Mobile Performance Metrics for Resource Management

Krisztián Pándi
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
Email: Pandi.Kriszttian@aut.bme.hu

Hassan Charaf
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
Email: Charaf.Hassan@aut.bme.hu

Abstract—In information technology two recent trends can be observed. Mobile computer technology has greatly evolved providing various rich functionalities. Capabilities of mobile terminal is given, more performance can be obtained with adding additional resources. Cloud computing is recognized to be a new area for solving performance issues. These technologies used together can enhance the user experience of the mobile terminal. Sophisticated mobile resource management layer needs to be used, that can utilize extra performance of the cloud. The basis of effective resource management is a precise knowledge of the hardware and software capabilities, furthermore, the usage of mobile terminal. Performance metrics serve as an input for resource management. This study will present architecture of mobile performance measurement and profile creation layer. Key performance metrics are collected and investigated. Attributes of profile creation are also gathered. Architecture is examined, whether it can be later used as an input for mobile resource management.

I. INTRODUCTION

The aim of the study is to determine the performance of the mobile terminal. Performance measurement and usable metrics are mandatory for our later research, what is resource management in mobile terminal. The goal of the resource management 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 [1] to provide high performance, flexible and low cost on-demand computing services. Emerging complexity of the application used in 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 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. As such a formula amplifies adapted and 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. Despite of changing attributes, mobile terminal have a usage history, where environment can recurrence. Therefore profile creation is recommended, based on data collected from mobile terminal usage.

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 methodology; (ii) recommendation to gather the characteristics of mobile phone usage and creating a user specific profile; (iii) discussion on the architecture of mobile performance metrics and profile creation architecture.

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 conclusions are presented. Finally, in Section VI questions for the future work is described.

II. RELATED WORK

The need for measuring performance of computing machines arisen early, and solutions exists for this problem [2]. Early days were dedicated for creating standardized benchmarking programs, and various of them were created (Standard Performance Evaluation Corporation (SPEC) [3]) or EEMBC [4] etc. Common attributes of these benchmarks is the batch style execution, what is measuring program executing time and speed, less is the executing time 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, 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, adding a possibility to run these application parallel. Possible performance evaluation scenarios are briefly presented in [5], special attention on challenges for software engineer. Research paper [6] focuses on performance metrics in wireless communication, proposing an analytical model. Currently available performance metrics are mainly coming 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 [7]. One solution is to measure CPU load, and battery power, and send it to the application for decision making [8], as battery time is key point of the mobile terminal. Online collection of dynamic software metrics (number of invocations, and response time) was considered [9]; drawback is that application code must be changed to insert measurement code. Paper [10] propose metrics for service oriented systems, namely size, coupling, performance, and resource utilization. Other research [11] suggests a formal model based hypotheses, using performance metrics.

To summarize; currently available performance metrics mainly come from traditional computing world.

III. PERFORMANCE MEASUREMENT - METHODOLOGY

Performance measurement is a rather complex task, as we have seen in Section 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. Our proposed architecture will use this method in performance measurement part. From performance metric point of view, these tests have very limited usability; global throughput of the system often cannot be predicted 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. This method will be a part of the recommended architecture; 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 in this, because the application usage can be monitored, together with applications that are running in parallel, giving the resource management mechanism usable basic data.

IV. PERFORMANCE MEASUREMENT ARCHITECTURE

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 computing 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 [12] 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, our proposal to test it in long term and creating a profile from these data.

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 Basemark [13], GL benchmark [14], WP bench [15] 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 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>
<thead>
<tr>
<th>Applications</th>
<th>CPU</th>
<th>Storage</th>
<th>Network</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser + Flash</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
<td>++</td>
</tr>
<tr>
<td>Browser No Flash</td>
<td>+</td>
<td>+</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>Games</td>
<td>+++</td>
<td>++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>Office</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Image viewer</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Chat text</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Voice chat</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Online media content</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Storage backup with compression</td>
<td>++</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
</tr>
</tbody>
</table>

Table I 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 lags and slow interaction. This information besides the raw performance metrics, is mandatory for resource management layer.

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.

V. Conclusion

Based on the architecture shown in Fig. 1, a test application was implemented to verify usability of the architecture. Implementation was done on the mobile terminal. One software and hardware platform was picked, to have results as early as possible. Application itself was designed to be as much platform independent as possible, but due to GUI and GPU differences of certain platforms it could not be done in full extent. Test application contained performance measurements with a GUI that can send the data to the server. The test suite contained the following test subparts:

- CPU test
- Storage test
- Memory test
- GPU test
- Network test

Mobile processors perform well in standard applications, but in comparison with desktop CPU they have performance lack. Therefore, a simplified CPU benchmark was used, based in prime number check, but with reduced memory and CPU usage. Storage in mobile terminal can be the bottleneck of the overall performance, thus it must be checked in separate metric. Both read and write performance is checked. Memory performance and availability is obtained via random and sequence memory allocation test. GPU performance is checked with simple 3D cube rendering, which enable the usage GL10 and GL11 features. Network bandwidth and latency is checked, in a long term. It can be set as a background task to be able to create a real profile. Latency/ping results are collected, because beside the bandwidth it has a key role in software responsiveness.

All these software metrics and results are collected together with the operating and hardware system details. The obtained data is stored locally and can be sent to server, with keeping the anonymity of the device, to have a collection of the results. This is needed for optimization purposes; not all the test must be repeated in all mobile terminals. With system information it is possible to track the hardware changes also. For example storage performance can change by replacing the hardware, so it should be tested from time to time. We suggest running parts of these benchmarks not only in batch mode, one after each other, but in parallel, to be able to determine the performance when different applications are using the same system resource. This method is not mentioned in any of related works.

Performance metrics is one input for profile creation mechanism. The other one is the resource and application usage history. The aim of the performance metrics together with profile creation is the provide input for the resource management layer. Such a layer can benefit from application usage history of a user. Thus, beside performance metrics application usage history is collected in long term. In regular period running processes are gathered and stored with time stamp, together with current network connectivity setting. Based on data collected, some kind of profile, usage pattern can be calculated; even usage characteristics can be predicted.

Test application was tested on some mobile terminals, the data and results are collected. See Table II. for results. The performance measurement are reduced to elapsed time measurement, so higher is the value, the lower is the performance. In bandwidth test, it is not the case; higher value means higher throughput.

Suggested profile creation architecture is a part of resource management layer, which task is to determine the optimal place for a task to run; on mobile or in the cloud. Based on performance metrics and profile data, a formula is needed to be defined, to be able to decide about task execution place fast. Discussion of suggested resource management is a topic of other research paper; it will collect the user’s application execution requests, decide about the place to run, do the necessary communication, data exchange, and come back with application UI or with the result of the execution.

Based on test application it seems that the performance metrics and profile creation architecture can reach its goal to provide input data for resource management layer.

VI. Future work

The presented architecture is capable of providing input data for resource management layer. Collectable performance metrics are be limited to extent that serves the effective resource management, without unnecessary data. Future work will focus on creating resource management architecture, what will give a feedback for profile architecture. This will refine the metrics and the data collection also.

Acknowledgment

We would like thank the AAIT department for feedbacks and review and supportive environment. This work was par-

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>EXAMPE MEASUREMENT USING PERFORMANCE METRICS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>device 1</td>
</tr>
<tr>
<td>Memory(R/W)</td>
<td>2242/10063</td>
</tr>
<tr>
<td>3D</td>
<td>2005</td>
</tr>
<tr>
<td>Storage</td>
<td>95</td>
</tr>
<tr>
<td>CPU</td>
<td>34</td>
</tr>
<tr>
<td>Ping</td>
<td>65</td>
</tr>
<tr>
<td>Bandwidth(Up/Down)</td>
<td>2474/3960</td>
</tr>
</tbody>
</table>
tially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013) organized by VIKING Zrt Balatonfüred.

REFERENCES


