Prediction based Auto Scaling for Cloud Applications

Chirag A. Patel1, J.S.Shah2, Hiren R. Patel3

1Ph.D. Scholar, Gujarat Technological University, Gujarat
2Ph.D. Supervisor, Gujarat Technological University, Gujarat
3Ph.D. Scholar, Gujarat Technological University, Gujarat

ABSTRACT

Objectives: Cloud computing provides highly scalable environment to applications. Due to dynamicity of application workload, static allocation of resources is not suitable. Scalability provides a way to increase or decrease resources of particular application at runtime. Static allocation of resources may lead to several problems like over provisioning and under provisioning. If resources are allocated statically by considering peak workload (over provisioning), majority of time, resources will be underutilized and users will have to pay more. If allocation is done based on average workload, it will not be possible to achieve performance objective while peak load (Under provisioning). Horizontal scalability is provided by means of adding or removing VM instances whereas, vertical scalability can be achieved by increasing or decreasing resources allotted to particular VM. Automatic scaling of resources based on workload change reduces provisioning costs and helps clients to achieve performance objectives. Methods: Automatic scaling can be done either by reactive way or by proactive way. In reactive auto scaling, resources are scaled as a reaction of some event. In our work, proactive auto scaling method is suggested based on application workload prediction. Time series analysis based ARIMA model is used for workload prediction. Findings: Proactive auto scaling increases or decreases application resources in advance based on prediction. This helps to improve application performance as well as to minimize SLA violations. It also reduces unnecessary cost incurred to cloud user. Application/Improvement: ARIMA based time series prediction is used to predict future workload. Scaling decision is taken based on predicted future workload as well as current application average response time.

Keywords: Server consolidation, SLA, Response time, ARIMA, Scalability, Cloud Computing, DCSim

1.INTRODUCTION

Cloud computing is an Internet based computing environment which offers various resources in form of services like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) etc. Users have to pay according to the consumption of particular service. One of the main features of Cloud Computing is that it provides highly scalable environment to applications i.e. Users can increase or decrease application resources as per demand. Scalability can be of two types- Horizontal and Vertical. Horizontal scalability is provided by means of adding or removing VM instances. Vertical scalability can be achieved by increasing or decreasing resources allotted to particular VM. Now a day, more and more applications are migrated to Cloud.

Resource assignment to cloud applications can be done either by statically or dynamically. Static allocation of resources may lead to several problems like over provisioning and under provisioning. If resources are allocated statically by considering peak application load (over provisioning), majority of time resources will be underutilized and users will have to pay more. If allocation is done based on average application load, it will not be possible to achieve performance objective while peak load (Under provisioning).

Automatic scaling of resources based on workload change reduces provisioning costs and helps clients to achieve performance objectives. The cloud provider, on the other hand, should attempt to consolidate load onto highly utilized physical machines, in order to reduce wasted power consumption [1].
Application auto scaling can be classified into two categories named - Reactive and Proactive. In case of reactive auto scaling, application resources are scaled based on present status whereas, proactive auto scaling uses sophisticated techniques for prediction of future demand and based on that scaling is done. In our work, proactive auto scaling method is suggested based on application workload prediction. Time series analysis based ARIMA model is used for workload prediction.

Determining and providing right amount of resources to cloud based applications is a problem, which is dealt by many researchers. Dynamic provisioning can be done by two ways: Reactive and Proactive.

1. Describes dynamic VM allocation, relocation and auto scaling algorithms. For scaling application, response time is considered. Stress handling algorithm is suggested in this work. Reactive approach is used to increase or decrease resources allotted to cloud based application. Here resources are increased or decreased based on some predefined thresholds. Application performance related parameters can be considered for finding threshold values. The threshold values can be based various parameters like current workload, response time, CPU or memory utilization of VM etc. Once resource requirement is identified, cloud environment takes some time to provide resources to application. During this time SLA violation may occur. Amazon Web Services provides an auto scale feature (AWS Auto Scale) , allows users to specify conditions under which VM instances should be added or removed.

Opposite to reactive approach, proactive approach uses sophisticated techniques for prediction of future demand and based on that scaling is done. Forecasting engine is suggested as a component in Open Nebula. Here SARIMA (Seasonal Auto Regressive Integrated Moving Average) method is used to predict future workload. Linear regression based work load prediction is done and self healing concept is described. Web application is considered which has many services and each service is implemented as one or more VMs. VM level scaling is achieved. Other paper describes the challenges involved in auto scaling in the cloud. It develops a look ahead resource allocation algorithm based on model-predictive control for workload forecasting that is used for resource auto scaling. For workload forecasting used second order autoregressive moving average (ARMA) method.

Section 2 discusses about proactive auto scaling and prediction methods. Proposed algorithm and methodology is discussed in section 3. Section 4 represents experiments and results. At last, section 5 contains conclusion and future work.

2. PROACTIVE AUTO SCALING AND PREDICTION

Workload prediction is the main challenge in proactive auto scaling. Workload prediction can be done by several methods like Threshold based, Reinforcement learning (RL), Queueing theory, Control theory, Time series Analysis (TSA) etc. Time series analysis based method detects pattern and predicts future values of workload. There are many techniques based on TSA for prediction like Autoregressive (AR), Moving Average (MA), Autoregressive moving average (ARMA), Autoregressive integrated moving average (ARIMA) etc. Time series can be stationary or non stationary. Stationary time series means for each time interval (Xt, Xt+ τ) where τ is the time difference (lag) between two data points, the mean and variance of the process must be constant and independent of t. Non stationary time series can be converted to stationary time series by differencing.

**Figure1** (a) Stationary Time series (b) Non Stationary time series

ARIMA model is generally denoted as ARIMA(p, d, q) where parameters p, d, and q are non-negative integers, p is the order of the Autoregressive model, d is the degree of differencing, and q is the order of the Moving-average model. The parameter d of ARIMA indicates number of times differencing is applied. The values of q and p are determined by
analyzing the autocorrelation and partial autocorrelation plots of the historical data, respectively. In our work we have used ARIMA model for workload prediction.

3. PROPOSED METHODOLOGY

Our approach considers application response time and workload forecasting for making auto scaling decisions. High level diagram of our work is as bellow.

3.1 Proposed Methodology Diagram

Above block diagram can be summarized as below:

1. If application average response time is above the SLA threshold and predicted average workload is also above current average workload go to step 2 Else go to step 3.
2. Perform scale up operation
3. Perform scale down operation
4. At end perform VM relocation and consolidation operations

4. EXPERIMENTAL RESULTS

We have used R for the purpose of time series analysis. It is widely used open source environment for doing complex statistical computing and analysis. For modeling our scenarios, DCSIM- an event driven simulator for Cloud Environment is used. We have performed two experiments. Scenarios, results and graphs are as bellow

Experiment 1:

Table below shows various configuration parameters for our first experiment.

Table 1 Experiment 1 configuration parameters

<table>
<thead>
<tr>
<th>Host Configuration</th>
<th>2 quad core, 2.5 GHz CPU, 16 GB of memory.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU-upper-threshold: 75%</td>
</tr>
<tr>
<td></td>
<td>CPU-lower-threshold: 60%</td>
</tr>
<tr>
<td></td>
<td>CPU-target-threshold: 70%</td>
</tr>
<tr>
<td>VM</td>
<td>Configuration: 1 Virtual core, 1GB RAM.</td>
</tr>
<tr>
<td>No. of applications</td>
<td>5</td>
</tr>
<tr>
<td>SLA warning threshold</td>
<td>0.3 seconds</td>
</tr>
<tr>
<td>SLA safe threshold</td>
<td>0.2 seconds</td>
</tr>
</tbody>
</table>
Graph in next figure shows the behavior of the virtual machines. It shows how number of virtual machines increases or decreases. In case of workload forecasting it considers the average response time and workload forecasting. Virtual machine is added only if in near future workload is going to increase. In case of without workload forecasting virtual machine added by considering only average response time.

**Table 2** Experiment results

<table>
<thead>
<tr>
<th></th>
<th>Without forecast</th>
<th>With forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SLA Achievement</strong></td>
<td>88.417%</td>
<td>89.117%</td>
</tr>
<tr>
<td><strong>Active Hosts</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max: 11.0 mean: 5.095 min: 1.0</td>
<td></td>
<td>max: 11.0 mean: 5.024 min: 1.0</td>
</tr>
<tr>
<td>CPU-util: 52.215% MEM-util: 55.445%</td>
<td></td>
<td>CPU-util: 53.157% MEM-util: 54.826%</td>
</tr>
<tr>
<td><strong>Active VMs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max: 100.0 mean: 37.989 min: 7.0</td>
<td></td>
<td>max: 97.0 mean: 36.852 min: 7.0</td>
</tr>
<tr>
<td><strong>Aggregate Penalty</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total: 70304 max: 3.0 mean: 0.204 min: 0.0</td>
<td></td>
<td>total: 67512 max: 3.0 mean: 0.196 min: 0.0</td>
</tr>
<tr>
<td><strong>Auto scaling operations</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>added: 341 removed: 260 failed: 0</td>
<td></td>
<td>added: 312 removed: 233 failed: 0</td>
</tr>
<tr>
<td><strong>CPU Under provisioning</strong></td>
<td>4.457%</td>
<td>2.494%</td>
</tr>
</tbody>
</table>

**Figure 3** Workload versus number of active VMs
Figure 4 Comparison of application response time

Graph above shows the behavior of the response time. In case of forecasting for maximum response time stays below the SLA warning threshold.

Experiment 2:

Table below shows various configuration parameters for our first experiment.

<table>
<thead>
<tr>
<th>Scenario 2 configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host Configuration</strong></td>
</tr>
<tr>
<td>Configuration: 2 quad core, 2.5 GHz CPU, 16 GB of memory. CPU-upper-threshold: 75%, CPU-lower-threshold: 60%, CPU-target-threshold: 70%</td>
</tr>
<tr>
<td><strong>VM</strong></td>
</tr>
<tr>
<td>Configuration: 1 Virtual core, 1GB RAM.</td>
</tr>
<tr>
<td><strong>No. of applications</strong></td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td><strong>SLA warning threshold</strong></td>
</tr>
<tr>
<td>0.3 seconds</td>
</tr>
<tr>
<td><strong>Sla safe threshold</strong></td>
</tr>
<tr>
<td>0.2 seconds</td>
</tr>
<tr>
<td><strong>Prediction Model Used</strong></td>
</tr>
<tr>
<td>ARIMA</td>
</tr>
<tr>
<td><strong>History Considered</strong></td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td><strong>Traces Used</strong></td>
</tr>
<tr>
<td>claranet, epa, sdsc</td>
</tr>
<tr>
<td><strong>Simulation Time</strong></td>
</tr>
<tr>
<td>10 Hours.</td>
</tr>
</tbody>
</table>
Table below shows results obtained from second experiment.

<table>
<thead>
<tr>
<th></th>
<th>Without forecast</th>
<th>With forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SLA Achievement</strong></td>
<td>67.67%</td>
<td>70.265%</td>
</tr>
<tr>
<td><strong>Active VMs</strong></td>
<td>max: 358, min: 246.682, mean: 21.0</td>
<td>max: 356, mean: 244.151, min: 21.0</td>
</tr>
<tr>
<td><strong>Aggregate Penalty</strong></td>
<td>total: 99976, max: 7.0, mean: 2.824, min: 0.0</td>
<td>total: 89266, max: 7.0, mean: 2.522, min: 0.0</td>
</tr>
<tr>
<td><strong>Auto scaling operations</strong></td>
<td>added: 305, removed: 24, failed: 0</td>
<td>added: 300, removed: 20, failed: 0</td>
</tr>
<tr>
<td><strong>CPU under provisioning</strong></td>
<td>15.528%</td>
<td>13.686%</td>
</tr>
</tbody>
</table>

### 5. CONCLUSION AND FUTURE WORK

Experiment performed by considering various scenarios. In each scenarios experiment first performed without considering the workload forecasting and afterwards experiment performed by considering the workload forecasting using ARIMA model of time series analysis. By comparing the results from various scenarios we can conclude that considering workload forecasting maximizes the SLA achievement and minimizes the number of scaling operations, Virtual Machines and hosts. In future we can compare results with large number of hosts, applications and VMs. We can use some other prediction techniques for comparison.

### References

[1] Integrating Cloud Application Auto scaling with Dynamic VM Allocation Michael Tighe and Michael Bauer Department of Computer Science


AUTHOR
Chirag A. Patel received the B.E. and M.E. degrees in Computer Engineering from Gujarat in 2002 and 2011, respectively. He is now with Vishwakarma Govt. Engg. College, Chandkheda, Ahmedabad. Currently he is pursuing PhD from Gujarat Technological University, Ahmedabad.