A Mobile Cloud Offloading Technique for Image Classification Service
using HIPI

1NANDHINI L.K and 2UDHAYAKUMAR. S
1Assistant Professor, Department of CSE, Rajalakshmi Engineering College, India
2Associate Professor, Department of CSE, Rajalakshmi Engineering College, India

ABSTRACT
The evolution of technology with cloud computing has exalted in a way where the scarcity that exists with mobile devices like poor computation capacity and lower battery life can be outstripped. The best solution to overcome these disadvantages is Computational offloading. The renovation of smartphones has moved in where its demand of resources for every single application has increased. The solution to overcome this issue is Resource as a service where individual resources are deployed. Merger of cloud computing and mobile computing eventually lead to mobile cloud computing. To achieve context based offloading, the need of mobile situation is essential whether to offload the data or not. Our approach is to perform an image processing application in a mobile device. To process a computationally intensive task like this a mobile phone utilizes a maximum computational resource. We implement Hadoop image processing interface library along with the interpretation of opencv to perform higher throughput image processing along with Map Reduce programs. The Hadoop image bundle in HIPI is converted into opencv-mat format that are used to detect face count. We further focus on user mobility and low energy consumption of device where the user can process at any public place in a smart city with minimum requirement of WIFI

Keywords: Cloud computing, Hadoop image processing interface, service-offloading.

INTRODUCTION
Emerging New Technology
The technology and its perspective of application keeps changing all the time. It is a step by step process that follows innovation, invention and diffusion for processes [20]. Because of the demand of innovative stuffs the technology field will expand exponentially. The impact of this change is suitable for computing where a lot of innovation and growth happening every single minute for a particular application and product. One visible change is proprietary service where the publisher owns the Intellectual property where the users need to pay for license to an open source platform; the users can modify the software. Similarly one commendable change is On-premise applications to cloud.
computing. The interpretation of mobile computing, grid computing, autonomic computing and finally cloud computing has its own pros and cons. The innovation starts where there is a flaw in an existing application or technology. Over various decades, the research towards cloud computing has developed in different ways where it inherited from its predecessors like grid computing, autonomic computing. Attracted with its various features it has been estimated that there will be around 3.6 billion cloud users in 2018[8].Similarly mobile cloud computing has been evolved to provide rich mobile applications.

**Impact of mobile-cloud in smart cities**

When mobile environment collaborates with the services that are provided by cloud computing that are led to mobile cloud computing technology. Smartphones that are more prevailing than normal mobile phones with its features like high memory storages, improved mega pixel camera, video calling etc. Enhancing higher technology and improving the quality of life, smart cities has several characteristics that are build up to give a sophisticated infrastructure to its citizens. Scenario in smart cities has got up with several parts like user devices, sensing nodes, access points, wireless infrastructure and sensing nodes [2]. The usage of cloud being extended to mobile computing which called mobile cloud computing. As a part of its services in smart cities contains various set of applications with data sharing points [3]. Making higher usage in smart cities mobile cloud can feature itself as an important attribute provisioning services to its customers. And more importantly gathering data in smart services is done by mobile devices which is called crowd sensing.

**Service offloading in mobile cloud**

The development of multi core is the perfect step to make advancement in technology. Despite this advancement it is not possible to load as many core for mobile phones as in computer. But today phones have become an important possession in our day to day life. From Browsing to games, Internet in any smart phone has become mandatory over past few years. Stunned by their endless applications and device merger it has been estimated that there will be around 4.7 billion mobile users all around the world[4]. Though mobile devices have tremendous storage capacity and enormous services , a mobile phone has its own limitations. They include higher battery consumptions, limited memory compared to desktop or laptop services, lesser performance on intensive operations. Hence it is better to offload the task that is needed ,to a reliable source where the intensive operations can be solved and send back to the mobile phone. This makes better solutions for any task evaluation that we feel impossible with mobile phones. This has also lesson huge battery consumption in smartphones thereby improving the performance.
The vision of this paper is to provide a detailed procedure for service offloading on mobile cloud computing and to improve the processing speed in offloading. The main aspect of making mobile applications richer to feature them to explicit properties like context awareness and ubiquity mainly for mobile cloud computing. They are derived from mobile internet applications for web applications rendering its different characteristics like desktops.

**RELATED WORK**

In this section we are going to discuss about the work that has been done previously using various Methodologies. The one we are going to discuss here is computational offloading. This work uses genetic algorithm by providing service workflow. It provides a set of possible solutions for offloading for which the globally optimized solution is taken. Each possible solution corresponds to a chromosome in the population that consists of genes. It chooses mobile cloud computing model that contains five different strategies such as mobile device workflow, mobile service, service workflow, cloud server, base station, offloading strategy. It gives us near optimal solution by using the offloading services OS = {01, 02,03….0n}. Based on the value of offloading service it decides the data to be offloaded or not. The other work contains adaptive computation offloading service that contains the cost function to transfer the data for offloading services. It follows the methodologies of MACS architecture for offloading system. It shows that the overhead of CPU is more hence the calculation of using this strategy should improvise the needs of high memory. The next paper that has taken offloading games on a computational platform that has led to various creativity with respect to memory usage and privacy that has made advantage of accuracy in the offloading process.

**MOTIVATION**

India is presently the third highest number of internet users in the world. The growth of smart phones are extensively increased over the past few years due to the usage of Internet, browsing and social media. Recent statistics has said in India, 71 % of the 371 million mobile internet users are from urban side[6]. This growth has proved that expectation of each user increases day by day towards smart phone. Even though the smart phone users are promised with high storage and huge number of applications, the compatibility of a mobile phone is always less significant compared to any desktop or laptops devices in terms of battery usage.

**Scenario 1**

Consider any scenario where a user at a railway station in a smart city fully equipped with free Wi-Fi. He has a necessity to process a bunch of photos for face recognition. There are various apps that are available in Google app store where various face recognition is done. Comparatively these apps are slow in processing images and have an image limit maximum of 100. More over 40% of the
recognition are false positive. This becomes a hindrance if the user wants to process a large amount of application. Thus this makes a situation of service offloading as it is impossible with the smart phone. The storage of data and processing large scale applications has sent to a private cloud. The performance of face recognition will be comparatively higher when compared to be higher in mobile cloud. We make this example even easier by explaining with the following step by step procedure:

**Step 1: Input images**
First select the bunch of images that we want to process in cloud from mobile cloud. We work exclusively on android platform, so we use devices on android platform.

**Step 2: Decision support system**
In the android phone we choose to have a decision support system that should decide whether offloading should be done to private cloud or public cloud.

**Step 3: cloud system**
The cloud system we prefer for private cloud is Open Stack, where the offloading process is done. If it is not possible in the private cloud we make use of public cloud like Amazon Web Services(AWS). However the mobile phone should decide whether it is private or public.

**Step 4: Face recognition and registration**
The facial features of a person have been collected like nose tip, eye center and mouth center. The image registration process involves normalizing set of images. The normalization process reduces the intensity of values.

**Step 5: Hadoop Image Processing Interface**
All the images that are stored for processing in facial recognition are stored as image bundle using Hadoop Image processing interface. The image bundle is first culled (filtered) and they are processed into mapper and reducer phase

**Step 6: Output phase**
The final output will be the set of images with the given features, processed and sent back as output to the mobile

**Scenario 2**
Considering another scenario where user’s smart phone has huge volume of images. The user who is in a public spot wants to select an image with one particular background (say sea-level) which he has
in his mobile travelling various countries and he needs to share in one of the social media website. One obvious solution for the user would be mobile cloud where he sends the image bundle to a private cloud built with Hadoop image processing interface (HIPI). Once the images are sent, the images are processed and sent back to the user sending the processed image. Within a short period of time he is done with his task with the application of mobile cloud. This is totally energy efficient as the battery of the mobile phone is saved. The technology which has been given here is not restricted to smart phones but also with any device with android application including tablets.

**PROPOSED WORK**

The following figure explains about the overall work that happens in the process in terms of private cloud. The mobile device which is incapable of performing an image processing application is sent to the cloud server which is near to the mobile is sent through WIFI. Then the cloud server that is build with the components like host operating system (here we use ubuntu 14.04), openstack juno that contains ubuntu image on top of it, where we implement hadoop distributed file system and hadoop image processing interface to perform parallel mapreduce for images. We prefer WIFI over mobile data since the speed of WIFI is higher when compared to mobile data. The entire set of process happens and the images

![HIPI process](image)

*Figure 1. HIPI with the interpretation of opencv*

**Hadoop image processing interface**

There has been a wide range of using images that is shared, used and uploaded every single day. The biggest challenge is to manage them out of which every image uploaded or shared should be saved without any data loss. That creates a conviction for any user towards the social medium which he uses it. According to annual Internet Trends report by Mary Meeker, by average of about 1.8 billion digital images are uploaded around the world, every single day. So today’s smart phone world every user has numerous images to be saved, retrieved and edited. We now explain about hadoop image processing
interface library in detail in this module [2]. It is generally an image processing library to be used with the Apache Hadoop Map Reduce parallel programming framework. It abstracts the highly technical details of hadoop system and in the implementation of computer vision literature it is very flexible [2]. It overcomes the inconvenience of standard hadoop map reduce programs, to first pass the images as string ,then in each map task the images are decoded. But using our HIPI library the hadoop image bundle the specification of input is used to distribute images in the HIPI image bundle across all map nodes. Moreover the library itself does InputFormat and RecordReader class for the distribution of input as user get the float images directly. The working takes place as this. The initial processing stage is the culling step which filters th images in a Hadoop image processing bundle. The culler class does not fully decode the images which save the processing time. Image bundle processed through Culler class maximizes the data locality. Then individual images are presented to the Mapper as objects. The process of cropping, scaling and color scale conversion is provided by the classes Byte image and Float image. The images received from the mapper, are taken and transferred to the reducer by the built in map reduce shuffle algorithm. This minimizes the network traffic. Then the reducer tasks are done and executed and their output is generated

**OPENCV in HIPI processing**

The process that takes hadoop image processing interface needs the HIPI data to get processed of how .hib images get accessed to. The process that starts with processing the jpeg images that are converted to .hib images from internet and the URL is given to the Hadoop image processing interface. This is

![Stack representation of offloading process](image)

*Figure 2. Stack representation of offloading process*

where the process starts. The entire process happens as a part of map reduce job where the mapper converts the HIPI float image in to opencv mat image and detect the faces in the images. The reducer counts the number of faces detected and outputs the number of faces and files detected. The processing that takes place involves only few milliseconds for higher memory of image data. That
makes the process easier for offloading and the further implementation. We would like to give an
analysis of time taken for performing the entire process with .hib images and for opencv format. The
graph 1 gives the time taken for performing jpeg images using hadoop downloader and graph 2 shows
the time taken for processing images in opencv. The below is the given diagram of which HIPI
process completely happens. The conversion of jpeg to .hib is explained in the first portion where as
opencv interpretation is explained further.

HIPI in Openstack
When offloading is needed to compute a particular image processing application we need a cloud
platform to make the purpose even easier. To access the computation from any place which just needs
a minimum requirement of WIFI to transfer the images from mobile to cloud. In this module we will
discuss about deploying the image bundle using hadoop image processing interface in a cloud
environment. This can be done using implementation of openstack. OpenStack is a set of software
tools for building and managing cloud computing platforms. It lets users to deploy virtual machines
and other instances that handle different tasks for managing a cloud environment. OpenStack has the
benefit of thousands of developers all over the world working in tandem to develop the strongest,
most robust, and most secure product that they can. The necessity why openstack is needed in cloud
environment is that openstack provides infrastructure as a service by creating new instances upon
which other cloud components can run. The important aspect of this paper is deploying cloud for our
process and then proceeding to mobile communication.

Offloading system from mobile to cloud
The key requirement of this paper is to offload services that need images from mobile phone to be
transferred to cloud. The operating system which we focus here is android application to make our
job even more clarified as android as become the predominant in mobile applications. There are
certain preconditions that need to be followed in any mobile before offloading. In addition we have
a decision support system with our mobile device which is used to determine the set of images or
the processing application is compatible with openstack. [9]The preconditions are:
1. **Decision type**: The images should be offloaded or not to be offloaded.
2. **Size of code**: what is the compiled code size where serviceis initiated?
3. **Memory**: The cost of the memory that mobile consumesfor.

V. ALGORITHM
The algorithm explanation is for the processing of images using opencv and how the images are used
for hadoop image processing interface and opencv for detecting face count. The first part of algorithm
shows about hadoop map reduce where the input image bundle are converted in to hadoop image
bundle using hadoop downloader tool. The output of this process is given as input to opencv where the images are taken for detection of face count. The face detector from the cascade file with resource are used for face detector from where the native library is loaded to obtain image from image source.

**Algorithm:**

**Input:**
- Ij- jpeg,png images
- Hb- Hadoop image bundle

**Output:**
- Fc- face count performed for the input HIB images

**Algorithm:**
1. Input the set of input images in JPEG format
2. Perform map reduce procedure:
   - class Mapper
     - map(a D_id,b doc)
     - for all term ∈ doc b do
     - Emit(term t, count 1)
   - class Reducer
     - method Reduce(term t, counts [c1, c2,...])
     - sum = 0
     - for all count c in [c1, c2,...] do
     - sum = sum + c
     - Emit(term t, count sum)
3. HibDownload.sh
   
   if(inputimage < 15 Kb)
   
   { return “storage size inefficient”; }
   
   else
   
   // Use line number and a unique key assigned to each map task to generate a unique filename.
   
   String tempPath = conf.get("downloader.outpath") + key.get() + uniqueMapperKey + ".hib.tmp";
   
   Boolean = conf.getBoolean("downloader.yfcc100m", false); 
   
   // Create new temporary HIB
   
   HipiImageBundle hib = new HipiImageBundle(new Path(tempPath), conf); 
   
   // Get path to output HIB
   
   FileSystem fileSystem = FileSystem.get(conf); 
   
   Path outputPath=conf.getPath("downloader.outfile"); 
   
   // Create HIB for writing

8

1932
HipiImageBundle(outputHibPath, conf);
    HipiImageBundlehib=new hib.openForWrite(true);
    // Iterate over the temporary HIB files created by map tasks
for (Text tempString : values)
{
    // Open the temporary HIB file
Path tempPath = new Path(tempString.toString());
    HipiImageBundleinputBundle = new HipiImageBundle(tempPath, conf);
} 
4.face count.sh
   //face count actually starts if the hadoop image bundle is present
while ( hib == true )
{
    // create a face detector from the cascade file in the resources
    String xmlPath = "/home/nandhini/project/opencv-examples/lbpcascade_frontalface.xml";
    // detect faces in the image.
    MatOfRect faceDetections = new MatOfRect();
    Native_load_library is loaded to obtain the images from image source
}

CLOUD OFFLOADING ARCHITECTURE
The architecture diagram contains the entire set of components that takes place for offloading. The mobile device contains the mobile rich application that makes the application rich to perform extensively high processes. The offloading decision agent that decides the type of offloading that need to perform ie public or private cloud. The images that are sent to the resource rich server that contains private cloud which is installed with hadoop image processing interface to perform map reduce tasks parallel. When it comes to public cloud the entire set of process happens in Amazon EMR. The images that are processed are offloaded back to the mobile device.

Table1: Comparison of HIPI in mobile and server

<table>
<thead>
<tr>
<th>Feature</th>
<th>HIPI in mobile</th>
<th>HIPI in server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application possibility</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Memory sufficiency</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Processing speed support</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

If the decision agent comes up with public cloud then we use mobile data and if it is with private
cloud we use WIFI, since we send images to resource rich server which is near to the user’s location. This assures time reduction as the methodology assures faster processing with usage of WIFI. The work performance in mobile is comparatively much lower than resource rich server.

![Architecture of computation offloading](image)

*Figure 3. Architecture of computation offloading*

Some of the individual components in detail starting with resource rich server to openstack are as follows.

1. **Rich mobile application:**
Rich mobile applications are a part of rich internet applications that makes the mobile applications richer to perform various properties like context awareness. They are used mainly for mobile cloud computing where mobile applications are deployed for cloud computing. Further, mobile rich applications are used to process various applications like Natural language processing (NLP), hadoop image processing interface (HIPI), and augmented reality (AR).

2. **Offloading decision agent:**
The offloading decision agent is used to provide a quality communication between the mobile device and cloud server. Once the decision is successful the details of the user’s device are stored in server.

3. **Resource rich server:**
The entire process of offloading taken place in Resource rich server set up with openstack. It does not need to be a rack server, where it is enough to be a dual core processor with sufficient memory.

4. **Openstack:**
Openstack lets users deploy virtual machines and other instances that handle different tasks for managing a cloud environment on the fly. It makes horizontal scaling easy, which means that tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning
up more instances. For example, a mobile application that needs to communicate with a remote server might be able to divide the work of communicating with each user across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users.

5. Connectivity:
The mode of sending the input the nearby resource rich server with private cloud, we use WIFI and for public cloud in case of cloud bursting option, we use internet.

THRESHOLD CONDITIONS
The following are the threshold conditions that should be followed for the successful mobile offloading.

A. Risk Threshold
Consider two functions $d_i$ and $d_j$ where $d_i$ is the return offloading gain and $d_j$ is the risk offloading gain. When divide $d_i/d_j$then it should give a minimum possible value of threshold such that mobile device should be compatible to offload data.

B. Memory threshold
The RAM size of the mobile devices should be 4 GB and the battery should be minimum of 2000 MAh. When $R_s$be the RAM size of the system and $R_m$ be the RAM size of the mobile then $R_s>R_m$.

C. Mobile threshold
The formulated mobile device is suitable for offloading from the below formulae as

$$M(f)= M_l* M_w+(1-M_w)*E_m;$$

where

$M_l$ is the overall execution time of the system to process entire data.

$M_w$ is the coefficient weight of the mobile.

That is when battery consumption is a compulsory concern we can take higher values of $M_w$. Then if the energy drop is taken $M_w$ takes lower values. Minimum addressable value of $M_w$ is 0.5. $E_m$ is the energy consumption factor. The hardware component that should be followed for offloading has the minimum requirement of table below

Table 2: component requirements

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Mobile</th>
<th>Desktop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Dual Core</td>
<td>Quad Core</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB</td>
<td>8GB</td>
</tr>
<tr>
<td>OS</td>
<td>Android 4.2</td>
<td>Ubuntu 14.04</td>
</tr>
</tbody>
</table>

EXPERIMENTS AND RESULTS
The experiments and results we attach here is mainly for hadoop image processing interface as the changes here has the major work. The input formats and output formats are described and the image size of input images and output images are described. We have also considered the threshold
conditions for storage and risk threshold are also considered.

The images are taken in the jpeg, gif and .png format as input as taken as output into .hib format and they are map reduced. As we choose the images in input format as jpeg and png, the image size will be stored in hdfs will be more. The 12buntu image which is shown here shows the implementation of hadoop image processing interface which includes software like hadoop 2.7.1, gradle 3.1 for performing the entire set of process. The process of this conversion either in hadoop image processing interface or the openstack process is implemented in 12buntu operating system. The experimental screenshots are described directly from the processed output.

![Figure 4. HDFS before HIPI process](image1)

The above figure shows the images that are stored in the .jpeg format in HDFS. We use hibimport tool to perform the operation which loads the jpeg images in hadoop distributed file system. A sample of 100 images are taken as input here which showed a size of 20 MB. The input images stored here is taken as .txt format by the hdfs.

![Figure 5. HDFS after Images in hadoop image bundle format](image2)

The web UI that contains the images converted to .hib format is stored in the hadoop distributed file system. 100 Sample images are taken in jpeg format stored in hdfs and are converted to .hib format with respect to hadoop file system. The images with input size of 20 MB are here converted to 128. The process that takes place here is about using opencv in HIPI for converting float image to mat format. Then the system is set for the usage of opencv and from there the process of face count
actually starts. The dimension where the process is done it performs the conversion here and there by giving the user for proceeding to fetch for nodes of processing the image data.

![Figure 6. Accessing nodes for face count](image1.png)

![Figure 7. Image processing work](image2.png)

The final part of performing the process where opencv from where the map reduce procedure where the mapper part loads the native library and write the number of faces detected in the context. The reducer counts the number of faces and outputs the number of face detected. The following screenshots represents the openstack portion that implemented alongside the private cloud deployment with ubuntu image. The process further went on like doing the entire application on the cloud and giving the virtual perspective. The first image shows the final execution page of openstack that mentions the IP address of its respective components horizon and keystone. It further mentions the amount of processing time details, along with its default username and password. A user builds the image by the given IP address of the horizon. The following image shows the instance page to launch the particular image. Here we have chosen the ubuntu. This instance launch also involves network, metadata details. The following figure explains the system flavor that includes the RAM size, disk size, VCPUs etc. The following image shows the boot up page of ubuntu that starts up.
The following is the graphical representation for the HIPI process for the input image bundle. This representation is the time taken for the hadoop image processing interface to process the images in the jpeg format to hib format. The processing involves hadoop process the images for map reduces. Hence the time taken for the process involves these time.

**Table 3: Time taken for hib conversion**

<table>
<thead>
<tr>
<th>Input image Bundle (Mb)</th>
<th>Time taken for processing Hlb files(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.28</td>
</tr>
<tr>
<td>15</td>
<td>10.49</td>
</tr>
<tr>
<td>20</td>
<td>11.04</td>
</tr>
<tr>
<td>25</td>
<td>11.26</td>
</tr>
<tr>
<td>30</td>
<td>11.38</td>
</tr>
<tr>
<td>35</td>
<td>12.04</td>
</tr>
<tr>
<td>40</td>
<td>13.05</td>
</tr>
<tr>
<td>45</td>
<td>13.58</td>
</tr>
<tr>
<td>50</td>
<td>14.02</td>
</tr>
</tbody>
</table>

The graphical representation for the above table value is given below. The X axis determines the time (sec) where the Y axis determines the image bundle size.

The next table representation is for the time taken for face count processing with input image bundle in the interpretation of opencv.
Table 4: Time taken for face count

<table>
<thead>
<tr>
<th>Input image bundle (MB)</th>
<th>Time taken for face count (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4.12</td>
</tr>
<tr>
<td>15</td>
<td>4.28</td>
</tr>
<tr>
<td>20</td>
<td>4.38</td>
</tr>
<tr>
<td>25</td>
<td>4.39</td>
</tr>
<tr>
<td>30</td>
<td>4.49</td>
</tr>
<tr>
<td>35</td>
<td>4.56</td>
</tr>
<tr>
<td>40</td>
<td>4.59</td>
</tr>
<tr>
<td>45</td>
<td>5.04</td>
</tr>
<tr>
<td>50</td>
<td>5.21</td>
</tr>
</tbody>
</table>

The graphical representation of the above table is given here. The X axis determines the input image bundle in MB where the Y axis implies the time taken for face count in seconds. The image size is taken with the minimum of 5 MB and a maximum of 50 MB.

Figure 9. Time taken for face count

CONCLUSION AND FUTURE WORK

In this paper we have discussed about the service offloading mechanism that should be used with hadoop image processing interface for computation of huge image data. Using this we have given dynamic offloading strategy that even can be used in smart cities where we can offload our data in public places like railway stations where WIFI should be the minimum requirement. This has made the significantly improved the quality of service (QOS), network status and data size. In our future we will perform the computation with lower energy consumption in mobile devices. Thus offloading strategy which we are using improves the efficiency in using the image size for which the systems do not have any limitations. Using the threshold conditions, the computation is done to the best of optimized solution. Further analyses need to done on the openstack part of which the comparison of cloud deployment in various software is being sufficient and advantage for this particular work.
REFERENCES


[4] Andrey Sozyki and Timofei Epanchintsev (2015), MIPr – a Framework for Distributed Image Processing Using Hadoop, Institute of Mathematics Mechanics UrB RAS, Yekaterinburg, RussiaUral Federal University, Yekaterinburg, Russia at 9th international conference on Application on information and communication technology


[13] Uma Nandhini and Latha TamilSelvan “Computational Analytics of Client Awareness for Mobile Application Offloading with Cloud Migration” on “KSII transactions on internet and information systems.
