Cost Evaluation of Computation Offloading on Mobile Devices

by

SathishKumar Pachamuthu

Advisor
Dr. Mohan Kumar

Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York
Acknowledgements

I would like to express my sincere gratitude to my project advisor, Prof. Mohan Kumar for his continuous support of my capstone project, for his patience, guidance and invaluable knowledge. I would also like to thank Prof. Lenoid Reznik for all the valuable comments and suggestions he gave during the milestone and poster presentations of my project. Also, I would thank all my fellow classmates who gave important feedbacks on my project during the presentations.
Introduction

In today’s world there are many resource rich mobile applications but even the most modern smartphones are not equipped to deal with these applications in terms of both computing power and energy required to complete these resource consuming tasks. Mobile cloud computing can be used to overcome this problem where the computationally intensive tasks are offloaded to a powerful and resourceful cloud server. But offloading a task is also not always the solution as there is a communication cost associated with uploading the input data to cloud server and downloading the output data from the cloud server. So there should be a mechanism to decide when to offload a task to a cloud server and when to execute the task locally on the mobile CPU. This project tries to solve this problem. In this project, a cost function which factors in both time and energy consumption is used to make the offloading decision.
Literature Review

As smartphones become increasingly popular, the research work on the mobile cloud computation offloading has also increased.

Most of the research either concentrate on reducing total execution time or reducing energy consumption. The research works explained in [5], [6] and [7] provide various methods to save energy while offloading. One of the popular techniques to reduce energy consumption is to partition the code into server side and client side using the profiling information. One such method is explained in [5]. In this paper the energy consumption of each function in an application is estimated and is represented as a graph where the nodes are the functions and the edges are the respective energy consumption. Then the max-cut min-flow algorithm is used to find the server side and client side partition to obtain maximum energy saving.

Most of the techniques used in reducing the total execution time also use the same partition profiling technique used for reducing battery power consumption as explained in [8] and [9]. The profiling of the data can be static or dynamic. i.e. the previously measured execution time of a function could be used as the estimated execution time for that function or it could be estimated dynamically using the run time information like network bandwidth, the data size of the input and output parameters, processing speed of the CPUs, etc.

There have not been much research work on reducing both execution time and energy consumption simultaneously. [1] proposes a method to reduce both time and energy at the same time. The solution proposed in this project is similar to [1].
Methodology

The method that is used in this project to decide whether to offload a task to remote cloud server involves a cost function defined in terms of execution time and energy consumption. The cost function basically computes how much reduction in execution time and energy consumption is obtained by offloading a task. It also uses a weighting factor to determine the importance level of time and energy.

Cost Function

The cost function is defined as follows

\[ \alpha \left( \frac{T_{\text{cloud}} - T_{\text{local}}}{T_{\text{local}}} \right) + (1 - \alpha) \left( \frac{E_{\text{cloud}} - E_{\text{local}}}{E_{\text{local}}} \right) \]

Where

- \( T_{\text{local}} \) = Time to execute the module locally on the mobile CPU
- \( T_{\text{cloud}} \) = Time to execute the module when it is offloaded to a cloud server
- \( E_{\text{local}} \) = Energy consumption of local execution of the module on mobile CPU
- \( E_{\text{cloud}} \) = Energy consumption when the module is offloaded to a cloud server
- \( \alpha \in [0,1] \) is the weighting factor given to Execution time and Energy consumption.

if \( \alpha =0 \), the cost function totally depends on Energy consumption,
if \( \alpha =1 \), the cost function depends only on Execution Time.

\( T_{\text{local}} \) = Time to fetch and load data into memory + time to execute the code in the module on the mobile CPU

\( T_{\text{cloud}} \) = Time to fetch and load data into memory + network transmission time to upload and download data + Time to execute the code on cloud server

\( E_{\text{local}} \) = Energy consumed to fetch and load data into memory + Energy consumed to execute the module on mobile CPU

\( E_{\text{cloud}} \) = Energy consumed to fetch and load data into memory + Energy consumed to transmit the I/O data
Decision Making scheme

The flow of application execution is shown in Fig.1. When the application starts, the input task, time and energy preferences are received from the user. The application also has previously measured execution time and energy consumption for list of tasks. When the input task is given, the estimated execution time and energy consumption for the given task is selected form the stored values and the value of the cost function is computed.

If the computed value is less than zero, then the task is offloaded to cloud server for execution otherwise if it is greater than zero the task is executed locally on the mobile device.

If the local execution time is lower than the cloud execution time, \( \alpha \left( T_{\text{cloud}} - T_{\text{local}} \right)/T_{\text{local}} \) would be positive but if the local execution consumes more energy than the cloud execution, then \( (1-\alpha) \left( E_{\text{cloud}} - E_{\text{local}} \right)/E_{\text{local}} \) would be negative then the sum of these two values that determines the overall advantage is used for the offloading decision.
Experimental Setup & Evaluation Approach

An Android face recognition application has been developed to test the decision making scheme. The face recognition application was developed using java openCV library, bytedeco [13]. When an image is uploaded to the application, it detects the number of faces in the image and also the name of the person in the image provided the detected face matches with a face the application already has in its database. This database is created by providing training images to the application. The detected faces are indicated by a green box. If the detected face is matched with one of the trained faces, then the name of the person is written on the top of the green box. The process flow of the face recognition application is explained in the Fig. 2.

![Fig. 2 Process flow of face recognition application](image)

The Android face application is deployed on a LG Nexus 5 mobile device having Qualcomm MSM8974 snapdragon 800x 4 2.2656Ghz processor, 2 GB of RAM and 2300 mAh Li-Po battery. A Dell Inspiron 15R laptop having Intel i5-3120M @2.50Ghz processor and 8Gb of RAM is simulated to act as a cloud server. The mobile device and laptop are connected using a Wi-Fi network and LipeRMI[12] is used to transfer the input and output data between the two. The execution time of a task is calculated programmatically and Trepn profiler [10] application is used to measure the energy consumption of the face recognition application. In order to measure the energy consumption of executing a particular block of code in the application, Trepn specific intents are added to the Android code and broadcasted, which would measure the energy consumption consumed in executing that block.
Each of the measurement of the execution time and energy consumption are taken 19 times and averaged to get a normalized value. The execution time and the energy consumption of the face recognition application are measured for images of various sizes and for various set of training images.

Images of pixel size 300x168, 620x413, 1000x660, 1222x814, 1198x1000 and 2000x1000 are used to compute the execution time and the energy consumption. The file size of these images are 7 kb, 86 kb, 110 kb, 158 kb, 283 kb and 501 kb respectively. These images are processed under different training images of size 20, 50, 100 and 200 to comprehend better how the computational intensiveness affect the execution time and energy consumption in both local and cloud execution. The measured values of execution time and the energy consumption of both local mobile CPU execution and the offloaded cloud server execution of a task are used by the cost function to decide whether to offload a task or not.
Results & Analysis

After measuring the values for all the images, the percentage reduction in execution time and energy consumption while offloading compared to the local execution is calculated. The values of the percentage reduction in execution time is shown in Table 1.a and the percentage reduction in energy consumption is shown in Table 1.b. The graphical representation of these values are shown in Fig 3.a and Fig 3.b.

<table>
<thead>
<tr>
<th>Image size</th>
<th>300x168 (7kb)</th>
<th>620x413 (86kb)</th>
<th>1000x660 (110kb)</th>
<th>1222x814 (158kb)</th>
<th>1198x1000 (283kb)</th>
<th>2000x1000 (501kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Trained Images</td>
<td>20</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>21.50%</td>
<td>30.61%</td>
<td>35.15%</td>
<td>41.82%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-4.17%</td>
<td>11.48%</td>
<td>24.09%</td>
<td>36.11%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-19.80%</td>
<td>5.02%</td>
<td>21.08%</td>
<td>32.70%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-39.25%</td>
<td>-1.27%</td>
<td>10.18%</td>
<td>22.46%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-67.84%</td>
<td>-17.27%</td>
<td>0.51%</td>
<td>17.58%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-77.91%</td>
<td>-40.41%</td>
<td>-17.25%</td>
<td>12.17%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.a Percentage reduction in execution time
In Fig 3.a, it can be seen that as the size of the image increases, the reduction in execution time decreases for all sizes of the training set. This is due to the addition of longer network transmission time needed to upload and download a larger image to the total execution time. Also the reduction percentage increases for images of all size when the number of trained images increases. As the size of the training set increases, the computation becomes intensive, so a less powerful mobile processor would take longer time to process an image where a powerful cloud server could process the image faster. As a result, the maximum gain by offloading is achieved when the pixel size of an image is smaller and the number of images in the training set is larger. There is no gain in offloading when there are few trained images and the pixel size of an image is larger.
It is evident from 3.b that the difference in energy consumption between the local execution and cloud execution decreases as the image size increases. This is due to the fact that uploading a larger image through the network takes more time and as a result consumes more energy. Also, as the size of the training images increases, the difference in the energy consumption increases. When the size of the training set increases, more CPU cycles are required to process an image resulting in consumption
of more battery power whereas the energy required to offload an image more or less remains the same regardless of the size of the training images.

The maximum gain in energy by offloading is obtained when the size of the training set is larger and the size of the image is smaller. The minimum gain is obtained when the size of the training set is smaller and the size of the image is larger. This is in line with the results obtained for the execution time.

Also the execution time and energy consumption obtained using the cost function value for the different values of alpha is plotted in a graph and analyzed.

20 Trained Images:

The measured execution time and energy consumption values for the training set of size 20 is shown in Table 2.

When the size of the training images is 20, the computational intensiveness is less and so the processing time of an image. In this case, the advantage of processing an image locally on mobile CPU increases as the image size increases, as there is longer network transmission time compared to the processing time but the local execution consumes more energy than the energy required when the processing is offloaded.

<table>
<thead>
<tr>
<th>Training Set Size : 20</th>
<th>300x168 (7kb)</th>
<th>620x413 (86kb)</th>
<th>1000x660 (110kb)</th>
<th>1222x814 (158kb)</th>
<th>1198x1000 (283kb)</th>
<th>2000x1000 (501kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Local</td>
<td>Cloud</td>
<td>Local</td>
<td>Cloud</td>
<td>Local</td>
<td>Cloud</td>
</tr>
<tr>
<td>Time (ms)</td>
<td>241.4</td>
<td>189.5</td>
<td>290</td>
<td>302.1</td>
<td>338.3</td>
<td>405.3</td>
</tr>
<tr>
<td>Energy (J)</td>
<td>5.21</td>
<td>2.76</td>
<td>5.35</td>
<td>3.01</td>
<td>5.48</td>
<td>3.26</td>
</tr>
</tbody>
</table>

Table 2

The value of the cost function is computed using these values for different values of α (alpha) and is shown in Fig. 4.c. When the value of α is lower, the energy consumption is given more importance and the value of computed cost function is lower than zero as the energy consumption of cloud execution is lower than the local execution. As the value of α increases, the execution time is given more importance and value of the cost function goes up as the local execution time is less than the cloud execution time.

Based on the value of the cost function, the corresponding execution time and the energy consumption for different values of α (alpha) are shown in Fig. 4.a and Fig. 4.b depending on
whether it is a local execution or cloud execution. For example, for the image of size 2000x1000 (501kb), the value of the cost function goes above zero when the value of the alpha is above 0.3. As we know that there is a gain in offloading only when the computed value of the cost function is below zero, cloud execution is selected for values of alpha below 0.3 and local execution is selected for values of alpha above 0.3. This is also reflected in the execution time and energy consumption graphs. The cloud execution values are selected until the value of alpha is 0.3 and the local execution for alpha above 0.3.

Fig. 4.a

Fig. 4.b

Fig 4.c
50 Trained Images:
When the size of the training images is 50, the computation becomes more intensive. As a result, offloading is chosen in far more cases than when the size of the training set is 20. For example, if you look at the image of size 1222x814 (158kb) in Fig. 5.c, the value of the cost function is above 0 for values of alpha above 0.5 when the training set size is 20 whereas for training set of size 50, the value of the cost function is below zero for all values of alpha except for 1.

![Fig. 5.a](image)

![Fig. 5.b](image)
100 trained Images:
When the size of the training image set reaches 100, the cloud execution is selected for all the values of alpha except the largest image as the local execution time and the energy consumption is higher compared to the cloud execution for all the selected images.

200 trained Images:
When the size of the training image set reaches 200, the cloud execution is selected for all the values of alpha for all the selected images.
Sample Run

A sample execution of the application is shown in Fig. 6.a and Fig. 6.b. The selected image file is of size 501kb having 2000x1000 pixels and the size of the training set is 20. In Fig 6.a, energy consumption is given preference as the selected value of alpha is zero and so cloud execution is selected. The time taken to complete the task is shown at the bottom.

In Fig 6.b, execution time is given preference as the selected value of alpha is one and so local execution is selected. The time taken to complete the task is less than the previous example but the consumption of energy is more.

The user is given more flexibility of choosing energy or time while executing a task.
Conclusion

For the image size 300x168, both energy and time could be reduced by offloading for training set of size 20 or above. For the images of size 620x413 and 1000x660, both energy and time could be reduced by offloading for training set of size 50 or above. For the images of size 1222x814 and 1198x1000, both energy and time could be reduced by offloading for training set of size 100 or above. For the image of size 2000x1000, both energy and time could be reduced by offloading for training set of size 200 or above. The general trend is that reduction in execution time and energy consumption by offloading gets bigger as the size of the training images increases.

Future Work

This experiment has used the already measured execution and energy consumption to evaluate the cost function. Instead of using the measured execution time and the energy consumption, the values could be determined dynamically by using the processor speed, size of the image that is to be processed and the network bandwidth. Also this experiment was performed using a WiFi network with a constant bandwidth. The same experiment could be repeated under different networks like 3G, 4G etc and also under different network bandwidths.
References


