Acta Cybernetica **22** (2015) 171–181.

Mobile Resource Management Load Balancing Strategy*

Krisztián Pándi† and Hassan Charaf‡

Abstract

This paper is dealing with mobile resource management that distributes mobile device processes between cloud computing virtual machine and mobile device. Expectation is that mobile device user experience will increase. Load balancing is an important part of mobile resource management. In order to be able to solve load balancing issues, discussion is made how currently available methods can be adapted to our resource manager. In this article, we investigate load balancing procedures, methods and their customization, with a particular attention on mobile and cloud computing requirements. As a result, we expect that important design aspects will become apparent. Profile based load balancing method is recommended, what combines static and dynamic load balancing strategy. Using profile to identify usage scenario of mobile device can lead to increased system responsiveness, thus experience improvement.

Keywords: cloud, mobile, load-balance, resource management, network aggregation

1 Introduction

Mobile devices have become a part of everyday life. Due to their small size it is always at hand and has relatively high calculating capacity, which offers many possibilities. The use of mobile devices diverse, ranging from simple web-browsing to complex corporate application usage; this impose different requirements. Therefore, the resource need of applications can widely vary. Despite the increasing performance, there are scenarios, especially in case of parallel execution and CPU intense application, where resource requirements exceed the capabilities of the device. If for example CPU or APU is used at full load energy demand will significantly increase; mobile device has limited battery capacity, so it should be managed carefully.

*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).

†E-mail: Pandi.Krisztian@aut.bme.hu

‡E-mail: Charaf.Hassan@aut.bme.hu

DOI: 10.14232/actacyb.22.1.2015.11

Cloud computing promises [2] to provide high performance, flexible and low cost on- demand computing services. In cloud computing platform as a service models (PaaS) providers deliver a scalable platform; operating system, executing environment for various programming languages and database if needed. The raw performance (number of CPU cores, memory, storage/disk size) of cloud computing virtual machines or virtual instances can be adjusted on wide range; from single core to 32-36 cores, RAM from 0.75GiB to 112-224GiB, disk size from 382GB depending on price. Thus, cloud computing virtual machine resources compared to mobile devices are significantly larger and more scalable. From now on we will use cloud computing resources/environment term in cloud computing virtual machine resources/environment sense.

It seems tangible to use cloud resources in mobile device. In our previous article [13] we suggested such architecture of mobile resource management, which can utilize benefits of cloud computing, expanded with smart using of available network interface parallel. 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.

There are numerous processes or tasks running on mobile device; system processes, user triggered tasks etc. Idea is to move user tasks into the cloud computing environment, virtual machine, where these task are processed faster, saving CPU time and increasing battery lifetime. Proposed resource management layer will decide about moving certain task to cloud virtual machine if overall processing time is expected to decrease with this decision.

In addition to obvious benefits in synergy between mobile device and mobile terminal, there are number of open questions. First bottleneck comes from cloud computing attribute; resources are available only over the network, therefore for smooth user experience, high bandwidth network connection with low latency is required. Mobile devices use wireless connection, whats connection quality and throughput may vary. Besides that, this additional performance of cloud computing has price, the provided service is usually not for free; thus good resource management architecture must pay attention to that aspect.

One of the important open questions is the load balancing strategy of suggested resource management layer. In this article load balancing topic will be investigated; how it can be inserted into current resource management architecture, which strategy is more forward-looking and most advantageous. Load balancing strategy has a key role in resource management architecture; it must meet several parallel requirements. The main goal of load balancing is to optimize resource usage. In proposed resource management architecture currently two resources are available; mobile device and cloud computing environment. Well-chosen load balancing strategy may benefit from extra resource, and can lead to increased performance and reliability.

Target of the resource management is user experience enhancement of the mobile device, and if possible, prolonging the battery lifetime. In order to achieve this, CPU load and network utilization should be kept low, as these are mainly responsible for power consumption. Target resources of the resource management are: CPU, network bandwidth, GPU, memory. The primary target of the load

balancer is task to be executed on mobile device; a decision has to be made about task place to run in order to achieve better overall performance; on mobile device or in the cloud computing environment.

In current article software load balancing open questions are investigated, which were aroused during resource management architecture planning and realization.

The main contributions of this paper are: (i) discussion of currently available load balance methods (ii) discussion on how available load balance methods can be adapted to resource management that distributes tasks between mobile device and cloud computing virtual machine resource; (iii) recommendation is given on load balance method, which combine static and dynamic load balancing strategy, that uses profile already available from our resource management.

The rest of this paper is organized as follows: Section II presents related work. In Section III we discuss load balancing strategies and our load balancing proposal for mobile resource management architecture. Finally, in Section IV conclusions and questions for the future work is described.

2 Related work

A summarization work on cloud computing is presented in [11]. In [16] resource management mechanism for clouds is described, mainly focusing on traditional cloud topology with high bandwidth network access.

Synergy is proposed between mobile device and cloud computing in [10], with complex approach including cloud and mobile device endpoint. Authors of [6] implemented a framework to partition the workload of services, with Web interface service on mobile devices.

CloneCloud's [3] approach automatically transforms mobile applications to be able to run in cloud computing environment. It contains a flexible application partitioning mechanism and execution runtime that enables unmodified mobile applications running in an application-level virtual machine, placed in the cloud. A cloned virtual machine with virtual hardware is needed, with complex profiler, migration handler.

In [14] article some load balance algorithms are investigated; mainly from cluster task distribution perspective. Mixed algorithm is proposed by authors that improves the performance of the cluster system. From our perspective, the proposed method is too closely linked to the clusters. It regularly collects performance parameters (CPU and memory utilization) from calculating nodes in clusters, and based on residual load rate, the actual load balancing is executed. Network load and throughput is not taken into the consideration at all.

Article [9] takes more into account the characteristics of cloud computing environment. Proposed algorithm is based on job size, and the jobs are equally spread, so the complete computing system is load balanced, therefore no virtual machine remains underutilized. Data transfer cost is promised to be minimized but without any details. Utilizing all virtual machines equally approach is not suitable for cloud computing and mobile device hybrid resource management, as our goal is to free

one resource from utilization.

Study [8] is focusing on dynamic load balancing strategy and gains result with delaying job execution when the system is fully used. Authors pay attention on network delay to avoid scheduling errors.

Net profit-based load balancing mechanism is presented in this [15] paper. Each process is associated with a net profit value, that represents the cost of migration task. Possible benefit is gained by moving a task to another node. Although it focuses on social networked multiagent system, the approach can be useful to resource management architecture as well.

In [1] article some of load balancing techniques are investigated. It is partly dealing with problem how to handle nodes that have very different processing speed, with some shallow comparisons of algorithms.

Article [5] is dealing with scalable, low-error load balance measurement. Mentioned technique collects and reconstructs system wide measurements with low error that can be useful during evaluation of load balancing methods.

Research paper [12] deals with performance measurement topic; how to improve the performance of complex system. In details the relationship between throughput and response time is examined, which is often taken as a reciprocal type of relationship. Throughput measurement is more straightforward, response time is more difficult and tended to be measured what's easy to measure. Interesting statement is that performance is a feature of software application.

Paper [7] introduces a distributed hash table (DHT) based on load balance algorithm with pending pool idea, which divides available resources to light and heavy peers.

In our research paper [4] an Energy efficient code generator was introduced. It decides automatically at compile time, which task should run on the smartphone and which task can be offloaded. The main focus was energy efficiency, it was measured that the energy consumption of the mobile phone was reduced with 35% using computation offloading. This paper suggests an architecture where the decision is made in run time, rather than compile time.

3 Load Balancing Strategy In Mobile Device

Load balancing task is to decide where certain application should be executed; it is used for distributing workload across computing resources.

Load balancing has to solve several problems. In current resource management architecture, the main idea is to use cloud computing resource - virtual machine - to enhance the performance of mobile devices. Place of the resource management is in the mobile device. It must identify and prepare applications to move into the cloud.

Main task of resource management layer is to decide about moving certain application, task or process for execution to the cloud computing environment. This decision is made on expected overall performance and battery life gain. There should be an enhanced user experience. Moving to cloud for execution has several

aspects to be considered; overhead of communication time and energy wise, locality of the data needed for process execution, where should be the result data stored after the execution, dependency between applications etc. Place of the resource management is obviously in the mobile device. It must identify and prepare applications to move into the cloud. Such methods are recommended that do not use more CPU and battery life for task distribution than that is gained with resource management; fast and simple algorithms are more beneficial, possible calculated in advance. Profile creation can be solution for that as it builds up a priori knowledge about user habits.

First problem to deal with is the process/task properties. It is obvious that not all process have equal run cost and the knowledge about this cost is not known in advance. There can be certain dependencies between applications, if it is present; they cannot run in parallel, if not; there is no such rule to follow. Other important information about these processes is the locality of the needed data; this mandatory for reducing communication overhead. Easiest load balancing method is when the applications always have the same calculation cost. The harder is when tasks have different known amount of calculation cost. Hardest problem when no information is available about calculation cost in advance, only after the application finished running. In our architecture this problem can be solved with heuristic method. Profile is created from mobile device history; therefore an approximate application run can be predicted.

Processes can have dependencies to other tasks; processes that do not depend on each other on input or output data or on calculated result can be executed parallel without any preparation. It is more problematic when application has predictable dependence structure, and most problematic when this structure is changing dynamically. These problems can be also solved with profile creation, and harnessing usage information from it.

When certain application is not always running on one computer there is always a communication penalty. Efficient load balancer method must reduce this overhead to the greatest extent. Communication pattern extracted from profile can ease load balancing strategy.

In proposed resource management the scheduling strategy must be chosen. Based on information availability load balancing can be divided to following:

- Static scheduling
- Semi-static scheduling
- Dynamic scheduling.

At static scheduling all necessary information is available in advance that needs to be known for proper load balancing. For example CPU, memory, network etc. usage and load is available, and decision can be made in advance about which task where to run. This restriction seems very hard, but later it becomes clear that our proposal can deal with it. Algorithms in this case make decisions before the real execution starts.

At semi-static scheduling the restriction is not so strict; information must be available and known at certain well defined points (startup, each time step etc.). In this case offline algorithms can be used, between defined points of course.

At dynamic scheduling no a priori information is available; there is no opportunity to calculate anything in advance or to make shortcuts in load balancing calculation.

Let us examine first the static load balancing methods, as they are planned to be used in our architecture. Well known Round Robin Scheduling algorithm dispatches tasks to different resources in a rotating manner. It is a simple algorithm, with low calculating cost; neither load nor network connection status is considered. Generally speaking, this method does not take into account the current state of the resource; therefore bottlenecks may occur. Despite these facts round robin mode works quite effectively if resources are subject of load balancing that are equal in processing speed and memory.

Weighted Round Robin method tries to overcome some defects of Round Robin algorithm. It dedicates a weight to each resource; higher weight means higher calculation capacity. In some versions this weight is calculated in advance (Ratio member), in others it can be calculated runtime (Dynamic Ratio).

Dynamic load balancing algorithm takes into account resource's exact current load status, and based on that performs the load balancing. Weighted least connection is based on active network connection; the new task is balanced to the resource with less active connection. The connection can be substituted with other computing resource without algorithm change. The weighted form of this method has an extra attribute; a weight is given to resource, which can be also set statically or dynamically. With this weight a balancing strategy can be influenced. Other possible dynamic algorithm is Pick-Kx, what only considers resource's current load, regardless of its overall capacity.

Based on investigated load balancing algorithms a custom and adapted one is proposed; CPU utilization and performance need of the method should be kept low to be able to limit CPU usage, what has a strong connection with battery lifetime. In current architecture there are mobile devices and single cloud computing virtual machine. In short term, we do not plan to use the calculating capacity of nearby mobile devices; it will be a topic of further research. Also, exploring more than one cloud computing resource is not considered, it is also a topic of further study. These restrictions narrowed the problem space to two player; single mobile device and single cloud computing resource (virtual machine). From our previous study it can be seen that additional resources do not fundamentally change the resource management architecture and the load balancing algorithm.

Proposed resource management architecture is periodically collecting information about mobile device system state. On time base this information is collected: CPU load, memory usage, battery, network load. Network load is checked with DDMS and LogCat, CPU usage with built in function checking /proc/stat, memory with Debug.MemoryInfo(), battery status with BatteryManager. These tools are available on Android platform, and the usage of them is implemented into the resource management. So all data is available for dynamic load balancing from

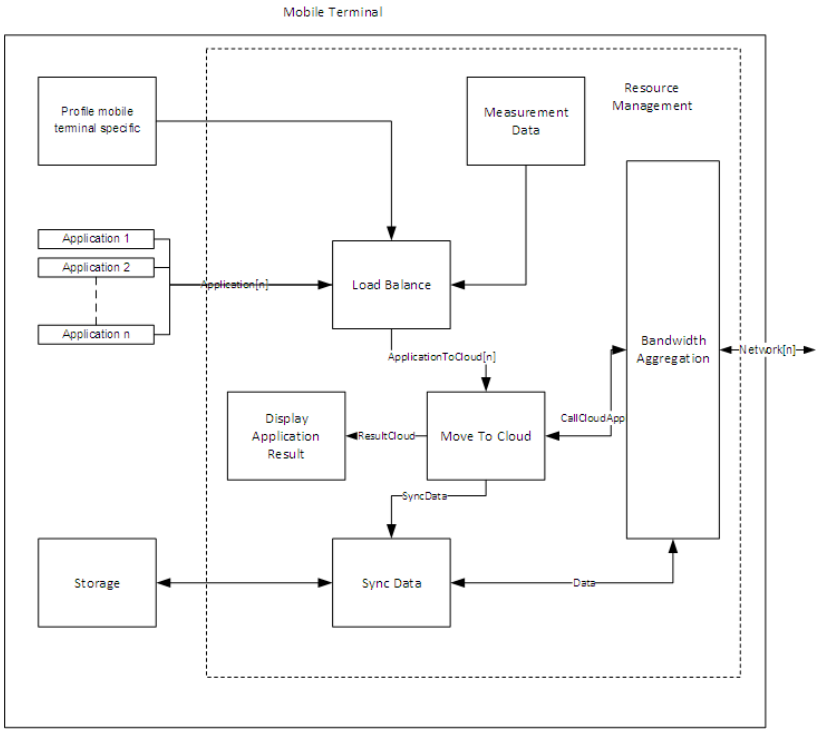

Figure 1: Resource management architecture

mobile device side. On the other hand, cloud computing resource status is not measured; compared to mobile device its performance is at least one magnitude higher. One thing is measured, the characteristics of connections.

Our load balancing method is a mixture of static and dynamic one; on cloud computing part no dynamic measurement is performed, on mobile device part every parameter is measured. As we seen earlier at static load balancing all necessary information must be available and known in advance. These conditions are partly met. Profile mechanism of the resource management collects information about usage of the mobile device. During the profile creation, the performance metrics of the certain mobile terminal is collected, and gathered. Profile collects and contains information about application usage history of the mobile terminal. Runtime of given application is checked, profile management block registers this time for every application executed on mobile device. Available network connections are also monitored on time and location base. Usually it can be predicted, that the user which application is going to execute, under what circumstances. Heuristic based on user profile allows calculating in advance the task moving strategy; computing power with battery life as well can be saved. Further advantage is that the algorithm and data collection can be simpler and shall be carried out less frequently; again this

Table 1: Measured execution time with different number of application running, with and without load balancing

Application count	No load balancing [s]	With load balancing [s]
1	32	33
2 (no profile)	68	55
2 (profile)	68	51

leads to battery saving. The fallback to the dynamic load balancing happens when user of mobile device abandons usual habits and places; the profile cannot contain such information, so new heuristic must be build. In such case, changed resources and environment must be investigated; CPU usage, memory, bandwidth etc must be checked. When this new environment is learned by profile, next time this will be again a simple static load balancing. Figure 1 shows the resource management architecture with load balance part.

4 Conclusion and Future work

The proposed load balancing strategy based on preliminary results is capable to effectively distribute load between mobile device and cloud computing resource. Measurements were focusing on idea and method validation. Our investigation showed that the mixture of static and dynamic load balancing can lead to satisfactory results; needed calculating capacity of proposed resource management can be kept low. Dynamic load balancing is used when profile cannot immediately adapt to changing environment.

Measuring arrangement can be quite complex; therefore simplifications were introduced, which may distort results. With variation of test setup and increasing numbers of measurement this can be decreased. So far limited number of measurement is available. These small results show that the resource management architecture can enhance the overall performance, see Table 1. The measurement is slightly a corner case; application with relative high calculating need was used during the measurements (elapsed time was measured between application start and stop). With this test setup overall performance of architecture was tested.

There were three basic scenarios; in first scenario one application was running on mobile device. In case of load balancing enabled only slight execution time increase was measured; resource management has a small calculation overhead. In second and third scenario two applications were running parallel (same application), the difference is only that in second scenario profile usage was disabled, in third scenario profile usage was enabled. It can be seen that application finished execution faster if resource management architecture with load balancing was used.

Application used for measurement is an artificial one (simple prime calculator); it is mainly CPU intensive, with high calculation cost; some IO utilization is added.

It is written in JAVA, using Android 4.2.2 platform (Snapdragon 400, 4x ARM Cortex-A7, 1Gib RAM). Cloud computing part: platform is Azure (medium virtual machine: 2 cores, 3.5 GiB memory), software written in JAVA, using jdk 7u72. Data that needs to travel between cloud resource and mobile terminal is limited; input parameters are below 1k bytes, result is below 1k bytes. Utilization of network layer was not in the focus, mainly the latency will influence the final result. To be able to benefit from profile, algorithm can continue its run from previous state; already calculated values are stored in a file. This file can travel and be synchronized between mobile device and cloud computing environment.

The goal of the load balancing layer is to enhance the usage of the mobile device. The task of the load balancer can be eased with profile; if a mobile device current position and performance surrounding is well known, static load balancing can be applied (weighted round robin). E.g. high quality Wi-Fi network is available, it is known which application will be be executed; these applications together with their needed input parameters can be prepared to be executed in cloud computing virtual machine or on mobile device. In our measurement this scenario leads to best performance, the calculation time was the shortest.

If the usage history of mobile device cannot be matched with previous profile, a fallback to dynamic load balancing happens. This has some drawbacks, current status of the device must be checked (active applications, network, battery status) and applications in the cloud cannot be prepared in advance. A weighted dynamic load balancing is picked in our implementation, weight is on CPU load, mobile device CPU load is expected to be less than on cloud computing. From measurement it can be seen that profile can enhance the mobile device performance.

Applications best suited for this hybrid architecture are CPU intensive applications. Applications should have all the necessary data available for processing locally. As we saw, with background synchronization taken place in areas with high networks bandwidth, identified from profile, the data availability can be increased, so more applications can be moved to cloud. Task that have high processing need, and limited number of result are ideal for our recommended hybrid architecture.

Energy consumption is the focus of another article. If application has a high CPU need on long term, or there is another CPU demanding application ready to run - it can be known from profile - sending data through wireless connection can have good results. It is also investigated in our another study how to aggregate network connections in mobile device.

Further tests will follow and needed to gather more data and to refine load balancing strategy proposed. It is planned to use the calculating capacity of nearby mobile devices; also, exploring and using more than one cloud computing resource should be considered.

References

[1] Bindu, P.L.H., Venkatesan, R., and Ramalakshmi, K. Perspective study on resource level load balancing in grid computing environments. In *Electronics*

Computer Technology (ICECT), 2011 3rd International Conference on, volume 6, pages 321–325, April 2011.

[2] 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.

[3] 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.

[4] Fekete, K., Csorba, K., Vajk, T., Forstner, B., and Pandi, K. Towards an energy efficient code generator for mobile phones. In *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on*, pages 647–652, Dec 2013.

[5] Gamblin, Todd, de Supinski, Bronis R., Schulz, Martin, Fowler, Rob, and Reed, Daniel A. Scalable load-balance measurement for spmd codes. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 46:1–46:12, Piscataway, NJ, USA, 2008. IEEE Press.

[6] 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.

[7] Hsiao, Hung-Chang and Chang, Che-Wei. A symmetric load balancing algorithm with performance guarantees for distributed hash tables. *IEEE Transactions on Computers*, 62(4):662–675, 2013.

[8] Hui, Chi-Chung and Chanson, S.T. Improved strategies for dynamic load balancing. *Concurrency, IEEE*, 7(3):58–67, Jul 1999.

[9] Kaur, Jaspreet. Comparison of load balancing algorithms in a cloud. In *International Journal of Engineering Research and Applications*, pages 1169–1173, May 2012.

[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] 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.

[12] Millsap, Cary. Thinking clearly about performance. *Queue*, 8(9):10:10–10:20, September 2010.

[13] Pándi, Krisztián and Charaf, Hassan. Network bandwidth aggregation based mobile resource management. In *4th IEEE International Conference on Cognitive Infococommunications - CogInfoCom 2013.*, CogInfoCom 2013., pages 608–612, Piscataway, NJ, USA, 2013. IEEE Press.

[14] Tong, Ruixia and Zhu, Xiongfeng. A load balancing strategy based on the combination of static and dynamic. In *Database Technology and Applications (DBTA), 2010 2nd International Workshop on*, pages 1–4, Nov 2010.

[15] Wang, Wanyuan and Jiang, Yichuan. Migration cost-sensitive load balancing for social networked multiagent systems with communities. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 127–134, Nov 2013.

[16] 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.