|  |  |  |
| --- | --- | --- |
| **Module Code**  **COMP5850M** | School of Computing  University of Leeds  **Coursework 1 - Report** | University of Leeds logo |

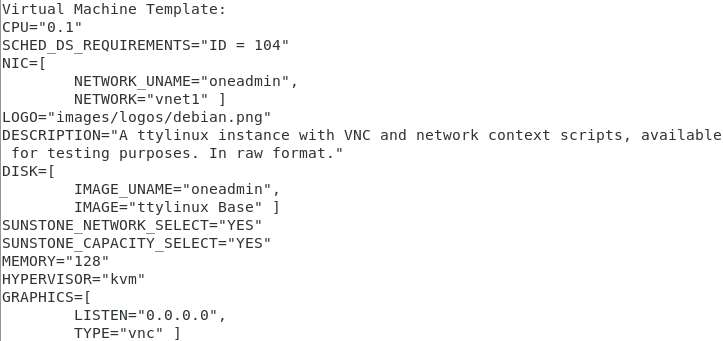
Full Name: Username:

Coursework Title: VIM Deadline Date: 09/03/2020

**Part 1: Java OpenNebula Cloud API (OCA) (10 marks)**

*Provide an explanation of the implementation of this task. The inclusion of the entire code is not required but you may include snippets if you wish.*

*VM template (1 mark)*



*Information OpenNebula provides about the VM (1 mark)*



*Measure the time it takes to instantiate/delete the VM. To get these measurements you are expected to run the experiments n times (e.g. n = 5). A statistical analysis (average, standard deviation) is expected. (2 marks)*

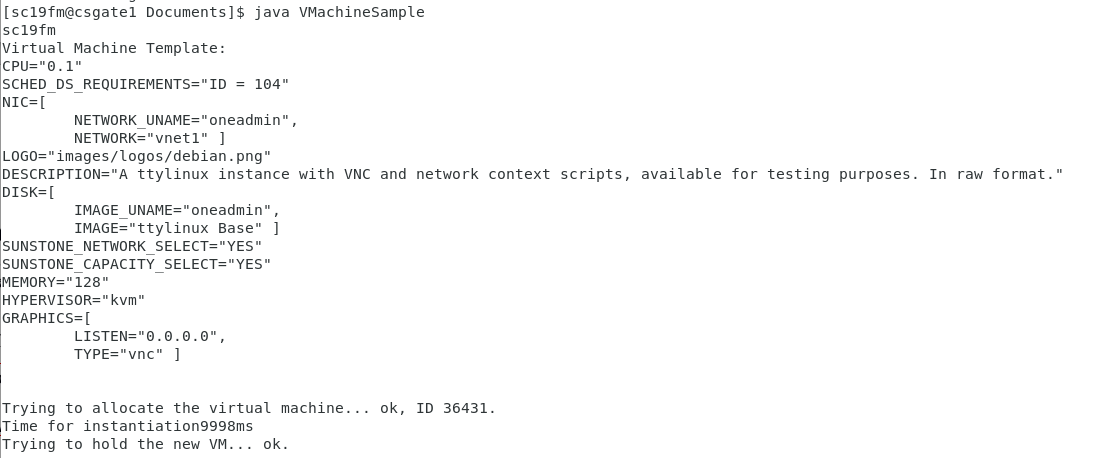
|  |  |  |
| --- | --- | --- |
| **Run No.** | **VM instantiation time** | **VM deletion time** |
| **1** | 8934 ms | 157 ms |
| **2** | 8486 ms | 128 ms |
| **3** | 9998 ms | 126 ms |
| **4** | 11203 ms | 135 ms |
| **5** | 8111 ms | 167 ms |
| **Average** | 9346.4 ms | 142.6 ms |
| **Standard Deviation** | 1123.22 | 16.43 |

*Explain how you have obtained these measurements (2 marks)*

Use currentTimeMillis() method to get the current system time (startTime) before instantiation a VM. Then I write a loop, keep checking the vm’s status. Until the vm is up running, break the loop. Then use the same method to get the time (endTime) after instantiation. And the instantiation time is the endTime minus the startTime.

Then use the same way to get the VM deletion time. Set startTimeDelete before delete. Set endTimeDelete after finalizeVM() method and make sure there is nothing wrong. Finally use these two numbers to calculate the deletion time.

*Evidence of successful run, e.g. screenshot (4 marks)*



****

**Part 2: VM Migration (15 marks)**

*Provide an explanation of the implementation of this task. The inclusion of the entire code is not required but you may include snippets if you wish.*

*Requirements (2 marks)*

Migrate existing VM to a better host. Considering the host’s load balance, CPU usage, memory usage and disk usage. Ideally, the host with lower load balance, CPU usage, etc should be better.

*Solution Design (2 marks)*

Check all of the hosts’ information in the host pool. Assign each factor with a weight. As far as I concerned, the number of VM shouldn’t be the main factor. Because some of the VMs require more CPU and memory, and some of them are not. The more important factor should be the CPU, memory and disk usage. So I assign 0.5 to the number of VM, assign 1 to other 3 factors.

As for these usage, I use the maximum usage divided by the current usage, then multiply 100 to represent these usage.

Finally, I add up the four parameters with each weights for each host. Then the host with lowest number is exactly the host I want the VM to migrate.

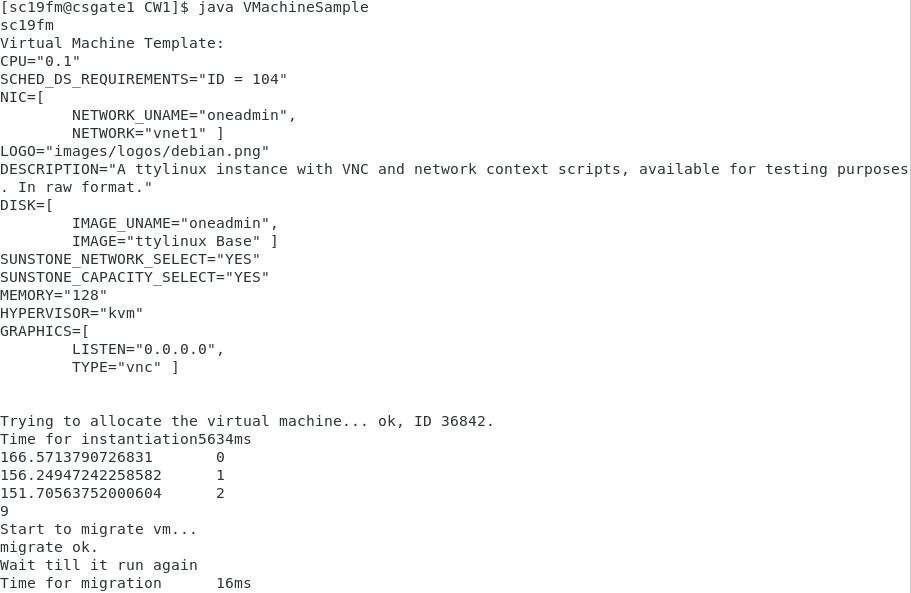
*Implementation (2 marks)*

Get the host id, CPU usage, memory usage and disk usage from the host pool. Then put each host with it’s properties into an array. Sum up these number with each weight. Then compare with each other, find the lowest and it’s host id. Live migrate the vm to the target host. Wait until the vm is up running again. Finally, check the target host information, make sure the vm is already in the target host.

*Measure the time it takes to migrate the VM. (2 marks)*

|  |  |
| --- | --- |
| **Run No.** | **VM migration time** |
| **1** | 15 ms |
| **2** | 24 ms |
| **3** | 21 ms |
| **4** | 19 ms |
| **5** | 16 ms |
| **Average** | 19 ms |
| **Standard Deviation** | 3.29 |

*Evidence of successful run, e.g. screenshot (3 marks)*

**

**

**

*Discussion of the results (4 marks)*

The host with more number of VM doesn’t mean it’s CPU and memory are occupied more. To consider if a host is suitable for migration, we should focus on it’s CPU, memory and disk usage. More basically, we should see the host’s status, whether it’s on or off, and it’s allocated CPU, memory is available or not. And if the hosts are in the same cluster, which means within the same LAN, it should take shorter time to migrate. I use live migrate the VM to a new host, which means transfer running vm between hosts without disconnecting the client or application. And this way only take a small time to migrate a VM.

**Part 3: Resource Scaling and Performance/Energy Consumption Trade-Off**

**(10 - 25 marks, depending on application and challenge)**

*Details of the application considered (stress, MPI, Hadoop, other) (1-3 marks)*

Use hadoop data parallel processing job, run MapReduce application on up to 4 Vms and 4 hosts.

*Design of the experiments (1-4 marks)*

I design 6 experiment:

1. 1 MapReduce application run on 1 VM and 1 physical host
2. 1 MapReduce application run on 2 VMs and 1 physical host
3. 1 MapReduce application run on 2 VMs and 2 physical hosts
4. 1 MapReduce application run on 4 VMs and 1 physical hosts
5. 1 MapReduce application run on 4 VMs and 2 physical hosts (2 VMs for each host)
6. 1 MapReduce application run on 4 VMs and 4 physical hosts

Then use Zabbix-based monitoring infrastructure to keep an eye on the power consumption, CPU usage.

*Implementation of the experiments (1-4 marks)*

Expand the .txt file size for the application to 100MB. In this way can make the application run for longer time, and it’s better for the obversation.

*Discussion of results (3-10 marks))*

*Evidence of successful run, e.g. screenshot (4 marks)*

**Any other comments:**