Load Pattern Identifier (LoPI) for Dynamic Threshold based Auto-scaling in Cloud Environment

M. Kriushanth, Research Scholar
Department of Computer Science
St. Joseph's College (Autonomous)
Tiruchirappalli, TN, India.

L. Arockiam, Associate Professor
Department of Computer Science
St. Joseph's College (Autonomous)
Tiruchirappalli, TN, India.

***Abstract* — Cloud services are available anywhere, anytime with pay-as-you-go as on demand self-servicing basis. The numbers of services are also increasing with respect to the numbers of users. Scalability is the most important technique to provide the cloud services on demand. Scaling can be done in two methods as auto-scaling or dynamic scaling and manual scaling. Auto-scaling mechanism facilitates to scale up and scale down, scale in and scale out the computing resources at any point of time when demand increases. Without virtualization the cloud resources cannot be possible to get. Virtualization is the emulation of one or more workstations/servers within a single physical computer. In simple term, Virtualization is the emulation of hardware within a software platform. Full utilization of a physical server definitely directs to the physical server failure. Most of the CSPs applied the static threshold techniques in the physical server to overcome the failure. Dynamic threshold technique assists to operate physical server at the maximum and reduce the cost. Effective resource utilization may further reduce the service cost. Threshold values can be set in virtual machine apart from physical server. The load or workload may not be the same at all time. The sudden increase of the workload is called flash crowd. Load pattern identification helps to set the resources readily available when the request increases. The suitable workload identification helps the effectiveness of the scalability. To find out the load pattern, request rate and scaling size LoPI is used. The real time demand and the forecasted workload is compared with interpolation/Extrapolation and Correlation test. Rule engine is used to analyze the LoPI and generate the rule. Finally dynamic threshold is set to Auto-scale the cloud resources.**

Keywords* — *Cloud Computing, Scalability, Auto-scaling, Virtualization, Threshold, Workload.

I. INTRODUCTION

Cloud services are available anywhere, anytime with pay-as-you-go as on demand self-servicing basis. The numbers of services are also increasing with respect to the numbers of users. Scalability is the most important technique to provide the cloud services on demand. Scaling can be done in two methods as auto-scaling or dynamic scaling and manual scaling. Auto-scaling [1] mechanism facilitates to scale up and scale down, scale in and scale out the computing resources at any point of time when demand increases. Without virtualization the cloud resources cannot be possible to get. Virtualization is the emulation of one or more workstations/servers within a single physical computer. In simple term, Virtualization is the emulation of hardware within a software platform. This allows a single computer to take on the role of multiple computers [2]. The emulation of a server is called a virtual machine instance. The various types of VM instance are standard, high memory, high CPU and shared core [3].

Full utilization of a physical server definitely directs to the physical server failure. Most of the CSPs applied the static threshold techniques in the physical server to overcome the failure. Dynamic threshold technique assists to operate physical server at the maximum and reduce the cost [4], [5]. Effective resource utilization may further reduce the service cost. Threshold values can be set in virtual machine apart from physical server.

The workloads may not be the same at all time. Cloud user requests from small instance to a large number of users. The sudden increase of the workload is called flash crowd. The most popular CSPs are Amazon, Microsoft, Google, Rackspace, IBM and so on. In recent years the Infrastructure as a Service has changed and transformed lot. AWS provides different types of IaaS (Ex. CPU, GPU, Storage, Network, Memory, Container Event driven compute functions, Auto scaling and Load Balancing) to the end user [6].

Workload pattern identification helps to set the resources readily available when the request increases. The suitable workload identification helps the effectiveness of the scalability. To find out the pattern, request rate and scaling size LoPI is used. The real time demand and the forecasted workload is compared with interpolation/Extrapolation and Correlation test. Finally set the dynamic threshold is set to Auto-scale the cloud resources.

II. RELATED WORKS

Dynamic scalability enables users to quickly scale up or down underlying infrastructure. Several challenges arise when considering computing instances such as non-deterministic acquisition time, multiple VM instance types, unique billing models and user budget constraints. Deadline and budget constraint for cloud infrastructure to accommodate changing workload based on application level performance metrics job deadline. Ming et al. [7] have used windows Azure to test the experiment which takes 10 minutes to start an instance and shutting time is quite stable around 2-3 minutes. VM startup delay plays an important role in cloud Auto-Scaling mechanism. In their experiment, VM instances were billed by hours; fraction consumption of an instance hour was also counted as a full hour. Ming et al. [8] proposed architecture to finish the job before the deadline to bring out the cost effectiveness. VM startup delay could not only affect the performance, but also dominates the utilization rate and the cost for short deadline.

One of the goals of cloud computing is to allocate the resources that are needed for the customers and charge accordingly. The service providers like Amazon EC2 [9] currently offers 11 VM instance types like a standard machine for most types of application, high CPU and high memory used to finish the job before stipulated time or the deadline. Ming et al. [7] accomplished the goal by dynamically allocating/de-allocating VMs as a scheduled task on most cost efficient instances. The evaluation of the experimental results has shown the total cost saving from 9.8% to 40.4%.

The auto-scaling techniques are basically used to automate the scaling and to reduce the waiting time and cost. Tania et al. [10] proposed auto-scaling techniques for elastic application in cloud environments. Auto-scaling can be done by proactive and reactive scaling. Proactive is much more cost effective than reactive. The first approach is static threshold-based rules. When the CPU utilization has reached 70% for more than 5 minutes, it adds 2 instances. If it reaches 30%, it will reduce the instances. The performance metrics can be considered by request rate, CPU load or average response time.

Tania et al [4] compared the auto-scaling techniques for cloud environments. The scenario is a web application, deployed over a set of VMs. They focused on the load balancer and the business tier. Requests arrive to web application, load balancer will distribute requests among the VMs based on the set of policies such as random, round-robin and least-connection. Each task will be assigned to a single VM.

Static threshold based rules tested with set of threshold values. The results show the reactive techniques based on the CPU load for two different values and boot up time. Techniques are rules, Dynamic thresholds, Right scale, Dynamic threshold right scale and Integral controller.

Dynamic threshold based rules tested with reactive techniques are Moving Average (MA), Linear Regression (LR) and Exponential Smoothing (ES). The experimental results shows that the cost of VMs and Service Level Objective (SLO) violations of each mathematical model.

Mohan et al [11] proposed the threshold based auto-scaling of virtual machines in cloud environment. Cloud service cost can be reduced by effective utilization of the resources and resource wastage can be minimized. The application requirement may vary over time depends on many factors (Ex. Instance load). User may run different applications like simple application to some complex accounting. In such circumstances, if the VM capacity is fixed there is a possibility of mismatch between the VM capacity and application resource requirement. To rectify the issue a threshold based auto-scaling of virtual machines in which VMs will be dynamically scaled based on the application resource utilization (CPU and Memory) is considered.

Elasticity is one of the key governing properties of cloud computing that has major effects on cost and performance directly. Most of the popular infrastructure providers such as amazon web services, windows azure, Rackspace etc. work on threshold based auto-scaling. In current IaaS environments there are various other factors like “Virtual Machine (VM)-turnaround time”, “VM-stabilization time” etc. that affect the newly started VM from start time to request servicing time. If these factors are not considered while auto-scaling, it will affect directly on Service Level Agreement and users response time. So that these threshold should be a function of load trend, which makes VM readily available when needed. Karteek et al [12] developed an approach where the thresholds adapt in advance and these thresholds are function of all the above mentioned factors.

Web application providers have the potential to scale virtual resources up or down to achieve cost-effectiveness in the pay-per-use cloud business model have not yet been achieved. Jiang et al [13] proposed a optimal cloud resource auto-scaling scheme at the virtual machine level of web application. This scheme automatically predicts the number of web requests and discovers an optimal cloud resource demand with cost-latency trade-off based on the demand the scheme makes a resource scaling decision that is up or down in each time unit re-allocation.

An enterprise or tenant will use cloud service interfaces to acquire or release resources directly. This process can be automated by a CSP by providing auto scaling capability where a tenant set policies, which indicates under what condition resources should be auto scaled. Typical solutions are naïve causing spurious auto-scaling decisions. They are based on only thresholding triggers and the thresholding mechanism themselves are not cloud-ready. In a cloud, resources from three separate domains, compute, storage and network, are acquired or release on demand. The typical solution resources from these three domains are not auto-scaled in an integrated fashion. Integrated auto-scaling prevents further spurious scaling and reduces the number of auto scaling system to be supported in a cloud management system. A cloud resources auto scaling system called integrated and automatic cloud resource scaling (IACRS) proposed by Masum et al [14], that addresses and overcome the limitations of thresholding mechanism, different domain auto-scaling performance metrics, compute, storage and network resources. The IACRS supports cloud ready advanced thresholding mechanism, integrates performance metrics from multiple domains in making scaling decisions an scales network resources in combining with resources from the other two domains ad vice versa.

Scalability is a key point for the success of any business involving the web and providing services to end user request that may vary drastically from one time to another. Sizing a system to provide performance guarantees under peak traffic can be cost prohibitive. Fillipo et al [15] extended flexiscale public cloud with auto scaling mechanisms and compared with amazon services. The analysis identified the useful patters for the execution of web application in the cloud and at underflying the critical factors that affect the performance of the two service providers.

Performing scaling activities manually is time consuming so cloud system provides solutions to automatically scale out or predefined polices. Polices are usually composed by a set of scaling rules, each of which is defined by one or more conditions and a set of actions to be performed When those conditions hold. Conditions are typically defined specifying thresholds over a set of performance metrics including CPU utilization, disk I/O and bandwidth usage.

Beloglazov at al. [16] proposed an approach using Best Fit Decreasing algorithm to allocate VMs among hosts. The advantage of this approach is the stable utilization of resources while the disadvantage is that threshold policy may results in unnecessary migration of VMs and switching on/off hosts.

III. MOTIVATION

Static threshold values are used if the CPU load is more than 70% scale out or add instance, if CPU load is less 30%, then scale in or reduce instances. It is very difficult to set the exact threshold values and this must be done manually. An incorrect adjustment cause oscillation in the number of VMs and it leads to bad performance. Threshold values could be of any combination, in literature [Set, 14] shows that threshold values for low CPU utilization is 45% and high utilization is 75%. Memory utilization threshold values are 35% and 65% respectively. From the literature [Hai, 13] shows that, the best suitable upper threshold values are from 70 % to 90%. The most significant drawback of the threshold based techniques is the difficulty of setting suitable threshold values [Ali, 14]. An auto - scaling technique is used to increase and decrease the resources with respect to the present workload scenario. Dynamic threshold values helps to utilize the resources when the workload oscillation taking place, it reduces the waiting time of CU and overprovisioning.

The following literatures motivates that, threshold should be a function of load trend, which makes VM readily available when needed [Kar, 13]. Setting the exact upper CPU threshold value is difficult, and that control the triggering of the auto scaling policy [Hai, 13]. Rule-based auto-scaling methods are simple, it is difficult for them to satisfy dynamic load patterns [Hye, 13]. Many authors have specified, If the CPU utilization reached upper threshold value for more than 2 to 5 minutes, add the resources. To overcome unnecessary on/off hosts and VM migrations.

IV. CIRCUMSTANCES SCENARIO

A. *Auto-scaling with Unlimited Resources*

Sufficient resources are available in CSP, but the user request is not up to the level as expected. Over capacity leads to the resource wastage and request mismatch.

B. *Auto-scaling with Unlimited Request*

More cloud user requesting for the service, CSP has only limited resources. In such condition the resources provisioned up to maximum level, if the available resources are utilized entirely, to scale the new resource is time consuming. By the time new user request will be in request queue.

C. *Workload is Higher than the Upper Threshold Value*

Request is coming gradually at a particular point of time. Ratio of the request and resource, is higher than the upper threshold value (ex. 85% of request coming in but upper threshold value is 80%). For the fraction of user request, CSP has to provide service from newly generated resources. In such condition 5% of the requests have to accommodate in available resources if the threshold is dynamic.

D. *Workload Pattern*

User requests dynamically changes every time from small, medium, large and extra-large. Small instances take lesser time to response compared to large instances. To scale the resources, CSP need to find out the scaling size and pattern of the load.

E. *Sudden Increase and Decrease in Workload*

When sudden workload increases or decreases, it is very difficult to handle for CSP to provision the resources. CSP doesn’t have sufficient resources at that particular point of time. CU has to wait in a queue to avail the service. That definitely leads to the SLA violation and QoS service degradation.

F. *Load is higher/lower than the Forecasted Demand*

The workload is lower or higher than the forecasted demand, there is mismatch between the demand or capacity. Resource under provisioning causes the overhead to the CSP and overprovisioning causes the SLA violation and QoS degradation.

V. OBJECTVE AND METHODOLOGY

A. *Objective*

The main objective in this chapter is to find out the scaling size and pattern of the workload to set the threshold values dynamically.

B. *Methodology*

User request from 1 to n coming to CSP, Load balancer receives the request and looking for the available resources. If the resources are available, Load balancer sends the request to the resource pool to provision the resources. The Load Pattern identifier keeps on monitoring the workload to find out the pattern and scaling size. Rule engine receives the information from LoPI and generate the rule to add, remove or manage with available resources with dynamic threshold values. Finally the resources are provisioned to the user. Fig. 1. describes the methodology followed in this chapter.

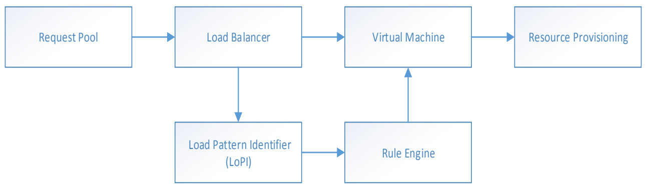

Figure 1. Dynamic threshold based Auto-scaling methodology

VI. DYNAMIC THRESHOLD BASED AUTO-SCALING

All cloud resources are provided based on static threshold values. Workload may be increasing gradually or there must be some oscillations. Finding out workload pattern and scaling size could set the threshold values dynamically and optimally utilize the resources. The dynamic threshold based auto-scaling approach using the load pattern identifier implied in this paper is shown in Fig. 2. It focuses to set the threshold values dynamically according to the pattern of the workload.

A. *Request Pool (RP)*

The cloud user requests come through the request pool. The cloud user's credentials and requests will be stored in resource pool. It will categorize all the requests and finally the requests forwarded to the load balancer.

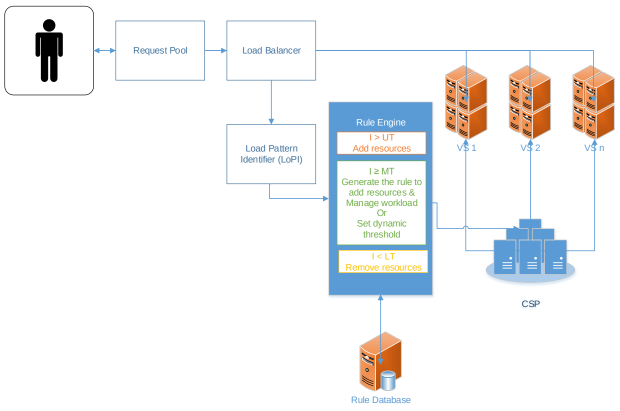

Figure 2. Dynamic Threshold Based Auto-scaling

B. *Load Balancer (LB)*

Load balancing is a technique to enhance resources, developing parallelism, exploiting throughput invention and to reduce response time through the appropriate distribution of the application. Load balancer is to send the user request (or) workload to the available cloud resources.

C. *Load pattern identifier (LoPI)*

The IaaS instances are micro, small, medium, large and extra-large. AWS [AWS, 15] instances of five categories are General purpose, Compute optimized, GPU instances, Memory optimized instances and Storage optimized.

The order of the load coming to the request handler is called load pattern. The light weight requests are served within seconds or minutes [5] and heavy weight requests are served in minutes. CSP cannot expect the same workload at all time. It won't be possible to serve in a shorter period of time unless if they are proactive. To find out the pattern of the load during heavy workload LoPI approach is used. It keeps track of the cloud user request and number of CU requests being served at a time interval. LoPI identifies the load condition and trigger the command to rule engine to avoid the CU waiting in queue for service.

It keeps on monitoring the load in a load balancer. LoPI find out the number of CU request in load balancer and number of request served at a particular point of time. The pattern keeps on changing with respect to the load conditions. It separates the type of load into small, medium, large and extra-large, identifies the CPU, memory and storage load conditions are also monitored. Eventually, it updates the load status to the rule engine.

1) *Interpolation/Extrapolation:*

UK natural grid dataset realtime data is condidered for the demand [17]. Set of time searies forecasting methods applied and the more accurate with good fitness statistics method has choosen. The forecasted demand [18] and time has taken for comparison with the present load pattern. Lagrange's polynomial Interpolation and Extrapolation is applied to find out the under and over provisioning.

$$P(x) = \frac{(x-x2)(x-x3)\dots(x-xN)}{(x1-x2)(x1-x3)\dots(x1-xN)} y_1 + \frac{(x-x1)(x-x3)\dots(x-xN)}{(x2-x1)(x2-x3)\dots(x2-xN)} y_2 + \dots + \frac{(x-x1)(x-x2)\dots(x-xN-1)}{(xN-x1)(xN-x2)\dots(xN-xN-1)} y\mathrm{N} \qquad (1)$$

The interpolating polynomial of degree N-1 through the N points y_1 = f(x_1), y_2 = f(x_2),…., y_N = f(x_N) is given explicitly by Lagrange's classical formula. There are N terms, each a polynomial of degree N-1 and each constructed to be zero at all of the x_i, except one, at which it is considered to be y_i.

2) *Correlation*

Correlation, Partial Autocorrelation and Autocorrelation test is applied to one day forecasted demand and the load. The test results are showed in the following figures. One day load and the forecasted load is shown in Fig. 3.

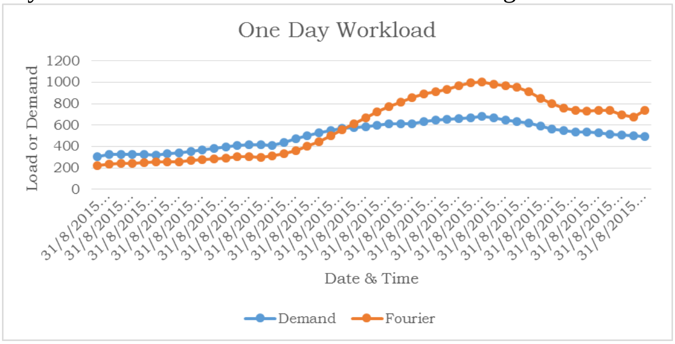

Figure 3. Load and Foresacted load

Pearson, Prearman and Kendall correlation test summary of statistics are shown in Table 1, Correlation matrix in Table 2, p-values in Table 3 and Coefficient of determination in Table 4.

TABLE I. SUMMARY OF STATISTICS

Variable	Observations	Obs. with missing data	Obs. without missing data	Minimum	Maximum	Mean	Std. deviation
Kendall							
Demand	48	0	48	307.000	681.000	505.458	118.654
Fourier	48	0	48	222.409	1002.334	590.791	277.822
Spearman							
Demand	48	0	48	307.000	681.000	505.458	118.654
Fourier	48	0	48	222.409	1002.334	590.791	277.822
Pearson							
Demand	48	0	48	307.000	681.000	505.458	118.654
Fourier	48	0	48	222.409	1002.334	590.791	277.822

TABLE II. CORRELATIN MATRIX

	Variables	Demand	Fourier
Kendall	Demand	1	0.859
	Fourier	0.859	1
Spearman	Demand	1	0.956
	Fourier	0.956	1
Pearson	Demand	1	0.945
	Fourier	0.945	1

TABLE III. P-VALUES

	Variables	Demand	Fourier
Kendall	Demand	0	0.000
	Fourier	< 0.0001	0
Spearman	Demand	0	0.000
	Fourier	< 0.0001	0
Pearson	Demand	0	0.000
	Fourier	< 0.0001	0

TABLE IV. COEFFICIENTS OF DETERMINATION

	Variables	Demand	Fourier
Kendall	Demand	1	0.739
	Fourier	0.739	1
Spearman	Demand	1	0.914
	Fourier	0.914	1
Pearson	Demand	1	0.893
	Fourier	0.893	1

Fig. 4. to Fig 7. Shows the Partial and Autocorrelation of demand and Fourier demand.

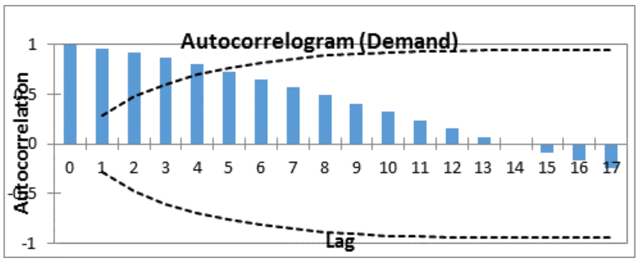

Figure 4. Autocorrelation for Demand or load

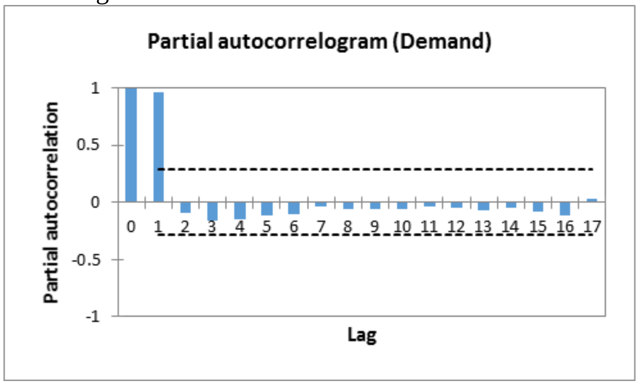

Figure 5. Partial Autocorrelation for Demand

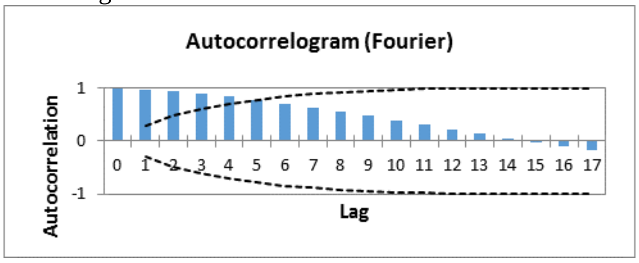

Figure 6. Autocorrelation for Fourier

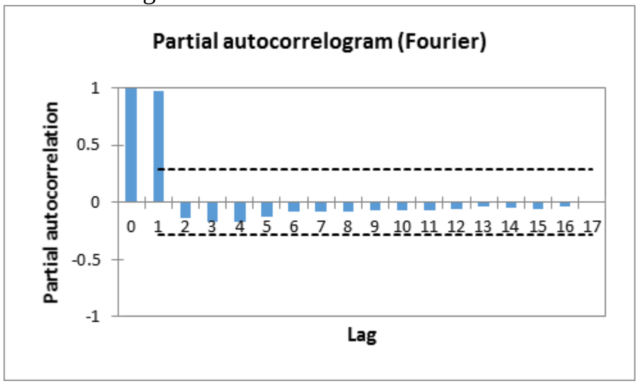

Figure 7. Partial Autocorrelation for Fourier

Table 5 shows the Normality test and white noise tests of Fourier and Demand.

Statistic	DF	Value	p-value
Box-Pierce	6	30.523	< 0.0001
Ljung-Box	6	35.410	< 0.0001
McLeod-Li	6	295.686	< 0.0001
Box-Pierce	12	185.625	< 0.0001
Ljung-Box	12	239.565	< 0.0001
McLeod-Li	12	442.520	< 0.0001

Fig. 8 shows the cross correlation of demand and Fourier.

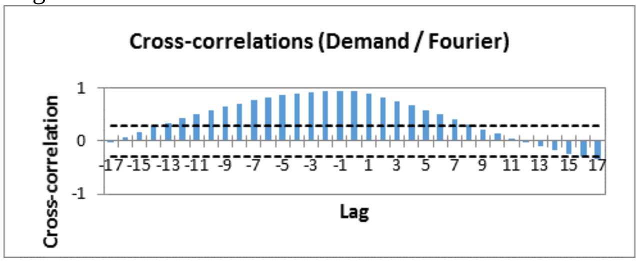

Figure 8. Cross-correlations for Demand and Fourier

D. Scaling size

Workload consists of three types, CPU, Memory, Storage and Network. CPU load is measured as

$$\mathrm{CPU} = \sum_{k=1}^{n} Ck \,/\, \mathrm{N} \tag{2}$$

Where, n is the number of nodes, CPU utilization represented as Ck. Memory load is measured as

$$\mathrm{M} = \sum_{k=1}^{n} \mathrm{Mem}\mathit{Use} \,/\, \mathrm{T}\mathit{Mem} \tag{3}$$

Where, memory load consists of memory used and total memory. Storage load measured as

$$\mathrm{ST} = \sum_{k=1}^{n} \mathrm{St} \,/\, \mathrm{TSto} \tag{4}$$

Where, storage (ST) load consists of storage used and the total storage. Network load measured as

$$\mathrm{NT} = \sum_{k=1}^{n} \mathrm{St} \,/\, \mathrm{TSto} \tag{5}$$

Where, network (NT) load consists of network used and the total network.

E. Rule engine (RE)

Rule engines' activity is to receive the updated information from LoPI and to generate dynamic rule. According to the LoPI direction, rule engine decides to manage the workload till the new VM is ready. Upper limit of the threshold value is 80%, lower limit is 20% and indication threshold limit is 70% and 30%. The middle threshold value is 50%, if the requests reach 70%, a rule is applied to check the ability to handle the requests. If the available resource is enough, it manages the workload. If it exceeds 70% of threshold value, apply the dynamic rule to increase the resources. If it reaches 80% threshold value, add resources when request goes below 20% threshold value remove underutilized resources. If the workload is bit higher than the resources available, set the dynamic threshold values to accommodate the requests with available resources. Meanwhile, direct and deal the request with available resources.

Auto-scaling mechanism works when the rule engine triggers the dynamic rule. Auto-scaling is used to add and remove resources as per the scaling size.

F. Rule Datbase (RD)

The generated rule is stored in rule database for future reference. The generation of rule for the similar dataset for future is time consuming, to avoid delay in the rule generation process the rule are stored in the RD.

VII. Conclusion and Future Work

Auto-scaling mechanism used to make resources available according to the users need or to provide a cost effective service. Identifying the exact amount of resources needed, at a particular point of time is an important factor for the better resource utilization, it might further reduce the cost of the service. Most of the CSPs used the static threshold based rules to provide the resources as fast as possible. To avoid the problems in static threshold, the dynamic threshold based auto-scaling using the load pattern identifier (LoPI) is proposed. Our future work is to validate the proposed method to a large set of real time data.

References

[1] Amazon Auto Scaling in Cloud Computing", http://aws.amazon.com/autoscaling/30.05.2012

[2] The Art of Service, "Cloud Computing Specialist Certification Kit – Virtualization".

[3] Google Instance types", [Online]. https://developers.google.com /compute/pricing/30.05.2013.

[4] Tania Lorido-Botran, Jose Miguel-Alonso and Jose A. Lozano," Comparison of Auto-scaling Techniques for Cloud Environments", Proceedings of Intelligent Systems group, Actas de las XXIV Jornadas de Paralelismo, ISBN 978-84-695-8330-2.

[5] M. Kriushanth and L. Arockiam, "Cost Aware Dynamic Rule Based Auto Scaling of Infrastructure as a Service in Cloud Environment", In proceedings of International Conference on Advanced Computing & Communication Technologies for High Performance Application (ICACCTHPA'14), International Journal of Computer Applications (IJCA), June 2014, ISSN 0975-8887, pp. 35-40.

[6] Aws instance types, http://aws.amazon.com / 08.05.2015

[7] Ming Mao, Jie Li and Marty Humphery, "Cloud Auto-scaling with Deadline and Budget Constraints", Proceedings of 11th IEEE/ACM International Conference on Grid Computing, 2010, ISBN 978-1-4244-9349-4, pp 41-48.

[8] Ming Mao and Marty Humphery, "Auto-scaling to Minimize Cost and Meet Application Deadline in Cloud Workflows", Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE 2011, E-ISBN 978-1-4503-0771-0, pp 1-12.

[9] Amazon Web services, http://aws.amazon.com/what-is-cloud-computing/ 05.12.2013

[10] Tania Lorido-Botran, Jose Miguel-Alonso and Jose A. Lozano,"Auto-scaling Techniques for Elastic Applications in Cloud Environments", Technical Report EHU-KAT-IK-09-12, Department of Computer Architecture and Technology, University of the Basque Country, September 5, 2012, pp1-44.

[11] Mohan Murthy M K, Sanjay H A and Anand J, "Threshold Based Auto Scaling of Virtual Machines in Cloud Environment", Inproceedings of the International Conference on Network and Parallel Computig, Ilan, Taiwan, September 18-20, 2014, LNCS Vol 8707, Print ISBN 978-3-662-44916-5, Online ISBN 978-3-662-44917-2, ISSN 0302-9743, pp 547-256.

[12] Kartheek Kanagala and K. Chadra Sekaran, "An Approach for Dynamic Scaling of Resources in Enterprise Cloud", In procedings of the IEEE international Conference on Cloud Computing Technology and Science, IEEE, 2013, ISBN 978-0-7695-5095-4, pp. 345-348.

[13] Jing Jiang, Jie Lu, Guangquan Zhang and Guodong Long, "Optimal Cloud Resource Auto-Scaling for Web Applications", 13th Internatinal Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM, 2013, ISBN 978-0-7695-5, pp. 58-65.

[14] Masum Z. Hasan, Edgar Magana, Alexandar Clemm and Sree Lakshmi D. Gugreddi, "Integrated and Autonomic Cloud Resource Scaling", IEEE, 2012, ISBN 978-1-4673-0269-2, pp. 1327-1334.

[15] Filippo Lorenzo Ferraris, Davide Franceschelli, Mario Pio Gioiosa, Donato Lucia, Danilo Ardagna, Elisabetta Di Nitto and Tabassum Sharif, “Evaluating the Auto Scaling Performance of Flexiscale and Amazon EC2 Clouds”, 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE, 2012, ISSN 978-0-7695-4934-7/12, pp. 423-429.

[16] Beloglazov A and Rajkumar Buyya, “Energy Efficient Allocation of Virtual Machines in Cloud Data Centers”, In proceedings of 10th International Conference on Cluster, Cloud and Grid Computing, IEEE/ACM, 2010, pp. 577-578.

[17] Data explorer UK National grid real time demand data, http://www2.nationalgrid.com/uk/Industry-information/Electricity-transmission-operational-data/Data-Explorer/. Dated – 17.08.2015

[18] M. Kriushanth and L. Arockiam, “User Demand Forecast (UDF) for Dynamic Schedule Based Auto-scling in Cloud Environment”, CSI - 50th Golden Jubilee Annual Convention, Springer, Scheduled to be held during 02nd - 05th December, 2015(Inpress).