Performance Study of Cloud Computing Back-end Solutions For Mobile Applications

Guilherme Macedo*, Christina Thorpe**
School of Computer Science
University College Dublin
Email: guimecps@gmail.com* christina.thorpe@ucd.ie**

***Abstract*—Cloud Computing provides essential tools for building modern mobile applications. In order to leverage the advantages of the Cloud for developing and scaling applications, mobile developers must perform a technical analysis of the options currently available on the market. The objective of this paper is to investigate the various considerations of hosting mobile applications' back-end in the Cloud, more specifically, the ease of deployment and the application performance. We conducted a comprehensive performance analysis of three popular Platform-as-a-Service providers. Results show that there are important differences in the performance and other aspects of deployment that should be considered by mobile application developers.**

***Keywords*—Cloud Computing, Performance, Software**

I. Introduction

For many software developers working on mobile applications, the features and resources available on mobile devices are not sufficient to provide the level of Quality of Experience (QoE) users have come to expect. Although, the current top of the market mobile devices come with high processing power (multiple core processors), large storage and RAM, mobile developers must take advantage of the high-speed data connections available and look for alternatives on how to provide a richer experience for their applications.

Cloud Computing, a concept that can be traced back as far as the 1960s [1], has evolved greatly over the past fifty years to its current form that provides a combination of service models such as: Software-as-a-Service (SaaS), Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS). PaaS in its nature was created to offer businesses and independent developers the flexibility to have a robust, reliable and scalable platform to deploy and run software applications without the usual necessary system administrator skills or a high-risk upfront monetary investment. However, there are multiple companies offering PaaS solutions on the market, ranging from starting plans free of cost to expensive enterprise plans. Therefore, it can be daunting for software developers to understand what is in fact offered for free, what are the options when making investments and which service offers the best performance.

This study aims to provide a valuable resource for software developers working on an internet service delivered through a mobile application, by providing a comprehensive performance comparison between three of the available PaaS solutions on the market. The related work in the literature have important differences when compared to the work in this paper. They either compare the performance analysis of only IaaS solutions [2] or mixing IaaS with PaaS solutions [3]. The performance comparison presented in this paper compares only PaaS solutions and it is focused on how the platforms perform when running a simple back-end of an e-commerce application. The three PaaS solutions included in the study were: Heroku [4], Red Hat OpenShift [5] and Amazon Web Services Elastic Beanstalk [6]. A sample mobile application back-end was developed and deployed on the selected platforms.

A number of test cases were defined to compare the performance of the providers. Automation scripts were developed for running the tests and collecting the metrics from each run. The automation was a key part of the work since the tests were performed multiple times during the day and night, taking into consideration a possible difference in load during peak and off-peak times in a multi-tenant environment.

The remainder of the paper is organised as follows: Section II provides details about the various Cloud solutions and motivates this work. Section III presents a review of related studies previously published in the literature. Section IV describes the sample mobile application developed for this research. Section V presents and discusses the results of the performance comparison. Finally, Section VI concludes the paper.

II. Cloud Solutions

Cloud Computing, has evolved greatly since its inception to its current model that provides a combination of IT services implemented in a scalable architecture. Software applications can be deployed, have their data stored and be accessed via the Internet. The main difference between the Cloud computing model and the traditional computing model is that Cloud Computing is sold on-demand (pay-per-use basis) and the service is managed by the provider [7].

There are different IT service models in Cloud Computing, originally implemented to allow a wide range of service delivery models. The three main models are:

1) Software-as-a-Service (SaaS): Provides an application to be used on-demand by the users. This service delivery model includes infrastructure and the software product. Users are charged according to software licenses that vary providing different features of the product, as well as data storage used by the application (e.g., Salesforce.com, GMail, and Facebook).

2) Platform-as-a-Service (PaaS): Provides hardware and software tools required for application development. As both

hardware and software are hosted in the providers infrastructure, users are free to focus soley on the application development (e.g., Heroku, RedHat OpenShift, AWS Elastic Beanstalk).

3) Infrastructure-as-a-Service (IaaS): Provides users with highly automated computer resources complemented by storage and networking capabilities. Those resources are owned and hosted by the provider and offered to users on-demand (e.g., Amazon Web Services (AWS), and Google Compute Engine).

This study focuses on three different commercial PaaS solutions: Heroku, Red Hat OpenShift, and Amazon Web Services Elastic Beanstalk. All the three solutions were selected because they are owned by reliable companies, easy to use, and they offer an initial level of service free of cost. Figure 1 illustrates a comparison on what is offered free of cost by the three studied Cloud providers:

Name	CPU	RAM [MB]	Archi. [bit]	Disk [GB]
Heroku (1X Dyno)	1	512	64	0.3
OpenShift (Basic Gear, Small) *	1	512	64	1
Elastic Beanstalk (free tier) **	1	1024	32/64	5

Fig. 1. Available Resource Characteristics for the Free Instance Types Offered by the Three Studied Providers

The notes in the table denote that OpenShift (*) offers three of such instances on the free plan. Elastic Beanstalk (**) offers 750 hours of the instance for free over a limited period of twelve months.

A. Cloud Computing for Mobile Application Development

Although there are several reasons why Cloud Computing has become a valuable resource for mobile applications, one of the key reasons is the broad network access. That is, the combination of the Internet, advanced wireless technologies, and electronics, mobile devices are capable of being completely connected and constantly available [8].

The importance of Cloud Computing for the development of mobile applications has been studied and discussed for some time. In 2010, an ABI report noted that the number of Mobile Cloud Computing subscribers in 2008 was 42.8 million and also predicted that by 2015, more than 240 million business customers will be leveraging Cloud computing services through mobile devices, driving revenues of $5.2 billion [9]. This reinforces the importance of Cloud Computing for mobile applications progress.

Mobile applications that are driven by the use of Cloud Computing can be categorised as in Figure 2, which presents a summary of the different categories as well as advantages and challenges that are important to be aware of.

III. Related Work

A large body of work can be found in the areas of mobile applications/services development using Cloud computing. However, these publications typically do not include a performance comparison. They tend to focus on how Cloud computing can be applied to increase mobile applications' capabilities as well as listing the advantages identified [10], [11], [12].

	Mobile Cloud Storage	Audio/Video Streaming	Interactive Services	Cloud based Rendering	Media Analytics
Sample applications /services enabled on mobile devices	Storing and accessing Photos, Music, Files	Streaming audio, video; Cloud DVR	Video Chat; Remote Desktop; Interactive advertisements	Mobile Gaming; Augmented Reality; Telemedicine	Differentiated Services/Billing; Personalized services
IaaS, PaaS features needed	Cloud Storage with high availability and integrity	Cloud Transcoding, Transrating, Caching	Cloud Transcoding, Transrating	Multi-core GPUs; Efficient cloud rendering;	Cloud media usage/QoE probes; Media classification engines
Advantages	Ubiquitous access from any device; Synchronizing between devices	Low capex; High scalability with demand	Easier support for multiple devices/platforms	Enables highest quality rendering; Multi-player, multi-platform	Unified analytics for media usage across devices and networks
Challenges	Ensuring content security, privacy; Additional wireless traffic	Cloud service cost; Cloud energy, cooling costs	Response time; Video quality; Cloud service cost; Cloud energy, cooling costs	Response time; User experience; Cloud service cost; Additional wireless traffic; Cloud energy, cooling costs	Data protection; privacy

Fig. 2. Analysis of different categories of potential Cloud Mobile Media applications, including cloud capabilities needed, advantages derived by having the applications based on the cloud, and challenges to make the applications successful [10]

Wang et. al. [10] focus on Cloud Mobile Media (CMM) applications and services; they discuss important points on how Cloud Computing is valuable for the Mobile Gaming industry. Christensen [8] highlights clearly how the capabilities of a mobile application can be extended with the Cloud. [8] was used as a reference when designing the architecture and technologies to include in the back-end of the e-commerce application used in this study.

A more limited number of publications focus on comparing Cloud computing platform providers. While some present a performance comparison [2] [3], others simply compare the cloud platforms architecture and features available [13]. Although the study published by Iosup et. al. [2] was very inspirational due to the high level of details provided from the performance analysis performed, it focused on IaaS services instead of PaaS. It also uses a Many-Tasks Scientific Computing application instead of a commercial application. It is reasonable for their study of Many-Tasks Computing (MTC) application to use IaaS since this category of service offers more flexibility on scaling and achieving higher processing power.

In the area of performance analysis of PaaS services, Li et. al. [3] was extremely valuable as Microsoft Azure and Google AppEngine were included, however the comparison focuses on IaaS. The work in this paper is different to what is published in the literature as it is focused only on PaaS services and on testing an e-commerce web service application. The results of this study are targeted towards either researchers, independent developers, or small start-up companies with limited or no resources to invest in new infrastructure.

IV. Case Study: Performance Comparison of Three PaaS Providers

This section describes the development of a complete and working Java back-end web service application, with the use of Java Enterprise Edition 7, JAX-RS with Jerseys implementation and PostgreSQL database technologies. Additionally, Apache Maven and Git were used during development for build automation and version control, respectively.During the testing stage, performance tests were executed from JMeter and the applications metrics were collected with New Relic.

A. Application Features

The developed application represents the back-end of an example e-commerce application, called Macedo Store. As a

typical e-commerce application, it allows users to perform the following actions which were later used as workloads in the performance tests:

- Access the index page/screen of the service;
- Search for a specific product;
- List all products registered on the store;

In addition, the following actions were also implemented but not used on the tests due to being Administrator level only:

- Insert a new product;
- Delete an existing product;
- Insert a new category;
- Delete an existing category;
- Insert a new location;
- Delete an existing location;

The Macedo Store back-end application was deployed in free accounts of three different PaaS providers. The free account option was selected based on the target of this study: researchers, independent software developers, and small start-ups with an interest in releasing an innovative service without any investment. However, there is an option to invest and easily scale the application in case it succeeds. One objective of this study is to prove that PaaS can be used as an easy, powerful and reliable way of increasing the processing power of mobile applications. The e-commerce application selected is considered somewhat representative, but the point is that any application that requires either heavy processing or a large database (storage) would benefit of having a back-end on the cloud. Once the back-end studied has a large database and an application running on an application server, that process and executes business logic before querying the database and returning the right data to the client app, could be used and the result would be similar.

B. Application Development

In order to enable a set of performance tests to be executed against the different PaaS providers selected for this study, a simplified version of the back-end of an e-commerce application was developed using the Java Enterprise Edition 7 and Figure 3 represents a high level diagram of the application.

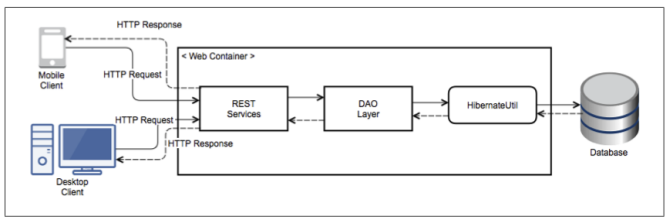

Fig. 3. High Level Diagram of Back-end Application

Since the objective of this study is to provide information on how Cloud computing can help software developers to easily create reliable mobile applications or services, the developed back-end application was developed using RESTful Web Services that are known for working well in a mobile device environment [8]. Since it is included in the Java Enterprise Edition 7, the Java API for RESTful Services (JAX-RS) was used to develop the RESTful Web Services available in the application.

For data persistence, Hibernate was selected in order to keep the development fully in Java, without the need to write SQL codes to create the database model and operations performed by the application. The back-end application was developed using the Data Access Object design pattern, achieving a separation between the low level of data accessing operations and the high level business services.

C. Application Deployment

For a Java application to be deployed on Heroku, it must use Maven as the build tool. This is required because Maven is used on the Heroku side to automatically build the application after its code has been checked-in via Git.

Similarly to Heroku, every OpenShift application is created with a Git repository, which only the user who created the application has access. If the OpenShift Web Console is used to create an application, it is necessary to retrieve and clone the Git URL from the page. In case of applications created using the rhc command line tool, the step above would not be required as a copy of the repository is automatically downloaded. After copying all the files in the new directory, it is necessary to add everything into Git using the add and commit commands. Once the commit has been completed, in order to deploy the application and automatically start a build on the OpenShift platform, the git push command is needed.

Although many Java developers may not have previous experience using Git and to learn how to use it can be time-consuming, the Eclipse IDE is a standard on the Java development industry. Taking advantage of this fact, AWS Elastic Beanstalk offers a much simpler way of deploying Java applications on its platform. Once the developer has completed the steps to setup the AWS Toolkit for Eclipse, the user can add AWS Elastic Beanstalk as a new server in Eclipse. By following this process in the IDE, every time the Run On Server action is performed for the Java application will be automatically deployed on the Elastic Beanstalk platform.

D. Performance Tests

A set of performance tests were defined and executed against the same Java back-end application deployed and running on the three different PaaS selected for this study. Results can be used as a decision support for developers evaluating whether Cloud Computing should be considered for their application. In addition, the published results have potential to help in choosing the most suitable platform for particular use cases.

Since the tested application is an e-commerce back-end exposed through a set of RESTful Web Services, the test cases were separated by the following actions:

- **Test Case 1** : Access the index page of the service;
- **Test Case 2** : Search for a specific product;
- **Test Case 3** : List all products in the store;
- **Test Case 4** : Random sequence of test cases 1, 2 and 3, including a thinking pause of two seconds.

Test case 1 - 4 were executed individually, for a duration of 5 minutes with a 1 minute of ramp-up period, against each PaaS for a wide range of users as described below:

- Tests 1, 2 and 3 were tested ranging from 80 to 600 users;
- Test 4 was tested ranging from 800 to 1400 users.

The described range of users for the tests was selected after initial testing using a wide number of combinations to find the limits of the free plans from each platform. It is important to note that in order to obtain meaningful results according to the objectives of the study, the application's database was populated with over 200 *Products*, belonging to different *Locations* and *Categories*. This was sufficient to emulate an e-commerce application in an initial stage, suitable for the level of service provided by the PaaS free plans.

1) Considerations: When executing performance tests on applications running on a PaaS solution, it is important to consider that due to the nature of PaaS services, applications run in a multi-tenant environment. Therefore the following points need to be considered:

- The developer has no control over the infrastructure resources used to run the application;
- The infrastructure is shared among multiple customers from which activity on the platform is unpredictable;
- For realistic results in a PaaS environment, tests need to be repeated ideally in a different time (e.g. peak and off-peak hours).

New Relic is a powerful performance managed SaaS-based application that is available as an add-on for Heroku and OpenShift. It was used to gather the metrics when testing the three platforms; the Rest API was used to retrieve raw data from the server, enabling a comprehensive analysis and comparison between the obtained results for each PaaS.

Lastly, the hardware specifications of the machine used to perform the tests as well as its available connectivity are shown in Figure 4:

Machine	MacBook Pro (Retina, 15-inch, Mid 2014)
Operating System	OS X Yosemite (10.10.2)
Processor	2.8 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Storage	500GB Flash (SSD) Storage
Internet connection	60Mb Cable Broadband

Fig. 4. Test Environment Specification

2) Test Methodology: Considering the high volume of tests needed (e.g. high number of five minutes test repetitions for four different test cases, repeated at different times), a Shell script automating the execution of the defined test suite was developed. The developed script automated the following sequence of actions:

- Application restart;
- HTTP request to the Web Services which lists all the available locations (implemented to guarantee that the application is on a Steady state before commencing the tests);
- Capture of Start Time of the tests (required when calling the New Relic API to download the raw logs);
- Customization of JMeter test case (.JMX) file applying the correct number of users for the given test; Capture of End Time of the tests;
- Make an HTTP request to the New Relic API passing the following parameters: Application Number, API Key, a list of names from metric to be downloaded, start and end time.

When executing the tests, each iteration would result in the following list of files: JMeter result log files (.JTL); JSON file containing raw logs retrieved from New Relic.

V. Results and Analysis

Figure 5 illustrates a summary of the performance tests executed against the free account of Heroku (CP1), OpenShift (CP2) and AWS Elastic Beanstalk (CP3):

		CP1			**CP2**			**CP3**	
	Min	Max	Average	Min	Max	Average	Min	Max	Average
Test Case 1									
Condition 1	700	780	740	480	600	540	560	600	580
Condition 2	860	1200	1030	920	1040	980	1140	1220	1180
Test Case 2									
Condition 1	260	280	270	320	340	330	460	500	480
Condition 2	380	460	420	560	580	570	580	600	590
Test Case 3									
Condition 1	280	340	310	160	220	190	260	500	380
Condition 2	360	420	390	360	400	380	460	780	620
Test Case 4									
Condition 1	1250	1300	1275	1250	1300	1275	850	900	875
Condition 2	1400	1450	1425	1600	1700	1650	1300	1450	1375

Fig. 5. Number of Users/sec before Consistent Errors

When analysing the results presented in Figure 5, the minimum, the maximum and the average number of users per second at the initial point of each of the conditions are identified and described as following:

- **Condition 1** : represents a condition which a small, but consistent, error rate starts being identified during tests execution.
- **Condition 2** : represents a condition which a high and consistent error rate starts occurring during testing. In this study, an error rate of 5% or above 5% was included in this condition.

Considering an e-commerce application, such figures provide an initial idea of how different each platform responded to a high load of users, which of them have a more stable performance and what is the maximum number of users the application can support before requiring a scale-up.

A. Analysis

Figures 6 and 7 show how performance metrics such as throughput and response time are related to the number of users during each test case execution. Values shown are an average (summary data from the New Relic API) of multiple execution for a particular time range [1]

[1]https://docs.newrelic.com/docs/apis/rest-api-v2/requirements/calculating-average-metric-values-summarize

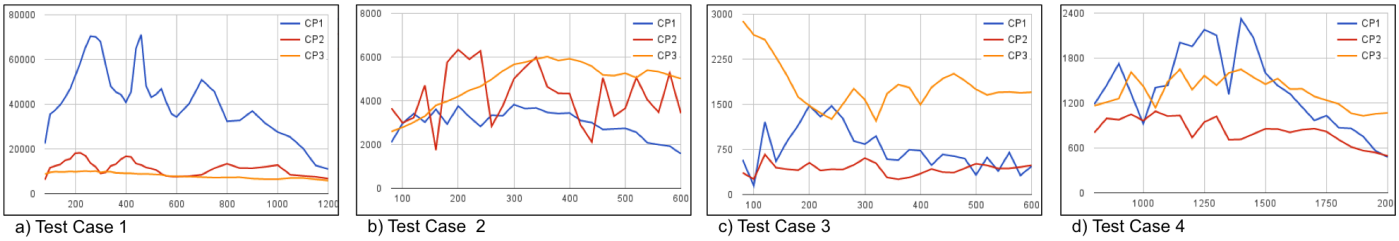

Fig. 6. Throughput Results for Test Case 1 - 4. X axis represents number of users. Y axis represents number of requests/min

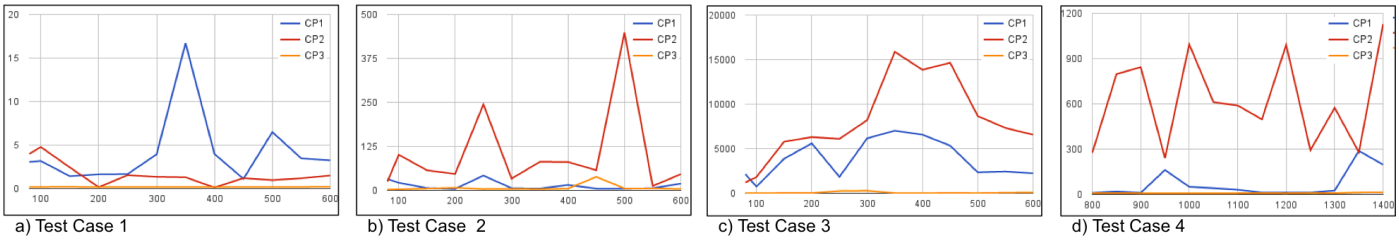

Fig. 7. Response Time Results for Test Case 1 - 4. X axis represents number of users. Y axis represents response time (ms)

1) Test Case 1: Test Case 1 tests how each cloud platform behaves under a heavy load accessing its index page, without any database calls being made. The graph represented by Figure 6(a) graphs the throughput for each Cloud platform during the execution of the Index test case. It can be observed that Heroku presents a much higher throughput, although OpenShift and Elastic Beanstalk are more stable during the test. In addition, as the number of users increases all platforms begin to achieve a similar throughput.

Analysing the response time obtained from Test Case 1, presented by Figure 7(a), it can be observed that Elastic Beanstalk (CP3) managed the load consistently, which would result in a more responsive mobile application. Although Heroku (CP1) presented, during most of the test case, a much higher throughput, Figure 7(a) shows that it delivered the highest response time between the three platforms, including two separate peaks that can also be related to the drop in throughput seen in Figure 6(a).

2) Test Case 2: This test case was created to test the performance of each platform for a high range of users performing a search in the application. The search used on the test case was the *Product* search, which resulted in complex database queries being executed. The complexity is due to the relationship bwteen the *Product* table with different tables in the database (e.g., *Category* and *Location*). An important point is that in this test case, random searches were formed and executed instead of running the same search multiple times.

The results presented in Figure 6(b) highlight the superior performance of Elastic Beanstalk against the other two platforms. Elastic Beanstalk not only delivered a higher throughput but also performed in a more stable and consistent way when comparing with the results from Heroku and OpenShift. The results suggest that there is a considerable overhead when using both Heroku Postgres and PostgreSQL running in OpenShift compared to running it directly in the Amazon AWS infrastructure, which is the case for Elastic Beanstalk.

The response time observed from Test Case 2 (Figure 7(b)) confirm the overall better performance of Elastic Beanstalk, although Heroku also presented stability and similar results for a large part of the execution. Matching with non-stable throughput delivered by OpenShift, response time for this platform was identified as mostly higher (at some points more than 1000% higher).

3) Test Case 3: This test case is technically the most difficult between all the tests; it consists of multiple users listing all the products registered on the application. During this test, Amazon's platform had a more consistent and efficient performance. As illustrated in Figure 6(c) and Figure 7(c), both throughput and response time from Elastic Beanstalk are significantly better than the other two platforms. The response time results for Amazon's platform range from 26.1ms to 98.8ms.

4) Test Case 4: This test case represents a realistic high number of users performing random actions in the application. For this test case, a thinking pause of two seconds was included between requests and the three previous test cases were added to the list of possible actions.

The response time results show that OpenShift presents relatively higher numbers. This may be due to the fact that the tests were performed from Ireland and OpenShifts small gears, available in the free plan, cannot run from Europe. However, both Heroku and Elastic Beanstalk offer such option to users from any plan. Representing a possible high load, Test Case 4 is the most suitable for use by developers when looking for the point in which scaling the application would become a necessity. All three platforms perform reasonably well until approximately the limit between 1300 and 1400 concurrent users; after this range it can be observed in Figure 6(d) that the performance starts to degrade.

These results confirm that PaaS solutions can be used as a quick and powerful way of enabling further capabilities and richer experience in mobile applications without requiring an initial investment. However, in case the developed application succeeds in the market, the options to scale the application

supporting a higher load is available without requiring any technical knowledge of the hardware and software involved with the scalability.

The data provided in Figure 5 indicates that the free plans of both Heroku and OpenShift are capable of handling an average of 1275 concurrent users performing random tasks in the application, with a very low error rate (below 5% as described for Condition 1). Although Elastic Beanstalk provided an overall more stable performance and lower response time, during the tests performed for the setting, this platform was able to handle an average of 875 concurrent users before an error rate higher than 5% started being observed. It is important to highlight that the objective of the testing was to compare the performance of the three platforms in the same setting instead of looking for the absolute best performance, applying tuning techniques.

B. Summary

During this study, we investigated multiple resources needed by mobile developers considering to use a commercial PaaS solution to either enrich or expand a mobile application. In order to provide a technical insight on how each platform performs, a comprehensive performance test was executed on the free plan of three important commercial PaaS solutions currently available. Since the back-end application used simulates a real life e-commerce application, the presented results have a high potential for helping developers with the decision of whether using a PaaS solution would be the right option, in addition to which PaaS would be the most suitable.

The performance tests results indicate a overall better performance of Elastic Beanstalk. This can be initially identified by the values presented in Figure 5 and confirmed by analysing the graphs presented in Figures 6 and 7 with the metrics from each test case. Since this study focused on the free plan of three PaaS solutions, it is suggested as an extension to repeat the study for the following two additional scenarios: (i) Conduct further performance testing with a compatible paid version of each platform. (ii) Test the same application on additional PaaS solutions. By completing the two scenarios suggested, developers would have all the information and data necessary for the main phases of a new mobile applications project: planning, implementation and expansion.

VI. Conclusion

Developing mobile applications to leverage the power of Cloud computing is a complex task for any software developer. The complexity of Cloud computing and the high number of solutions available on the market may be discouraging. From the research carried out in this study, particularly after performing the practical case study, it is clear that PaaS is the Cloud Computing service most suitable for mobile developers: even without previous experience with such services nor with system administration skills, it is possible for a developer to have an application deployed and running very quickly.

Between the three PaaS solutions studied, Heroku and Red Hat OpenShift provide a comparable experience on the development and deployment stages of the project. Elastic Beanstalk has an advantage on deploying the application, due to its powerful Eclipse plug-in. On the performance side, Elastic Beanstalk is positioned ahead of Heroku and OpenShift. However, the better performance may not be reasonable due to two considerable disadvantages: developers are directly exposed to the IaaS level of AWS and its free plan is currently valid only for a period of 12 months.

Although some individuals may consider the direct contact with the service's infrastructure an advantage, it may be a disadvantage when there is a lack of administrator skills. Regarding the AWS free tier, besides the 12 months limit, it was identified as confusing to understand how the infrastructure utilisation is calculated. Red Hat OpenShift has the advantage of being open source, however for European developers the fact that it does not let free plan users run applications from Europe may be a disadvantage as the tests identified a consistently higher response time for this platform. With a better performance and higher number of available add-ons on its marketplace, Heroku can be considered the most suitable platform for the target audience of this study.

Acknowledgment

This work was supported, in part, by Science Foundation Ireland grant 10CEI1855 and in part by Science Foundation Ireland grant 13RC2094.

References

[1] L. M. Kaufman, "Data security in the world of cloud computing," *Security & Privacy, IEEE*, vol. 7, no. 4, pp. 61–64, 2009.

[2] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 931–945, 2011.

[3] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 1–14.

[4] "Inc. heroku cloud application platform," http://www.heroku.com, accessed: 2015-04-25.

[5] "Inc. red hat. openshift online marketplace," https://marketplace.openshift.com/home, accessed: 2015-04-25.

[6] "Amazon aws elastic beanstalk - application management - paas," http://aws.amazon.com/elasticbeanstalk/, accessed: 2015-04-25.

[7] W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing," *Cloud computing: Principles and paradigms*, pp. 1–44, 2011.

[8] J. H. Christensen, "Using restful web-services and cloud computing to create next generation mobile applications," in *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, 2009, pp. 627–634.

[9] "Abi research: Enterprise mobile cloud computing," https://www.abiresearch.com/market-research/product/1004607-enterprise-mobile-cloud-computing/, accessed: 2015-04-25.

[10] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *Multimedia, IEEE Transactions on*, vol. 15, no. 4, pp. 870–883, 2013.

[11] D. Kovachev, Y. Cao, and R. Klamma, "Building mobile multimedia services: a hybrid cloud computing approach," *Multimedia tools and applications*, vol. 70, no. 2, pp. 977–1005, 2014.

[12] H. Cho and M. Choi, "Personal mobile album/diary application development," *Journal of Convergence*, vol. 5, no. 1, 2014.

[13] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, "Comparison of several cloud computing platforms," in *Information Science and Engineering (ISISE), 2009 Second International Symposium on*. IEEE, 2009, pp. 23–27.