EFFECTIVE RESOURCE MANAGEMENT IN MOBILE CLOUD COMPUTING

[1]D.NAVEEN, [2]V KAVITHA
P.G. Student, Associate Professor
Computer Science, Sri SaiRam Engineering College,Chennai,India
Email: [1]nvn882@gmail.com, [2]kavitha.CSE@sairam.edu.in

Abstract— The aim is to achieve improve quality of mobile services. Computing and Radio Resources is a problem in Mobile Cloud Computing environment. Resource Management techniques provides resource reservation. Utilization is decrease and revenue will be increase for mobile cloud providers. In this paper the problem of the finite and rather small battery energy capacity in today's mobile devices has limited the functionality that can be integrated into these platforms or the performance and quality of applications that can be delivered to the users. Different mobile cloud providers cooperatively share the resource in resource pool to optimize the allocation of resource in order to satisfy the user demand and share the revenue in mobile cloud providers.

Index Terms— Mobile Cloud computing, Resource Pool, Cloud Service Provider (CSP).

I. INTRODUCTION

Mobile cloud computing, which integrates cloud computing into a mobile computing environment, is a new paradigm for supporting mobile users. Continued evolution of mobile systems including smart phones and tablet-PCs has resulted in ever more powerful yet more power hungry embedded systems with advanced functionality and high performance. Unfortunately, the increase in volumetric/gravimetric energy density of (rechargeable) Batteries has been much slower than the increase in the power Demand of these devices, resulting in a short battery life in these devices and a "power crisis" for the Smartphone technology development and product line expansion. Therefore, an effective solution to achieve a reasonable balance between the performance and power consumption of applications is required to ensure the overall service quality of the mobile devices.

To provide an alternate method of managing the applications in a mobile device, the concept of mobile cloud computing (MCC) system has been employed. The idea is to move the processing, memory, and storage requirements of some applications from the resource limited mobile devices to the resource unlimited cloud.

Resource management techniques such as resource reservation is a key approach to maintain the quality of service (QoS) Performance. This is particularly important for mobile cloud Computing applications with real-time requirements. Efficient Resource management methods are required to maximize the Utilization of the resources and thereby maximize the revenues of the mobile cloud service providers. To tackle this resource management problem and also to increase the revenue of mobile cloud service providers, a resource pool can be created. Specifically, multiple cooperative mobile cloud service providers can share their radio and computing resources in the pool. As a result, the resources that are not used by one service provider can be

used by other service providers, when required, and thereby, the resource Utilization can be increased.

II OPTIMIZATION MODELS

We formulate and solve three optimization models to obtain the optimal decisions on allocation of resources.

Linear programming formulation: We propose a basic optimization model for the cooperative mobile cloud service providers. We apply this model when all the resource allocation parameters are deterministic. Specifically, when the model observes the exact values of the system parameters, the model will make a decision on whether to support application Instances from users or not.

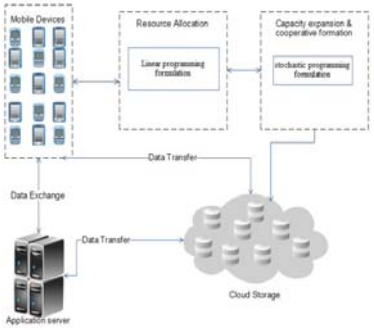
Stochastic programming formulation: We propose this model for the cases when the system parameters are random. The stochastic programming model requires the probability distributions of the random parameters (e.g., available resources and users' demand). The cooperative mobile cloud service providers can apply this model to make decisions in two stages. In the first stage, the providers make a decision to admit application instances based on the statistical information (e.g., probability distribution) about the available resources.

Robust optimization formulation: We apply this model when we only know the ranges of the values of the random parameters (e.g., resource requirements). The conservativeness of the solution from the robust optimization model is adjustable, giving flexibility to the Resource allocation to mobile applications.

We propose a game model for cooperation formation among mobile cloud service providers to decide whether they should cooperate and create the resource pool or not. The solution of the game model is a stable cooperation strategy (or Nash equilibrium) which ensures that the rational mobile cloud service providers cannot unilaterally change their decisions. In addition, the providers can decide on the amount of resources to contribute to the resource pool. This is referred to as the capacity expansion for which we analyze the stable strategy (or Nash equilibrium).

The proposed framework provides multiple cooperative mobile cloud service providers can share their resources. In fact single request handles multiple processes. This is based on well optimized resource management process and their own benefits Given energy consumption is reduced and computation of mobile device and cloud server decreased.

III. MOBILE CLOUD COMPUTING MODEL



Mobile cloud computing model

The Mobile devices can access the cloud based on the Programming formulation. There are two levels we can access the cloud. In first level allocating the resource to the cloud. The data is send only bit type not character format. In second level once the data is sending means compressing that data in capacity expansion based on stochastic programming. Finally we can retrieve the original data with less size.

We consider the MCC environment as shown in .The users can download the mobile applications from an application server. The mobile application in the MCC is divided into two parts, i.e., local computing modules and remote computing modules. The local Computing modules run-on a mobile device and the remote computing modules run on a computing server in a data center. The data transfer between local and remote computing modules of the mobile applications requires a wireless access network and a wired network. Specifically, the wireless access network is for the communication between the mobile device and the wireless base station. The wired network is for the communication among the wireless base station, application server, and computing server. Running the mobile applications needs the radio resource (i.e., bandwidth) and computing resources (e.g.,

memory and CPU of a server) and they have to be reserved. We assume that the wired network has abundant bandwidth and reservation is not required. Nevertheless, the model can be extended easily to consider bandwidth reservation in the wired network as well. In the MCC environment, when a user wants to run mobile applications, she will send a request to the application server through wireless and wired connections.

The application server contacts the base station and data center to obtain the radio and computing resources. If the available radio and computing resources are sufficient, the user can run the mobile application. The remote computing modules will be downloaded and run on the computing server in the data center for that user.

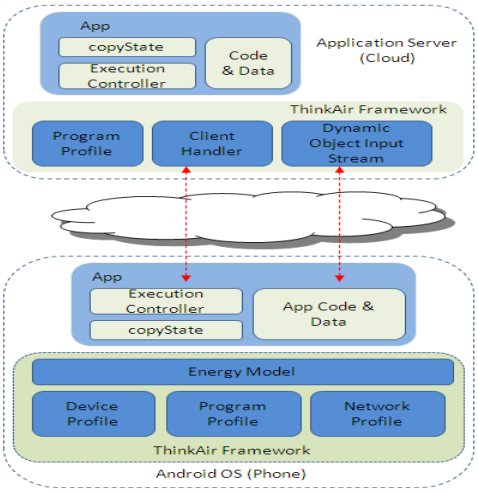
IV DESIGN GOALS AND FRAMEWORK

The design of Mobile cloud computing is based on some assumptions which we believe are already, or soon will become, true:(1) Mobile broadband connectivity and speeds continue to increase, enabling access to cloud resources with relatively low Round Trip Times (RTTs) and high bandwidths; (2) As mobile device capabilities increase, so do the demands placed upon them by developers, making the cloud an attractive means to provide the necessary resources; and (3) Cloud computing continues to develop, supplying resources to users at low cost and on-demand. We We reflect these assumptions in Mobile cloud computing through four key design objectives.

(i) Dynamic adaptation to changing environment.
As one of the main characteristics of mobile computing environment is Rapid change, Think Air framework should adapt quickly and Efficiently as conditions change to achieve high performance As well as to avoid interfering with the correct execution of Original software when connectivity is lost.
(ii) Ease of use for developers. By providing a simple interface for developers, Think Air eliminates the risk of misusing the framework and accidentally hurting performance instead of improving it, and allows less skilled and novice developers to use it and increase competition, which is one of the main driving forces in today's mobile application market.

(iii) Performance improvement through cloud computing. As the main focus of Think Air, we aim to improve both Computational performance and power efficiency of mobile devices by bridging smart phones to the cloud. If this bridge becomes ubiquitous, it serves as a stepping stone towards more sophisticated software.

(iv) Dynamic scaling of computational power. To satisfy the customer's performance requirements for commercial grade service, Think Air explores the possibility of dynamically scaling the computational power at the server side as well



Overview of Mobile cloud computing

V. COMPILATION AND EXECUTION

In this section we describe in detail the process by which a developer writes code to make use of Mobile cloud computing, covering the programmer API and the compiler, followed by the execution flow.

A. Programmer API
Since the execution environment is accessed indirectly by the developer, Think Air provides a simple library that, coupled with the compiler support, makes the programmer's job very straightforward: any method to be considered for offloading is annotated with remote

This simple step provides enough information to enable the Think Air code generator to be executed against the modified code. This takes the source file and generates necessary remote able method wrappers and utility functions, making it ready for use with the framework - method invocation is done via the Execution Controller, which detects if a given method is a candidate for offloading and handles all the associated profiling, decision making and and communication with the application server without the developer needing to be aware of the details.

B. Compiler

A key part of the Think Air framework, the compiler comes in two parts: the Remote able Code Generator and the Customized Native Development Kit (NDK). The Remote able Code Generator is a tool that translates the annotated code as described above. Most current mobile platforms provide support for execution of native code for the performance critical parts of applications, but cloud execution tends to be on x86 hosts, while most Smartphone devices are ARM-based, therefore the Customized NDK exists to provide native code support on the cloud.

C . Execution Controller

The Execution Controller drives the execution of remote able methods. It decides whether to offload execution of a particular method, or to allow it to continue locally on the phone. The decision depends on data collected about the current environment as well as that learnt from past executions.

When a method is encountered for the first time, it is unknown to the Execution Controller and so the decision is based only on environmental parameters such as network quality: for example, if the connection is of type WiFi, and the quality of connectivity is good, the controller is likely to offload the method. At the same time, the profilers start collecting data. On a low quality connection, however, the method is likely to be executed locally.

D. Execution flow

On the phone, the Execution Controller first starts the Profilers to provide data for future invocations. It then decides whether this invocation of the method should be offloaded or not. If not, then the execution continues normally on the phone. If it is, Java reflection is used to do so and the calling object is sent to the application server in the cloud; the phone then waits for results, and any modified local state, to be returned. If the connection fails for any reason during remote execution, the framework falls back to local execution, discarding any data collected by the profiler. At the same time, the Execution Controller initiates asynchronous reconnection to the server.

If an exception is thrown during the remote execution of the method, it is passed back in the results and re-thrown on the phone, so as not to change the original flow of control.

V1. Application Server

The Mobile cloud computing Application Server manages the cloud side of offloaded code and is deliberately kept lightweight so that it can be easily replicated. It is started automatically when the remote Android OS is booted, and consists of three main parts, described below: a client handler, cloud infrastructure, and an automatic parallelization component.

A. Client Handler

The Client Handler executes the Think Air communication protocol, managing connections from clients, receiving and executing offloaded code, and returning results.

To manage client connections, the Client Handler registers when a new application, i.e., a new instance of the Think Air Execution Controller, connects. If the client application is unknown to the application server, the Client Handler retrieves the application from the client, and loads any required class definitions and native libraries. It also responds to application level ping messages sent by the Execution Controller to measure connection latency.

Following the initial connection set up, the server waits to receive execution requests from the client. A request consists of necessary data: containing object, requested method, parameter types, parameters themselves, and a possible request for extra computational power. If there is no request for more computational power, the Client Handler proceeds much as the client would: the remote able method is called using Java reflection and the result, or exception if thrown, is sent back. There are some special

cases regarding exception handling in Think Air, however. For example, if the exception is an Out Of Memory Error, the Client Handler does not send it to the client directly; instead, it dynamically resumes a more the task to it. In the case that the client asks for more clones to powerful clone (a VM), delegates the task to it, waits for the result and sends it back to the client. Similarly, if the client explicitly asks for more computational power, the Client Handler resumes a more powerful clone and delegates the the task to it. In the case that the client asks for more clones to execute its task in parallel, the Client Handler resumes needed clones, distributes the task among them, collects and sends results back to the client. Along with the return value, the Client Handler also sends profiling data for future offloading decisions made by the Execution Controller at the client side.

B. Cloud Infrastructure

To make the cloud infrastructure easily maintainable and to keep the execution environment homogeneous, e.g., w.r.t. the Android-specific Java bytecode format, we use a virtualization environment allowing the system to be deployed where needed, whether on a private or commercial cloud. There are many suitable virtualization platforms available, e.g., Xen , QEMU , and Oracle's VirtualBox. In our evaluation we run the Android x86 port 2 on VirtualBox . To reduce its memory and storage demand, we build a customized version of Android x86, leaving out unnecessary components such as the user interface and built-in standard applications.

Our system has 6 types of VMs with different configurations of CPU and memory.The VM manager can automatically scale the computational power of the VMs and allocate more than one VMs for a task depending on user requirements. The default setting for providers to support users to run application instances utilizing resources in the pool.

computation is only one VM with 1 CPU, 512MB memory, and 100MB heap size, which clones the data and applications of the phone and we call it the primary server. The primary server is always online, waiting for the phoneto connect to it. The second type of VMs can be of any configuration which in general does not clone the data and applications of a specific phone and can be allocated to any user on demand - we call them the secondary servers. The secondary servers can be in any of these three states: powered-off, paused, or running. When a VM is in powered-off state, it is not allocated any resources. The VM in paused state is allocated the configured amount of memory, but does not consume any CPU cycles. In the running state the VM is allocated the configured amount of memory and also makes use of CPU.

VII. USER CENTRIC MOBILE CLOUD COMPUTING

The next-generation MCC applications demand tight integration of the physical and virtual functions running on the mobile devices and cloud servers, respectively. Moreover, due to the mobility of mobile users and changes in the application running environment, the MCC application functions are not fixed on their running hosts.

VIII. CONCLUSION

Mobile applications will require the computing resources of mobile devices as well as the servers in a data center in the cloud Therefore, the mobile cloud service providers have to provide both radio and computing resources and jointly optimize them to achieve the maximum revenue For revenue management (i.e., to divide the revenue obtained from the resource pool among cooperative providers) the concepts of core and Shapley value from the cooperative game theory have been applied. Optimization models is used to determine the decision of

REFERENCES

[1] Rakpong Kaewpuang, Dusit Niyato, *Member, IEEE*, Ping Wang, *Member, IEEE*, and Ekram Hossain, *Senior Member, IEEE* "**A Framework for Cooperative Resource Management in Mobile Cloud Computing**" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 31, NO. 12, DECEMBER 2013

[2] Shuang Chen, Yanzhi Wang, Massoud Pedram " **A Semi-Markovian Decision Process Based Control Method for Offloading Tasks from Mobile Devices to the Cloud**" Globecom 2013 - Symposium on Selected Areas in Communications pg no 2885-2890.

[3] K. Kumar and Y. Lu, "**Cloud computing for mobile users: Can offloading computation save energy?,**" Computer, vol. 43, no. 4, pp. 51–56, 2010.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "**A survey of mobilecloud computing:Architecture, applications, and approaches,**" Wireless Communications and Mobile Computing (WCMC), 2011. Z. Li, C. Wang, and R. Xu, "Computation offloading to save

[5] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "**A game theoretic resource allocation for overall energy minimization in mobile cloud computing system**," in Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design, ser. ISLPED '12. NewYork, NY, USA: ACM, 2012, pp. 279–284.

[6] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic**,"Adaptive offloading inference for delivering applications in pervasive computing environments,**" in Per Com, 2003, pp. 107–114.

[7] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: "**Elastic execution between mobile device and cloud**". In Proc. of EuroSys, 2011