Threshold Based Auto Scaling of Virtual Machines in Cloud Environment

M.K. Mohan Murthy, H.A. Sanjay, and Jumnal Anand

Nitte Meenakshi Institute of Technology, Bangalore
{maakem,sanju.smg,anandsbj1989}@gmail.com

Abstract. Cost effectiveness is one of the reasons behind the popularity of Cloud. By effective resource utilization cost can be further reduced and resource wastage can be minimized. The application requirement may vary over time depending on many factors (for instance load on the application); user may run different types of application (a simple MS word to complex HPC application) in a VM. In such cases if the VM instance capacity is fixed there is a high possibility of mismatch between the VM capacity and application resource requirement. If the VM capacity is more than the application resource requirement then resource will be wasted; if the VM capacity is less than the application resource requirement then the application performance will degrade. To address these issues we are proposing threshold based auto scaling of virtual machines in which VMs will be dynamically scaled based on the application resource utilization (CPU and Memory). Using our approach effective resource utilization can be achieved.

Keywords: Cloud Computing, Auto Scaling, Virtual Machines.

1 Introduction

In cloud paradigm software, infrastructure, and platform are given as services. In this work we are considering infrastructure (Virtual Machine). IaaS providers provide Virtual Machine (VM) to end user. User will use VM instance to host/run his applications and he will pay some amount as per the SLA (Service Level Agreement). Many organizations moving towards private cloud; effective resource utilization, cost reduction, and easy maintenance are some of the reasons behind this. Employees in the organization will get the VM instances. They have to login to these instances to use them. Whether it is commercial cloud or private cloud following two scenarios are possible

1.1 Scenario 1: User Hosts Different Application on the VM

User may use VM to host different applications from a simple MS word to complex accounting software. If the VM instance is static (normally this will be the case) user has to select VM instance in such a way to match the application which has the maximum resource requirement. In this case for instance if the user uses the VM to

C.-H. Hsu et al. (Eds.): NPC 2014, LNCS 8707, pp. 247–256, 2014.

run his application which has the maximum requirement only for 2 hours in a day then in remaining 22 hours resource will be wasted. If the application resource requirement is more than the VM then it leads to application performance degradation.

1.2 Scenario 2: Application Requirement Vary over Time

Consider a database application which needs more resource when the transactions are happening. If the transactions are not there it doesn't need high resources. In case of the static VM instance this will lead to resource wastage.

Migration of application from one VM to another address' above mentioned issues but it is having many disadvantages. It is time consuming, tedious, not cost effective, and error prone. If VM is dynamically scaled according to the application requirement resource wastage can be minimized.

To address the above mentioned issues we have developed and tested threshold based auto scaling of VM mechanism, in which the VM is auto configured according to the application requirement. In threshold based auto scaling the resource utilization of the VM is monitored. If they exceed the predefined threshold values then VM capacity will be increased or decreased dynamically without shutting them down according to the need, which minimizes resource wastage.

1.3 Up-Scaling

In this work we have considered RAM and CPU utilization of the VM. As the resource requirement of the application increases, the RAM and CPU utilization of the VM increases. At some point the application resource requirement will become more when compare to VMs capacity as a result performance of the application degrades and ultimately it will hang. To avoid this problem when the CPU and Memory utilization of the VM crosses the predefined maximum threshold value we will increase the RAM and CPU capacity of the VM.

1.4 Down-Scaling

As the resource requirement of the application decreases, the RAM and CPU utilization of the VM decreases. This will lead to resource wastage since the VM capacity is not fully utilized. To avoid resource wastage when the CPU and Memory utilization of the VM crosses the predefined minimum threshold value we will decrease the RAM and CPU capacity of the VM. Monitoring and scaling of RAM and CPU of the VM are two independent tasks.

Rest of the paper is organized as follows section 2 gives brief description about the related work; section 3 explains the threshold based auto scaling of VM; section 4 describes the algorithms used to upscale/downscale the VM; section 5 talks about the experimental setup and results, followed by conclusion.

2 Related Work

There are couple of efforts related to dynamic resource provisioning in cloud environment. In [1] auto-scaling of the VM instances with respect to the load, where load is defined as number of jobs submitted and the deadline and budget to complete the submitted jobs are considered. In the proposed system the VM must be rebooted after scaling it. So there will be time delay and performance degradation. In some scenarios rebooting of VM is not acceptable.

In [2] a novel architecture is presented for the dynamic scaling of web applications based on thresholds in a virtualized cloud computing environment. This work illustrates the scaling approach with a front-end load balancer for routing and balancing user requests. Web applications are deployed on web servers installed in virtual machine instances. In [3] a cloud computing architecture is constructed with a front-end load balancer, a virtual cluster monitor system and an auto-provisioning system. The front-end load balancer is utilized to route and balance user requests to cloud services deployed in a virtual cluster. The virtual cluster monitor system is used to collect the statistics of the usage of physical resources in each virtual machine in the virtual cluster. The auto-provisioning system is used to dynamically provision the virtual machines based on the number of the active sessions or the use of the resources in the virtual cluster.

In the works [2] [3] front end load balancer is used for load balancing on virtual machines. In these works a new instance of VM is added to the VM cluster if the resource utilization crosses the upper threshold. If the utilization is below the lower threshold then VM instances are removed from the cluster. In these works scaling of the VM based on the hosted application requirement is not addressed. In [4] a model-driven engineering approach is presented to optimize the configuration, energy consumption, and operating cost of cloud auto-scaling infrastructure to create greener computing. This work concentrates on energy consumption and budget constraints. In this work pre-configured static VM instances are used. This will lead to resource wastage as well as application performance degradation.

In all of these works cloud is scaled by adding new VMs. They are not considering auto scaling of a single VM which is required in the scenario where user will be using a VM to run his applications in private or public cloud. Our work is about to auto scale a VM based on the threshold values.

3 Threshold Based Auto Scaling of VM

Application requirement may change over time and also user may host different applications (which have different resource requirement) on the VM. In these cases fixed VM capacity may lead to resource wastage or application performance degradation. This can be addressed by dynamically scaling the VM according to the hosted application requirement. In threshold based auto scaling the resource utilization of the VM is monitored. If they exceed the predefined threshold values then VM capacity will be increased or decreased dynamically according to the need without shutting down the VMs, which minimizes resource wastage. High level system overview is shown in figure 1.

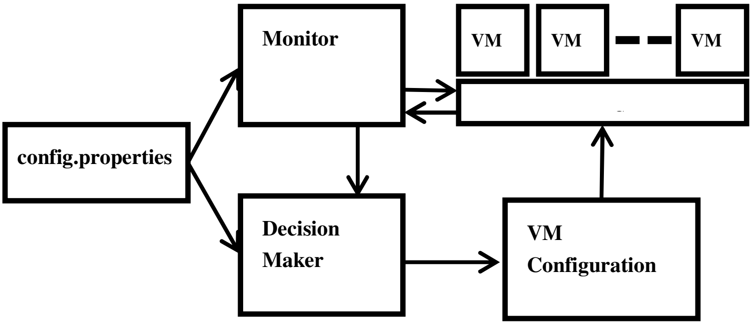

Fig. 1. Auto Scaling system overview

Auto-scaling system has the following components

3.1 Monitor

The monitor component monitors the VMs; it reads the CPU and Memory utilization and passes this data to Decision Maker component. It uses Xen APIs to get the CPU and Memory utilization of the VMs. It sends request for the CPU and Memory utilization of VMs to the XCP using Xen APIs. By default it monitors all the active VMs or we can make it to monitor only specific VMs by configuring the corresponding values in config.properties. The time interval to send the request to XCP to get the VMs statistics can be configured in config.properties.

When the Monitor module starts it will read all the configuration properties from the config.properties and monitors the VMs as per the values set to different property parameters in config.properties file.

3.2 Decision Maker

Decision maker module gets the VM statistics from Monitor module and it also read the threshold values from the config.properties file, compares against the VM statistics and decides whether to up/down scale the VM and conveys this decision to VM configuration module. The information passed to the VM configuration module includes the VM ID which should be scaled, whether scaling should happen to RAM or CPU and how much scaling should happen. All the threshold values and scaling values (helps in to take the decision of how much scaling should happen) are configurable which are stored in the config.properties file.

There is a possibility that VM's CPU and RAM utilization may exceed the threshold value for few seconds and again come back to the normal values. If the monitor module gets these values it will trigger up/downscaling of the RAM/CPU of the VM. In the next iteration monitor module gets the normal values again this triggers down/up-scaling of the RAM/CPU of the VM which results in unnecessary up/down scaling of VMs. To avoid this problem we have introduced configurable properties called cpuiteration (min and max) and memoryiteration (min and max).

Any positive integer value from 0 to n can be set to the cpuiteration and memoryiteration. Both are independent of each other, min memoryiteration and min cpuiteration are used in case of down-scaling, max memoryiteration and max cpuiteration are used in case of up-scaling. The Decision Maker initiates up/downscaling only if the RAM and CPU utilization of the VM exceeds the threshold values in the successive number of iterations specified in the cpuiteration (min and max) and memoryiteration (min and max) .

4 Scaling Algorithms

We have written two separate algorithms for memory scaling and CPU scaling. The working principle of both the algorithms is same.

In case of the memory scaling algorithm each VM's memory utilization (Mx) is read and compared with max memory threshold value (Mmx). If Mx is greater than or equal to Mmx then the max memory utilization counter (MTx) is increased and min memory utilization counter (MTm) is reset.

4.1 Memory Scaling

```
Step 1: for each VMx Read Memory utilization Mx
Step 2: if the Mx>=Mmx
    - Increment the MTx for the VMx
    - Reset MTm for the VMx
else if Mx<=Mmn
    - Increment the MTm for the VMx
    - Reset MTx for the VMx

Step 3: if MTx>TTMt and if free memory available
    - initiate the VM up-scaling for memory
    - go to step 5
else if MTm >TTMt
    - initiate the VM down-scaling for memory
    - go to step 5
Step 4: Go to Step 1
Step 5: Reset the MTx and MTm
Step 6: Go to Step 1
```

4.2 CPU Scaling

```
Step 1: for each $VM_x$ Read CPU utilization $C_x$
Step 2: if the $C_x>=C_{mx}$
      - Increment the $CT_x$ for the $VM_x$
      - Reset $CT_m$ for the $VM_x$
   else if $C_x<=C_{mn}$
```

```
        - Increment the $CT_m$ for the $VM_x$
        - Reset $CT_x$ for the $VM_x$
Step 3: if $CT_x > TTC_t$ and if computing resources available
        - initiate the VM up-scaling for CPU
        - go to step 5
  else if $CT_m > TTC_t$
        - initiate the VM down-scaling for CPU
        - go to step 5
Step 4: Go to Step 1
Step 5: Reset the $CT_x$ and $CT_m$
Step 6: Go to Step 1
```

VMx - Virtual Machine identifier
Mmx - Memory maximum threshold defined
Mmn - Memory minimum threshold defined
Cmx - CPU maximum threshold defined
Cmn - CPU minimum threshold defined
T – Time interval defined to read the Memory, CPU utilization of VMs
TTMt - Time threshold counter defined for Memory.
TTCt - Time threshold counter defined for CPU.
CTx - Max CPU iteration count.
MTx - Max memory iteration count.
CTm - Min CPU iteration count.
MTm - Min memory iteration count.
Cx - CPU utilization of VMx
Mx - Memory utilization of VMx

5 Experimental Setup and Results

A Cloud Environment is set up using Xen Cloud Platform (XCP) [5]. XCP includes Xen Hypervisor, Xen API tool-stack, vSwitch etc. XCP is an open source enterprise-ready server virtualization and cloud computing platform. Many of the existing IaaS providers are using the customized XEN to create the virtualization infrastructure. XCP delivers the Xen hypervisor with support for a range of guest operating systems including Windows and Linux network and storage support, management tools in a single, tested installable image, which is also called XCP appliance. It also supports the dynamic scaling of virtual machine. XCP APIs are used to get the memory and CPU utilization statistics of the VMs and to up-scale and down-scale the VMs.

5.1 Memory Scaling

XCP supports two types of memories: static and dynamic. Each will have minimum and maximum range. The static memory maximum defines the maximum amount of physical memory that the guest operating system can address from the time the guest boots up until the time the guest shuts down again. It is not possible to change static memory when the VM is running. In case of the dynamic memory it is possible to increase/decrease the range when the VM is running. XCP provides a feature called dynamic memory controller. Using the following API's provided by the XCP we increase/decrease the dynamic memory within the valid range whenever required.

```
xe vm-param-set uuid=<uuid> memory-dynamic-{min, max};
```

uuid is the identifier which uniquely identifies a VM.

5.2 CPU Scaling

CPU scaling can be done by either modifying the CPU cap or CPU weight. The CPU cap optionally fixes the maximum amount of CPU a domain will be able to consume. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc... The default, 0, means there is no upper cap [5]. CPU weight of a VM decides how much CPU is allocated to that VM. A domain with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended host. Legal weights range from 1 to 65535 and the default is 256 [5]. In our work we have used the CPU cap to scale the CPU allocated to VM. The following API provided by the XCP is used.

```
xe vm-param-set uuid<uuid> VCPUs-params:cap=<value>
```

The Cloud Server is setup on a 4 Core Machine with Intel Xeon W3250 processor with 2.67 GHz, 12 GB of DDR3 RAM, 1 TB Hard disk with 7200 RPM. The desktop machine which is used to monitor the VMs for memory and CPU utilization of the application has the Intel Core2 Duo processor with 2.66 GHz clock speed, 1 GB of RAM, 500 GB hard disk, connected over an Ethernet LAN. CentOS 5.7 Operating System is used.

To generate load on VMs we have used both computing intensive programs and memory intensive programs. Once the programs are started the CPU and memory utilization of the VMs will increase. We have set the scaling factor for memory as 1.25 and for CPU it is 2. The utilization count is set to 3 minutes i.e. if the resource usage exceeds the upper threshold value for 3 minutes continuously, then the corresponding VMs will be allocated more resources as specified in the scaling factor (up-scaling). If the resource usage is below the lower threshold value for 3 minutes continuously, then the resource will be de-allocated from the VM as per the scaling factor (down-scaling).

We have setup the threshold values as follows
Upper threshold – 80%, Lower threshold – 25%

Figure 2 and 3 shows memory up-scaling and downscaling respectively. Figure 4 and 5 shows CPU up-scaling and downscaling respectively. Since we have set the utilization count to 3 minutes, we can observe the scaling at 4th minute in all the cases.

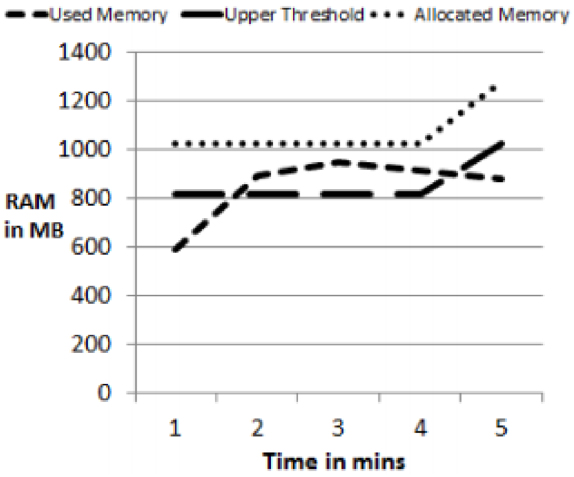

Fig. 2. Memory up-scaling

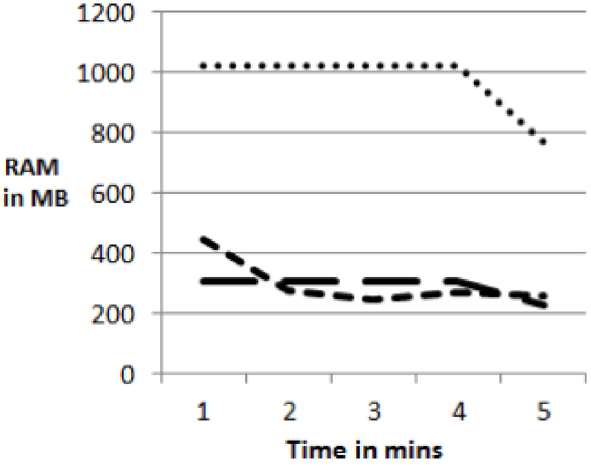

Fig. 3. Memory down-scaling

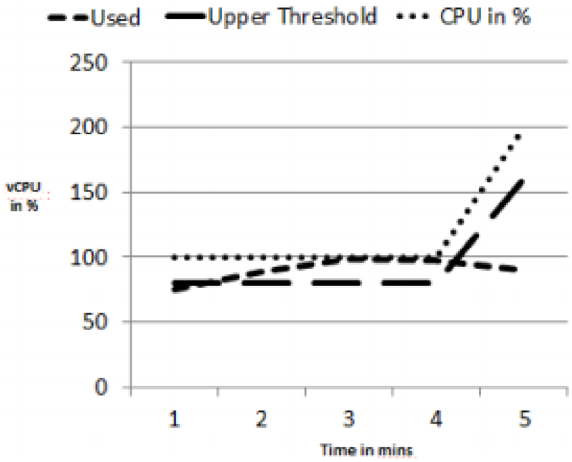

Fig. 4. CPU up-scaling

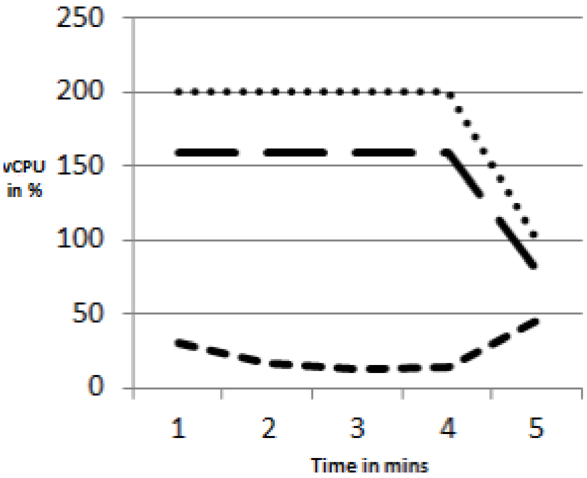

Fig. 5. CPU down-scaling

Following is our observations

- Downscaling minimizes the resource wastage.
- Up-scaling make sure that the application performance is not compromised.
- Choosing the right threshold values is very important for the success of our approach.

A lower threshold value result in fluctuation of the VM capacity and a higher threshold value makes our algorithms less responsive to the change in resource utilization of the VM.

6 Conclusion

By adopting effective resource utilization techniques resource wastage can be minimized. Our threshold based auto-scaling is one such technique in which VM is dynamically scaled as per the application resource requirement, thereby minimizing the resource.

Selecting proper threshold values is very important factor in the success of our approach. A lower threshold value leads to frequent change in VM configuration and a higher value reduces the responsiveness of VM to adapt to the new resource requirement. We can use several techniques to find out optimum threshold values such as history based, mathematical model etc. In future we are planning to build a feedback based system to dynamically scale the VM according to the application requirement. At present our dynamic scaling system is threshold based where threshold values are static and predefined. In feedback based system we will monitor the application's resource utilization and it will be used as feedback to define the threshold values.

References

1. Mao, M., Li, J., Humphrey, M.: In: Schwarz, T.S.J., Miller, E.L.: Cloud Auto-scaling with Deadline and Budget Constraints. Department of Computer Science University of Virginia Charlottesville, VA, USA 22904 {ming, jl3yh, humphrey}@cs.virginia.edu (2011)
2. Chieu, T.C., Mohindra, A., Karve, A.A., Segal, A.: Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment. In: 2009 IEEE International Conference on e-Business Engineering (2009)
3. Hung, C.-L., Hu, Y.-C., Li, K.-C.: Auto-Scaling Model for Cloud Computing System. Dept of Computer Science & Information Engineering, Providence University {clhung, ychu, kuancli}@pu.edu.tw
4. Doughertya, B., Whiteb, J., Schmidta, D.C.: Model-driven Auto-scaling of Green Cloud Computing. Institute for Software Integrated Systems, Vanderbilt University, Campus Box 1829 Station B, Nashville, TN 37235, Email:{briand,schmidt}@dre.vanderbilt.edu bECE, 302 Whitemore Hall, Virgnia Tech, Blacksburg, VA 24060, Email:julesw@vt.edu
5. http://wiki.xen.org/XCP_Design_and_Architecture