International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 6 20 – 25

Variability in Behavior of Application Service Workload in a Utility Cloud

Monika Sainger
Research Scholar (CSE)
Mewar University, Rajasthan

Dr. K. P. Yadav
Supervisor
IIMT, Greater Noida

Dr. H. S. Sharma
Supervisor
Mewar University, Rajasthan

Abstract— Using the elasticity feature of a utility cloud, users can acquire and release resources as required and pay for what they use. Applications with time-varying workloads can request for variable resources over time that makes cloud a convenient option for such applications. The elasticity in current IaaS cloud provides mainly two options to the users: horizontal and vertical scaling. In both ways of scaling the basic resource allocation unit is fixed-sized VM, it forces the cloud users to characterize their workload based on VM size, which might lead to under-utilization or over-allocation of resources. This turns out to be an inefficient model for both cloud users and providers. In this paper we discuss and calculate the variability in different kinds of application service workload. We also discuss different dynamic provisioning approaches proposed by researchers. We conclude with a brief introduction to the issues or limitations in existing solutions and our approach to resolve them in a way that is suitable and economic for both cloud user and provider.

Keywords-*Elasticity, variability, application service workload, dynamic provisioning.*

*****

I. INTRODUCTION

In a cloud computing environment, elasticity refers to the user's ability to acquire and relinquish resources on-demand. Here, applications with time-varying workloads can request for variable resources over time that makes cloud a convenient option for such applications. In the traditional computing model, users would own and maintain resources which can meet the demand of peak workload. Rest of the times, resources would be under-utilized. Elasticity in clouds is the feature that makes users enable to not to own the resources for peak workload, rather request for more resources when demand increases and release resources when not required. The ability to pay for use of cloud resources eliminates the up-front commitment for resources by cloud users [1].

II. ELASTICITY DEFINED

Elasticity, being one of the central characteristics of cloud computing, is still used by different researchers and cloud providers to mean in different ways. Open Data Center Alliance (ODCA) defines elasticity [2] as "the configurability and expandability of the solution. Centrally, it is the ability to scale up and scale down capacity based on subscriber workload".

Herbst et. al [3] have discussed the following definition: "Elasticity is the degree to which a system is able 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." They define elasticity metric in two cases: elasticity while scaling up the resources on increasing workload and, elasticity while scaling down the resources on decreasing demand. They measure elasticity by the delay in acquiring resources and the amount of under-provisioned resources when the demand increases. When demand decreases, elasticity is measured by delay in releasing resources and amount of over-provisioned resources. In paper [4], elasticity is calculated by finding dynamic time warping (DTW) [5] distance between the demand (required resources) and supply (allocated resources).

In a report, Kuperberg et. al [6] have identified several characteristics of elasticity such as effect of reconfiguration which is nothing but the amount of resources added/re- moved with respect to change in workload, how frequent are reconfiguration points, re- action time by the system to adapt to the changed resource configuration. A Quality of Elasticity (QoE) metric based on a weighted sum of several factors has been proposed by Mika et. al [7]. He considered the factors like price-performance ratio which quantifies the performance received for a certain expenditure related to scaling out the deployment infrastructure, infrastructure pricing by cloud provider, billing granularity, VM provisioning speed etc.

All of the research above mainly emphasize on how the system reacts to the changes in workload. However, when measuring elasticity, an important consideration must be to choose the workloads which exhibit significant variability because otherwise the measurement can be misleading. Hence, in this paper, the notion of capturing variability in workload is discussed first which is then connected with the elasticity of the system.

III. VARIABILITY IN APPLICATION SERVICE BEHAVIOR (WORKLOAD)

We introduce in this paper, variability, a term to characterize workloads that exhibit significant change in their resource demand that is variable with time. We can measure workloads by several metrics depending on the kind of workloads. For CPU intensive workloads, number of tasks/jobs per unit time can be used as an indicator of workload. For I/O intensive jobs, number of requests per unit time reaching the server can be an indicative workload. In this paper, mainly I/O workloads are considered for case study and workload metric used is request rate. Further, variability of workload can be defined using several characteristics of workload. Here, a gradient based approach is used to

 ISSN: 2321-8169
20 – 25

measure variability. In this approach, an approximate gradient (since the workload is discrete) is calculated at each point of the workload. We can calculate approximate gradient as follows:

$$\Delta(W(t)) \approx \frac{W(t) - W(t-1)}{(t) - (t-1)} = W(t) - W(t-1) \quad (1)$$

Where $W(t-1)$ and $W(t)$ denotes the workload at time $t-1$ and t respectively, and $\Delta(W(t))$ denotes the approximate gradient or the change in workload at time t. It is positive when the workload increases and negative when it decreases. When the workload is constant, the gradient is zero at that point as gradient denotes the rate of change of workload. However, if there is a slight change in workload, gradient at those points would be approximately close to zero. Hence, to check whether the approximate gradient is close to zero or not, a threshold is used and the absolute value of gradient $|\Delta(W(t))|$ is compared against the threshold. This threshold is set on the workload so that it captures if there is a big divergence between the workload in the previous cycle and next cycle. Variability is calculated as a percentage of points where the approximate gradient is greater than the threshold over all the points. Mathematically, if T denotes the threshold, then variability can be defined using a step function s( ) which maps the positive arguments to 1 and negative arguments to 0. If the absolute gradient is greater than threshold, then the difference ($|\Delta(W(t))| - T$) would be positive. Hence, taking the summation of step function of this difference expression at each point would count all of the points where the workload is variable. Using this, the variability can be:

$$\text{Variability} = \sum_{t=1}^{n} \frac{s(|\Delta(W(t))| - T)}{n} \times 100 \quad (2)$$

where, n is the length of the workload, T denotes the threshold, $|\Delta(W(t))|$ denotes the absolute value of approximate gradient, and step function s() is defined as follows:

$$s(t-a) = \begin{cases} 1 \text{ for } t \geq a \\ 0 \text{ otherwise} \end{cases}$$

We have demonstrated different kind of workloads to evaluate the variability fig(1): i) A square shaped wave of a two-level step workload, ii) a periodic workload with sine wave shape, iii) web server workload we used in our work, iv) random noise. Their respective approximate gradients are calculated and shown in fig (2). The variability calculated as a percentage for all of the workloads is 7.94%, 82.84%, 60.25% and 86.61% respectively with a threshold of 0.04 (almost zero).

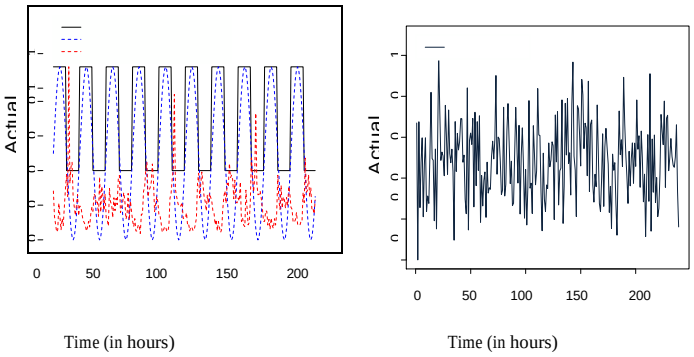

Fig 1.Different workloads to evaluate variability metric

It shows that the noise is highly variable, as expected. Square wave shaped workload is the least variable since the most of the time it remains constant. Sine wave shaped workload is variable 80% of the total time except for the topmost and bottommost points where it is remains almost constant. Web server workload is less variable than perfect sine shaped workload, but it still has significant variability. However, the mere fact that variability metric for square wave shaped workload is low does not imply that it does not require elastic resource allocation. Rather the fact is that it needs elastic resource allocation at a larger scheduling window rather than at every scheduling cycle like in case of Sine wave shaped workload. Here, variability attempts to identify those workloads which need short-term dynamic resource allocation decisions, or which need frequent scheduling of change resource allocation.

As shown in fig. (2), the workload is actually continuous and it has been converted into discrete form by averaging it out over a time period. This time period plays an important role in the correctness of finding variability. It is important that this time period should be small enough to capture the variations of the workload. If this time period is too large, there is a possibility of missing out the important variations in the workload. Intuitively, if the variability does not change on decreasing the time period further, then that time period can be selected to represent workload. Further, considering workloads which have considerable variability is important when measuring and defining elasticity of a system. Hence this metric is used in defining elasticity of the system in the next subsection.

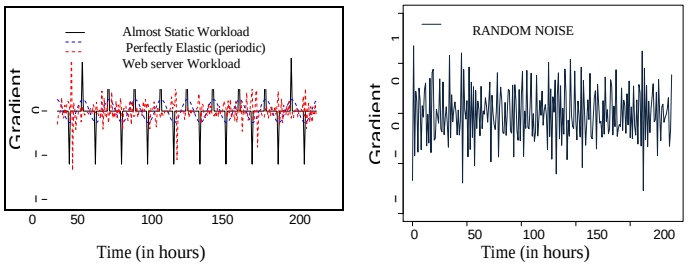

Fig. 2. Gradient of Workloads

Generally there are two classes of definitions of elasticity; one is based on system factors such as time to allocate resources once the resources are requested, reaction time of the system after the resources have been allocated, how often the configuration of the system can change and unit of resource

allocation (VM size) etc., another class is based on closeness of requirement and supply of resources which actually encompasses all factors described above. This is because all of the factors like delay in allocation of resources, scheduling points etc. finally cause a mismatch in demand and supply curve, leading to over-allocation or under-allocation. In our work we refer to the second class of definition of elasticity. In [8] [9], elasticity is defined as follows:

Considering the workload is significantly variable, elasticity of a resource allocation system S is defined by the closeness of resource requirement RR (t) and the allocated resources RS (t) with respect to change in workload W (t).

Here, resource requirement is used to denote the amount of resources which are sufficient enough such that performance of application does not go down (service level agreement violations do not occur) and at the same time, they are not over-provisioned. The point worth noticing is that the resource requirement depends on the service level agreement (SLA). For example, if the SLA mentions that the average response time of the re- quests for an application hosted on cloud to be less than 100 ms, the resource requirement could be higher than the SLA which mentions the average response time limit of 200 ms. The another term used in this definition apart from the previous definitions is that the workload is considered to be significantly variable, which means that the variability of the application workload being considered for measuring elasticity of the system should be greater than a threshold.

Further, the closeness of resource requirement and provisioning can be measured by number of metrics. Intuitively, for a significantly variable application workload, if the resource allocation matches exactly with the resource requirement, then the system is perfectly elastic. One resource allocation system S1 is more elastic than the other S2 if S1 allocates resources more closer to requirement. Fox example, if S1 takes lesser time to allocate resources than S2 after the requirement is known, then S1 is more elastic as the allocation and requirement would be closer for S1. Similarly, if S1 predicts the resource requirement more accurately than S2, then again S1 would be more elastic.

IV Elasticity in current IaaS cloud systems

Generally, in current IaaS cloud systems, the resource allocation unit is a fixed-size virtual machine (VM). Hence, from Infrastructure-as-a-Service point of view, elasticity means to acquire and relinquish VMs dynamically based on requirement. Usually, IaaS cloud providers provision a fixed set of VMs with different configurations. For example, Amazon provides few standard instance types (VMs) [10] like small, medium, large, extra large instance with increasing resource configuration. Apart from the standard instances, it also provides resource-specific instances like High- Memory, High-CPU, High-I/O instances. To scale the resources, users can use the scaling strategies mentioned below.

Horizontal Scaling: It is a scaling method in which resource scaling is achieved by adding (or removing) more number of VMs to support the changing demand of application. For the horizontal scaling to be a feasible option, the user application must be designed in a way such that it can be distributed onto multiple machines i.e. a multitier application. Also, one component can itself be built to distribute among different VMs and then using a load balancer to distribute the workload among the VMs. Fig. 3shows the how horizontal scaling takes place in IaaS. Here, VM1, VM2 and VM3 run the same component of the application and a load balancer distributes the load among these VMs. For horizontal scaling, another VM can be added and the load balancer redistributes the load among all of the VMs. The main advantage of horizontal scaling is that powerful servers are not needed to support the increased workload of applications; rather commodity servers can be used to do the same in a distributed fashion.

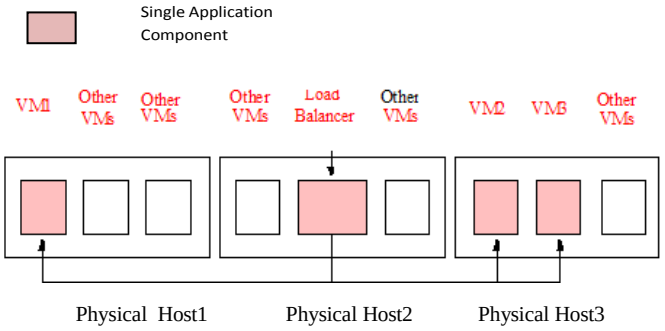

Fig.3 Horizontal scaling in IaaS

Vertical Scaling: It is a scaling method in which the change in workload is handled by migrating the application to a different VM (might be on a different physical host). Fig. 4shows how the vertical scaling can be achieved in Amazon EC2 IaaS Cloud by migrating the application among small, medium and large instance VMs. Although migration is a costly operation and incurs some penalty in terms of availability and performance during migration, but for applications which can't scale horizontally, resource scaling is achieved by migration only.

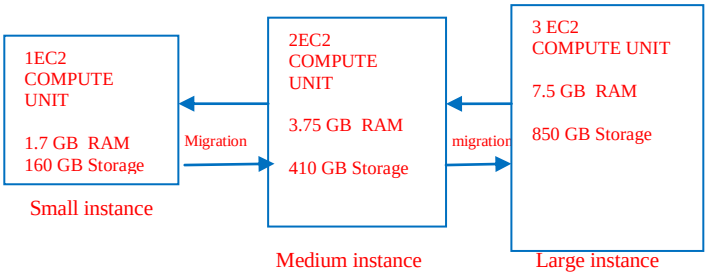

Fig. 4 Vertical Scaling in Amazon EC2 Iaas Cloud

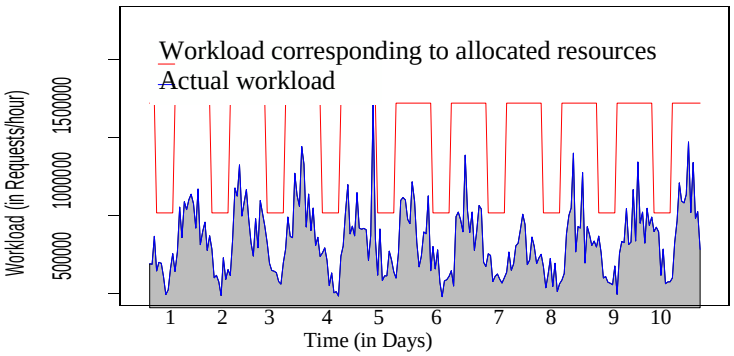

Fig. 5 Typical workload characteristics of web application

Fig. 5 shows the actual workload of a web server hosted in our institute. In the figure the x-axis represents time in days and the y-axis indicates the web server workload, measured as the number of HTTP requests per hour, received by the server. High number of requests is received by the server during the day as compared to night times. Using the existing resource provisioning model, for this workload, a user would demand two types of VMs as represented by the peak and trough of the workload corresponding to allocated resources curve, for the vertical scaling. In case of horizontal scaling, the peak might represent workload corresponding to two VMs and trough might correspond to workload for one VM. From the Figure, it is visible that for supporting an average request rate of about 200 requests/hour, one might allocate a server of capacity 500 requests/hour so as to handle the maximum load. This obviously, leads to idle resources most of the time. Hence this mode of allocation is coarse-grained because it does not change in accordance to the variations observed in the workload. Further, calculations show that the effective utilization of resources as per this allocation is just 45.5307% (assuming linear relationship between workload and resources), which is the ratio of area under curve of the actual workload to the workload corresponding to allocated resources. In such cases, cloud users end up paying more than what they actually use. Existing provisioning models are not very efficient for the cloud providers too. Although, there are idle resources available, the provider can not release them for better usage. In summary, static allocation of VMs in the current IaaS systems leads to the following problems:

- Users are forced to characterize their resources on coarse-grained level because of static VM sizes leading to under-utilized resources or under-performing applications.
- Cloud users pay even for the idle resources.
- Idle resources cannot be further allotted by the cloud providers.

So, variable resource requirement by variable workload cannot be met by static allocation. Hence, a more flexible and dynamic resource allocation mechanism is required that would help to achieve fine-grained resource allocation close to requirement. Various researchers have proposed dynamic provisioning proposals which change the allocation based on a trigger decision.

V Dynamic Provisioning Techniques

In this section, we discuss various proposals by researchers related to dynamic provisioning techniques. There are two broad categories in which these techniques can be classified. First category includes the techniques where provisioning decisions are taken based on the immediate state of the system. Second category includes the techniques where provisioning is done based on a forecast using previous resource usage history, resource usage trend, load stability etc.

A. Reflexive Scheduler based techniques

The term Reactive scheduler means the scheduler that takes provisioning decisions based on the immediate state of the system. One of the practically used such frameworks is Auto Scaling [11] which is a service provided by Amazon where users can declare various rules based on which the scaling of the resources is done automatically on behalf of the user in Amazon EC2 cloud. It can scale the resources periodically by the specified schedule by the user (for example, everyday at 12:00:00). It can also scale the resources dynamically by specified conditions from the user (like change in system load). The conditions can be specified by the user based on the values of some metrics (which can measure system load, for example). Amazon provides some inbuilt metrics or users can define their own metrics, which are measured by a monitoring engine, called Amazon CloudWatch. CloudWatch can be configured to generate alarms based on the conditions specified by the user, which in turn, can trigger the scaling activities.

Figure 6 demonstrates how Amazon Auto Scaling works and its integration within the system [12]. Consider the AWS user as a SaaS Provider, which hosts its web application onto the Figure 2.9: Limitations in dynamic provisioning techniques based on immediate state
EC2 cloud. AWS user sets up the Launch Configuration to launch new instances, CloudWatch metrics to monitor, policies to trigger CloudWatch Alarm, and scaling policies based on the CloudWatch Alarm. Typically, an AWS user would configure scale-in and scale-out policies for the increasing and decreasing system load. For example, alarms can be configured that trigger auto scaling policies to launch

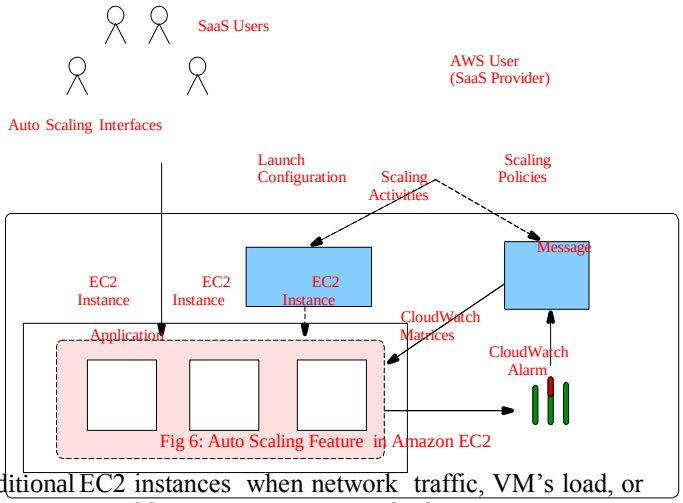

Fig 6: Auto Scaling Feature in Amazon EC2

additional EC2 instances when network traffic, VM's load, or other measurable statistic, gets too high, say 90% usage. Depending on the application requirements, the scaling policies can use both horizontal and vertical scaling. This shows a close feedback based loop, which takes scaling actions based on monitoring feedback. Hence, using Auto Scaling feature, users can use elasticity automatically. However, the CloudWatch measures the VM's load and take decisions based on that. But in certain case, VM's usage might not be too high but the server can still be saturated because of virtualization overhead. CloudWatch and other such commercially available monitoring engines have this limitation that they cannot measure virtualization overhead.

B. Forecasting based techniques

The above limitations can be overcome using forecasting based techniques that derive meaningful prediction based on the history of resource usage, workload, etc.. In [14], Bobroff et al. have introduced

dynamic server migration and consolidation algorithm. They stated that variability in the workload accrue the benefits of dynamic placement of VMs in terms of reduced resource allocation, shown by an analytical formula that they have derived. The forecasting algorithm that they use first removes periodic components in the resource usage (that they represented in form of time series), then it represents the residuals by Auto-regressive process. Their results show the reduction in number of physical machines as compared to static allocation. Their paper also discusses the problem in virtualized domain but the approach can be directly applied to cloud systems.

Gong et al. in [15] have proposed PRESS (PRedictive Elastic ReSource Scaling) scheme for cloud systems. For workloads with repeating patterns, PRESS derives a signature for the pattern of historic resource usage and uses that signature in its prediction. To calculate the period of repeating pattern, they calculate the Fast Fourier Transform (FFT) and find out the most dominating frequency out of it. For workloads without repeating patterns, PRESS uses discrete-time Markov chain with a finite number of states for short-term prediction.

Lim et al. in [17] have applied a control theory based approach where in they build an elasticity controller, which dynamically adjusts the number of virtual server instances in cloud and rebalances the load among servers. A control policy based on a performance metric provides the feedback to the controller to take appropriate actions.

Gemma Reig and Jordi Guitart in [18] propose a Prediction System that combines statistical and Machine Learning techniques. For immediate prediction of resources (CPU demand in their case), they use basic statistical techniques like Local Linear Regression, Moving Average, Last value prediction. Machine learning based techniques have been applied for the long term prediction of resources for capacity planning.

VI Conclusion

In the previous research works it has been shown that reactive scheduler based technique of provisioning always lags the demand in time as can be seen by their results in Fig. 7 (from [13]). Here, dotted lines denote the number of VMs provisioned in the current state, and dark line shows the workload in terms of number of jobs. When it is identified that provisioned resources are not consistent with the required ones, more VMs are provisioned. Due to this lagging effect, the newly added VMs take some time to boot up, get configured and handle the increased workload. This causes SLA violations for some amount of time, whenever there is an increase in demand.

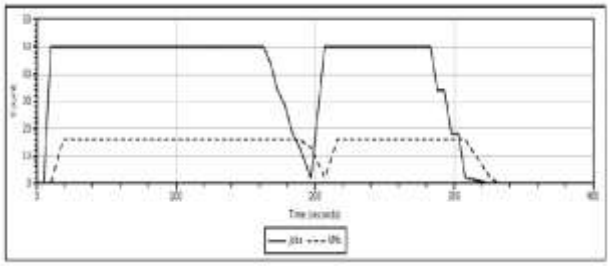

Fig. 7 limitations in dynamic provisioning techniques based on immediate state.

The research works using forecasting based technique for prediction focus mainly on building elastic systems aiming for the accurate prediction while keeping SLA violations minimum. Though PRESS [15] does a padding of 5-10% in the actual prediction to avoid SLA violations, but they did not give basis for this number. Further, they have used a constant percentage, whereas the padding is workload dependent. Then, CloudScale [16] applies different techniques to find the appropriate amount of padding, but the formulation of proper SLA penalties in terms of performance metrics is missing.

Also, most of the elastic IaaS approaches deal with VM CPU dynamic provisioning using history of VM usage only. However, VM usage is not the true indicator of the workload of customer. Instead, for example in case of I/O workloads, request rate could be a true indicator of the workload. And, if the allowable response time limits are allowed to be changed, then the resources required to sustain the same workload (such that SLA violations do not occur) could be different. Hence, working with resource usage for the application can potentially lead to wrong interpretations.

Form the above discussion we can say that for variable workloads, significant reduction in the resource allocation can be achieved using dynamic allocation of resources along with almost negligible SLA violations. For that we need to have an elastic framework that enables fine grained resource allocation by dynamically adjusting the size of VM as provided by a forecasting module. Forecasting module forecasts the user workload, which is then translated into resource requirements based on a cost model. The basic intuition of this cost model is to reduce the resource cost of the user by enabling resource allocation closer to what is actually used, without compromising on the application performance.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672.

[2] "Open data center alliance: Compute infrastructure as a service rev, 1.0," 2012. [Online]. Available: http://www.opendatacenteralliance.org/docs/ ODCA Compute IaaS MasterUM v1.0 Nov2012.pdf.

[3] R. R. Nikolas Roman Herbst, Samuel Kounev, "Elasticity in cloud computing: What it is, and what it is not," in ICAC 2013, To be published, 2013.

[4] N. R. Herbst, "Quantifying the Impact of Configuration Space for Elasticity Bench- marking," Study Thesis, Faculty of Computer Science, Karlsruhe Institute of Tech- nology (KIT), Germany, 2011.

[5] E. Keogh, "Exact indexing of dynamic time warping," in Proceedings of the 28th international conference on Very Large Data Bases, ser. VLDB '02. VLDB Endowment, 2002, pp. 406–417. [Online]. Available: http://dl.acm.org/citation.cfm?id=1287369.1287405 5

[6] J. v. K. Michael Kuperberg, Nikolas Herbst and R. Reussner, "Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms," Informatics Innovation Center, Karlsruhe Institute of Technology, Karlsruhe, Germany, Tech. Rep., 2011. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000023476 6

[7] M. Majakorpi, "Theory and practice of rapid elasticity in cloud applications," Study Thesis, Department of Computer Science, UNIVERSITY OF HELSINKI, Swedish, 2013. 7.

[8] J. Weinman, "Time is Money: The Value of On-Demand," Jan. 2011. [Online]. Available: www.joeweinman.com/Resources/Joe Weinman Time Is Money.pdf

[9] S. Islam, K. Lee, A. Fekete, and A. Liu, "How a consumer can measure elasticity for cloud platforms," in Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ser. ICPE '12. New York, NY, USA: ACM, 2012, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/2188286.2188301 .

[10] "Amazon EC2 Instance Types," 2013. [Online]. Available: http://aws.amazon. com/ec2/instance-types/ .

[11] "Amazon Auto Scaling," 2013. [Online]. Available: http://aws.amazon.com/autoscaling.

[12] "Amazon Auto Scaling Developer Guide." [Online]. Available: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/Welcome.h%tml

[13] M. Murphy, B. Kagey, M. Fenn, and S. Goasguen, "Dynamic provisioning of virtual organization clusters," in Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, 2009, pp. 364–371.

[14] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, 21 2007-yearly 25 2007, pp. 119 –128.

[15] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in Network and Service Management (CNSM), 2010 International Con- ference on, oct. 2010, pp. 9 –16.

[16] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in Proceedings of the 2nd ACM Symposium on Cloud Computing, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 5:1–5:14. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038921.

[17] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," in Proceedings of the 7th international conference on Autonomic computing, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1809049.1809051.

[18] G. Reig and J. Guitart, "On the anticipation of resource demands to fulfill the qos of saas web applications," in Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, sept. 2012, pp. 147 –154.