Towards a Resource Elasticity Benchmark for Cloud Environments

Andreas Weber
Karlsruhe Institute of Technology, Germany
andreas.weber4@student.kit.edu

Nikolas Herbst
Karlsruhe Institute of Technology, Germany
herbst@kit.edu

Henning Groenda
FZI Research Center for Information Technology, Karlsruhe, Germany
groenda@fzi.de

Samuel Kounev
Karlsruhe Institute of Technology, Germany
kounev@kit.edu

ABSTRACT

Auto-scaling features offered by today's cloud infrastructures provide increased flexibility especially for customers that experience high variations in the load intensity over time. However, auto-scaling features introduce new system quality attributes when considering their accuracy, timing, and boundaries. Therefore, distinguishing between different offerings has become a complex task, as it is not yet supported by reliable metrics and measurement approaches. In this paper, we discuss shortcomings of existing approaches for measuring and evaluating elastic behavior and propose a novel benchmark methodology specifically designed for evaluating the elasticity aspects of modern cloud platforms. The benchmark is based on open workloads with realistic load variation profiles that are calibrated to induce identical resource demand variations independent of the underlying hardware performance. Furthermore, we propose new metrics that capture the accuracy of resource allocations and deallocations, as well as the timing aspects of an auto-scaling mechanism explicitly.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Benchmarking, Metrics, Cloud

Keywords

Resource, Elasticity, Load Profile, Demand, Supply

HotTopiCS'14 March 22 2014, Dublin, Ireland
Copyright 2014 ACM 978-1-4503-3059-6/14/03 ...$15.00
http://dx.doi.org/10.1145/2649563.2649571 .

1. INTRODUCTION

Cloud providers nowadays offer their services with a "Pay-Per-Use" accounting model to increase flexibility and efficiency with respect to traditional offers. Modern cloud platforms offer auto-scaling mechanism enabling *resource elasticity*. With an elastic cloud service, the provider dynamically adapts resource allocations according to the customer's demand and the customer pays for the actually consumed resources. This business model is referred to as "Utility Computing" [1].

Cloud providers strongly advertise their elastic capabilities. However, currently no benchmark methodology exists allowing to compare the elasticity of different offerings. Cloud providers offer tools that allow customers to implement scaling rules based on monitored system metrics. However, varying the parameters of these rules leads to different behaviors and finding the optimal parameter configuration is not trivial. An elasticity benchmark could help to compare these parameter configurations in an objective manner. Additionally, researchers have proposed various elasticity strategies that define adaptation processes for cloud systems [6, 9]. For such more complex strategies, a benchmark would help to evaluate the actually achieved elasticity and compare different approaches against each other.

Previous works [2, 12, 3] on benchmarking cloud offerings neglect certain aspects of elasticity. For example, they measure the scale-up behavior without considering the scale-down behavior. Additionally, elasticity is not measured as a distinct attribute, but it is often mixed up with efficiency. Furthermore, the employed load profiles for benchmarking do not reflect a realistic variability of the load intensity over time. The approach proposed in [8] takes a business perspective on the evaluation of elasticity. They analyze the financial impact of different elastic cloud solutions for a set of load scenarios. This is a valid approach for a cloud customer who must make a cost-based decision between alternative cloud offerings. However, this approach mixes up the evaluation of the business model and the evaluation of the technical aspects of elasticity.

It is our goal to analyze elasticity in the "Infrastructure-as-a-Service" (IaaS) context. To stress that scaling in the IaaS context is realized by scaling of the underlying infrastructure resource allocations, the term resource elasticity will be used throughout the paper. We propose a benchmarking methodology and metrics which are capable of quantifying resource elasticity of IaaS cloud systems. In the envisioned approach, the main idea for evaluating resource elasticity is based on comparing a changing resource demand over time with the actual allocation (supply) of resources. The varying resource demand is induced by resource specific loads. To allow the usage of load profiles with realistic variation in load intensity, our approach incorporates the modeling of characteristic load variations using the LIMBO toolkit [16]. Different hardware performance of the compared systems may affect their scaling behavior and can hamper an objective elasticity evaluation. This issue is tackled by applying strict assumptions about the scaling behavior, which are calibrated step-wise to match the system behavior. The results of this system scalability analysis are then processed to adjust the load intensity variations in a way that all evaluated systems are stressed in a comparable manner - even if their underlying resources exhibit different performance. The quantification of the achieved resource elasticity is done using metrics that capture the core aspects of elasticity [7]. We plan to refine the proposed metrics in an iterative manner. The benchmarking approach is planned to be evaluated on a private cloud by using a configurable elasticity mechanism for which the influences of different configurations can be observed. In a first step, we focus on processing resources (CPUs) as resource type and on scaling-out/in (adding/removing nodes).

The remainder of this paper is structured as follows: Sec. 3 gives several foundations for the context of resource elasticity benchmarking. This includes the definition and discrimination of important terms, information about the variety of existing elasticity mechanisms, and explanations of requirements for elasticity targeted benchmarks. Sec. 2 summarizes the state of the art in the field of elasticity evaluation by reviewing selected related works and their limitations. In Sec. 4, we describe our envisioned approach in greater detail. After a coarse grained overview of the proposed benchmark, the limitations of the approach and possible future enhancements are discussed. This discussion is followed by sections that describe the three main aspects of a proposed benchmarking methodology: the modeling of realistic load intensity variations as benchmark input, the calibration of the benchmark to take into account different hardware performance, and definition of suitable metrics to quantify resource elasticity.

2. RELATED WORK

This section analyzes existing approaches to elasticity evaluation. Previous approaches to evaluate elasticity either analyze elasticity only to a limited extent or take the perspective of a cloud customer who is interested in the financial implications of different elasticity behaviors.

Specialized approaches: A simple approach to measure some aspects of elasticity was proposed by Binning et al. [2]. In their paper, they define initial ideas for measures that capture different aspects of cloud systems. One aspect is the adaptability of systems to peak loads. This is one aspect of elasticity although the authors do not use the term elasticity while describing the metric. Binning et al. suggest to use the ratio between issued requests and requests that are answered within a given response time as a measure. The latter is a very rough measure. It leaves open, if the peak was big enough to enforce an adaptation at all or if the peak was so big that even at the upper scaling bound the system is unable to handle the requests within the required response time. Another aspect of elasticity is measured by Li et al. [12]. They use a metric called *Scaling Latency* for which they manually request new virtual instances and measure the time between each request and the time the requested instance is made available. This reflects one aspect that influences the elasticity of a system. Nevertheless, the real scaling behavior strongly depends on the adaptation mechanism that triggers the creation or removal of instances. Dory et al. proposed an approach to measure the elasticity for cloud databases [3]. They measure elasticity by analyzing the distribution of response time during a scale-out process. This approach may be valid for analyzing the scale-out behavior, but it does not consider the quality of de-provisioning.

Business perspective approaches: In [8], Islam et al. present a concept that allows cloud customers to evaluate the financial implications of choosing different elastic cloud providers. In contrast to previous works, this paper analyzes over- and under-provisioning. It takes into account the fact that the amount of allocated resources is not necessarily equal to the amount of resources the customer is charged for. The load profiles used for the evaluation are a set of simple mathematical functions including linear functions, exponential functions, sines containing plateaus of different lengths. These load profiles are one step towards a realistic variation of load intensity, but still the use of a workload model that captures the expected variability of load intensity in a more accurate manner is missing. Another issue that is problematic is that customers who want to apply this benchmark, need to specify an appropriate penalty for the violation of service level objectives (SLOs). The specification of this penalty can have a high impact on the results of the evaluation. In this case, the quantification of elasticity aspects is mixed up with the business model.

3. FOUNDATIONS

This section provides some relevant background for elasticity benchmarking. Firstly, we roughly describe the architecture of elastic cloud systems. This description is followed by a section explaining terms commonly (miss-)used in the cloud context. After this differentiation, we discuss resource elasticity in more detail.

3.1 Elastic Cloud System Architecture

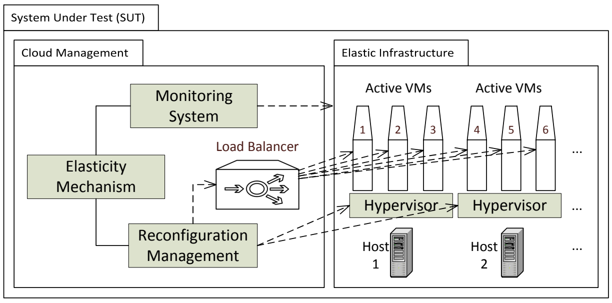


Figure 1: Blueprint architecture of elastic systems

Figure 1 shows a blueprint architecture of a simple elastic cloud system. Elastic cloud systems typically consist of two parts: A scalable infrastructure and a management system. As a basic service, cloud providers offer infrastructure to their customers in the form of virtual machines with network access and storage. The hypervisor acts as virtualization layer that allows a shared usage of the underlying physical hardware. When customers need more resources they have - depending on the provider - at least one of two options. They can either ask the provider to assign more resources to their virtual machine (scale-up) or request additional virtual machine (VM) instances (scale-out). Sometimes even a combination of both methods is possible. The first option is limited by the amount of resources the underlying hardware can provide. As soon as multiple instances are available, the incoming load must be distributed. This task is performed by a *load balancer*. The load-balancer forwards incoming requests according to a configured scheme, e.g., round robin, to the VM instances. The scalable infrastructure is managed by a *cloud management server*. It offers different services via modules. The *reconfiguration management* module supports the creation of new virtual machines and allows starting and stopping them. A *monitoring* module allows the collection of monitoring data about the virtual machines and about the underlying physical infrastructure. Often, the cloud management server also offers an *elasticity mechanism*. This mechanism uses monitoring data to evaluate if it is necessary to scale the resources and triggers reconfigurations of the elastic infrastructure accordingly. It also reconfigures the load-balancer if this is required due to a reconfiguration of the elastic system. Thus, the system adapts itself according to the demand and the customer does not need to reconfigure the system himself every time his demand changes. The realized elasticity depends on internals such as provisioning time but also on the used elasticity mechanism. Therefore, the *system under test* (SUT) for the proposed benchmark includes the scalable infrastructure, the load-balancer, and the cloud management system.

3.2 Terms and Differentiation

In the context of cloud computing, the terms efficiency, scalability and elasticity are commonly used without a clear distinction by referring to a precise definition. Although these terms are related to each other, they describe different properties. This section explains the meaning of each property in the context of cloud computing and the relations between them.

3.2.1 Efficiency

The Oxford Dictionary defines efficiency for the context of systems and machines as “achieving maximum productivity with minimum wasted effort or expense”. The way productivity and wasted effort are measured, strongly depends on the context. For computing systems, the term efficiency is tightly coupled with performance and can be split into cost, energy and resource efficiency:

Cost efficiency describes to what degree a system is able to achieve maximum productivity with minimum costs.

Energy efficiency describes to what degree a system is able to achieve maximum productivity with minimum energy consumption.

Resource efficiency either describes to what degree a system is able to achieve maximum productivity with minimal use of resources (system property), or describes the efficiency of an underlying resource unit (resource property).

For efficiency measurements, black box approaches are commonly applied.

3.2.2 Scalability

Scalability describes the degree to which a subject is able to maintain application specific quality criteria when it is applied to large situations. Although the term is frequently used, statements about scalability often lead to only a vague impression about the analyzed subject [4]. Many authors have tried to overcome this issue by proposing their own definitions or systematic ways to analyze scalability. The most important insights that are shared by several authors are summarized in the following paragraphs.

Scalability is fulfilled within a range according to a specific quality. Therefore, statements like “The system is scalable” do not provide much insight. Every system is scalable to some extent. Discriminating is based on the range within and the quality to which a system is scalable. Whereas the range is typically specified by an upper scaling bound, the quality usually describes the growth of a measured quality criteria. Possible qualities include linear or exponential growth, for example.

Scalability refers to *input variables* that are scaled. Scalability describes how the subject reacts when one or more input variables, sometimes referred to as attributes [15] or independent variables [4], are varied. Examples for such *input variables* are problem size, number of concurrent users or number of requests per second.

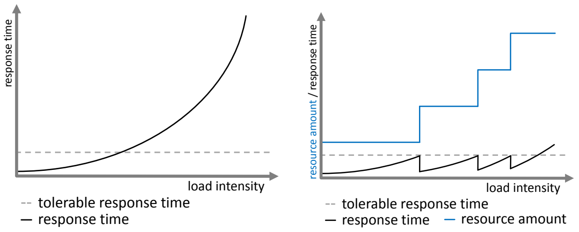
Scalability is measured by evaluating at least one quality criteria. To measure how the subject reacts, one or more quality criteria have to be observed while input variables are varied. These quality criteria are sometimes referred to as performance measures [15] or dependent variables [4]. Examples for quality criteria are memory consumption, I/O device usage, or response time.

With the help of the terms *input variable* and *quality criteria*, scalability in the cloud context can be described more precisely than commonly practiced. Typically the *input variable* for scalability analysis of cloud systems is the load intensity. It describes how much work a system has to handle in a given time span. Load intensity can be varied either by different work unit sizes or by varying the arrival rate of work units. There are two kinds of *quality criteria* for cloud systems:

Service levels: A service level can be described by measures like response time or abort rate. Cloud customers usually specify service level objectives (SLOs) which define the minimal acceptable service level for their application. Service levels are normally specified with the help of probability distributions over the measures.

Resource amounts: Resources are required means to conduct certain types of work. The amount of consumed resources can be measured for different resources and at different abstraction levels. Possible resource types are CPU cores or memory as physical resources, as well as virtual server instances, threads or locks, as software resources. Different abstraction levels cater for different granularity. For processing resources for example the resource amount can be measured by the number of used CPU cycles, physical CPUs, or virtual machines. The latter one is a special case as a virtual machine is a container resource, that contains several other resources.

Cloud customers typically want to offer their end users a constant service level that is independent of the *input variable* load intensity. Thus, *quality criteria* that are defined in SLOs should always be satisfied. This means the used resource amount characterizes the scaling behavior, as it has to increase when the load intensity increases. To emphasize that the scaling behavior of a cloud system is based on scaling of underlying resources, the term *resource scaling* will be used throughout this paper when referring to such systems.



(a) No resource scaling (b) Resource scaling

Figure 2: Resource scaling allows cloud systems to comply with predefined service levels as the load intensity increases

Figure 2 illustrates the difference between a system that uses resource scaling and one that does not. Here, a maximal tolerable response time is defined as service level. However other measures that define a service level are possible, too. In case the amount of resources for a system is fixed, the system response time will increase when the load intensity increases. As soon as the response time exceeds the predefined threshold, the system is not usable anymore. The scalability of this system with respect to response time is therefore very limited. In contrast, a system whose underlying resources can be scaled is able to comply with the maximum tolerable response time even for a higher load intensity. The scalability with respect to response time of this system is higher compared to the system without resource scaling. Still, the scalability is limited - as the maximum amount of underlying resources is limited. Note that the exponential increase of the response time is just exemplary. Other growth characteristics are also possible. Moreover other measures that measure a service level could be used in place of response time, too.

It's important to understand, that scalability does not contain any temporal aspect. In the context of cloud computing, scalability does not make any assumption about when the resources are scaled. Scalability just describes how much additional resources a system needs when the load increases to be able to offer a constant service level. Thus, scalability does not provide any information about the system's ability to scale resources on demand in a fast and accurate manner, it even does not make any assumptions about the existence of an automated scaling mechanism.

Resource scaling can be achieved in two different ways, that are often referred to as scaling dimensions:

Vertical scaling or scaling up/down refers to varying the amount of resources by adding/ removing resources to an existing resource node.

Horizontal scaling or scaling out/in refers to varying the amount of resources by adding/removing nodes to a cluster.

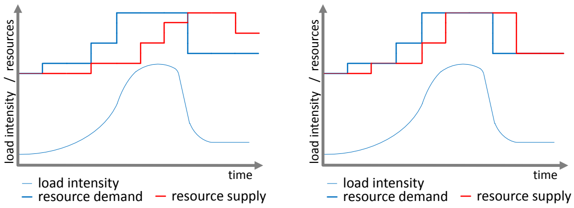
Migration is mentioned in [6] as a third scaling method. Migration describes the transference of a virtual machine from one physical location to another for global infrastructure or locality optimisation. Since the number of assigned resources typically changes, whereas the number of virtual instances does not change, migration can be treated as a special case of vertical scaling.

3.2.3 Resource Elasticity

Elasticity is known in physics and likewise in economics. In physics, elasticity is a material property that describes to which degree a material returns to its original state after being deformed. In economics, elasticity describes the responsiveness of a dependent variable to one or more other variables. On a high level of abstraction one could argue elasticity captures how a subject reacts to changes that occur in its environment.

For the context of cloud computing elasticity was previously discussed in [7]. Here, we build upon this work and further refine it. While scalability - in the cloud context - describes the degree to which a system is able to adapt to a varying load intensity by using a scaled resource amount, elasticity reflects the quality of the *adaptation process* in relation to load intensity variations over time. Thus, elasticity adds a temporal component to scalability. As elasticity describes properties of an adaptation process, elasticity requires the existence of a mechanism that controls the adaptation.

The impacts of different resource elasticity mechanism in cloud systems is illustrated by a simple example.



(a) System A (b) System B

Figure 3: System A and B are equal except for their elasticity mechanisms. The elasticity mechanism of system B is superior compared to system A since it is able to match the demand closer.

Figure 3 shows the behaviour of two systems that are equal except for their elasticity mechanisms. In particular, their underlying resources have the same efficiency and the scalability of both systems is equal as well. Thus, for an arbitrary load intensity profile, both systems require the same amount of resources to comply with predefined SLOs. In this example, the second system exhibits a better elasticity. The red curve - resource supply - matches the blue curve - resource demand - closer when comparing System A to System B. System A's adaptation process reacts faster and more precisely to changes in load intensity than the one of System B. To compare the elasticity of both systems in a quantitative manner metrics are needed. A set of initial metrics is discussed in Section 4.3.

Comparing elasticity becomes more complex, when the system's underlying resources have different efficiency or performance characteristics and the degree to which the considered systems scale may differ.

Definition.

In [7], the following definition for resource elasticity has been proposed: "Elasticity is the degree to which a system is able to adapt to load 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."

Prerequisites.

Before evaluating elasticity, several prerequisites should be checked beforehand cf. [7].

Autonomic Scaling: Elasticity is the result of an adaptation process that scales resources according to the load intensity. Evaluation of elasticity therefore requires that this process is specified. The adaptation process is usually realized by an automated mechanism. However, the adaptation process may also contain manual steps. A notable aspect in the latter case is that the repeatability of measurements in that case may be limited.

Resource Type: Elastic systems scale system resources. The type of resources can be quite different. There are base resources like CPU, memory or disk storage, and there are container resources, which comprise multiple base resources and are very common in cloud systems. To avoid comparing apples to oranges when evaluating elasticity, systems should be compared that use the same resource types.

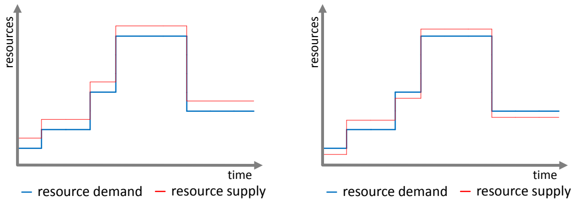
Resource Scaling Unit: The amount used resources can be measured in different units, e.g., CPU time slice shares, processors or virtual machines. If elasticity is analyzed by comparing resource demands to actual resource consumption, it is crucial to use the same units when comparing different systems.

Scaling Method: The different types of scaling methods are explained later in this section. Comparing elastic systems that are based on different scaling dimensions is desirable. Nevertheless, this should be done with care as the choice about the scaling method may have side effects such as different resource scaling units.

Scalability Bounds: The scalability of every system is limited. The scalability bounds depend on the maximum amount of available physical resources and on the service level constraints that are specified in SLOs. Elasticity comparisons should be performed within a scaling range that is supported by all compared systems.

Core Aspects.

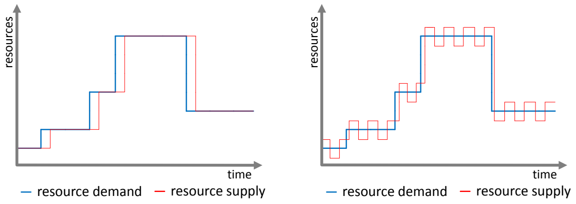
Accuracy: Fig. 4(a) shows a system that over-provisions at all times. This could be due to a very conservative adaptation process, aiming to never violate SLOs. Although System C reacts very fast, it does not match the demand as close as possible and should therefore be considered less elastic than an ideal elastic System (not illustrated, identical curves).



(a) System C (b) System D

Figure 4: Systems with imperfect adaptation accuracy.

Fig. 4(b) shows another System D that also always adapts at the exact points where the demand changes. But, in contrast to System C it over-provisions and under-provisions. Systems C and D have in common that they seem to react immediately when the demand changes. But although they react fast, both systems do not match the demand very precisely. Thus, *accuracy* can be seen as one core aspect of elasticity.



(a) System E (b) System F

Figure 5: Systems with imperfect adaptation timing.

Timing: To illustrate the timing aspect of elasticity, a possible supply curve could look like shown in Figure 5. Fig. 5(a) shows the behavior of a hypothetical System E that is able to match the resource demand, but with some delay. System E could be a system that needs some time to perform its adjustments after the resource demand changes. Similar, one can imagine a system that performs allocation activities in advance before the demand actually changes. Such a system proactively foresees changes. A further way how the supply curve can be modified is shown in Fig. 5(b). Whereas, the curve for available resources generally matches the curve for the resource demand, the available resources seem to be updated with - an unnecessary - high frequency. It can be argued that Systems E and F have a timing behavior that is not ideal. Therefore, the *timing* of the adaptation process can be seen as a second core aspect. It is valid to argue that System F not only has a bad *timing*, but also its *accuracy* is not optimal. Although *accuracy* and *timing* are not orthogonal dimensions, these core aspects help to describe and compare elasticity in a structured way. Metrics that capture the core aspects of elasticity are proposed in Section 4.3.

Mechanisms.

This section gives a short overview of existing elasticity mechanisms and shows how they can be classified according to a taxonomy. The broad variety of different elasticity mechanisms warrants the need for a benchmark to evaluate the quality of different mechanisms.

A cloud system with resource elasticity is a self-adaptive system. Resources are allocated according to a changing demand. In [13], Salehie and Tahvildari present a taxanomy of self-adaptive systems. Although they target self-adaptive systems on a high abstraction level, most variation points are applicable to systems with elastic resource scaling. Galante and de Bona present in their survey [6] a comparable taxonomy targeted at resource elasticity. Without going into too much detail or explicitly picking advantages of an individual mechanism, some relevant aspects that appear in at least one of the taxonomies are highlighted in the following. Hereby, aspects limiting the comparability as well as aspects that motivate the need for a benchmark are emphasized. The target *abstraction layer* (e.g. IaaS, PaaS) for elasticity mechanisms may be different. This is one reason why the *unit* of the scaled resources or even the *type* of considered resources can be different. Elasticity mechanisms can make use of different *scaling methods* [6] to adjust the amount of available resources. As outlined earlier in this section, resource *type*, *unit* and method can limit the comparability of elasticity. Elasticity mechanisms can be *reactive* or *proactive/predictive*. Reactive mechanisms start the adaptation process as soon as they detect a changed demand. Due to the time needed for the adaptation itself, the available resources match the demand only after some delay. Predictive systems extend reactive ones. They try to foresee demand changes in order to provision the correct amount of resources in time. By intuition, mechanisms that contain predictive elements should perform better than mechanisms that are purely reactive. Apart from these temporal characteristic of elasticity mechanisms many alternative exist for different methodical *realization issues* cf. [13, p.13ff]. All of them have their own advantages and disadvantages. A benchmark can reveal their impact on elasticity.

3.3 Elasticity vs. Efficiency and Scalability

Efficiency is a term that can be applied to both, a part of a system, e.g., a single resource, or to an entire system. In any case it reflects the ability of the subject to process a certain amount of work with smallest possible effort.

Scalability describes the degree to which a system is able to adapt to a varying load intensity by scaling resource allocations to maintain a predefined service level. Improving scalability normally means reducing scaling overhead and therefore leads to improved efficiency. Still, high efficiency is not necessarily due to good scalability. There is also no direct implication for the opposite direction.

Elasticity reflects the sensitivity of a system's scaling process in relation to load intensity variations over time. Thus, scalability is a prerequisite for elasticity. Normally, better elasticity results in higher efficiency, as high elasticity implies appropriate resource allocation and usage. The other way around this implication is not given. No direct implications exist between scalability and elasticity or vice versa.

4. BENCHMARKING APPROACH

In [5], the author mentions relevance, repeatability, fairness, verifiability and economic efficiency as main characteristics of a benchmark in general, and highlights further challenges for cloud system benchmarks like locality. The main requirements for a cloud benchmark are grouped into general requirements, implementation requirements, and workload requirements. To fulfill these requirements, before we execute a load profile, the benchmark performs a scalability analysis in a calibration step. The scalability analysis and load profile calibration are explained in Section 4.2. The calibrated load profile is then executed with the help of the workload generator. Finally, metrics that evaluate the elasticity are calculated. Section 4.3 explains ideas for metrics in greater detail. The proposed approach is not a black box approach and therefore access to the cloud management server is required for the benchmark execution.

The activity diagram shown in Figure 6 depicts the benchmark workflow. Since the scalability analysis requires a manual allocation of resources, the elasticity mechanism has to be switched off before conducting the scalability analysis. After the latter is completed, the elasticity mechanism is turned on again. The results of the scalability analysis are used to calibrate the load profile. After the calibration, the SUT is penetrated with the adjusted load profile. At the same time, the resource allocation (supply) is monitored. Afterwards, the elasticity is evaluated by applying metrics.

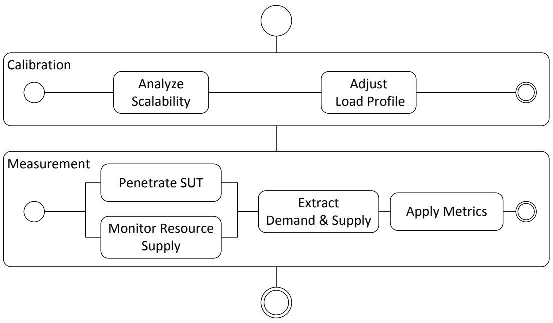


Figure 6: Diagram of benchmark workflow.

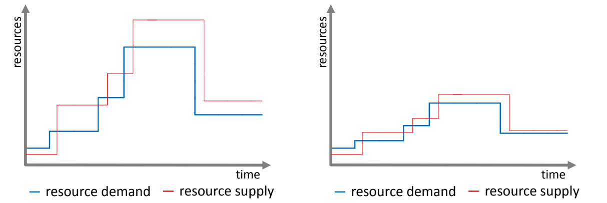
4.1 Load Intensity Variation

A good benchmark uses realistic load profiles to stress the SUT in a representative manner. Workloads are commonly modeled either as closed workloads or as open workloads. Whereas in closed workloads new job arrivals are triggered by job completions, arrivals in open workloads are independent of job completions. The elastic behavior of a system is usually triggered by a change in load intensity. Hence, for elasticity benchmarking it is important that the variability of the load intensity is modeled realistically. As this can be achieved with an open workload model, the developed benchmark will use an open workload model. Load profiles typically consist of a mixture of several patterns. These patterns can model linear trends, bursts that are characterized by an exponential increase, or patterns which model the general variability over a day, a week or a year. V. Kistowski et al. present in [17] a meta-model that allows the modeling of variable load intensity behaviors. They offer the LIMBO toolkit described in [16] to facilitate the creation of new load profiles that are either similar to existing load traces or contain different desired properties like a seasonal pattern and additional bursts. The usage of this toolkit and the underlying meta-model allows the creation of realistic load variations that are still configurable. Thus, the load profiles used for benchmarking can be adapted with low effort to suit the targeted domain.

4.2 Benchmark Calibration

The resource demand of a system for a fixed load intensity depends on two factors: The efficiency of a single underlying resource unit and the overhead caused by combining multiple resources units. Both aspects can vary from system to system and are related to distinct properties namely efficiency and scalability. Elasticity is a different property and should be measured separately. One way to do so relies on analyzing the scaling capabilities of a system before evaluating elasticity. After such an analysis, it is known how many resources are needed to satisfy a given static resource demand. The resource demand can then be expressed as a function of the load intensity: $resourceDemand = f(intensity)$. With the help of this function, which is specific for every system, the amount of resources actually used.

Figure 7 shows how two Systems G and H react when they



(a) System G (less efficient resources)

(b) System H (more efficient resources)

Figure 7: Two systems that are exposed to equal load profiles. The induced resource demand on System H is lower than on System G due to a higher efficiency of the underlying resources.

are exposed to the same load profile. Since the resources of System H are more efficient than those of System G, system H can handle the load with less resources than System G. For both systems there exist some points in time where the systems over-provision and other points in time where they under-provision. Comparing their elasticity is difficult as the resource demand of both systems is different. The idea of our approach is to adjust a given load profile in a way that the induced resource demand changes occur equal in amount and experiment time on all systems. By doing so, the possibly different efficiency of the underlying resources units as well as different scaling behaviors are compensated. With an equal resource demand it is now easier to compare the quality of the adaptation process and thus the evaluation of elasticity can be done in a fair way. While the process of adjusting load profiles in a way that the same resource demand is induced on every compared system is currently under development and will be specified more precisely in our future work, the next paragraph presents a description of a preliminary version of the preceding scalability analysis.

Scalability Analysis: The scalability capabilities of a system are analyzed using an iterative process. The process assumes a predefined SLO that is needed for the evaluation if the system is capable to serve requests at a static load intensity by using just a fixed limited amount of resources. The SLO has to be chosen according to the targeted domain. Within an optional evaluation step, the impact of different SLOs will be analyzed. Since the scalability analysis should not evaluate any elastic behavior, the elasticity mechanism of the system may not be turned on during the analysis. The scaling is controlled manually. The analysis is started with one resource instance. Of course the load balancer has to be configured in a way that all requests are forwarded to this instance. This instance is now exposed to a constant small load. The load intensity should be that small that the system can process the load without violating the SLO. Now, the load intensity is increased in small steps. After each adjustment the SLO compliance is checked. As soon as the system cannot process the load anymore without violating the SLO, the system is reconfigured. One additional resource unit is added and the load balancer is configured accordingly. After the reconfiguration the system should be able to comply with the SLO again. The load intensity is now increased again stepwise. This process is repeated until either there are no additional resources which could be added to the system, or even after adding a new resource the SLOs are still violated, e.g., due to an increased scaling overhead. In both cases the upper scaling bound is reached and the scalability analysis is finished.

As an alternative approach for finding the maximal load intensity that a system can withstand using a given amount of resources, we plan to implement binary search instead of a stepwise increase of intensity. Possible binary search based load picking algorithms have been presented by Shivam et al. [14].

4.3 Metrics

This section discusses elasticity metrics, that we plan to evaluate and further refine in our future work. The following metrics that capture different aspects of elasticity have been proposed by Herbst et al [7].

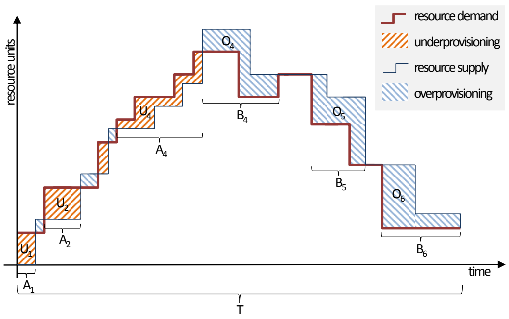


Figure 8: Basic elasticity measures. Source: [7]

Figure 8 illustrates how several metrics can be derived by using functions of resource demand and resource supply over time. Basic measures are defined by Herbst et al. as follows:

- $\overline{A}$ is the average time to switch from an under-provisioned state to an optimal or over-provisioned state and corresponds to the average speed of scaling up.
- $\sum A$ is the accumulated time in under-provisioned state.
- $\overline{U}$ is the average amount of under-provisioned resources during an under-provisioned period.
- $\sum U$ is the accumulated amount in under-provisioned state.
- $\overline{B}, \sum B, \overline{O}, \sum O$ are defined similarly for over-provisioned states.
- T total duration of evaluation period

These base measures are used to define the following metrics:

- speed (for scaling up/out): $\sum A$
- precision (for over-provisioning periods): $P_d = \frac{\sum O}{T}$
- elasticity (for scaling up/out): $E_u = \frac{1}{\overline{A} \times \overline{U}}$

Metrics for scaling down/in are defined accordingly. The *precision* metric captures the elasticity aspect called *accuracy* in Sec. 3.2.3. The metric *speed* can be one way to reflect the *timing* behavior which is the second core aspect of elasticity. This metric cannot be derived when a mechanism either constantly over-provisions or constantly under-provisions. Additionally, a small value for the speed metric is not necessarily an indicator for good elasticity. A system that provisions and de-provisions resources with an unnecessary high frequency, e.g. System F as shown in Figure 5(b), may have a small value for the speed metric although its elasticity is suboptimal.

One alternative way of measuring the timing behavior is based on comparing the number of scale up (scale down) events D_u (D_d) of the demand with the number of scale up (scale down) events A_u (A_d) for the allocated resources.

The absolute difference of both should be as small as possible. Too many scale events for the resource supply (see Figure 5(b)) as well as too few few scale events are both indicators for a bad timing behavior. Thus, the following two metrics can be used to characterize the timing behavior:

$$scale\_up\_timing = \frac{|D_u - A_u|}{D_u} \tag{1}$$

$$scale\_down\_timing = \frac{|D_d - A_d|}{D_d} \tag{2}$$

The metrics explained above capture certain aspects of elasticity separately. An alternative approach is to use metrics that characterize elasticity in a more global manner. One way to do so bases on comparing the curves for resource demand and supply by using the dynamic time warping (DTW) [10] distance. This approach was demonstrated as a way to measure the elasticity of thread pools [11]. The use of this metric or other metrics that capture the similarity between two curves present an alternative approach for quantifying elasticity.

5. CONCLUSIONS

We motivate the need for a benchmark capable of evaluating the resource elasticity of cloud systems. Although different approaches for evaluating elasticity already exist, they are either immature or have a different non-technical perspective. As a basis for our new benchmark under development, we firstly differentiate the terms efficiency, scalability and elasticity before we analyze elasticity in detail. The most important existing approaches are presented and analyzed with respect to the perspective they take on elasticity and the issues that do not satisfy the requirements for benchmarking. We outline the envisioned benchmarking methodology and discuss its limitations. The approach is based on inducing the same resource demand on all compared systems by adjusting a realistic load profile according to the scaling characteristics of the analyzed systems. With the help of several proposed metrics, the induced demand is compared to the actual resource supply and thus the elasticity of the analyzed systems is evaluated.

6. 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*, 53(4):50–58, Apr. 2010.

[2] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing. How is the weather tomorrow?: towards a benchmark for the cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*, DBTest '09, pages 9:1–9:6, New York, NY, USA, 2009. ACM.

[3] T. Dory, B. Mejías, P. V. Roy, and N.-L. Tran. Measuring Elasticity for Cloud Databases. In *Proceedings of the The Second International Conference on Cloud Computing, GRIDs, and Virtualization*, 2011.

[4] L. Duboc, D. S. Rosenblum, and T. Wicks. A framework for modelling and analysis of software systems scalability. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 949–952, New York, NY, USA, 2006. ACM.

[5] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun. Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In R. Nambiar and M. Poess, editors, *Selected Topics in Performance Evaluation and Benchmarking*, volume 7755 of *Lecture Notes in Computer Science*, pages 173–188. Springer Berlin Heidelberg, 2012.

[6] G. Galante and L. C. E. d. Bona. A Survey on Cloud Computing Elasticity. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 263–270, Washington, DC, USA, 2012. IEEE Computer Society.

[7] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in Cloud Computing: What it is, and What it is Not (Short Paper). In *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013)*. USENIX, June 2013.

[8] 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*, ICPE '12, pages 85–96, New York, NY, USA, 2012. ACM.

[9] B. Jennings and R. Stadler. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, pages 1–53, 2014.

[10] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive dataset. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '99, pages 1–11, London, UK, 1999.

[11] M. Kuperberg, N. R. Herbst, J. G. von Kistowski, and R. Reussner. Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms. Technical report, Karlsruhe Institute of Technology (KIT), 2011.

[12] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 1–14, New York, NY, USA, 2010. ACM.

[13] M. Salehie and L. Tahvildari. Self-adaptive Software: Landscape and Research Challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[14] P. Shivam, V. Marupadi, J. Chase, T. Subramaniam, and S. Babu. Cutting corners: Workbench automation for server benchmarking. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 241–254, Berkeley, CA, USA, 2008. USENIX Association.

[15] M. van Steen, S. Van der Zijden, and H. J. Sips. Software engineering for the scalable distributed applications. In *Computer Software and Applications Conference, 1998. COMPSAC '98. Proceedings.*, pages 285–292, 1998.

[16] J. G. von Kistowski, N. R. Herbst, and S. Kounev. LIMBO: A Tool For Modeling Variable Load Intensities (Demo Paper). In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE 2014)*. ACM, March 2014.

[17] J. G. von Kistowski, N. R. Herbst, and S. Kounev. Modeling Variations in Load Intensity over Time. In *Proceedings of the 3rd International Workshop on Large-Scale Testing (LT 2014), co-located with the 5th ACM/SPEC International Conference on Performance Engineering (ICPE 2014)*. ACM, March 2014.