Asian Journal of Computer Science and Technology
ISSN: 2249-0701 Vol.8 No.S3, 2019, pp. 1-6

Robust and Secure Framework for Mobile Cloud Computing

Pallavi Alava[1] and G. Radhika[2]
[1]Student, [2]Assistant Professor,
[1&2]Department of Computer Science and Engineering, School of Engineering and Technology,
Sri Padmavati Mahila Visva Vidyalayam, Tirupati, Andhra Pradesh, India
E-Mail: alava.pallavi@gmail.com, radi323@gmail.com

Abstract **- Smartphone devices are widely utilized in our daily lives. However, these devices exhibit limitations, like short battery lifetime, limited computation power, small memory size and unpredictable network connectivity. Therefore, various solutions have been projected to mitigate these limitations and extend the battery period of time with the employment of the offloading technique. In this paper, a unique framework is projected to offload intensive computation tasks from the mobile device to the cloud. This framework uses Associate in Nursing improvement model to work out the offloading decision dynamically supported four main parameters, namely, energy consumption, CPU utilization, execution time, and memory usage. Additionally, a new security layer is provided to shield the transferred data within the cloud from any attack. The experimental results showed that the framework will choose an acceptable offloading decision for various forms of mobile application tasks whereas achieving important performance improvement. Moreover, different from previous techniques, the framework will defend application knowledge from any threat.**
Keywords: **Mobile Cloud Computing, Smartphone Devices, Security, Computation Offloading**

I. INTRODUCTION

Smartphones give a broad vary of applications, such as face detection, increased reality, image and video processing, and video gaming and speech recognition. These applications are complicated, and therefore the demand for computing resources is increasing. However, despite the advancements in smartphones, the extent of battery life has remained joined of the most challenges in rising process requirements through battery upgrade[1]. Cloud computing permits access to unlimited resource over the net[2]. Cloud computing provides many advantages, like self-service provisioning, elasticity, broad network access, resource pooling, low costs, and ease of utilization, among others. Thus, mobile cloud computing is introduced to overcome the limitations of smartphone devices. Mobile cloud computing may be a new paradigm that integrates cloud computing technology and mobile devices to increase the battery lifetime and increase application performance. Recent studies have projected to offload all or a part of the mobile applications from mobile device to the cloud for remote execution [3]. These frameworks are designed to make a trade-off between one or more constraints, such as energy consumption of the mobile device, CPU utilization, execution time, remaining battery life, and data transmission amount on the network, within the offloading decision [4]. However, most of those models do not consider memory usage as a constraint within the offloading decision. Memory usage is one in all the most resources consumed by mobile applications. Additionally, security techniques don't seem to be applied in the protection of offloaded information from attacks. Therefore, this work in the main focuses on building a brand new model that mixes most of the mentioned constraints to improve the performance of mobile applications and to protect the applying information from any attack. We specifically projected a unique framework that uses computation offloading to offload only the intensive tasks of mobile applications. We developed an improvement model responsible for determinant the offloading decision. The main results and contributions of this paper are as follows

1. This work proposes a unique framework that offloads only intensive tasks rather than offloading all applications, thereby requiring less network communication.
2. AN improvement model is developed to see the offloading call dynamically at runtime supported four main constraints, namely, execution time of the task, C.P.U. utilization, memory usage and energy consumption.
3. A new security layer is other to code the infoof the task before transferring to the cloud aspect by using AES encryption technique.
4. Three differing kinds of mobile applications are used in the experimental studies to check this framework and to show the selection of a correct offloading decision for improved application performance. The rest of the paper is organized as follows. Section a pair of discusses the state-of-the-art procedure offloading frameworks and their drawbacks...

II. RELATED WORK

Numerous approaches are recently proposed to handle the challenges of mobile devices by offloading the computation tasks to the cloud resources for remote execution. A number of these approaches migrate only a method from the mobile device to the cloned virtual machine(VM) on the cloud. A combination of static analysis and dynamic identification modules is used to partition the application and confirm that method is migrated to the cloud. An entire smartphone system on the cloud and used a profiler module to observe

the remote execution of the strategies using an execution controller. The most downside of and is that the energy-consuming demand of basic synchronization with the clone VM on the cloud [5] what is more, application knowledge are not protected against attacks throughout transfer to the cloud. The [6] synchronization drawback is handled by offloading only the intensive services and not the complete method to the cloud. Additionally, the authors build a model to work out the offloading decision for these services. However, this model is very simple and static and forever prefers remote execution. In certain cases, a death penalty services on the mobile is superior to offloading to the cloud. The transferred data should be protected by applying any security technique. Other frameworks involving the partition of the applying and the offload of intensive ways area unit projected [7].

Theseframeworks additionally use a whole number linear programming model like our framework in creating offloading choices. Total latency, remaining battery life, and energy consumption constraints area unit thought aboutin creating the offloading decision while not adding any memory usage thought and security to the offloading model. In contrast, in the total android application is offloaded from the mobile device to the cloud, which is resource intense as a result of the massive quantity of transferred knowledge over the network. Additionally, the application sent to the cloud should be safe, thus any security technique should be protected.

The minimization of the data transmission and therefore the energy consumption are the most goals of, that offloads only the resource-intensive services and exploits from Software-as-a-Service model for the configuration of intensive services on the cloud server. The same as desires basic synchronization between the mobile device and therefore the cloud server node that consumes extra battery power and makes the offloaded knowledge liable to attacks. However, this framework used a discovery service to get the hardware info of the cloud resources each minute, thereby intense extra energy. Additionally, the transferred knowledge weren't protected from attacks. This formula comprised three main parts, namely, computation offloading choice, CPU clock frequency control in local computing, [8] and transmission power allocation in cloud computing.

III. FRAMEWORK ARCHITECTURE AND DESIGN

In this section, we justify the design of the framework and show however its modules will communicate to attain the look goals of the system. Additionally, the linear optimization model is outlined. A close determination of the offloading decision is additionally showed. Then, we offer associate algorithmic program that clarifies however this framework works. Finally, we establish the specified steps to feature this framework throughout development.

A. Framework Architecture: The framework design consists of six modules, namely, estimator, profile, network and bandwidth monitor, decision maker, mobile manager, and cloud manager.

First, the framework works at the method level, wherever the developers have to be compelled to add AN annotation (@Remote) specifically intensive strategy at the developing step. These strategies should need further computation and might be offloaded to the cloud for remote execution. These strategies should not a) depend upon the user interface or b) use any I/O mobile device like GPS, camera, or measuring device.

1. Estimator

The estimator module is responsible for identifying these strategies for native execution on the mobile device and remote execution on the cloud with totally different input sizes (stored as a sample) at the installation step. Then, the module obtains the values of execution time, memory usage, CPU utilization, and energy consumption for every annotated technique for these totally different input sizes (minimal eye application is employed to live the energy consumption and electronic equipment utilization). Finally, the values are communicated and sent to the profile module.

2. Profile

The profile module obtains the values of execution time, memory usage, CPU utilization, and energy consumption from figurer module for every annotated technique. Then, the module creates a brand new file for every technique and stores these values into the file. These files are updated when every running method and utilized by the decision maker module as a history-based go into the offloading decision.

3. Network and Bandwidth Monitor

This module only monitors this standing of the network and gathers cell connection state and its information measure, Wi-Fi association state and its information measure, and signal strength of cell and Wi-Fi connection (get this data using programming code). Then, this data is shipped to {the call|the choice} maker module to support the determination the offloading decision.

4. Decision Maker

The choice build, that is, the core module of the planned framework, contains associate whole number linear programming model and decision-making rule that predicts at runtime wherever the annotated strategies are executed. The goal of the model is to search out associate application partitioning strategy that minimizes the energy consumption, transfer knowledge, memory usage, and electronic equipment utilization, in smartphones, subject to bound constraints. Let us assume that we've n variety of annotated methods that will be offloaded to the cloud for remote execution, that is, M1,M2,…..,Mn. every technique I

consists of a collection of parameters, namely, input size (input I), memory usage (memo I), electronic equipment utilization (CPU I), and battery consumption (power I), for native execution. Different parameters, like memory used for security (memo_sec I), battery consumption used for security (power_sec I), and electronic equipment used for security (CPU_sec I) also are thought of if the method is offloaded for remote execution. During this model, xi is introduced for every technique I, that indicates whether or not the method is executed regionally on the mobile device (xi=0) or offloaded for remote execution (xi=1).

The objective function is represented as follows:
$\min x\in 0,1(ctransfer*wtr+cmemory*wmem+cCPU*wCPU+cpower*wpower)$
Where:$Ctransfer=\Sigma inputi*xini=1$
$Cmemory=\Sigma memoi*(1-xi)+ni=1\Sigma memo\_seci*xini=1$
(1)
$CCPU=\Sigma CPUi*(1-xi)+ni=1\Sigma CPU\_Seci*xini=1$
$Cpower=\Sigma poweri*(1-xi)+ni=1\Sigma power\_seci*xi$

C transfer, Cmemory, CCPU, and Cpower represent price for transferring the input size, memory used, CPU used, and power consumed for technique I severally. Wtr, wmemo, wCPU, and wpower are the weights for every these costs, that result in completely different objectives.

According to the target perform in the 3 constraints that have to be handled with care are as follows: Minimize the memory utilized by the appliance ways on the mobile device. Memory price is split into 2 elements. The primary half is that the memory used once the strategy of the appliance is executed locally on the mobile device, whereas the second half is employed to write in code the information before transferring to the cloud within the offloading case. This memory cost should not exceed the on the market memory on the mobile device. The strategy is restricted to a threshold worth that is set supported the fraction between range of running applications and therefore the total size of memory. This constraint may be written as follows:

$$\sum_{i=1}^{n} memo_i*(1-x_i)+\sum_{i=1}^{n} memo\_sec_i*x_i\le M_{th} \quad (2)$$

Where M_{th} is that the memory threshold.

a) Minimize the Overall Execution Time: That is, the second constraint, for the applying. The overall time for corporal punishment the applying strategies remotely on the cloud should be but the overall time for corporal punishment the strategies of the application regionally on the mobile device. Let SM and SC be the processor speeds (instruction per second) of the mobile and also the cloud, severally, and C be the amount of instructions concerned within the methodology invocation. Csec represents the amount of instruction to cipher the information before transfer to the cloud. If the number of information needed for the tactic execution is D and also the network information measure is B, then the time needed to transfer this knowledge is D/B. Therefore, the overall time for corporal punishment strategies on the cloud is split into the subsequent 3 parts: the time consumed by encrypting the information, knowledge transmission, and also the time execution on the cloud. This constraint may be drawn as follows:

$$Exe\_time\_local>Exe\_time\_cloud$$

Where Exe_time_local is total time for the native execution of the method, as calculated as follows:

$$Exe\_time\_local=\frac{C}{S_M}*x_i \quad (4)$$

Exe_time_cloud is total time for offloading the tactic for remote execution, as calculated as follows:

$$Exe\_time\_cloud=\left(\frac{C_{sec}}{S_M}+\frac{C}{S_C}+\frac{D}{B}+t_{overhead}\right)*x_i \quad (5)$$

Where to overhead is the overhead time of our framework.

b) Minimize The Whole Energy Consumption: The last constraint for the objective perform. This constraint deals with the energy consumed by execution the appliance methodology. This constraint is often diagrammatic as follows:

$$Energy\_con\_local>Energy\_con\_cloud \quad (6)$$

If the mobile consumes PM Watts (W) for methodology computation regionally, Pd W for being idle, Psec W for encrypting the data of the method, and Pr W for transferring to the cloud, then the whole energy consumed regionally on the mobile de- vice are often calculated as follows:

$$Energy\_con\_local=PM*\frac{C}{S_M}*xi \quad (7)$$

The total energy consumed for remote execution on the cloud is often calculated as follows:

$$Energy\_con\_cloud=(PSec*\frac{C_{sec}}{S_M})+(Pd*\frac{C}{S_C})+(Pr*\frac{D}{B})*xi$$

Finally, when we tend to solve this formulation, every methodology xi are often determined whether or not for native execution (xi=0) or offloading to the cloud (xi=1). Within the experiments, particle swarm optimization (PSO) java code is employed to implement the linear model. Particle swarm optimization could be a heuristic world optimization methodology associate degreed an

optimization algorithm supported swarm intelligence. PSO is comparable to the Genetic rule (GA) and Emmet Colony optimization (ACO) within the sense wherever they're population-based search ways. however GA and ACO are thought of as ex- pensive machine value compared with PSO and that we need to minimize the general consumption, therefore, PSO is employed as best optimization rule with significantly higher machine potency to resolve our optimization drawback.

5. Mobile Manager

The mobile manager module is responsible for causing a computer file containing the method code and its needed libraries at the installation step. The mobile manager handles the execution of the method supported the model decision. If the tactic is dead regionally on the mobile device, the files areupdated with new values through the profile module. However, if the choice is to offload the method, then the mobile manager encrypts the offloaded data by victimization AES technique and communicates with the cloud manager module to transfer this knowledge with the method name. Finally, the mobile manager receives and delivers the results to the appliance

6. Cloud Manager

The cloud manager module is that the solely module deployed on the cloud side. This module is written strictly in Java. Therefore, any application will take pleasure in the projected framework to offload its computation to any resource that runs the Java Virtual Machine (JVM). Communication between the cloud manager and also the mobile manager modules is managed by wading bird communication middleware within the 1st communication, at the installation step, a computer file containing the tactic code and its required libraries are sent to the cloud. Then, the cloud manager receives the ways data and decrypts them within the following run. Then, the manager executes the tactic remotely and sends the result back to the mobile manager module with the new values to be updated by the profile module.

B. Framework Execution Flow

This section discusses the flow of execution of the proposed framework. Formula one illustrates the elaborated processes of the framework and the way the offloading decision for the annotated technique is created.The time complexness for this formula is portrayed by O(n) and doesn't consume further resources from the mobile device[9]. Firstly, at the developing step, the mobile application is partitioned the ways into 2 sorts. The primary kind is that the computing-intensive ways that are annotated by developer and needs a lot of computation resources. Whereas the second is that the ways that rely on the device hardware and should be dead regionally as mentioned higher than. Then, at the execution step, the choice maker module reads the annotated technique name (first form of methods) and checks the network status using network and information measure monitor module whereas the applying is running. once no association or failing association happens, then all of the ways are dead regionally on the mobile device; otherwise, the choice maker module reads the transfer knowledge size, memory usage, electronic equipment utilization, and energy consumption through the profile module, wherever these values are keep at the installation step and updated throughout every run. Then, the improvement model is resolved. The improvement model determines that technique is executed regionally on the mobile device and that technique is offloaded for remote execution.

Algorithm 1: Framework execution flow
Input: Input size, memory usage, CPU utilization and energy consumption for each annotated method.
Output: Execution place and result for each method.

1. Read annotated methods name.
2. Check the current network status using Network & Bandwidth monitor Module.
3. if there isn't connection then
4. Execute the method locally on the mobile device.
5. else
6. for each method i do
7. Read C transfer, C memory, and CCPU and C power through the profile module.
8. Solve the optimization model in and determine the offloading decision.
9. if the decision is offloading then
10. Encrypt the method data using AES Algorithm.
11. Send it to the cloud for remote execution.
12. Return Result back to the mobile device (communication managed using Mobile Manager & Cloud Manger Modules).
13. else
14. Execute the method locally on the mobile device.
15. end if
16. end for
17. end if
18. Update the profile file with new values.

C. Integrating the Framework in Mobile Application

In this section, we clarify the specified steps to integrate the projected framework within the mobile application. Our frame- work works on the method level and uses a Java refection and annotation to spot the ways which might be offloaded[10].

First, at the developing step, the developer has to add associate annotation (@Remote) on top of every intensive technique that may be offloaded for remote execution. Thereafter, every android application goes through 3 main builders to get the APK installation file as shown in Fig a pair of the primary builder is that the android PreCompiler, that generates the Java supply files from your android resources and Java supply files for any service interface. Second, the Java Builder compiles the generated files from the primary builder. Last, the package builder obtains all

compiled files associated packages them in an APK file. Our framework adds a replacement extra builder, known as the category and Jar Generator. This builder creates a category that contains all annotated ways code that will be dead remotely and therefore the connected libraries needed to execute these ways on the cloud. Then, the builder generates a binary file from created category and libraries that is distributed to the cloud at the installation step. The result from this builder is combined with the result from the Java Builder associated embodied in an APK file through the Package Builder.

IV. EVALUATION AND ANALYSIS

The proposed framework is evaluated using 3 different kinds of mobile applications, as shown in Table one. The experimental results live four parameters for running the appliance methods regionally on a mobile device and once offloading the strategies to the cloud by exploitation the framework. These parameters include interval, computer hardware utilization, battery consumption, and memory usage. The analysis shows however these applications will take pleasure in the planned framework for performance improvement.

A. Experimental Setup

The experimental setup for testing the planned framework consists of a Samsung Galaxy S and mobile de- vice, a server node, and a Wi-Fi wireless network of radio type 802.11g. The Samsung Galaxy S and GT-I9001 runs on android platform four.4.2 with Qualcomm MSM8255T computer hardware, 512 MB memory, and electric battery capability of 1650 mAh at3.7 volts integrated with a Wi-Fi interface. The server node runs Microsoft Windows seven final 64-bit operational system with Intel®Core(TM) i5-2500 computer hardware with two.4Gc frequency, four GB RAM capability, 600 GB magnetic disk, and a hundred Mbps network interface[11]. The mobile device accesses the wireless network via Wi-Fi wireless network affiliation of radio kind 802.11g, with the obtainable physical layer rate of fifty four Mbps. within the analysis, very little eye V2.41 computer code is employed to live interval, computer hardware utilization, battery consumption, and memory usage.

B. Experimental and Analysis Results

The application ways might have information as input for execution. Information square measure transferred over the network and hold on the cloud side if this methodology is offloaded for execution.Therefore, this information is susceptible to attacks. Crypto algorithms are required to make sure the safety of information and communications Crypto algorithms are classified as Circulate key algorithms and uneven key algorithms. Circulate key algorithms additionally referred to as single key, use a personal (shared secret) key to execute secret writing and coding method, whereas uneven key algorithms additionally referred to as public key, use a public (shared) key to execute secret writing anduses alternative personal key coding processes. The foremost well-known symmetric algorithms are DES, TDES, AES, RC6, Twofish, Blowfish, Serpent, and MARS, whereas RSA, DSA, PGP, SSH, and SSL square measure the well-known uneven algorithms[12].The AES algorithm is chosen and used as encryption technique to shield the transferred information of the applications.

In the analysis, the planned applied mathematics model of the framework is applied to the 3 applications that choose the right decision for all applications. Within the experimental work, 3 completely different scenarios are used to prove this result. Within the 1st state of affairs, the applying ways square measure dead regionally on the mobile device. Within the second scenario, the applying ways square measure offloaded for execution on the cloud by using the planned framework model while not applying any security technique. Within the last state of affairs, the applying ways are offloaded for execution on the cloud by using AES formula to shield the transferred information over the network and to indicate the consequences of adding this layer on the framework.

In the planned framework, the linear model is employed to work out the offloading decision and selects an accurate decision for the 3 applications once finding for the parameters of the ways. However, the framework is changed to permit the quick sort application to run remotely on the cloud to prove our result.

In the experimental work for the face detection application, six pictures with 360, 480, 1260, 1315 KB and nine and 11 MB sizes are used. 5 pictures with 100×100, 200×200, 300×300, 512×512, and 1024×512 resolutions are used for the Gaussian blur application. 5 arrays of 100, 500, 5000, 50000, and 100000 parts are utilized in the quick-sort application. Every application is executed twenty times for every inplace, and therefore the average values square measure obtained.

Face detection and Gaussian blur applications square measure dead in 3 totally different scenarios. The short kind application is executed using only 2 eventualities as a result of additional time is required for offloading to the cloud. Therefore, the addition of a security layer isn't required once an area execution decision is advantageous. The amount of parts within the array is depicted by the coordinate axis, and interval in milliseconds is depicted within the coordinate axis. The common interval for executing the face detection technique or mathematician blur technique within the absence of the planned framework is 5–8.5 s. the common interval once the applications are executed using the planned framework is one.5–2.5 s with none security and regarding two.8–3.5 s once victimization AES as security formula for encrypting the transferred knowledge to the cloud. If we execute the applying regionally while not mistreatment our framework, the smartphone applications utilizes associate degree 30 minutes of the process unit

|CPU|C.P.U.| central processor| processor |mainframe |electronic equipment |hardware |computer hardware on the average (except for the quick-sort application as a result of it's an easy task that takes a lot of processing for offloading than executing locally). The variation in CPU utilization will increase forthe applying that needs giant computation and small data for transfer.

The typical memory usage for executing face detection and mathematician blur applications are 50 and 26 MB severally. However, the ratios decrease to 35 and 20 MB while not security and thirty eight and 24 MB once using AES as a security layer to safeguard the transferred information. Therefore, the projected framework will improve quite half-hour of the memory usage ratio for face detection application and 100 percent for executing Gaussian Blur on the cloud.

V. CONCLUSION

In this paper, a unique secured, optimized framework is planned to enhance the potency of offloading computation from the mobile device to the cloud. This framework will offload only the appliance strategies that consume substantial mobile resources. The offloading decision is formed using a developed 0–1 number applied mathematics model. This call is formed dynamically at runtime supported four constraints, namely, memory usage, C.P.U. utilization, energy consumption, and execution time. The framework also adds a replacement security layer that uses associate AES technique to shield the strategies knowledge before transferring to the cloud within the offloading case.

REFERENCES

[1] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Communications Surveys & Tutorials*, Vol.15, No.1, pp. 179–198, 2013.

[2] G. Motta, N. S. Fondrini, and D. Sacco, "Cloud computing: An architectural and technological overview",*International Joint Conference on Service Sciences*,Vol. 3, pp. 23–27, 2012.

[3] A.U.R. Khan, M.Othman, S.A. Madani, and S.U. Khan, "A survey of mobile cloud computing application models", *IEEE Communications Surveys & Tutorials*, Vol.16, No.1, pp. 393–413, 2014.

[4] M. Shiraz, A. Gani, R.H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, Vol.15, No.3,pp. 1294–1313, 2013.

[5] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for Smartphone's", *International Conference on Mobile Computing, Applications*,Vol.76, pp. 59–79, 2010.

[6] F. Xia, F. Ding, J. Li, X.Kong, L.T. Yang, and J.Ma, "Phone2cloud: Exploiting computation offloading for energy saving on Smartphone's in mobile cloud computing," *Information Systems Frontiers*, Vol.16, No.1, pp. 95–111, 2014.

[7] W. Zhang, Y. Wen, K. Guan, K. Dan, H. Luo, and D.O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, Vol.12, No.9, pp. 4569–4581, 2013.

[8] M. Shiraz and A.Gani, "A lightweight active service migration framework for computational offloading in mobile cloud computing," *Journal of Supercomputing*, Vol.68, No.2, pp. 978–995, 2014.

[9] Shiraz, A. Gani, A. Shamim, S.Khan, and R.W. Ahmad, "Energy efficient computational offloading framework for mobile cloud computing," *Journal of Grid Computing*, Vol.13, No.1, pp. 1–18, 2015.

[10] W.Z. Zhang, H.C. Xie, and C.H. Hsu, "Automatic memory control of multiple virtual machines on a consolidated server," *IEEE Transactions on Cloud Computing*, vol.5, no.1, pp. 2–14, 2017.

[11] Y. Li, M. Chen, W. Dai, and M. Qiu, "Energy optimization with dynamic task scheduling mobile cloud computing," *IEEE Systems Journal*, No.99, pp. 1–10, 2017.

[12] J. Toldinas, R. Damasevicius, A. Venckauskas, T. Blazauskas, and J. Ceponis, "Energy consumption of cryptographic algorithms in mobile devices", *ElektronikaIrElektrotechnika*, Vol. 20, No.5, pp. 158–161, 2014.