International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 9 5638 - 5643

Computation off loading to Cloud let and Cloud in Mobile Cloud Computing

Rushi Phutane
Department of Information Technology,
PICT, Pune 411043, Maharshtra ,India,
phutane_rushi@yahoo.co.in

Prof. Tushar Rane
Department of Information Technology,
PICT, Pune 411043, Maharshtra ,India,
ranetushar@yahoo.com

Abstract *:-*Computation offloading in mobile cloud computing means transfer of execution of mobile application in the mobile device to the virtual mobile device in the Cloud or the Cloudlet. Intensive application like Nqueens puzzle, tower of hanoi when they execute on the Compute mobile device they consume more time because of the low computational power and limited battery capacity of mobile devices. By offloading computation to resource rich Cloud, energy consumption on the mobile device can be saved considerably and limitations of mobile devices can be overcomed. However offloading becomes difficult, when internet connectivity is interrupted due to hostile environment. Hence we offload compute intensive task from mobile device to resource rich surrogate machines in local network and overcome limitations of mobile devices when the internet connection is missing.

keywords:-*Computation Offloading,Surrogates,Cloud,Nquuens Puzzle,Tower of Hanoi Puzzle*.

_____****_____

I. Introduction

Smart phones and tablets are the devices increasingly used by everyone today to perform their day today activities.
The main point of attraction of them is for gaming and feature rich applications like voice search. Such demand has led to the development of new hardware and software technologies for the mobile devices. Inspite of the continuous technological improvements, still the mobile devices faced some of the limitations like computation capabilities of mobile devices are limited compared to the computers. These devices are powered by the battery, which is limited in sense therefore energy constraint has not been solved satisfactorily also mobile devices are accompanied by limited memory capacity. To overcome these potential limitations, it has been suggested to offload code execution from the mobile node to external machines. This strategy has many potential advantages:

(i) Reduced application execution time.
(ii) Reduced battery consumption and
(iii) the possibility to execute applications whose resource demand could exceed the capabilities of mobile nodes.

However, semiconductor and telecommunication technologies are improving at the faster rate these days, so high speed Wireless networks of increasing bandwidth allow these mobile units to connect to external machines through wireless networks and form complex distributed system. When these external machines are located in the cloud, It becomes easy to manage them together,provide them as utility over the network to tackle large computation problems.Where cloud computing is emerging technology which enables on demand network access to a shared pool of configurable resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released from anywhere anytime in the world. Giving users the freedom to collaborate from different locations.Mobile cloud computing intends to utilize the resource from the cloud to enhance the computing capability of the mobile device. It helps to overcome the limitations of mobile devices in particular of the processing power and data storage. Mobile cloud computing helps to extend battery life by transferring the execution of compute intensive application to the cloud. This task is called the computation offloading. In this technique application can be adaptively split and parts are offloaded to the cloud. Which portions of the splited parts of the application are executed remotely is decided at runtime based on resource and network availability.

The cloud infrastructure is usually located at far away distance from the mobile device that can introduce the network delays or network connection might get interrupted due to bad weather condition. This affects the user experience of the application. To overcome this problem, it has been proposed to use close by servers (referred to as cloudlets) typically located at the wireless access points (APs) where mobile nodes connect to so that they are at just "one hop" distance from the mobile node.When a mobile user sends service request to the mobile cloud system, Cloudlet or Cloud can be selected based on the location of the mobile users.Then one or more android virtual machines can be allocated based on amount of computations to be done. Then the process of offloading can be done.

The paper is organized as follows first we discuss the related works and common approaches in implementing the mobile cloud offload architecture.Then in the third part we discuss the system model of our system.Then in the fourth part we discuss the architecture of the system.In the fifth part we discuss the results obtained from experiements performed. Finally we conclude our paper .

II. Related Work

Earlier work in this area is Internet Suspend and Resume[1] where they move whole VM or OS image along with the running process. But it faced the problem of large amount of data to be transferred over the network. These limitation were overcome in the VM synthesis approach[7].Where part of VM is send from the mobile to nearby cloud that already holds base VM of same type. Then automatic partitioning of application is used in the Coign[2] to reduce communication cost of partition components.Where this application is partitioned statically. This did not provide the better user experience as application is run in changing environmental conditions so to overcome the limitations application is partitioned dynamically by taking various environmental conditions into account and used in the systems like Clone Cloud[5].These limitations overcome by partitioning the application dynamically at binary level by taking various environmental conditions into account. An analysis is done offline to decides which binary pieces are to be migrated to the cloud. MAUI[6] enables fine-grained energy-aware offloading, provides a graph of program's methods and divides those methods into local and remote groups. Meanwhile it uses online profiling and a history-based approach to decide whether a method should be executed remotely. However, it provides a server for each application, which is inefficient when handling many new applications. When the cloud is many latency away from the mobile device it introduced the communication delay. Which hampered the application performance.So we developed the computation system which can offload computations to cloud and cloudlet.

III. System Model

We have developed computational offloading system where in mobile device can access wireless access point to connect to the cloudlet and to cloud through Internet.

Compute intensive applications are run on the mobile device and remote virtual machines. The constraints of the mobile device like limited computational capabilities and battery power can be overcome by shifting computation to an infrastructure consisting of . (i) cloudlet i.e. server with computational capabilities which are located one hope distance from the mobile device. (ii) convectional remote cloud accessible over the internet.Since comnunication and computation aspect play a key role in mobile cloud computing . we introduce their models in the comming sections.

3.1. Communication Model

We first introduce the communication model for our computation offloading system.The wireless access point can be either wifi access point created through the hotspot or we can access the wireless network created through macrocell base station that manages the uplink/downlink communications of mobile device users. We denote $a_n \in \{0,1,00,01,10,11\}$ computation offloading decision of mobile user. .Specifically, when we have a_n=0 or 00 or 01 user decides to compute its task locally on the mobile device. When we have a_n=1 then the user chooses to offload the computation to the remote android virtual machine. Based on the next input it is decided to offload the computations to cloudlet or cloud. When we have a_n=10, the computations are shifted to cloudlet and when we have a_n =11, the computations are shifted to cloud.

3.2. Computation Model

We then introduce the computation offloading model. Where the processing speed of mobile devices can be enhanced by offloading.The condition to improve performance can be formulated as below.For this the program was divided into two parts.One of the parts executed on the mobile device and other run on the remote android virtual machine.Let s_m be the processing speed of the mobile device.Suppose W is the amount of computation for the part executed on the remote machine.The time T_m to execute offloaded part on the mobile device is

$$T_m = \frac{w}{s_m} \quad \text{.....(1)}$$

Time taken to offload the second part on the remote machine is $\frac{d_i}{B}$ seconds .Where d_i is the input data and B is the bandwidth of the network. The improvement in the performance is achieved when execution of second part and time taken to shift the part is performed faster than the time taken to execute second part on the mobile device.

Let s_s be the speed of the server. The time T_s taken to offload and execute the second part is

$$T_s = \frac{d_i}{B} + \frac{W}{s_s} \quad \text{.....(2)}$$

Offloading improves performance when Equation 1 > Equation 2:

$$\frac{W}{s_m} > \frac{d_i}{B} + \frac{W}{s_s} \rightarrow w \times \left(\frac{1}{s_m} - \frac{1}{s_s}\right) > \frac{d_i}{B} \quad ..(3)$$

The above inequality holds when W is large ,Server speed s_s is fast,small amount of data d_i exchanged , the bandwidth B is high.We also conclude that if $\frac{w}{s_m} < \frac{d_i}{B}$, offloading cannot improve performance

even if server is infinitely fast (i.e. $s_s \rightarrow \infty$)

Hence the task which are compute intensive (large w) with light data exchange(small d_i) should be considered.[4]

3.3. Save Energy

Limited energy is one of the constraints of mobile devices. Energy is heavily consumed when compute intensive algorithms are executed on mobile devices. To overcome this limitation we have to shift the execution of the compute intensive parts computation to remote android virtual machine.The following analysis explains the conditions when offloading saves energy.

Suppose p_m is the power required to execute of compute intensive part on the mobile device.The energy E_m required

to execute the compute intensive task on the mobile device can be obtained by modifying Equation 1.

$$E_m = p_m \times \frac{w}{s_m} \quad \text{.....(4)}$$

Let p_c be the power required to send data from the mobile device over the network
and p_i be the power consumed during sending the data over the network and receiving the result.When p_c and p_i are substituted in the Equation 2 we obtain Equation 5.

$$E_s = (p_c \times \frac{d_i}{B}) + (p_i \times \frac{W}{S_s}) \text{.....(5)}$$

Offloading saves energy when Equation 4 > Equation 5.

$$p_m \times \frac{w}{Sm} > (p_c \times \frac{d_i}{B}) + (p_i \times \frac{W}{S_s}) \text{ ..... (6)}$$

$$w \times (\frac{p_m}{s_m} - \frac{p_i}{S_s}) > p_c \times \frac{d_i}{B} \text{ ..... (7)}$$

From Equations 3 and 7 we conclude that for offloading to save energy, heavy computation (large w) and light communication (small d_i) should be considered.[4]

IV. Architecture Of The System

The below figure shows the architecture of the computation offloading system. On the client side if we want to execute the compute intensive method then it should be annoted with @Remote.This method invocation is done via the ExecutionController, which detects if a given method is a candidate for offloading and handles all the associated profiling, decision making and communication with the application server(remote android virtual machine) without the developer needing to be aware of the details.

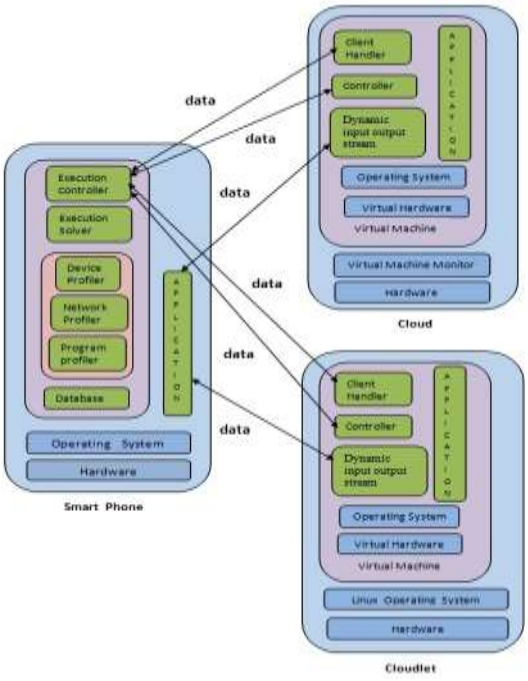

Figure 1: Architecture Of The System

4.1.Compiler

The task of compiler is to translate the annotated code to form suitable for the remote execution because code executed on cloud is on the x86 hosts while most smartphone processing is done on the ARM based processors.

4.2. Execution Controller

The Execution Controller is program that decides whether to shift the execution compute intensive algorithm to remote machine or to allow it to execute it locally on the mobile device. This decision is made based on the data collected about the current environment as well as that learned from past executions. When the compute intensive method is encounter for the first time to the Execution Controller for execution so the decision to offload the method is made based on the environmental parameters such as network bandwidth. After this when method is encounter the decision to execute it based on the methods past execution time and environmental parameters.

4.3. Application Server

When the compute intensive method is offloaded to cloud the application server handles the execution of the code and it the container for the Client handler.

4.4. Client Handler

The task of Client handler on the remote side is to register the application when the new instance of the application connect to remote android virtual machine. After this it simply returns the results for the method offloaded for remote execution.Client handler also responds to application level ping message sent by the Execution Controller to measure the connection latency.

4.5. Profiling

Profiler are important of the system.They help the system to make correct decision regarding offloading.They analyse the environmental conditions and provided necessary data to decision model.The profilers consists of three parts:

4.5.1. Hardware Profiler

The Hardware Profiler monitors the device characteristics like CPU,Screen, Wifi and 3G and feeds the information into decision model which helps making offloading decision.

4.5.2. Software Profiler

The Software Profiler monitors number of parameters concerning progam execution. This information is recorded using Android Debug API.The information recorded is as follows Number of instructions executed ,Thread CPU time of the method , Overall execution time of the method.

4.5.3. Network profiler

Network profiler monitors network state. The parameters tracked network profiler are like round trip time (RTT), network bandwidth. This information is used in decision making whether to execute the method locally or remotely.

V. Experiemental Evaluation

We evaluate our computation offloading system using three applications. The first application implements well known nqueens problem.The second application involves solving the Tower of Hanoi problem. These applications are choosen because it represents truely compute intensive problem.

5.1.Setup of Evaluation

The hardware we are using in the evaluation is as follows. A Samsung galaxy android phone GT-S5360 based on the android platform 2.3 is used in the evaluation. A Laptop with Intel i3 processors acts as a cloudlet and android virtual machine running in the cloud(AWS EC2)[12] that can handle the offloaded computations.The details of the hardware components for the mobile device and laptop are shown in the table.

Table 1 : Hardware componets of Mobile devices and Laptop

Hardware Component	Android Milestone	Laptop
Processor	ARM A8 600MHz	Intel(R) Core(TM)i3-32217U CPU @1.80 GHz 1.80 GHz.
Memory	256MB	8GB
WLAN	Wi-Fi 802.11 b/g/n	
OS	Android 2.3 Android x86 image.	Windows 8 (64 bit). Linux Server (Ubuntu 12.04 LTS-64 bit).

5.2.Experiement No.-1:

In this experiement we implement the algorithm that solve the n queens puzzle.
The algorithm find all possible solutions for the given number of queens N. The algorithm tried all possible arrangements of queens on the NXN grid so that they don't conflict horizontally, vertically and diagonally and gave the solutions.For example. At N = 8, possible arrangements of eight queens on a 8X8 board, but only 92 solutions. In this algorithm as more number of queens are used to find solution much more steps are taken to find all the solution.We run the N-Queens on the mobile device and the cloudlet and on the cloud seperately. for N=1 to N=13. For N=14 the n queens problem becomes heavily compute intensive and will take hours to finish on the mobile device, Therefore computations after n=14 should not to be offloaded.

Table 2 : Execution Time of Nqueens puzzle.

Number of Queens (N)	Local Time (ms)	Cloudlet Time (ms)	Send Time (ms)	Recieve Time (ms)	Cloud Time (ms)	Send Time (ms)	Recieve Time (ms)	Solutions
8	7	5	76	76	4	1770	1770	92
9	21	18	74	74	17	1518	1518	352
10	1155	264	18	18	110	589	589	724
11	4001	725	369	369	110	1109	1109	2680
12	4251	4108	34	34	3751	557	557	14200
13	24871	24221	18	18	24219	15563	15563	73712

The above table 2 shows the time to solve N queens puzzle on the mobile device, cloudlet, cloud, the number of solutions for the N-Queens Puzzle, Send time and recieve time of the puzzle.

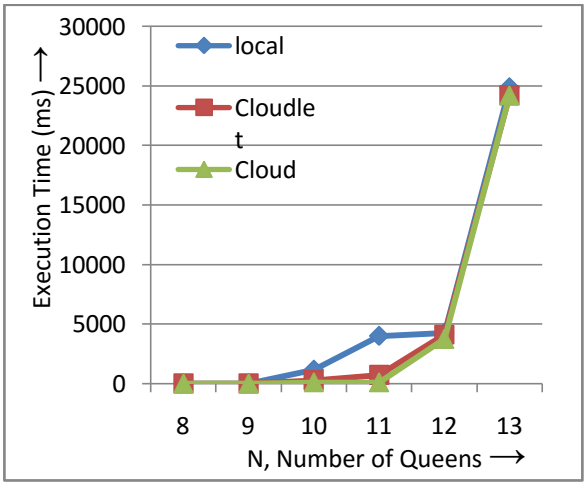

Figure 2:Execution time of the Nqueens puzzle for 8 ≤ N ≤ 13.

Figure 2 shows the time duration of execution of the puzzle.From *N* =8 to N =13 ,the execution speed of problem on the local device is acceptable compared with the remote speed for N=1 to N=9, but after *N* =10 , the remote speed dominates to be the better .With the increase of the queens number, the local execution time increases outstandingly, We see in the table 2 above that time taken to solve the puzzle for N=10, in the cloud and cloudlet is less than that on the phone. Therefore execution of the puzzle was shifted on the Cloud or Cloudlet. Cloudlet use was more efficient way to shift the computation because it can be reached with one hope latency than Cloud. Which was at far away from mobile user i.e. many hope latency away from user. Which required larger network bandwidth

and more time is consumed for transmission. So the time taken in transmitting the data to Cloudlet was less compared to the Cloud and therefore execution on the cloudlet should be preferred.

5.3. Experiement No.-2:

In this experiement we implement the algorithm that solve the Tower of Hanoi problem.The input to this algorithm is number of disks N that are to be moved from one rod to other rod.For example, In Tower of Hanoi puzzle the rod A with disks stacked on one above the other in decreasing order. Besides this rod A there were two other rods B and C. Here the objective was to move the disk from rod A to rod C using rod B for intermediate storage. As disk were heavy, they can moved only one at a time. In addition, at no time can a disk be on top of smaller disk. The puzzle can be played with any number of disks, The minimum number of moves required to solve a Tower of Hanoi puzzle was 2^n - 1, where *n* is the number of disks

Table 3 : Execution Time of Tower of Hanoi for 15≤N≤22.

Number Of Disks (N).	Local Time (ms).	Cloudlet Time (ms).	Send Time (ms).	Receive Time (ms).	Cloud Time (ms).	Send Time (ms).	Recieve Time (ms).	Number of Moves.
15	1497	1123	42	40	299	1708	1698	32767
16	2202	2112	56	53	559	1780	1764	65535
17	4059	3810	55	52	1147	1676	1613	131071
18	8352	8010	52	47	2269	1505	1489	262143
19	14314	13521	44	41	4549	1759	1734	524287
20	28977	27214	52	48	9146	1559	1544	1048575
21	57594	56594	50	44	18119	1540	1531	2097151
22	118081	115071	50	47	36517	1575	1564	4194303

The above table 3 shows the time taken to solve the Tower of Hanoi on the Mobile device, Cloudlet, Cloud ,the solutions return by the algorithm and send and recieve time of the puzzle. Here in this experiment we considered number of disk N from 15≤N≤22. The algorithm was used to calculate the number of movement of disk made by the algorithm to solve the problem.

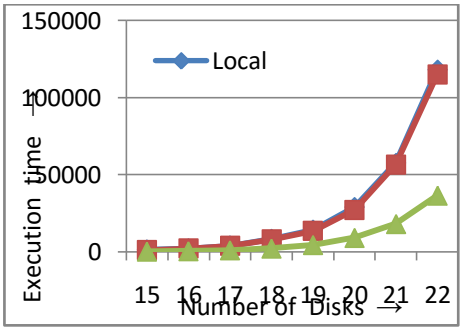

Figure: Execution Time for Tower of Hanoi 15 ≤ N ≤ 22.

The figure shows the time taken to solve the problem for executing problem on the Mobile phones, Cloudlet and Cloud.The execution speed of problem on the local device is acceptable compared with the remote speed for N=1 to N=16, but after *N* =16 , the remote speed dominates to be the better .With the increase in the number of the disk N, the local execution time increases outstandingly.When these problems were executed remotely on more powerful processors, execution time was less compared to that on the mobile device. Therefore problem execution was shifted on the Cloud or Cloudlet. Cloudlet use was more efficient way to shift the computation because it can be reached with one hope latency than Cloud. Which was far away from mobile user i.e. many hope latency away from the user. Which required larger network bandwidth and more time is consumed for transmission. So the time taken in transmitting the data to Cloudlet was less compared to Cloud .

VI. Conclusion and Future Work

When compute intensive applications are executed on the mobile phones user has to face the large response time which is not acceptable . This execution time can be reduced through offloading the computation to the remote android virtual machine.and application response time can be improved. Energy consumption on the mobile device can be saved which indicates users can have more battery time compared to the local execution. If offloading to cloud is not possible to network connection interruption.Then offloading can be done with local surrogates called cloudlet which are at one hop latency away from mobile device accessible over wifi hotspot.

We extend our work further by parallelizing application offloaded. By parallelizing the application offloaded it helps in improving the performance. We also take help of multisite offloading to fully use the capabilities of distributed computing to enhance the performance of the application.

VII. References

[1] Kozuch M., Satyanarayanan M., " Internet Suspend/Resume",Pub in: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02), 2002,Page(s):40 - 46.

[2] Hunt G. C. and Scott M. L., "The Coign automatic distributed partitioning system,", Pub in :Proc 3rd Symposium on Operating systems design and implementation (OSDI), 1999,Page(s): 187-200.

[3] Hoang T. Dinh, Lee C., Niyato D., and Wang P., "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches",Pub.in: Wireless Communications and Mobile Computing, 2013,Volume:13, Issue :18, Page(s):1587–1611.

[4] Kumar K., Liu J., Lu H. Y., Bhargava B., "A Survey of Computation Offloading for

Mobile System", Pub. in: Journal of Mobile Networks and Applications, 2013, Volume: 18,Issue:1,Page(s)- 129-140

[5] Chun B. G., Ihm S., Maniatis P., Naik M., Patti A., " CloneCloud: Elastic Execution between Mobile Device and Cloud ", Pub in: Proceeding EuroSys '11 Proceedings of the sixth conference on Computer systems, 2011,Page(s) 301-314.

[6] Cuervoy E., Balasubramanianz A., Cho D., Wolmanx A., Saroiux S., Chandrax R., Bahl P., "MAUI: Making Smartphones Last Longer with Code Offload ", Pub. in:Proc.MobiSys '10 Proc. of the 8th international conference on Mobile systems, applications, and services, 2010, Pages 49-62

[7] Satyanarayanan M., Bahl P., Caceres R., and Davies N., The Case for ´VM-Based Cloudlets in Mobile Computing", Pub.in: IEEE ,Pervasive Computing, 2009,Volume: 8, No. 4, Page(s). 14–23,

[8] Kemp R., Palmer N., Kielmann T., and Bal H., "Cuckoo: a computation offloading framework for smartphones.",Pub. in: Mobile Computing, Applications, and Services, Springer Berlin Heidelberg ,2012, Vol. 76,pp.: 59-79, ISBN 978-3-642-29336-8

[9] Gordon M. S., Jamshidi A. D., Mahlke S., Mao M. Z., Chen X.," COMET: code offload by migrating execution transparently ",Pub in:OSDI'12 Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation, 2012, Page(s) 93-106 .

[10] Liu F. , Shu P.; Jin H. ; Ding L. , Yu J. ; Niu D. ; Li B., " Gearing Resource-Poor Mobile Devices With Powerful Clouds: Architectures,Challenges and Applications ",Pub. in:Wireless Communications, IEEE, 2013, Vol.20 , Issue:3, Page(s):14 - 22.

[11] Zhang L., Tiwana B., Qian Z., Wang Z., Dick R. P., Mao Z., and Yang L. Accurate Online power estimation and automatic battery behavior based power model generation for smartphones. In Proc. Int. Conf. Hardware / Software Codesign and System Synthesis, 2010.

[12] "Amazon Elastic Compute Cloud (EC2)", Available at: https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:sort=instanceType

[13] "Android Developers", Available at: http://developer.android.com/training/basics/activity -lifecycle/starting.html