International Journal of Computer Applications (0975 – 8887)
Volume 120 – No.14, June 2015

Impact of Elasticity on Cloud Systems

Pancham Baruah
Department of Computer Engineering,
Dr. D. Y. Patil School of Engineering and Technology, Charoli Bdk, Pune, India

Arti Mohanpurkar
Department of Computer Engineering,
Dr. D. Y. Patil School of Engineering and Technology, Charoli Bdk, Pune, India

ABSTRACT
Today's computing world and application market is dominated by cloud technology. Cloud computing provides a powerful computing paradigm and deliver services over the network and has emerged as a new enterprise model. With Cloud computing, the service providers can provide on-demand services to users as needed. In cloud systems, enormous resources are involved and computations are done at a very vast scale which enables users to access huge amount of resources on demand. But there is uncertainty of the demand of cloud resources by the end users as it can vary depending on the time. Also it becomes costly affair in maintaining sufficient resources to meet peak resource requirements all the time. This is where dynamic scalability or elasticity comes into picture. Elasticity of cloud is very necessary as it allows the servers to resize the virtual machine deployed in the system and thereby fulfilling the requirement of new resources. Elasticity can be considered as the next great achievement which is getting much focus. In this paper, an effort has been put to analyze the impact of elasticity on cloud systems and how it will benefit the Cloud implementers to improve the systems performance and reduce the operation cost.

Keywords
Cloud computing; Elasticity; Performance; cost optimization; throughput

1. INTRODUCTION
Cloud computing has become the most sought after technology in today's world and has become a significant part for many business and scientific applications. The boom of handheld devices has accelerated the pace of growth of cloud as the application usage has become ubiquitous. Many big market players such as IBM, Microsoft and Google provide large-scale public cloud services. However, on-demand workload scheduling has become critical as the applications in cloud can be bombarded with dynamic workloads [1]. Beyond technological advances, cloud computing also holds promises to change the economic landscape of computing. The costing of cloud resources is both a fundamental component of the cloud economy and an essential system parameter for the cloud operator, because customer usage pattern and the utilization level of infrastructure is directly impacted by it. Considering the current market scenario, static pricing remains the dominant form of pricing today. In static pricing scheme, the Cloud user predefines its requirements to the Cloud service provider. The requirements are in terms of computing resources, storage areas, virtual machines etc. In this way, all the required resources are reserved by the cloud service provider well in advance. This technique can be termed as fixed pricing technique where the price is calculated based on the resources that are being reserved [3]. However, fixed pricing technique suffers from underutilization of resources from Cloud service provider point of view and monetary issues from cloud users' point of view. Therefore, in order to strategically influence demand in order to better utilize unused cloud capacity, and to generate more revenue, it is intuitive to adopt a dynamic pricing policy. Dynamic pricing strategy will help to better tackle with unpredictable customer demand. This is where elasticity and dynamic provisioning of Cloud infrastructure comes into picture. 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 beneficial for deploying applications that provide services for third parties, e.g. traditional e-commerce sites, financial services applications and bioinformatics applications. The application owner can ideally scale up the resources if the workload of a service increases (e.g. more end users start submitting requests at the same time) and thus used to maintain the Quality of Service (QoS) of their service [2]. They can also scale down the resources used when the workload eases down. The same thing can be implemented in the form of automatic provisioning of the resources in the cloud which can be called as elasticity. Within this context, elasticity (on-demand scaling), also known as redeploying or dynamic provisioning, of applications has become one of the most significant features. Elasticity empowers a Cloud Service Provider to reduce the cost of resources and also to increase the performance of the system

2. LITERATURE SURVEY
2.1 Existing Systems
An extensive literature survey has been done related to elasticity and dynamic scaling of applications. Paper [1] discusses the authors have described a novel architecture for the dynamic scaling of web applications based on thresholds in a virtualized Cloud Computing environment. They have also illustrated a scaling approach with a front-end load-balancer for routing and balancing user requests to web applications deployed on web servers installed in virtual machine instances. The scaling of internet applications have also been discussed. Here focus is put more on web applications and their scalability. In paper [2], the author discusses about the grouping of homogeneous data and processes the same in chunks. The application workloads which are of similar types have been categorized under one group and are termed as homogeneous workload. Due to homogeneity of tasks, the processing time reduces which helps in doing a task in lesser amount of time and thus reducing cost. In paper [3], stress has been put on assigning a checkpoint during the computation process. Checkpoint keeps an eye on the tasks and computations which are being repeated and avoids the same using a pre saved values in

checkpoint buffer. Paper [4] familiarizes us with the various mechanisms of xen virtualization which can be extended in a cloud environment. Xen is the underlying technology on which virtualization works. Xen is an x86 virtual machine monitor that allows multiple commodity operating systems to share conventional hardware in a safe and resource managed fashion, but without compromising the performance or functionality of the system. The same is achieved by providing an idealized virtual machine abstraction to which operating systems such as Linux, BSD and Windows XP, can be ported with minimal effort. In paper [5], the authors have developed a customized framework for cluster management of virtual. After going through all the papers it can be referred that early works were more focused on system level tuning and underlying computing resources such as CPU and memory and mostly considered single-tier setups and architecture. Few papers were there in which an application was classified as multi tier and multiple level deployment was considered. It consisted of segregating down the end-to end response time tier wise and conducts the worst-case capacity estimation to ensure applications meeting the peak workload. This was more of a conventional way of measuring performance. 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. Although scaling of traditional applications, which are often hosted on physical servers, shares many similarities with that of cloud applications, 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. However in cloud environment, focus is put more 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.

2.2. Proposed System

A proper methodology is required for the implementation of an elasticity handling framework in the cloud environment. To make this task easier, Markov Decision Processes (MDPs) has been adopted as the mathematical modeling framework. The proposed process consists of two steps. First, an expressive model of elasticity actions is presented and second, bargaining them for devising concrete policies which can further take dynamic provisioning decisions. Markov Decision Processes (MDPs) has chosen been because MDPs can capture both the probabilistic and non deterministic aspects of the problem. The non-deterministic approach handles the various possible elasticity and the probabilistic nature and helps in to take account of the effects of the unpredictable environments evolution. Also, elasticity probabilistic models are used in the decision making process to describe, drive and analyze cloud resources. It is also helpful to capture the uncertain behavior of systems elasticity. MDP model is also used to additionally capture non-determinism and this form the basis of the proposed approach. There are also numerous other approaches where MDPs are used to handle both offline and runtime decision making. The dynamic 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 principle approach is capable of yielding higher utility. The performance of system resources has also been handled. The aim is to distribute the system load across all the free VMs and acquire a higher utilization rate. This will ensure that the cost is optimized for the cloud resources. Also the dynamic addition of Virtual machines ensures that the system is scalable and the performance is not degraded.

3. 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 bulk e-mail sending application to capture the typical behavior of the overall system. Also for simplicity, focus is made only on applications that are deployed on the resources of single IaaS cloud provider.

3.1 Reducing the cost

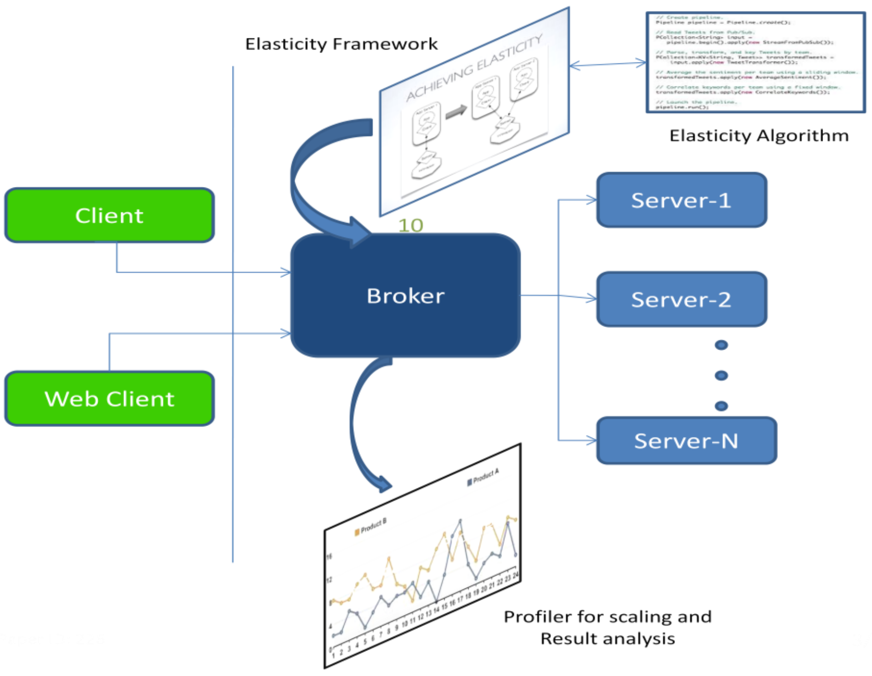
For the application that we have considered, the workload significantly depends on the number of emails that are sent to the mail delivery Server (also called as Mail Transfer Agent Server). Below are the two main points which are considered as part of motivation factor included with elasticity of cloud. When the application is initially deployed, few servers of this application are hosted across different VMs to support a small number of customers. As the demand increases, the application should be able to scale up itself. A vital factor 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. These workloads can be light, medium and heavy workload. Each workload places varying demands on different tiers of the application. This is helpful in simulating a real usage scenario of the actual world. In the primarily light workload, the email delivery application simply creates a mailing and sends it via email assembly server and mail transfer agent. The email assembly server is the personalization engine of the application where different components and contents of an email are created according to the target user. For light load, the templates that are used for email are of lesser size and fewer contents like less textual matter and less images. Also, we set a throttling factor in the applications configuration so that lesser number of mailings are sent per unit time. The light workload mainly stresses the service tier including the Apache and Tomcat servers. Then comes the medium workload where we increase the contents of the templates with more text and images. Also, personalization urls are included which gets personalized dynamically. For medium workload, the mailings are sent to a larger set of audience and throttling factor is set to a medium level, so that a higher workload is created on the server as compared to light workload. Finally, the typical higher workload is considered which simultaneously stresses the service and storage tiers and so the number of servers in both two tiers is increased. For higher workload, we choose a message template having large number of textual and image contents. Also, when the mailings are received in the end users system, which has been simulated by a local loopback control in this case, some tracking data is generated. The tracking data signifies the actions of the actual end users and shows their behavior towards the emails which they have received.

3.2 Improving the system performance

The cloud environment is very uncertain as far as the workload in the application is concerned. This dynamic nature of cloud can be formulated in the form of two types of uncertainties that exist in the application workload. They are: (1) the workload volume, which can be represented by the arrival rate of incoming requests per time unit (2) the workload type, such as three types of workload i.e low, medium and high workloads. Taking the above two points into consideration, the elastic scaling must be adaptive to the changing workload, and such adaptive scaling can have three interpretations. First of all, to scale the application up and down, bottleneck tiers of applications should be automatically identified. Secondly, there can be after affects of fixing a bottleneck because fixing at one tier may create another bottleneck at a different tier of the application. Therefore scaling should be performed as an iterative process. For example, if multiple Apache and Tomcat servers are added to the service tier, the bottleneck is shifted to the storage tier. Finally, to rapidly restore acceptable application performance agile scaling is needed. Agility signifies the easiness with which an application can be scaled quickly and easily without or minimal disruption. In the coming sections of this paper an algorithm is explained that address both the challenges effectively. The approach is implemented and evaluated by using the Amazon Cloud platform as an example. The advantage of using the Amazon Cloud platform is that it supports a fine-grained pricing strategy which simplifies the evaluation of the parameters. However, the approach and algorithms are generic and can be applied on most IaaS environments.

4. ARCHITECTURE

The architecture of the overall system can be depicted in the figure 1. The email delivery application is deployed over the cloud. There can be multiple instances of delivery servers consisting of message assembly servers and mail transfer agents. When a user of the application triggers the sending of email actions, load is created. This load has to be distributed equally across all the servers of the virtual machines. For the current architecture, an elasticity framework is developed which will present along with the application layer. The framework continuously monitors the load on the existing servers and also monitors the performance of the systems. This is done dynamically at a predefined time interval and is constantly monitored. The monitoring and controlling decisions are taken by the Monitor and Control component which are present as a separate entity. The information collected is stored in a stable data storage. The framework reads the latest state of the application along with the load on the servers and the performance parameters. Once it detects the increase in load on a particular server, it triggers an action point to scale up the number of servers. The configuration of the new server to connect to the existing system is handled by the framework. Servers at the 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. The storage tier servers, such as the MySQL database, are used for managing application data. Usually, each application has a set of demands and constraints specified by the application owner in the form of a Service Level Agreement. Typically, the performance demand is defined by the maximum end-to-end response time for a request. The cost constraint impacts 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.



```
// Create pipeline.
Pipeline pipeline = Pipeline.create();

// Read Tweets from Pub/Sub.
PCollection<String> input =
    pipeline.begin().apply(new StreamFromPubSub());

// Parse, transform, and key Tweets by team.
PCollection<KV<String, Tweet>> transformedTweets =
     input.apply(new TweetTransformer());

// Average the sentiment per team using a sliding window.
transformedTweets.apply(new AverageSentiment());

// Correlate keywords per team using a fixed window.
transformedTweets.apply(new CorrelateKeywords());

// Launch the pipeline.
pipeline.run();
```

Fig 1: Architecture of the framework

5. ALGORITHM

For a standalone application like the email delivering system, there is a Virtual cluster monitor. It can detect whether the load on the server 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 elasticity algorithm is implemented in auto provisioning system, and Virtual cluster monitor system as component part of the framework is used to control and trigger the scale-up and scale-down in elastic provisioning system on the number of virtual machine instances based on the statistics of the scaling indicator. The algorithm accepts the number of clusters of Virtual Machine. There are different parameters which are calculated dynamically and optimized output is calculated to give the front load balancer the best decision. The different parameters consists of the VC which is a virtual cluster consisting of virtual machines. The machines which are active in a particular cluster is handled by the parameters VMns. Since the number of sessions for a virtual machine is limited, a separate variable is defined to handle it. Also there are threshold value storage variables which handle the minimum and maximum values of threshold.

Algorithm : Elasticity Algorithm

```
Input: n: number of Clusters
1 VC: a Virtual Cluster consists of VMs that run the same
computational system
 VMns: number of active session in a virtual machine
 SiMax: maximum sessions for a virtual machine of a
Cluster
 Supper bound: upper-threshold of session
 Slower bound: lower-threshold of session
 Ebelow: a set records of virtual machines that exceed the
session upper-threshold
  Output: Front Load Balancer
2 for i = 1 to n do
3       for each VM element of 2 VCi do
4               if (VMns=SiMax >= Supper bound) then
5                       e = e + 1
6                       if (VMns=SiMax >= Lower
                        bound) then
7                       b = b + 1
8                       end
9                       Record VM to E below
10              end
11              if (e == VCi) then
12              Provision and start a new VM that runs
                the same system as VCi
                Add new VM to FLB (Front Load-
                Balancer Set)
13              end
14      end
15 end
16 if (b >= 2) then
17      for each VM in Ebelow do
18      if VMns == 0 then
19      RemoveVMfromFLB (Front Load Balancer Set)
        DestroyVM EmptyEbelow
20      end
21 end
22 end
```

6. METHODOLOGY

6.1 Definitions

Two terms are discussed here which forms the basis of central focus

6.1.1 Performance

The performance defines the state and condition of the systems and tells about how a system is performing in various states of workloads. It 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. In our case, all requests are performed in batches called request sets which are being performed as low, medium and high workloads. This helps in decreasing variability and improving accuracy in measurement of time.

6.1.1 Elasticity

It is the ability to adapt to workload changes by provisioning and de-provisioning 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 also called as auto scaling or auto provisioning. It is a defining factor for the overall implementation of the framework.

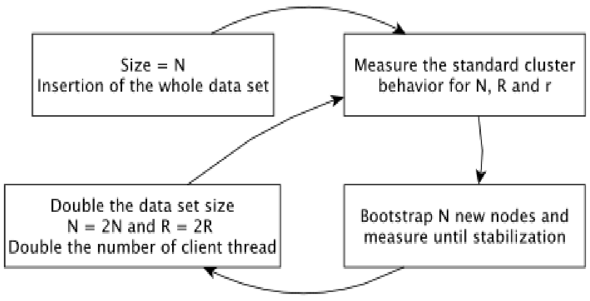

Size = N
Insertion of the whole data set
Measure the standard cluster behavior for N, R and r
Double the data set size
N = 2N and R = 2R
Double the number of client thread
Bootstrap N new nodes and measure until stabilization

Fig 2. Elasticity Methodology

Figure 2 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 the emails records are injected to the email delivery server.

2) The elasticity test is started by injecting different request sets. The request sets varied according to the workload. For low workload, 10000 records are being injected. The payload for each record has been kept as 1KB. For medium workload, 50000 records are injected simultaneously in a thread of 25. The payload has been kept as 5KB. For high workload, 100000 records have been injected in a thread of 50 with payload size of 10KB. The time for performing each request set is measured. This measurement is repeated until the cluster is stable. This gives the variability for a stable cluster.

3) New nodes are bootstrapped 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 higher payload records as many times as needed but with unique IDs for each insert. This is done by injecting unique records having
unique domains.

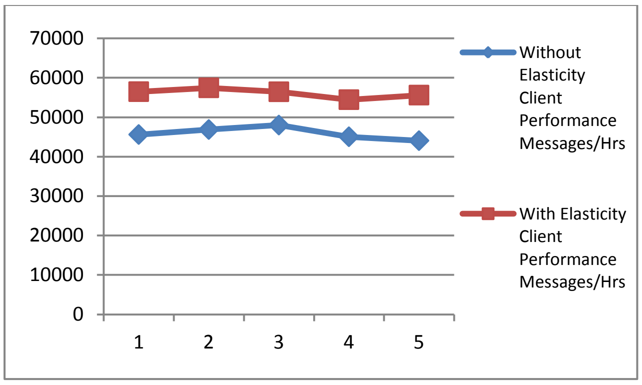
5) To continue the test for the next transition, step (2) to (4) can be continued with a doubled number of requests and a doubled number of threads.

6.2 Justification of the methodology

The methodology that is being used above can be justified by the following points. The aim to analyze the impact of variability of the actions in the framework. To characterize the variability, one approach can be to use the standard deviation of request set times and a statistical test to compare the standard deviations. However, standard deviation is too sensitive to normal cluster operations like compaction and disk pre-allocations. Therefore, 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. All the important information about the elasticity (time needed to stabilize, loss of performance, and variability) are captured by this characterization. It also normalizes it into a dimensionless number that can be used for comparisons. This comparison numbers can be well plotted in a graphical manner and presented in a reporting format to make the analysis of the performance easier for the admin of the framework.

7. TEST RESULTS AND COMPARISON

As previously mentioned, the application deployed over cloud is an email delivery application. There are transactional mailings which are a type of mailing that are always in active state and acts like a listening socket. As soon as records are injected to the same they are sent via the mail transfer agent via the SMTP port over a local loop. In the below result set, only a part of test numbers are mentioned that are collected for a test done with 50 concurrent threads. The payload used and the pattern of injection remains the same for both the tests i.e with elasticity and without elasticity.



Without Elasticity	With Elasticity	% Increase in Client Performance
Client Performance	Client Performance	
Messages/Hrs	Messages/Hrs	
45613	56478	23.82
46878	57451	22.55
48004	56450	17.59
45046	54450	20.88
44056	55605	26.21

Since there is an increase in the performance of the overall system, the cost of operation of the system gets reduced.

Without Elasticity

Payload size of records (in Kb)	API response (Milli Seconds)			Message delivery latency (Seconds)			Client Performance	Server Performance	Threads
	Min	Max	Avg.	Min	Max	Avg.	Messages/Hrs	Messages/Hrs	
1	30	18460	208	1	192	6.1	45613	45610	50
3	40	22350	228	1	139	6.10	46878	46870	50
5	30	28454	215	1	106	6.00	48004	47950	50
7	30	29580	210	1	139	7.10	45046	45003	50
10	20	27541	221	1	533	10.94	44056	44001	50

With Elasticity

Payload size of records (in Kb)	API response (Milli Seconds)			Message delivery latency (Seconds)			Client Performance	Server Performance	Threads
	Min	Max	Avg.	Min	Max	Avg.	Messages/Hrs	Messages/Hrs	
1	20	13023	154	1	134	3.2	56478	56475	50
3	32	18756	167	1	127	5.40	57451	57450	50
5	21	24121	165	1	65	4.90	56450	56445	50
7	28	22313	187	1	87	5.50	54450	54445	50
10	24	31212	169	1	454	8.3	55605	55605	50

8. ADVANTAGES OF THE DISCUSSED WORK

The elasticity technique empowers a cloud system to handle the incoming requests more effectively. It is well capable of handling sudden load requirements via its dynamic decision technique to modify the virtual server environment. This results in increasing system performance, maintaining higher resource utilization and reducing energy cost.

9. CONCLUSION

In this paper, cloud elasticity scenarios have been discussed. The cloud architecture discussed in this paper consisted of a cloud elasticity framework which contained different components like Front-end load balancer, a Virtual cluster monitor system and an Auto-provisioning system. We have seen that the elasticity techniques are capable of handling sudden load requirements, increasing system performance, maintaining higher resource utilization and reducing energy cost. This will ultimately result in reducing cost and increasing the performance of the overall system.

10. ACKNOWLEDGEMENTS

The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance.

11. REFERENCES

[1] Zhen Xiao, Senior Member, IEEE, Qi Chen, and Haipeng Luo, Automatic Scaling of Internet Applications for Cloud Computing Services, IEEE TRANSACTIONS ON COMPUTERS, VOL. 63, NO. 5, MAY 2014

[2] Jianfeng Zhan, Member, IEEE, LeiWang, Xiaona Li,Weisong Shi, Senior Member, IEEE, Chuliang Weng, Wenyao Zhang, and Xiutao Zang, Cost- Aware Cooperative Resource Provisioning for Heterogeneous Workloads in Data Centers, IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 11, NOVEMBER 2013

[3] Sangho Yi and Derrick Kondo,INRIA Grenoble Rhne-Alpes, France and Artur Andrzejak , Zuse Institute Berlin (ZIB), Germany Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud published in EC project eXtreemOS (FP6-033576) and the ANR project Cloudshome (ANR-09-JCJC-0056-01)

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. HoR. Neugebauer, I. Pratt,and A. Warfield, Xen and the art ofvirtualization, in Proc. ACM Symp. Oper. Syst. Princ.(SOSP03),Oct. 2003, pp. 164177.

[5] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker, Usher: An extensible framework for managing clusters of virtual machines, in Proc. Large Install. Syst. Admin. Conf. (LISA07), Nov. 2007, pp. 116.

[6] B. Sotomayor et al., F. Capacity Leasing in Cloud Systems Using the Opennebula Engine, Proc. Cloud Computing and Applications (CCA 08), 2008.

[7] M.R. Palankar et al., Amazon S3 for Science Grids: A Viable Solution? Proc. Intl Workshop Data-Aware Distributed Computing (DADC 08), pp. 55-64, 2008.

[8] W. Zhou et al., Scalable Group Management in Large-Scale Virtualized Clusters. http://arxiv.org/abs/1003.5794, 2011.

12. AUTHORS 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.

Prof Ms Arti Mohanpurkar received the B.E. and M.E degrees in Computer Science Engineering, and pursuing Ph.D. Now she is HOD of Computer Engineering Department, Dr. D. Y. Patil School of Engineering & Technology, Savitribai Phule Pune University .India