Computers and Electrical Engineering 40 (2014) 1636–1649

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Performance evaluation of cloud computing platforms using statistical methods ☆

Gültekin Ataş [a], Vehbi Cagri Gungor [a,b,*]

[a] Department of Computer Engineering, Bahçeşehir University, Besiktas 34353, Istanbul, Turkey
[b] Department of Computer Engineering, Abdullah Gül University, 38039 Kayseri, Turkey

ARTICLE INFO

Article history:
Available online 5 May 2014

ABSTRACT

Cloud computing is a very attractive research topic. Many studies have examined the infrastructure as a service and software as a service aspects of cloud computing; however, few studies have focused on platform as a service (PaaS). According to recent reports, demand for enterprise PaaS solutions will increase continuously. However, different sectors require different types of PaaS applications and computing resources. Therefore, an evaluation and ranking framework for PaaS solutions according to application needs is required. To address this need, this study presents the most essential aspects of PaaS solutions and provides a framework for evaluating the performance of PaaS providers. It also proposes a suitable set of benchmarking algorithms that can help determine the most appropriate PaaS provider based on different resource needs and application requirements. Performance evaluations of three well-known cloud computing PaaS providers were conducted using the analytic hierarchy process and the logic scoring of preference methods.

1. Introduction

Cloud computing has been one of the most popular topics in computer technologies in the past decade. Although the idea itself is not new, commercial and broad use of cloud computing began in the early 2000s. There are three types of services provided in cloud computing: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Initially, the most prevalent service type was SaaS. However, with recent advances in Internet technology, the other cloud service types have become more widely available. PaaS is offered with the infrastructure that is provided in IaaS (hardware, virtual machines, and protocols). In addition to IaaS, development frameworks, run-time libraries, deployment systems, source code control systems, and storage systems are also available. Therefore, consumers do not need to spend time installing, managing, and configuring such systems. Providers handle all these administrative and operational burdens to facilitate the related cloud services [1].

Companies of every scale have started to use cloud solutions in the last decade [2]. The forecasts and trends differ; however, they show that high rates for the usage of cloud computing platforms are expected in the near future. Governments are also making huge investments and policy changes to migrate cloud solutions. U.S. government institutes have announced their cloud computing policies and progress in migrating to cloud computing solutions [3,4]. India, China, Brazil, Russia, South Africa, Turkey, and South Korea are also making huge investments in cloud computing, as well as supporting

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Danielo G. Gomes.

* Corresponding author at: Department of Computer Engineering, Abdullah Gül University, 38039 Kayseri, Turkey. Tel.: +90 0352 224 8800.
E-mail addresses: gultekin.atas@stu.bahcesehir.edu.tr (G. Ataş), cagri.gungor@bahcesehir.edu.tr, cagri.gungor@agu.edu.tr (V.C. Gungor).

http://dx.doi.org/10.1016/j.compeleceng.2014.03.017
0045-7906/

companies with stimulus packages. It has been stated that most of the World Economic Forum 2011 participants believe that the impact of cloud computing will exceed the impact of mobile technologies [5]. Therefore, as is the case for other cloud services, PaaS solutions will become an indispensable option for businesses of any size. Companies will need to determine the most suitable PaaS for their requirements. However, it is expected that no single PaaS provider will be able to offer the best performance in terms of all performance metrics due to the many different needs of companies looking for PaaS solutions. Thus, a system that helps evaluate the provided cloud computing solutions based on application requirements is needed.

Until recently, many studies have examined key performance metrics for cloud computing. However, even though central processing unit (CPU) power and memory bandwidth are critical key performance metrics for PaaS and IaaS, these metrics have not been studied in detail. In related literature, CPU power has been roughly tested with benchmarking algorithms based on floating-point calculations, or response times have been measured using some scientific applications. Such testing methods may be suitable for IaaS; however, they are not suitable for PaaS. In addition, these algorithms are not easy to use and configure; thus, they may not be widely adopted as performance test benchmarks. Therefore, an appropriate set of PaaS performance metrics and benchmarks are two primary open research issues.

To address these needs, in this study, 19 different functions grouped in three main categories are proposed to evaluate PaaS solutions based on different application requirements. Commercial PaaS alternatives, such as Cloud Foundry, Heroku, and OpenShift, are tested using these functions. In addition, these commercial PaaS providers are compared by applying two performance evaluation methods: the Analytic Hierarchy Process (AHP) and logic scoring of preference (LSP). Performance evaluations have been performed for four different scenarios, which correspond to different computing resource requirements and different application types. Unlike existing studies, in which database services have been studied separately relative to cloud computing, the proposed framework evaluates database services together with other PaaS computing resources. The main contributions of this study can be summarized as follows.

- A detailed feature framework for PaaS solutions is proposed, which consists of 19 performance functions in three main categories. This framework spans the most essential aspects of PaaS solutions and provides a comprehensive performance evaluation framework for PaaS alternatives.
- A set of suitable benchmark algorithms, such as the Whetstone algorithm for CPU performance, the Stream algorithm for memory bandwidth, and a specific set of basic queries for database operations, are proposed to evaluate PaaS solutions in detail.
- Four different scenarios are presented to simulate different cloud computing applications with specific resource requirements. The AHP and LSP methods are applied to these scenarios separately. To the best of our knowledge, this is the first study in which the LSP technique has been applied to the selection of PaaS problems. In addition, the stability of three commercial PaaS providers (i.e., Cloud Foundry, Heroku, and OpenShift) is compared.

The remainder of this paper is organized as follows. In Section 2, we present related work on performance evaluations for cloud computing platforms. In Section 3, we discuss the benchmarking algorithms used in our performance tests and present the features of the proposed framework. In Section 4, we provide a mathematical overview of the AHP and LSP methods. Section 5 provides performance metrics, details of the test environment, performance test data, and evaluation and discussion of the AHP and LSP results. Finally, Section 6 concludes the paper.

2. Related work

A number of studies have focused on performance evaluations of cloud computing. Most have studied IaaS providers or IaaS infrastructure. Some have focused on the properties of alternatives without any performance comparisons [6–9], and others have performed some tests. However, they have not presented a detailed comparison framework or evaluation method [10–14].

In the related literature, the Cloud Services Measurement Initiative Consortium (CSMIC) has defined several categories or service measurement indexes to map key performance indicators, such as agility, assurance, accountability, economics, privacy, security, performance, and usability [15]. These metrics are business-relevant indicators; thus, the presented performance metrics, such as accuracy, functionality, suitability, and interoperability, do not directly match technical performance metrics of cloud computing platforms. However, there are many established information technology (IT) performance metrics that are more suitable for technical PaaS performance comparisons, such as response time and megabytes per second (MB/s).

In another study, six main criteria, i.e., storage, virtualization, network, management, security, and vendor support, have been proposed [16]. In that study, Abiquo, Eucalyptus, Mosaic, Nimbus, and OpenNebula were the evaluated platforms; however, no detailed performance evaluations were presented.

Other studies have compared several platforms (i.e., AbiCloud, Eucalyptus, Nimbus, and, OpenNebula) according to their features or structures [7,8], such as deployment type, scalability, virtual machine (VM) support, operating system (OS) support, compatibility, disk image options, disk image storage, hypervisors, customizability, dynamic host configuration protocol, and network issues. However, performance was not detailed in terms of technical functions.

AbiCloud, Eucalyptus, OpenNebula, and Nimbus have also been compared in another study [9] according to 13 features: architecture, programming language, deployment model, supported hypervisors, data/VM memory, area of application, user interface, OS licenses, robustness against error, interoperability, security, VM creation tools, and compatibility with public cloud providers. These are the features that companies will compare when selecting a solution; however, technical performance metrics must also be considered. In addition, in that work, alternatives were not tested; only their features were listed. However, recommendations and comparisons were proposed for each platform for different business needs.

Amazon Elastic Compute Cloud (EC2), GoGrid, ElasticHosts, and Mosso have been evaluated in another study; however, this study focused primarily on IaaS [10]. Therefore, boot times and allocation and release times of virtual machines were some of the measured items. Larry McVoy's (LM) bench suite was used to measure CPU performance; however, it is based on only floating operations. In that study, some parts of the high performance computing challenge benchmark (HPCC) and Stream were also used as benchmarks. In addition, this study was based on many-task scientific computing. Thus, it may not be a suitable application domain for the IT industry. Amazon EC2, ElasticHosts, and BlueLock have also been tested and compared as IaaS solutions [11]. That study tested CPU and memory performance and disk input/output (I/O). They used Simplex as a benchmarking tool for CPU performance, Stream for memory performance, and the flexible input output (FIO) tool for disk I/O benchmarking. However, they did not employ any statistical method for comparison and ranking.

A public cloud (Microsoft Azure) and a private cloud (Nimbus) have been compared considering the needs of scientific applications [12]. Although scientific applications can be consumers of cloud services, future trends show that companies will use cloud platforms more extensively. Although the tests were detailed in that study, a performance test approach should be based on different resource and business requirements. In addition, a statistical comparison method is needed to effectively select a service provider among the alternatives.

In another study, obstacles and opportunities of cloud computing have been discussed. Amazon, Microsoft Azure, and Google AppEngine were studied as examples of different cloud computing service types [17]. This work focused on the advantages and disadvantages of cloud computing; however, a comparative performance evaluation was not included. Although some memory and I/O test results were reported, cost evaluation received more emphasis. Since each example was a different service type, performance tests were not performed.

Amazon Web Services, Microsoft Azure, Google AppEngine, and Rackspace CloudServers were tested and compared in another study [18]. The tested topics were CPU performance, memory, and disk I/O, and the metrics were response time, network throughput, and latency when downloading/uploading 1 KB and 10 MB files. However, they only compared values in test result data tables; no statistical evaluation method was employed.

Mao and Humphrey [19] have examined Amazon EC2, Windows Azure, and Rackspace. Their primary focus was on VM performance. They investigated VM size and operating systems and tested startup times. They compared the CPU and memory performance and disk capacity of these cloud alternatives, which are important parameters.

Amazon EC2, Microsoft Azure, and Rackspace were also investigated by another study [20]. The studied solutions focused on IaaS; therefore, the authors compared configuration and boot performance. To evaluate computing performance, Numerical Aerodynamic Simulation Parallel Benchmarks were used. They compared the cost of each solution against computing performance. The overall evaluation items were computing performance and cost; thus, the test results shown in charts were sufficient to select the best option.

Another study has compared the performance of a website. In this study, one instance was running on Azure, one was running on a local server, and the third instance was running on a commercial web hosting company [13]. The performance of these three installations was compared. The website of a vocational school was used as the test case. However, such a comparison may not be sufficient for many companies as a basis for evaluation and decision making. Large business operations require more computational resources and have more complex structures than a school's website. Since the metrics and performance items employed in this study were limited, it was easy to compare the alternatives by simply reviewing test data. Therefore, no evaluation method was required. The use of online web applications to evaluate performance has also been discussed [14]. The examined application was an online bookstore with several interactions. A performance comparison was made based on this application. Similar to the previous study, this study did not provide clearly separated functional areas to compare solutions, and an online bookstore application cannot simulate many real world applications. In addition, they did not focus on performance comparisons.

Another study has conducted a performance evaluation and comparison of cloud providers [21], in which a set of comparison metrics and an evaluation framework were proposed. They used some of the CSMIC metrics, such as agility, assurance, accountability, cost, security, and performance, for comparison of the alternatives. The AHP method was used to evaluate these numeric and non-numeric comparison criteria. However, performance metrics were not studied in detail, and the only tested performance metric was response time. They focused on functional features rather than performance.

The AHP method has been used to compare SaaS solutions [22]. The authors defined the selection parameters as functionality, architecture, usability, vendor reputation, and cost. They also proposed a parameter breakdown and reported that their values for each weight were obtained through surveys. However, they simply compared SaaS products (names not provided) according to these weights. They did not include a technical performance test.

Other studies have performed evaluations of web services. Essentially, in these studies, the performance and properties of the providers were measured and recorded in a database. When a service is requested by the user, they attempt to match the needs, which are submitted in configuration files, with the database records and ranks of the available service providers. LSP is one of the methods used to score the available services in this area. In one of these studies, researchers used LSP to

dynamically select a web service according to a given Extensible Markup Language configuration file for their application [23]. They investigated an online payment service scenario. In this scenario, a fictitious company has the choice of selecting one of four different payment services, and according to the parameters (e.g., the device that customers use to connect to the e-commerce site, credit card type, and country), this study uses the LSP method to match or rank the best service and then use that service. In a similar study, LSP was employed in a software algorithm used to rank services in the selection of a web service procedure [24]. They combined ordered weighted averaging (OWA) operators with LSP. Similarly, Maheswari and Karpagam [25] used LSP and OWA operators for semantic web service selection. Their reported data repository contains functional and non-functional properties of services.

Although all these studies provide valuable foundations in the area of cloud computing, none provide comprehensive and detailed performance metrics for PaaS alternatives. A suitable set of benchmarks is required to support such a detailed performance evaluation. In addition, in the existing studies, database performance is neglected and is not considered a part of the solution. Very few studies have employed good statistical methods to evaluate results.

3. Summary of benchmarks and evaluation methods

Here we provide a summary of the benchmarks used in our performance evaluations, and the proposed main and sub functions for PaaS evaluations are presented. In addition, the AHP and LSP methods, which are used to statistically evaluate the examined cloud computing platforms (i.e., Cloud Foundry, Heroku, and OpenShift), are introduced.

3.1. Benchmarks

Many well-established benchmarking concepts exist in the IT sector. For computing power performance benchmarking, the Whetstone benchmarking algorithm has been used. This algorithm has been adopted by sector leaders for several decades. The Whetstone algorithm was originally based on floating operations, similar to many other benchmarking algorithms, such as Linpack, LM suite, and HPCC. However, new features have been included to measure other types of operations [26]. The Whetstone benchmark algorithm is a synthetic benchmark that attempts to solve several different types of mathematical functions. It checks the performance of eight different computing areas such as floating-point operations with array elements, floating-point arrays as parameters, if-else conditional jumps, integer/fixed-point arithmetic, procedure calls with floating-point operations, and trigonometric and other mathematical functions (e.g., exponential, square root, and logarithmic operations). We have slightly modified the original Whetstone algorithm's code to conduct tests with JavaServer Pages (JSP). These changes are not related to calculations or loops; they simply transform a Java application to a JSP application.

For database performance benchmarking, we have coded basic select, update, delete, and insert queries in Java to evaluate the off-the-shelf performance of the native database infrastructures offered by the providers. In the database benchmarking, two different test sets are performed for each query type. In the first query type, each query's scope is a single record, and the query is repeated 100 times in a loop. Every time a query is performed, the connection, statement, and result set are released and created again. In the second query type, each query's scope was 100 records, and the connection, statement, and result set are again released and created for each query. The single record queries represent database operations such as transactions that target single records in tables. Queries that target 100 records represent bulk operations such as jobs running during end-of-day operations.

For main memory performance benchmarking, the Stream benchmark algorithm is used. This algorithm measures sustainable real world bandwidth not theoretical peak values. It is a widely known and applied algorithm in memory bandwidth performance tests [27]. The Stream algorithm is also a synthetic benchmark that creates synthetic workloads, not real world workloads. The benchmark measures bandwidth performance using four long vector/matrix operations, such as "copy", "scale", "add", and "triad" (a combination of the other three). The algorithm assigns one array to another to measure the "copy" operation time and bandwidth. For a "scale" operation, the algorithm multiplies elements of an array by a scalar value. In the "add" operation, it adds elements of an array to elements of another array and assigns the results to a third array. "Triad" performs the calculations of the three operations together (copy, scale, and add). In our evaluations, we used the average results for these four operations.

The Stream algorithm creates large matrices or arrays that consist of double type (i.e., 64-bit floating point) objects. The sizes of the arrays can be adjusted according to the size of the required workload. We adjusted the parameters to obtain 128 KB, 1 MB, and 100 MB memory objects. These workload sizes are used to represent small, medium, and large data for memory performance.

3.2. Proposed functionalities and their decomposition for PaaS solutions

The proposed performance criteria framework for selecting the best PaaS solution is categorized by 19 sub functions organized into three main categories. The decomposition of these components is detailed in Table 1.

4. Evaluation methods

There are three main performance functions and 19 sub functions in our framework to evaluate the results according to customer requirements. Determining which platform performs better is considered a multiple criteria decision making

Table 1
Proposed criteria framework.

Main category	Functions	Sub-functions
CPU performance	Floating point with array elements	
	Floating point with array as parameter	
	Conditional jumps (if then else)	
	Integer arithmetic (fixed point)	
	Trigonometric functions (sin, cos, etc.)	
	Procedural calls with floating point	
	Array reference assignments	
	Mathematical functions (exponential, square root, logarithm operations, etc.)	
Database performance	Single record	Read
		Update
		Insert
		Delete
	100 Records	Read
		Update
		Insert
		Delete
Memory performance	Small size (128 KB)	
	Medium size (1 MB)	
	Large size (100 MB)	

(MCDM) problem [28,29]. There are many mechanisms or methods for ranking the performance indicators of this type of problem. Two of them are used in this study, i.e., the AHP and LSP methods. To the best of our knowledge, LSP has not previously been used for performance comparison of PaaS solutions. The AHP and LSP methods are applied to four different scenarios that simulate four main types of computing resource requirements. These methods were chosen because of their flexibility of choice (e.g., weighting requirements as customer preference) and their systematic decomposition of performance elements.

4.1. Analytic hierarchy process

The first method employed to solve the MCDM problem is the AHP mechanism [28]. This process has been selected because it is widely used in many different research areas, including computer sciences [21,30]. The AHP model first decomposes a problem into primary components that are structured hierarchically. These components are goal, criteria, and alternatives. AHP uses relative values for performance variables. The AHP hierarchical model of the problem is shown in Fig. 1.

The AHP method uses pairwise comparison and an eigenvector method to calculate relative ranking. The criteria for the main and sub functions are accompanied by weights. Thus, anyone can change the weights and obtain the corresponding evaluation results for their functionality priorities. In this study, the AHP is used with the Relative Service Ranking Vector (RSRV) for performance evaluation. This method has also been used in the work of Garg et al. [21].

The calculations and processing begin from the leaves (i.e., sub functions that cannot be decomposed further) and proceeds to the topmost node, which is the final score or comparison result of the overall system in the proposed framework. The first operation in the process is to find the relative values. The relative values are calculated for the test results of the atomic sub functions for all PaaS alternatives. For the CPU and memory bandwidth categories, higher values are better. Thus, to find the relative values for each provider, the test result of the provider is divided by the results for the other providers. However, for database tests, the time for each operation is measured; thus, lower values are better. Therefore, to find the relative value, the division operation is reversed in the calculations, i.e., the other providers' test values are divided by the test result of the examined provider.

Let $P(p_1,\ldots,p_n)$ be the set of PaaS providers, where n is the number of providers. Let $X(x_1,\ldots,x_k)$ be the set of performance item test values, where k is the number of performance items (e.g., computing power, memory, or their sub functions, such as floating-point performance or copying large files to memory). Let $W(w_1,\ldots,w_k)$ be the set of weights determined by the user. The relative performance item test result values for a provider form the Relative Service Ranking Matrix (RSRM) shown below.

$$\boldsymbol{RSRM_k} = \begin{bmatrix} x_1/x_1 & x_1/x_2 & \ldots & x_1/x_n \\ x_2/x_1 & x_2/x_2 & \ldots & x_2/x_n \\ \ldots & \ldots & \ldots & \ldots \\ x_n/x_1 & x_n/x_2 & \ldots & x_n/x_n \end{bmatrix} \quad (1)$$

Let $\boldsymbol{V_k^i}$ be the eigenvector for the ith square of the RSRM matrix for the kth performance item. The RSRM matrix is iteratively squared until the eigenvector difference of two successive ones is negligible. The RSRV matrix is obtained when the difference of $\boldsymbol{V_k^i}$ and $\boldsymbol{V_k^{i-1}}$ is negligible, as shown in Eqs. (2) and (3). The RSRV matrix is calculated for each performance item in a node. Then, each RSRV is combined as columns of the RSRM of the upper node of the function tree. The RSRM of the node

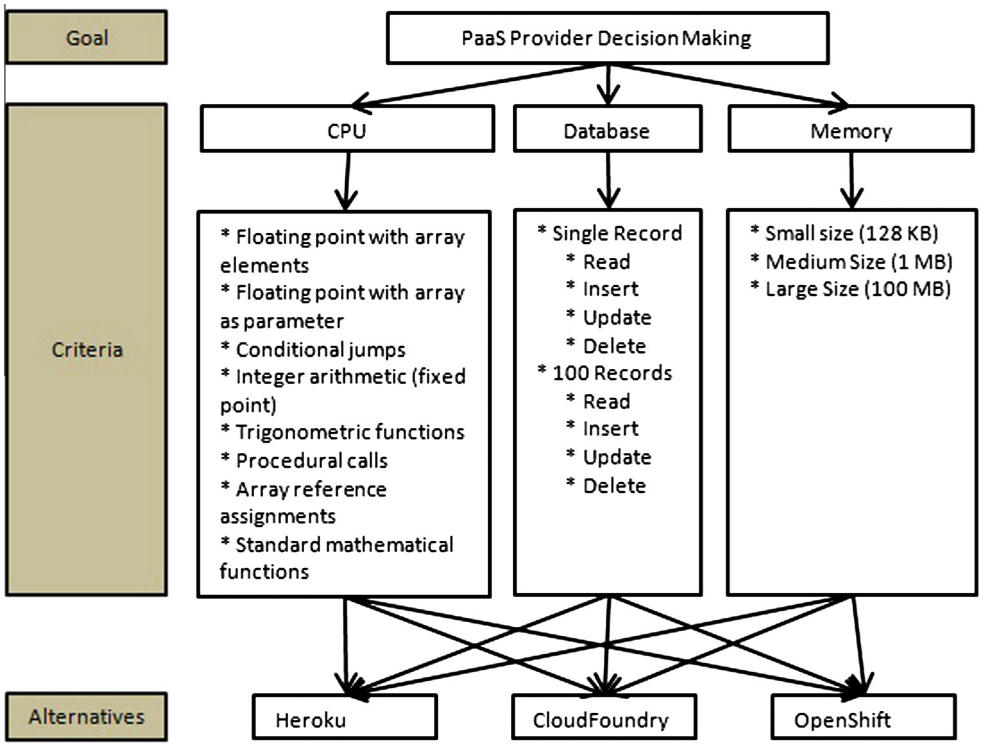

Fig. 1. AHP model of PaaS evaluation problem.

is multiplied by the weights vector, which is composed of weights for each item in the RSRM. Then, the RSRV of the function node is obtained. This calculation and formation of the RSRM and RSRV procedures are repeated until the RSRV of the root node (i.e., the overall system) is obtained, as expressed by Eq. (4). The $\boldsymbol{W_k}$ weight vector for the performance items is expressed by Eq. (5). The overall RSRV is calculated as the multiplication of the overall system RSRM and this weight matrix, as expressed by Eq. (6).

$$\boldsymbol{RSRV_k} = \boldsymbol{V_k^i}, \boldsymbol{V_k^i} - \boldsymbol{V_k^{i-1}} \simeq 0 \tag{2}$$

$$\boldsymbol{RSRV_k} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix} \tag{3}$$

$$\mathrm{RSRM_{node}} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1k} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \dots & \dots & \dots & \dots \\ v_{n1} & v_{n2} & \dots & v_{nk} \end{bmatrix} \tag{4}$$

$$W_{\mathrm{node}} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_k \end{bmatrix} \tag{5}$$

$$\mathrm{RSRV_{node}} = \mathrm{RSRM_{node}} \otimes W_{\mathrm{node}} \tag{6}$$

4.2. Logic scoring of preferences

The LSP method is a generalized and extended version of several scoring methods [31]. The LSP method is rooted in continuous preference logic. Similar to the AHP method, each performance variable is decomposed until no further

decomposition is possible or until each item can be measured and evaluated by itself. In this study, these performance variables are decomposed as shown previously in the proposed framework presented in Section 4.1.

For each performance variable, X_i ($i = 1,\ldots,n$), an acceptable value range is defined, i.e., $X_{\min}$, $X_{\max}$, and the values between them. The elementary preference E_i indicates the ratio of variable satisfaction according to needs. Therefore, the value of performance variable X_i should be converted to E_i to rank each compared system. This mapping function is denoted by the elementary criterion function G_i, which, for simplicity, should be a piecewise linear function. The elementary preference is calculated according to Eq. (7).

$$E = \begin{cases} 0, & X_i \le X_{\min} \\ G_i(X_i), & X_{\min} < X_i < X_{\max} \\ 100\%, & X_i \ge X_{\max} \end{cases} \tag{7}$$

Hence, the elementary preference value will be $0 \leqslant \mathrm{Ei} \leqslant 100\%$. Here the maximum and minimum values are referred to as cut-off points. The elementary criterion function can be selected as a preference scale, as expressed by Eq. (8).

$$G_i(X_i) = (x_i - x_{\min})/(x_{\max} - x_{\min}) \tag{8}$$

Therefore, the elementary preference values are calculated using the test values, Eq. (8), and acceptable ranges that are suitable for intervals of the test results. Using these n elementary preferences for n performance variables, the global preference E is calculated using a stepwise aggregation technique. This is a function of all preferences, as expressed by Eq. (9).

$$E = L(E_1,\ldots,E_n) \tag{9}$$

The function L can be modeled as a weighted power mean of an appropriately selected power of these preference values. Assuming that $e_1,\ldots, e_k$ are input preferences of the aggregation block, and $W_1,\ldots, W_k$ are weights that represent the relative significance of these inputs, the output preference e_0 is expressed as follows.

$$e_0 = \left(W_1 e_1^r + \cdots + W_k e_k^r\right)^{1/r}, \quad \text{where } W_1 + \cdots + W_k = 1, \quad W_i > 0, \quad \text{and } i = 1,\ldots,k \tag{10}$$

The aggregation function is employed for each sub performance item. Then, another aggregation function is employed to calculate the preference of the category containing these sub performance items. The aggregation continues until the top or system level preference is obtained. Therefore, the function L represents the aggregation of all aggregations. The overall aggregation scheme of the system is shown in Fig. 2.

Here the power r is a real number, and it must be appropriately chosen to construct logical relations of the inputs of the aggregation function. It is a function of d, i.e., the disjunction degree. Here d is an indicator of the average position of e_0, which is between the maximum and minimum. d is defined so that $0 \leqslant d \leqslant 1$; $d = 0$ results in $e_0 = e_{\min}$; $d = 1$ results in $e_0 = e_{\max}$. This means that r is a complex function of d, i.e., $r = \rho(d)$.

Here ρ is a complex function of d. For special values of d, r transforms the function shown in Eq. (10) into specific formations. The weighted power r makes the function pure conjunction when it takes the value of negative infinity. This is referred to as the minimum function. When it is -1, the function becomes the harmonic mean. When it takes the value zero, it makes the function a geometric mean function. When it is +1, the function becomes the square mean, and when it is positive infinity, the function becomes pure disjunction, which is referred to as the maximum function. Therefore the preference aggregation function is expressed as follows.

$$e_0 = \left((W_1 e_1)^{\rho(d)} + \cdots + (W_k e_k)^{\rho(d)}\right)^{1/\rho(d)} \tag{11}$$

This is referred to as the *generalized conjunction/disjunction* or *andor*. Typically, d is not calculated; it is already calculated for special cases of the logical function. The names of these special cases, their symbols, and value for r are given in Table 2. The r used in the table is the number of inputs for the aggregator (not the r in the equations above). $r5$ values are used for the CPU aggregator since the value for five, and eight inputs are approximately the same in practice.

The aggregation functions are selected so that the logical relations of the items enter the aggregator. These relations are listed in Table 2. For the CPU aggregation function, the medium quasi-conjunction (QC) (denoted CA in Table 2) is selected to determine the value of d because the value of sub items should not be compensated by the value of another sub item. In addition, sub items are not strongly related. Following similar reasoning, we believe that more synchronization is required for database sub functions compared to CPU sub functions. All sub database functions (read, update, insert, delete) are indispensable for any enterprise application. Therefore, they should not be replaceable or compensated for one another. Hence, strong QC− (C+−) is selected. For database functions (i.e., single record and 100 records) and memory functions, weak QC+(C−+) is selected because they are not required simultaneously (i.e., may compensate each other). In addition, for the overall system aggregation function, strong QC (C+) is selected. The selection of operation types for the relationships mentioned here is based on the intuitive understanding of the importance of functions. We attempt to roughly map simultaneity with highly required functions by considering enterprise applications.

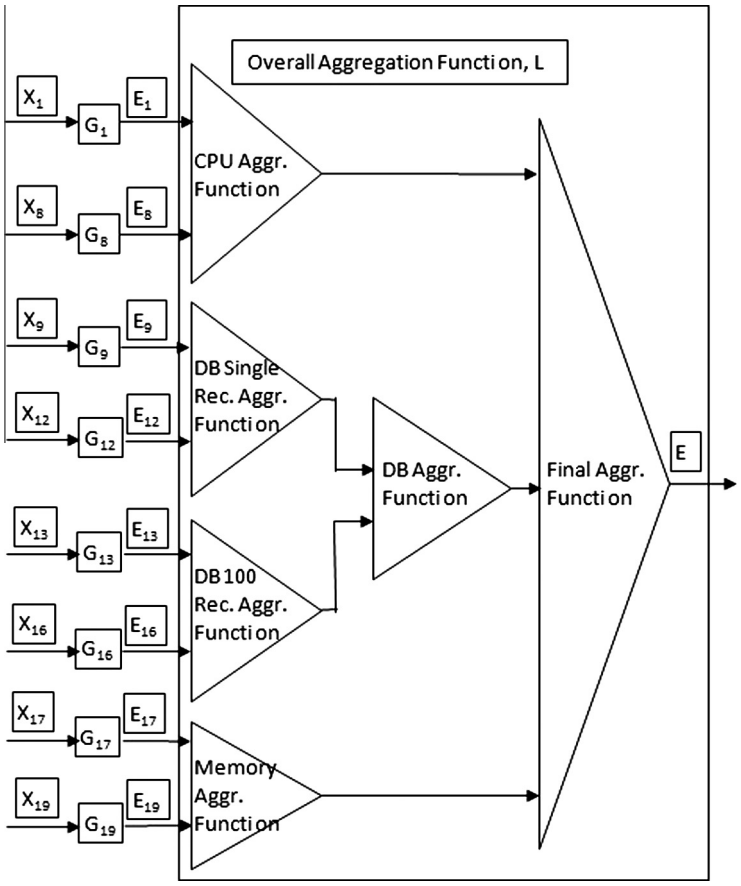

Fig. 2. Aggregation function scheme for the PaaS evaluation problem.

Table 2
Logical functions and parameters for aggregation functions.

Operation	Symbol	*d*	*r*2	*r*3	*r*4	*r*5
DISJUNCTION	D	1.0000	+Infinity	+Infinity	+Infinity	+Infinity
STRONG QD(+)	D++	0.9375	20.6300	24.3000	27.1100	30.0900
STRONG QD	D+	0.8750	9.5210	11.0950	12.2700	13.2350
STRONG QD(−)	D+−	0.8125	5.8020	6.6750	7.3160	7.8190
MEDIUM QD	DA	0.7500	3.9290	4.4500	4.8250	5.1110
WEAK QD(+)	D−+	0.6875	2.7920	3.1010	3.3180	3.4790
WEAK QD	D−	0.6250	2.0180	2.1870	2.3020	2.3840
SQUARE MEAN	SQU	0.6232	2.0000			
WEAK QD(−)	D−−	0.5625	1.4490	1.5190	1.5650	1.5960
ARITHMETIC MEAN	A	0.5000	1.0000	1.0000	1.0000	1.0000
WEAK QC(−)	C−−	0.4375	0.6190	0.5730	0.5460	0.5260
WEAK QC	C−	0.3750	0.2610	0.1920	0.1530	0.1290
GEOMETRIC MEAN	GEO	0.3333	0.0000			
WEAK QC(+)	C−+	0.3125	−0.1480	−0.2080	−0.2350	−0.2510
MEDIUM QC	CA	0.2500	−0.7200	−0.7320	−0.7210	−0.7070
HARMONIC MEAN	HAR	0.2274	−1.0000			
STRONG QC(−)	C+−	0.1875	−1.6550	−1.5500	−1.4550	−1.3800
STRONG QC	C+	0.1250	−3.5100	−3.1140	−2.8230	−2.6060
STRONG QC(+)	C++	0.0625	−9.0600	−7.6390	−6.6890	−6.0130
CONJUNCTION	C	0.0000	−Infinity	−Infinity	−Infinity	−Infinity

5. Performance metrics and results

Here the performance evaluations of three different platforms and their test results are presented with the employed metrics. We used two different evaluation methods, i.e., the AHP and LSP methods, to compare the results for the three

platforms. Performance tests were conducted using the three main functions composed of 19 sub functions of the proposed framework. The performance metrics used in these measurements are as follows.

MFLOPS: Millions of Floating Point Instructions per Second is the number of floating-point operations performed by the CPU in one second. Most CPU benchmarking only measures this floating-point calculation performance. This metric is used for the Whetstone benchmark results for floating point with array elements, floating-point array as a parameter, and array reference assignment category results.

MOPS: Millions of operations per second is used for non-floating-point operations. Generally, only floating-point operations are measured; however, this type of operation is not measured. For example, in this study, it is used to measure conditional (if-else) jumps, fixed-point (integer) operations, trigonometric calculations, procedural calls, and mathematical operations, such as square root or exponential calculations.

Response time (ms): For each function of CPU performance evaluation, the Whetstone algorithm uses a routine to complete or a problem to solve when it calculates the MFLOPS/MOPS metrics. Response time is the time required to complete a routine or problem. This metric is used to show test results in charts. However, since it is closely related to the MFLOPS/MOPS metrics, response time is not used in the AHP and LSP evaluation methods. MFLOPS/MOPS are preferred because the problem of the algorithm may not be adopted in general.

Average total operation time for a single record (ms): For each basic database operation (read, update, insert, delete), this is the average total elapsed time for connection creation, statement creation, and operation execution. For this metric, the used queries affect only a single record.

Average total operation time for 100 records simultaneously (ms): Similar to the previous metric, this is the average total elapsed time for connection creation, statement creation, and operation execution. However, for this metric, the queries used affect 100 records.

Memory bandwidth (MB/s): This measures the data transfer to and from memory. Three different sizes of memory objects are used to measure memory performance under different conditions. These memory object sizes are 128 KB, 1 MB, and 100 MB.

The Whetstone algorithm is used to test CPU performance, and the Stream algorithm is used to test memory bandwidth. Basic Java database methods are used to test database operations; i.e., basic DriverManager.getConnection(), connection.createStatement(), and statement.executeQuery() method. The test database has one table with five columns, representing an employee table. The table record is approximately 150 bytes, containing varchar and integer-type fields. Test algorithms are applied with Java using the Eclipse integrated development environment. Java code is implemented in JSP because the overall results of the platform are intended to be measured; e.g., response time begins with receiving operation instructions and ends when the instructions complete. Therefore, the performance of any additional tiers (e.g., database system implementation and other background services) is included. JSP is also preferred because of the ease of code deployment and data collection over web pages. The eGit plug-in is used to automate uploading applications to the cloud. The Chrome Internet browser is used to render the test web pages that begin the required operations and return the results to the client. For all tests, one test cycle is performed to warm up the platform, i.e., starting the application and loading cache. The Whetstone tests are repeated 355 times because looping CPU tests causes timeouts. Single record database queries are repeated 1400 times, and multi record database operations are repeated 1100 times. Small- and medium-sized memory tests are repeated 1000 times; however, large memory tests are repeated 300 times because looping large size tests results in timeouts.

The most recommended or most popular database services have been used for each provider. If a database service is recommended by the provider and is one of the most provided services by other providers, then that database service is used for testing. If it is not widely provided, then the second most recommended or most popular database service offered by the provider is used. The MySQL database service is used for OpenShift and Cloud Foundry tests. Heroku also provides the MySQL service; however, their first recommendation is the PostgreSQL database service. PostgreSQL is selected rather than MySQL for Heroku because it is also provided by most PaaS providers. In addition, the test setup contains a simple table; thus, no tuning is required. The tests aim to show the default basic setup of database system implementation for each platform. For complex databases and relational data structures, tuning would be required.

Heroku's configuration is the basic evaluation subscription. Heroku names a processing unit dyno, which is an isolated container with computing resources in their own virtualized environment. A basic Heroku subscription provides one dyno, which includes 512 MB memory and $1\times$ CPU share. The CPU computing power is not declared to the user or the public. There are claims that the computing power of one dyno is approximately one micro-EC2, which is at maximum two Amazon EC2 computing units (ECU); one ECU is equivalent to a 2007 1.0–1.2 GHz Xeon processor. OpenShift names its computing unit a gear. Their basic subscription has one small gear. A small gear has 512 MB memory and 250 threads. They also do not declare the equivalent CPU processing power. In addition, Cloud Foundry does not provide details about computing resources because it is still in beta and does not employ a resource versus cost plan. Cloud Foundry does declare the memory resource amount; however, they do not specify CPU power. Thus, direct comparison with other platforms is limited. They cannot be verified by the performance test; therefore, benchmarks are the only way to obtain performance figures.

All charts showing data are plotted with average data values with plus and minus standard deviations. Fig. 3 shows the test results of the Whetstone algorithm for CPU performance in MFOPS/MOPS. Fig. 4 shows the Whetstone algorithm test results relative to the time required to solve the algorithm's problems or equations. As can be seen, trigonometric and mathematical function calculations are the most time consuming functions for all three platforms. This may not be a

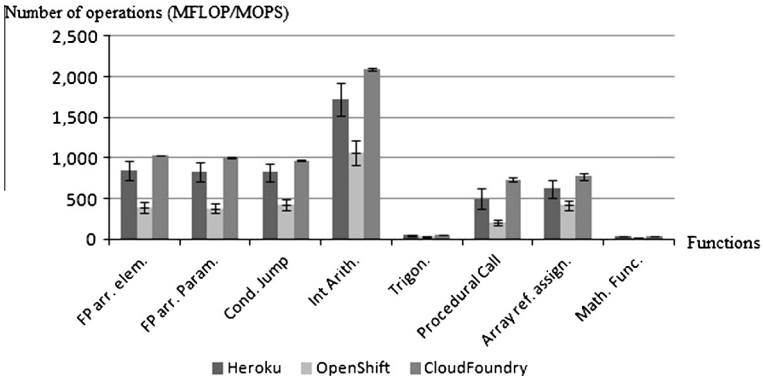

Fig. 3. Whetstone test results for MFLOP/MOPS.

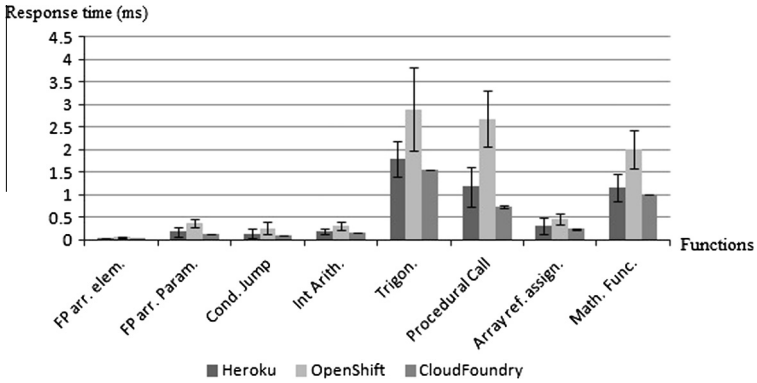

Fig. 4. Whetstone test results for response time.

problem for many applications because they may not require trigonometric, exponential, or square root functions. However, operation time for scientific and graphical applications will increase because these functions are required.

Procedural calls appear to require more time than floating-point operations, which is rational. This is almost inevitable for all applications; therefore, decreasing the number of procedural calls will increase application performance. This is a well-known issue, and it can be clearly seen in these test results. However, decreasing the number of procedural calls will degrade the application's structure and readability. Floating-point operations and fixed-point calculations are the best areas for the three platforms because they are the most common operations for any application.

For nearly all CPU sub functions, Cloud Foundry demonstrates the best performance. Heroku follows with a relatively small percentage difference. However, OpenShift's performance is inferior to the others. Generally, OpenShift performs 50 or more percent worse than Cloud Foundry and Heroku.

Fig. 5 shows the database operation times for queries that affect single records. OpenShift and Cloud Foundry perform nearly equally well; only a very slight difference can be seen. OpenShift performs better for approximately 0.7–0.8 ms for read and insert operations than Cloud Foundry. In addition, Cloud Foundry performs better for approximately 0.2–0.25 ms better than OpenShift for update and delete operations. However, the most obvious observation to be made from the chart is that Heroku performs very badly for single record database operations. Heroku requires approximately five to six times more time to perform single record queries than OpenShift and Cloud Foundry.

Similarly, Fig. 6 shows the operation times for queries that affect 100 records simultaneously. Again, OpenShift and Cloud Foundry perform similarly. However, Cloud Foundry provides the best performance in this sub category. Cloud Foundry is better than OpenShift in update, insert, and delete functions, requiring approximately 1.3–4.1 ms on average. However, OpenShift shows better performance with read operations, requiring approximately 2.4 ms on average. Again, Heroku demonstrates the worst performance for 100 record database queries. Heroku requires four to approximately ten times more time to complete the operations than the other two platforms. This is a significant problem for Heroku because database operations are very common. This may be due to database system implementation or network issues.

Memory bandwidth test results are shown in Fig. 7 for three different workload sizes. Heroku and Cloud Foundry perform similarly in this category. However, Heroku performs slightly better than Cloud Foundry for 1 MB and 100 MB workloads. Cloud Foundry performs better than Heroku for the 128 KB workload. Here, OpenShift demonstrates the worst performance

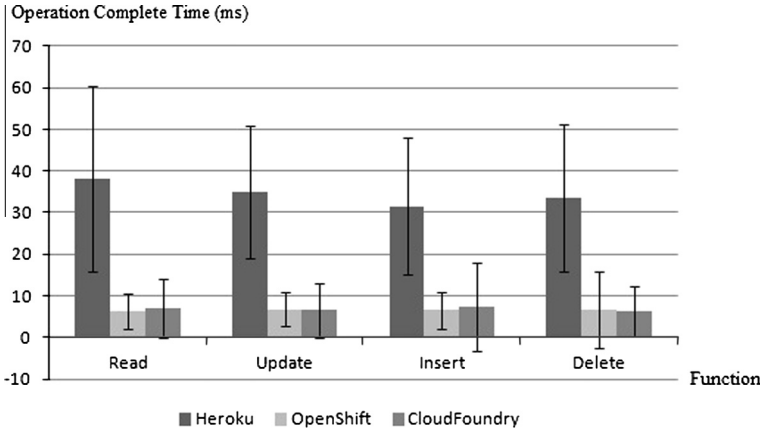

Fig. 5. Database test results for a single record.

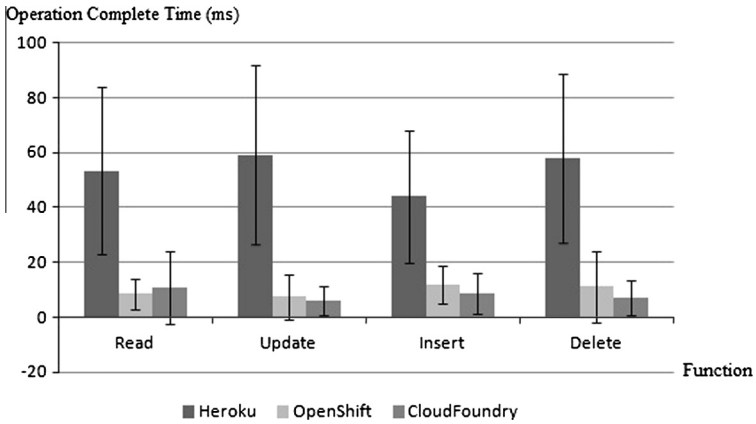

Fig. 6. Database test results for 100 records.

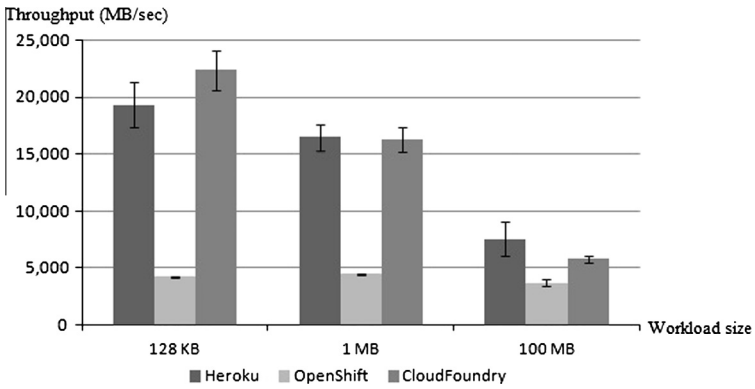

Fig. 7. Stream test results for memory bandwidth.

with remarkable differences. Cloud Foundry and Heroku perform 1.5–5 times better than OpenShift. This is an important problem for OpenShift because memory is a critical resource for nearly all applications.

As there are 19 functions categorized under three different main functions, it is difficult to evaluate which platform is more suitable for the specific needs of an application. Therefore, four different scenarios are used to compare evaluation results. The first is an enterprise application scenario that roughly represents the resource requirements of a typical business

application. The second scenario is a scientific or graphical application that represents high computation power and memory but minimal database operations. The third scenario represents an application that requires database-intensive operations, and the fourth scenario is an equally weighted function scenario that does not represent a real-world application. The fourth scenario is used to evaluate all functions and sub functions with equal importance. In Table 3, the four scenarios are denoted S1, S2, S3, and S4, respectively.

The weights used for each scenario are shown in Table 3. The weights are selected approximately to highlight more important resources considering the general properties of the applications in these scenarios. Table 4 shows the results of the evaluations with the AHP and LSP methods. Both methods indicate that Cloud Foundry demonstrates the best overall performance among the three platforms.

This is expected because Cloud Foundry ranks first or second in almost all categories. In the cases in which Cloud Foundry ranked second, there is only a small difference between the first and second rankings. However, the other two platforms demonstrate the worst performance in some categories, and in some of those categories, they show significant differences with the next closest ranking.

The score ranges from zero to one for both methods, where zero is the worst and one is the best achievable score. However, the AHP method uses a relative calculation; thus, it is considerably more difficult for a platform to obtain a score of one with the AHP method. For example, a platform must perform 100 times better than the other two alternatives to obtain a score of one with the AHP method. However, to obtain a score of one in the LSP method, a platform must only meet or exceed the maximum accepted value in the range.

From inspecting the stability of the results, we observe that the best performing platform is different for each main category. Cloud Foundry is the most stable platform with very little deviation in CPU test results. This deviation ranges from

Table 3
Weights used for the four scenarios.

Function	S1	S2	S3	S4	Function	S1	S2	S3	S4	Function	S1	S2	S3	S4
CPU	0.50	0.50	0.20	0.34	FP with array element	0.10	0.10	0.10	0.125					
					FP array as parameter	0.10	0.10	0.10	0.125					
					Conditional jumps	0.22	0.20	0.30	0.125					
					Integer arithmetic	0.20	0.10	0.10	0.125					
					Trigonometric function	0.05	0.20	0.05	0.125					
					Procedure call	0.22	0.15	0.20	0.125					
					Array reference assignment	0.10	0.05	0.10	0.125					
					Mathematical functions	0.01	0.10	0.05	0.125					
					Total weight for function	*1.00*	*1.00*	*1.00*	*1.00*					
DB	0.30	0.05	0.50	0.33	Single record	0.80	0.80	0.50	0.50	Read	0.40	0.20	0.25	0.25
										Update	0.20	0.30	0.30	0.25
										Insert	0.30	0.40	0.40	0.25
										Delete	0.10	0.10	0.05	0.25
										Total weight	*1.00*	*1.00*	*1.00*	*1.00*
					100 Records	0.20	0.20	0.50	0.50	Read	0.40	0.20	0.25	0.25
										Update	0.20	0.30	0.30	0.25
										Insert	0.30	0.40	0.40	0.25
										Delete	0.10	0.10	0.05	0.25
										Total Weight	*1.00*	*1.00*	*1.00*	*1.00*
					Total weight for function	*1.00*	*1.00*	*1.00*	*1.00*					
Memory	0.20	0.45	0.30	0.33	128 KB	0.84	0.10	0.30	0.34					
					1 MB	0.15	0.30	0.40	0.33					
					100 MB	0.01	0.60	0.30	0.33					
					Total weight for function	*1.00*	*1.00*	*1.00*	*1.00*					
Overall	1.00	1.00	1.00	1.00										

Table 4
Summary of AHP and LSP evaluation results.

Methods	Scenarios	Heroku	OpenShift	Cloud foundry
AHP	Enterprise application scenario	0.2904	0.2532 ✗	0.4565 ✓
		!		
	Scientific application scenario	0.3838 !	0.1993 ✗	0.4169 ✓
	Database-intensive scenario	0.2459 ✗	0.3034 !	0.4506 ✓
	Equal weighted scenario	0.2938 !	0.2597 ✗	0.4465 ✓
LSP	Enterprise application scenario	0.1062 ✗	0.1355 !	0.6998 ✓
	Scientific application scenario	0.4312 !	0.1817 ✗	0.6035 ✓
	Database-intensive scenario	0.2246 !	0.1695 ✗	0.7164 ✓
	Equal weighted scenario	0.1638 ✗	0.1642 !	0.6596 ✓

Best: ✓; Moderate: !; Worst: ✗; Min. Score: 0 – Max. Score: 1.

0.2% to 5.4% with an average of 1.5%. Heroku's deviation ranges from 11.5% to 25.8% with an average of 15%, and OpenShift's deviation ranges from 12.7% to 15.9% with an average of 14.5%.

For database operations, all platforms demonstrate high deviations. Heroku demonstrates the minimum deviations (57.5–71.5% with an average of 66.7%), followed by OpenShift (70.4–148.2% with an average of 99.4%). Cloud Foundry shows the worst deviations (90.9–163.7% with an average of 112.6%).

For memory performance, deviations are again low. The most stable platform is OpenShift (1.3–7.8% with an average of 3.5%), followed by Cloud Foundry (5.6–7.8% with an average of 6.7%). Here Heroku demonstrates the worst deviations (7.0–20.7% with an average of 12.7%).

Relative to cost, Cloud Foundry has not declared enterprise pricing. Heroku charges $0.05 per h for each dyno after the first free dyno. OpenShift charges $0.04 per small gear after the first three small gears. Thus, for the cases with only small differences in ranking, OpenShift may be considered preferable because it has a lower cost than Heroku. However, if there is no need to purchase a high number of dynos or gears, the price difference is not significant. In addition, the pricing of cloud services is variable because the cloud market is growing daily, and new alternatives are expected to emerge frequently.

The summary of the evaluations of the performance tests is as follows.

- The best performer is Cloud Foundry. The second platform changes according to the weights of the resources required. LSP ranks OpenShift second in two scenarios and third in two other scenarios. The AHP method ranks Heroku second in three scenarios and ranks OpenShift second in one scenario.

Stability in database operations is an important issue for all three platforms. OpenShift, Heroku, and Cloud Foundry all need to improve database operations significantly. Consumers will demand more stable operation times for database operations because such operations are critical for most applications.

6. Conclusions

Market studies show that enterprise cloud computing services will be in high demand in the near future. Thus, customers need to know which cloud computing solution performs well at specific functions. Thus, the evaluation and comparison of services must be based on common and most used or required resources. To address this need, this study has proposed the most critical functions and sub functions of PaaS solutions. In the proposed evaluation framework, a suitable set of benchmarking algorithms is also provided to determine the most suitable PaaS solution according to different application requirements. Performance evaluations of three different PaaS providers, i.e., Cloud Foundry, Heroku, and OpenShift, were conducted using the AHP and LSP methods relative to computing power, database operations, and memory bandwidth. Both AHP and LSP methods are widely used evaluation methods for different research areas. Both methods use weights to evaluate the effect of a tested function on the results. However, we believe that the LSP method's structure is more reliable because its weighted powers approach allows the construction of logical relationships among requirements for an examined system. In the AHP method, if an alternative has lower values in an important category with respect to other alternatives but has much higher values in another category, it can obtain higher scores. This means that, in terms of overall ranking, the higher ranking aspects of the platform can compensate the lower ranking aspects. This issue is decreased with the use of weights; however, this compensation may lead to ineffective decision making. All three platforms demonstrated stability in terms of CPU and memory operations. However, all three alternatives must improve stability of database operations. In future, we plan to perform performance evaluations of cloud computing platforms that employ parallel computing. Such a study will help reveal the dynamics of high-computing resource utilization in PaaS solutions, as well as how these platforms perform under high volumes of processing requirements.

References

[1] Furht B. Escalante A. Handbook of cloud computing. Boston, USA: Springer US; 2010. p. 22–3.
[2] Du L. Pricing and resource allocation in a cloud computing market. In: 12th IEEE/ACM International symposium on cluster. Cloud and Grid Computing; 2012.
[3] Kundra V. 25 Point implementation plan to reform federal information technology management. Washington: The White House; December 9, 2010.
[4] Maluf DA, Shetye SD, Chilukuri S, Sturken I. Lost in cloud. Aerospace Conference, 2012 IEEE; 2012.
[5] Murugesan S. Cloud computing gives emerging markets a lift. IT Professional (Volume: 13, Issue: 6); 2011.
[6] Buyya R, Yeo CS, Venugopal S. Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities. In: 10th IEEE international conference on high performance computing and communications; 2008.
[7] Peng J, Zhang X, Lei Z, Zhang B, Zhang W, Li Q. Comparison of several cloud computing platforms. In: 2nd International symposium on information science and engineering. Shanghai, Hong Kong; December 26–28, 2009. pp. 23–7.
[8] Sempolinski P, Thain D. A comparison and critique of eucalyptus, opennebula and nimbus. In: 2nd IEEE international conference on cloud computing technology and science, Indianapolis, USA; November 30, 2010 to December 3, 2010. p. 417–6.
[9] Wind S. Open source cloud computing management platforms: introduction, comparison, and recommendations for implementation. 2011 IEEE Conference on Open Systems (ICOS2011), Langkawi, Malaysia, September 25–28, 2011. p. 175–9.
[10] Iosup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema DHJ. Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distrib Syst 2011;22(6):931–45.
[11] Salah K, Al-Saba M, Akhdhor M, Shaaban O, Buhari MI. Performance evaluation of popular cloud IaaS providers. In: 6th International conference on internet technology and secured transactions, Abu Dhabi, United Arab Emirates, December 11–14, 2011. p. 345–9.

[12] Tudoran R, Costan A, Antoniu G, Bougé L. A performance evaluation of azure and nimbus clouds for scientific applications. In: 2nd International workshop on cloud computing platforms – CloudCP '12, New York, USA; 2012. p. 1–6.
[13] de Costa PJP, de Cruz AMR. Migration to windows azure – analysis and comparison. In: 4th Conference of enterprise information systems – aligning technology, organizations and people (CENTERIS 2012), Procedia Technology, vol. 5; 2012. p. 93–102.
[14] Binnig C, Kossmann D, Kraska T, Loesing S. How is the weather tomorrow? Towards a benchmark for the cloud. In: 2nd International workshop on testing database systems, DBTest, New York, USA; 29 June 2009.
[15] Cloud Services Measurement Initiative Consortium (CSMIC). Service measurement index. Carnegie Mellon University Silicon Valley, California, USA; 2011.
[16] Voras I, Mihaljevic B, Orlic M. Criteria for evaluation of open source cloud computing solutions. In: 33rd International conference on information technology interfaces, Dubrovnik, Croatia, June 27–30, 2011. p. 137–42.
[17] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. Above the clouds: a Berkeley view of cloud computing. Electrical Engineering and Computer Sciences Technical Report, University of California; 2009.
[18] Li A, Yang X, Kandula S, Zhang M. CloudCmp: Comparing Public Cloud Providers. In Internet Measurement Conference; 2010. p. 1–14.
[19] Mao M, Humphrey M. A performance study on the VM startup time in the cloud. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD); 2012.
[20] Roloff E, Diener M, Carissimi A, Navaux POA. High performance computing in the cloud: deployment, performance and cost efficiency. IEEE 4th International conference on cloud computing technology and science (CloudCom); 2012.
[21] Garg SK, Versteeg S, Buyya R. A framework for ranking of cloud computing services. Future Gener Comput Syst 2013;29(4):1012–23.
[22] Godse M, Mulik S. An approach for selecting software-as-a-service (SaaS) product. In: IEEE International conference on cloud computing; 2009.
[23] Yu HQ, Molina H. A modified logic scoring preference method for dynamic web services evaluation and selection. The 2nd European Young Researchers Workshop On Service Oriented Computing. University of Leicester, United Kingdom; June 11–12, 2007.
[24] Yu HQ, Reiff-Marganiec S. A method for automated web service selection. IEEE Congress on Services 2008, 6–11 July 2008, Hawaii, USA; 2008.
[25] Maheswari S, Karpagam GR. Applying logical scoring preference method for semantic web service selection. Int J Comput Appl 2013;65(19):38–46.
[26] Longbottom R. Roy Longbottom's PC benchmark collection [online]. <http://www.roylongbottom.org.uk> [accessed 13 January 2013].
[27] McCalphin J. The Stream benchmark java code [online]. <http://www.cs.virginia.edu/stream/FTP/Contrib/Java/STREAM.java> [accessed 13 January 2013].
[28] Saaty TL. The analytic hierarchy process. New York, USA: McGraw-Hill; 1980.
[29] Saaty RW. The analytic hierarchy process—what it is and how it is used. Math Model 1987;9(3–5):161–76.
[30] Tran VX, Tsuji H, Masuda R. A new QoS ontology and its QoS-based ranking algorithm for web services. Simul Model Pract Theory 2009;17(8):1378–98.
[31] Dujmovic JJ. A method for evaluation and selection of complex hardware and software systems. In: The 22nd international conference for the resource management and performance evaluation of enterprise computing systems, San Diego, USA, CMG 96 Proceedings 1, December 10–13; 1996. p. 368–78.

Gültekin Ataş received a BSc in Electric and Electronic Engineering from Boğaziçi University, Istanbul, Turkey in 2006 and an MSc in Computer Engineering from Bahçeşehir University, Istanbul, Turkey in 2013. His research interests include software engineering and cloud computing platforms. He also works as a software analyst at Softtech A.S., Istanbul, Turkey.

Vehbi Çağrı Güngör received a Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, USA in 2007 after working at the Broadband and Wireless Networking Laboratory. Currently, he is an Associate Professor and Department Chair of Computer Engineering at Abdullah Gül University, Kayseri, Turkey. His current research interests are smart grid communications, cloud computing, next-generation wireless networks, wireless ad hoc and sensor networks, cognitive radio networks, and IP networks.