An Efficient Elasticity Mechanism for Server-Based Pervasive Healthcare Applications in Cloud Environment

Tushar Bhardwaj
Research Scholar, Electronics & Computer Discipline, IIT Roorkee, India
tushariitr1@gmail.com, tus19.dpt2014@iitr.ac.in

Subhash Chander Sharma
Professor, Electronics & Computer Discipline, IIT Roorkee, India
scs60fpt@iitr.ac.in, subhash1960@rediffmail.com

Abstract - The server-based e-health applications generates external workload (number of client requests) during the application's execution respectively. The main challenge in hosting e-Health applications in the Cloud environment is that there is no optimality of when and what amount of cloud's virtual resources should be leveraged in order to deliver non-stop and real-time pervasive healthcare services. In this paper, authors showcases the implementation of elasticity for server-based pervasive healthcare applications. The authors have extended CloudSim toolkit and designed a elastic framework, that provides elasticity for the e-health applications. To the best of our knowledge, this is the first study that uses CloudSim toolkit in addressing horizontal elasticity for server-based applications. The experimental results show that the proposed elasticity mechanism delivers better response time, finish time, and an efficient resource utilization as compared to the existing resource allocation approaches.

Keywords - Cloud Computing, Elasticity, Server-based Pervasive Healthcare Applications, Resource Utilization, CloudSim, Quality of Service

I INTRODUCTION

Due to the exponential increase in the cost of healthcare services, the healthcare organizations are bound to adopt a new implementation models in which the pervasive e-health applications are hosted in the Cloud Computing infrastructure. In this model, the e-health service providers (HSP) will be able to deploy e-health services rapidly on-the-fly basis. An e-Health service comprises of three basic characteristics [1]: *(a) Sensor Data Acquisition Time :* The time required to collect the data is called as acquisition time. This time is independent of the cloud service used. *(b) The Propagation delay :* It is the time taken between the sending and receiving of the information at the e-Health service side [2]. *(c) The Processing Time :* It is the time required to process the received information at the cloud server end. It comprises of compute, storage, I/O, and send notifications etc. The Cloud service provider requires to optimize the propagation delay and the processing time in order to fulfill the Quality of Service (QoS) parameters, as requested by the end-users. In this study, we mainly focus on the *processing time* of the e-Health application in cloud.

The processing time of the server-based e-health application depends on various factors. *First:* the server-based workload depends on the number of external user requests. *Second:* In server-based applications the elastic monitoring module generally collects the information from the virtual machine (VM) where the application is running. *Third:* The server-based applications consumes the computing resources as per the workload variation. The main challenges for hosting these types of e-Health applications in the Cloud environment is that there is no optimality of when and what amount of cloud's virtual resources should be leveraged in order to deliver non-stop and real-time pervasive healthcare services so as to respect the (QoS) parameters. The current elasticity mechanisms provided by various cloud providers is based on monitoring of the incoming workload (number of user requests) and few studies focuses on the resource utilization (CPU, memory, storage, I/O requests etc) of the virtual machine hosting a particular application over time [3,4].

Keeping in mind the limitations of existing elasticity mechanisms, the contributions of this paper are enlisted below :
(a) Elasticity mechanism for server-based pervasive healthcare applications: In server-based applications, the elastic controller node is capable of scheduling, monitoring, analyzing, the load of virtual machines running the e-health applications. In addition, it executes the elastic actions as per the output of analysis module.
(b) *Elastic Framework:* To realize the construction of an elasticity mechanism for e-health applications, we design and develop a framework built over CloudSim [5] toolkit. The existing CloudSim implementation neither supports pervasive healthcare ecosystem, nor the elasticity mechanisms. To the best of our knowledge, this is the first study that (i) uses CloudSim toolkit in addressing the elasticity mechanism, and (ii) elasticize server-based pervasive healthcare applications.
(c) Results: The experimental results shows the effectiveness in two-folds, first the successful implementation of elasticity mechanism and second the proposed elasticity mechanism delivers better response time, finish time, and an efficient resource utilization as compared to the existing resource allocation approaches.

The rest of the paper is structured as follows. In Section 2, the authors have highlighted the related work for dynamic resource allocation policies in cloud computing environment. Section 3 describes the concept and working of elasticity mechanism for server-based pervasive healthcare applications. Section 4 describes the experimental results of the proposed elasticity mechanism and the working efficiency of the elastic framework. Finally, the conclusive remarks and directions of future work are outlined.

0-7695-6406-2/17/$31.00 ©2017 IEEE
DOI 10.1109/HPCCWS.2017.00016

II RELATED WORK

In this section, we have tried to showcase the work done related to dynamic resource allocation in Cloud Computing for server-based e-Health applications. In [9], the study showcases the challenges inherent in resource allocation in distributed cloud. There are four main challenges: resource modelling, resource offering and treatment, resource discovery and monitoring and resource selection and optimization. In [6], the authors proposes a mechanism to reallocate sensors to various servers to minimize the quantity of extra servers with Number of sensors, Number of idle servers being the parameters optimized. In [7], an auto scaling mechanism of virtual resources on the basis of Transmission rate of sensors to handle the various data volumes and type is proposed. In [8], Dynamic scale out of virtual machines to maintain the response time of the user requests with Response Time, Computing worker nodes (VMs) as parameters. In [1] an auto-scaling mechanism (MOSt++) on the basis of Percentage of violated requests that allows to calculate the best provisioning plan for complex e-health service requests. In [10], A provisioning technique that automatically adapts to the incoming workload changes related to the application and guarantees the QoS to the end-users. The parameters optimized are VM hours, Requests Rejection Rate, Resource Utilization, Response Time. In [13], shows a framework which is execution of streamlining with Dynamic Resource Allocation (DRA) managing virtualization machines on physical machines. The outcomes affirmed that the virtual machine which stacking turns out to be too high, it will naturally relocate to another low stacking physical machine without administration intruding. In [11], presents an elastic controller that enables the application to adjust the virtual resources automatically on the basis of internal structure and load on the virtual machine hosting that application. In [12], shows that the greedy strategy is the least, the time of the random strategy is the largest, and the time of the three sequence strategies is moderate. In [14], A threshold-based dynamic resource allocation plan for distributed computing that powerfully designate the virtual resources (virtual machines) among the distributed computing applications in view of their heap changes (rather than designating resources expected to meet crest requests) and can utilize the limit technique to advance the choice of resources reallocation.

III ELASTICITY MECHANISM FOR SERVER-BASED PERVASIVE HEALTHCARE APPLICATION

To realize the proposed model, we partially adopt the iterative, stepwise, and functional concepts of autonomous computing proposed in [16]. This module consists of two very important phases : Monitor and Analyzer.

A. Monitor Phase

Initially, the average load on all the running virtual machines (VMs) are continuously monitored. It can be simply done by adopting a default resource monitoring software provided by public clouds [15]. The main drawback and reason to not use these type of software is that it monitors the resources at a certain interval with predefined performance metrics.

Therefore, we have implemented monitoring locally on each VM. This phase calculates the load on each virtual machine over a given time. Load of a virtual machine is the total length of tasks that are assigned to the specific VM as given in equation(1).

$$Load[VM, time(t)] = \frac{TotalNumberofTasks(T), time(t)}{ProcessingspeedofVM(mips), time(t)} \quad (1)$$

B. Analyzer Phase

It collects the outputs from the monitoring phase and predicts the future resource utilization of each virtual machine to meet the QoS and SLA requirements. If the VM is incapable to handle the incoming data packets then the application provisioner instantiates few more instance of VMs (the number of new VMs are provided by *Algorithm 1*) and transfer the tasks to it. On other hand, if the designated VMs are under loaded, then this phase directs the application provisioner to shut down certain VMs.

The proposed approach is modeled using a network of queues. The virtual machine provisioner is modelled to have a M/M/∞ request handling queuing station. Whereas, each virtual machine has an M/M/1/k queue. The parameter k, the queue size, can be estimated and defined according to the ratio of negotiated response/service($T_{service}$) time to the execution time (T_{exec}), equation 2.

$$k = Tservice / Texec \quad (2)$$

IV RESULTS

In this section, authors have discussed about the experimental results for the proposed elasticity mechanism for the server-based e-Health applications.

We have assumed that each incoming data packet requires 100 seconds to be processed in an idle VM. The SLA parameters are fixed with the following limits; *MaxResponseTime = 10 seconds, MinResponseTime = 5 seconds, MaxRejectionPercent = 5, vmLoadMAX = 70%, vmLoadMIN = 20%*. For uniformity, the virtual machines are set with a processing speed of 100 million instructions per second (MIPS) and the task which gets executed whenever each data packets receives by VM is having 1000 million instructions (Mis). Figure 1 showcases the results of the experiments conducted with the help of the proposed elasticity mechanism. The proposed model is labelled as Elastic, whereas static resource allocation policies are labelled as Static-*.

(a)Data Packets / Tasks: Figure 1(a), shows the number of data packets generated and corresponding tasks. The WBAN node sends data packets every 5 seconds based on the Poisson distribution rate. We run the simulation for 30 minutes, hence a total of 300 times the node sends the data packets. *Note : Here task is the process of saving each data packet to the repository.*

(b)Data Packets Rejection : The proposed elastic resource allocation does not guarantees zero percent rejection rate of the data packets. We started the experiment with static number of VMs ranging from 5 VMs to 20 VMs. The data packets rejection percentage observed from figure 1(b) is 26.943 (for 5 VM), 10.416 (for 10 VM), 1.986 (for 15 VM), zero (for 20 VM), 1.17 (for Elastic). Therefore, when we leverage the computing power of 20 VMs for handling data packets the rejection rate comes to be zero. But, this incurs certain QoS parameters that our proposed elastic mechanism overcomes.

(c)Variation in Virtual Machines : Figure 1(c) showcases the

number of virtual machines running. In this experiment, the number of VMs in the data center varied between 5 and 49, with an insignificant rejection rate. It has been observed that number of VMs required at peak time is 49, which is larger than 20 VMs required in Static allocation with zero rejection rate. But, the next section justifies the importance of the elastic module. Figure 1(d) describes the variation of the virtual machines over the course of processing the total tasks.

(d)Response time : The proposed approach delivers the minimum response time for processing the incoming data packets. This is a huge advantage in terms of pervasive healthcare scenario, where the response time is very prominent requirement. Figure 1(e) illustrates the variation in response time. Static (5 VM) takes the maximum response time (176.175 seconds), whereas the proposed Efficient Elastic module takes the minimum (12.812 seconds).

(e)Virtual Machine Hours : Figure 1 (f) describes the variation in VM hours utilised for each allocation policy. It can be clearly observed that the proposed module have the minimum number of VM hours (2.833 hours), whereas static 20 VMs takes highest number of VM hours (3.691 hours).

(f)Finish time : The proposed approach also delivers the minimum finish time for processing the incoming data packets. Figure 1 (g) illustrates the variation in finish time. Static (5 VM) takes the maximum finish time (144.733 seconds), whereas the proposed elastic module takes the minimum (34 seconds).

V. Conclusions And Future Work

To conclude, this study provides an efficient resource allocation mechanism at the cloud end which ensures the optimal use of virtual resources and delivers a real-time healthcare services to the users. The proposed approach is modeled using a network of queues. The modeling parameters of those queues are obtained via the monitoring phase and elastic resource allocation module.

The results are presented in Section IV, which justifies that the proposed approach is more suitable in providing elasticity to the server-based pervasive health-care applications. It can be observed that there is a trade-off in the rejection percentage to the response and finish time. In addition, the proposed approach delivers better resource utilization in terms of virtual machine hours. In future, authors are intended to work towards a proactive resource allocation mechanism that allocates and de-allocates the computing resources at cloud end. We have already set up a cloud computing laboratory using OpenNebula. Therefore, authors are intended to develop the same framework using the real cloud computing infrastructure and performs experiments in the real world in the future.

References

[1] Rachkidi E, Cherkaoui EH. Towards Efficient Automatic Scaling and Adaptive cost-optimized eHealth Services in Cloud 2015.

[2] Propagation delay, https://en.wikipedia.org/wiki/Propagation_delay.

[3] Vaquero LM, Rodero-merino L, Lyon LIPENS, Group GA. Dynamically Scaling Applications in the Cloud 2011;41:45–52.

[4] Wang L, Zhan J, Shi W, Member S, Liang Y. In Cloud , Can Scientific Communities Benefit from the Economies of Scale ? 2012;23:296–303.

[5] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw - Pract Exp 2011;41:23–50. doi:10.1002/spe.995.

[6] Nguyen N, Hasan Khan MM. A closed-loop context aware data acquisition and resource allocation framework for dynamic data driven applications systems (DDDAS) on the cloud. J Syst Softw 2015;109:88–105. doi:10.1016/j.jss.2015.07.026.

[7] Ahn YW, Cheng AMK, Baek J, Jo M, Chen HH. An auto-scaling mechanism for virtual resources to support mobile, pervasive, real-time healthcare applications in cloud computing. IEEE Netw 2013;27:62–8. doi:10.1109/MNET.2013.6616117.

[8] Pandey S, Voorsluys W, Niu S, Khandoker A, Buyya R. An autonomic cloud environment for hosting ECG data analysis services. Futur Gener Comput Syst 2012;28:147–54. doi:10.1016/j.future.2011.04.022.

[9] Endo P, Palhares ADA, Pereira N, Goncalves G, Sadok D, Kelner J, et al. Resource allocation for distributed cloud: concepts and research challenges. IEEE Netw 2011;25:42–6. doi:10.1109/MNET.2011.5958007.

[10] Calheiros RN, Ranjan R, Buyya R. Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments 2011:295–304. doi:10.1109/ICPP.2011.17.

[11] Galante G, Carlos L, Bona E De. The Journal of Systems and Software A programming-level approach for elasticizing parallel scientific applications. J Syst Softw 2015;110:239–52. doi:10.1016/j.jss.2015.08.051.

[12] Xu X, Hu H, Hu N, Ying W. Cloud task and virtual machine allocation strategy in cloud computing environment. Commun Comput Inf Sci 2012;345:113–20. doi:10.1007/978-3-642-35211-9_15.

[13] Yang C-T, Cheng H-Y, Huang K-L. A Dynamic Resource Allocation Model for Virtual Machine Management on Cloud. Grid Distrib Comput 2011;261:581–90. doi:10.1007/978-3-642-27180-9_70.

[14] Lin W, Wang JZ, Liang C, Qi D. A threshold-based dynamic resource allocation scheme for cloud computing. Procedia Eng 2011;23:695–703. doi:10.1016/j.proeng.2011.11.2568.

[15] W. Publishing, Xen ® Virtualization, 2007.

[16] J.A. Frey, F. Baitinger, J.M. Gdaniec, IBM Unified Resource Manager introduction and overview, IBM J. Res. Dev. 56 (2012) 16. doi:10.1147/JRD.2011.2179133.

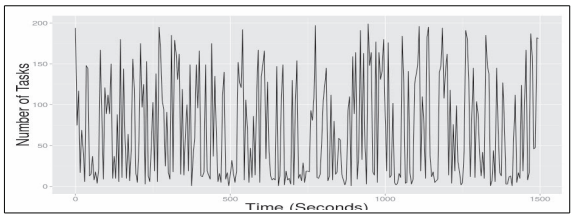
[17] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, M.N. Alsaleh, TeachCloud: a cloud computing educational toolkit, Int. J. Cloud Comput. 2 (2013) 237–257. doi:10.1504/IJCC.2013.055269.

Algorithm 1 : Efficient Resource Utilization

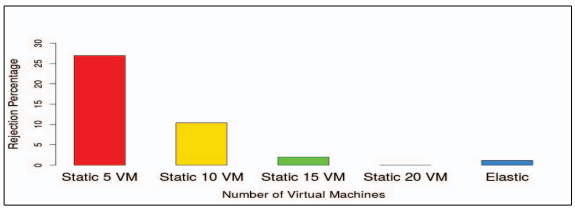
Data Given:
QoS parameters : Response Time, Makespan Time, Resource Utilization
SLA parameters : MaxResponseTime, MinResponseTime, MaxRejectionPercent, ,vmLoadMAX,vmLoadMIN.
MaxVM : Maximum number of virtual machines for handling incoming data packets
MinVM : Minimum number of virtual machines for handling incoming data packets
Output : currentnumberofVM, number of VM able to meet QoS.

1. currentnumberofVM = current number of virtual machines.
2. avgRejection = expected average rejection in a M/M/1/k queuing model given the transmission rate (λ) and average request execution

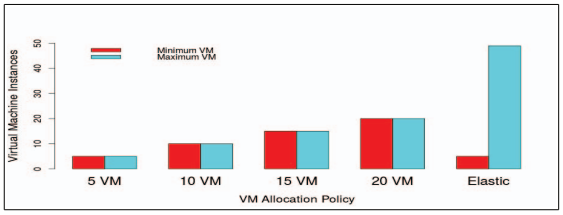
time(T_{exec}).
3. avgResponse = expected average response time in in a M/M/1/k queuing model given avgRejection,λ and T_{exec}.
4. avgVMLoad = expected average load on virtual machines.
5. oldNumberofVM = currentnumberofVM.
6. **if** (avgResponse > MaxResponseTime || avgRejection > MaxRejectionPercent || avgVMLoad > vmLoadMAX) **then**
7. Print"More Computing Resources Required"
8. currentnumberofVM = currentnumberofVM + currentnumberofVM/2;
9. **if** (currentnumberofVM > maxVM) **then**
10. currentnumberofVM = maxVM;
11. **end**
12. **end**
13. **else**
14. **if** (avgResponse < MinResponseTime && avgVMLoad < vmLoadMIN) **then**
15. Print"No Requirements"
16. currentnumberofVM = currentnumberofVM - minVM;
17. **if** (currentnumberofVM < minVM) **then**
18. currentnumberofVM = oldNumberofVM;
19. **end**
20. **end**
21. **return** currentnumberofVM;



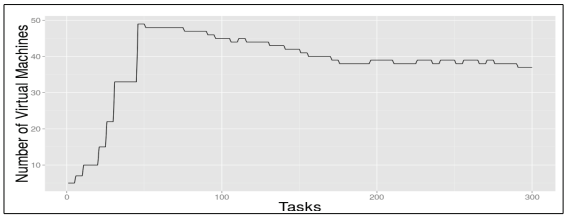
(a) Task rate in a server-based e-health application



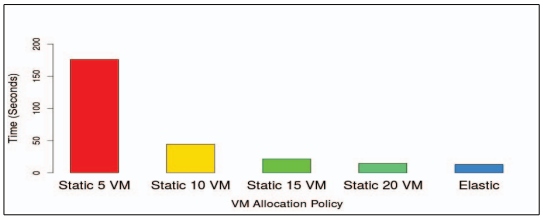
(b) Data Packets Rejection Percentage Comparison



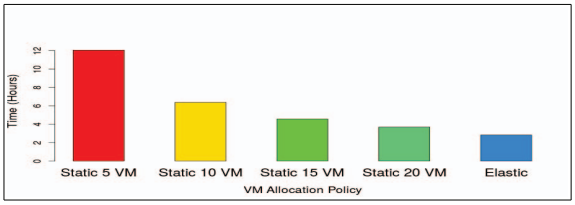
(c) Variation in number of Virtual Machines



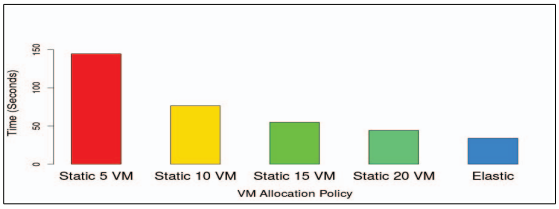
(d) Number of Virtual Machines in Elastic Environment



(e) Average Response Time Comparison



(f) Virtual Machine Hours Comparison



(g) Average Finish Time Comparison

Figure. 1. Performance of Proposed Elasticity Mechanism