Adaptive Workload Offloading For Efficient Mobile Cloud Computing
Jayashree Lakade
Venus Sarode
# Table of Contents

1 Introduction .................................................................................................................. 3  
1.1 Objective .................................................................................................................. 3  
1.2 Problem Description ................................................................................................. 3  
1.3 Benefits to OS Class .................................................................................................. 3  

2 Theoretical bases and literature review ...................................................................... 3  
2.1 Problem Definition ..................................................................................................... 3  
2.2 Problem Solution ........................................................................................................ 4  
2.3 A different solution from others and why better ..................................................... 4  

3 Hypothesis .................................................................................................................... 4  

4 Methodology ................................................................................................................ 4  
4.1 Input data ..................................................................................................................... 4  
4.2 Algorithm design ........................................................................................................ 4  
4.3 Languages used .......................................................................................................... 6  
4.4 Tools used .................................................................................................................... 6  
4.5 How to generate output ............................................................................................... 6  
4.5.1 Building Cloud Project – MobileCloudServer) ..................................................... 6  
4.5.2 Building Scheduler Project – AdaptiveScheduler ............................................... 7  
4.5.3 Running the project ............................................................................................... 7  
4.6 How to test against hypothesis .................................................................................. 8  

5 Implementation ............................................................................................................ 9  
5.1 Code ............................................................................................................................. 9  
5.2 Design document and flowchart ................................................................................. 9  

6 Data Analysis ................................................................................................................ 9  
6.1 Output Analysis .......................................................................................................... 9  
6.2 Comparing output against analysis ......................................................................... 10  
6.3 Abnormal case explanation ....................................................................................... 10  

7 Conclusion .................................................................................................................... 11  

8 Bibliography ................................................................................................................ 11  

9 Appendices .................................................................................................................. 13  
9.1 Flowchart: .................................................................................................................. 13
1 Introduction

Mobile devices face a growing demand to support computationally intensive applications like 3D graphics and computer vision. Even though mobile processors are becoming more capable, local processing capabilities are still inherently limited by processor power density and device battery life. Meanwhile, high bandwidth wireless networks have become ubiquitous and are being used to connect mobile devices to the cloud.

1.1 Objective
Our objective is to study cloud computing and implement it for mobile devices so that they can perform intensive applications which otherwise require very long execution time on the device itself. This new algorithm is directed at cloud computing systems in order to maximize efficiency and also improve battery life.

1.2 Problem Description
The availability of increasingly richer applications is providing surprisingly wide range of functionalities and new use cases on mobile devices. Even tough mobile devices are becoming increasingly more powerful; the resource utilization of richer application can overwhelm resources on these devices. At the same time, ubiquitous connectivity of mobile devices also opens up the possibility of leveraging cloud resources. Seamless and flexible path to mobile cloud computing requires recognizing opportunities where the application executes on cloud instead of mobile device.

1.3 Benefits to OS Class
Operating Systems focus on improving system performance by adaptively scheduling processes across multiple cores, considering the available resources and making optimum usage of such resources.
In this project we are trying to develop an algorithm that takes into consideration the mobile device processor, battery usage and network availability at given time and offload the workload trying to make best use of these resources. This will surely given an idea as to how such decisions are taken by real operating system, rather such logic can be integrated into the current mobile operating system for cloud computing in future.

2 Theoretical bases and literature review

2.1 Problem Definition
The problem dealt with in this project is to execute CPU intensive processes on mobile devices (on cloud) in very short time, maintain the history of execution and make offloading decisions based on history to effectively use mobile device resources such as processor and battery.
2.2 Problem Solution
The solution is to offload the CPU intensive processes to cloud, thus saving the mobile device battery and making mobile processor available for native applications. Also cloud computers being more powerful than mobile devices, the execution times will be much shorter. The decision to offload the process can be made adaptive by maintaining the history of past executions.

2.3 A different solution from others and why better
Our solution aims at utilizing the dynamic features such as remote processor availability, network resources, which are used even some of the other approaches. However we have added extra adaptable behavior by taking into consideration the recent execution history. The weighted mean run times and battery usage based on past execution helps take the offloading decision even better. This can be further extended to use the concept of locality reference, etc.

3 Hypothesis
Traditional approach is to execute everything on mobile device. In order to speedup execution and save battery life, mobile computing can be offloaded to cloud. Our aim is to prove that by making adaptable decisions to offload computing to cloud, we can demonstrate significant gain in speed and battery life. Also, the system will perform better over time as our solution takes into consideration recent execution patterns.

4 Methodology

4.1 Input data
We have chosen the semi-prime factorization algorithm for large numbers as CPU intensive problem to demonstrate our proposal. Input to this problem is any semi-prime number. Sample input numbers could be 62615533, 8992285676317, 37460083163, 86438697847, 1220310534168151, etc.

4.2 Algorithm design
We propose an algorithm for offloading decision based on features such as speedup gain, battery usage at any given time.

Start threads to collect CPU, WI-FI & 3G battery usage at regular intervals

If network connectivity available then
If remoteServer.IsAvailable() then

If optimizeForSpeed then
    remoteExecTime =
        calcWeightedMeanRemoteExecTime();
    localExecTime =
        calcWeightedMeanLocalExecTime();

    if localExecTime > remoteExecTime * gainFactor then
        try
            runRemotely()
        catch
            runLocally()
        else
            runLocally()
        endif

    else // optimize for battery life
        remoteExecBatteryUsage =
            calcWeightedMeanRemoteBatteryUsage();
        localExecBatteryUsage =
            calcWeightedMeanLocalBatteryUsage();

        if localExecBatteryUsage >
            remoteExecBatteryUsage * gainFactor then
            try
                runRemotely()
            catch
                runLocally()
            else
                runLocally()
            endif

        else
            runLocally();
        endif

end if

updateHistory(); // update execution time for local/remote
updateResourceUsage(); // update battery usage for
// Local/remote

END
The weighted mean is calculated using the following formula:

\[ \bar{x} = \frac{\sum_{i=1}^{n} (x_i w_i)}{\sum_{i=1}^{n} w_i}, \]

4.3 Languages used
For Cloud part – Java, Python
For Mobile part – Android, Java

We have used RESTful web services for communication between mobile and cloud server.

4.4 Tools used
Tools used for this project are -
Eclipse 4.3 with Android SDK
Apache Tomcat 7.0.42
Apache Maven 3.0.4
PowerTutor(\url{http://powertutor.org/}) - used in scheduler for battery usage calculations

4.5 How to generate output
Steps to build and run the project are as follows -

4.5.1 Building Cloud Project – MobileCloudServer)

1) The folder hierarchy for cloud project is as follows
   - pom.xml
   - src
     o main
       ▪ java
       ▪ resources
       ▪ webapp
         o index.jsp
         o WEB-INF
           o web-application-context.xml

2) Run following commands on terminal where pom.xml file in MobileCloudServer project is present

$mvn clean
$mvn package

This creates a new folder named ‘target’ where pom.xml exists and generates war file in it.

3) Copy MobileCloudServer.war file from target folder into tomcat’s webapps directory.

4) Configurable parameter setup:
   a. Open file ‘web-application-context.xml’ in tomcat’s ‘webapps/MobileCloudServer/WEB-INF’ folder
   b. Set tomcat directory path for property ‘tomcatDir’
   c. Set python installation path for property ‘pythonDir’ as follows

<property name="tomcatDir" value="/Applications/apache-tomcat-7.0.42" />
<property name="pythonDir" value="/usr/bin"></property>

4.5.2 Building Scheduler Project – AdaptiveScheduler

1) Setup the android SDK build environment as described at the following page: http://developer.android.com/sdk/index.html
2) Unzip AdaptiveScheduler.zip and save it in a directory
3) Open eclipse from the android SDK and import the AdaptiveScheduler project into eclipse
4) Build the project

4.5.3 Running the project

1) Start tomcat webserver and access url http://<server-ip>:8080/MobileCloudServer/test. A message ‘Hello from AWOFEMCC mobile cloud server’ should be displayed
2) The android application can be run from the android SDK eclipse IDE itself. You can choose to run the application on an emulator or on an android phone attached to the computer with USB cable. If using a phone, ensure that it has USB debugging turned on. Using an android phone is preferred for fully exploring the application’s capabilities.
3) When the application starts running on emulator/phone, set the correct cloud server address in the settings in android application. Screenshot shown below
On the application home screen, the input semi-prime number provided can be scheduled to run explicitly on phone or on the cloud, or the scheduler can also be asked to schedule the factorization job.

4.6 How to test against hypothesis
The basic test is to compare the execution time and battery usage for remote execution over local execution. This confirms the possibility of efficiently executing
computationally intensive problem on mobile devices, which are otherwise impossible or time consuming.
The Next test is to check the scheduling decision made by application when the history is built over time and see if the scheduling decision is beneficial.
There could be a test to check the scheduling decision under different modes viz. optimizing speed and optimizing battery life.
Application can be tested against presence/absence of network connectivity

5 Implementation

5.1 Code

The complete source code including following items has been submitted as P3.
1) Cloud project
2) Scheduler project
3) README
4) Project report

5.2 Design document and flowchart

Flowchart can be found in see Appendix (9.1)

6 Data Analysis

6.1 Output Analysis

The following graphs show the comparison between the execution time when a semi prime factorization is executed on phone, cloud and with scheduler-under two different modes viz. speed optimization, battery optimization. From following graph we can clearly see the difference between various execution times. We can observe that for smaller input the execution on phone might be better but as we go on increasing the inout size, using scheduler proves to be the better option.
Graph showing execution time of semi-prime numbers on phone as well as cloud

6.2. Comparing output against analysis
From the above graphs we can conclude that we can now perform high CPU bound tasks on mobile devices with the help of cloud computing. Comparison between the execution time shown in the above graph clearly states that with the help of developing network technology we can now perform all the CPU intensive jobs on mobile devices and meanwhile mobile CPU is free to perform other important tasks which are required to be performed. Moreover it can also save battery as most of work is done by cloud and not by the device.

6.3. Abnormal case explanation
As our application makes use of past history to make precise decisions, it can sometimes happen that last few numbers that we have entered are very small digits numbers, but the next number that comes for execution is too large. Now in this case when the past history is evaluated for purpose of making decisions, it can happen that when history is evaluated it has no record of large numbers. So the output in this case can be abnormal. The app might not understand how to handle such situations and can large amount of time to execute.
7 Conclusion
We proposed a scheduling algorithm that tries to make use of cloud resource to augment the resources for mobile applications. We used learning based algorithm with features which includes not only the system resource utilization and performance related features but also higher level features. We proposed an algorithm for offload decision based on the expected gain while migrating to the cloud. We can conclude that, with the help of cloud servers we are able to perform CPU intensive jobs on mobile devices and we with help of adaptive scheduler the app can make decisions, with which we are able to reduce execution time and also improve the battery life.

8 Bibliography


http://dl.acm.org/citation.cfm?id=1066125
9 Appendices

9.1 Flowchart:

Flowchart showing how the execution takes place