A Computation Offloading Framework to Optimize Energy Utilisation in Mobile Cloud Computing Environment

Nitesh Kaushik
Computer Science and Engg. Department,
DCRUST, Murthal

Jitender Kumar
Computer Science and Engg. Department,
DCRUST, Murthal

ABSTRACT

Newly emerged computing concept Mobile cloud computing, is a combination of mobile computing and cloud computing. Mobile Cloud Computing (MCC) enables mobile applications to get built, powered and hosted using cloud resources. As few years back mobile devices were merely used for making calls but nowadays enormous applications can be run on top of the mobile devices. Mobile systems, such as smart phones, have become primary computing platform for users. Still there are some challenges like battery life, computation time etc that resists from implementing applications that are computation intensive. Several approaches have been proposed for addressing these problems. In this paper, we focused on augmented execution of mobile applications on cloud resources, more often known as offloading and formulate the partitioning of elastic mobile datastream applications as on optimization problem by minimizing the cost function which is combination of Communication energy and computation energy. We further investigated the offloading problem by considering the Service Level agreement (SLA) negotiated maximum waiting time. Genetic algorithm is used to find the optimum offloading solution and the results are evaluated by simulating our approach and comparing it with the all mobile-side execution and all cloud-side execution.

Keywords

Mobile Cloud Computing, Datastream applications, Elastic applications, partitioning.

1. INTRODUCTION

According to NIST, Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction[1]. Cloud computing is a computing paradigm that enables users to make use of the cloud resources like processing, memory and storage that are not physically present at the users location. Arrival of 3G and 4G Technologies has fuelled the Cloud Computing market in the area of Mobile Computing and that is why the MCC approach is gaining popularity. Offloading has gained attention among mobile cloud computing researchers, and various studies by L.Yang [12], K.Kumar [13], Huerta-Canepa [6], Ricky K.K. [9] have proved the utility of Cloud Computing in the area of MCC. Although Mobile devices of current generation have powerful processors and other resources, but still battery backup of such devices is a bottleneck. So to encounter these problems many approaches has been proposed like CloneCloud[14] execution by and two tier Cloud approach by M.Satyanaryan [10]. We have focussed on augmenting smart phone applications on the Cloud Platforms. In this paper, an energy-aware approach for partitioning and offloading execution has been proposed considering the Service Level negotiated time.

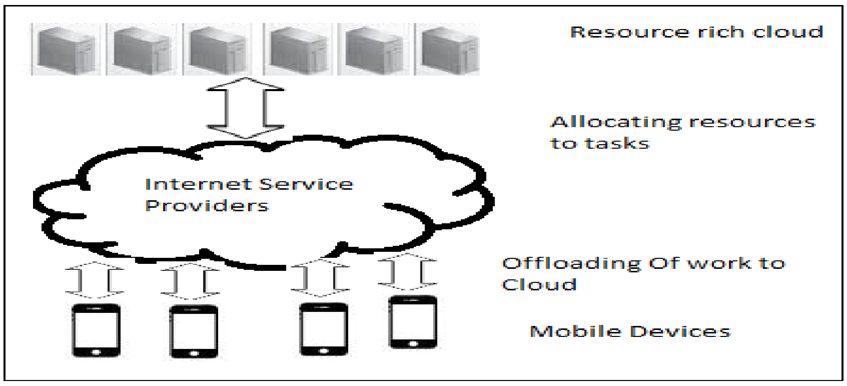

Resource rich cloud
Allocating resources to tasks
Internet Service Providers
Offloading Of work to Cloud
Mobile Devices

Fig 1. Mobile cloud computing architecture

This framework consists of runtime systems that support adaptive partitioning and distributed execution of the data stream applications. In order to achieve better energy performance, computation intensive and energy consuming components are being offloaded to cloud. Our contributions in this paper are:

- Partitioning framework has been proposed for mobile data stream applications to offload partitions to cloud using constraint energy.
- A genetic algorithm is designed to run at cloud side for optimal partitioning of elastic datastream applications based on the various parameters profiled online.
- Components of application will be divided into two groups, one group runs locally, the other group runs at cloud side based on cost function.

In rest of the paper, we first present the related works in the Section II. Then we describe our model and present the formula to partition problem. in section III. In section IV implementation has been designed. In section V and section VI, we present numerical model and performance evaluation. In section VII we have concluded our study with future work. In last we concluded our paper with references in section VIII.

2. RELATED WORK

Various studies have identified longer battery lifetime as the most desired concern of mobile systems[2],[3],[5]. Many applications are too computation intensive to perform on a mobile system. So offloading these applications to cloud side is a good solution. But there must be some criteria for deciding which part of any application should be offloaded and which has to be performed locally. Over the last few years researchers have proposed approaches in which either mobile device users are working collaboratively to get the better performance or offloading of the work has been done to the cloud.

Studies by Huerta-Canepa [6], J. Kangasharju [7], A.Berl [8],proposed solutions for solving a common problem by a network of mobile devices. Results have shown that execution time as well as energy consumed can be enhanced using these approaches. These architectures are acceptable when number of devices want to do the same task. Given task can be divided into a number of components and with some conditions can be allotted to all devices in that network. The limitations of these approaches are that devices should remain connected till the work completes. Devices overhead also increases in monitoring. Also sometimes work cannot be broken into a number of tasks that can be distributed among all mobile devices in that network. In that situation work should be offloaded to a resource rich environment. Also there security is the biggest concern in these systems.

Ricky K.K. [9], proposed the migration of application level java thread that performs Stack-On-Demand asynchronous execution (SOD_AE) to migrate tasks to the cloud. In this approach, the system does not migrate whole threads or processes among mobile devices and cloud nodes. This design exploits the temporal locality of stack-based execution, in which the most recent execution state always is on the top segment of a stack. However this approach supports task migration at application level. But still some application parts are there that can produce better results when performed at mobile side instead of sending to cloud side, such as image retrieval, voice recognition and navigation. So there must be some criteria to group some tasks that must be migrated to the cloud and others should be executed at mobile side for better results. Works related to decrease latencies caused by internet is also proposed. M.Satyanarayan [10] advocates to have a tiered approach for offloading computation to nearby Cloudlets. These cloudlets are resource rich platform in comparison to smartphones, and can be used by the nearby devices using Wi-Fi for better results and less latencies. A somewhat similar work is carried by E. Marinelli [11] , suggests a platform of some mobile nodes, in which master is deployed on a Personal Computer(Resource-rich platform) and smartphones to work as slaves. Although this architecture shows better performance for data processing but are not good enough for scaling demands.

K. Kumar [13], has argued that whether offloading computation can be a better option. Sending computation to another machine is not a new concept. Various cost studies focus on whether to offload computation to a server. In these studies cost functions consists of cost of transfer and cost of execution. The generalised result can be stated as that the amount of data to be exchanged among devices and the bandwidth provided are the deciding factor. L. Yang [12], provides a mechanism to decide which component should be offloaded to the cloud for better throughput. A formula for throughput is proposed that decides the throughput on the basis of execution time but missed the energy factor.

A data has been provided in the table below that shows the applications with their resource requirements when executing. As from the table below it can be seen that applications like face recognition, video streaming and Augmented reality needs more computation resources as well as energy.

Table 1. Applications with their resource requirement

Applications	Compute intensity	Network bandwidth	Network latency
Web mail	Low	High	High
Social networking	Low	Medium	Medium
Web browsing	Low	Low	High
Online gaming	High	Medium	Low
Augmented reality	High	Medium	Low
Face recognition	High	Medium	Low
HD video streaming	High	Medium	Low

3. OUR APPROACH

3.1. System model

Let the application have n number of components, denoted by C_1 C_2 … C_n . Each component has several properties like number of CPU instructions required to execute component, total data to migrate to the cloud for further execution, power needed per instruction if executed at mobile side, power needed per unit instruction while offloading to the cloud and power consumed while in idle state. Also at the time of partitioning decision, bandwidth of the network and mobile device's resources also required.

We have following assumptions in our model:

- All components running on the mobile side will have equal resources.
- The input data of the data stream application is acquired from the sensors on the mobile device, and output data should also be delivered to the mobile device.
- At cloud side, VMs with unique functionality are created so that components require same functionality will be offloaded to that VM.
- Middleware which runs at Cloud side as well as mobile side guides the partitioning.

3.2. Our Middleware

Resource Manager: This module monitors required properties of mobile device like bandwidth of the network, resources at mobile side and power consumption per unit time by various mobile device hardware. Then this module forwards these properties to resource monitor running at middleware to generate optimal partition.

Local Execution Manager: This module monitors execution of components running locally at mobile side and after completion provides result for further execution.

Offload Manager: Offload manager transmits the data to sequential execution tracker running at middleware, to further direct data to particular component that has been created at cloud side to perform unique operation. After execution completes at cloud VM, receiving of data at mobile device is responsibility of this module.

Sequential Execution Tracker: Main purpose of this module is to keep track of elastic execution and synchronization of components. This module communicates with Cloud offload manager to offload some specific component to suitable VM.

Resource Monitor: Resource monitor receives data from Resource manager running at mobile side and sends data to application behaviour generator which according to these values, break a particular application into components. That estimation of execution energy and transmission energy can be defined for all components.

Optimization solver: Optimization solver uses output from Application behaviour generator and generates an optimal partition so that the energy consumed in execution of a particular application is minimized. This optimal partition is forwarded to Sequential execution tracker running at middleware and also to Sequential execution tracker at mobile device for better synchronization.

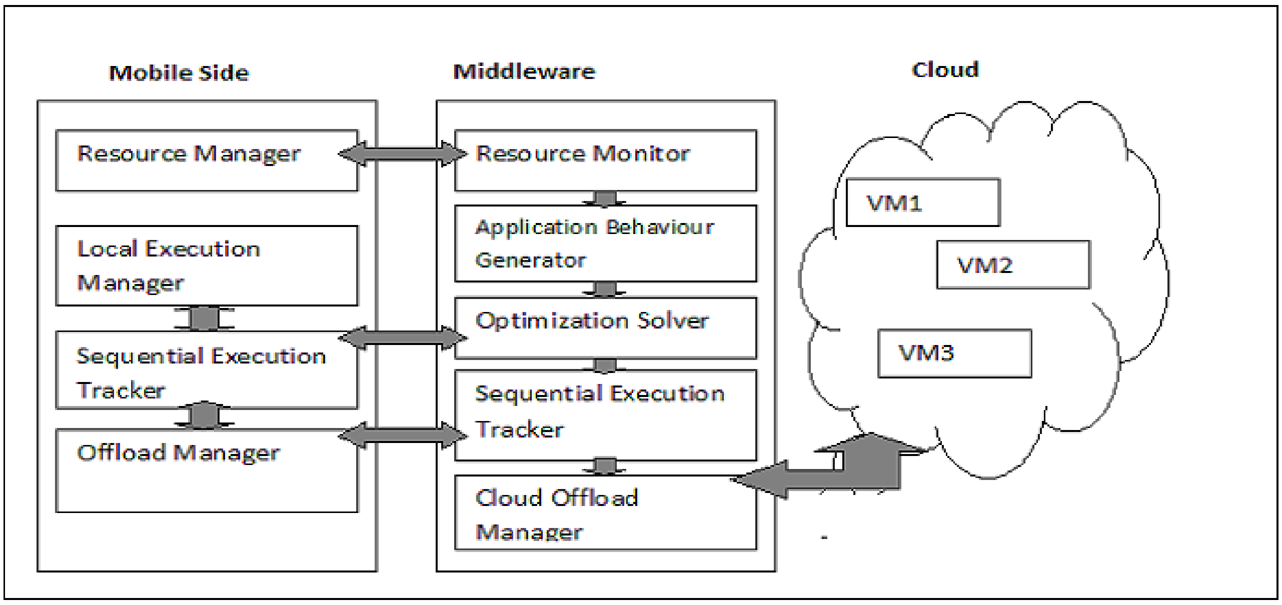


Fig 2. Our Middleware

Application Behaviour Generator: This module takes input from Resource Monitor and uses those values to break an application into a number of elastic modules so

Cloud Offload manager: Cloud offload manager gets migration data from sequential execution tracker and offload to a suitable VM according to the functionality required by component. It communicates to VMs and is responsible for offloading data and fetching output to middleware.

3.3 Problem Formulation

Given all components of dataflow application with their properties, wireless network characteristics (e.g. bandwidth) and mobile resources (e.g. energy and computing resources), partitioning will be performed. The partitioning problem is a process of dividing all components into two groups, one for offloading to the cloud and other one to execute locally keeping energy and time as constraints.

The formula for optimizing problem is as below.

$$\text{Energy} = \min \left(\sum E_{execution} + \sum E_{offloading} + \sum E_{idle} \right) \quad (1)$$

where $E_{execution(i)} = dec_i * ((I_i / speed_{local}) * P_{execution})$,

$E_{offloading(i)} = (1- dec_i) * (Data_{mig(i)} / bandwidth) * P_{offloading}$

and $E_{idle(i)} = (1- dec_i) * (I_i / speed_{cloud}) * P_{idle}$

We introduce dec_i for component i, which indicates whether the component i is executed locally (for $dec_i = 1$) or remotely (for $dec_i = 0$). The resulting string dec_1, $dec_2 \ldots dec_n$ represents the required partitioning of the application. The variable mob_{res} is the status of resources at mobile side, and I_i refers to number of instructions required to execute that component. $Data_{mig}$ shows the total data that will be migrated for execution of a component at cloud side. $P_{exe(i)}$, $P_{off(i)}$ and $P_{idle(i)}$, shows the power needed for executing a unit instruction of that component at mobile side, power needed to offload unit data to the cloud and power dissipated by mobile device at idle time respectively.

4. IMPLEMENTATION

Since offloading execution to cloud from mobile device is to enhance throughput and energy consumed. Results in various researches have shown [13], [10] that offloading to more resource-rich platform is beneficial if energy consumed to execute locally is more than energy needed to transfer that component.

$$E_{execution(i)} - E_{transfer(i)} > 0 \qquad (2)$$

Total energy consumed in execution will be sum of energies consumed by all the components whether executed locally or locally.

Energy consumed in offloading components to cloud is

$$E_{offload(i)} = E_{transfer(i)} + E_{idle(i)} \qquad (3)$$

Where $E_{transfer(i)} = Ptransfer*Data_{mig(i)} / Bandwidth$
and $E_{idle(i)} = P_{idle}*(I_i / speed_{cloud)}$

And total energy consumed in executing components locally is

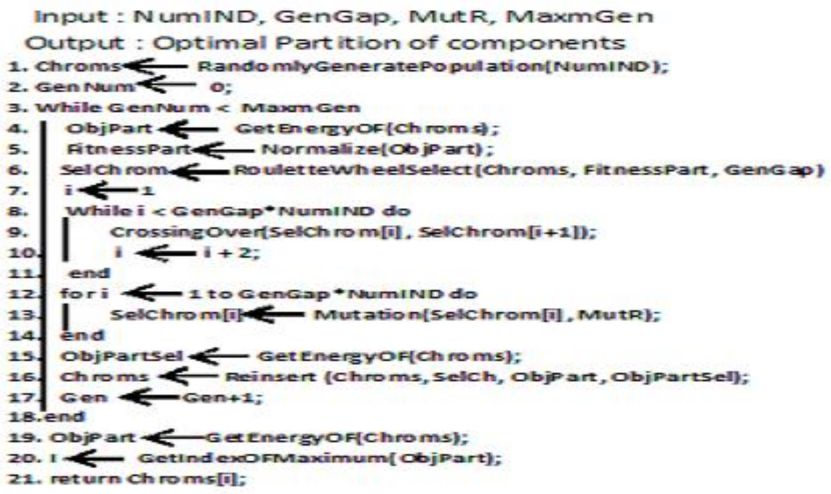
$$E_{local(i)} = P_{local}*(I_i / speed_{mobile}) \qquad (4)$$

We consider our optimal partition, an array of binary values in which $dec_i=0$ corresponds to execution at cloud side and $dec_i=1$ corresponds to execute component at mobile side. Total energy consumed for each component running can be written as

$$E_{total(i)} = dec_i * (E_{local(i)}) +(1 – dec_i)*E_{offload(i)} \qquad (5)$$

This equation will consider only one energy cost as if component is executed at cloud side then former part will become zero (i.e. $dec_i = 0$), and if executed at mobile side then latter part will become zero ($1- dec_{(i)} = 0$).

We have used genetic algorithm to obtain the optimal partitioning of the application. Genetic algorithm starts with randomly producing an original population whose number of individuals is a constant. In each generation, Fitness of each individual is evaluated. Then fittest individuals are selected from the current population for breeding. Then next generation is produced by crossing over and mutation among individuals. In the last generation a string with binary values is passed to our optimization formula which provides the partitioning decision according to values.



```
Input : NumIND, GenGap, MutR, MaxmGen
Output : Optimal Partition of components
1. Chroms ← RandomlyGeneratePopulation(NumIND);
2. GenNum ← 0;
3. While GenNum < MaxmGen
4.     ObjPart ← GetEnergyOF(Chroms);
5.     FitnessPart ← Normalize(ObjPart);
6.     SelChrom ← RouletteWheelSelect(Chroms, FitnessPart, GenGap)
7.     i ← 1
8.     While i < GenGap*NumIND do
9.         CrossingOver(SelChrom[i], SelChrom[i+1]);
10.        i ← i + 2;
11.    end
12.    for i ← 1 to GenGap*NumIND do
13.        SelChrom[i] ← Mutation(SelChrom[i], MutR);
14.    end
15.    ObjPartSel ← GetEnergyOF(Chroms);
16.    Chroms ← Reinsert (Chroms, SelCh, ObjPart, ObjPartSel);
17.    Gen ← Gen+1;
18. end
19. ObjPart ← GetEnergyOF(Chroms);
20. I ← GetIndexOFMaximum(ObjPart);
21. return Chroms[i];
```

Fig 3. Our Approach

5. NUMERICAL MODEL

If component is offloaded to cloud then energy will be sum of energies for sending data and idle CPU and display power energy. And if executed at mobile side then energy consumed by component is the energy taken to execute that component.

Power consuming part of mobile device	**Power consumption (in mW/sec)**
Display	900
CPU working	400
CPU idle	50
Data transfer using 3G	750

Table 2. Power consumption by different hardware components of mobile devices

Our cost minimization equation is to minimize the total energy consumed to execute a complete application.

$$\text{Minimize } (\sum E_{execution} + \sum E_{offloading} + \sum E_{idle}) \qquad (6)$$

Condition for offloading a component to cloud is that the energy consumption in offloading must be less than the energy of execution.

$$E_{execution} > E_{offload} \qquad (7)$$

In more specific form we can rewrite equation (7) as follows.

$$(P_{execution} + P_{display}) * t_{execution} > P_{transfer} * t_{transfer} + (P_{idle\text{-}cpu} + P_{display}) * t_{idle} \qquad (8)$$

$t_{execution}$, $t_{transfer}$, t_{idle} are time for executing component at mobile device, transferring component to cloud and time to offload at cloud side respectively. And values to time can be derived according to the bandwidth and clock speed of mobile device and cloud machine.

According to table , values for power consumption by hardware components of mobile device can be considered and model can be given as in (8).

$$(4+9)*(I_i / speed_{mobile}) + 9 *(I_i / speed_{mobile}) > 7.5*(Data_{mig(i)} / bandwidth) + (9+0.5)*(I_i / speed_{cloud}) \qquad (9)$$

If normalised value of $speed_{cloud}$ is used then equation is reduced to (10).

$$22*(I_i / speed_{mobile})> 7.5*(Data_{mig(i)} / Bandwidth) + 9.5*(I_i / speed_{mobile}) \qquad (10)$$

And hence can be finally reduced to a relation for deciding offloading, between total data to migrate and number of instructions to execute or offloading a component to save energy the transfer time to execution time ratio should be greater than 1.4. Hence in our scenario best energy performance will be there if a component is offloaded to cloud after the value crosses this ratio.

6. PERFORMANCE EVALUATION

We have considered Energy consumption as our main performance metric. In the table 3 we have shown the normalised values that we have taken for our evaluation. Value for bandwidth is 1. It shows relative value of bandwidth in respect to total data to offload for a particular component is taken in the range of [0,1]. Similarly values for mobile clock speed and cloud machine clock speed is 1 and 4 respectively and total number of instructions to execute a component is in range of [0,1]. Total number of components shows that a particular elastic application can be broken into such number of modules that can be offloaded at any time for execution.

Our results show the effect of all these parameters that are needed to decide the cost in terms of energy of an application. In each experiment, we choose one parameter as the variable, which is indicated by '*' , while assigning other parameters as constant values. Also in our graphs we have shown the energy cost in three approaches.(a) all components executed at mobile side, (b) all components executed at cloud side and (c) components executed according to our partitioning algorithm.

In fig 4(a), effect of different mobile clock speed is shown. As it shows that in the realistic conditions our approach consumes less energy as compared to other two approaches. But when clock speed of mobile device is very less, offloading maximum components of an application to cloud is energy saving solution and if the clock speed of our mobile device is relatively better then executing all components locally is optimal solution.

Fig 4(b) shows that increasing clock speed of cloud virtual machine decreases energy consumption as the execution of offloaded component can be fast and hence idle time is decreased so as energy consumed when idle. Fig 4(c) shows that increasing bandwidth gives better results when most components are offloaded to cloud device. Execution according to our approach gives better results each time however when bandwidth is increased 150%, offloading components saves energy as time taken to offload is lesser.

In Fig 4(d), graph between energy consumed and computational overhead is shown. Our algorithm divides components according to their energy consumption for execution either at cloud side or at mobile side. Our algorithm provides a partitioning decision considering waiting time as well as execution time at mobile device as well as at cloud VM. In fig 4(e) as the volume of data migration increases, local execution of components gives better results.

Fig 4(f) shows the effect of increasing components on energy consumed. Our algorithm gives better results according to our fitness function as breaking components in two groups depending on the energy consumption in offloading or execution. In fig 4(g) various energy consumption according to various SLA delay tolerance is shown. These are the agreements given by the cloud provider and better value shows lesser energy consumption.

In fig 4(h), number of components running at mobile side according to ratio of transfer time to execution time is shown. As the ratio increases the number of components executed at mobile side increases because this decreases the energy consumption in offloading component to cloud.

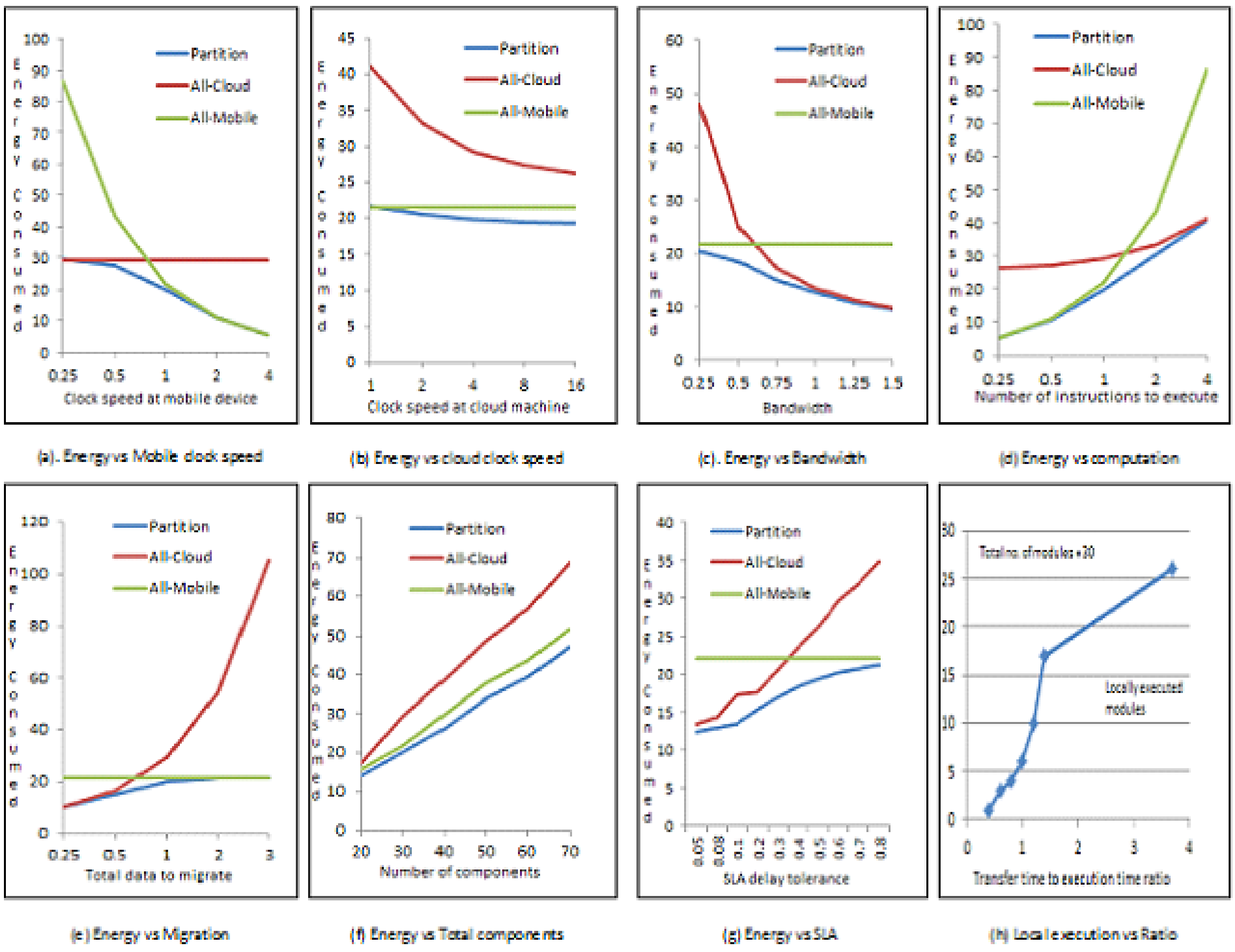


Fig 4. Simulation results

In fig 5(a). We have shown that the exact energy consumption after considering a waiting time at cloud side is considerebly greater than the energy consumed if waiting time is not there. We see that if waiting time is more at cloud side then executing maximum components at mobile side is best optimal solution. In fig 5(b) we have seen that the number of components executed at cloud side diminishes as the waiting time increases

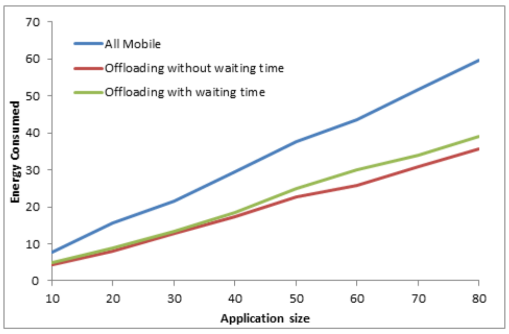


Fig 5(a). Energy consumption with waiting time

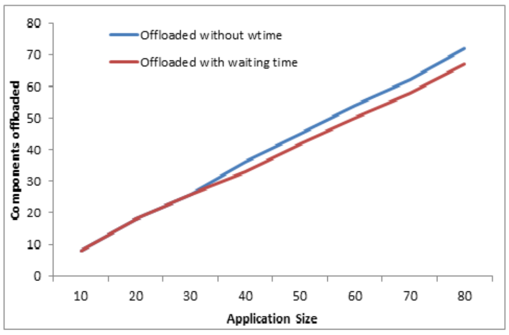


Fig 5(b). Components offloading comparison with waiting time

7. CONCLUSION & FUTURE WORK

As we have seen that battery life is one of the main constraints of mobile devices. In this paper we have designed a framework for elastic applications to get better energy performance. We have also designed genetic algorithm under this framework to solve partition problem. Algorithm considers various hardware components of mobile device that consumes power to produce the optimal solution. Simulation results have shown that our approach produces better results by 40 % than executing all components at cloud side and by 35% than executing all components at mobile side.

In our future work, we will try to conduct experiments on real-world applications based on our approach to see the results more practically. We have assumed that each VM at cloud side is enough resource rich to execute all tasks from queue in SLA negotiated time. But dynamic scaling of VMs has to be studied. Security overheads are also there to consider in our future work which we have assumed to be negligible in this architecture.

8. REFERENCES

[1] P. Mell and T. Grance, “The NIST Definition of Cloud Computing” [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] M. Satyanarayan, “Fundamental Challenges in Mobile Cloud Computing,” Proceedings ACM Symposium Principles of distributed Computing, ACM Press, 1996. [4] Shamim Hossain, “ThoughtsOnCloud Cloud Computing conversations led by IBMers”, [Online]. Available: http://thoughtsoncloud.com/2013/06/mobile-cloud-computing/

[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”, 2009, [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[4] E.Truyen, B.Robben, B.Vamhaute, T.Coninx ,W.Joosen, and P. Verbaeten, “Portable Support for Transparent Thread Migration in Java.” In Proceedings of 2nd International Symposium on Agent Systems and Applications and 4th International Symposium on Mobile Agents 2000, Zurich, Switzerland, Sept. 13-15, 2000.

[5] Wikipedia, “Mobile phone – Wikipedia , the free encyclopedia, http://en.wikipedia.org/wiki/Mobile_phones.

[6] Gonzalo Huerta-Canepa, Dongman Lee, “A Virtual Cloud Computing Provider for Mobile Devices”, *ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond. MCS’10*, 2010, San Francisco, California, USA: June 2010.

[7] J. Kangasharju, J. Ott, and O. Karkulahti, "Floating Content: Information Availability in Urban Environments," *Proceedings of the 8th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom) (WiP)*, Mannheim, Germany: 2010.

[8] A. Berl, H. Meer, H. Hlavacs, and T. Treutner, "Virtualization in energy-efficient future home environments," *IEEE Communications Magazine*, vol. 47, 2009, pp. 62-67. December 2009

[9] Ricky K.K. Ma, Cho-Li Wang, “Lightweight Application-level Task Migration for Mobile Cloud Computing .” In Proceedings of 26th IEEE International Conference on Advanced Information Networking and Applications, 2012.

[10] M. Satyanarayan, P. Bahl, R. Caceres, and N. Davies. “The Case for VM-based Cloudlets in Mobile Computing,” IEEE Pervasive Computing, 8(4), 2009.

[11] E. Marinelli, “Hyrax: Cloud Computing on Mobile Devices using MapReduce.” Thesis Report to School of Computer Science, Pittsburgh, PA. Sept 2009.

[12] L. Yang, J. Cao, S. Tang, Tao Li, Alvin T. S. Chan, “A framework for Partitioning and Execution of Data Stream Application in Mobile Cloud Computing.” IEEE Fifth International Conference on Cloud Computing, 2012.

[13] K. Kumar and Yung-Hsiang Lu, “Cloud Computing for Mobile Users: Can Offloading computation saves Energy?.” IEEE Computer Society. April 2010.

[14] B.-G. Chun and P. Maniatis, “Augmented Smartphone Applications Through Clone Cloud Execution,” in *Proceedings of the 12th Workshop on Hot Topics in Operating Systems (HotOS XII)*. Monte Verita, Switzerland: USENIX, 2009.

[15] R. F. Lopes, and F. J. D. S. E. Silva. “Migration Transparency in a Mobile Agent Based Computational Grid.” In Proc. Of the 5th WSEAS Intl. Conf. on Simulation, Modelling and Optimization, pp. 31-36, Greece, August 17-19, 2005.