ABOUT USING MOBILE DEVICES AS CLOUD SERVICE PROVIDERS

Marc Jansen
University of Applied Sciences Ruhr West, Computer Science Institute, Bottrop, Germany

Keywords: Cloud Computing, Mobile Devices, Platform-as-a-Service.

Abstract: In recent years, the number of reasonable powerful mobile devices increased. In 2011, the number of smartphone (e.g.) increased to more than 300 million units. A lot of research has already been conducted with respect of mobile devices acting as Cloud Service consumers, but still not much effort is put on mobile devices in the role of Cloud Service providers. Therefore, this paper presents an approach that allows to utilize mobile devices like smart phones or tablets as Cloud Service providers. In order to make this a reasonable approach, some of the occurring problems are discussed and it is shown how the presented architecture is able to overcome these problems. Last but not least, this paper describes some performance tests of the chosen implementation for mobile Web Services.

1 INTRODUCTION

As the number of reasonable powerful mobile devices increased a lot in recent years (e.g. according to (IDC, 2011) the number of smartphones increased to more than 300 million units in 2011), the usage of these kinds of devices becomes more and more interesting in various scenarios. Also, with respect to Cloud Computing scenarios there is a lot of research published (e.g. Manjunatha et al., 2010) that uses mobile devices as consumers of services offered in the Cloud. Still, there is not much work to be found when it comes to mobile devices acting as Cloud Service providers.

Nevertheless, one of the most serious problems nowadays with software development for mobile devices is the heterogenity of devices that are available on the market. With respect to the work presented in this paper, the most stressing problem of heterogenity is the number of different operating systems for mobile devices. According to (Tudor, Pettey, 2010), there were at least five different operating systems for smartphones available on the market in 2010. Furthermore, not only the operating systems on different mobile devices differ, but also the complete development process (starting with different programming languages) differ dramatically from one to the other device.

Therefore, this paper presents an approach that allows to deploy Cloud like services on mobile devices like cell phones or tablets.

Of course, beside the problem of the heterogenity of devices, a number of other problems also arise when Cloud Services are deployed on mobile devices. Here, this paper also discusses how problems that usually occur if Cloud Services are to be deployed on mobile devices, can be solved and how the presented architecture supports the solution of these kinds of problems.

Since the approach for the implementation of mobile Web Services chosen for the example implementation is special with respect to its polling mechanism, some performance tests for this approach are presented in this paper also.

Nevertheless, this paper only describes a technical approach and does not consider security related problems that might come into play, if a certain service should be run on a mobile device, where the owner of the devices is not aware of each and every service running on his devices.

2 STATE OF ART

As already mentioned there is already some research around the topic of mobile devices acting as clients in Cloud Computing scenarios. For example

Jansen M..
ABOUT USING MOBILE DEVICES AS CLOUD SERVICE PROVIDERS.
DOI: 10.5220/0003961401470152
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 147-152
ISBN: 978-989-8565-05-1

(Manjunatha et al., 2010) describe an approach based on a domain specific language (Deursen, Kling, Visser, 2000) that allows the development of a so called cloud-mobile-hybrid application. Basically, in this kind of applications, the core functionality is provided in a Cloud Computing scenario, whereas a small and tiny client application makes use of this Cloud Service to allow a mobile consumer to use the Cloud Service from a mobile device.

In (Mishra, Elespuru, Shakaya, 2009) the authors describe a mobile MapReduce system that allows to solve portions of a problem on mobile devices. Therefore, this approach could be seen as one of the first Software-as-a-Service implementations that is heavily based on mobile devices.

Furthermore, some work has already been published with respect to mobilde devices as Web Service providers, e.g. (Li, Chou, 2011) describe an approach based on a modified HTTP protocol that allows to provide Web Services on mobile devices.

Furthermore, in (Jansen, 2012) another approach for providing Web Services on mobile devices based on standardized protocols is described. Additionally, this approach provides the ability to overcome some of the usual problems by providing services on mobile devices, such as frequent network changes and so on.

3 EXAMPLE SCENARIOS

Just to show how reasonable it might be to have a certain service running on a mobile device, this section describes two scenarios that can be implemented with the help of the describe approach.

The first example is sort of comparable to a location based service: one of the most important facts about mobile devices is, that these kind of devices are more like a pack of different sensors, than a single device. Usually, mobile devices nowadays are equipped with a GPS sensor that allows to track the position of a device, an Accelerometer that allows to track the acceleration of the device, a compass to track the heading of the device and many other sensors as well. Therefore, it makes perfect sense either to use the informations provided by these sensor in order to provide contextualized information to the owner of the device while using a specific software, or to make use of these kind of information in order to share informations with others. Here, the first example scenario is a fairly easy one related to the current position of the device. Imagine Person A wants to know the current temperature at a certain location. In order to get this question answered, Person A can just raise the question for the current temperature along with the geo-coordinates of the location he/she is interested in, to a Cloud Service. Then a mobile device that runs the approach described in this paper will retrieve the question raised to the Cloud and can, if the device is currently located within the area of the location in question, answer the question about the temperature.

The second scenario is a completely different one: Another major advantage of mobile devices is the number of devices available. As said before (IDC, 2011), already in 2011 the number of smartphones increased 300 million units. Therefore, if an approach similar to the one described in this paper would be deployed at least to a subset of all available smartphones, this would lead to a tremendous amount of computational power. Furthermore, another positive aspect of smartphones is the fact that these kinds of devices are connected permanently to the internet usually. Beside using the tremendous computational power of these devices, also other scenarios might be reasonable, e.g. making a survey among customers might lead to a question send to the Cloud and answered by a tremendous number of mobile users in a very narrow time range. Here, of course, the feedback of the owner of the mobile device is important, what provides a new scalability dimension for mobile Cloud Computing based services.

4 IMPLEMENTATION

In order to describe the example implementation of the presented approach, this section first provides a classification of the approach. The second subsection describes the approach more clearly and provides a presentation of the example implementation.

4.1 Classification of the Described Approach

According to the NIST definition for Cloud Computing (Mell, Grand, 2011) Cloud Computing consists basically of three different service models. Within this definition the most low level service model is the Infrastructure-as-a-Service model, in which infrastructural resources are provided on a flexible basis. The most top level service model is the Software-as-a-Service model in which a

complete software stack is provided to the end user in a flexible way.

The here presented approach is located in the layer in between these two layers, the so-called Platform-as-a-Service layer. This layer allows the user to deploy user created software of a certain programming language and by certain libraries. The user of a such a service does not have to handle the underlying hardware or software configuration.

Therefore, the major goal for the presented approach is to provide an environment that allows to flexibly deploy pieces of software into a Cloud Computing scenario that consists of mobile devices.

4.2 Description of the Implementation

In order to achieve the goal to flexibly deploy pieces of software in a Cloud Computing scenario consisting of mobile devices, first a decision about the programming language in which the software that should be deployed to the Cloud has to be implemented, must be taken. As already described in the introduction, a number of different programming languages are usually used for the implementation of platform dependent mobile applications. Since the presented approach should of course be able to run on wide variety of different mobile applications (and their according operating systems), a platform dependent programming language does not seem to be the preferred solution. Therefore, a programming language that runs on the common classes of mobile devices would be the logical choice. One of the most prominent candidates of this kind of programming language is probably JavaScript. Not only since NodeJS (Hughes-Croucher, Wilson, 2012), JavaScript is a fairly well recognized programming language not only on the client side of web applications, but also for server side code. Since JavaScript is the basis for a lot of applications spread in the WWW, modern mobile devices are able to interpret the language and to execute the according programs.

Hence, the example implementation for the described approach provides a flexible way to deploy JavaScript code to mobile devices. In order to allow a flexible deployment of new JavaScript programs to a mobile device, a Web Service approach is used that allows to run Web Services on a mobile device. As already said in the state-of-art section, a number of different approaches exist that flexibly allow to deploy Web Services on mobile devices. For the example implementation, the approach described in (Jansen, 2012) was used. With the help of this approach, a limited number of Web Services gets deployed on the mobile devices that later-on provide the Cloud Services. In the first example implementation three Web Services where deployed on these mobile devices:

1. *Deployment Web Service*: this Web Service allows to deploy a JavaScript program on the mobile device.
2. *Task Web Service*: this Web Service provides the possibility to send a certain task to the JavaScript software, formerly deployed with the help of the Deploy Web Service, and to receive the calculated results.
3. *Undeploy Web Service*: this Web Service allows to undeploy formerly deployed JavaScript programs.

This very limited set of Web Service of very basic tasks, still provides enough power to build a solution to the major goal of the presented approach.

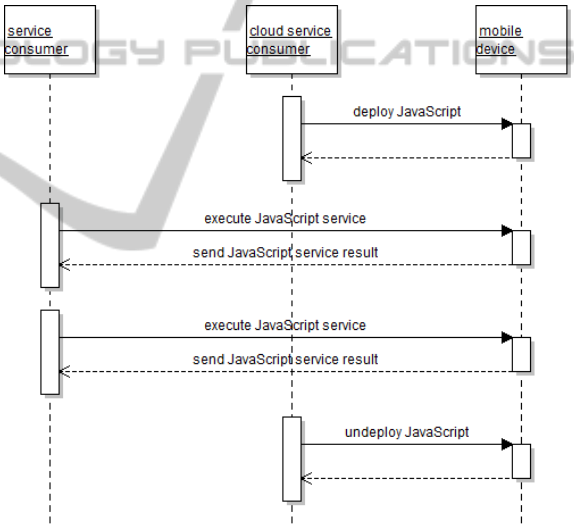

Figure 1: Sequence diagram of the usual service sequences.

Figure 1 shows the usual sequence of service calls either from the view of the cloud service consumer (that is the person who deploys the JavaScript program to the mobile Cloud) and from the service consumer (the one that later-on executes the deployed programs in the Cloud).

First of all, the program that should later-on execute the single tasks in the Cloud, has to be deployed to the Cloud consisting of mobile devices. Later-on a service consumer can execute the deployed programs in order to get his tasks performed. If the deployed software is no longer used, or the cloud service consumer wants to

deactivate his implementation, the Undeploy Web Service can be called in order to remove the program from the mobile Cloud.

The implementation of the three mentioned Web Services is based on a simple XML dialect that allows to describe the tasks that are necessary in order to fulfill the Web Service. Therefore, the XML dialect allows to describe certain necessary information for the according Web Services:

1. *Deployment Web Service*: beside the JavaScript code itself, that should be deployed to the mobile Cloud, also a unique identifier for the program needs to be determined in order to later-on identify the piece of software that should be executed.
2. *Task Web Service*: in order to be able to execute a certain JavaScript task, the Web Service needs to be provided either with the unique identifier of the JavaScript program that should be executed, along with the parameters that should be passed to the program in order to execute the special task for the user.
3. *Undeploy Web Service*: this Web Service simply needs the unique identifier that represents the JavaScript code that should be undeployed.

Already this very limited set of implemented Web Services allows to provide a minimal implementation in order to tackle the major goal of flexibly deployment of small pieces of software to a Cloud Computing scenario consisting of mobile devices.

5 PERFORMANCE TESTS

Since the chosen approach for the implementation of the Web Services uses a polling mechanism, one concern of this approach is the question of its performance. In order to get a first idea how good or bad this implementation behaves with respect to performance issues, a simple performance test was implemented.

5.1 Description of the Test Scenario

For the performance test, we implemented a very simple mobile Web Service. This service only calculates the sum of two given integers and returns the according value as the result. The major advantage of such a simple mobile Web Service is that almost the complete time for the mobile Web Service call is dedicated to the communication, and almost no amount of the round-trip time is used for the calculation itself. Since the communication is the complex part of the presented approach this way of performance testing seemed to lead to results that provide the best overview about the communication performance of the presented approach. As a test scenario we used a usual client (running on a usual PC) which had to do a number of service requests to the mobile Web Service.

In order for being able to compare the results against the performance of usual Web Service calls the same test scenario was implemented just the other way round: we implemented a usual Web Service (running on a usual server) and called this Web Service from a mobile device. Here, the basic idea was that we wanted to use the same hard- and software environment with minimal changes and also the network environments should be the same in all of the tests.

Furthermore, we were interested in the communication performance in different network settings. Therefore, we performed the same tests in basically four different network settings. For each of the tests the (mobile) Web Service and its consumer where running:

- … in the same (WiFi) network,
- … different networks, and the mobile device was connected via WiFi,
- … different networks, and the mobile device was connected via UMTS
- … different networks, and the mobile device was connected via GPRS

Therefore, we conducted eight different test cases. Four for the different network constellations with a mobile Web Service running on a mobile device and a Web Service client running on a usual PC, and four test cases where the Web Service was running on a usual Server and the client was running on a mobile device.

In the test cases where the (mobile) Web Service provider and the client have not been connected to the same network, the central components for the implementation of the polling mechanism have been deployed to a server running via Amazon Web Services (AWS), as a Cloud Computing provider.

5.2 Results of the Test

Within each of these eight test cases, one hundred service calls where performed and the time for each

of these service calls was measured.

The results for the mobile Web Service in the different network scenarios are shown in Figure 2.

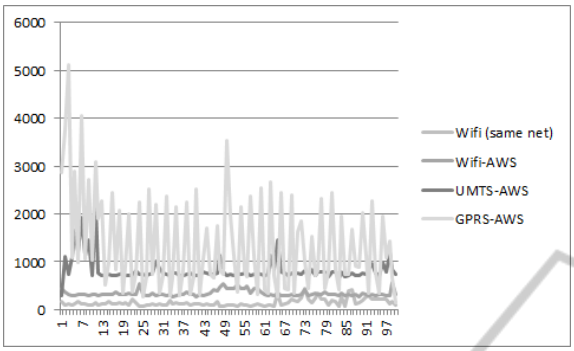

Figure 2: Results for the mobile Web Service in the different network constellations.

As expected the performance for the mobile Web Service calls are pretty good and pretty constant if the mobile device is connected with a WiFi network. The average time if both the mobile Web Service provider and the client are connected to the same WiFi network was M = 147.69ms (SD = 76.00ms). Having the mobile Web Service provider connected to a different, still WiFi, network the average time for one service call calculates to M = 339.04ms (SD = 61.71ms).

Of course we measured less performance of the service calls when the mobile Web Service provider was connected to a mobile network. The results for the UMTS based network connection of the mobile Web Service show an average of M = 827.55ms (SD = 250.35ms) for each service calls, while the results for the GPRS based network are even worse. Here, the average for a single service call calculates to M = 1355.96ms (SD = 986.38ms). As it could be seen by the values for the standard deviation, also the performance of single service calls differs dramatically, e.g. the minimum time measured within the UMTS scenario was MIN = 283ms and the maximum was MAX = 2169ms. Hence, the results for the GPRS based scenario are even worse, with a MIN = 142ms and MAX = 5123ms.

Within the second step of the test, we tried to compare the performance results with the performance that a usual Web Service call has. Therefore, as already described earlier, we established the same test, but this time the Web Service was not running on a mobile device but on a usual server, while the Web Service client was running on a mobile device, again in the four different network settings. The results of these tests are shown in Figure 3.

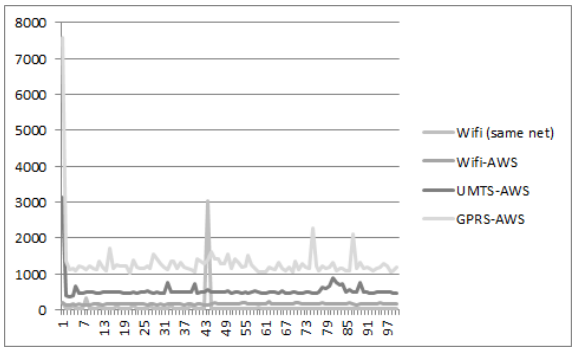

Figure 3: Results for the usual Web Service calls in the different network constellations.

As it could be seen, the results are from both perspectives, the overall performance and the standard deviation in the different network settings better. A usual Web Service call, if the Web Service provider and the mobile Web Service consumer are connected to the same WiFi network has an average round-trip time of M = 61.16ms (SD = 301.36ms). Still within the scenario where the Web Service client was connected to a different (still WiFi) network the average performance was M = 156.71ms (SD = 15.24ms).

Again the values for the Web Service client connected to a mobile network are little bit less. In case of the UMTS network, the average service call had a performance of M = 528.55ms (SD = 273.34ms). Again, the results for the GPRS based network have been worse. Here, the average for each of the service calls calculates to M = 1299.10ms (SD = 658.75ms).

The next step was to compare the different results. The major goal of this comparison was to get an idea of how good the performance of the presented approach for mobile Web Service calls is, in comparison to usual Web Service calls. Therefore, we first calculated the difference in the average performance of a single Web Service call in the different scenarios and in a second step we calculated the percentage of the performance difference in the different scenarios. The results are shown in Figure 4.

	WiFi (same net)	WiFi-AWS	UMTS-AWS	GPRS-AWS
difference	85.53ms	182.33ms	299.00ms	56.86ms
percentage	137.60%	116.35%	56.57%	4.38%

Figure 4: Comparison of the usual Web Service calls and the mobile Web Service calls in the different network scenarios.

Here it can be seen that the performance of the presented approach is not really good if the mobile

Web Service is connected to a WiFi network in comparison to usual Web Service calls. The results for the mobile Web Service provider and the client connected to the same network, show a performance overhead of 137.60%, and still if the mobile Web Service is provided within a different WiFi network, the performance overhead is about 116.35%. But, if the mobile Web Service is connected to a mobile network the performance overhead is not that dramatic anymore. In case of the UMTS network, the overhead was limited to 56.57% and for the GPRS based network, the overhead was still lower at 4.38%. Therefore, on the basis of our test results, it could be said that the performance of the presented approach for mobile Web Services (in comparison to usual Web Services) seems to become better the lower the network bandwidth is. This could best be seen by the results for the GPRS based network, where the actual overhead in our test was below 5%.

6 CONCLUSIONS AND OUTLOOK

As explained at the beginning of this paper, the increasing number of powerful mobile devices provide a reasonable basis for powerful Cloud Computing scenarios based on mobile devices. Furthermore, the example implementation described in this paper shows that it is technically feasible to implement Platform-as-a-Service scenarios based on mobile devices.

Of course the described example implementation is still very fundamental and does not provide a very rich infrastructure for that kind of scenarios. Therefore, the future work should clearly go in the direction of providing more advanced administrative methods, available through more powerful Web Services. Additionally, more advanced features like limitation of different applications only to a limited number of mobile devices or specific amount of other resources.

The chosen approach for the implementation of Web Service on mobile devices seems to make sense, since the performance test still show reasonable results.

Additionally, a number of security related issues show up when Cloud Computing services are to be deployed to devices owned by individuals. Here, also some research should be invested in order to overcome limitations resulting from legal problems.

ACKNOWLEDGEMENTS

This work was partly supported by an Amazon AWS research grant.

REFERENCES

Deursen, A., Klint, P., Visser, J., Domain-specific languages: an annotated bibliography. SIGPLAN No., 35(6): 26-36, 2000

Hughes-Croucher, T., Wilson, M., Node: Up and Running: Scalable Server-Side Code with JavaScript, O'Reilly Media, 2012

IDC Worldwide Quarterly Mobile Phone Tracker, January 27, 2011.

Jansen, M., Getting Serious About Providing Mobile Web Service, In: Proceedings of the 8th International Conference on Web Information Systems and Technologies, Porto, Portuguese, 2012

Li, L., Chou, W., COFOCUS – Compact and Expanded Restful Services for Mobile Environments, In: Proceedings of the 7th International Conference on Web Information Systems and Technologies, Noordwijkerhout, The Netherlands, 2011

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development, In: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, Indianapolis, IN, US.

Mell, P., Grand, T., The NIST Definition of Cloud Computing, National Institute of Science and Technology, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, last visited: 31.01.2012

Mishra, S., Elespuru, P., Shakya, S., MapReduce system over heterogeneous mobile devices, In: Workshop on Sofware Technologies for Future Embedded and Ubiquitous Systems, 2009

Tudor, B., Pettey, C., 2010. Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010, Smartphone Sales Increased 96 Percent, Gartner, http://www.gartner.com/it/page.jsp?id=1466313, last visited 28.01.2012