Acta Cybernetica **21** (2013) 165–176.

Performance Metrics Based Mobile Resource Management[*]

Krisztián Pándi[†] and Hassan Charaf[†]

Abstract

Mobile computer technology has greatly evolved in the recent years. Cloud computing is recognized to be a new area for solving performance issues. Mobile terminal can take advantage from cloud computing. To cope with these new resources and fulfill new quality and performance requirements a more sophisticated architecture and resource management is necessary. The basis of effective resource management is a precise knowledge of the hardware and software capabilities. Performance metrics serve as an input for resource management. This study will present architecture of mobile resource management using cloud resources. The main task of such resource management is to decide which application where to run; on the mobile terminal or in the cloud. This paper identifies key components of resource management, settles the tasks and relationships between them.

Keywords: performance metrics, cloud, mobile, resource management

1 Introduction

A mobile terminal can use cloud for solving performance issues and to obtain richer user experience. The aim of this study is to present an architecture for mobile resource management, that can benefit from cloud computing. Performance measurement and usable metrics are necessary for our later research: decision making mechanism implementation. The goal of the mechanism is to decide where is the optimal place for a certain service/application to run; on the mobile terminal itself or on public cloud computing server. Hence a performance and usage of the mobile terminal should be determined.

Cloud computing promises [5] to provide high performance, flexible and low cost on- demand computing services. Emerging complexity of the application used in

[*]This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013).

[†]Budapest University of Technology and Economics, Department of Automation and Applied Informatics, E-mail: {Pandi.Krisztian,Charaf.Hassan}@aut.bme.hu

DOI: 10.14232/actacyb.21.1.2013.12

mobile terminals implicate harnessing these extra performance resources. Applications with distributed components differ from traditional non distributed applications in numerous attributes, such as communication type and overhead, latency, concurrency etc.

The task of the proposed mobile terminal resource and service management is to decide where an application or service should be executed. To effectively fulfill this complex task a sophisticated and dedicated decision formula is needed. This formula uses dedicated software and performance metrics. Mobile terminal coupled with distributed system can be dynamic, changing over time, resulting CPU and network load changing. Therefore mobile terminal as a part of the distributed hierarchy needs to have very different metrics than traditional software and performance metrics. With mobile computer technology progress, the software and hardware platform becomes more and more complex, together with the amount of the tasks meant to be processed. Mobile terminals have some special features in comparison with traditional computing; small size, dependence on limited battery lifetime, computing power is changing, possible presence of 3D hardware, network bandwidth is limited, and almost exclusively wireless, relatively small display size and special user input.

Usually similar applications are used in mobile terminals and in traditional computers thus similar user experience is expected. Therefore, with comparably less performance nearly the same look and feel is required. In consequence of that capabilities of the mobile hardware should be efficiently harnessed with smart resource management and load balancing.

The main contributions of this paper are: (i) discussion on applicable mobile performance measurement and metrics ; (ii) recommendation to gather the characteristics of mobile phone usage and creating a user specific profile; (iii) discussion on the architecture of mobile resource management, which uses cloud computing resources to enhance the user experience of the mobile terminal.

The rest of this paper is organized as follows: Section II presents the related work. In Section III we discuss the performance measurement methodology, based on that we recommend a performance measurement and profile creation layer in Section IV. In Section V we discuss the architecture for mobile resource management. Finally, in Section VI questions for the future work is described.

2 Related work

The need for measuring performance of computing machines emerged early, and solutions exist for this problem. Early days were dedicated for creating standardized benchmark programs, such as SPEC [9] or EEMBC [11] etc. Common attributes of these benchmarks are the batch style execution, what is measuring program executing time and speed. The less is the executing time the higher is the system performance. All of these benchmarks are measuring not only the program execution, but the operating system itself, with its interrupts, caching etc, what make the result ambiguous. This batch like execution does not simulate fully the everyday operation and usage; it is not detailed enough. Other benchmarks see the optimal metrics in picking commonly used applications from the application pool, and running them one by one. Although this is only a derivative of previous solution, there is a possibility to run these applications parallel. To summarize: currently available performance metrics mainly come from traditional computing world.

Adaptive Mobile Systems solutions proposed a model, where applications or services are traveling from performance lack devices to device in idle. Adaptation is an application attribute to change its behavior when surrounding environment changes (CPU load, battery power etc.). Adaptation strongly depends on current system performance, so some efforts are made to define metrics. One solution is to measure CPU load, and battery power, and send it to the application for decision making [14], as battery time is a key point in mobile terminals. Online collection of dynamic software metrics (number of invocations, and response time) was considered [13]; a drawback is that the application code must be changed to insert measurement code. In [15] metrics for service oriented systems are proposed, namely size, coupling, performance, and resource utilization.

Effective resource managements static and standard goal is to extract as much performance as possible from available hardware and software base. As technology changed, the resource management methods and focus may change from time to time as well. Efforts are made to research this field; mobile cloud computing middleware is presented in [16] focusing on service and service bus. Computer service performance is presented in [7], numerical modeling the response time of the services.

A summarization work on cloud and grid computing is presented in [12]; most interesting part is when it states it is often impractical to assume such detailed priori knowledge of management policies for all the resources will be available to resource brokers in a large and dynamic heterogeneous environment. Recommendation is to use resource management "in the dark". Another article [4] quantitatively compares the layered queuing and historical techniques including thoughts on how they could be combined. In [17] resource management mechanism for clouds is described, with compound framework that integrates hierarchical structure and P2P architecture and combines their characteristics. The mentioned paper is mainly focusing on traditional cloud topology with high bandwidth network access. Although some methods can be used for mobile architecture.

A potential synergy between mobile terminal and cloud is proposed in [10], it is found that cloud deployed applications that employ mobile devices as end-points are particularly exciting due to the high penetration of mobile devices. The value of the article in complex approach to both end of the cloud, the cloud itself and the possible endpoint (mobile as well).

Authors of [8] created a framework which partitions the workload of complex services in a distributed environment and keeps the Web service interfaces on mobile devices. Article describes a service like approach, assuming that every application can be executed as a web service. Our approach is similar to this, but with following difference:

- All the functions/applications are partly executed locally on mobile terminal, thus resources of the mobile device are always employed. In our opinion it must not be always the case. Video content can be streamed, gps content can be sent via network etc.

- Only heavy duty tasks are executed in the cloud. In our architecture, we do not make such a difference between tasks.

An interesting approach is the CloneCloud [6], a system that automatically transforms mobile applications to benefit from the cloud. It contains a flexible application partitioner and execution runtime that enables unmodified mobile applications running in an application-level virtual machine, placed in the cloud. This approach is transforms single-machine execution into distributed execution automatically, but part of the process must run on the mobile. A cloned virtual machine with virtual hardware is needed, with complex profiler, migration handler. Our architecture does not recommend such a granular execution, where threads and their content are travelling from Cloud to mobile terminal. We propose data migration rather than thread migration, and that full jobs would execute in one place, not partial part of the jobs.

Neither of the method focuses on profile creation, what can be beneficial during resource optimization. The proposed resource management strongly rely on information coming from profile.

3 Performance measurement - methodology

Performance measurement is a rather complex task, as we have seen in chapter II. and there is a decision to be made which path to follow. One approach is to measure pure performance of certain hardware component, namely CPU, GPU, storage, network bandwidth etc. This gives us a very good detailed picture of capability of given hardware component. Obviously in a complex system, there is no clean testing of component, because system parts strongly depend on each other. This dependency must be handled; impact can theoretically be minimized with careful design, dependency can be taken into account saying it comes with the method. Despite mentioned drawbacks, this method is suitable for checking basic capabilities of the hardware itself. From performance metric point of view, these tests have very limited usability; global throughput of the system often cannot be predicated from atomic parts of the system. Although, it can be used if the service or application component strongly depends on hardware component. For this a good real life example is hard to give. Taking games as an example (they are highly performance consuming applications), it can be seen that they do not exclusively depend on GPU capabilities of the hardware. Game application also depends on input peripheries, storage, or network bandwidth of a device. Other example is media streaming; network is a strong factor of the overall performance, beside CPU properties, not to mention storage size and speed. Cloud computing is a good example where these separate benchmarks come in hand; to have a good service management mechanism, basic performance characteristics must be known.

Another approach is to measure performance on application base. The basic idea is to collect commonly used applications, and run them in a batch way, one after the other, in specific order. Problems with this approach are:

- how to decide which application to run in this benchmark
- can a common application basket be selected from numerous application pools
- which applications and how long to run
- how to assign a weight to an application, to calculate a final score
- what consequences can be derived from different application run for the new application

For commonly used applications this performance measurement is suitable for comparison, but for a new, or custom application it does not add much to our knowledge. Additional problem of this performance measurement method is that there is no knowledge about real application running on the device. This can be avoided if on the fly measurements are implemented on the device. We suggest this method; a data collector can create a profile, based on the mobile terminal usage. The concept is that long term measurement is implemented on the mobile device, which monitors the terminal usage. The data is collected and evaluated, and based on that a real life benchmark can be created and collected.

Other not frequently mentioned aspect of the performance measurements is the multithreaded multi-core environment. Todays mobile hardware CPU is multi-cored, and this must be included in the performance measurements. The simplest way is to gain performance measurement, to launch the same application twice, and see its behavior, or simply launch the application and see whether it scales with multiple-core. Data collection is also meaningful, because the application usage can be monitored, together with applications that are running in parallel, giving the resource management mechanism usable basic data.

4 Performance measurements

One of the focus of this study is what kinds of performance measurements are mandatory for effective application management. The management needs to decide where certain application should run, on mobile terminal or in the computing cloud. Special challenge of cloud computings resource management is that the application/service must run and finish in the fastest way. Driving factor of the resource management is to gain speed in application calculation and running time; enhanced user experience is expected. Decision has to be made how this requirement can be achieved. At first glance it seems that every application must run in cloud (as [8] recommends for example), because that will lead to highest user experience. Unfortunately this is not always true. Cloud computing has a bottleneck, it can have a huge computing capacity, but there are several criteria to harness it:

- all data must be present in the cloud computing environment
- all data will be calculated in cloud computing environment
- data travel from client (currently mobile terminal) to server, or to client from server is expensive from performance point of view.
- client should be online

Effective cloud computing usage depends on the available network connection quality, mostly on latency and available bandwidth. Although network bandwidth is increasing, it cannot keep up with the CPU performance and storage capacity; the gap is opening. And even if several megabits are present in wired connection, wireless connection will have more limited bandwidth. A good benchmark will test the available bandwidth in longer term, to assist to effective decision making mechanism. Long term measurement can have more benefits; time and locality dependent map can be made containing data about available bandwidth, the time interval for this bandwidth can be used etc. With this profile application management can be enhanced. So, the most important metric is the available network bandwidth.

Additional straightforward metrics are; CPU, GPU, storage I/O, keyboard input capacity. For our architecture an important performance metric is the user mobile application usage characteristics. For example, if the user tends to use more application at the same time, it is a good idea to take it into account. User experience is enhanced, if time consuming applications are moved into the cloud, like a background task, while providing more performance to other applications.

For performance measurement currently available solution can be used (e.g. for GPU Basemark [1], GL benchmark [2], WP bench [3] etc.). For missing benchmarks a custom one can be made focusing on certain parts of the hardware. This applies for example to I/O performance of the mobile device storage, because the built in component may vary from model to model. The key is the comparability and the ease together with multi platform implementation.

In Fig. 1 code migration measures are listed. Although it is code and not application specific, it gives us an overview what performance metrics are necessary for further evaluation and decision making. It is worth to mention, that with increasing number of metrics, the decision matrix is getting more and more complex.

The question is what to do with performance measurements results. It is not necessary to test every device of the mobile terminal. An online database stored in the cloud is suitable for effective resource management needs. This online database can hold performance information about the known mobile terminal models, and if a terminal is connecting to cloud, it can be updated and used.

Fig. 2 shows profile creation procedure. Performance measurement results are collected from mobile terminal runtime results, or downloaded from network

Attribute	**Metric**
Software	Size of executable module (e.g. Java .class file)
	Size of a serialized object
	Size of an in-memory object
	Size of an extra-memory for execution of a method
	Number of executable statements/instructions
	Size of serialized parameters
	Number of method invocations
Performance	Method execution time
	Method invocation time
	Instance migration time
	Class migration time
Resource Utilization	Network bandwidth
	Network usage
	Size of available memory
	Memory usage
	Processing power
	Processing usage

Figure 1: Code Migration Performance Measures

database containing device specific information. The database prevents basic device characteristics to be rechecked all the time, for example CPU statistics, GPU presence etc. Performance metrics are handled to profile manager. Application and resource usage history is also collected and provided to profile manager, for further processing and decision making.

Table 1: Application categories based on performance requirement

Applications	**Performance requirement**			
	CPU	***Storage***	***Network***	***GPU***
Browser + Flash	+++	++	+++	++
Browser No Flash	+	+	++	
Games	+++	++	++	+++
Office	++	+	++	
Image viewer	++	++	++	
Chat text	+		+	
Voice chat	++		++	
Online media content	+++	++	+++	+++
Storage backup with compression	++	+++	+++	
VNS/SSH X	+		++	

Table 1 shows a categorization of applications used in mobile terminals. Some performance requirements are collected and weighted with + sign, showing the extent of the usage. This gives us an overview on performance critical parts. These application can run parallel, or if additional CPU core is available, on different cores. Without user experience dropdown, an application can run parallel, if it does not use the same resource heavily. If a race condition occurs; it will lead to

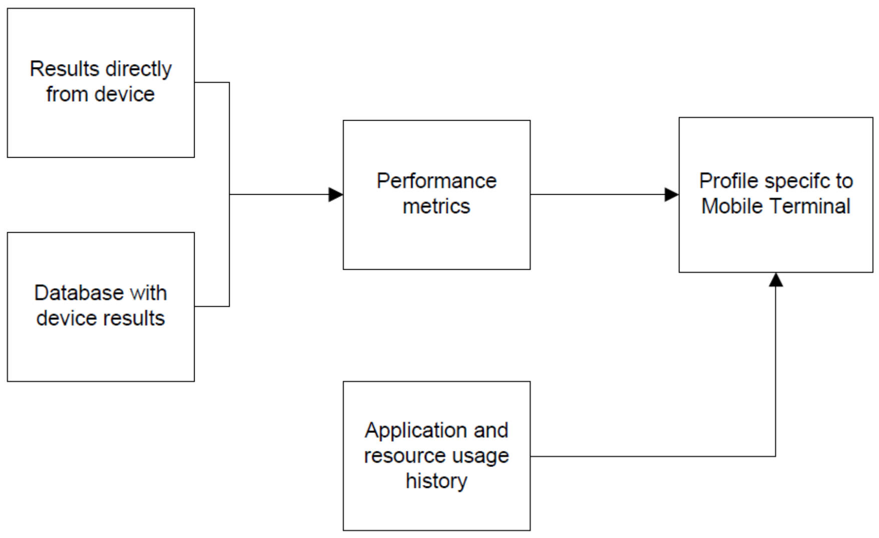
Results directly from device
Database with device results
Performance metrics
Profile specifc to Mobile Terminal
Application and resource usage history

Figure 2: Performance metrics and profile creation

lags and slow interaction.

This paper does not deal with performance measurement of cloud. It is a task of later research, to verify whether it is necessary to verify the performance of the cloud. In this paper we can safely assume that beside the network overhead, cloud has more calculating and storage I/O capacity in comparison to mobile terminal.

5 Resource management in mobile terminal

In the previous chapters the performance measurement of mobile terminal was discussed, which will serve as an input for the resource management. The task of such management layer is to decide where certain application to be run.

In this chapter the architecture of a resource management system will be discussed. The place of the resource management is in the mobile terminal. If no network connection is available, or it is relatively slow, every application will run on the mobile terminal. Mobile terminal must remain usable if some hardware resource is not available or has low performance.

Currently we do not deal with how it can be technically done to have the same application or part of the application available in cloud or in the mobile terminal. It is a scope of future research. We assume that they are present, and every application can run either on the mobile terminal or in the cloud. This hypothesis might seem strong; in related work [12][13] solution for this can be found, although we listed the problematic parts of them. In future the topic will be checked in more detail.

The first problem to deal with; where the data physically is. For applications heavily dependent on input data, or dealing with large data, from performance point of view it is necessary to have the data locally. Mobile terminals have wireless connection, which is relatively slow; network interface should be warily used, to avoid unnecessary and frequent data sending.

As an example an image viewer application will manipulate pictures faster if they are available on local storage, and photos taken with the device are certainly there. Other case if the images are placed in the cloud, that way cloud manipulation can be faster. This leads us to another aspect; security and background synchronization.

Using cloud computing to aid resolving a performance bottleneck in mobile computing has an important aspect; financial cost of the used cloud resource. Detailed discussion of this topic is not the task of this study although the suggested resource management is capable of dealing with it. In the profile the user can define the cost attribute (limit, preferences etc) and the decision is influenced by these factors.

Data can travel from mobile terminal to the cloud if the synchronization is enabled. It can be done when the network is not used, and in this way large data can be transferred to or from cloud unnoticeably. To achieve this, earlier mentioned profile making is necessary. It needs to be known, or at least predicted, when the network will be in idle state, when it is heavily used. Table 1 comes in hand in this work item, it can be known that what type of application is using which resource of the terminal; with that information the smart synchronization can be realized. Another benefit from profile mechanism is that heavily used data can be identified. If a user is normally editing and modifying the attachments of an email, as an example, it is good to keep them on both storage places, to be ready for network unavailability and to allow data to be processed on mobile terminal and cloud as well.

From user profile parallel used application can be identified. This leads to an interesting opportunity; optimization for that type of usage can be done in the resource management. Application place to run can be modified not only based on performance metrics, but based on the user mobile usage. If application is producing large data usually serving for other applications as an input; it is good to run them in the same place. But if there is no large data involved, run them separately, and making some resource free for other applications. Network availability can be taken into account in this case, together with safe data storage.

There is a feature in every operating system, to put processes to background till further usage. This enables the operating system to do some optimization with resources, unload the application partly or totally from memory, and place it to swap memory. Similar feature might be feasible in the resource management too; background task can be moved to or from cloud freeing mobile terminal resource.

Battery usage and state should be monitored during decision making. If the mobile terminal is running out of the battery, application heavily using CPU can be moved to the cloud, regardless of performance degradation. This could be set as a configuration parameter similarly to some operation systems, user could choose between energy schemes, defining the behavior in such case. Performance of the mobile terminal can be measured on the device itself, or the common data can be purchased form the online database. Profile information, or other user specific, but not so sensitive information must be stored on the mobile terminal. Beside obvious data protection, the resource management layers need this information constantly, even if network connection is not available.

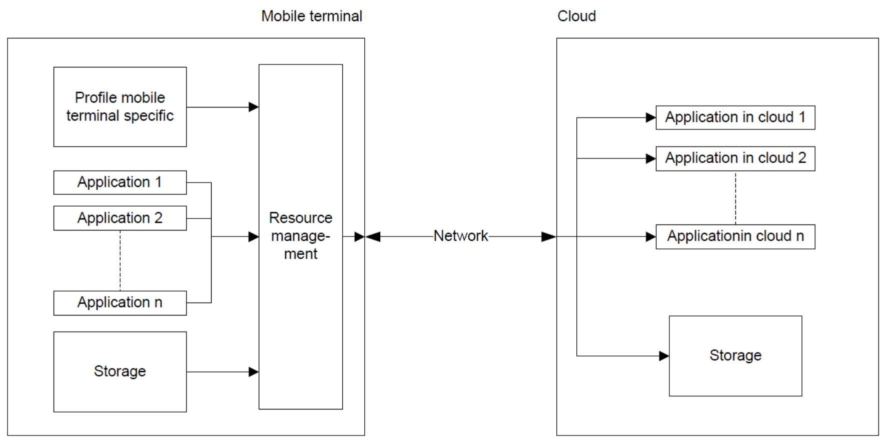

Figure 3: Resource management architecture

Fig. 3 shows the resource management architecture with key parts. Profile part, what is mobile terminal specific, already contains necessary performance metric information. Profile changes on the fly, with actualized performance information and application usage collected from mobile terminal. If the resource management decides that application will run faster, application from cloud is called, and returned information is transferred transparently to mobile terminal. This happens unnoticeably, the user should feel the speed enhancement from this management. Information is transmitted through available and constantly changing network bandwidth. Storage block represents mobile terminal local data; it can be synchronized automatically to cloud if user enables this feature.

6 Future work

The presented architecture is capable of enhancing the user experience and harnessing extra performance given by cloud computing environment. Realization in test implementation will follow, to confirm resource management architecture. Collectable performance metrics should be limited to extent that serves the effective resource management, without unnecessary data. Cloud computing environment should be checked from performance point of view. The financial cost of the cloud computing should be a part of the architecture as well, how and where is a part of future work. We assumed that every application can be run in cloud or in the mobile terminal. Technical and theoretical background of this topic will be checked in future work.

References

[1] Basemark. http://www.rightware.com/benchmarking-software/product-catalog/, accessed on Jan. 20. 2013.

[2] Glbenchmark. http://www.glbenchmark.com/benchmarks.jsp, accessed on Jan. 20. 2013.

[3] Wp bench. http://www.windowsphone.com/hu-hu/store/app/wp-bench/e447e949-01c0-43f1-8b65-76d752d7d305, accessed on Jan. 20. 2013.

[4] Bacigalupo, D.A. Resource management of enterprise cloud systems using layered queuing and historical performance models. In *2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8, April 2010.

[5] Buyya, Rajkumar. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 1–, Washington, DC, USA, 2009. IEEE Computer Society.

[6] Chun, Byung-Gon, Ihm, Sunghwan, Maniatis, Petros, Naik, Mayur, and Patti, Ashwin. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[7] Gani, Hendrik, Dr, Supervisor, and Ryan, Caspar. The impact of runtime metrics collection on adaptive mobile applications. 2005.

[8] Hassan, Mahbub, Zhao, Weiliang, and Yang, Jian. Provisioning web services from resource constrained mobile devices. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, CLOUD '10, pages 490–497, Washington, DC, USA, 2010. IEEE Computer Society.

[9] J.L.Henning. Spec cpu2000: Measuring cpu performance in the new millenium. *IEEE Computer*, 33(7):28–35, 2000.

[10] Lehman, Tobin J. and Vajpayee, Saurabh. We've looked at clouds from both sides now. In *Proceedings of the 2011 Annual SRII Global Conference*, SRII '11, pages 342–348, Washington, DC, USA, 2011. IEEE Computer Society.

[11] Levy, M. Evaluating digital entertainment system performance. *IEEE Computer*, 38(7):68–72, 2005.

[12] Majumdar, Shikharesh. Resource management on clouds and grids: challenges and answers. In *Proceedings of the 14th Communications and Networking Symposium*, CNS '11, pages 151–152, San Diego, CA, USA, 2011. Society for Computer Simulation International.

[13] McGregor, John D., Cho, Il-Hyung, Malloy, Brian A., Curry, E. Lowry, and Hobatr, Chanika. Collecting metrics for corba-based distributed systems. *Empirical Softw. Engg.*, 4(3):217–240, September 1999.

[14] Noble, Brian D., Satyanarayanan, M., Narayanan, Dushyanth, Tilton, James Eric, Flinn, Jason, and Walker, Kevin R. Agile application-aware adaptation for mobility. In *Proceedings of the sixteenth ACM symposium on Operating systems principles*, SOSP '97, pages 276–287, New York, NY, USA, 1997. ACM.

[15] R.Rossi, Z. Tarri. Software metrics for the efficient execution of mobile services. In *in Proc of ECOWS06 Workshop on Emerging Web Services Technology*, December 2006.

[16] Wang, Qian. Software metrics for the efficient execution of mobile services. In *SOA's Last Mile-Connecting Smartphones to the Service Cloud*, pages 80–87, September 2009.

[17] Xingye, Han, Xinming, Li, and Yinpeng, Liu. Research on resource management for cloud computing based information system. In *Proceedings of the 2010 International Conference on Computational and Information Sciences*, ICCIS '10, pages 491–494, Washington, DC, USA, 2010. IEEE Computer Society.