THE EFFECT OF THE RESOURCE CONSUMPTION CHARACTERISTICS OF CLOUD APPLICATIONS ON THE EFFICIENCY OF LOW-METRIC AUTO SCALING SOLUTIONS

Ha Van Cu[1], Nguyen Hong Son[2]

[1]R&D Lab, Faculty of Information Technology, Post and Telecommunication Institute of Technology, Ho Chi Minh City, Vietnam
[2]Faculty of Information Technology, Post and Telecommunication Institute of Technology, Ho Chi Minh City, Vietnam

ABSTRACT

Auto scaling is a service provided by the cloud service provider that allows provision of temporary resources to the subscriber's systems to prevent overloading. So far, many methods of auto scaling have been proposed and applied. Among them, solutions based on low-level metrics are commonly used in industry systems. Resource statistics are the basis for detecting overloading situation and making additional resources in a timely manner. However, the effectiveness of these methods depends very much on the accuracy of the overload calculation from low-level metrics. Overloading is mentioned in solutions that usually favor a shortage of CPU resources. However, the demand for resources comes from the application running on that each application has the characteristics of demanding different resource types, with different CPU, memory, I/O ratios so it can not just be statistically on CPU consumption. The point of view here is that even though based on low level resources, the source for calculation and forecasting is the characteristic of the resource needs of the application. In this paper, we will develop an empirical model to assess the effect of the application's resource consumption characteristics on the efficiency of the low-metricauto scaling solutions and propose an auto scaling solution that is calculated based on statistics of different types of resources. The results of the simulations show that the proposed solution based on multiple resources is more positive.

1. INTRODUCTION

Over decades of development, cloud computing has become a key component of modern information systems. Many of the advances in technology have contributed to the development of the cloud. Everyone recognizes that thanks to the explosive growth of broadband technology, the community has been able to use cloud computing smoothly. Cloud computing services are also abundant and easily accessible thanks to the development of Internet technology. Sales from cloud computing is on the rise. In a world trend that is entering Industry 4.0, cloud computing continues to affirm its position with many intelligent applications that use the cloud platform.

Today, the deployment of the entire enterprise IT infrastructure in the cloud is no longer a rarity for many reasons. First of all, the need to reduce the cost of infrastructure and operating costs because sharing resources and energy in the cloud is much cheaper than investing in private infrastructure. Another reason is that many modern applications that businesses want to use, especially those that target mobile end-users, IoT, and ecommerce, are trending toward using services provided from the cloud such as web services, no SQL database. Implementing, changing, and upgrading applications in the cloud is also quicker and more convenient. Information security services are also well supported.In addition to the ability to protect infrastructure and services from cloud service providers, subscribers can rapidly deploy their own

DOI: 10.5121/ijccsa.2018.8501

security measures to protect their applications. Another reason is that the power and capacity provided by the cloud for the subscriber can vary according to demand. Building and providing auto scaling services to subscribers of cloud service providers like Amazon EC2, Azure is very meaningful.With auto scaling, when the enterprise resources are overloaded, the cloud can dynamically add temporary resources to maintain the normal service of the information system and also automatically retrieve the resources when it returns to normal state.The auto scaling service not only benefits the cloud clients but also facilitates the cloud service provider to maximize revenue as it can increase the number of subscribers and can leverage idle resources in flexible way.From the meaning of auto scaling that the implementation of this technique has become a topic of concern.

Typically, cloud systems allow subscribers to choose auto scaling types along with select the load balancer. How the auto scaling technique implemented within the service system depends on the cloud providers. The auto scaling technique follows the trend of positive surveillance, based on the state of the system to make the decision to scale resources. Decision-making can be made using accurate measurement data or forecasting technologies, but whatever the type, auto-scaling solutions are set up with parameters that directly or indirectly relate to processing resources. However, processing resources are not the entire story of meeting service requirements, because when the type of resources are not scarce and other resources are insufficient, service requests are still waiting. As in [8], it has been shown that memory intensive applications also heavily influence the performance of the cloud. We therefore need to approach the problem of auto scaling from the perspective of resource consumption characteristics of applications run on the cloud. The applications will feature different resource needs, such as simulation, brute force applications that use large amounts of processing power, and memory intensive applications like data analysis applications. There are also many applications that require both processor power and memory in high for execution, such as graphics and animation applications.

In this study, we will demonstrate that auto scaling solutions based only on processing resources are not quality-assured when cloud-based applications are altered.First of all, we build a typical auto scaling algorithm based on processor utilization monitoring and model a subscriber cloud infrastructure with an autoscaler to derive parameter calculations necessary for evaluating the auto scaling solution. The evaluation will be through computer simulation results based on the Cloudsim tool. The rest of the paper is structured as follows: Section 2 shows some researches that deal with auto scaling and application load on the cloud. Section 3 details the construction of a subscriber cloud infrastructure model with an autoscaler as the basis for simulating and evaluating auto scaling solutions. This section also presents a typical auto scaling algorithm based on processor utilization monitoring used for evaluation purposes in the paper. Section 4 presents computer simulations along with analysis and recommendations for auto scaling solutions tailored to the changing of cloud applications. The paper ends with the conclusions set out in section 5.

2. RELATED WORKS

The auto scaling solution can be grouped by reactive and proactive type in which reactive will adjust resource when overload occurs while proactive will forecast and actively adjust resource first to not overload appear. It is also possible to distinguish auto scaling solutions based on the use of metric measurements as either low-resource measurement or high-level measurement [11]. Low level parameters such as resource utilities and high level parameters such as number of users, request rate, average response time, session creation rate, throughput, service time, and request mix. We have explored the real-life auto scaling (industry system) and typical works related to the construction of auto scaling solutions. We are particularly interested in load-bearing solutions and applications running on cloud-based systems requiring auto scaling. Law-based

solutions are widely used in the industry autoscalers such as Amazon EC2 [5], Microsoft Azure or OpenStack, the key components are pre-established rules based on resource measurements. For example, “if CPU utilization Exceeding 0.7 adds an instance and if less than 0.3 the instance is destroyed." This method is reactive and based on low level resource measurement. This is the solution that can be implemented by cloud providers only. Also related to low-level measurements in [2], the authors propose a hybrid method between reactive and proactive applied to an intermediary enterprise managing a virtual private cloud that provides resources to subscribers. The authors also mentioned the necessity of selecting the metric and determining the exact thresholds for auto scaling from which to provide the GoS quantity and the standard for auto scaling based on GoS. However, in terms of performance, GoS service time is still calculated on the basis of CPU processing power and workload only distinguishes between the two types of requirements is OD (on demand) and AR (advance reservation). Other proactive methods based on high-level metrics such as in [1] have analyzed and pointed to the challenges of auto scaling. The first challenge is to accurately predict future workload, and then calculate the amount of resources needed to increase or decrease according to the workload and finally to overhead the transition when the number of resources changes. Based on the analysis of the challenges, the authors proposed a method of distributing resources using model predictive techniques. Resources are then distributed or retrieved based on the application's optimal utility over a predetermined prediction time. However, the workload used here is the number of current users in the system, which does not reflect the characteristics of the application. Similarly, in [3] an adaptive control approach can be used to change the number of VMs allocated to a service based on load change monitoring and also future load forecasts. However, the basis of calculation is based only on the speed of the service request. Another auto scaling solution is proposed in [4] by predicting future service needs using standard time-series analysis techniques. The scaling depends on the change of throughput in 10 minutes. Similarly, the authors in [6] also used time series analysis to propose a resource forecasting model based on double exponential smoothing to improve forecast accuracy, but based on the overall variability of workload.

In general, active monitoring solutions are based on actual metrics of processing resources or forecasting methods using high-level metrics such as request rate, average response time, session creation rate, throughput, service time, and request mix. Most solutions do not take into account the difference in demand for different resource types of applications. With auto scaling solutions for the web app like [7], if the demand for resources is balanced between types (CPU, Memory, Bandwidth), forecast accuracy will not be affected much. But when the application is switched to CPU intensive or memory intensive, be inadequate is inevitable.

In the interest of evaluating solutions, [9] also developed a fuzzy rule-based system to compare auto-scaling solutions using reinforcement learning techniques, Fuzzy Q-learning (FQL) and Fuzzy SARSA learning (FSL), the results also show that FSL is effective for periodic workloads. Similarly, in [10] the authors provide a simulation model for evaluating different auto scaling policies that use workflows as the input for evaluations. However, both [9] and [10] do not address the specific needs of different resource types of applications.

3. CLOUD MODEL WITH AN AUTOSCALER

As stated for the purpose of this paper, we need to examine whether the effectiveness of the auto scaling solutions is affected as the demand for different resource types of application changes. To do that we will firstly model a subscriber cloud infrastructure that subscribes to horizontal auto scaling.

Subscribers are scaled automatically by the number of virtual machines added or reduced once is N machine, we have:

$$numOfVMScale(t)=N \times scaleEvent(t) \quad (1)$$

with numOfVMScale(t) is number of virtual machines (VM) added or reduced at time t
scaleEvent (t) = 1 or 0 or -1 depending on the decision function, called D(). In case the auto scaling algorithm is implemented in an active metric monitoring style, the autoscaler keeps track of the remaining resources,denoted as remainResource (t), which introduced to the function D() along with a threshold.

$$scaleEvent(t)=D(remainResource(t),Threshold) \quad (2)$$

Considering the load generated by the request i to VM as I instrutions, C memory in Megabytes, and bandwidth B in Mbps, denoted as request(Ii,Ci,Bi) and load of VM is total of load of requests at concerning time t, called requests(t). The configuration of the virtual machine has MIPS processing resources, the executable memory M, and BW bandwidth, written as VMresources (MIPS, M, BW). The remaining resources will be computed using the f() method based on requests(t) and VMresources (MIPS, M, BW).

$$remainResource(t)= f(requests(t),VMresources(MIPS,M,BW)) \quad (3)$$

With n requests at concerning time t, requests(t) as following:

$$\text{requests(t)} = \text{request}\left(\sum_{i=1}^{n} I_i , \sum_{i=1}^{n} C_i , \sum_{i=1}^{n} B_i\right) \quad (4)$$

The method f() can be a way of subtraction of processing power in MIPS or memory capacity in Megabytes.

To evaluate the effectiveness of the use of hiring resources including additional virtual machine resources under the auto scaling service here define a metric called VMxTime, if the time hourly called a VM hour and a VM hour means opening a virtual machine for an hour. The cost of consuming resources is attributed to the VM hour by taking the virtual machine number multiplied by the corresponding open time in hour. Let VM Usage be the virtual machine consumption during some time T, it is expressed as follow:

$$VM\ Usage = \int_0^T NumOfVMs(t)dt \quad (5)$$

NumOfVMs(t) is numbers of VMs opening at time t. The item is expressed by formulation:

$$numOfVMs(t)=currentNumofVMs(t) + numOfVMsScale(t) \quad (6)$$

The larger the VM Usage, the greater the cost to the subscriber

.

Determine the average completion time of a request (Ii, Ci, Bi):

Let I be the total number of instructions for all requests processed during time T and C is the total memory capacity of all memory needs of all requests processed during time T. The T/I ratio is the average time that an instruction is completed and T/C ratio is the average time required to response one Mbyte. If the CPU biased request, use the ratio T/I and if the request for the RAM intensive, use the ratio T/C. We can refer to the 1000 units, the request has 1000 instructions, or 100MB memory requirements, multiply 1000T/I or 100T/C.To evaluate the quality of the auto scaling mechanism uses the average virtual machine consumption per request and the average time spent completing a request. Compare cases based on these two parameters.

Auto scaling algorithm:

The service is done by horizontal scaling and assuming that the load balancer is well-balanced, ensuring that the deviation and the load balancer are matched as described in [12].

```
1.Set the threshold value Vth;
2.Set idle time T1; //before destroying a VM
3.Set sample time T2;// statistical time
4.Parallel:
Each subscribed VM {
Collect and calculating the average CPU utility during T2;
 If (the average CPU utility > Vth) {
                        create VM;
                        register the VM to Loadbalancer;
                        }
5.Parallel:
Each opened VM {
                Count the idle time;
                If (idle time>T1) {
                Delete VM from register list of Loadbalancer;
                destroy the VM;
                }
```

4. COMPUTER SIMULATIONS

The simulation programs in this section are made with Cloudsim. We build a data center with configuration information as follows:

Data center has 50 hosts, each host has 32 Pe, 1000MIPS/Pe, 16GB RAM, 10Gbps, 1000GB storage. Virtual machine scheduling according to time-shared policy.
Virtual machines have 1000MIPS with 1Pe; 4096MB RAM, 1000Mbps, and the scheduling policy for cloudlets is time-shared. Suppose the enterprise was granted 4 VMs initially. The request (cloudlet) corresponds to length = 5000MI, filesize = 1913 and outputsize = 1000, pe=1.Cloudlets are also customized according to different resource consumption characteristics for testing.

4.1 AUTO SCALING USING THE CPU THRESHOLD

In this simulation,we use the CPU utilization threshold of 0.95 (Vth). The provisioning is a series of requests, with each request being an application executed on a virtual machine with a very high CPU consumption characteristic and a relatively small memory requirement of less than 0.25. We calculate VMxTime consumption statistics, service completion statistics. Figure 1 shows the number of virtual machines opened and closed over time (NumOfVMs (t)) for this experiment.

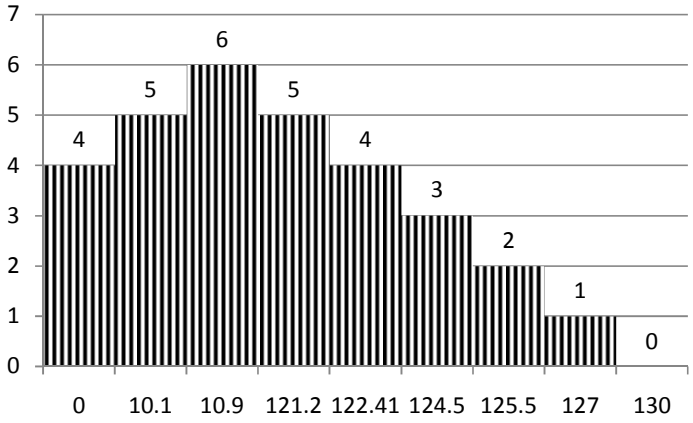

Figure 1. Virtual machines opened and closed over time with low RAM requests

With statistical results, the average VMxTime cost for a request is 11.22 VM seconds and the average completion time for a request is 34 seconds.

Next, experiments performed with applications still have very high CPU resource requirements and average RAM needs are below 0.45. The resulting virtual machine number is opened and closed as shown in Figure 2.

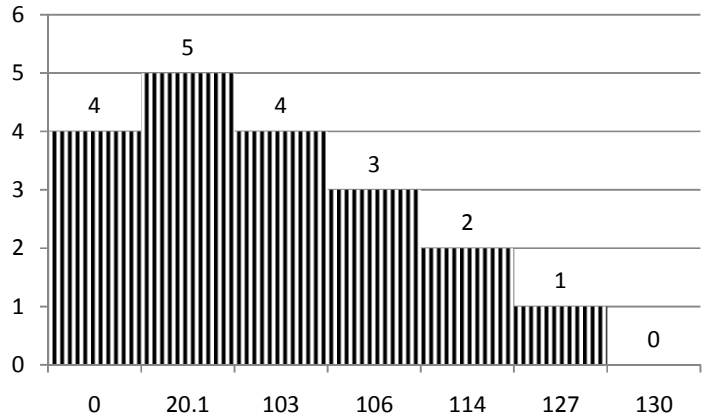

Figure 2. Virtual machines opened and closed over time with moderate RAM requests

The average virtual machine consumption per request is 16.3 VM seconds and the average completion time for a request is 47 seconds. The results show that compared to the small RAM demand, the virtual machine consumption per request has increased by 45% and the average request completion time has increased by about 38%.

Continuing to test the impact of memory needs on auto scaling, we have applications with low CPU utilization, maximum 0.4, and memory requirements are high, over 0.5. The result of the number of virtual machines opened and closed over time is depicted in Figure 3.

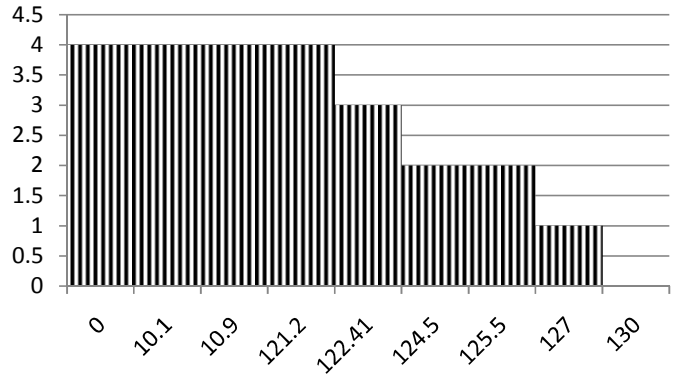

Figure 3. The number of virtual machines opened and closed over time, corresponding to the needs of small CPU and high RAM

As can see in Figure 3, almost no virtual machines have been added for a long time, but the cost of virtual machine time has increased significantly, with an average of 20.7 VM seconds per request. Possibly due to lack of resources, the average time to complete a request also jumps to 63 seconds. In general, when the application has a large CPU and small RAM, the parameters are doubled.

4.2 AUTO SCALING USING BOTH CPU AND MEMORY THRESHOLDS

From the simulation results above, we propose performing auto scaling considering both the CPU threshold and Memory threshold. To ensure SLA in real-world situations, applications that have different resource-consuming characteristics need to perform auto scaling across multiple resource thresholds with the rule that whenever any threshold is violated, resources must be added immediately. We validated the solution with a CPU threshold of 0.95 and a RAM threshold of 0.9. The applications used for the simulation have high CPU requirement, above 0.71 and memory requirement are also high, over 0.5.

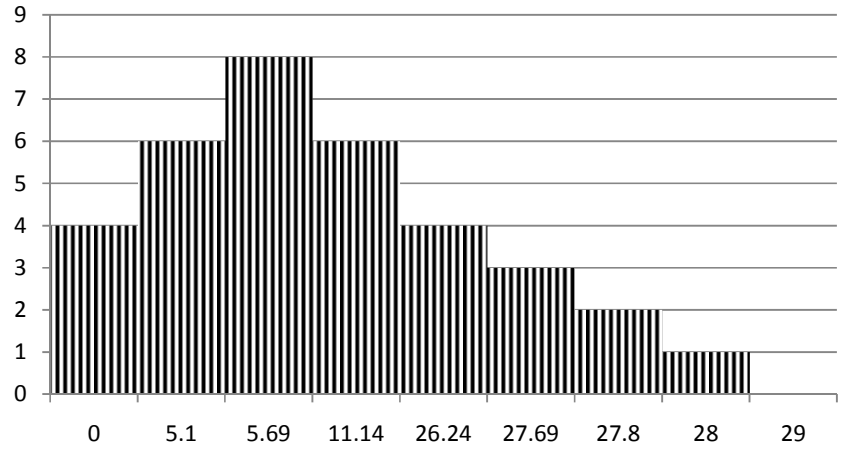

Figure 4. Number of virtual machines opened and closed over time with two types of threshold solution

The result of the number of virtual machines opened and closed over time with solutions using two types of resource thresholds is shown in Figure 4. Although the number of VMs added is greater but the average virtual machine cost per request is only 12.7 VM seconds, has dropped a

lot although the application has high memory requirements. Meanwhile, the average completion time of a request has dropped to 11.7 seconds. This demonstrates that the amount of resources has been added in time, even for applications with high demand for resources

5. CONCLUSIONS

Actually the cloud-based applications are varied and erratic, difficult to forecast. The results in the paper show that when the application changes, the resource consumption characteristic will change, which makes the determination of the overload situation inaccurate. If the overload is incorrect, the auto scaling process does not guarantee SLA and resource wastage. Auto scaling solutions that use low-level metrics if only using the threshold of CPU resources will have the risk of over-forecasting the wrong load. Therefore, the combination of resources to predict overload in auto scaling solutions is necessary. The combination of both the CPU and Memory thresholds proposed in this paper has reduced the cost of resource consumption and shortened the response time to a request. This achievement is due to the proposed solution that mitigates the effects of the change in resource consumption characteristics of the application.

REFERENCES

[1] Nilabja Roy, Abhishek Dubey, Aniruddha Gokhale (2011), Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting, The IEEE 4th International Conference on Cloud Computing.

[2] Anshuman Biswas, Shikharesh Majumdar, Biswajit Nandy, Ali El-Haraki (2017), A hybrid auto-scaling technique for clouds processing applications with service level agreements, Journal of Cloud Computing: Advances, Systems and Applications.

[3] A. Ali-Eldin, J. Tordsson, and E. Elmroth (2012), An Adaptive Hybrid Elasticity Controller for Cloud Infrastructures. In IEEE NOMS

[4] H. Fernandez, G. Pierre, and T. Kielmann (2014), Autoscaling Web Applications in Heterogeneous Cloud Infrastructures. In IEEE IC2E, 2014.

[5] Amazon.(2018).Amazon Auto Scaling Service. Retrieved from http://aws.amazon.com/autoscaling/

[6] J. Huang, C. Li, and J. Yu. (2012) Resource prediction based on double exponential smoothing in cloud computing. In Intl Conf on Consumer Electronics, Communications and Networks, pages 2056–2060

[7] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, Adel Nadjaran Toosi (2017),Auto-scaling Web Applications in Clouds: A Cost-Aware Approach, Journal of Network and Computer Applications, Vol. 95, Pages 26-41.

[8] Pooja, Asmita Pandey (2014), Impact of Memory Intensive Applications on Performance of Cloud Virtual Machine, Proceedings of 2014 RAECS UIET Panjab University Chandigarh

[9] Hamid Arabnejad, Claus Pahl, Pooyan Jamshidi and Giovani Estrada (2017), A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling, The 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)

[10] Alexey Ilyushkin, Ahmed Ali-Eldin, Nikolas Herbst, Alessandro V. Papadopoulos, Bogdan Ghit, Dick Epema, Alexandru Iosup (2017), An Experimental Performance Evaluation of Autoscaling Policies for Complex Workflows, Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE)

[11] Chenhao Qu, Rodrigo N. Calheiros, Rajkumar Buyya (2018), Auto-Scaling Web Applications in Clouds: A Taxonomy and Survey, ACM Computing Surveys, Vol. 51, No. 4, Article 73.

[12] Nguyen Hong Son, Nguyen Khac Chien (2017), Load Balancing in Auto Scaling-Enabled Cloud Environments, International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol. 7, No. 5, pp 15-22.

AUTHORS

Ha Van Cu,received his master degree in Computer Science from Hong Bang International University in HCM City in 2014. Heis doing a PhD candidate in Computer Science. His research interests include cloud computing,information security and data science.

Nguyen Hong Son, received his B.Sc. in Computer Engineering from the University of Technology in HCM city, his M.Sc. and PhD in Communication Engineering from the Post and Telecommunication Institute of Technology Hanoi. His current research interests include communication engineering, network security, computer engineering and cloud computing.