ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A review on cloud elasticity for system performance improvement and cost reduction

Pancham Baruah
ME-CN
Department of Computer Engineering
DY Patil School of Engineering & Technology
Charoli, Pune – India

Abstract: Cloud Computing as a new enterprise model has become popular to provide on-demand services to user as needed. Cloud Computing is essentially powerful computing paradigm to deliver services over the network. Computing at the scale of the Cloud system allows users to access the enormous resources on-demand. However, a user demand on resources can be various in different time and maintaining sufficient resources to meet peak resource requirements all the time can be costly. Therefore, dynamic scalability which can also be called as elasticity is a critical key point to the success of Cloud computing environment. Dynamic resizing is a feature which allows server to resize the virtual machine to fill the new requirement of resources in a cloud environment. Though there are enormous applications hosted on cloud now days, but the next big thing which will be focused on will be the elasticity. In this paper, an effort has been put to explain the cloud elasticity concept and how it will benefit the Cloud implementers in reducing operation cost and also to improve the system's performance as a whole.

Keywords: Elasticity, Scalability, Performance, Cloud, SaaS, Cost effective

I. INTRODUCTION

Cloud computing has become a significant part for many business and scientific applications. Many large organizations such as Amazon, Google and IBM provide large-scale public cloud services. However, on-demand workload scheduling has become critical as there are dynamic workloads in the applications. Beyond technological advances, cloud computing also holds promises to change the economic landscape of computing. Pricing of the cloud resources is both a fundamental component of the cloud economy and a crucial system parameter for the cloud operator, because it directly impacts customer usage pattern and the utilization level of infrastructure. Static pricing remains the dominant form of pricing today. However, it is intuitive to adopt a dynamic pricing policy in order to strategically influence demand in order to better utilize unused cloud capacity, and to generate more revenue. Dynamic pricing emerges as an attractive strategy to better cope with unpredictable customer demand. This is where elasticity of Cloud infrastructure comes into play. Amazon has already introduced an elastic computing cloud platform (EC2) and started "spot pricing" feature for its instances, with the spot price dynamically updated to match supply and demand. Most application owners are appealed by the pay-as-use model. It eliminates the costs of buying, installing and maintaining a dedicated infrastructure for running an application. Moreover, most IaaS providers allow the application owners to scale up and down the resources used based on the computational demands of their applications, thus letting them pay only for the amount of resources they use. This model is appealing for deploying applications that provide services for third parties, e.g. traditional e-commerce sites, financial services applications, online healthcare applications, gaming applications, media servers and bioinformatics applications. If the workload of a service increases (e.g. more end users start submitting requests at the same time), the application owner can ideally scale up the resources used to maintain the Quality of Service (QoS) of their service. When the workload eases down, they can then scale down the resources used. Within this context, elasticity (on-demand scaling), also known as redeploying or dynamic provisioning, of applications has become one of the most important features. Elasticity enables a Cloud Service Provider to reduce the cost of resources and also to increase the performance of the system.

II. Literature Survey

a) Existing System

An extensive literature survey has been performed related to elasticity and dynamic scaling of applications. Early work were more focused on underlying computing resources such as CPU and memory and considered single-tier. Further investigations classify an application into multiple tiers. It consisted of breaking down the end-to-end response time by each tier and conduct the worst-case capacity estimation to ensure applications meeting the peak workload. Overall, the single-tier model can be viewed as a special case of a multi-tier model and the latter model can guide the scaling in a more accurate way. However, although scaling of traditional applications, which are often hosted on physical servers, shares many commonalities with that of cloud applications, these two types of scaling technique have different emphases. Conventional techniques mainly concentrate on how to schedule compute nodes to meet the Quality of Service requirements of applications by predicting their long term demand changes. In contrast, clouds focus on providing metered resources on-demand and on quickly scaling applications up and down whenever application demand changes. Further investigations, therefore, are needed to address the challenges brought about by this requirement for high elasticity and how it will benefit in reducing the cost of operations and improving system performance.

b) Proposed System

The approach followed consists of two steps. First, an expressive models of elasticity actions is presented and second, leverage them for devising concrete policies that can take elasticity decisions. Markov Decision Processes (MDPs) has been adopted as the mathematical modeling framework, because MDPs can capture both the non-deterministic and probabilistic aspects of the problem. several possible elasticity actions can be applied for which it can be categorised as Non-deterministic, whereas the probabilistic behavior helps in to take account of the effects of the unpredictable environment's evolution. Probabilistic models are used in the decision making process, to describe, drive and analyse cloud resource elasticity. By utilizing probabilistic models, the uncertain behaviour of systems elasticity can be captured. In order to additionally capture non-determinism MDP models are used, which form the basis of our approach. Similarly to our implementation, there are numerous other approaches where MDPs are used to handle both runtime and offline decision making. The resizing of a cluster has been considered here as the main form of elasticity, i.e., dynamically modifying the number of VMs with a view to optimizing a utility function. While the main objective is to render elasticity decision policies more dependable, the principled approach is capable of yielding higher utility.A higher utility of system resoures will ensure that the cost is optimised for the cloud resources. Also the dynamic addition of Virtual machines ensures that the system is scalable and the performance is not degraded.

III. Motivation

Two main scenarios have been considered here towards cost and system performance. Without loss of generality, a simple example is used based on an e-commerce website to capture the typical behaviour of such applications. Also for simplicity, focus is made only on applications that are deployed on the resources of single IaaS cloud provider. As discussed in the introduction, the workload for such applications depends on the number of end users submitting requests at the same time. Below are the two main points which can be considered as part of motivation factor included with elasticity of cloud.

Challenge 1: Reducing Cost of usage of Cloud resources

In a highly scalable cloud environment where computing resources are consumed as a utility such as water and electricity, application owners would expect to spend the least cost for the desired application performance. To this aim, the elastic scaling must take cost-aware criteria into consideration and use them to guide application scaling. These criteria should be aware of both the cost of adding a server (e.g., an Apache or a MySQL) and the performance effect brought by this scaling up (e.g., reducing response time). When the application is initially deployed, five servers of this application are hosted across different VMs to support a small number of customers. When the demand increases, the application should be scaled up. An interesting point here is that this scaling process is greatly influenced by the behavior (i.e., the type of workload) of the application itself. Three typical types of workloads are examined, where each workload places varying demands on different tiers of the application. In the primarily ‘‘browsing’’ workload, end users mainly browse webpages and preview products. This workload mainly stresses the service tier including the Apache and Tomcat servers, so their resources are saturated and the number of these servers needs to be increased. By contrast, the primarily ‘‘ordering’’ workload mainly stresses the storage tier including the MySQL database and so the number of these database servers needs increasing. Finally, the typical ‘‘shopping’’ workload simultaneously stresses the service and storage tiers and so the number of servers in both two tiers is increased.

Challenge 2: Improving the system Performance.

Due to the dynamic cloud environment, two types of uncertainties exist in the application workload: (1) the type of workload, such as three types of workload i.e low, medium and high; (2) the volume of workload, which is denoted in terms of the arrival rate of incoming requests, namely the number of incoming requests per time unit. In this context, the elastic scaling must be adaptive to the changing workload, and such adaptive scaling has triple meanings. First of all, bottleneck tiers of applications should be automatically identified both for scaling up and down. Secondly, scaling should be performed as an iterative process because fixing a bottleneck tier may create another bottleneck at a different tier of the application. For instance, the bottleneck is shifted to the storage tier if multiple Apache and Tomcat servers are added to the service tier. Finally, agile scaling is needed to rapidly restore acceptable application performance. In the remainder of this paper an algorithm is explained that address both challenges effectively. The approach is implemented and evaluated by using the IC-Cloud platform as an example. The advantage of using the IC-Cloud platform is that it supports a fine-grained pricing strategy (e.g., VM instances are charged by minute rather than by the hour as in Amazon WS) which simplifies the evaluation. However, the approach and algorithms are generic and can be applied on most IaaS environments.

IV. ARCHITECTURE

There can be different types of cloud application such as end user application or infrastruture application. Examples of infrastructure applications are DNS servers, email servers or databases. These type of applications have often simple structures, such as having one or two tiers. By contrast, the structure of an end-user application is more complex. In this work, an end-user application has been used as an example because it can also incorporate the simpler infrastructure application scenario. In a multi-tier application, servers are categorized into different tiers according to their functionalities.

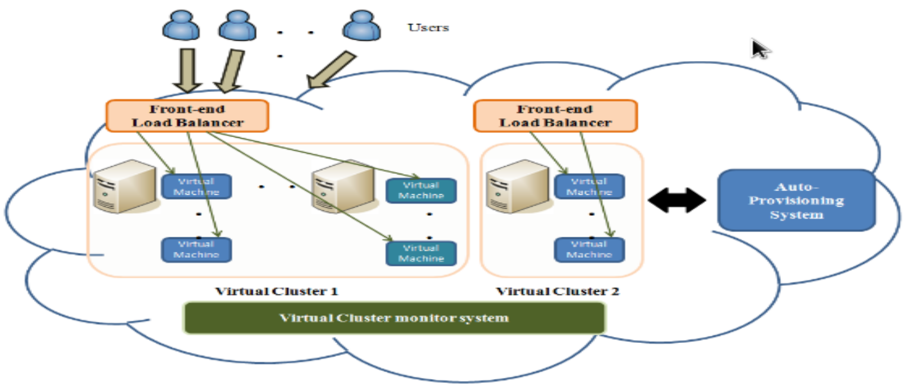
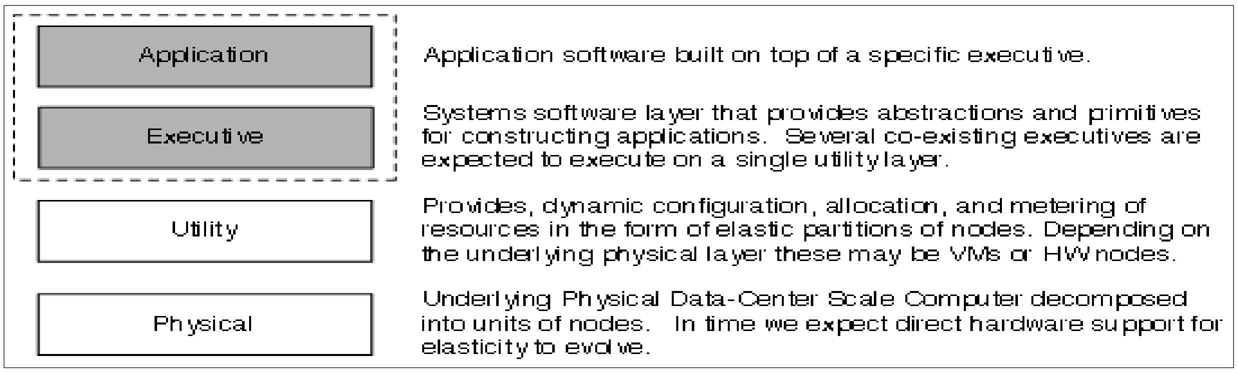


Fig 1: Cloud Architecture and elastic components

Servers at the two load balancing (LB) tiers, distribute requests to servers at the service or storage tiers; servers at the service tier, such as Apache and Tomcat, are responsible for handling HTTP requests and implementing business logic; and servers at the storage tier, such as the MySQL database, are used for managing application data. In addition, servers at the service or storage tier can be further divided into different sub-tiers. Typically, each application has a set of demands and constraints specified by the application owner in the form of a SLA. A performance demand is defined by the maximum end-to-end response time for a request. This time can either be an average response time or a high percentile of response time distribution (e.g., 90% of the response times should be less than 2 s). A cost constraint is the budget of the total application deployment. In addition, each tier has a resource constraint that restricts the maximum number of servers in this tier. For instance, the maximum number of Tomcat servers is 10.



Application software built on top of a specific executive.

Systems software layer that provides abstractions and primitives for constructing applications. Several co-existing executives are expected to execute on a single utility layer.

Provides, dynamic configuration, allocation, and metering of resources in the form of elastic partitions of nodes. Depending on the underlying physical layer these may be VMs or HW nodes.

Underlying Physical Data-Center Scale Computer decomposed into units of nodes. In time we expect direct hardware support for elasticity to evolve.

Fig. 2 Different Layer s of elasticity logic implementation

V. ALGORITHM

For web application, there is a Virtual cluster monitor system. It can detect whether the number of active HTTP sessions are over the threshold in a virtual cluster. For distributed computing task, Virtual cluster monitor system is able to detect whether the number of virtual machine are over the threshold of use of physical resources in a virtual cluster. The auto-scaling algorithm is implemented in Auto-provisioning system, and Virtual cluster monitor system is used to control and trigger the scale-up and scale-down in Auto-provisioning system on the number of virtual machine instances based on the statistics of the scaling indicator.

Elasticity Algorithm

Input: n: number of Clusters

VC: a Virtual Cluster consists of VMs that run the same computational system

VMns: active session number in a virtual machine

*Si*Max: maximum sessions for a virtual machine of *i*-th Cluster

Supper_bound: session upper-threshold

Slow_bound: session low-threshold

Ebelow: a virtual machine set records virtual machines that exceed the session upper-threshold

Output: Front Load-Balancer

```
1. For i = 1 to n
2. For each VM element_of VCi
3. If (VMns/SiMax >= Supper_bound) then
4. e = e + 1
5. If (VMns/SiMax <= Slow_bound) Then
6. b = b + 1
7. Record VM to Ebelow
```

```
8. If (e == | VCi|) then
9. Provision and start a new VM that runs the same system as VCi
10. Add new VM to FLB //Front Load-Balancer Set
11. If (b >= 2) then
12. For each VM in Ebelow
13. If (VMns == 0) then
14. Remove VM from FLB //Front Load-Balancer Set
15. Destroy VM
16. Empty Ebelow
```

VI. Methodology

Definitions:

1) Performance:

The performance is characterized by the time needed to complete a given number of requests with a given level of parallelization. The chosen levels of parallelization and number of requests used during the measurements are explained in the step by step methodology. For this article, all requests are performed in batches called request sets . This helps in decreasing variability and improving accuracy in measurement of time.

2) Elasticity:

Elasticity is defined as the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible. It is a defining characteristic, that differentiates it from previously proposed computing paradigms, such as grid computing. This dynamic variation in the use of computer resources to meet a varying workload is called "elastic computing".

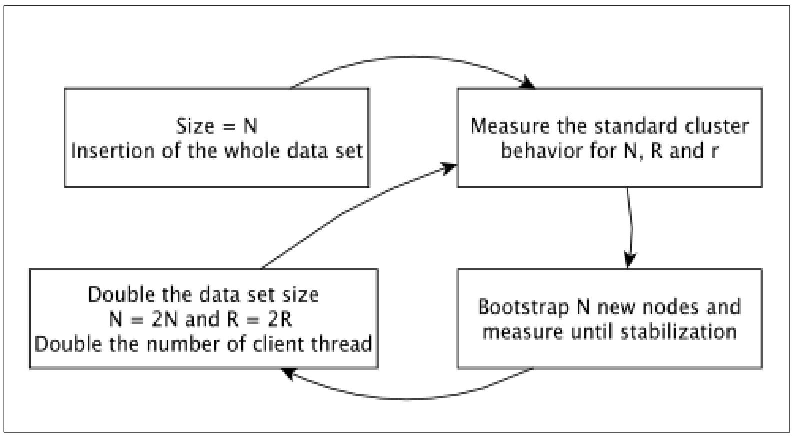

Size = N
Insertion of the whole data set
Measure the standard cluster behavior for N, R and r
Bootstrap N new nodes and measure until stabilization
Double the data set size
N = 2N and R = 2R
Double the number of client thread

Fig:3 Elasticity Methodology

Figure 3 depicts the step by step methodology used during the tests. The methodology is based on the following parameters :N the number of nodes,R the size of a request set and r the percentage of read requests. In practice, the methodology is defined by the following steps:

1. A cluster of N= 6 nodes is started up with and all the Wikipedia articles are inserted.
2. The elasticity test is started by performing request sets that each contain R = 10000 requests with r = 80% read requests and as many threads as there are nodes in the cluster when the elasticity test begins. The time for performing each

request set is measured. (Therefore the initial request sets execute on 6 threads each serving about 1667 (10000/6) requests.) This measurement is repeated until the cluster is stable. This gives the variability for a stable cluster.

3. New nodes are bootstraped to double the number of nodes in the cluster and continued until the cluster is stable again. During this operation, the time measurements continue. It is assumed that the cluster is stable when the last 5 request sets have delta times less than the one measured for the stable cluster.
4. Then the data set size is doubled by inserting the Wikipedia articles as many times as needed but with unique IDs for each insert.
5. To continue the test for the next transition, step (2) to (4) can be continued with a doubled number of requests and a doubled number of threads.

Justification of the methodology

One approach to characterize the variability is to use the standard deviation of request set times and a statistical test to compare the standard deviations. However, experience shows that the standard deviation is too sensitive to normal cluster operations like compaction and disk pre-allocations. This is why the delta time characterization is used instead. Because it is based only on the average values, it tends to smooth these transient variations. The median of all the observed delta times is used instead of the average to be less sensitive to the magnitude of the fluctuations. Remark that the standard deviation are still used as part of the characterization of the elasticity. This characteri- zation captures all the important information about the elasticity (time needed to stabilize, loss of performance, and variability) with the two surface areas (A and B) and normalizes it into a dimensionless number that can be used for comparisons. Finally, the number of observations needed to have an idea of the normal behavior of a database cluster cannot be fixed in advance. Experience shows that, from one system to another, high variability in performance can arise at different moments. This variability is mainly due to the writes of big files on the disk, like compactions, disk flushes, and disk pre-allocations, all of which can happen at very different moments due to the randomness of the requests and the technical choices made by each database. The variability has a measurable result that will be discussed in the result section. In practice, the observations were stopped when the performance and standard deviation got back to the level observed before the compactions or disk pre-allocations happened

VII. Advantages of The Discussed Work

The elasticity techniques are capable of handling sudden load requirements, increasing system performance, maintaining higher resource utilization and reducing energy cost.

VIII. Conclusion

In this paper , cloud elasticity scenarios have been discussed.The cloud architecture discussed in this paper consisted of a Front-end load balancer, a Virtualcluster 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 use of physical resources of each virtual machine in a virtual cluster. The Auto-provisioning system is utilized to dynamically provision the virtual machines based on the number of the active sessions or the use of the resources in a virtual cluster. All these components are the worker components of an elastic implementation. In addition, the idle virtual machines are destroyed, and then the resources are able to be released. In the other hand, the energy cost can be reduced by removing the idle virtual machines. Thus, the elasticity techniques are capable of handling sudden load requirements, increasing system performance, maintaining higher resource utilization and reducing energy cost.

Acknowledgement

I would like to thank Prof. Mrs Arti Mohanpurkar (HOD, Computer Science Department), DY Patil SOET, Pune for her valuable guidance to publish the work on this paper.

References

1. "Elasticity in Cloud Computing: What It Is, and What It Is Not" by Nikolas Roman Herbst, Samuel Kounev, Ralf Reussner Institute for Program Structures and Data Organisation Karlsruhe Institute of Technology Karlsruhe, Germany.
2. Amdahl, G.: Validity of the single processor approach to achieving larg e-scale computing capabilities. In: AFIPS Conference. p. 483485 (1967)
3. T. Dory, "Study and Comparison of Elastic Cloud Databases: Myth or Reality?" pldc.info.ucl.ac.be, Programming Languages and Distributed Computing (PLDC) Research Group, Universite catholique de Louvain, Tech. Rep., Aug. 2011
4. A Cost-aware Elasticity Provisioning System for the Cloud by Upendra Sharma, Prashant Shenoy (University of Massachusetts),Sambit Sahu and Anees Shaikh(IBM Watson Research Center)

AUTHOR(S) PROFILE

Pancham Baruah, received the B.E degree in Computer Science & Engineering from PES College of Engineering, Aurangabad in 2008 and is currently pursuing his M.E (CN) in D.Y Patil SOET, Pune. His area of interest lies in Performance and scalability analysis of applications, Cloud technology and Graphic Designing.