ISSN:0975-9646

Performance Augmentation Of Mobile Devices Using Cloud Computing

Monica Gajbhe, Dr. S. R. Sakhare
Department of Computer Engineering, VIIT- Pune,
Savitribai Phule Pune University
Pune, India

***Abstract*—Smartphones are now becoming extremely popular with an extensive range of heavy applications like gaming, video editing, and face recognition etc. These kinds of applications constantly require intensive computational power, memory, and battery. But even if Smartphones has this extensive range of applications it faces the challenge of limited processing power, memory, battery, storage, energy, etc. To overcome this problem the solution that is well known today is offloading the computation extensive task to the cloud and executing it on cloud and bringing back results on the mobile. This procedure is called as computation offloading. Many studies have been carried on till now on this computation offloading. In this paper we have proposed efficient offloading algorithm which will determine at the runtime whether to offload computation on cloud or to execute the task locally.**

***Keywords*-Smartphones; Computation; Offloading; Cloud;**

I. Introduction

Smartphone's, tablets, PDAs, wearable devices are playing significant roles in human lives due to their faultless features and productivity. The easiness of user friendly interface, high definition graphics, customized application instalment, compact size, portability, multi-card SIM facility, multi-mode service operation, multi-band connectivity, high responsiveness, multimedia application support, etc features of mobile devices have fascinated the attentions of users worldwide. The demand of mobile phones is growing day by day. The survey states that in 2014, more than 1.1 billion phones will be shipped worldwide and it is expected to rise over 1.5 billion in 2017 as predicted by International Data Corporation Market Research Company [1]. In-spite of development in computer and communication technology, mobile devices can't reach their full potential for the execution of high-end applications. These include computer vision applications, augmented reality applications, face and object recognition applications, natural language processing applications, file indexing in mobile sys-tem applications, virus scanning applications, optical character recognition applications, image and video processing applications, health monitoring applications, 3D gaming applications, etc. These applications require high processing speed, large memory and battery backup to execute efficiently and accurately. However, these application requirements are not fulfilled by mobile devices. Therefore, to improve the resource scarcity of mobile phones, the computation offloading provides the best solution in mobile cloud computing.

According to Wikipedia [15], mobile cloud computing is defined as "The state-of-the-art mobile distributed computing paradigm comprising of three heterogeneous domains of mobile computing, cloud computing, and wireless networks that aims to enhance computational capabilities of resource-constrained mobile devices towards rich user experience." The storage performance of mobile devices can be optimized by the services like Microsoft Azure [3], Amazon S3 [4], Google Drive [5], etc. In the same manner to augment the execution capabilities, compute servers are provided. Examples of the computation service providers are Amazon EC2 [16], OpenStack Compute [7], Eucalyptus [8]. The computation offloading is the method of migrating entire or partial application tasks from mobile device to the resource-abundant and powerful cloud servers for execution bringing optimization in objective function[9]. From contribution point of view, we have proposed a algorithm which will decide whether to offload the computation on cloud or run it locally. The section II deals with the Mobile Augmentation and its types. In section III we have given the Literature survey. The system architecture and Algorithm details of the proposed system are respectively explained in section IV and section V. In section VI we have given the mathematical model of the system. Section VII and Section VIII deals with the implementation and expected result discussions. The paper is concluded in section IX.

II. Mobile Augmentation And Types

A. Mobile Augmentation

Mobile augmentation in brief is the process of increasing, enhancing and optimizing computing capabilities of mobile devices by using various feasible approaches that can be hardware as well as software [7]. Mobile device is any battery-operating computing entity which is able to interact with user and execute transactions, store data and communicate using wireless technology. Augmentation approaches include hardware and software. Hardware approaches include manufacturing high-end physical components such as CPU, memory, Storage and battery. Software approaches include computation offloading, resource aware computing, remote service request, remote data storage, wireless communication and fidelity adaptation.

B. Mobile Augmentation types

There are two major types of mobile augmentation approaches.

1) Hardware:

In hardware approach Smartphone's are exploited with powerful resources, multicore CPU with high clock speed, large mobile screens, long lasting battery. However augmenting hardware is hindered by several obstacles. Smartphone's handiness decrease by generating powerful processor, large storage and big screen. It also generates additional heat, increase size and weight of the Smartphone. Using long lasting battery in small mobile is not suitable with current technology [7]. The enlargement of Smartphone resources contribute to faster battery drainage and short battery life. Also all this hardware enhancement increase the cost of mobile tremendously.

2) Software:

Software-oriented mobile augmentation approaches are classified into six groups, namely remote execution, remote storage, multi-tier programming, live cloud-streaming, resource-aware computing, and fidelity adaptation.

a) Remote Execution: In this resource-hungry components of mobile applications are mi-grated to the cloud. It conserves local re-source like battery, memory, and storage of mobile devices, but also enables execution of intense processing applications in Smart-phone.
b) Remote Storage: Remote storage is the process of expanding storage capability of mo-bile devices using remote storage resources. It enables maintaining applications and data outside the mobile devices and provides re-mote access to them.
c) Resource aware computing: In resource-aware computing efforts especially resource requirements of mobile applications are de-creased utilizing the application-level re-source management methods using application management software such as compiler and OS and lightweight protocols. Resource conservation is performed via efficient selection of available execution approaches and technologies. Ex. In energy constraint one should choose 2G over 3G since 2G consumes less energy.
d) Live Cloud Streaming: In live cloud streaming applications, entire processing take place in the cloud and results are streaming to the mobile devices. However, usability of cloud-streaming is hindered by latency, network bandwidth, portability, and network traffic cost.
e) Fidelity adaptation: Fidelity adaptation is an alternative solution to augment mobile de-vices in the absence of remote resources and online connectivity. In this method local resources are conserved by decreasing quality of application execution ex. YouTube. In YouTube choice of quality is asked to user.
f) Multi-tier Programming: The main idea in this type of mobile applications is to reduce the client-side computing workload and develop the applications with less native resource requirements. Computationally intensive components of the applications are executed outside the device, whereas the user interface and native codes e.g., accessing to the device camera remain inside the device for execution.

III. Motivation

Smart phones are now used by 80% of the population in the world [11]. Mobile execution platform is being used for more and more tasks like games, capturing, micro payments etc Mobile phones are now capable of performing many complex task that are useful for the mobile users. But there is limitation to the use of these mobile devices. But mobile devices will always be resource-poor (CPU, storage capacity), less secure, with unstable connectivity and with less energy. Offloading task on mobile devices to cloud can optimize the execution time by utilizing cloud resources.

IV. Relevance In Current Scenario

Mobile phones are set to become the universal interface to online services and cloud computing applications. However, using them for this purpose today is limited to two configurations: applications either run on the phone or run on the server and are remotely accessed by the phone. These two options do not allow for a customized and flexible service interaction, limiting the possibilities for performance optimization as well. Mobile devices are resource sensitive. Some applications on mobile consume large computation power of mobile decrease its performance which can include various factors like battery, memory etc. So offloading some task from mobile to cloud server can enhance the performance of the mobile. But it is not always feasible to upload all the application on cloud. So we will be developing the cost model which will calculate the offloading cost of the application and according to the cost it will decide whether to upload or not.

V. Literature Survey

1) A study on virtual machine deployment for application outsourcing in mobile cloud computing.

This paper [8], explores about offloading the computational load to cloud server nodes. To offload the computational resources they have ensured that the VM deployment and management will require additional computing resources on mobile devices. These paper analyses the impact of VM deployment on the execution time of application. They have used CloudSim for VM deployment and management for application processing in simulation environment. CloudSim is the simulation toolkit that provides simulation framework to model the working of VM deployment and management for application processing in cloud computing infrastructure. The advantages are this work is that it can save power and have an efficient bandwidth utilization .It also support User preferences and provides less execution cost. Disadvantages are the additional overhead for deployment and utilization of more computing re-sources and increase in execution time.

2) CloneCloud.

CloneCloud [1] is fine grained and thread-level. It is cloud-based, application partitioner .It migrates entire mobile platform into the cloud VM. It executes the mobile application inside the VM without performing any changes in the application code. It allows the local execution of remaining mobile application when remote server is running the intensive components. CloneCloud used a combination of static analysis and dynamic profiling to partition applications to optimize execution time and energy. The disadvantage of this approach is the Communication overhead, in frequent synchronization of the shared data between the mo-bile and cloud. Decrease the power of cloud is another disadvantage in this paper.

3) The case of VM-based cloudlets in mobile Computing

Virtual machine (VM) technology is been explained in this architecture to rapidly instantiate customized service software on a nearby cloudlet and by using these services over a wireless LAN .

The mobile device will act like a thin client with respect to the service. A cloudlet is a trusted and resource rich computer or cluster of computers that's available for use by nearby mobile devices by connecting to the internet.

By using cloudlet we can simplify the challenge of meeting the peak bandwidth demand of many users interactively generating and receiving multimedia such as high-definition video and high-resolution images. Speedy customization of infrastructure for varied applications emerges as a critical requirement, and our results from a proof-of-concept model suggest that VM technology can indeed help meet this necessity. Here we have to concentrate in reducing the WAN delays, Congestion and Failures occurring in the cloudlets.

4) Tactic based remote execution for mobile computing

In this paper [10], the author bring in a compact declarative form called Tactic in which automated dynamic repartitioning of mobile applications can be reconciled and the useful knowledge about an application relevant to remote execution is being captured. Chroma is a tactic based remote execution system that performs comparably to a runtime system that makes perfect partitioning decisions. Here the author also shows that Chroma can automatically utilize extra resources in an over provisioned environment to enhance application performance. In this paper, they have considered on three applications of the above technique (natural language translation, speech recognition, and face recognition) and shown that the tactics for each is much less than one percent of total code size. In addition, Chroma can optionally make use of extra resources in an over-provisioned environment and allows us to achieve lower latencies for the three applications.

5) Cuckoo: a Computation Offloading Framework for Smartphone

This paper [11] aims to enhance the performance of Smartphone application by reducing the energy usage. The author proposes a Cuckoo framework by the development of Smartphone applications that can undertake computation offloading and provides a dynamic runtime system, which decide at runtime, whether a part of an application will be executed locally or remotely. It is developed in Android platform. A resource manager application for Smartphone users and a programming model for developers, which integrates into the Eclipse, build system. It gives maximum computation speed and minimum energy usage. There are some limitations shown in this paper. Here Cuckoo does not still support callbacks. It does not support any form of security. It supports only stateless service. Cuckoo can perform both early and late binding to remote resources.

6) Using History to Improve Mobile Application Adaptation

In this paper [3], the authors use historical application log data to predict the fidelity of an application. Fidelity is the application-specific notion of the integrity of a computed result or data object and also decides its resource consumption. A lower fidelity results in lower resource consumption. The ultimate objective of fidelity adaptation is to improve a mobile users computing experience by delivering result fast, with low battery drainage and little destruction of the user. Augmented Odyssey is an operating system platform for adaptation, with a history based prediction system that monitors, logs and predicts application resource consumption as a function of the fidelity.

The current prototype has a CPU overhead of 0.22 percent for a typical application. The disadvantage is that acquired history logs are only for hardware platform that they might ever use.

7) The case of cyber foraging

In this paper R. Balan et.al [9] looks at the concept of cyber foraging and talks about specifically on how the use of surrogates can help in two distinct situations. First, he demonstrates how it can reduce cache miss service times in mobile file access. Second, he shows how it enables compute-intensive applications like language translation and augmented reality to execute on mobile hardware.

Cyber foraging is a common approach studied by many to augment the capability of resource constrained mobile devices. The fundamental idea is to dynamically discover and make use of nearby resources, aka surrogates, to offload the execution of an application or parts of an application running on a mobile device. The benefit of this research is increased resource availability and dynamic remote execution. The disadvantage is they are inconsistent and they need application developer to modify the application.

8) MAUI: Making Smartphones last longer with code offload

MAUI is fine-grained energy aware offloading technique in which the application is executed either locally in the mobile devices or remotely migrated to the cloud. It uses code portability method to maximize the energy saving. The authors focus on the programming reflection and make use of serialization to decide its networking costs. The application is partitioned dynamically in this paper. There are some limitations of this method. While using the power-save mode, it can impair the overall energy consumption. It takes a considerable amount of time for programmer to annotate offloadable methods.

9) Towards an elastic application model to augment the computation capabilities of mobile devices with cloud computing

The author here proposed an elastic framework that uses the cloud resources to run the resource intensive component of the mobile device. In this model, they have created Weblets by partitioning the mobile application into small components. In order to increase systems robustness by decreasing the communication operating cost and latency, the author made the weblets with least dependency with each other.

The weblet execution is dynamically configured together perform locally or remotely, based on the weblets resource partitioning, execution configuration quality, and criteria of offloading. The advantage in this paper is that in order to improve the overall execution performance and improve user experience, the system is able to execute the weblets both locally and remotely. Elastic application model gives more attention to the user preferences by enabling different running modes of a single application.

10) MCACC: New Approach for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing

This paper [6] proposes framework called Mobile Capabilities Augmentation using Cloud Computing (MCACC). In this framework, any mobile application is divided into a group of services, and then each of them is either executed locally on the mobile or remotely on the Cloud based a novel dynamic offloading decision model. Here, the decision is based on five real-time metrics: total execution time, energy consumption, remaining battery, memory and security. The extensive simulation studies prove that both heavy and light applications can benefit from this proposed model while saving energy and improving performance compared to previous techniques. The proposed MCACC turns the Smartphone's to be smarter as the offloading decision is taken without any user interaction.

VI. Proposed System Architecture

An illustration of our architecture is provided in Fig 5.1 Our system architecture consists of five key components including profiler, decision manager, offloading manager, local execution manger and remote execution manger. Among all of these components the decision manager is the core and the area of extensibility.

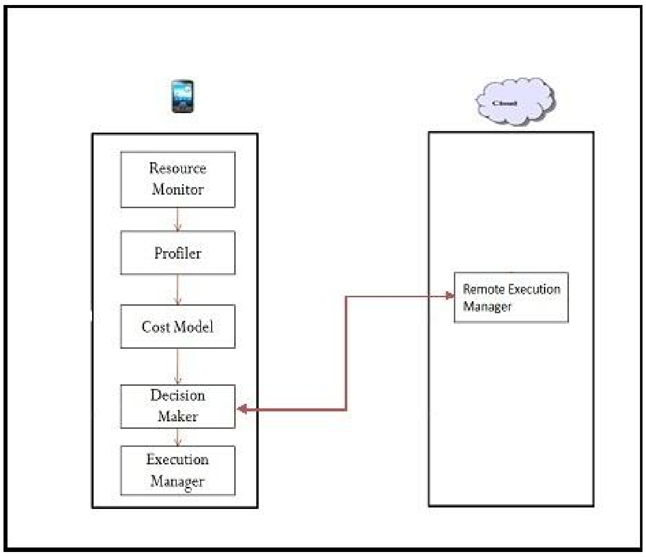


Fig. 1. Proposed System Architecture

1. Profiler: It is used to get profiler information from Smartphone's. It is used to get values of various decision metrics like available memory, remaining battery, CPU type and task complexity.

2. Decision Manager: It decides whether to offload the application to cloud or not. Application checks profiler and triggers the offloading decision making algorithm. When it decides to run application locally it calls local execution manager otherwise it calls offloading manager to hand over offloading computation to cloud.

3. Offloading Manager: It sends required input data to remote execution manager and receives results returned by it.

4. Local Execution Manager and Remote Execution Manager: Local and remote execution managers are mainly responsible to manage execution of the application. Local execution manager executes the application on Smartphone itself. When remote execution manager gets input data from offloading manager then it executes offloading computation on cloud and send results back to offloading manager.

VII. Workflow Of Proposed Decision Making Algorithm

Offloading decision making algorithm is core part of our research. Our offloading decision making algorithm is based on various real time metrics like remaining battery level, task complexity, processor type, file size and available memory. Fig 5.2 shows workflow of offloading decision engine.

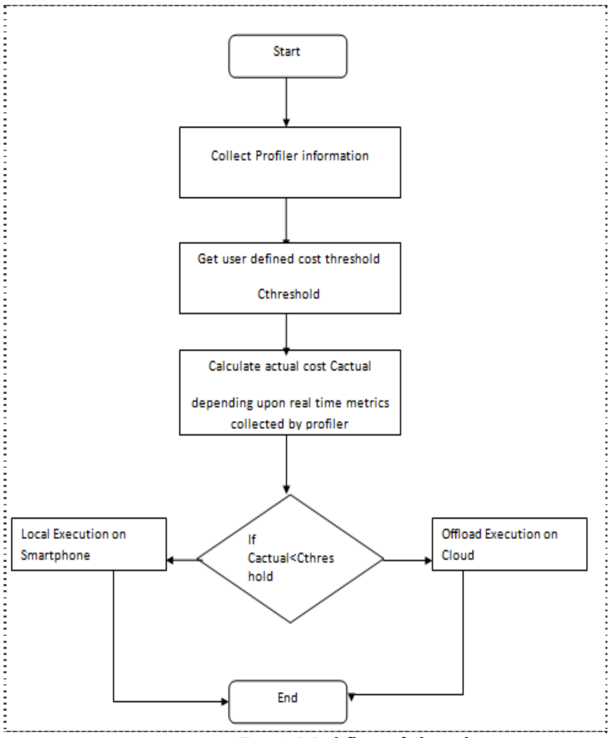

Start
Collect Profiler information
Get user defined cost threshold
Cthreshold
Calculate actual cost Cactual
depending upon real time metrics
collected by profiler
Local Execution on Smartphone
If Cactual<Cthreshold
Offload Execution on Cloud
End

Fig. 2. Workflow of algorithm

- Step 1: It gets profiler information like application size, task complexity, processor type, remaining battery level and available memory.
- Step 2: It gets users cost threshold Cthreshold specified by user.
- Step 3: It calculates cost upon various real time metrics defined using following formula:
  Cactual = (FileSize _ TaskComplexity _ ProcessorType)=(RemainingBatteryLevel _

Available Memory)

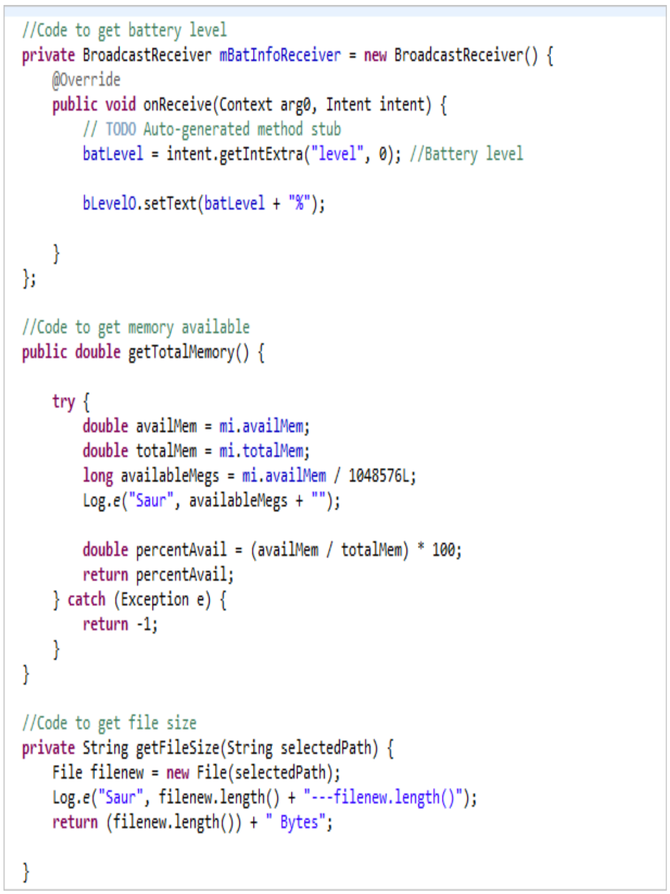
- Step 4: In this we compares user defined cost threshold with calculated cost and if calculated cost is less than that of Cthreshold then it decides run application locally on Smartphone and then stops. Otherwise it decides to offload computation from Smartphone to cloud and algorithm ends.

VIII. IMPLEMENTATION

In environmental setup, at the Smartphone side we are using Sony Xperia M dual with android operating system in version 4.2. It has a Qualcomm MSM8227 CPU with 1Ghz frequency, 2GB memory. We use Wi-Fi network to connect to cloud in our experiments. As to the cloud we use a laptop running Hadoop in version 2.3.0 in windows operating system (Windows 8.1) to serve as a cloud. It has Intel core i5-4210U CPU with 1.70 GHz frequency, 8 GB memory and
1 TB hard disk.

Following are the code snippets from project:

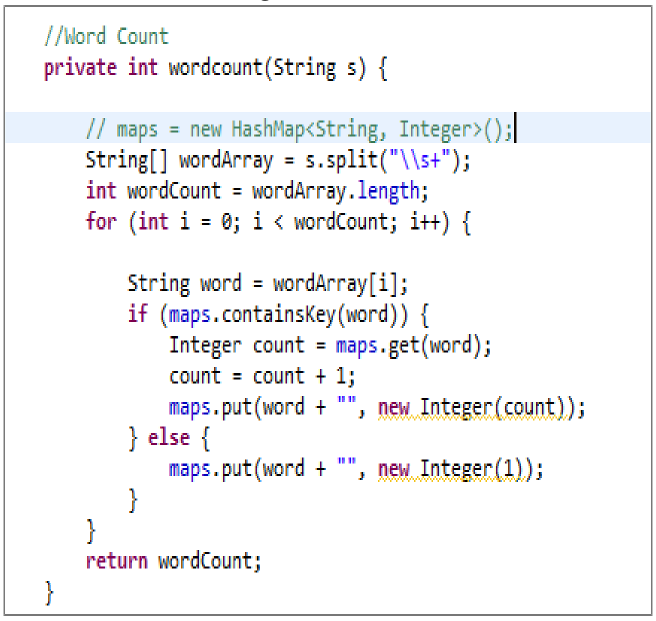
1. Profiler Code:



```
//Code to get battery level
private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context arg0, Intent intent) {
        // TODO Auto-generated method stub
        batLevel = intent.getIntExtra("level", 0); //Battery level

        bLevel0.setText(batLevel + "%");

    }
};

//Code to get memory available
public double getTotalMemory() {

    try {
        double availMem = mi.availMem;
        double totalMem = mi.totalMem;
        long availableMegs = mi.availMem / 1048576L;
        Log.e("Saur", availableMegs + "");

        double percentAvail = (availMem / totalMem) * 100;
        return percentAvail;
    } catch (Exception e) {
        return -1;
    }
}

//Code to get file size
private String getFileSize(String selectedPath) {
    File filenew = new File(selectedPath);
    Log.e("Saur", filenew.length() + "---filenew.length()");
    return (filenew.length()) + " Bytes";

}
```

Fig. 3. Profiler code

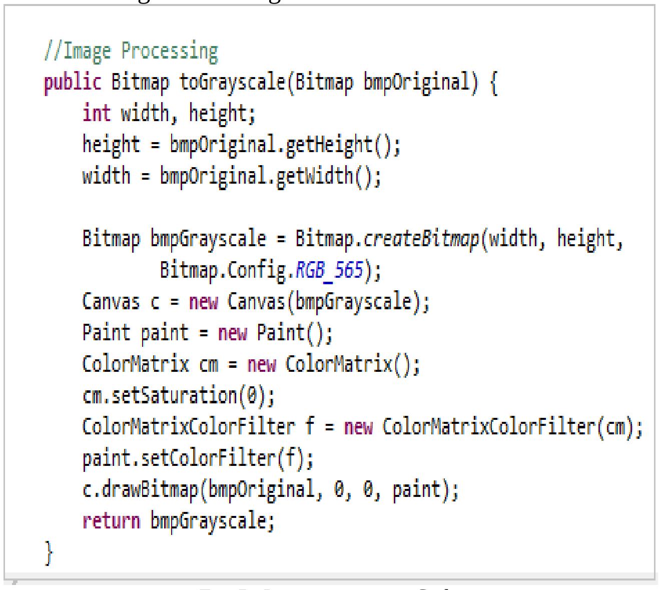
2. Word Processing Code:



```
//Word Count
private int wordcount(String s) {

    // maps = new HashMap<String, Integer>();
    String[] wordArray = s.split("\\s+");
    int wordCount = wordArray.length;
    for (int i = 0; i < wordCount; i++) {

        String word = wordArray[i];
        if (maps.containsKey(word)) {
            Integer count = maps.get(word);
            count = count + 1;
            maps.put(word + "", new Integer(count));
        } else {
            maps.put(word + "", new Integer(1));
        }
    }
    return wordCount;
}
```

Fig. 4. Word processing Code

3. Image Processing Code:



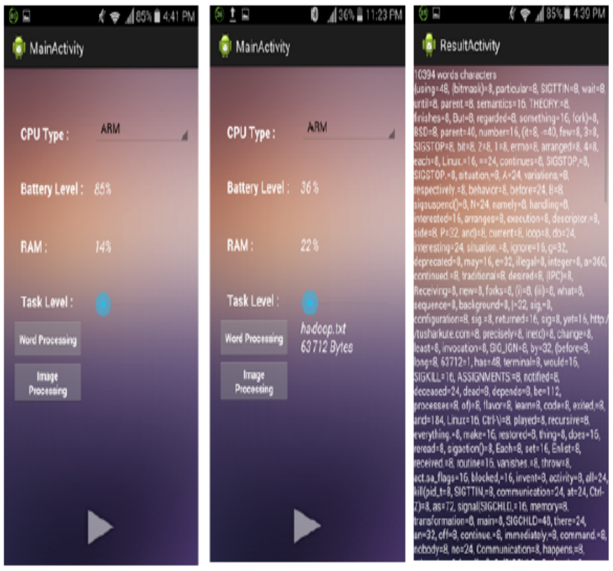
```
//Image Processing
public Bitmap toGrayscale(Bitmap bmpOriginal) {
    int width, height;
    height = bmpOriginal.getHeight();
    width = bmpOriginal.getWidth();

    Bitmap bmpGrayscale = Bitmap.createBitmap(width, height,
            Bitmap.Config.RGB_565);
    Canvas c = new Canvas(bmpGrayscale);
    Paint paint = new Paint();
    ColorMatrix cm = new ColorMatrix();
    cm.setSaturation(0);
    ColorMatrixColorFilter f = new ColorMatrixColorFilter(cm);
    paint.setColorFilter(f);
    c.drawBitmap(bmpOriginal, 0, 0, paint);
    return bmpGrayscale;
}
```

Fig. 5. Image processing Code

IX. RESULTS OBTAINED

1. Word Processing Result on Local:

The fig shows the result of execution of wordcount program on mobile device.



10394 words characters
{using=48, (bitmask)=8, particular=8, SIGTTIN=8, wait=8, until=8, parent.=8, semantics=16, THEORY:=8, finishes=8, But=8, regarded=8, something=16, fork)=8, BSD=8, parent=40, number=16, (it=8, -=40, few=8, 3=8, SIGSTOP=8, bit=8, 2=8, 1=8, errno=8, arranged=8, 4=8, each=8, Linux.=16, ==24, continues=8, SIGSTOP,=8, SIGSTOP.=8, situation,=8, A=24, variations,=8, respectively.=8, behavior=8, before=24, B=8, sigsuspend()=8, N=24, namely=8, handling=8, interested=16, arranges=8, execution=8, descriptor.=8, side=8, P=32, and)=8, current=8, loop=8, do=24, interesting=24, situation.=8, ignore=16, g=32, deprecated=8, may=16, e=32, illegal=8, integer=8, a=360, continued.=8, traditional=8, desired=8, (IPC)=8, Receiving=8, new=8, forks=8, (i)=8, (ii)=8, what=8, sequence=8, background=8, I=32, sig,=8, configuration=8, sig.=8, returned=16, sig=8, yet=16, http://tusharkute.com=8, precisely=8, inetd)=8, change=8, least=8, invocation=8, SIG_IGN=8, by=32, (before=8, long=8, 63712=1, has=48, terminal=8, would=16, SIGKILL=16, ASSIGNMENTS:=8, notified=8, deceased=24, dead=8, depends=8, be=112, processes=8, of)=8, flavor=8, learn=8, code=8, exited,=8, and=184, Linux=16, Ctrl-\=8, played=8, recursive=8, everything.=8, make=16, restored=8, thing=8, does=16, reread=8, sigaction()=8, Each=8, set=16, Enlist=8, received.=8, routine=16, vanishes.=8, throw=8, act.sa_flags=16, blocked,=16, invent=8, activity=8, all=24, kill(pid_t=8, SIGTTIN,=8, communication=24, at=24, Ctrl-Z)=8, as=72, signal(SIGCHLD,=16, memory=8, transformation=8, main=8, SIGCHLD=48, there=24, an=32, off=8, continue.=8, immediately,=8, command.=8, nobody=8, no=24, Communication=8, happens,=8,

Fig. 6. Word Processing Result on Local

2. Image Processing Result on Local:

The fig shows the result of image processing program on mobile.

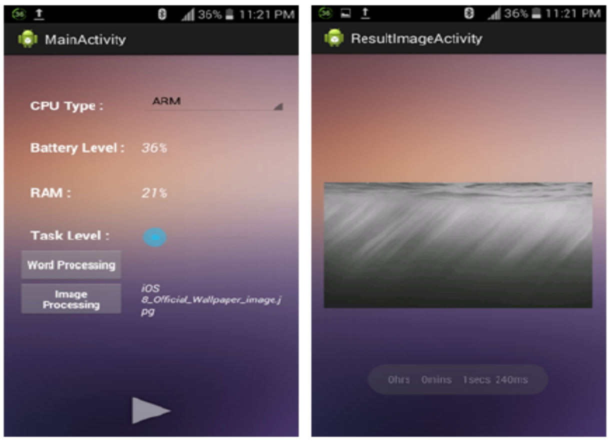
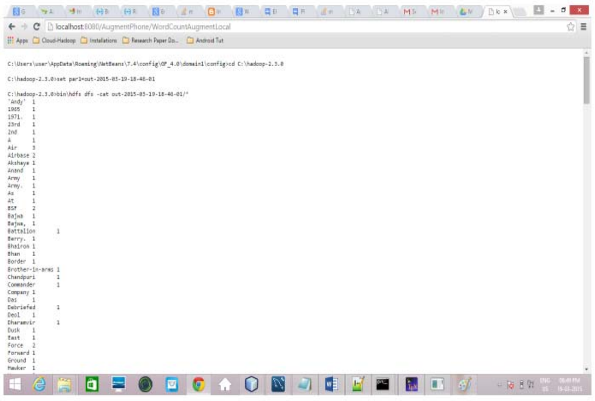


Fig. 7. Image Processing Result on Local

3. Word Processing Result on Cloud:

The fig shows the result obtained by running the application on cloud.

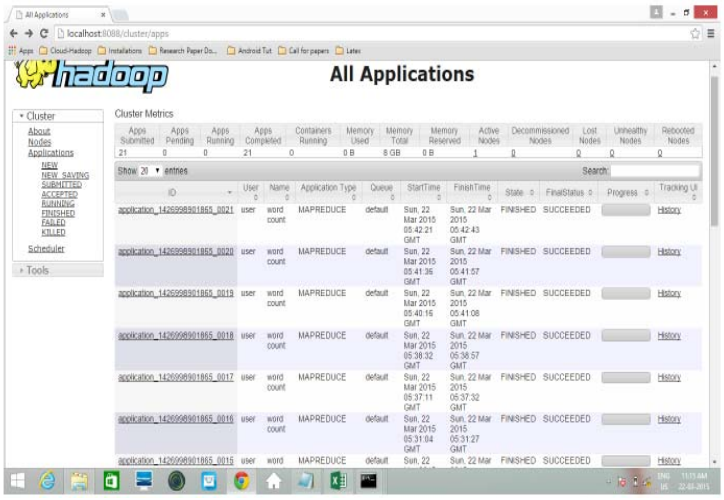


```
C:\Users\user\AppData\Roaming\NetBeans\7.4\config\GF_4.0\domain1\config>cd C:\hadoop-2.3.0

C:\hadoop-2.3.0>set par1=out-2015-03-19-18-46-01

C:\hadoop-2.3.0>bin\hdfs dfs -cat out-2015-03-19-18-46-01/*
'Andy'  1
1965    1
1971.   1
23rd    1
2nd     1
A       1
Air     3
Airbase 2
Akshaye 1
Anand   1
Army    1
Army.   1
As      1
At      1
BSF     2
Bajwa   1
Bajwa,  1
Battalion       1
Berry.  1
Bhairon 1
Bhan    1
Border  1
Brother-in-arms 1
Chandpuri       1
Commander       1
Company 1
Das     1
Debriefed       1
Deol    1
Dharamvir       1
Dusk    1
East    1
Force   2
Forward 1
Ground  1
Hawker  1
```

Fig. 8. Word Processing Result on Cloud

4. Applications Running on Cloud:

The fig shows the no of applications running on cloud.



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
21	0	0	21	0	0 B	8 GB	0 B	1	0	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1426998901865_0021	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:42:21 GMT	Sun, 22 Mar 2015 05:42:43 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0020	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:41:36 GMT	Sun, 22 Mar 2015 05:41:57 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0019	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:40:16 GMT	Sun, 22 Mar 2015 05:41:08 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0018	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:38:32 GMT	Sun, 22 Mar 2015 05:38:57 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0017	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:37:11 GMT	Sun, 22 Mar 2015 05:37:32 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0016	user	word count	MAPREDUCE	default	Sun, 22 Mar 2015 05:31:04 GMT	Sun, 22 Mar 2015 05:31:27 GMT	FINISHED	SUCCEEDED		History
application_1426998901865_0015	user	word	MAPREDUCE	default	Sun, 22	Sun, 22 Mar	FINISHED	SUCCEEDED		History

Fig. 9. Applications Running on Cloud

The fig shows the execution time of Word Processing Application under different input sizes on Cloud and Smartphone.

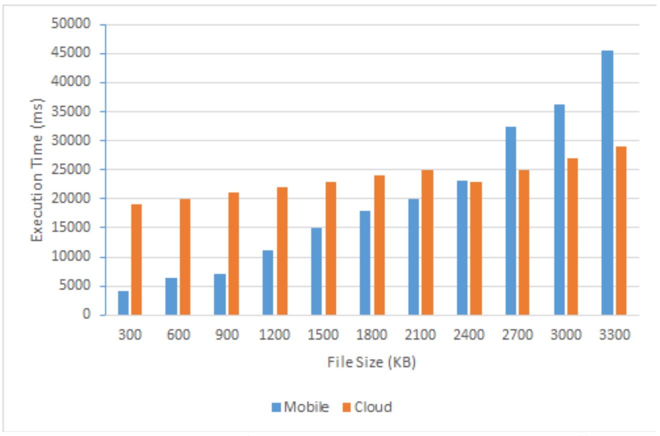


Fig. 10. Execution time of Word Processing Application under different input sizes on Cloud and Smartphone.

X. CONCLUSION

Augmenting computing capabilities of mobile devices, especially Smartphone's using cloud infrastructures and principles is an emerging research area In this project we try to augment the battery life and reduce CPU consumption. We are sure that our proposed architecture will help us to achieve that.

ACKNOWLEDGEMENT

I wish to express my sincere gratitude to the administration of Department of Computer Engineering of VIIT, Pune, India and SPPU University for providing the academic environment to pursue research activities. In particular I would like to thank Dr. S.R. Sakhare, (Head of department) Computer Engineering (VIIT, Pune) for his invaluable guidance.

REFERENCES

[1]. B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, `CloneCloud: Elastic execution between mobile device and cloud,' in Proc. ACM EuroSys11, Salzburg, Austria, pp. 301314, 2011.
[2]. D. Huang, `Mobile cloud computing,' IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter, vol. 6,no. 10, pp. 2731, 2011.
[3]. D. Narayanan, J. Flinn, and M. Satyanarayanan. "Using history to improve mobile application adaptation." In Proc. of the 3rd IEEE Workshop on Mobile Computing Systems and Applications, 2000.
[4]. E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu,R. Chandra, and P.Bahl, `Maui: making smartphones last longer with code offload,' in Proc. 8th international conference on Mobile systems, applications, and services. ACM, pp. 4962, 2010.
[5]. Kumar and Y.-H. Lu, `Cloud computing for mobile users: Can offloading computation save energy?',Computer, vol. 43, no. 4, pp. 5156, 2010
[6]. M Elgendy, A Shawish, M Moussa,`MCACC: New Approach for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing',Science and Information Conference August 27-29, 2014
[7]. M Shiraz, S Ablofazli, Z Sanaei, A Gani "A study on virtual machine deployment for application outsourcing in mobile cloud computing. In springer,2012.
[8]. M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies,`The case for vm-based cloudlets in mobile computing', IEEE Pervasive Computing, (4), 2009.
[9]. R. Balan, J. Flinn, M. Satyanarayanan, S.Sinnamohideen, and H. Yang, `The case for cyber foraging.'In Proc. of the 10th ACM SIGOPS European Workshop , 2002.
[10]. R. K. Balan, M. Satyanarayanan, S. Park, and T.Okoshi, "Tactics-based remote execution for mobile computing." in Proc. ACM MobiSys'03, San Francisco, California, USA, 2003, pp. 273–286.
[11]. Roelof Kemp, Nicholas Palmer, Thilo Kielmann and Hentri Bal, `Cuckoo: A Computation o_oading Framework for Smartphones' in Springer,2012.
[12]. Saeid Abolfazli, Zohreh Sanaei, Alizadeh, Abdullah, Feng Xia, `An Experimental Analysison Cloud-based Mobile Augmentation in Mobile Cloud Computing', IEEE Transactions on Consumer Electronics, vol. 60, no. 1, February 2014
[13]. Saeid Abolfazli, Zohreh Sanaei, Ejaz Ahmed, Abdullah, Rajkumar, `Cloud-Based Augmentation for Mobile Devices:Motivation, Taxonomies, and Open Challenges', IEEE COMMUNICATIONS SURVEYS AND TUTORIALS,vol. 16,no. 1, FIRST QUARTER 2014
[14]. X. W. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, `Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing,' Mobile Networks and Applications, vol. 16, no. 3, pp. 270284, 2011.
[15]. M. Hogan, F. Liu, A. Sokol, and J. Tong, "NIST Cloud Computing Standards Roadmap," Jul. 2011. [Online]. Available: http://www.nist.gov/itl/cloud/upload/NIST SP-500-291 Jul5A.pdf
[16]. The Amazon Elastic Computing website. [Online]. Available:http://aws.amazon.com/ec2/. 2013.
[17]. Webpage on Android Developer AIDL [Online]. Available: http://developer.android.com/guide/components/aidl.html. 2013.
[18]. M Gajbhe, S.Sakhare, 'A Survey on Mobile Cloud Augmentation,' in IRD , vol. 3 pp. 40, 2015.
[19]. M Gajbhe, S.Sakhare, 'Augmenting Performance Of Mobile Devices Using Cloud Computing ,' in Cpgcon , 2015.