('default','page_under_construction'
)),
          (T('Contact Us'), False, URL('default','contact'))
          ]
-
\ No newline at end of file
+
diff -ur baadalorig/models/sanity_model.py modbaadal/models/sanity_model.py
--- baadalorig/models/sanity_model.py   2017-11-16 23:23:01.955670428 +0530
+++ modbaadal/models/sanity_model.py    2017-11-16 23:37:46.751652287 +0530
@@ -15,7 +15,6 @@
      from applications.baadal.models import *  # @UnusedWildImport
 ##############################################################################
 from cont_handler import Container
-from container_create import get_node_to_deploy, list_container
 from helper import execute_remote_cmd, log_exception
```

```
 from host_helper import HOST_STATUS_UP, get_host_domains
 from images import getImage
@@ -319,6 +318,7 @@
     cont_check= []
     cont_list = []

+    from container_create import get_node_to_deploy, list_container
     nodes = get_node_to_deploy()
     containers = list_container( showall=True);

diff -ur baadalorig/models/task_scheduler.py modbaadal/models/task_scheduler.py
--- baadalorig/models/task_scheduler.py 2017-11-16 23:23:01.955670428 +0530
+++ modbaadal/models/task_scheduler.py  2017-11-16 23:37:46.751652287 +0530
@@ -15,8 +15,8 @@
 from log_handler import logger, rrd_logger
 from host_helper import HOST_STATUS_UP
 from load_balancer import find_host_and_guest_list, loadbalance_vm
-from container_create import install_cont, start_cont, stop_cont, suspend_cont,\
-    resume_cont, delete_cont, restart_cont, recreate_cont, backup_cont
+#from container_create import install_cont, start_cont, stop_cont, suspend_cont,\
+#    resume_cont, delete_cont, restart_cont, recreate_cont, backup_cont
 from gluon import current
 current.cache = cache

@@ -40,16 +40,16 @@
         VM_TASK_CLONE               :    clone,
         VM_TASK_ATTACH_DISK         :    attach_extra_disk,
         VM_TASK_SAVE_AS_TEMPLATE    :    save_vm_as_template,
-        VM_TASK_DELETE_TEMPLATE     :    delete_template,
-        CONTAINER_TASK_CREATE       :    install_cont,
-        CONTAINER_START             :    start_cont,
-        CONTAINER_STOP              :    stop_cont,
-        CONTAINER_SUSPEND           :    suspend_cont,
-        CONTAINER_RESUME            :    resume_cont,
-        CONTAINER_DELETE            :    delete_cont,
-        CONTAINER_RESTART           :    restart_cont,
-        CONTAINER_RECREATE          :    recreate_cont,
-        CONTAINER_COMMIT            :    backup_cont
+        VM_TASK_DELETE_TEMPLATE     :    delete_template
+#        CONTAINER_TASK_CREATE       :    install_cont,
+#        CONTAINER_START             :    start_cont,
+#        CONTAINER_STOP              :    stop_cont,
+#        CONTAINER_SUSPEND           :    suspend_cont,
+#        CONTAINER_RESUME            :    resume_cont,
+#        CONTAINER_DELETE            :    delete_cont,
+#        CONTAINER_RESTART           :    restart_cont,
+#        CONTAINER_RECREATE          :    recreate_cont,
+#        CONTAINER_COMMIT            :    backup_cont
     }


diff -ur baadalorig/modules/vm_helper.py modbaadal/modules/vm_helper.py
--- baadalorig/modules/vm_helper.py     2017-11-16 23:23:01.959670428 +0530
+++ modbaadal/modules/vm_helper.py      2017-11-16 23:37:46.751652287 +0530
@@ -988,9 +988,8 @@
         logger.debug(current_disk_file)

         xmlfile = domain.XMLDesc(0)
-        root = etree.fromstring(xmlfile)

-        if(live_migration!=True):
+        if(live_migration==False):
             rc = os.system("cp %s %s" % (current_disk_file, diskpath))

             if rc != 0:
@@ -1003,6 +1002,7 @@
             if domain.isActive:
                 domain.undefine()

+                root = etree.fromstring(xmlfile)
                 target_elem = root.find("devices/disk/target")
                 target_disk = target_elem.get('dev')
                 #
```

```
diff -ur baadalorig/modules/vm_utilization.py modbaadal/modules/vm_utilization.py
--- baadalorig/modules/vm_utilization.py        2017-11-16 23:23:01.959670428 +0530
+++ modbaadal/modules/vm_utilization.py 2017-11-16 23:37:46.751652287 +0530
@@ -30,6 +30,7 @@
 import rrdtool
 import libvirt

+VM_UTIL_1_MIN = 10
 VM_UTIL_10_MINS = 1
 VM_UTIL_24_HOURS = 2
 VM_UTIL_ONE_WEEK = 3
@@ -72,6 +73,9 @@
         start_time = '-3m'
     elif period == VM_UTIL_ONE_YEAR:
         start_time = '-1y'
+    elif period == VM_UTIL_1_MIN:
+        start_time = 'now-' + str(5*60)
+
     cpu_data = []
     mem_data = []
     dskr_data = []
@@ -392,7 +396,13 @@
 def get_host_temp_usage(host_ip):
     rrd_logger.info("Checking temp on host"+str(type(host_ip)))
     command='ipmitool sdr elist full'
-    ret=execute_remote_cmd(host_ip, 'root', command, None,  True)
+    try:
+        ret=execute_remote_cmd(host_ip, 'root', command, None,  True)
+    except Exception, e:
+        temp=0
+        ret=""
+        rrd_logger.debug("ipmitool not installed")
+
     if str(ret).find("Inlet Temp")!=int(-1):
         rrd_logger.debug("Entering")
         temp=get_impi_output('ipmitool sdr elist full | grep "Inlet Temp"',host_ip)
@@ -407,7 +417,13 @@
 def get_host_power_usage(host_ip):
     rrd_logger.info("Checking power on host"+str(host_ip))
     command='ipmitool sdr elist full'
-    ret=execute_remote_cmd(host_ip, 'root', command, None,  True)
+    try:
+        ret=execute_remote_cmd(host_ip, 'root', command, None,  True)
+    except Exception, e:
+        pwr_usage=0
+        ret=""
+        rrd_logger.debug("ipmitool not installed")
+
     if str(ret).find("System Level")!=int(-1):
         pwr_usage=get_impi_output('ipmitool sdr elist full | grep "System Level"',h
ost_ip)
     if str(ret).find("Pwr Consumption")!=int(-1):
diff -ur baadalorig/views/admin/hosts_vms.html modbaadal/views/admin/hosts_vms.html
--- baadalorig/views/admin/hosts_vms.html       2017-11-16 23:23:01.963670428 +0530
+++ modbaadal/views/admin/hosts_vms.html        2017-11-16 23:37:46.751652287 +0530
@@ -4,84 +4,171 @@
 {{=util_form}}
 </div>

-{{for vmhosts in hostvmlist:}}
-<h2 class="title">{{=A('host :'+ str(vmhosts['host_ip']), _href=URL(r=request,c='ad
min' ,f='host_config', args=[vmhosts['host_id']] ))}}</h2>
-<div id="host_utilization_{{=vmhosts['host_id']}}"></div>
-    <div class="TABLE">
-    <table id="host_{{=vmhosts['host_ip']}}">
+{{response.files.append(URL('static', 'css/tablecustom.css'))}}
+<br/>
+<br/>
+<h1>Assignment 7x (Host Monitoring Dashboard)</h1>
+<h2> Group 20 : Rachit Arora, Vaibhav Bhagee, Ankush Phulia and Kabir Chabbra </h2>
+<br/>
+<h3> Utilization data : a. Memory and CPU load percentages, b. Disk Read and Writes
 and c. Network Read and Writes. </h3>
```

```
+<br/>
+
+<h2 class="subtitle" id="pageUtilHeader"></h2>
+
+  <div class="container">
+    <table id="sortTable1" class="tablesorter">
+    <thead>
       <tr>
-        <th>Name</th>
-        <th>Owner</th>
-        <th>Organisation</th>
-        <th>Private IP</th>
-        <th>RAM</th>
-        <th>vCPUs</th>
-        <th>Status</th>
-        <th>Settings</th>
-      </tr>
-      {{for vm in vmhosts['details']:}}
-      <tr>
-        <td>{{=vm['name']}}</td>
-        <td>{{=vm['owner']}}</td>
-        <td>{{=vm['organisation']}}</td>
-        <td>{{=vm['private_ip']}}</td>
-        <td>{{=vm['RAM']}}</td>
-        <td>{{=vm['vcpus']}}</td>
-        <td>{{=vm['status']}}</td>
-        <td>
-            {{=A(IMG(_src=URL('static','images/settings.png'), _height=18, _width=1
8),
-            _href=URL(r=request,c='user' ,f='settings', args=[vm['id']]), _id="vm_"
+str(vm['id']),
-            _title="Settings",
-            _alt="Settings")}}
-        </td>
-      </tr>
+        <th>Host IP</th>
+        <th>Memory Load</th>
+        <th>CPU Load</th>
+        <th>Disk Reads</th>
+        <th>Disk Writes</th>
+        <th>Network Reads</th>
+        <th>Network Writes</th>
+      </tr>
+      </thead>
+      <tbody>
+      {{for util_data in host_util_data_val:}}
+       <tr>
+          <td>{{=util_data['IP']}}</td>
+            {{if float(util_data['memory']) > 80:}}
+            <td><font color="red">{{=util_data['memory']}}%</font></td>
+            {{else:}}
+            <td>{{=util_data['memory']}}%</td>
+            {{pass}}
+            {{if float(util_data['cpu']) > 80:}}
+            <td><font color="red">{{=util_data['cpu']}}%</font></td>
+            {{else:}}
+            <td>{{=util_data['cpu']}}%</td>
+            {{pass}}
+            {{if int(util_data['diskr']) < 0:}}
+            <td><font color="red">{{=util_data['diskr']}}</font></td>
+            {{else:}}
+            <td>{{=int(util_data['diskr'])}}</td>
+            {{pass}}
+            {{if int(util_data['diskw']) < 0:}}
+            <td><font color="red">{{=util_data['diskw']}}</font></td>
+            {{else:}}
+            <td>{{=util_data['diskw']}}</td>
+            {{pass}}
+            {{if int(util_data['nwr']) < 0:}}
+            <td><font color="red">{{=util_data['nwr']}}</font></td>
+            {{else:}}
+            <td>{{=util_data['nwr']}}</td>
+            {{pass}}
```

```
+            {{if int(util_data['nww']) < 0:}}
+            <td><font color="red">{{=util_data['nww']}}</font></td>
+            {{else:}}
+            <td>{{=util_data['nww']}}</td>
+            {{pass}}
+          </tr>
       {{pass}}
-  </table>
+  </tbody>
+    </table>
   </div>
-  <br>
 {{pass}}

+
+<script src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
+
+{{for util_data in host_util_data_val:}}
+<div id="chartContainer0_{{=util_data['IP']}}" style="height: 370px; width: 80%;"><
/div>
+<div id="chartContainer1_{{=util_data['IP']}}" style="height: 370px; width: 80%;"><
/div>
+
+<script>
+window.onload = function () {
+
+var chart0 = new CanvasJS.Chart("chartContainer0_{{=util_data['IP']}}", {
+    animationEnabled: true,
+    title:{
+        text: "CPU Load ({{=util_data['IP']}})"
+    },
+    data: [{
+      type: "doughnut",
+      startAngle: 60,
+      indexLabelFontSize: 17,
+      indexLabel: "{label} - #percent%",
+      toolTipContent: "{label}: {y} (#percent%)",
+      dataPoints: [
+        { y: {{=util_data['cpu']}}, label: "Used" },
+        { y: {{=(100.0-float(util_data['cpu']))}}, label: "Free" }
+      ]
+    }]
+});
+var chart1 = new CanvasJS.Chart("chartContainer1_{{=util_data['IP']}}", {
+    animationEnabled: true,
+    title:{
+        text: "Memory Load ({{=util_data['IP']}})"
+    },
+    data: [{
+      type: "doughnut",
+      startAngle: 60,
+      indexLabelFontSize: 17,
+      indexLabel: "{label} - #percent%",
+      toolTipContent: "{label}: {y} (#percent%)",
+      dataPoints: [
+        { y: {{=util_data['memory']}}, label: "Used" },
+        { y: {{=(100.0-float(util_data['memory']))}}, label: "Free" }
+      ]
+    }]
+});
+chart0.render();
+chart1.render();
+
+}
+</script>
+{{pass}}
+
+
 <script>

 jQuery(document).ready(function(){
-       var JSONstring = '{{=host_util_data}}';
-       JSONstring = JSONstring.replace(/&quot;/ig,'"');
```

```
+  var JSONstring = '{{=host_util_data}}';
+  JSONstring = JSONstring.replace(/&quot;/ig,'"');

-       parse_utilization_data(jQuery.parseJSON(JSONstring));
-
+  parse_utilization_data(jQuery.parseJSON(JSONstring));
+
 });

 function get_utilization_data() {

-       var util_data = jQuery('#period_select_id').val();
+  var util_data = jQuery('#period_select_id').val();

-       jQuery.ajax(
-       {
-               url : '{{=URL('get_host_utilization_data')}}',
-               type: 'POST',
-               dataType : 'json',
-               data: {keywords:util_data},
-               success:function(data) {
-
-                       parse_utilization_data(data);
-               }
-       });
+  jQuery.ajax(
+  {
+    url : '{{=URL('get_host_utilization_data')}}',
+    type: 'POST',
+    dataType : 'json',
+    data: {keywords:util_data},
+    success:function(data) {
+
+      parse_utilization_data(data);
+    }
+  });
 }


 function parse_utilization_data(JSONstring){

-       $.each(JSONstring, function(host_id, host_util_data) {
-
-               display_str = '<b>Utilization data for ' + $('#period_select_id opti
on:selected').text() + '</b> [';
-               $.each(host_util_data, function(param, util_val) {
+  $.each(JSONstring, function(host_id, host_util_data) {

-                       display_str += param + ': ' + util_val + ', ';
-               });
-               display_str = display_str.substring(0, (display_str.length)-2);
-               display_str += ']';
-               $('#host_utilization_'+host_id).html(display_str)
-       });
+    display_str = '<b>Utilization data for ' + $('#period_select_id option:selected
').text() + '</b> <br> [';
+    flag = 0;
+    $.each(host_util_data, function(param, util_val) {
+
+      display_str += param + ': ' + util_val + ', ';
+      if(parseFloat(util_val) > 0){
+        flag = 1;
+      }
+    });
+    display_str = display_str.substring(0, (display_str.length)-2);
+    display_str += ']';
+    if(flag == 1){
+      display_str += '<br> <font color="red"> Host ' + host_id + ' is overloaded</f
ont>';
+    }
+    //$('#host_utilization_'+host_id).html(display_str)
+  });
```

```
 }
 </script>
diff -ur baadalorig/views/admin/vm_utilization.html modbaadal/views/admin/vm_utiliza
tion.html
--- baadalorig/views/admin/vm_utilization.html  2017-11-16 23:23:01.963670428 +0530
+++ modbaadal/views/admin/vm_utilization.html   2017-11-16 23:37:46.751652287 +0530
@@ -3,41 +3,133 @@
 {{=form}}
 </div>

+
+{{response.files.append(URL('static', 'css/tablecustom.css'))}}
+<br>
+<br>
+<h1>Assignment 7x (Per VM Monitoring Dashboard)</h1>
+<h2> Group 20 : Rachit Arora, Vaibhav Bhagee, Ankush Phulia and Kabir Chabbra </h2>
+<br/>
+<h3> Utilization data : a. Memory and CPU load percentages, b. Disk Read and Writes
 and c. Network Read and Writes. </h3>
+<br/>
+
 {{if len(vm_util_data) == 0:}}
 <h3>No Utilization Data</h3>
     {{else:}}

        <h2 class="subtitle" id="pageUtilHeader"></h2>

+    <div class="container">
     <table id="sortTable1" class="tablesorter">
     <thead>
       <tr>
-        <th>VM Name</th>
-        <th>Memory</th>
-        <th>CPU</th>
-        <th>Network Read</th>
-        <th>Network Write</th>
-        <th>Disk Read</th>
-        <th>Disk Write</th>
+        <th>Name</th>
+        <th>Memory Load</th>
+        <th>CPU Load</th>
+        <th>Disk Reads</th>
+        <th>Disk Writes</th>
+        <th>Network Reads</th>
+        <th>Network Writes</th>
       </tr>
       </thead>
       <tbody>
       {{for util_data in vm_util_data:}}
        <tr>
                        <td>{{=A(util_data['vm_name'],_href=URL(r=request, c='user',
f='show_vm_performance', args=[util_data['vm_id']]))}}</td>
+            {{if util_data['memory'] > 80:}}
+            <td><font color="red">{{=util_data['memory']}}%</font></td>
+            {{else:}}
             <td>{{=util_data['memory']}}%</td>
+            {{pass}}
+            {{if util_data['cpu'] > 80:}}
+            <td><font color="red">{{=util_data['cpu']}}%</font></td>
+            {{else:}}
             <td>{{=util_data['cpu']}}%</td>
+            {{pass}}
+            {{if util_data['diskr'] < 0:}}
+            <td><font color="red">{{=util_data['diskr']}}</font></td>
+            {{else:}}
             <td>{{=util_data['diskr']}}</td>
-            <td>{{=util_data['diskw']}}</td>
+            {{pass}}
+            {{if util_data['diskw'] < 0:}}
+            <td><font color="red">{{=util_data['diskw']}}</font></td>
+            {{else:}}
+            <td>{{=util_data['diskw']}}%</td>
+            {{pass}}
```

```
+            {{if util_data['nwr'] < 0:}}
+            <td><font color="red">{{=util_data['nwr']}}</font></td>
+            {{else:}}
             <td>{{=util_data['nwr']}}</td>
+            {{pass}}
+            {{if util_data['nww'] < 0:}}
+            <td><font color="red">{{=util_data['nww']}}</font></td>
+            {{else:}}
             <td>{{=util_data['nww']}}</td>
+            {{pass}}
           </tr>
       {{pass}}
        </tbody>
     </table>
+</div>
+{{pass}}
+
+<script src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
+
+{{for util_data in vm_util_data:}}
+<div id="chartContainer0_{{=util_data['vm_name']}}" style="height: 370px; width: 80
%;"></div>
+<div id="chartContainer1_{{=util_data['vm_name']}}" style="height: 370px; width: 80
%;"></div>
 {{pass}}
+
 <script>
+window.onload = function () {
+
+{{for util_data in vm_util_data:}}
+var chart0_{{=util_data['vm_name']}} = new CanvasJS.Chart("chartContainer0_{{=util_
data['vm_name']}}", {
+    animationEnabled: true,

+    title:{
+        text: "CPU Load ({{=util_data['vm_name']}})"
+    },
+    data: [{
+        type: "doughnut",
+        startAngle: 60,
+        indexLabelFontSize: 17,
+        indexLabel: "{label} - #percent%",
+        toolTipContent: "{label}: {y} (#percent%)",
+        dataPoints: [
+            { y: {{=util_data['cpu']}} , label: "Used" },
+            { y: {{=(100-util_data['cpu'])}} , label: "Free" }
+        ]
+    }]
+});
+var chart1_{{=util_data['vm_name']}} = new CanvasJS.Chart("chartContainer1_{{=util_
data['vm_name']}}", {
+    animationEnabled: true,
+
+    title:{
+        text: "Memory Load ({{=util_data['vm_name']}})"
+    },
+    data: [{
+        type: "doughnut",
+        startAngle: 60,
+        indexLabelFontSize: 17,
+        indexLabel: "{label} - #percent%",
+        toolTipContent: "{label}: {y} (#percent%)",
+        dataPoints: [
+            { y: {{=util_data['memory']}} , label: "Used" },
+            { y: {{=(100-util_data['memory'])}} , label: "Free" }
+        ]
+    }]
+});
+chart0_{{=util_data['vm_name']}}.render();
+chart1_{{=util_data['vm_name']}}.render();
+{{pass}}
+
+}
```

```
+</script>
+
+
+
+<script>
 jQuery(document).ready(function(){
     jQuery("#sortTable1").tablesorter({ headers: { 5: {sorter:"ipAddress"}} });
```