Cloud-based Software Services Delivery from the Perspective of Scalability

Amro Al-Said Ahmad and Peter Andras

School of Computing and Mathematics, Keele University, Newcastle-under-Lyme, UK

{a.m.k.al-said.ahmad, p.andras}@keele.ac.uk

Corresponding author: Amro Al-Said Ahmad

Mailing address: School of Computing and Mathematics, University of Keele, Newcastle-under-Lyme, ST5 5BG, UK.

Tel.: +44 (0)1782 733593

E-mail: a.m.k.al-said.ahmad@keele.ac.uk

Cloud-based Software Services Delivery from the Perspective of Scalability

Measuring and testing the scalability and performance of cloud-based software services is critical for the delivery of such services, and the development of cloud computing. There are three interconnected Cloud-based software services' performance aspects; both elasticity and efficiency are depending on the delivery of a sufficient level of scalability performance. In this work, we focused on testing and measuring the cloud-based software services scalability from a technical perspective. This paper uses technical scalability metrics that address both volume and quality scaling, inspired by earlier technical metrics of elasticity. We demonstrate the application of the metrics using a practical example and three demand scenarios and discuss the importance of these metrics. We show how our technical scalability metrics integrated into an earlier utility-oriented metric of scalability, in order to enable the scalability analysis from technical and production-driven perspectives.

Keywords: Measurement, Performance, Testing, Scalability, Software-as-a-Service (SaaS), Metrics.

1. Introduction

In any software system, scalability and performance assessments provide an important basis for future optimizations, and for developing new opportunities aimed to maximize scalability and performance [1]. The performance assessment and testing of cloud-based software services is critically important in order to support the Service Level Agreement (SLA) compliant quality of delivery of these services, especially in the context of rapidly expanding the quantity of service delivery [2]. There are three cloud-specific performance aspects that are key determinants of service quality delivery in the context of variable service demand: scalability, elasticity, and efficiency [3, 4].

Following [5] we adopt the definitions of these three performance aspects. Scalability is the ability of the cloud layer to increase the capacity of the software service delivery by expanding the quantity of the software service that is provided. Elasticity is

the level of autonomous adaptation provided by the cloud layer in response to variable demand for the software service. Efficiency is the measure of matching the quantity of software service available for delivery with the quantity of demand for the software service. These definitions focus on the technical side of cloud-based software services, however, we note that alternative, utility-oriented (i.e. economic cost/benefit focused), approaches are also used in the literature [6, 7].

Cloud-based applications should be scalable, and with auto-scaling and load-balancing features, such applications should be able to deal with sudden workload by adding more of the application instance(s). Furthermore, as cloud-based applications been offered as Software as a Services (SaaS), and the use of multi-tenancy architectures [8], emphasizes the need for scalability that supports the availability and productivity of the services and on-demand resources.

Recently a series of papers have been published addressing the area of measuring the elasticity of cloud-based provision of software services [9, 10]. There have been also works on the scalability of cloud-based software services from the utility perspective [6, 7, 9, 11]. However, relevant systematic reviews report only a very small number of works (mainly in the grey literature, e.g. project reports, MSc theses) which try to address the assessment of scalability of cloud-based software services from the technical perspective [5].

Measuring and testing scalability of cloud-based software services from a technical perspective is key for the assessment and testing of performance [1,12]. Both elasticity and efficiency performance depends on the delivery of a sufficient level of scalability performance. Understanding how components of the cloud-based software service system contribute to the scalability performance of the system helps in designing

appropriate test scenarios and identifying options for changes and upgrades that can improve the scalability performance of the system.

Utility oriented assessment of scalability [6] i.e. measuring the scalability from an economic and cost perspective, is insufficient for the above purpose since it measures scalability from a perspective that is abstract relative to the technical components and features of the system. Thus, it becomes very difficult and possibly even practically impossible to associate specific technical components and features with a specific impact on the utility-oriented scalability performance. This is due to the potential multiple impacts of such technical components and features on utility features of the system that get integrated into the utility-oriented scalability measurement of the system.

Here we follow ideas proposed in the context of measurements and metrics for cloud elasticity [13–15] to propose technical measurement and metrics for scalability of cloud-based software services. The work uses metrics [16] that address both volume and quality scaling for evaluating cloud-based software services scalability performance. The metrics can be useful in order to support effective measurement and testing of scalability performance of those services from a technical perspective. This work provides an extension to the previous work [17] by including an additional evaluation scenario, description of the related experiments and results, more details in the explanation of the results, and discussion of the new experimental results in relation with the proposed metrics. This work introduces a set of 80 new practical experiments to demonstrate the scalability metrics, consequently, the number of experiments Is 240 in total. Furthermore, we show how our technical scalability metrics integrate into an earlier utility-oriented metric of scalability [11] and calculate the values for each demand scenarios, in order to enable the scalability analysis from technical and production-driven perspectives. We demonstrate the application of the metrics to a concrete cloud-based software service

(OrangeHRM) run through the Amazon EC2 Cloud using three demand scenarios. We show how the metrics can be used to identify differences in the behaviour of the assessed system in the context of different usage scenarios.

The rest of the paper is structured as follows. First, we review briefly the relevant recent literature. Next, we present our approach to measure and quantify scalability of cloud-based software services and explain the metrics based on the measurement approach. Next, we present an application example using three different usage scenarios to demonstrate the measurement approach and metrics. Next, we discuss the implications and importance of the approach and metrics. Finally, the paper is closed by the conclusions section.

2. Related Work

A related review [18] on the provisioning of cloud resources and related research challenges, identify, among others predictable performance and scalable resource management as promising challenges. Gao et al. [19] reviewed testing in relation to cloud-based software services. They highlight scalability and performance testing as key research directions. Other similar relevant surveys [19, 20] focus primarily on cloud service elasticity. The systematic literature review by Lehrig et al. [5] provides very useful definitions of key cloud performance concepts such as capacity, scalability, elasticity, and efficiency, which we adopt in this paper.

There is a considerable number of recent papers that address the issue of measuring the elasticity of cloud-based services in technical terms [4, 9, 15, 21–25]. Herbst et al. [4] define a useful set of key concepts that allow technical measurement of cloud service elasticity (see Figure 1) such as the quantity and time extents for periods of time when the service provision is either below or above what is required by the service demand. They [4, 21] define as elasticity measures: the time shares and average time

lengths in under-provisioned and over-provisioned states; the amounts of excess (over-provisioned) and lacking (under-provisioned) resources per time unit; the averages of the excess and lacking resources; and the jitter, which is the number of resource adaptations during a given time period of provisioning the service. The up-elasticity metric is defined as the reciprocal value of the product of the average under-provisioned time length and average lack of resource. The down-elasticity is defined similarly. Further elaboration on these metrics is provided by [23], who introduced further components and ways of considering the above factors (e.g. scalability, functions of resource inaccuracy and reconfiguration time).

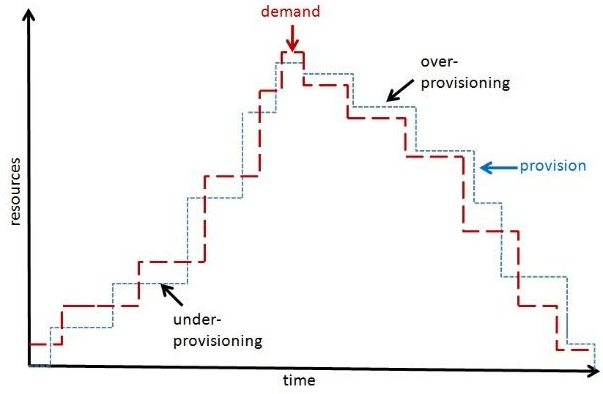


Figure 1. Key concepts for measuring elasticity.

In terms of measuring and quantifying scalability, we note the work of Hwang et al. [7, 11], which uses a utility-oriented definition of scalability. Their production-driven scalability measure includes the consideration of a quality-of-service measure and the cost of service, in addition to a more technically oriented performance metric [7, 11]. While this approach is likely to be useful from the perspective of utility, because of its reliance of multiple facets of the system (including cost measures), it is unlikely to be able to provide sufficiently specific and useful information in terms of contribution of system components to system scalability in a technical sense. Thus the usefulness of the

utility-oriented scalability metric is limited in the context of testing and technical improvement of the cloud-based provision of the software service.

Attempts to provide a more technically oriented measure or metric for cloud-based software service scalability are also limited. For example, Herbst et al. [4] provide a technical scalability metric, however, this is a rather elasticity driven metric (sum of over- and under-provisioned resources over the total length of time of service provision). Jamal et al. [27] describe practical measurements of throughput in systems with and without multiple virtual machines, without clearly formulating a specific measure or metric of scalability. Similarly, Jayasinghe et al. [13, 14] provides a technical scalability measure practically the system scalability in terms of throughput and CPU utilization of a set of virtual machine system settings but does not provide a generic metric or measure. Gao et al. [15] evaluate SaaS performance and scalability from the system capacity perspective, using the system load and capacity as measurements for scalability, a case study using a sample of Java-based program has been reported using Amazon EC2. Brataas et al. [28] offer two scalability metrics, one based on the relationship between the capacity of cloud software services and its use of cloud resources, while the other is the cost scalability metric function that replaces cloud resources with a cost. Another recent work [29] focuses on building a model to help to measure and compare different deployment configurations in terms of capacity, elasticity, and costs.

3. Scalability Performance Measurement

As noted in the Introduction, Scalability is the ability of the cloud-based system to increase the capacity of the software service delivery by expanding the quantity of the software service that is provided when such increase is required by increased demand for the service [5]. In this work, we are not concerned with the short-term flexible provision of the resources, which basically term elasticity of the service provision [22]. This work

focus is whether the system can expand the quantity of the service when this expansion is required by demand over a sustained period of service provision.

In principle, the increase of capacity usually happens by increasing the volume of service requests served by a single instance of the service provision software or by deploying multiple software instances, or by a combination of these two approaches. In general, we expect that if a service scales up ideally then the increase in demand for service should be matched by a proportional increase in the provision of the service such that the quality of the service does not change. Here the quality of the service may be seen for example in terms of average response time.

This ideal scaling behaviour of the system should be valid over a sufficiently long time scale, i.e. short-term mismatches between provision and demand, which are the subject of elasticity, are not relevant from the perspective of scalability. If the system does not scale according to the ideal manner, it recruits insufficient resources to deliver the increased volume of service without a change in the quality of the service. Generally, real systems are expected to operate below the level of the ideal scaling behaviour and the aim of measuring scalability is to quantify the extent to which the real system behaviour differs from the ideal behaviour.

To deliver the ideal scaling, we expect that the system increases the number of instances of the software proportionally with the increase in demand for software services, i.e. if the demand increases by 50% we would ideally expect the base number of software instances to increase by 50%. We expect also that the system maintains the quality of service in terms of maintaining the same average response time irrespective of the volume of service requests, i.e. an increase of 50% of demand we would ideally expect no increase in average response time. Formally, let us assume that D and D’ are two service demand volumes, D’ > D. Let I and I’ be the corresponding number of software

instances that are deployed to deliver the service, and let t_r and t'_r be the corresponding average response times. If the system scales ideally we expect that for any levels of service demand D and D'.

$$D' / D = I' / I \quad (1)$$

$$t_r = t'_r \quad (2)$$

Equation (1) expresses that the volume of software instances providing the service scales up with the demand for the service. Equation (2) expresses that the quality of service, in terms of average response time, remains unchanged for any level of service demand.

To measure the values of I and t_r the system must perform the delivery of the service over some sustained time, such that short-term variations, due to elastic response of the system, do not influence the system measurements. In practice this means that the number of software instances and the average response time should be calculated by averaging over a number of measurements during the execution of a demand scenario (e.g. every second), and a number of repeated applications of the same demand scenario, i.e. a pattern of demand presentation, which may include variation in the demand.

Demand scenarios may follow certain patterns expected to test the scalability of the system in specific ways. Three kinds of demand patterns that appear as natural and typical choices are; first, the steady increase followed by a steady decrease of the demand with a set level of the peak. The second scenario is a stepped increase and decrease, again with a set peak level of demand; with this scenario, we schedule to start with 10% of the demand size, then stepped increase 10% through time, while stepped down 10% through time. Finally, a varied stepped increase and decrease scenario, with a set peak level of demand. This scenario starts with 40% of the demand size, with a stepped increase of

20% through time until reaching the assigned demand size, while stepped decrease 10% through time. These three demand scenarios are shown in Figure 2. The aim to consider different demand patterns is to show how these impacts on the scalability performance of cloud-based software services. Any demand scenario has to be characterized by a summary measure of the demand level, which may be the peak level or the average or total demand level. This characteristic demand of a demand scenario is represented by D.

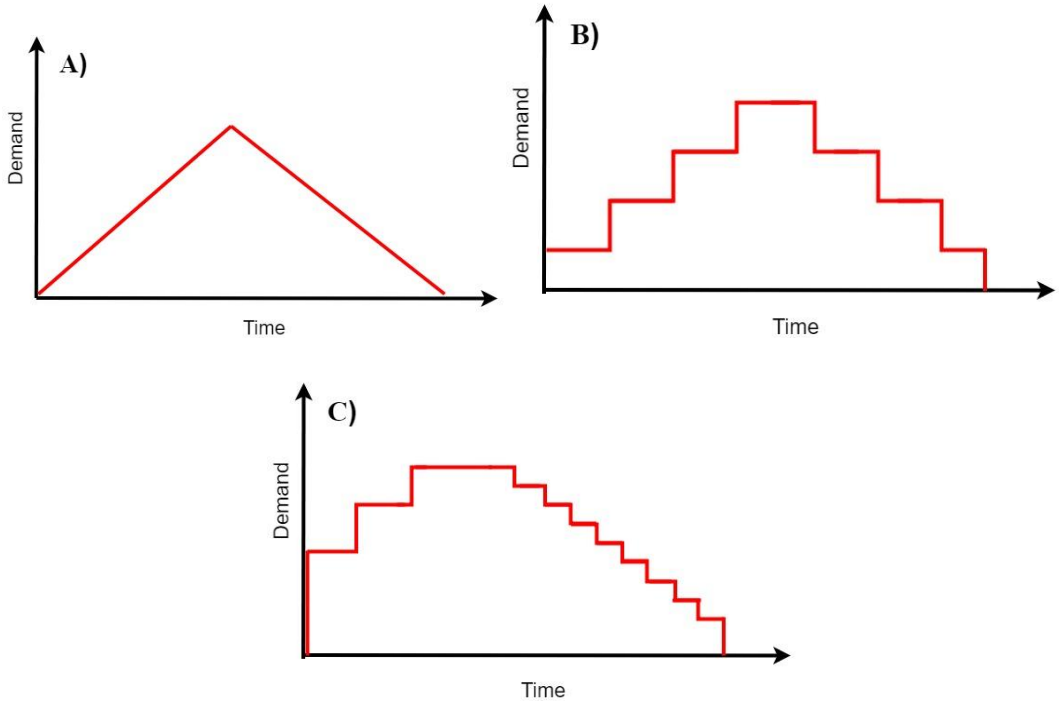


Figure 2. Demand scenarios: A) steady rise and fall of demand; B) stepped rise and fall of demand; C) varied stepped rise and fall of demand.

Naturally, real-world cloud systems are unlikely to deliver the ideal scaling behaviour. The difference between the ideal and the actual scaling behaviour of the system offers the possibility of defining technical scalability metrics for cloud-based software services. In terms of provision of software instances for the delivery of the services, the scaling is deficient if the number of instances is lower than the ideally expected number of software instances.

To quantify the level of deficiency we pick a demand scenario and start with a low level of demand D_0 and measure the corresponding volume of software instances I_0.

Then measuring the number of software instances I_k corresponding to a number (n) of demand levels D_k following the same demand scenario, we can calculate how close are the I_k values to the ideal I^*_k values ($I_k < I^*_k$). Following the ideal scalability assumption of equation (1) we get for the ideal I^*_k values:

$$I^*_k = (D_k / D_0) \cdot I_0 \tag{3}$$

Considering the ratio between the area defined by the (D_k, I_k) values, $k = 0,\ldots,n$, and the area defined by the (D_k, I^*_k) values we get a metric of service volume scalability of the system:

$$A^* = \sum_{k=1,\ldots,n} (D_k - D_{k-1}) \cdot (I^*_k + I^*_{k-1}) / 2 \tag{4}$$

$$A = \sum_{k=1,\ldots,n} (D_k - D_{k-1}) \cdot (I_k + I_{k-1}) / 2 \tag{5}$$

$$\eta_I = A / A^* \tag{6}$$

where A and A^* are the areas under the curves evaluated piecewise, calculated for actual and ideal I values and η_I is the volume scalability performance metric of the system. If η_I is close to 1 the system is close to ideal volume scalability, if it is close to 0, then the volume scalability of the system is much less than ideal.

Similarly, we can define the quality scalability of the system by measuring the service average response times t_k corresponding to the demand levels D_k. We approximating the ideal average response time as t_0, following the ideal assumption of equation (2). The quality scalability of the system is less than ideal if the average response times for increasing demand levels increase, i.e. $t_k > t_0$. By considering the ratio between the areas defined by the (D_k, t_k) values, $k = 0,\ldots,n$, and the area defined by the (D_k, t_0) values we get a ratio that defines a metric of service quality scalability for the system:

$$B^* = \sum_{k=1,\ldots,n} (D_k - D_{k-1}) \cdot t_0 = (D_n - D_0) \cdot t_0 \quad (8)$$

$$B = \sum_{k=1,\ldots,n} (D_k - D_{k-1}) \cdot (t_k + t_{k-1}) / 2 \quad (9)$$

$$\eta_t = B^* / B \quad (10)$$

where B and B^* are the areas under the curves evaluated piecewise, calculated for actual and ideal t values and η_t is the quality scalability performance metric of the system. If η_t close to 1 the system is close to ideal quality scalability if it is close to 0 the quality scalability of the system is much less than ideal.

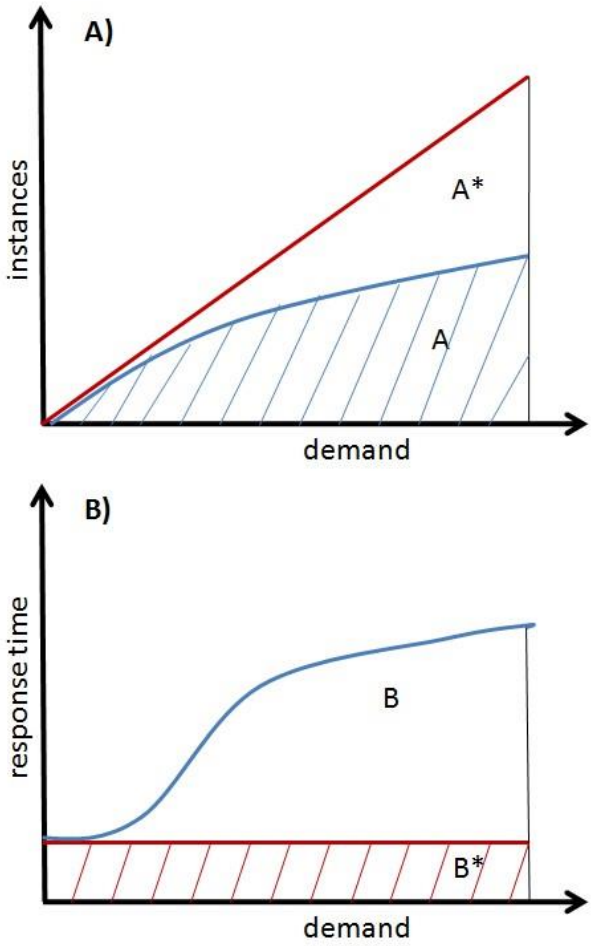


Figure 3. The calculation of the scalability performance metrics: A) the volume scalability metric is η_I, which is the ratio between the areas A and A* – see equation (6); B) the quality scalability metric is η_t, which is the ratio between the areas B* and B – see

equation (9). The red lines indicate the ideal scaling behaviour and the blue curves show the actual scaling behaviour.

The calculation of the two scalability performance metrics is illustrated in Figure 3. In Figure 3A, A^* is the area under the red line showing the ideal expectation about the scaling behaviour (see equation (1)) and A is the shaded area under the blue curve. The blue curve is under the ideal red line, indicating that the volume scaling is less efficient than the ideal scaling. In Figure 3B, B^* is the shaded area under the red line indicating the expected ideal behaviour (see equation (2)) and B is the area under the blue curve. The blue curve is above the ideal red line, indicating that the quality scaling is less than ideal. We chose nonlinear curves for the examples of actual scaling behaviour to indicate that the practical scaling of the system is likely to respond in a nonlinear manner to changing demand.

These scalability metrics allow the effective measurement of technical scalability of cloud-based software services. These metrics do not depend on other utility considerations (e.g. price of service, non-technical quality aspects), which makes them appropriate for testing the technical scalability of the system. This makes possible the use of these metrics in scalability tests that aim to identify parts of the system that have a significant impact on the technical scalability, and also the testing of the impact of any change made to the system on the technical scalability of the system.

Applying the scalability metrics to different demand patterns allows the testing and tuning of the system for particular usage scenarios and the understanding of how system performance can be expected to change as the pattern of demand varies. Such application of these metrics may highlight trade-offs between volume scaling and quality scaling of the system that characterize certain kinds of demand pattern variation (e.g. the impact of the transition from low-frequency peak demands to high-frequency peak

demands or to seasonal change of the demand). Understanding such trade-offs can help in tailoring the system to its expected or actual usage.

As important as measuring and testing scalability is, so is to collect the right measurements and to interpret those measurements using the right metrics. This thesis will develop a consistent interpretation of the fine-grained performance measurement data through the lenses of relevant scalability performance metrics. This interpretation enables a better understanding of the factors that influence performance metrics of the scalability of cloud-based systems and will help software engineers to fine-tune such systems to achieve better performance.

4. Application Example and Result

To demonstrate the applicability of the scalability metrics, we used the Amazon AWS cloud environment and the OrangeHRM[1]open source human resource software system as the cloud-based software service. To measure the scalability, the user demand scenarios were simulated using the Apache JMeter script[2] and run through Redline13[3] services after connecting our Amazon account to the service. To provide the scaling of the service we relied on the Auto-Scaling and Load-Balancer services provided by the Amazon AWS cloud.

An EC2 instance was set up and configured to host the targeted application through the Amazon EC2 management console. Both Auto-Scaling and Load-Balancer services were connected to the application instance, and the CloudWatch service to monitor the scaling performance and parameters was attached to the software services.

---

1 https://www.orangehrm.com/, AWS customers can subscribe to OrangeHRM as SaaS through AWS marketplace.

2 http://jmeter.apache.org/

3 https://www.redline13.com

The experimental data has been collected through both Redline13 and Amazon's CloudWatch services. In this study, the system average response time was measured as the average amount of time that the application takes to process an HTTP request after it has received one. The parameters of the Amazon EC2 virtual machines and Auto-scaling policies that have been used for the experiments are given in Table 1. The service requests consisted of an HTTP request to the main page of software by gaining login access using the following steps from the Apache JMeter:

- Path = /.
- Method = GET.
- Parameters = username, password and login button.

Table 1: Amazon AWS EC2 Virtual Machine Parameters

Virtual Machine Parameters			
Instance type: t2.micro			
vCPUs	RAM (GiB)	CPU Credits/hr	Storage (GB)
1	1.0	6	10
Auto Scaling Policies			
Add Instance	When 80% >= CPUUtilization < +infinity		
Remove Instance	When 30% <= CPUUtilization > -infinity		

To generate the workload demand scenarios using JMeter, *Thread Group* is used to generate the demand volumes for the first scenario. Then *jp@gc – Stepping Thread Group* is used to generate the demand volumes for the second scenario, and finally *jp@gc - Ultimate Thread Group* is used to generate the demand volumes of the third scenario. To ensure the repeatability of the test, RedLine13 services are used, which allows JMeter test scripts to be deployed easily inside our Amazon AWS domain and the tests to be repeated without the need to reset the test parameters, this allows efficient extraction of the data.

In this work, we used three demand scenarios. The first scenario follows the steady rise and fall of demand pattern shown in Figure 2A. The second scenario consists of a series of stepwise increases and falls in demand, conceptually similar to the demand pattern shown in Figure 2B. The third scenario consists of a varied series of stepwise increases and decreases in demand shown in Figure 2C.

Examples of the three kinds of experimental demand patterns (users running at runtime) are shown in Figure 4. We varied the volume of demand and experimented with four demand sizes: 100, 200, 400 and 800 service requests in total.

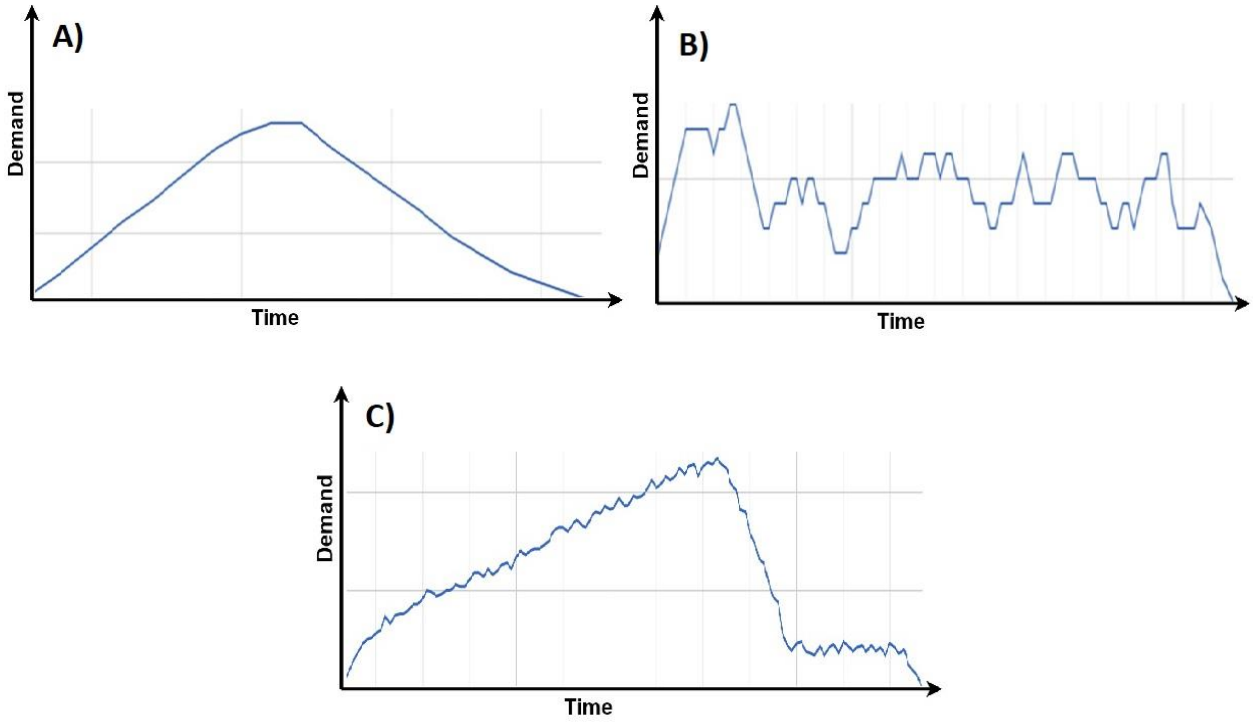
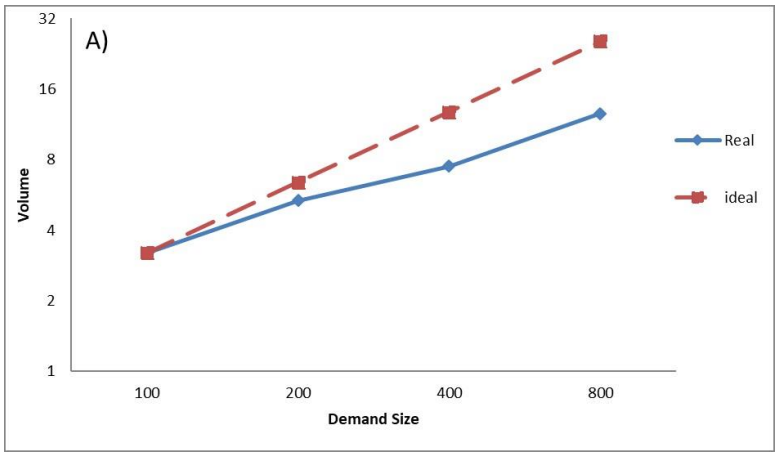
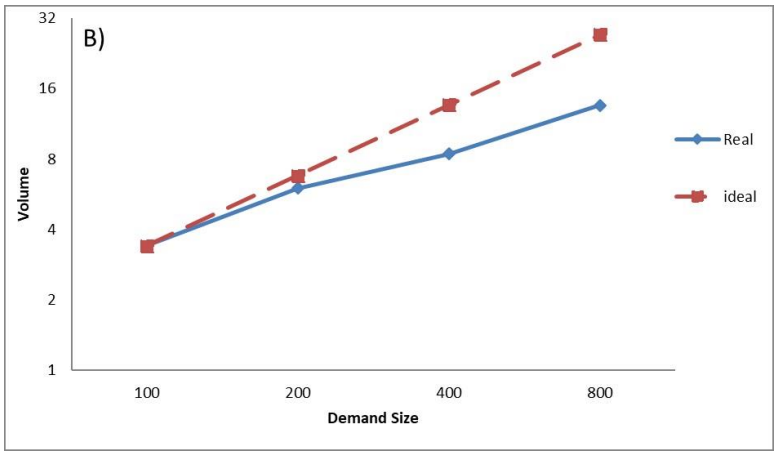


Figure 4. Typical experimental demand patterns: A) steady rise and fall of demand; B) series of step-wise increases and decreases of demand; C) varied stepped rise and fall of demand.

All experimental settings were repeated 20 times (i.e. demand pattern and demand volume combinations), in total 240 experimental were conducted. We calculated the average number of simultaneously active software instances and the average response time for all service requests for each experimental run. We also calculated the averages

and standard deviations of simultaneously active software instances and average response times over the 20 experimental runs. We note that the standard deviations are small relative to the averages over the 20 runs. The average number of software instances for the three scenarios and for the four demand levels are shown in Figure 5. The average response times for the three scenarios and four demand levels are shown in Figure 6.





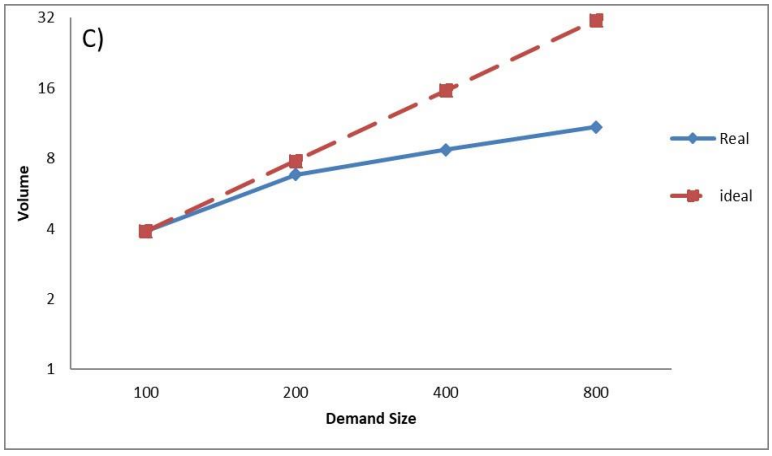


Figure 5. The average number of software instances: A) steady rise and fall of demand; B) series of step-wise increases and decreases of demand; C) varied stepped rise and fall of demand.

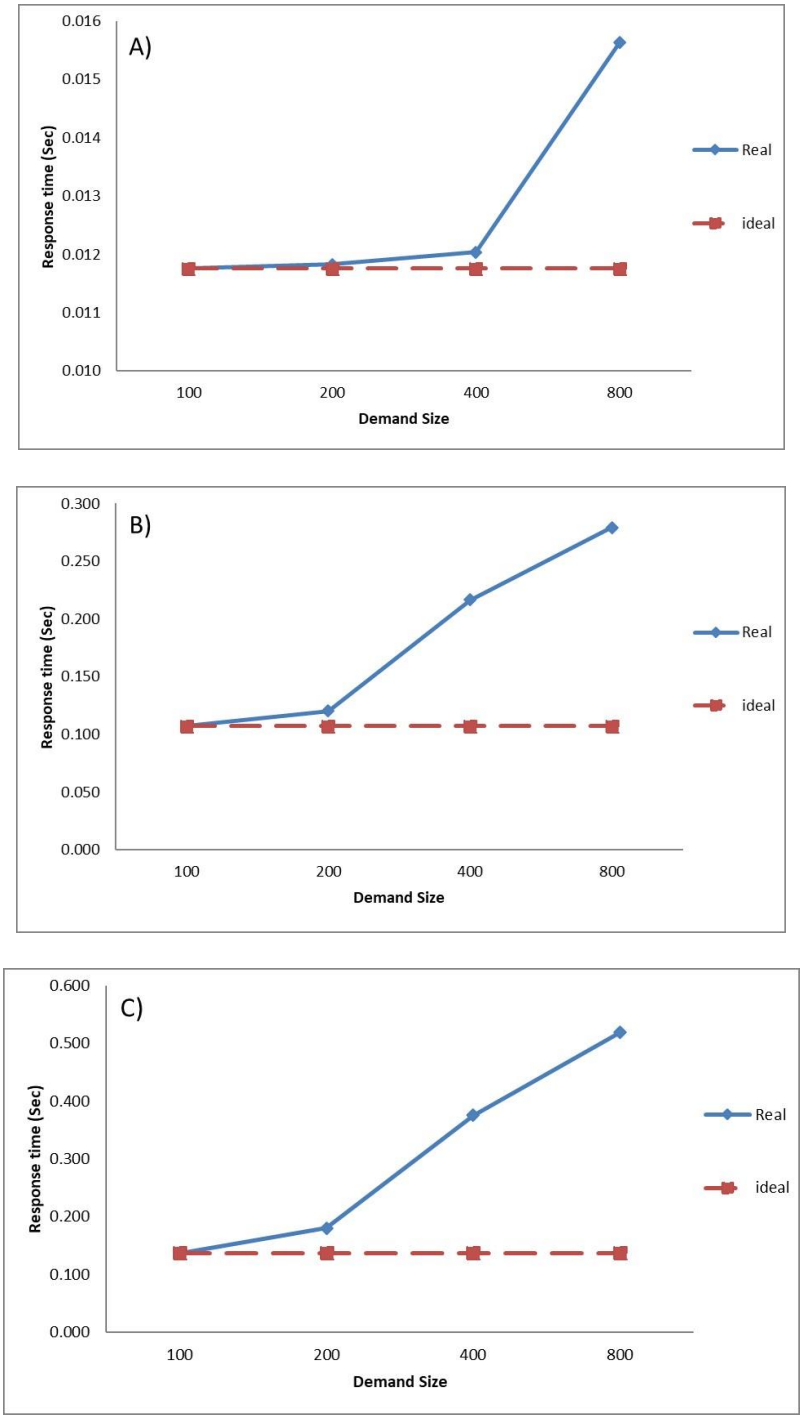


Figure 6. The average response times: A) steady rise and fall of demand; B) series of step-wise increases and decreases of demand; C) varied stepped rise and fall of demand.

We note that the application performs similarly in term of volume (instances) scaling for the first two scenarios (steady rise and fall of demand, and series of step-wise increases and decreases of demand), while in the varied stepped rise and fall of demand as shown in Figure 5C, the scaling acted slightly differently when demand hit 400 the scaling volume dropped.

The observed average response time values for the stepped rise and fall of demand scenario are shown in Figure 6B and for varied stepped rise and fall of demand in Figure 6C, starting from demand size of 200 the average response time increases significantly. In contrast, average response time values for the first scenario which shown in Figure 6A, have increased gradually from the demand size of 400 with less variation between values of average response times.

The values for the scalability metrics η_I and η_t for the three demand scenarios that we considered, are shown in Table 2. The calculated metrics show that in terms of volume scalability the first two scenarios are similar, the scaling being slightly better in the context of the scenario with step-wise increase and decrease of demand. The results show that the scaling volume for the third scenario dropped by 8-10 percent in comparison with the first two scenarios.

Table 2: Scalability Metrics

Scenario	Metric	
	η_I	η_t
Steady rise and fall	0.5687	0.9041
Step-wise increase and decrease	0.5882	0.5201
Varied Step-wise increase and decrease	0.4888	0.3834

In terms of quality scalability, the system scales much better in the context of the first scenario, steady rise, and fall of demand, than in the case of the second scenario with step-wise increase and decrease of demand, and the varied Step-wise increase and decrease scenario.

The values of the metrics indicate that in the context of variable demand scenarios (the second and third scenarios) - which is likely to be more realistic demand scenarios for many cloud-based software services - the quality scaling performance drops considerably in comparison with the simpler demand scenario.

5. Discussion

The proposed scalability metrics address both volume and quality scaling of cloud-based software services and provide a practical measure of these features of such systems. The works show how to integrate aspects of non-technical features [11] and also are distinct from elasticity oriented metrics [4]. This is important in order to support effective measurement and testing of scalability performance of the system from a technical perspective. Such analysis of scalability performance of those systems, can drive the design of scalability tests, system revision and upgrade focused on the improvement of scalability, or development of fine-grained monitoring of system' scalability performance. This investigation of realistic scalability performance can help to estimate the expectations of the system depending on demand scenarios and cloud platforms.

Having an effective measure of the volume and quality scalability of the system allows exploring the contribution of various system components to the scalability performance of the system. For example, using mutation testing [30] we can test the impact of small changes to particular components on the scalability performance. Alternatively, by instrumenting the whole code of the system [31] and then measuring its scalability through a range of demand scenarios we can identify the components of the system at various resolutions (e.g. units, classes, functions, methods) that contribute critically to variations in scalability performance. In this paper, the quality scaling is considered through the measurement of the average response time of the system. Other

aspects of quality scaling could be also used to define further similar but functionally distinct quality scaling metrics. For example, system throughput (i.e. the rate of successful delivery of service provision in response to service demand), or slowdown, or recovery rate [11] can be used for alternative quality scaling metrics. Expanding the range of quality scaling metrics provides a multi-factor view of quality scaling supporting the identification and definition of trade-off options in the context of quality-of-service offerings in terms of service scaling. The equations of the quality metric can be amended based on the nature of the quality factor that could replace or combine with the current quality scaling feature.

Due to the importance and need of measuring the scalability from an economic perspective, therefore, our scalability metrics can be integrated into the utility-oriented scalability metric proposed by Hwang et al. [9], by considering a combination of our metrics as the performance and/or quality components of the utility-oriented scalability metric. In [9] the utility-oriented scalability of the system is defined as the ratio of two utility oriented productivity metric values associated with two different configurations of the system (i.e. one configuration is a scaled-up version of the other). The utility oriented productivity metric (*P(Λ)*) is given as [9]:

$$P(\Lambda) = p(\Lambda) \cdot \omega(\Lambda) / c(\Lambda) \qquad (11)$$

Where *Λ* is the system configuration, *p(Λ)* is the performance component of the metric, *ω(Λ)* is the quality component of the metric and *c(Λ)* is the cost component of the metric. A natural way of integrating our technical metrics into this utility oriented framework is to use our volume and quality scaling metrics for the performance and quality components in (10) and thus re-define the productivity metric as

$$P(\Lambda) = \eta_I (\Lambda) \cdot \eta_t (\Lambda) / c(\Lambda) \qquad (12)$$

by adopting $p(\Lambda) = \eta_I(\Lambda)$ and $\omega(\Lambda) = \eta_t(\Lambda)$.

Table 3 shows the calculated values of the integrated productivity metric based on values of technical scalability metrics (see Table 2 for the metrics values) and cost (AWS t2.micro instance (0.0132$/hour)). It should be noted that the stepped scenarios (step-wise increase and decrease, and varied step-wise increase and decrease) which more realistic and powerful scenarios have scored lower than the simpler scenario. Our utility-oriented integrated scalability calculations show that in the case of the systems that we compared, the best choice is to use of simpler demand scenario on EC2.

Table 3: Integrated Scalability Metric

Scenario	Integrated metrics
Steady rise and fall	38.95
Step-wise increase and decrease	23.18
Varied step-wise increase and decrease	14.198

The authors also note that over-provision of cloud service instances that exceed the ideal scaling behavior is as much of an issue as under-provision, which been taken into account in future research. In the case of over-provision, the volume performance metric should be modified to cover over-provision scale, by considering the systematic nature of the deviation from the idea (downward or upward) in terms of its impact on the performance and on the geometric calculation in equation (5). On the other hand, the volume metric can be considered for extension to a larger volume.

Here we used three demand scenarios to demonstrate the effect of demands patterns on the scaling metrics. In principle, various demand scenarios may be used to fine-tune the cloud-based software service to fit particular demand scenario expectations. Similarly, considering a set of demand scenarios can also be used to identify changes in such scenarios that trigger interventions in terms of software upgrade or maintenance or direct investment of software engineering resources in the development of focused

upgrades for the system. Demand scenarios combined with multiple versions of quality scaling metrics can also be sued to determine reasonable quality-of-service expectations and likely variations of such expectations depending on changes in demand scenarios. We note the review [32] which concerns the study of the current practice of cloud service performance evaluation from system modelling perspective. It can be useful to adopt another demand scenario that already been used in the field, in order to track the impact of such scenarios.

We used only one cloud platform (Amazon AWS) and only one cloud-based software service (OrangeHRM) to demonstrate the application and usefulness of the scalability metrics [16]. Naturally, expanding the experiments to cover multiple cloud platforms and multiple cloud-based software services would provide a fuller picture of the application of the proposed metrics. Finally, we used one particular setting of the cloud service (i.e. virtual machine specification), one load generator and one auto-scaler to implement the demand scenarios and the scaling of the investigates cloud-based software service. Alternative load generators might have an impact on the values of the calculated metrics due to their implementation details, although in principle we would not expect a major impact of these on the reported results.

6. Conclusions

In this paper, two scalability metrics for cloud-based software services were introduced. One addresses the volume scalability of the service, while the other the quality scalability of the service. The metrics are based on simple principles of proportional scaling of the service volume and constant provision of the service quality and are defined using the differences between the real and ideal scaling curves for both the volume and quality scalability.

The proposed metrics can be used alone or integrate into utility oriented metrics of cloud-based service scalability [11]. In order to facilitate the scalability analysis of cloud-based software services from technical and utility-oriented perspectives. The metrics are demonstrated using a cloud-based software service (OragnceHRM) run on the Amazon AWS cloud platform and considering three demand scenarios. Our results show that the proposed metrics quantify explicitly the technical scalability performance of the system and also that they allow a clear assessment of the impact of demand scenarios on the cloud-based software service.

We believe that the proposed technical scalability metrics can be used to perform and design scalability testing of cloud-based software systems with the aim to identify system components that critically contribute to the technical scalability performance. Furthermore, the proposed metrics can be extended, by considering alternative service quality features, and combined with a range of demand scenarios to support the fine-tuning of the system, the identification of quality-of-service trade-offs, and estimation of realistic scalability performance expectations about the system depending on demand scenarios.

Future work will include the consideration of other cloud platforms (e.g. Microsoft Azure, Google Cloud, and IBM), demand workload generators and auto-scalers, and other cloud-based software services, so we get a wider range of measurements of the proposed metrics, extending the practical validity of the work.

Acknowledgment

This research is supported by a Ph.D. scholarship from Philadelphia University – Jordan for Amro Al-Said Ahmad.

Declaration of interest statement

The authors reported no potential conflict of interest.

References

[1] Liu HH. Software Performance and Scalability: A Quantitative Approach. Hoboken, N.J: Wiley Publishing; 2009.

[2] Atmaca T, Begin T, Brandwajn A, et al. Performance Evaluation of Cloud Computing Centers with General Arrivals and Service. IEEE Trans. Parallel Distrib. Syst. 2016;27:2341–2348.

[3] Becker M, Lehrig S, Becker S. Systematically Deriving Quality Metrics for Cloud Computing Systems. Proc. 6th ACM/SPEC Int. Conf. Perform. Eng. New York, NY, USA: ACM; 2015. p. 169–174.

[4] Herbst NR, Kounev S, Reussner R. Elasticity in Cloud Computing: What It Is, and What It Is Not. Proc. 10th Int. Conf. Auton. Comput. ({ICAC} 13) San Jose, CA: USENIX; 2013. p. 23–27.

[5] Lehrig S, Eikerling H, Becker S. Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. 2015 11th Int. ACM SIGSOFT Conf. Qual. Softw. Archit. 2015. p. 83–92.

[6] Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services. In: Hsu C-H, Yang LT, Park JH, et al., editors. Algorithms Archit. Parallel Process. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. p. 13–31.

[7] Hwang K, Shi Y, Bai X. Scale-Out vs. Scale-Up Techniques for Cloud Performance and Productivity. 2014 IEEE 6th Int. Conf. Cloud Comput. Technol. Sci. 2014. p. 763–768.

[8] AlJahdali H, Albatli A, Garraghan P, et al. Multi-tenancy in Cloud Computing. 2014 IEEE 8th Int. Symp. Serv. Oriented Syst. Eng. 2014. p. 344–351.

[9] Islam S, Lee K, Fekete A, et al. How a Consumer Can Measure Elasticity for Cloud Platforms. Proc. 3rd ACM/SPEC Int. Conf. Perform. Eng. New York, NY, USA: ACM; 2012. p. 85–96.

[10] Sharma U, Shenoy P, Sahu S, et al. A Cost-Aware Elasticity Provisioning System for the Cloud. Proc. 2011 31st Int. Conf. Distrib. Comput. Syst. Washington, DC, USA: IEEE Computer Society; 2011. p. 559–570.

[11] Hwang K, Bai X, Shi Y, et al. Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies. IEEE Trans. Parallel Distrib. Syst. 2016;27:130–143.

[12] Blokland K, Mengerink J, Pol M. Testing Cloud Services: How to Test SaaS, PaaS & IaaS Rocky Nook; 2013.

[13] Jayasinghe D, Malkowski S, Li J, et al. Variations in Performance and Scalability : An Experimental Study in IaaS Clouds Using Multi-Tier Workloads. IEEE Trans. Serv. Comput. 2014;7:293–306.

[14] Jayasinghe D, Malkowski S, Wang Q, et al. Variations in Performance and Scalability when Migrating n-Tier Applications to Different Clouds. IEEE 4th Int. Conf. Cloud Comput. 2011.

[15] Gao J, Pattabhiraman P, Bai X, et al. SaaS performance and scalability evaluation in clouds. Proc. 2011 IEEE 6th Int. Symp. Serv. Oriented Syst. IEEE; 2011. p. 61–71.

[16] Al-Said Ahmad A, Andras P. Measuring the Scalability of Cloud-based Software Services. 2018 IEEE World Congr. Serv. San Francisco: IEEE; 2018. p. 5–6.

[17] Al-Said Ahmad A, Andras P. Measuring and Testing the Scalability of Cloud-based Software Services. Fifth IEEE Int. Symp. Innov. Inf. Commun. Technol. (ISIICT 2018). Amman; 2018.

[18] Jennings B, Stadler R. Resource Management in Clouds: Survey and Research Challenges. J. Netw. Syst. Manag. 2015;23:567–619.

[19] Gao J, Bai X, Tsai WT, et al. SaaS Testing on Clouds - Issues, Challenges and Needs. 2013 IEEE Seventh Int. Symp. Serv. Syst. Eng. 2013. p. 409–415.

[20] Coutinho EF, Sousa FR de C, Rego PAL, et al. Elasticity in cloud computing: a survey. Ann. des Télécommunications 2015;70:289–309.

[21] Hu Y, Deng B, Peng F, et al. A survey on evaluating elasticity of cloud computing platform. 2016 World Autom. Congr. 2016. p. 1–4.

[22] Herbst NR, Kounev S, Weber A, et al. BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments. 2015 IEEE/ACM 10th Int. Symp. Softw. Eng. Adapt. Self-Managing Syst. 2015. p. 46–56.

[23] Bauer A, Herbst N, Kounev S. Design and Evaluation of a Proactive, Application-Aware Auto-Scaler: Tutorial Paper. Proc. 8th ACM/SPEC Int. Conf. Perform. Eng. New York, NY, USA: ACM; 2017. p. 425–428.

[24] Beltran M. Defining an Elasticity Metric for Cloud Computing Environments. Proc. 9th EAI Int. Conf. Perform. Eval. Methodol. Tools ICST, Brussels, Belgium, Belgium: ICST; 2016. p. 172–179.

[25] Kuhlenkamp J, Klems M, Röss O. Benchmarking Scalability and Elasticity of Distributed Database Systems. Proc. VLDB Endow. 2014;7:1219–1230..

[26] Ilyushkin A, Ali-Eldin A, Herbst N, et al. An Experimental Performance Evaluation of Autoscaling Policies for Complex Workflows. Proc. 8th ACM/SPEC Int. Conf. Perform. Eng. New York, NY, USA: ACM; 2017. p. 75–86.

[27] Jamal MH, Qadeer A, Mahmood W, et al. Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads. 2009 IEEE Int. Conf. Networking, Archit. Storage. 2009. p. 90–97.

[28] Brataas G, Herbst N, Ivansek S, et al. Scalability Analysis of Cloud Software Services. 2017 IEEE Int. Conf. Auton. Comput. 2017. p. 285–292.

[29] Lehrig S, Sanders R, Brataas G, et al. CloudStore — towards scalability, elasticity, and efficiency benchmarking and analysis in Cloud computing. Futur. Gener. Comput. Syst. 2018;78:115–126.

[30] Saleh I, Nagi K. HadoopMutator: A Cloud-Based Mutation Testing Framework. In: Schaefer I, Stamelos I, editors. Softw. Reuse Dyn. Syst. Cloud Beyond. Cham: Springer International Publishing; 2014. p. 172–187.

[31] Jayathilaka H, Krintz C, Wolski R. Performance Monitoring and Root Cause Analysis for Cloud-hosted Web Applications. Proc. 26th Int. Conf. World Wide Web Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee; 2017. p. 469–478.

[32] Duan Q. Cloud service performance evaluation : status , challenges , and opportunities – a survey from the system modeling perspective . Digit. Commun. Networks 2017;3:101–111.