RESEARCH ARTICLE OPEN ACCESS

Analysing The Cloud Services From Perspective of Elasticity

B.Susrutha [1], I.Shalini [2]
Assistant Professor
Department of Computer Sciene and Engineering

ABSTRACT

"Cloud computing platforms are in fad. These are being used extensively by some innovative companies with huge and abnormally growing computational needs, and traditionally enterprises are looking closely at the cloud as boon to run their own IT infrastructure. Many features of cloud platforms are excellent and attractive; let me give an example, cloud platforms may be low- cost by exploiting economies of vast scale and they might achieve high availability by replication and distribution. A major factor that is responsible for high sales for cloud platforms are that they provide high elasticity. In this paper, we focus on understanding elasticity of cloud services, and techniques for monitoring and evaluating them and also the characteristics of a platform that influence its elasticity. We introduce a new dimension less measure for elasticity. We see also that elasticity is a surprising measure that shows little correlation to performance and cluster size

Keywords:- Elasticity, Cluster Size, Cloud platforms

I. INTRODUCTION

In cloud computing, **elasticity** is defined as "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".Elasticity is a defining characteristic that differentiates cloud computing from previously proposed computing paradigms, such as grid computing. The dynamic adaptation of capacity, e.g., by altering the use of computing resources, to meet a varying workload is called "elastic computing".The concept of elasticity has been transferred to the context of cloud computing and is commonly considered as one of the central attributes of the paradigm.

Elasticity,[3] the ability to rapidly scale resources up and down on demand, is an essential feature of public cloud platforms. However, it is difficult to understand the elasticity requirements of a given application and workload, and if the elasticity provided by a cloud provider will meet those requirements .It allows cloud service developers and providers to trace their service behavior from the whole service level to the underlying [1]virtual infrastructure, extracting characteristics and providing crucial insight in their elastic behavior.

On a typical IaaS (Infrastructure as a Service)[3] cloud platform the elasticity infrastructure enables auto-scaling of instances for an application with user customised rules which periodically fire to check metric values, make decisions, and request an action in response ."Cloud Computing[8] is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications and services that can be rapidly provisioned and released with minimal management effort or service provider interaction". Five essential characteristics of cloud computing listed by NIST are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service.

II. WHY TO ADOPT CLOUD COMPUTING

Cloud computing[8] is explored and adopted by most of the companies, research institutions, academia and end users. The main reason to adopt cloud is the hypothesis of gettingresources in an elastic manner. As a matter of fact,elasticity is a fundamental characteristic in the cloud and provides resources elastically on on-demand basis.

The concept of cloud elasticity is the power to scale down and scale up of system resources based on workload variations of cloud users by adjusting the competence of cloud system resources. Cloud elasticity is frequently related with the scalability, an elusive difference presents between scalability and elasticity when observing a behavior of cloud system.

Scalability is frequently used to depict the processes and mechanisms employed to a system workload with properties such as proper capacity planning, predictable approach and modifications which can be formed manually. In elasticity conception, where resources are utilized by instances and workload changes occur dynamically in a short period of time. In addition, resources are allocated flexibly with fully automated or minimum manual support.

Ideally a cloud platform is infinitely and instantaneously elastic. An application could be scaled out indefinitely with increasing load, and this could happen as fast as the load increases with no degradation of response times. Resources would be available instantly and the application would be immediately deployed and available for use. A perfectly elastic cloud platform would be ideal for hosting interactive applications with strict response time requirements, and with spiky unpredictable workloads. These are difficult to host on traditional fixed and finite infrastructures as the quantity of resources is not known in advance, and the cost of keeping the resources available for occasional extreme load events is prohibitive.

However,[3] real clouds are not perfectly elastic. There will inevitably be a delay between when resources are requested, and when the application is running and available on it. The resource provisioning speed may depend on a number of factors including: the type of cloud platform (e.g. Infrastructure vs. Platform as a Service); the type, cost model, number, size, or speed of resources requested; the availability of spare resources in the requested region and the demand on the cloud platform from other users; the rate of increase (acceleration) of the workload; and any quotas or limits imposed by the cloud platform or the contract with them.

Cloud computing [7]enhances the mechanisms for sharing of remote resources by providing users with control over the remote resources (e.g. control over their configuration). Clouds also provide a virtualized platform for users to create and manage the software stack from the operating system to the applications. This particular type of cloud is known as an Infrastructure-as-a-Service (IaaS) cloud, as opposed to Platform-as-a-Service (PaaS) or Software-as-a-Service (SaaS) clouds. The customizability, complete control over the software stack, and on-demand access to IaaS clouds make them an attractive solution to the problem of dynamically extending the resources of a static site to adjust to changes in demand. Elastically extending sites with cloud resources poses a number of challenges that must be addressed. First, leveraging remote resources highlights the following security concerns:

How can we establish trust between the site and the added remote nodes? How can we provide an environment that would allow us to project assumptions underlying security mechanisms within the site onto a wide-area network? Users and the cloud providers have different perceptions of elasticity. Considering the abstractions provided by the cloud, the users only have access to interfaces, whereby is possible to access “infinite resources”. [6]In turn, the provider is responsible for acquiring, managing and upgrading the cloud infrastructure. In addition, it must also provide the mechanisms
to enable elastic provision of resources. Some cloud platforms provide interfaces and API’s that allow users changing its resources
manually. Other clouds provide solutions that include fully automated monitoring services, automatic allocation of resources and even load balancing. The solutions developed by academy are similar to those provided by commercial providers, but include new approaches and techniques for elastic resources provisioning.

Although many elasticity mechanisms have been proposed in the research literature and the commercial field in last years, there are no published works addressing the state-ofthe-art of cloud elasticity.Elasticity as the ability for customers to quickly request, receive, and later release as many resources as needed. The elasticity implies that the actual amount of resources used by a user may be changed over time, without any long-term indication about the future resources demands Ideally, to the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased
in any quantity at any time [6]. An elastic application is the one that is able to automatically adapt itself to changes in resources amount or request/release resources.

A key idea in our proposal to measure elasticity of a platform is to use a mix of workloads, that vary over time in different ways. Some workloads will rise and fall repeatedly, others will rise rapidly and then fall back slowly etc. For each workload, we examine the way a platform responds to this, and we quantify the impact on the consumer's finances. That is, we use a cost measure in dollars per hour, with a component that captures how much is wasted by paying for resources that are not needed at the time (overprovisioning), and a component to see how much the consumer suffers (opportunity cost) when the system is underprovisioned, that is, the platform is not providing enough resource for a recent surge in workloads.

III. FRAMEWORK FOR MEASURING ELASTICITY

The main objective of this paper is [5] to provide a way for a consumer to measure how well (or not) each cloud platform delivers the elasticity property. As in any hype-prone field, it is important to get common usage of marketable terms such as \elastic". Several efforts have tried to explain the meaning of this term, and others that are used about cloud platforms, and along with explanation, they point to aspects of the platform's performance that can be important for elasticity. I draw attention to the value of cloud elasticity as compared to the conventional client-server model in the context of perceived risks due to over- and under- provisioning.

The prestigious National Institute of Standards and Technology (NIST) has attempted to explain and define terms for the cloud1. For elasticity, NIST points to rapid provisioning and de-provisioning capability, virtually infinite resource capacity with unconstrained purchasable quantity at any single moment.

Elasticity when load declines is not only a function of the speed to decommission a resource, but also it depends on whether charging
for that resource is stopped immediately on decommissioning, or instead is delayed for a while (say till the end of a charging quantum of time). There has been significant work on developing new ways to conduct performance evaluation of public cloud providers, looking especially at aspects such as price/performance and scalability.A number of recent research efforts conducted in-depth performance analysis on the virtual machine instances offered by public cloud providers. For example, Stantchev et alintroduce a generic benchmark to evaluate the non-functional properties(e.g., response itime, transaction rate,availability etc) of individual cloud offerings for web services from cost-beneft perspective.
In general,[1] we have three main views from which cloud services
are described (Fig. 1): (i) design-time view, where we see the whole service dependency model (Cloud Service, Service Topology, Service Unit) and user-defined service requirements, (ii) run-time view, where instances of several Service Units are deployed and executed in virtual machines, and (iii) the virtual infrastructure, where several virtual machines, possibly grouped in virtual clusters, are used. From the design-time point of view, the DaaS has two service topologies, Data End and Event Processing, supporting horizontal scaling by addition and removal of VMs. The Data End service topology includes two service units, a Data Node holding data, and a Data Controller managing it. The Event Processing service topology also contains two service units: Load Balancer, distributing client requests, and Event Processing interacting with the Data End. At run-time, the Data End units uses Cassandra2 for its service units, and the Event Processing uses HAProxy3 for its Load Balancer service unit. Moreover, at run-time, due to user-defined elasticity requirements, service unit instances are added/removed dynamically, triggering allocation and deallocation of virtual machines at the virtual infrastructure level.
User-defined service requirements can be viewed as elasticity requirements, as they restrict the elastic behavior of the DaaS. To enforce such restrictions at the virtual infrastructure level, i.e. when to add/remove a VM, of what type, under which pricing scheme, the restrictions need to be linked and mapped to the run-time view. Monitoring data from therun-time view must also be linked back to the user-defined elasticity requirements. This enables the discovery of service units belonging to service topologies that cause requirements violations.

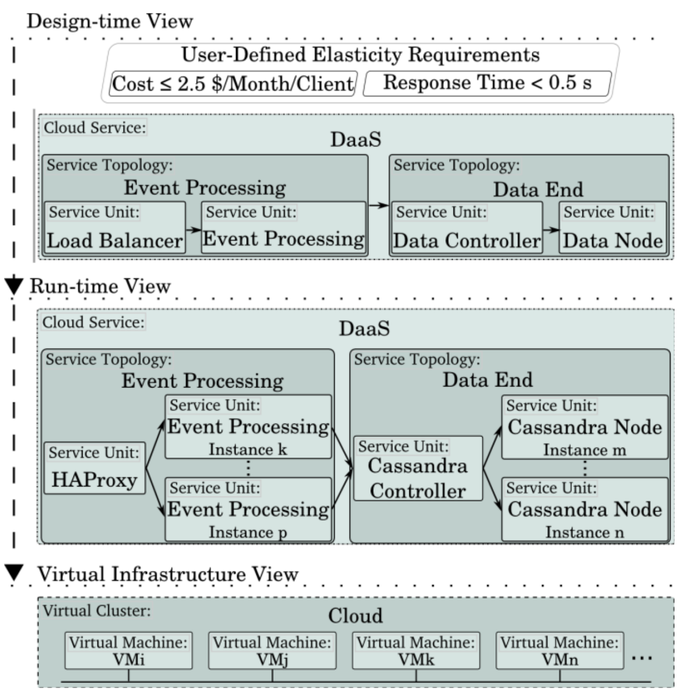

Fig. 1: Elastic Cloud Service Views

While using various monitoring techniques such as or we can capture monitoring data from the whole service level or virtual infrastructure level, such data typically does not answer the following crucial questions:

1) what should be the behavior of the service topologies and units when user-defined elasticity requirements are fulfilled

2)when is the service behavior elastic, i.e. adapting and fulfilling user-defined elasticity requirements

3)what is the cause of an elasticity requirement violation, traced from the design-time view to the underlying virtual infrastructure

4)How does the service elastic behavior evolve in time, i.e. what are the correlations and patterns in the service behavior

Capturing, describing and analyzing elastic behavior of cloud service is crucial not only for developers who build and optimize cloud services, but also for software controller that change the topology of such services at run-time, enforcing user-defined elasticity requirements. Such controllers need a monitoring and analysis mechanism that extracts elasticity characteristics, which can be used to refine user-defined elasticity requirements or predict the service behavior, leading to better service control and quality. This motivates us to investigate the following issues:

A)Which concepts can be used to capture the elastic behavior
of cloud services?

B)How to extract characteristics that describe the service elastic behavior to support both reactive and predictive control of elastic services?

C)How to analyze the cloud service behavior, detecting the source of user-defined elasticity requirements violations?

This paper focuses on capturing the properties of elastic services at multiple levels, providing support for analyzing their behavior from multiple views, and characterizing the elastic behavior of each cloud service based on user-defined elasticity requirements.

IV. A LOOK ON MEASURING ELASTICITY FROM VARIOUS PERSPECTIVES

On an ideal elastic platform, as application workload intensity increases, the distribution of the response times of an application service should remain stable as additional resources are made available to the application. Such an idealistic view is shown by the synthetic example in Fig.2, which also includes dynamic un-provisioning of resources as the application workload decreases. Note that the dynamic adaptation is a continuous (non-discrete) process in Fig. 2.

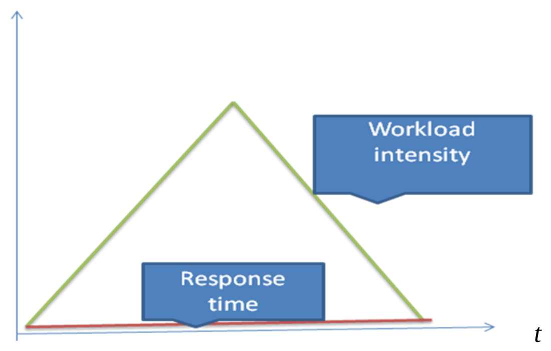

Fig.2. Schematic example of an (unrealistically) ideal elastic system with immediate and fullycompensating elasticity:

However, in reality, resources are actually measured and provisioned in larger discrete units (i.e. one processor core, one page of main memory, etc.), so a continuous idealistic scaling/elasticity cannot be achieved. On an elastic cloud platform, the performance metric (here: response time) will rise as workload intensity increases until a certain threshold is reached at which the cloud platform will provide additional resources. Until the application detects additional resources and starts making use of them, the performance will recover and improve - for example, response times will drop. This means that in an elastic cloud environment with changing workload intensity, the response time is in fact not as stable as it was in Fig. 2.

Now that we have determined a major property of elastic systems, the next question is: how to quantify elasticity of a given execution platform? Notice that it reflects the previously mentioned fact that the performance increases at certain discrete points. When quantifying to define and to measure elasticity, we have to quantify the temporal and quantitative properties of those points at which performance is increased.

4.1 Need for Intelligently Designed Workloads

Intelligently [9]designed workloads are extremely important when trying to witness elastic effects. A constant workload for example will never produce elastic effects on its own. In order to witness elastic effects workloads have to push the boundary of what already 10 provisioned resources can offer them, only then will a drop in performance and after that an increase be visible. One important aspect of designing workloads for elasticity benchmarking is to understand in which way the targeted execution platform scales and the triggering events which lead to elasticity. One example focusing on IBM Workload Manager (WLM) application can be found in [TV04]: When a low-priority process p1 is working alone on the execution platform, it can consume (almost) all available resources. Yet when a process p2 with a higher priority begins execution and demands resources currently used by process p1, the execution platform will re-assign resources to p2. Thus, the elasticity benchmark needs to explore the interplay between processes with different priorities. Additionally, as the resource demands of p2 rises, p1 will also have to release more and more of its resources to be assigned to p2. In this case we can witness elastic effects as p2's service demand and performance rise at the same time when new resources are provided to p2. Until now, existing benchmarks did not provide the functionality of workloads with the purpose of increasing workloads specifically to force resource reallocation.

4.2 An Approach for Measuring Resource Elasticity of Thread Pools

The evaluation concept of resource elasticity can **be** applied to various kinds of resources, even to virtualized ones that are not directly mapped to hardware resources. As a starting point and a proof-of-concept, we are now researching the elasticity behaviour of Java thread pools. Thread pools are an implementation of the pooling pattern, which is based on a collection (pool) of same-typed, interchangeable resources that are maintained continuously. Thread pools are heavily used in databases, application servers and other middleware/applications handling many concurrent requests. Similar to thread pools, connection pools (in DBMS drivers, such as JDBC) implement the pooling pattern with the same rationale. In a resource pool, even when a given resource instance in the pool is not needed, it is not released immediately because it is assumed that the instance may be needed in the near future: the runtime costs of releasing and (later) re-acquiring that resource are significantly higher than the costs of keeping an idle instance. After a certain time interval, the resource can be released when it hasn't been used - this "disposal delay" can be often be set in the implementations of the pool pattern. Beyond[9] "disposal delay", further configuration options are the minimum pool size (often called "core size"), the maximum pool size and the length of the queue that resides "in front of" the pool. For this report, we use the default thread pool implementation provided by the Java SE platform API, and the threads have to perform non-communicating, CPU-bound computation-only tasks in parallel. These task that form a configurable, multi-phased workload which we describe further below. The tasks consist of carefully-defined Fibonacci computation, with randomly-chosen starting values (to prevent function inlining as constant values), and with evaluation of the computation result (to prevent dead code elimination). We are especially interested in situations where the thread pool size increases in a "preventive" way, i.e. a pool where all threads are busy, and one arriving tasks triggers the addition of two new threads to the pool, even though just one new thread would be 11 sufficient. We are also interested in situations with a "fully-busy" where the arrival of a new task and the finish of another task's execution are temporally close, to see whether the pool immediately "overreacts" by allocating a new thread instance and accepts the task from the queue, rather than waiting for a (short) time in the hope that another task will be finished.

V.CONCLUSION

In this paper, we have presented a definition of resource elasticity in the context of cloud computing and virtualization. Starting with a definition of scalability, we have added temporal and quantitative aspects to define elasticity, and have presented the challenges of measuring elasticity in real-world scenarios. Finally, we have used threadpools as an elastic resource of the Java Virtual Machine. According to the presented works, we can verify that the elasticity Is already in use in the development of enterprise applications, especially, PaaS and client-server based applications. However, it still lack tools and frameworks to support the development of applications that could take advantage of the elasticity provided by IaaS clouds. In future this can be further enhanced to support and provide tools for IaaS clouds.

REFERENCES

[1] D.Moldovan,G Copil,HL Truong ,"MELA:Monitoring and analyzing elasticity of cloud services" Proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom 2013). doi:10.1109/CloudCom.2013.18..

[2] NR Herbst, S Kounev, R Reussner, **"E**lasticity in cloud computing,What it is and what it is not", Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24–28.

[3] P. Brebner, "Is your cloud elastic enough?: performance modelling the elasticity of infrastructure as a service

(iaas) cloud applications," in Proceedings of the third joint WOSP/SIPEW Intl. conference on Performance Engineering, ser. ICPE '12. ACM, 2012, pp. 263–266.

[4] T.Dory,B.Mejlas,P.Van Roy,N.Tran, "Comparative elasticity and scalability measurements of cloud databases".

[5] S Islam, K Lee,A Fekete, A Liu,"How a consumer can measure Elasticity for cloud platofrms", - Proceedings of the 3rd ACM/SPEC