Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

MEACC: an energy-efficient framework for smart devices using cloud computing systems*#

Khalid ALSUBHI[1], Zuhaib IMTIAZ[2], Ayesha RAANA[3], M. Usman ASHRAF[†‡3], Babur HAYAT[4]

[1]*Department of Computer Science, King Abdulaziz University, Saudi Arabia*

[2]*Department of Computer Science, University of Sialkot, Sialkot 51310, Pakistan*

[3]*Department of Computer Science, University of Management and Technology, Sialkot 51310, Pakistan*

[4]*Department of Computer Science, University of Lahore, Gujrat 50700, Pakistan*

[†]E-mail: usman.ashraf@skt.umt.edu.pk

Received Apr. 17, 2019; Revision accepted Oct. 11, 2019; Crosschecked May 20, 2020

Abstract: Rapidly increasing capacities, decreasing costs, and improvements in computational power, storage, and communication technologies have led to the development of many applications that carry increasingly large amounts of traffic on the global networking infrastructure. Smart devices lead to emerging technologies and play a vital role in rapid evolution. Smart devices have become a primary 24/7 need in today's information technology world and include a wide range of supporting processing-intensive applications. Extensive use of many applications on smart devices results in increasing complexity of mobile software applications and consumption of resources at a massive level, including smart device battery power, processor, and RAM, and hinders their normal operation. Appropriate resource utilization and energy efficiency are fundamental considerations for smart devices because limited resources are sporadic and make it more difficult for users to complete their tasks. In this study we propose the model of mobile energy augmentation using cloud computing (MEACC), a new framework to address the challenges of massive power consumption and inefficient resource utilization in smart devices. MEACC efficiently filters the applications to be executed on a smart device or offloaded to the cloud. Moreover, MEACC efficiently calculates the total execution cost on both the mobile and cloud sides including communication costs for any application to be offloaded. In addition, resources are monitored before making the decision to offload the application. MEACC is a promising model for load balancing and power consumption reduction in emerging mobile computing environments.

Key words: Offloading; Smart devices; Cloud computing; Mobile computing; Power consumption
https://doi.org/10.1631/FITEE.1900198 **CLC number:** TP393

1 Introduction

Over the last few decades, an exponential growth in smart mobile phones and their application usage has been recorded around the globe. The smart phone has been adopted as an appropriate and probably the optimum tool for communication using voice, text, and video. The use of the smart mobile phone is now expanding in business applications as well (Cao and Cai, 2017; Paranjothi et al., 2017; Sookhak et al., 2017; Zhou and Buyya, 2018). However, despite the growing demand for smart phone use and its applications, these devices have fallen short of passable resources like computational capacity and battery power. At the user end, there is a trade-off between mobile devices that use many resources and resource-constrained mobile devices (Cao and Cai, 2017).

‡ Corresponding author

* Project supported by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia (No. D-154-611-1440)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.1900198) contains supplementary materials, which are available to authorized users

ORCID: M. Usman ASHRAF, https://orcid.org/0000-0001-7341-8625

To mitigate this problem, different energy augmentation techniques have been proposed. These techniques have targeted the resource constraints in smart phones. For example, the concepts of "interprocess communication" and "distributed computing" were introduced to control computational resources among distributed devices. An emerging paradigm, cloud computing, uses virtualization technology to provide distributed computing resources to ease the computation burden on resource-constrained devices. The cloud virtualization technology associated with mobile applications is called mobile cloud computing (MCC). It provides enough storage and computing power outside the mobile device. Mobile cloud can be used to enhance the performance of multiple computing-intensive applications such as video games and scientific and natural language processing applications. Fig. 1 shows the general architecture of MCC.

The migration of tasks from the mobile device to the cloud to preserve mobile resources is known as offloading. In MCC, smart device offloading helps solve mobile computing resource issues using distributed resource providers rather than the mobile device itself. The offloading of some mobile device computations enhances the computation efficiency due to more storage and power capabilities over cloud servers. In computation offloading, entire or partial application tasks are migrated from the smart device to the resource-intensive cloud for execution, optimizing the objective (Kovachev and Klamma, 2012).

Numerous studies have focused on the challenges and problems of outsourcing computations from mobile devices to cloud servers for efficient use of heterogeneous computing resources. These challenges include selection of an optimal augmentation technique (Satyanarayanan et al., 2009; Cuervo et al., 2010; Chun et al., 2011; Kosta et al., 2012), mobile device battery life, Internet connection stability, service level agreement (SLA) violation when the number of mobile devices increases, mobile device mobility (Zhou and Buyya, 2018), and data security (Zissis and Lekkas, 2012). The above-mentioned challenges impact the overall performance (Robinson, 2009) of different techniques that have been proposed to mitigate the issues under consideration.

In the past few decades, the issues related to mobile cloud computing and mobile edge computing have become an interesting area of research. Different frameworks and optimal algorithms, especially bio-inspired algorithms, have been used to solve the problem of offloading. However, existing works have focused on execution cost, but have not considered the communication cost of offloading an application to the cloud.

In the approach proposed in this study, the communication cost and execution cost are evaluated for a file to be offloaded to cloud servers. While calculating the execution cost, different attributes are considered including CPU usage, random access memory (RAM) memory usage, and battery consumption for file execution. In contrast, the

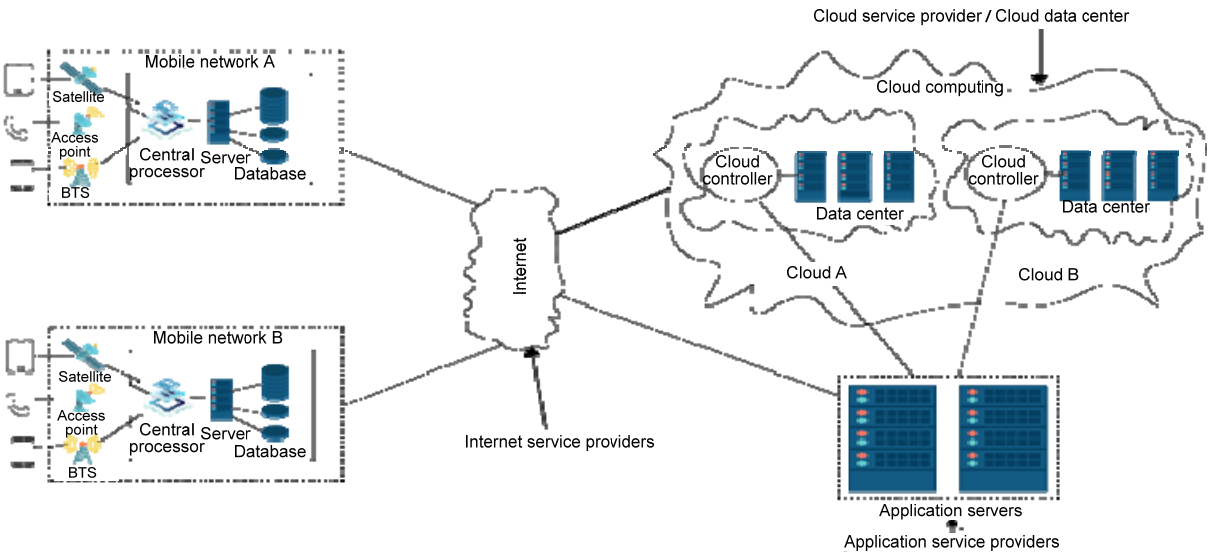

Fig. 1 Mobile cloud computing architecture

communication cost includes the response time, i.e., the time taken by any file to be offloaded to the cloud and give a response. Our contributions can be summarized as follows:

1. We prepare a Hadoop simulation environment for calculation of CPU performance, power consumption, and execution cost.

2. We propose a new approach named MEACC to make job offloading decisions. MEACC contains two algorithms, one for making offloading decisions and the other for file computation. The offloading decision algorithm is the baseline to determine a final decision to offload any task or not. The file computation supporting algorithm is used to complete the profiler log information; it reads the inputted file word by word and provides a total word count for the file.

3. We validate the proposed approach by performing a critical comparative analysis of our approach and state-of-the-art frameworks.

2 Background material and literature review

We now give a brief background of mobile cloud computing and related technologies used in this study.

2.1 Technology background

2.1.1 Cloud computing

In cloud computing, the computing resources, such as the processor, memory, and storage, are not physically present at the user's location. Instead, these resources are owned and managed by a cloud service provider and users access resources via the Internet (Creeger, 2009).

The cloud computing (CC) model enables convenient, on-demand network access to a shared pool of configurable computing resources. These resources may include networks, servers, storage, applications, and services. These can be highly provisioned and released with minimal management effort of the service provider (Mell and Grance, 2011). CC can be considered a combination of different networks, interfaces, hardware, storage, and services. For example, it provides many social networks such as Facebook and Twitter.

2.1.2 Mobile cloud computing

MCC is a combination of cloud computing, mobile computing, and wireless networks. In the mobile cloud, both data storage and data processing are outside the mobile device. In other words, the integration of cloud computing services and the mobile environment to put all types of cloud services and facilities on the mobile user's dashboard is called MCC.

MCC is used for resource-intensive computations and data requiring large storage space and great processing power for execution. These computations are shifted from the mobile device to the cloud, which provides powerful and centralized computing platforms. Cloud resources are accessed by any smart device over an Internet connection using an Android/Mac app or web browser. A graphical representation of MCC is as shown in Fig. 1. Moreover, mobile computing saves the power for data processing and storage on a mobile phone by moving these functions to the cloud. This feature of MCC attracts many users and, as a result, increases the number of mobile cloud subscribers. The major advantage of MCC is that mobile devices do not need powerful configurations (e.g., CPU processing and storage capacity) because any complicated module can be processed using the mobile cloud.

2.1.3 MCC features for smart-device users

MCC is of great interest to mobile users and service providers because of its several advantages (Dinh et al., 2013). Some important features are outlined below:

1. High storage capacity. Mobile devices have limited storage capacity. Considering this, MCC was developed to allow mobile users to store and retrieve a large amount of data in cloud storage through the Internet. For example, Google Photos allows users to put unlimited photos and videos on the cloud at any time and retrieve their uploaded photos/videos from any device just by accessing their account. Users can save a considerable amount of energy and storage space on their mobile devices because all images/videos are sent to and processed in the cloud.

2. Enhanced processing power. The applications that take a long time for their complete execution consume too many mobile resources. MCC helps minimize the execution cost for such resource-intensive applications and adaptively supports many tasks for data warehousing, managing, and

synchronizing multiple documents online. For example, transcoding, playing chess, and broadcasting multimedia services are supported by the cloud. In the written examples, all the complicated calculations for transcoding or an ideal chess move that may take a very long time when performed using the resources of smart devices are processed quickly in the cloud.

3. Enhanced reliability. Usually, data is stored only in mobile storage, where there is a great chance of loss due to system disaster, virus attack, or any such unanticipated reason. In contrast, when using mobile cloud storage, mobile data and applications are stored and backed up on more than one cloud server, which reduces the chance of data/application loss. Also, the cloud can remotely offer security services such as virus scanning, malicious code detection, and authentication, which in return increases the reliability of cloud resources.

4. Extended battery life of smart devices. Applications that require long execution time, complex processing, or large computations are migrated to cloud servers using computation offloading techniques to save the battery of end users' devices.

2.2 Literature review

Comprehensive work has been done in the field of MCC to offload heavy applications from smart devices to the cloud in an effort to augment the energy of these smart devices. A lot of frameworks, algorithms, and architectures have been developed using various cloud computing and Android supportive techniques. The objective of these proposed approaches is to enhance the energy efficiency and battery life of smart devices. This is done by offloading complete applications to the mobile cloud or by using the concept of elastic mobile applications (Zhang Q et al., 2010; Zhang XW et al., 2010).

Creeger (2009) and Kovachev and Klamma (2012) proposed a framework for computational offloading in MCC, named mobile augmentation cloud services. It supports dynamic application partitioning. Similarly, Mell and Grance (2011) calculated an offloading cost, and then decided whether to offload an application or not. Dinh et al. (2013) emphasized the security issues that occur when mobile devices communicate with the cloud. Retinal image recognition based biometric authentication was used for offloading and secure access to the cloud (Kovachev and Klamma, 2012).

Zhang L et al. (2017) focused on how energy offloading technology can be employed to save the energy in mobile devices, especially for real-time applications like video calling and live video games. Today's smart mobile devices support several advanced applications and can be used to perform a variety of massive computation tasks. Energy demand of mobile devices is increasing with the introduction of resource-intensive applications. Consequently, mobile devices consume more energy, especially during real-time video applications. Batteries in mobile devices do not have enough capacity to support extensive energy needs. In this work, two different real-time applications were tested. The first application is On-Live, which is a cloud gaming platform. The second application is an open-source chess game. After experimentation we found that offloading energy can satisfy long-time energy needs and decrease the load on mobile devices. This can be done by sharing the load of mobile devices through the cloud, which also provides data efficiently, especially for video streaming. It is also observed that a longer delay can affect the performance of devices, and energy offloading sometimes experiences packet loss, which can have a negative impact on the communication networks.

Zanni et al. (2017) introduced an offloading tool that is capable of scanning any mobile application that automates runtime offloading of a program to a remote server based on the predicted energy usage and computing time. This tool is based on automated analysis to determine which methods to execute on a remote server. After small changes, these methods run remotely on a server in a transparent way. Algorithms to select the methods that can run on both mobile devices and servers include: (1) APK parser; (2) classes and methods analysis; (3) methods sorting.

Tao et al. (2017) proposed an energy optimization problem for mobile edge cloud computing. Edge computing has advantages over cloud computing such as controllable latency and low energy consumption. Offloading data to a remote cloud can create high latency, which affects the performance of applications. According to the research on remote cloud computation, it is better to offload computation tasks to a nearby base station cloud. For every task, mobile has an expected consumption of energy. Finally, they

formulated the energy consumption minimization problem with constraints on resource capacity and delay. In particular, whether to offload a computation task or not is determined by mobile energy conditions and application requirements. Zhou et al. (2015) proposed a context-aware offloading decision algorithm. This algorithm considers multiple context changes, such as available network conditions, smart device information, and availability of multiple types of cloud resources (mobile device cloud, cloudlet, and public clouds). This research work also provides a general cost estimation model for cloud resources, which estimates the application execution cost including execution time and energy consumption. In addition, Gajbhe and Sakhare (2015) proposed that it is not always viable to offload every task to the cloud. They proposed a model to calculate the offloading cost and to decide whether to offload an application to the cloud based on that calculation. The proposed model consists of five key components, i.e., profiler, decision manager, offloading manager, local execution manager, and remote execution manager. The decision to offload is based on core real-time metrics of mobile devices: lasting battery level, complexity of the task, type of the processor, file size, and available main memory. In the proposed architecture, two types of costs were considered: actual cost and cost threshold. If the actual cost is greater than the threshold cost, then the task will be offloaded to the cloud; otherwise, the task is completed on the mobile.

Hasan et al. (2015) proposed a framework called Aura, which lets users create an ad hoc cloud using Internet of Things (IoT) and other computing devices that are available in the nearby physical environment. This framework is useful when the cloud server is not accessible, possibly due to a poor Internet connection. Users can start, stop, and restart computations as needed. The framework supports the migration of running computations between active nodes whenever the mobile devices move to a new physical location. For scheduling, communication, and synchronization between clients and the ad hoc network, the Aura framework has multiple controller nodes (Liu et al., 2013).

Mukherjee and De (2014) focused on offloading and importantly on data security issues during communication between a mobile client and the cloud. They considered computation time and task deadline as decision benchmarks for offloading. In their proposed architecture, femto cloud architecture and retinal image recognition based biometric authentication are used for offloading and secure access to the cloud, respectively (Mukherjee and De, 2014).

Elgendy et al. (2014) proposed a framework called mobile capabilities augmentation using cloud computing (MCACC). The framework divides any mobile application into a group of services to execute either on the mobile device or in the cloud. The execution decision is made based on five real-time metrics: total execution time, remaining battery, energy consumption, memory, and security. MCACC does not interrupt the user before taking any offloading decision, which makes the smartphones smart in reality. It consists of four core components: (1) decision manager, (2) offloading manager, (3) profiler, and (4) cloud manager. The first three components are deployed on the mobile device and the cloud manager component is deployed in the cloud.

Similarly, Jadad et al. (2016) proposed a realistic decision algorithm (RDA) using realistic data from the user environment to decide the offloading at runtime. The design of the MCC architecture is presented for both mobile and cloud cost-prediction models. The mobile side has components of realistic profiling, cost prediction model, and decision engine, which work in sequence. Similarly, on the cloud side, the realistic profiler gathers data from available resources, and the execution cost is predicted to determine whether to complete its processing on the mobile device. The proposed algorithm also checks the high-speed Internet and cloud server connectivity. On availability of both, predicted execution costs for both mobile and cloud are compared, and an optimal execution decision (local/offload) is made based on the minimum cost.

The offloading-related work discussed above reveals a research gap in augmentation of smart devices using computation offloading, which needs to be filled. This motivates us to propose an efficient solution to augment the resources of any smart device.

3 Proposed MEACC model

In this section, we present the MEACC architecture for augmenting the energy of smart devices

having limited resources. As mentioned in previous sections, the battery power does not increase as processing power and storage. The execution of resource-intensive applications on smart phones has resulted in excessive depletion of battery power. One main reason for such rapid reduction of battery power is the continuous running of some of the Android applications in the background. When any other application that requires heavy computation is loaded in the smart device's memory for execution, resources that are already being used are overloaded and indirectly use more battery power. The main objective of the proposed architecture is the best filtration of such applications, which may cause reduced battery lifetime of smart devices. In the subsequent sections, features of the proposed MEACC architecture and mathematical prediction models are presented.

The MEACC architecture is a decision-making function for local/remote execution of all applications that the mobile user wants to run. Fig. 2 shows the concise structure of MEACC.

The MEACC architecture comprises two main execution environments, one on the mobile side and the other on the cloud side.

3.1 Mobile-side execution

Mobile-side components are the front end of application execution and use the resources of the smart device or the cloud in the background. This algorithm resides at the mobile side to make a decision on offloading. The following are the core components of MEACC.

3.1.1 Profiler information

Profiler information is responsible for providing the complete details of the available resources on the mobile phone, including the processor state, available battery power, and memory. Moreover, in this component the resources required by the application for its complete execution are calculated. It initiates the execution of the algorithm to monitor the resources of the smart device and calculate the execution cost to run an application locally. The execution cost is calculated using

$$ET(file)=FileSize/mCPU(1-CPULoad), \quad (1)$$

$$BT(file)=BC(file)/ET(file), \quad (2)$$

$$TEC(local)=ET(file)+BT(file), \quad (3)$$

where ET is the execution time of the file/application to be executed, mCPU is the actual processing speed of the smart device, and CPULoad is the busy state of the mobile device for other applications and services running on the same device. BT is the battery time required by the application/file to be processed, and BC is battery consumption per file execution. TEC is the total execution cost for any application to be run using the smart device's resources.

3.1.2 Decision manager

This is the component that uses the profiler log information to make a one-to-one comparison of the results of the available and required resources. If the mobile phone has enough resources as per the demand

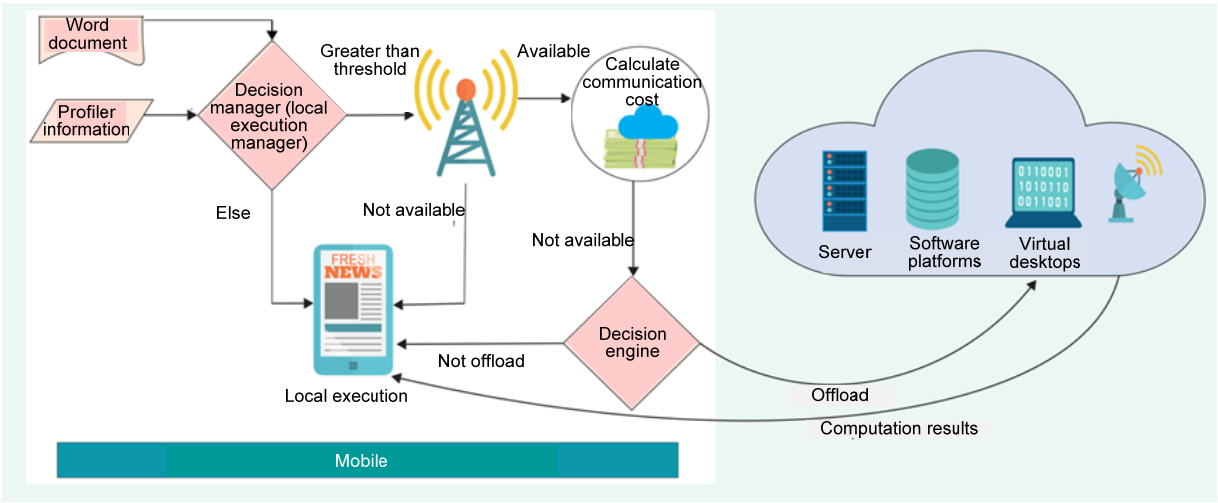

Fig. 2 Architecture of mobile energy augmentation using cloud computing (MEACC)

of the application, then the execution will be performed locally. Otherwise, the next component, i.e., the Wi-Fi availability checkpoint, is called to perform application execution.

3.1.3 Wi-Fi availability checkpoint

As the name suggests, this component checks if the smart device is connected to the Internet. If Internet access is available, then the next component, i.e., the Communication and cloud-side execution cost calculator, is invoked; otherwise, the application will be directed for local execution.

3.1.4 Communication and cloud-side execution cost calculator

This component calculates the communication cost to offload the application to the cloud for execution. Communication cost is an accumulation of the total time required to upload the application and return the results from the cloud to the mobile device. The communication cost and total execution cost on the cloud side are calculated using

$$CC(file)=FileSize/Speed\_of\_Internet, \quad (4)$$

$$TEC(cloud)=ET(file)+CC(file), \quad (5)$$

where TEC is the total execution cost on the cloud side, ET is the execution time calculated using Eq. (1), and CC is the communication cost for uploading the application to and downloading the application from the cloud.

3.1.5 Decision engine

The final decision concerning application offloading is made in this component. This component decides whether to offload after comparing the execution costs for the mobile and cloud sides. If the total execution cost on the cloud side is greater, the application will be directed for local execution on the mobile device.

3.2 Cloud-side execution

Cloud-side execution is completely hidden from the mobile user. If any application meets all the criteria to be offloaded, then all its computations will be performed using resources that are available in the cloud. After complete execution, the results will be returned to the mobile side and displayed to the end user. Algorithm 1 is the baseline for a final decision to offload any task or not. In this algorithm, another Android function is invoked that runs as a background service on the mobile side and provides the profiler information. This profiler log information is used to calculate the execution cost of the specified task on the mobile device. After checking Wi-Fi connection and calculating the communication cost for the inputted file, the final selection of the platform, either the mobile or the cloud, is made for the execution of the desired file.

Algorithm 1 Offloading decision

Input: TextFile
Output: LocalExecutionCost, CommunicationCost, ExecutionDecision(Local/Cloud)

```
1  ProcedureTakeDecision(Textfile)
2  FS←TextFileSize
3  Threshold←GetDeviceInfo
4  RR←EstimateRequiredResource
5  if Threshold[index]<RR[] then
6     if (Wi-Fi==true; CC+Const<LEC) then
7        return offload
8     else
9        return NotOffload
10    end if
11 else
12    return NotOffload
13 end if
```

The offloading decision algorithm takes a text file as input and decides whether to offload execution to the cloud based on two parameters: (1) local execution cost and (2) communication cost. First, the size of the input file is determined using the file computation algorithm described in Algorithm 2. The algorithm then obtains device resource status information (i.e., battery and CPU consumption, which are currently available in the device), called the threshold. The resources required for processing the input file, called required resources (RR), are estimated. The next step is to compare the threshold and the required resources. If the threshold resources (available resources) exceed the required resources, then the algorithm would simply go with the decision of local execution; otherwise, we need to offload the execution. Before making the final decision, the algorithm checks the availability of Wi-Fi and if the computation cost is less than the local execution cost. Then

it returns the decision concerning offloading file execution.

Algorithm 2 is the algorithm to complete the profiler log information. The algorithm reads the inputted file, word by word, and determines the total number of words in the file. After the complete file read, the execution time to complete this task is displayed on the dashboard.

Algorithm 2 File computation

```
Input: TextFile
Output: TotalWords, TotalCount, ExecutionTime
ProcedureTakeDecision(Textfile)
LF←Loadtxtfile
while (LF.readLine()=Null) do
   Text←readLine
   Array[]←Text.Split()
   Array.sort(Array[index])
   for (i=0; i<ArrayLenght; i=i+1) do
      for (j=i+1; j<ArrayLength; j=j+1) do
         if word=true then
            count++
            Words∪word
            TotalCount∪count
         else
            break
         end if
      end for
   end for
end while
return TotalWords, TotalCount, ExecutionTime
```

The file computation algorithm is basically used to compute the size of the input text file for the offloading decision algorithm. It takes the text file as the input and returns the total number of words in the file and the time required for execution of the file. The algorithm reads the file line by line and splits the text lines into words. Using arrays of words, it sorts the words alphabetically, prints an array, and finally counts the words with the help of a loop on computing array length.

4 Experiments and results

In this section we explain the selected experimental platform for the proposed MEACC architecture. In the experiments, we calculated the execution cost for different file sizes on the mobile or cloud side, and the communication cost was also considered for the total execution cost in the cloud. The details of the results are presented.

4.1 Experimental setup

To evaluate the proposed architecture, all experiments were performed in two different environments, one using an Android smart phone and the other a cloud simulator.

In the smartphone environmental setup, all the experiments were performed using the following three smart phones:

1. Samsung Galaxy Grand Prime SM-G350H specifications: CPU 1.2-GHz QUADCORE Qualcomm MSM8916 Snapdragon 410 Chipset and with 1 GB RAM.

2. Google Pixel 2 specifications: CPU Octa-core (4×2.35-GHz Kryo and 4×1.9-GHz Kryo) Qualcomm MSM8998 Snapdragon 835 Chipset with 4 GB RAM.

3. Google Pixel 3 specifications: CPU Octa-core (4×2.5-GHz Kryo 385 Gold and 4×1.6-GHz Kryo 385 Silver) Qualcomm SDM845 Snapdragon 845 Chipset with 4 GB RAM.

Multiple cloud simulators are available to obtain the experimental results. A few such simulators are CloudSim (Calheiros et al., 2011), GREEN CLOUD, and GDC CLOUD (Sinha and Shekhar, 2015). For this research work, we chose the Hadoop version 1.3.0 environment running on Linux Ubuntu 12.5 as the cloud. Hadoop provides the same task execution infrastructure as the real cloud infrastructure to execute the tasks.

4.2 Experimental data

The text document files with the rate of 600 KB were used in the range of 100 to 4800 KB. Because smaller size executions do affect precision and accuracy, the initial executions differed by 300 KB. All the data files were executed on mobile and cloud to calculate the execution cost of each file on these two environments. On the mobile side, two types of resource availability were considered: one is to check how many resources (processor, battery, RAM, etc.) will be consumed for complete execution of each sample data file, and the other is to check the free resources to be assigned to the sample data file for completing the execution. Based on the calculation results, MEACC will decide whether to offload the file computation or not.

4.3 Experimental results

The proposed algorithm was evaluated by executing all the chosen dataset files once on the selected smart phones and once on the cloud simulator. To implement the proposed algorithm for the offloading decision, we developed two Android apps named WordCounter and AnotherMonitor. The WordCounter app was developed to calculate the total execution time that a CPU would take to perform a total word count of the inputted text file. The AnotherMonitor app was used to obtain the mobile profiler information (mCPU usage, memory usage, and battery status). This app calculated the CPU load (mCPU load, memory load, and battery consumption) of the WordCounter app when we started the text file execution using WordCounter. The dataset text files were executed to calculate the time taken to complete the execution of each file. The graphical representation in Fig. 3 shows that the execution time is directly proportional to the file size. More time will be consumed to complete the execution as the file size increases (see Figs. S1–S3 in the supplementary materials).

The execution of any application on a smartphone consumes basic resources including memory, CPU, and battery power. During the execution of the dataset files on the smartphone, the resource consumption was determined to analyze the resource utilization for each input file.

As shown in Fig. 4, the rate of resource utilization varies from file to file. The impact/burden on the mobile CPU is less for small file sizes and more for large files. According to Fig. 4, if a 1200 KB file is executed on smartphone device 1 for 29 s with 64.7% usage of CPU, 19 604 KB memory, and the usage of CPU for this file size is 47.8%, then this 1200 KB file directly consumed battery for 29 s with mentioned resources and used reasonable battery power.

If the same file is executed on device 2 and then on device 3 with usage of 58.7% and 55.6% of CPU, 22 016 KB and 20 101 KB memory, and usage of CPU for file size 52.6% and 50.1%, respectively, then this 1200 KB file directly consumed battery for 16 s on device 2 and 14 s on device 3 with the mentioned resources and used reasonable battery power. Similarly, the rest of the input files consumed smartphone resources. An execution setup is presented in Figs. S4–S6 in the supplementary materials.

To obtain the results on the cloud side, the same dataset files were executed on the cloud simulator to calculate the time taken by each file to complete its execution remotely (see Table S1 in the supplementary materials). When offloading any file to the cloud, communication cost is the core issue to be considered for calculating the total execution cost on the cloud side. This cost must be afforded in terms of the time required to upload or download any data over the cloud. The transmission rate is measured as the number of bits per second. It also depends on the speed of the Internet connection being used. In other words, communication cost is the time required for transmission of any data to the cloud (uploading cost) and the time required to transmit the execution results from the cloud to the mobile device (downloading cost). For this research work, a Wi-Fi Internet connection of 9.3 MB bandwidth was used with 19.15 Mb/s downloading and 2.25 Mb/s uploading rates to calculate the communication cost for any of the dataset files. The communication cost was calculated using Eq. (4).

To calculate the total execution cost required for the complete execution of any dataset file, the computation cost is added to the communication cost. Fig. 5 shows the total execution cost to offload files of different sizes.

Further, we quantified the total execution cost with respect to the mobile and the cloud. Fig. 6 shows the execution cost for the dataset files to be computed in the cloud and on mobile devices.

We noticed a big difference between the mobile- and cloud-side execution costs. When executing a small file, fewer resources were consumed on the mobile device as compared to execution of a similar file in the cloud. The results showed that the rate of resource consumption is directly proportional to the file size because the quantified execution time over three different smart mobiles was 105, 34, and 32 s, respectively, for a file of 3000 KB. In contrast, the same file size was executed in 28.37 s in the cloud, where the observed communication cost and execution cost were 13.37 and 15 s, respectively. The proposed model minimized the execution overhead by 76.63, 5.63, and 3.63 s for each of the mentioned smart phones using the offloading mechanism. This overhead reduction improves the battery life in mobile devices.

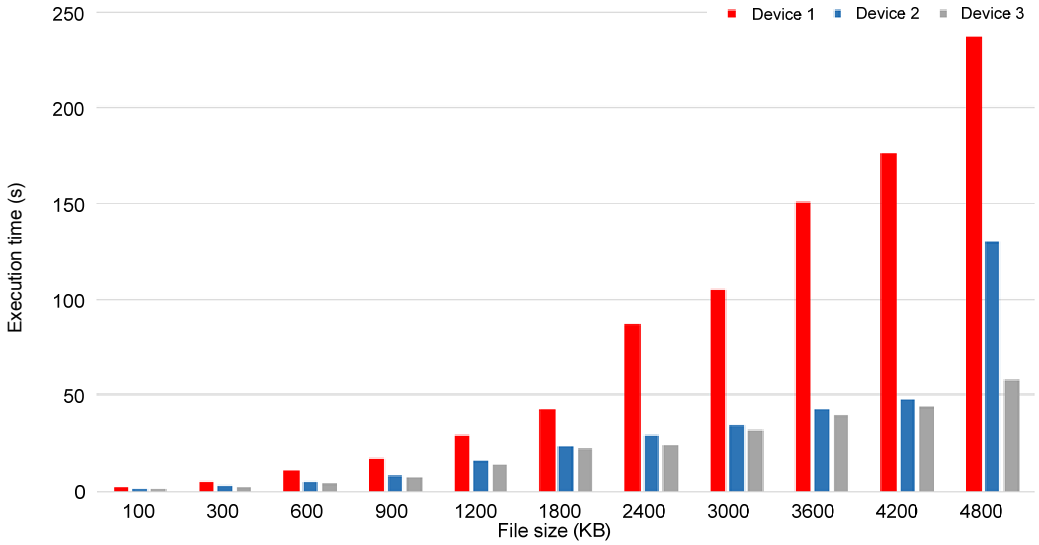

Fig. 3 Mobile-side execution time of devices 1, 2, and 3

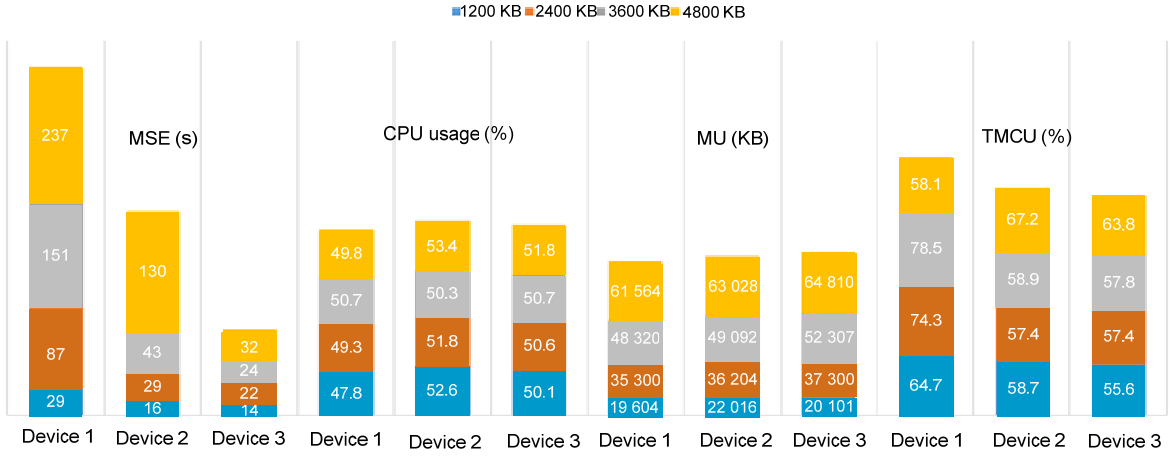

Fig. 4 Resource consumption on mobile devices
MSE: mobile-side execution time; MU: memory usage; TMCU: total mobile CPU usage

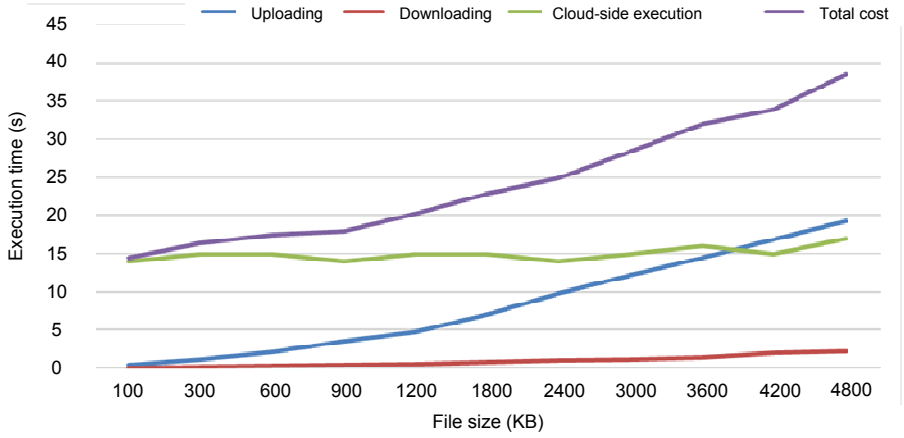

Fig. 5 Cloud-side total cost (communication cost+cloud-side execution cost)

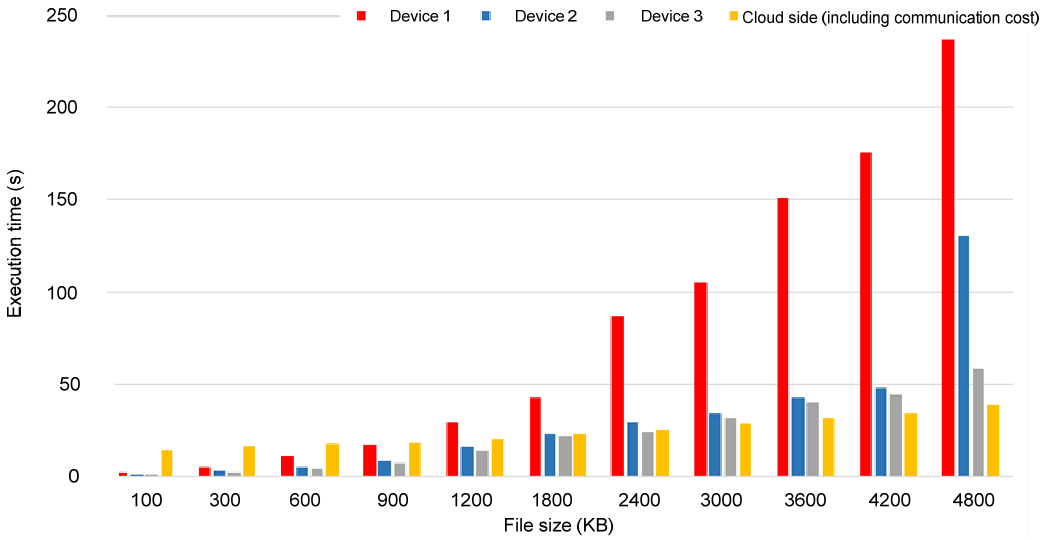

Fig. 6 Comparison of execution time

5 Discussion

To balance the resource utilization and reduce the power consumption in mobile devices, MEACC is proposed and implemented on different mobile platforms. To evaluate the objective metrics, different executions were performed for file sizes 100−4800 KB. This benchmark was implemented on three different mobile platforms including device 1 (Galaxy Grand Prime SM-G350H, D1), device 2 (Google Pixel 2, D2), and device 3 (Google Pixel 3, D3). During the implementations, we observed different attributes including mobile-side execution, cloud-side execution, communication cost while offloading data, and resource utilization. The primary objective was to offload the data for execution in the cloud when the execution cost on the mobile side is higher compared to cloud execution. In MEACC, when a document is provided for execution, the profiler information log provides all the information for that document to the decision manager, which determines different additional attributes if the data needs to be executed in the cloud. Based on these circumstances, we conducted a comparative study comparing the proposed MEACC model, the Monica model (Gajbhe and Sakhare, 2015), and the MCACC model (Elgendy et al., 2014). According to Fig. 7, it was observed that overall, MEACC outperformed the other models. For execution of smaller files, the Monica model took a bit less time, but the execution time increased gradually as the file size increased. For the file size of 2400 KB, the MEACC model completed the execution in about 85 s, and the Monica and MCACC models completed the execution in 102 and 138 s, respectively. We continued to increase the file size and found similar changes in the results.

Further, the same benchmark was observed on other platforms with similar specifications. This time, we noticed a drastic change in the execution time for the proposed MEACC model, which outperformed other models when the dataset size was increased up to 3000 KB (Fig. 8).

The reason for this change at large dataset sizes was due to correct selection of resources by the decision engine. To evaluate the behavior of the MEACC model on devices with advanced specifications, we again performed the experiments on a third mobile device, Google Pixel 3, with these specifications: CPU Octa-core (4×2.5-GHz Kryo 385 Gold and 4×1.6-GHz Kryo 385 Silver), Qualcomm SDM845 Snapdragon 845 Chipset with 4 GB RAM. We found better performance of the MEACC model with increased computation resources (Fig. 9). This time, MEACC took a maximum of 20 s for the larger file size 3000 KB. In contrast, the execution time was twice and three times in the Monica and MCACC models, respectively.

Further, we quantified the execution time in the cloud by computing a similar benchmark dataset. Fig. 10 demonstrates that the decision engine balanced the resources and offloaded data to the cloud for computation, where MEACC outperformed the other models throughout the execution. When

computing a large dataset with 3000 KB file size, the execution times of MEACC, Monica, and MCACC are 15, 27, and 40 s, respectively.

To evaluate the resource utilization in the cloud, we excluded the communication cost in the outcomes shown in Fig. 10. Because the communication overhead is a major challenge, when offloading the data from the local device to the cloud side, we include the communication cost along with cloud-side resource utilization (Fig. 11). Although the communication cost reduces the overall system performance, an efficient offloading strategy can overcome this overhead. Fig. 11 shows that the MEACC offloading strategy always outperformed Monica and MCACC and processed data efficiently in 40%−60% time.

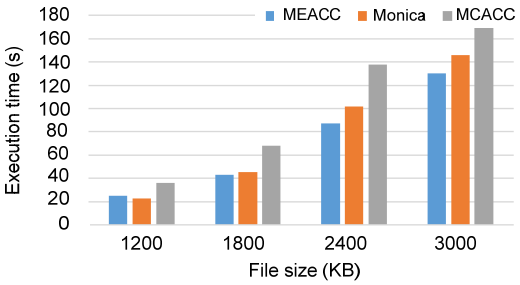

Fig. 7 MEACC vs. Monica and MCACC execution on device 1

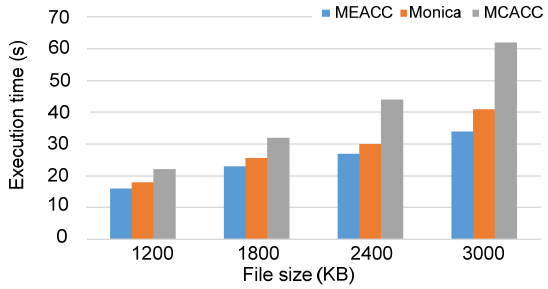

Fig. 8 MEACC vs. Monica and MCACC execution on device 2

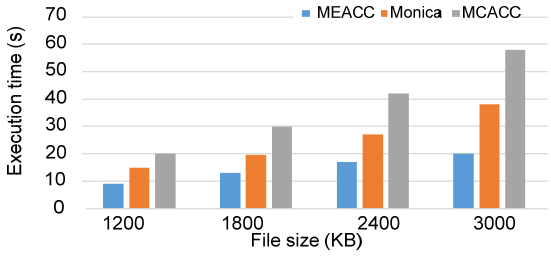

Fig. 9 MEACC vs. Monica and MCACC execution on device 3

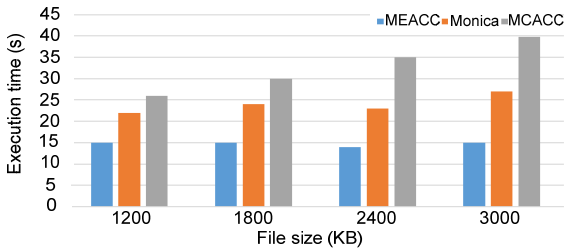

Fig. 10 MEACC vs. Monica and MCACC execution in the cloud without communication cost

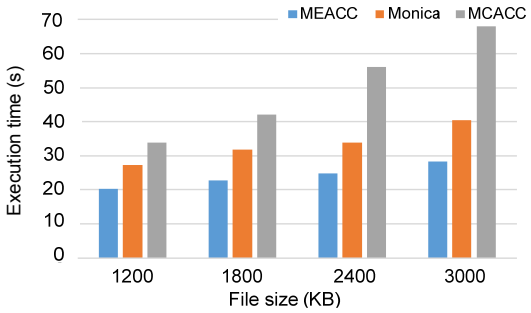

Fig. 11 MEACC vs. Monica and MCACC execution in the cloud with communication cost

6 Conclusions

In this research work, a decision-making approach called mobile energy augmentation using cloud computing (MEACC) is proposed to augment the battery life of smart mobile phones by making the offloading decision for resource-intensive tasks. On the cloud side, communication costs were calculated based on the Internet speed. The MEACC approach improved the offloading decision-making process by implementing an impartially precise execution cost prediction method that was evaluated by the Android application AnotherMonitor. The quantified execution cost was compared to the communication cost of offloading by evaluating the efficiency of MEACC. MEACC outperformed the Monica and MCACC models. The MEACC approach includes communication cost calculation to complete its processing, which makes it more scalable for multiple Internet service providers.

Because the current research deals only with textual data, in the future, our plan is to implement the proposed approach for any type of application to be processed on smart devices. We will also consider the MEACC model for large-scale emerging edge/fog

computing systems to increase the performance by reducing the communication cost.

Contributors

Babur HAYAT guided the research on offloading mechanism using cloud computing. Zuhaib IMTIAZ and Ayesha RAANA designed the research and drafted the manuscript. Khalid ALSUBHI helped organize the manuscript. M. Usman ASHRAF helped design the data collection mechanism and edited and finalized the paper.

Compliance with ethics guidelines

Khalid ALSUBHI, Zuhaib IMTIAZ, Ayesha RAANA, M. Usman ASHRAF, and Babur HAYAT declare that they have no conflict of interest.

References

Calheiros RN, Ranjan R, Beloglazov A, et al., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exp*, 41(1):23-50. https://doi.org/10.1002/spe.995

Cao HJ, Cai J, 2017. Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: a game-theoretic machine learning approach. *IEEE Trans Veh Technol*, 67(1):752-764. https://doi.org/10.1109/TVT.2017.2740724

Chun BG, Ihm S, Maniatis P, et al., 2011. CloneCloud: elastic execution between mobile device and cloud. Proc 6th Conf on Computer Systems, p.301-314. https://doi.org/10.1145/1966445.1966473

Creeger M, 2009. CTO roundtable: cloud computing. Queue, 7(5):1. https://doi.org/10.1145/1551644.1551646

Cuervo E, Balasubramanian A, Cho DK, et al., 2010. MAUI: making smartphones last longer with code offload. Proc 8th Int Conf on Mobile Systems, Applications, and Services, p.49-62. https://doi.org/10.1145/1814433.1814441

Dinh HT, Lee C, Niyato D, et al., 2013. A survey of mobile cloud computing: architecture, applications, and approaches. *Wirel Commun Mob Comput*, 13(18):1587-1611. https://doi.org/10.1002/wcm.1203

Elgendy MA, Shawish A, Moussa MI, 2014. MCACC: new approach for augmenting the computing capabilities of mobile devices with cloud computing. Science and Information Conf, p.79-86. https://doi.org/10.1109/SAI.2014.6918175

Gajbhe M, Sakhare SR, 2015. Performance augmentation of mobile devices using cloud computing. *Int J Comput Sci Inform Technol*, 6(4):3581-3587.

Hasan R, Hossain MM, Khan R, 2015. Aura: an IoT based cloud infrastructure for localized mobile computation outsourcing. Proc 3rd IEEE Int Conf on Mobile Cloud Computing, Services, and Engineering, p.183-188. https://doi.org/10.1109/MobileCloud.2015.37

Jadad H, Touzene A, Alzeidi N, et al., 2016. Realistic offloading scheme for mobile cloud computing. Proc 13th Int Conf on Mobile Web and Intelligent Information Systems, p.81-92. https://doi.org/10.1007/978-3-319-44215-0_7

Kosta S, Aucinas A, Hui P, et al., 2012. Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. Proc IEEE INFOCOM, p.945-953. https://doi.org/10.1109/INFCOM.2012.6195845

Kovachev D, Klamma R, 2012. Framework for computation offloading in mobile cloud computing. *Int Interact Multim Artif Intell*, 1(7):6-15.

Liu FM, Shu P, Jin H, et al., 2013. Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. *IEEE Wirel Commun*, 20(3):14-22. https://doi.org/10.1109/MWC.2013.6549279

Mell P, Grance T, 2011. The NIST Definition of Cloud Computing. NIST Special Publication 800-145, NIST, Gaithersburg.

Mukherjee A, De D, 2014. A cost-effective location tracking strategy for femtocell based mobile network. Proc Int Conf on Control, Instrumentation, Energy and Communication, p.533-537. https://doi.org/10.1109/CIEC.2014.6959146

Paranjothi A, Khan MS, Nijim M, 2017. Survey on three components of mobile cloud computing: offloading, distribution and privacy. *J Comput Commun*, 5(6):75210. https://doi.org/10.4236/jcc.2017.56001

Robinson S, 2009. Cellphone Energy Gap: Desperately Seeking Solutions. Strategy Analytics. Technology Report, Chicago, IL, USA.

Satyanarayanan M, Bahl V, Caceres R, et al., 2009. The case for VM-based cloudlets in mobile computing. *IEEE Perv Comput*, 8(4):14-23. https://doi.org/10.1109/MPRV.2009.82

Sinha U, Shekhar M, 2015. Comparison of various cloud simulation tools available in cloud computing. *Int J Adv Res Comput Commun Eng*, 4(3):171-176. https://doi.org/10.17148/IJARCCE.2015.4342

Sookhak M, Yu FR, Khan MK, et al., 2017. Attribute-based data access control in mobile cloud computing: taxonomy and open issues. *Fut Gener Comput Syst*, 72:273-287. https://doi.org/10.1016/j.future.2016.08.018

Tao XY, Ota K, Dong MX, et al., 2017. Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wirel Commun Lett*, 6(6):774-777. https://doi.org/10.1109/LWC.2017.2740927

Zanni A, Yu SY, Secci S, et al., 2017. Automated offloading of Android applications for computation/energy optimizations. IEEE Conf on Computer Communications Workshops, p.990-991. https://doi.org/10.1109/INFCOMW.2017.8116525

Zhang L, Fu D, Liu JC, et al., 2017. On energy-efficient offloading in mobile cloud for real-time video applications. *IEEE Trans Circ Syst Video Technol*, 27(1):170-181. https://doi.org/10.1109/TCSVT.2016.2539690

Zhang Q, Cheng L, Boutaba R, 2010. Cloud computing: state-of-the-art and research challenges. *J Int Serv Appl*, 1(1):7-18. https://doi.org/10.1007/s13174-010-0007-6

Zhang XW, Jeong S, Kunjithapatham A, et al., 2010. Towards an elastic application model for augmenting computing capabilities of mobile platforms. Proc 3rd Int Conf on Mobile Wireless Middleware, Operating Systems, and Applications, p.161-174. https://doi.org/10.1007/978-3-642-17758-3_12

Zhou BW, Buyya R, 2018. Augmentation techniques for mobile cloud computing: a taxonomy, survey, and future directions. *ACM Comput Surv*, 51(1):13. https://doi.org/10.1145/3152397

Zhou BW, Dastjerdi AV, Calheiros RN, et al., 2015. A context sensitive offloading scheme for mobile cloud computing service. IEEE 8th Int Conf on Cloud Computing, p.869-876. https://doi.org/10.1109/CLOUD.2015.119

Zissis D, Lekkas D, 2012. Addressing cloud computing security issues. *Fut Gener Comput Syst*, 28(3):583-592. https://doi.org/10.1016/j.future.2010.12.006

List of supplementary materials

Fig. S1 Total execution time on mobile device D1 (Galaxy Grand Prime)
Fig. S2 Total execution time on mobile device D2 (Google Pixel 2)
Fig. S3 Total execution time on mobile device D3 (Google Pixel 3)
Table S1 Total cloud execution costs (Hadoop simulator)
Fig. S4 Resource consumption on mobile device D1 (Galaxy Grand Prime)
Fig. S5 Resource consumption on mobile device D2 (Google Pixel 2)
Fig. S6 Resource consumption on mobile device D3 (Google Pixel 3)