Review and Implementation of Power Saving and Time Computation in Handheld Devices Using TDM

Ms. Sayali A. Sarode[1,] , Ms. Prerana P. Satale[2] , Ms. Ragini S. Donde[3], Ms. Tejashwini R. Hadpe[4], Prof. Anand N. Gharu[5]

[1] UG Students, Department of Computer Engineering, P. V. G. College of Engineering, Nashik
[2] UG Students, Department of Computer Engineering, P. V. G. College of Engineering, Nashik
[3] UG Students, Department of Computer Engineering, P. V. G. College of Engineering, Nashik
[4] Assistant Professor, Department of Computer Engineering, P. V. G. College of Engineering, Nashik

ABSTRACT

Running sophisticated software on smart phones may be result in poor performance and shortened battery lifetime because of their limited assets. lately, offloading computation workload to the cloud has become a auspicious solution to enhance both performance and battery life of smart phones. After all, it also consumes both time and energy to upload data or programs to the cloud and recall the results from the cloud. In this paper, we develop an offloading framework, named Ternary Decision Maker (TDM), which intent to shorten response time and reduce energy consumption at the same time. our targets of execution include an on-board CPU, an on-board GPU, and a cloud, all of which combined provide a more flexible execution environment for mobile applications. We conducted a real-world application, i.e., matrix multiplication, in order to evaluate the performance of TDM. Accordingly our experimental results, TDM has less faulty offloading decision rate than existing methods.Additionaly, by offloading modules, our method can achieve.

Keyword: - *Offloading, Ternary Decision Maker, Android, cloud computing, computation offloading.*

1. INTRODUCTION

EARLY 300 million smart phones were sold in 2010, and its number is expected to increase by 80% in 2011 [1]. In order to satisfy the needs of billions of users, smart phones feature versatile mobile applications. Examples of the latest functions include multimedia, real-time games, GPS navigation, and communication. Most of these mobile applications are user-interactive and data-processing intensive, both of which require quick response and long battery life. However, most commercial off-the-shelf smart phones, compared with desktops, are generally equipped with low-speed processors and limited-capacity batteries. Running elegant software on smart phones can result in poor performance and shorten battery lifetime. Therefore, it becomes a crucial issue in designing smart phones to deliver adequate performance and prolong battery life. A lot of advanced hardware technologies, such as instruction-level parallelism, leakage power control, and dynamic voltage scaling, have been proposed to improve processor speed and reduce energy consumption. Although advanced technology can deliver better performance, adopting high-end processors is not always appropriate for budget limited projects. Lately, cloud computing has become another possible solution to enhance the computing capability of smart phones. The cloud computing vendors provide computing cycles for the registered users to reduce computation and energy consumption of smart phones, such as Amazon Elastic Compute Cloud (EC2), Amazon Virtual Private Cloud (VPC), and PacHosting. It takes both time and energy to upload data or programs to the cloud and retrieve the results from the cloud. The computation capacity of the cloud can also affect the total execution time.There is a clear need for the development of a decision-making mechanism before offloading, in order to save both time and energy consumption.

2. LITERATURE SURVEY

Adaptive Offloading for Pervasive Computing Gu,X, Nahrstedt, K et. al It is challenging task to delivering a complex application on a resource constrained mobile device. A daptive offloading system enables dynamic partitioning of the application and efficient offloading of part of its execution to a nearby surrogate. This pervasive services without any modification of an application or degrading its fidelity, we propose an adaptive offloading system that includes two key parts namely a distributed offloading platform and an offloading inference engine. There are two important decision-making problems for adaptive offloading: adaptive offloading triggering and efficient application partitioning.[1]

Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme Rong fu,Cheng Wang, Zhiyuan Li et. al. They consider handheld computing devices which are connected to a server (or a powerful desktop machine) via a wireless LAN. Some devices are often possible to save the energy on the handheld computing devices by offloading its computation to the server. In this work, based on profiling information on computation time and data sharing at the level of procedure calls, we construct a cost graph for a given application program. Then we apply a sutaible partition scheme to statically divide the program into server tasks and client tasks such that the energy consumed by the program is minimized. Experiments are performed on a suite of multimedia benchmarks. Results show considerable energy saving for several programs through offloading.[2]

An Effective Offloading Middleware for Pervasive Services on Mobile Devices Khung Yang, jie Zhang et. al. The practical success of pervasive services running in mobile wireless networks and devices relies on their ability to provide effective and efficient offloading support, so as to satisfy the increasing demand for mobile devices to run heavier applications (e.g. those running on desktop PCs). Offloading is an effective mechanism for leveraging the severity of resource constrained mobile devices by migrating some computing load to nearby resource-rich surrogates (e.g. desktop PCs, servers) on home networks or their extension. This paper proposes a light-weight and efficient offloading middleware, which provides runtime offloading services for resource constrained mobile devices. The middleware considers multiple types of resources (i.e. memory, CPU and bandwidth) and carries out application partitioning and partition offloading in an adaptive and efficient manner. The corresponding algorithms are presented. The evaluation outcomes indicate the effectiveness and efficiency of this service offloading solution. [3]

Studying Energy Trade Offs in Offloading Computation or Compilation in Java-enabled Mobile Devices G. Chen, Kang, B.T,Kandamir,M et. al. Java-enabled wireless devices are preferred for various reasons. For example, users can dynamically download Java applications on demand. The dynamic download capability supports extensibility of the mobile client features and centralizes application maintenance at the server. Also, it enables service providers to customize features for the clients. In this work, we extend this client-server collaboration further by offloading some of the computations (i.e., method execution and dynamic compilation) normally performed by the mobile client to the resource-rich server in order to conserve energy consumed by the client in a wireless Java environment. In the proposed framework, the object serialization feature of Java is used to allow offloading of both method execution and bytecode-to-native code compilation to the server when executing a Java application. Our framework takes into account communication, computation, and compilation energies to decide where to compile and execute a method (locally or remotely), and how to execute it (using interpretation or just-in-time compilation with different levels of optimizations). As both computation and communication energies vary based on external conditions (such as the wireless channel state and user supplied inputs), our decision must be done dynamically when a method is invoked. Our experiments, using a set of Java applications executed on a simulation framework, reveal that the proposed techniques are very effective in conserving the energy of the mobile client. [4]

Cloud Computing for Mobile Users: Can offload-ing Computation Save Energy Kumar.K, Yung-Hsiang Lu et. al. The cloud heralds a new era of computing where application services are provided through the Internet. Cloud computing can enhance the computing capability of mobile systems, but is it the ultimate solution for extending such systems' battery lifetimes [5].

3. SYSTEM MODEL

Mobile applications usually adopt frequently used modules to process data the application first involves a module to process data with the size of N input, which is stored in memory. The data are then processed by CPU, co-processor, or cloud. Finally, the processed data with the size of N output are written back to memory. Assuming that Tcpu denote the execution time if the module is executed on a local CPU. We divide Tcpu into two parts. The first part is transmission time ttrans, which is used for fetching data from and writing them back to memory. Then the second part is pure computation time tcomp, then it is consumed by the CPU to execute codes. In addition, the bandwidth of memory access is mem, at which the data are read from or written into the memory by CPU, co-processor, or network interface. If the data processing is offloaded to the cloud, the network transmission speed is B. In addition, the power consumption of CPU is Pcpu, of co-processor is Pcop, and of network interface is Pnic. When the system is idle, its power consumption is Pbasic. Fig 1. Give an overview of our offloading framework, i.e., TDM, and introduce the mechanism of creating and updating factor tables. We then introduce the equations used a module when it is executed on local CPU, local GPU, or cloud. Finally, we describe the methodology of ternary decision making. Our TDM is a daemon that runs as a background process. The TDM includes two parts: factor measurement and ternary decision making. Before a system starts to run, we measure mem, Pbasic, Pcpu, Pcop, and Pnic. First we call, static factor is decision factor because it is deterministic and module independent.

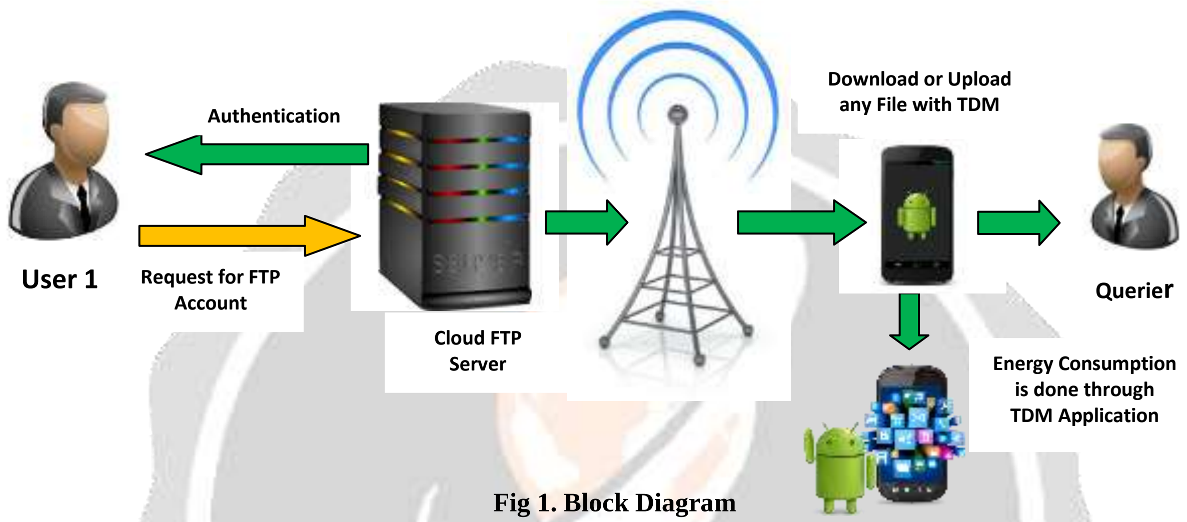
Authentication
User 1
Request for FTP Account
Cloud FTP Server
Download or Upload any File with TDM
Querier
Energy Consumption is done through TDM Application

Fig 1. Block Diagram

At runtime, when a module is invoked, TDM first checks the existence of the associated factor table of the module, which records necessary parameters to estimate energy consumption and execution time. In such case, if the factor table does not exist, the module will be executed on CPU directly, and their corresponding factor table will be created. TDM adds static decision factors, such as mem, Pbasic, Pcpu, Pcop, Pnic, cpu, and cop, to the factor table.

4. EXPERIMENT AND EVALUATION

Any CPU-intensive module, which has been widely used in the applications of encoding, decoding, image compression, and rotation. In order to evaluate the effectiveness of TDM for any offloading target applications. One is for the execution of local CPU; the other two are for local GPU (i.e., coprocessor) and the cloud. In the version of local CPU, it includes three steps: reading data, processing the multiplication, and storing the results. The Fig 2. Shows result of TDM chart.

5. CONCLUSIONS

We implemented a decision framework for better computation offloading. The decision is purely depends on estimated execution time and energy consumption. Our aim to save both execution time and energy consumption at the same time. Unlike previous works, which consider only binary decisions, our ternary decision is suitable for multiple offloading targets. In our experiment, we presented a case study to validate the applicability in different situations. Based on our decision framework, the any module tends to be offloaded to more powerful processors, such as local GPU or cloud. By offloading modules, we will achieve, at most, approximate 75.

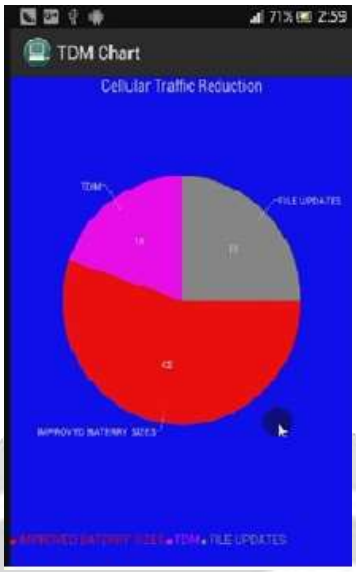
TDM Chart
Cellular Traffic Reduction
TDM
FILE UPDATES
IMPROVED BATTERY SIZES
IMPROVED BATTERY SIZES ● TDM ● FILE UPDATES

Fig. 2 TDM Chart

6. REFERENCES

[1]. X. Gu, A. Messer, I. Greenberg, D.Milojicic, and K. Nahrstedt, Adaptive offloading for pervasive computing, IEEE Pervasive Comput., vol. 3, no. 3, pp. 6673, Jul./Sep. 2004.
[2]. Z. Li, C. Wang, and R. Xu, Computation offloading to save energy on handheld devices: A partition scheme, in Proc. 2001 Int. CASES, 2001, pp. 238246.
[3]. S. Ou, K. Yang, and J. Zhang, An effective offloading middleware for pervasive services on mobile devices, Pervasive Mobile Comput., vol. 3, no. 4, pp. 362385, Aug. 2007.
[4]. G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and R. Chandramouli, Studying energy tradeoffs in offloading computation/ compilation in java-enabled mobile devices, IEEE Trans. Parallel Distrib. Syst., vol. 15, no. 9, pp. 795809, Sep. 2004.
[5]. K. Kumar and Y. Lu, Cloud computing for mobile users: Can offloading computation save energy Computer, vol. 43, no. 4, pp. 5156, Apr. 2010.