An Analysis the Energy Consumption of Dynamic Task Offloading and Task Execution in MCC

Vivek Kumar Shukla[1]*, Dr. Qaim Mehdi Rizbi[2]

[1] Research Scholar, Shri Krishna University, Chhatarpur, Madhya Pradesh, India

Email: vivekshukla.it@gmail.com

[2] Associate Professor, Department of Computer Science & Application, Shri Krishna University, Chhatarpur, Madhya Pradesh, India

Abstract- Cloud computing is a group of distributed computing resources that may run any application. MCC is the result of merging the cloud environment with mobile devices. computational offloading is carried out on the cloud in order to save the processing capacity of handheld devices. There will be a rise in computing as the number of sensors used increases, necessitating greater data analysis. Another problem with mobile environments is the power drain on batteries. One solution is to move the work to a remote location where it can be more efficiently executed. In this off-site setting, you'll find powerful cloud servers with plenty of processing capacity. One way that dense mobile apps were able to move their work to the cloud is through computational offload. why it's important to provide a proper explanation of the time constraint in wireless & distant environments, and how reaction time affects offloading for mobile devices & remote servers.

Keywords- Mobile Cloud Computing, Cloud Computing, Offloading, Energy aware, wireless

---------------------------------- x ----------------------------------

INTRODUCTION

The benefits of wireless connections have made mobile devices more appealing for on-demand application computing. However, mobile devices' limited battery life, processing power, & storage space severely limit their performance. The integration of cloud computing into the realm of networks allows for the efficient management of these delicate aspects of mobile devices. When a cloud environment is integrated with mobile devices, it results in MCC. The cloud is a collection of enormous computer resources that can manage any application. According to Chang et al. (2013), Chun et al. (2011), and Yang et al. (2012), MCC is one of the cutting-edge areas in computational research. For this reason, computational offload is carried out on the cloud so that mobile devices can conserve their processing power. One way that dense mobile apps were able to move their work to the cloud is through computational offload. Several high-tech sensors, such as global positioning systems (GPS), light, pressure, accelerometer, and magnetometer, are integrated into modern mobile devices. There will be a rise in computing as the number of sensors used increases, necessitating greater data analysis. Another problem with mobile environments is the power drain on batteries. One solution is to move the work to a remote location where it can be more efficiently executed. In this off-site setting, you'll find powerful cloud servers with plenty of processing capacity.

Recent innovations in mobile computing have prompted programmers to consider cloud-based, adaptable application development. For this reason, Kumar et al. (2010) suggest a method for offloading tasks. The authors proposed a method to reduce mobile devices' power consumption. The cloud will be used to offload jobs that require a lot of compute. Consequently, local conditions will see a decrease in battery use. There is a limit to how much calculation the mobile devices can do. Computational offloading for mobile cloud computing has been suggested by Rong (2003) and Li et al. (2001). The software running on mobile devices is moved to a server in the cloud. So, the total amount of time needed to finish the task will decrease. Offloading is an NP-completeness problem in MCC since it involves job division. Middleware design for handling dynamic task offloading was developed by Shumao et al. (2007) as a solution to the partitioning issue. A unique approach for offloading duties during runtime is incorporated into this middle architecture. As an example, Gu et al. (2003) suggested an inference engine for offloading that is based on fuzzy control. Based on work by Zhang et al. (2010), the algorithm's primary goal is to lessen the financial burden of networking wireless devices to distant servers. In order to run on adjacent remote servers, the algorithm divides the program and uploads it.

While an adequate description of the time limitation is required in both wireless and distant environments, current algorithms have failed to address the importance of reaction time in the offloading process for mobile devices & remote servers.

MCC ARCHITECTURE'S PROPOSED WORK-DYNAMIC TASK OFFLOADING

The architectural view of MCC's dynamic task offloading (DTO) is provided in Figure 1. The three parts of this ecosystem are the wireless medium, the cloud, & mobile devices. A dynamic task offloading issue, which is a critical concern in mobile environments, is the focus of this study.

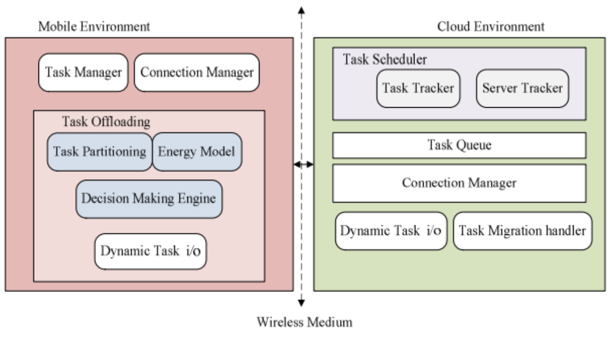

Figure 1. DTO in MCC Architecture

- **Task Manager**: Collects mobile device tasks. A combination of input data and calculation is used to manage the tasks & produce the output. The mobile device needs more power, processing power, and time to complete the tasks.
- **Connection Manager:** This component opens the wireless connection between the local & remote environments. It is important that the local client & remote establish a dependable connection.
- **Task Scheduler**: One way that cloud computing handles managing the usage of virtual machine instances is through task scheduling.
- **Task Queue:** Here are the tasks that have been transferred from the mobile environment to the cloud and are ready to be executed. From the queue, jobs are obtained and assigned to the servers that are available. The tasks are simply discarded if the queue is full.
- **Dynamic Task i/o:** One of the biggest problems with MCC is the execution of tasks in parallel. There is a wide variety of i/o workloads across jobs, each with its own unique needs for accessing data services..
- **Task migration handler:** This component enables the transfer of tasks between the local and remote environments.

PROPOSED ARCHITECTURE FEATURES

Offloading tasks dynamically: The framework uses task migration to offload tasks depending on whether the remote environment is suitable. A decision-making engine is a part of the architecture that, depending on the needs of the user, determines whether to offload the work to a mobile environment or the cloud.

Energy calculation: Due to the restricted battery power of mobile environments, energy management is of the utmost importance. To determine how much power is required by mobile devices, the cloud, and wireless networks, the suggested design incorporates an energy model into the mobile ecosystem.

Collective task execution: MCC was tasked with carrying out the entire job execution, affording to Zhang et al. (2013). On the other hand, according to the suggested design, the given job is split into 'n' jobs, & each task has its own decision variable that determines where to run.

FORMULATING THE PROBLEM

Imagine that there are n offloadable tasks in job T, denoted as T = {t1, t2, t3…tn}. Different amounts of processing power are required for different types of tasks (ti). The main problem with task offloading is deciding which tasks should run on the mobile device and which ones should be sent to a remote server for processing. Job T was divided into n tasks, as indicated in Figure 2. Think of it this way: task tn takes the output of t1 as input, and task t1 can't run without the output data of tasks t2 through tn-1.

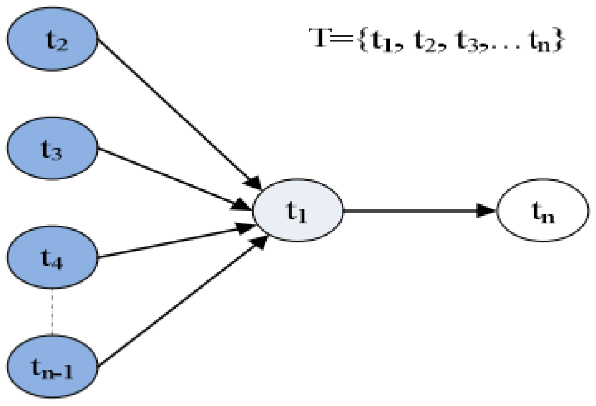

Figure 2. Data Communication among Subtasks

The process of DTO is introduced to solve the above issues. Time and energy used to complete each task are its primary focus. The job attributes inform the decision-making engine's task offloading decision.

DYNAMICS OF ENERGY-AWARE TASK OFFLOADING

An integral part of the decision-making engine in the suggested design is a dynamic task offloading algorithm, which determines if and where to offload the work. According to Wen et al. (2012), the decision-making engine must determine the local environment's energy consumption before job execution. In the suggested architecture's network model, which is based on the network model of Haung et al. (2012), the network availability during data transmission is assumed to change with the device's mobility. In Figure 2, we can see that work T is broken down into 'n' nonhomogeneous tasks before being delivered to the mobile environment. The symbol δ(t) represents the workload of every task. Given a decision variable φ(t) = {0,1}, a value of 0 indicates that the task is offloaded, while a value of 1 implies that the task proceeds in the local context.

$$\varphi(t)=\begin{cases}1 & \text{the task execution in local environment}\\ 0 & \text{otherwise}\end{cases} \qquad (1)$$

LOCAL ENVIRONMENT (εlocal)

The ti is the task executing on the local mobile environment having the computation speed (Clocal), then the completion time (τlocal) of the task ti is given as

$$\tau_{local}(t_i)=\delta(t_i)\times\varphi(t_i)\times C_{local}^{-1} \qquad (2)$$

Founded on the eq. (2.2), the function for energy consumption εlocal is defined as

$$\varepsilon_{local}(t_i)=P_{local}\times\tau_{local}(t_i) \qquad (3)$$

Where, Plocal denotes the power consumption of the mobile environment, τlocal denotes the completion time of the mobile environment.

REMOTE ENVIRONMENT(εcloud)

The ti is the task offloaded to the remote cloud environment, the completion time (τcloud) of the task ti is given as

$$\tau_{cloud}(t_i)=(I_d\times T_b^{-1})+(\delta\ (t_i)\times(1-\varphi_i(t_i))\times C_{cloud}^{-1}) \qquad (4)$$

Based on the eq. (3.4), the function for energy consumption εcloud is defined as

$$\varepsilon_{cloud}(t_i)=P_{idle}\times\tau_{cloud}(t_i) \qquad (5)$$

The variables Id, Tb, δ(ti), and τcloud stand for the input data that needs to be offloaded, the network's transmission bandwidth, the burden of task ti, and the processing speed of the cloud environment, respectively. Pidle is the power usage when the mobile environment is not in use.

INPUT DATA TO OFFLOAD

Offloading the task to a remote cloud environment requires the input data in order for the task to be executed. Offloading data from the local environment to the cloud requires a completion time τsend & energy consumption εsend, which are provided as

$$\tau_{send}(t_i)=\frac{I_d}{T_b} \qquad (6)$$

$$\varepsilon_{send}(t_i)=(P_{send}\times\tau_{send}(t_i))+\varepsilon_{network} \qquad (7)$$

Where, P_{send} represents the power consumption to offload the data and $\varepsilon_{network}$ denotes the energy consumption of the network.

OUTPUT DATA TO THE LOCAL ENVIRONMENT

The local mobile environment should receive the output data from task ti when its execution in the remote cloud environment is complete. The following are the given completion time τand energy consumption εfor the task ti.

$$\tau_{recieve}(t_i)=\frac{O_d}{T_b} \qquad (8)$$

$$\varepsilon_{recieve}(t_i)=(P_{recieve}\times\tau_{recieve}(t_i))+\varepsilon_{network} \qquad (9)$$

This is where the variables Precieve & Od stand for the power consumption and data output, respectively, to be acquired by the mobile environment from the cloud.

WIRELESS NETWORK ($\varepsilon_{network}$)

In order to create a connection, the proposed design takes into account three networks: Wi-Fi, 3G, and LTE. In the suggested paradigm, the bit rate determines the amount of energy that needs to be consumed. The network's energy consumption is provided as

$$\varepsilon_{network}=\varepsilon_{wifi}+\varepsilon_{3g}+\varepsilon_{lte} \qquad (10)$$

The network's energy consumption is determined using the equation (10). When designing a wireless medium, three factors are taken into account. The values of ε3g & εlte are set to zero in an MCC environment that utilizes a WiFi component as a wireless channel. If it leverages the LTE component for wireless communication, the remaining values of εwifi and ε3g are set to 0.

DECISION MAKING & COLLECTIVE TASK EXECUTION

• DECISION MAKING ALGORITHM

The decision-making engine is a crucial offloading module in the suggested design. Determining whether to perform in the local environment or outsource the task is the functionality of the decision-making engine. Task offloading is described in Algorithm 1, which mainly uses completion time and energy usage as its criteria. Task completion in a mobile setting must meet user expectations. So, it needs to be ready by the time the user specifies. The user threshold values are defined by the variables ᴛmin and εmin. In order to determine how long it will take to do the work, the algorithm takes the user-specified time into account. The local variable εlocal will be computed if the time it takes for the mobile environment to finish is shorter than the time the user has selected. The model calculates task 't' in the local mobile environment if it meets user requirements. In such a case, the model will transfer the workload to the nearest accessible server in the cloud.

Algorithm 1: Decision-Making

```
Input:  $T = \sum_{i=1}^{n} t_i$
Output: task execution in cloud/ mobile environment
Begin
Step 1: for i in 1 n do
Step 2: if δ(ti) ≤ 0 and φ(ti) 1 then
        Go to step 5
Step 3: for all δ(ti), if(φ(ti)==1)
                Calculate τlocal(ti);
                If τlocal(ti) ≤ τmin then
                        Calculate εlocal(ti);
                Otherwise
                         Goto step 4
                End if
                If εlocal(ti) ≤ εmin then
                        Execute ti in local mobile environment;
                Otherwise
                        Goto step 4
                End if
        End for
Step 4: if (φ(ti)==0)
                Calculate τcloud(ti) and εcloud(ti)
                Offload ti to the cloud server;
                Break;
        End if;
        End for;
Step 5: Return "do not offload";
        Break;
End
```

• COLLECTIVE TASK EXECUTION

In this section, we will look at how the local and remote environments work together to complete tasks. In particular, we take into account the scenario depicted in Figure 2, wherein, depending on the time & energy consumption compulsory to complete each activity, the tasks in the scenario are either executed locally or offloaded to a distant environment. Offloading the jobs to the distant environment is necessary since they are believed to have a lengthier completion time, from time t2 to time tn-1. In this local context, tasks t1 and tn are running. Equation (11) displays the overall energy usage of the delegated tasks.

$$\sum_{i=2}^{n-1} \varepsilon(t_i) = \varepsilon_{send}(t_i) + \varepsilon_{cloud}(t_i) + \varepsilon_{recieve}(t_i) \qquad (11)$$

Since t1 and tn are not running in the background like the other tasks, they are deciding to offload them. This has an effect on the amount of energy that is consumed.

$$\varepsilon(T) = \varepsilon_{local}(t_1) + \sum_{i=2}^{n-1} \varepsilon(t_i) + \varepsilon_{local}(t_n) \qquad (12)$$

Algorithm 2 introduces the technique for executing communal tasks while taking energy into consideration. The submitted work T's total energy usage is computed using this method. When calculating how much energy is required to complete offloaded tasks, variables like input/output data and network utilization play a significant role.

Algorithm 2: Energy-Aware Method for Collaborative Task Execution

```
Input: Decision from Algorithm 1
Output: Overall energy consumption of the Task
Begin
Step 1: if (Decision == offload)
        Calculate energy consumption of t in cloud
        Execute t
Step 2: if (Decision! = offload)
        Calculate energy consumption of t in mobile environment
        Execute t
Step 3: Combine the energy consumption of both environments
        Calculate ε(T)
End
```

The mechanism determines the energy consumption in the local environment if the task runs in the mobile environment. Ultimately, the sum of the two environments is used to determine the ε(T).

PERFORMANCE ANALYSIS

Here, we assess and analyze the performance of the suggested MCC environment architecture for dynamic task offloading.

SIMULATION SETUP

Android x86 running on AWS Software Development Kits and Programming Toolkits is used to implement the planned MCC architecture. On Amazon Elastic Compute Cloud (EC2), the cloud-side architecture is deployed. In order to generate EC2 instances, Kosta et al. (2013) utilized Android x86 AMI. Android devices with 3G, LTE, & Wi-Fi capabilities are utilized to implement the mobile side architecture. Instances in this model span three distinct regions, as indicated in Table 1, and they are all powered by the Android-x86 (2016) Amazon Elastic Compute Cloud API.

Table 1. Amazon EC2 Service Instances

Instances	CPU(GHz)	Cores	Location
Instance1	1.7GHz	2	Oregon
Instance2	2.4GHz	4	Singapore
Instance3	2.4GHz	4	Sidney

MEASUREMENT OF COMPLETION TIME

This research aims to show how efficient the suggested architecture is at executing tasks. As seen in Figure 3, the time required to complete the work grows exponentially with the size of the data involved. With reduced processing power, the local environment sees a dramatic rise in completion time. When carrying out a communal work, it is important to take into account both the immediate and distant surroundings. When comparing the distant environment to the collaborative work, the latter takes somewhat longer to complete.
The architecture that has been suggested can run in both local and distant environments at the same time. This indicates that certain work tasks are run locally while others are offloaded according to their weights when the job is partitioned.

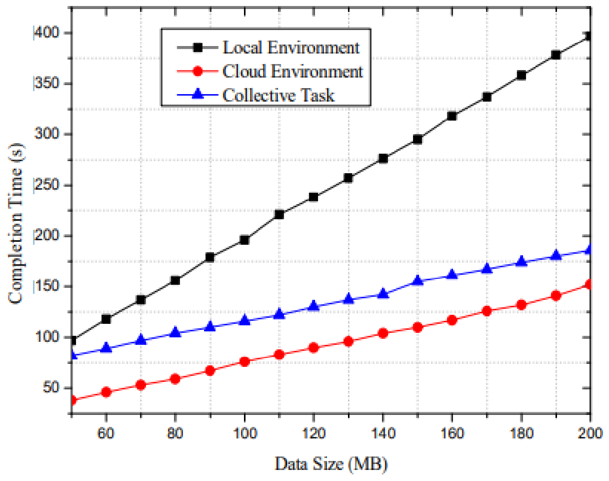

Figure 3. Task Completion Time over Data Size

Assuming that the task's execution varies according to the quantity of instructions, the suggested architecture accounts for this possibility. Adding more detailed instructions to a task causes it to take more time to complete, as seen in Figure 4. There is a comparison done between the immediate surroundings, the distant surroundings, and the group assignment. The time required to do the group project and the remote environment are almost same. However, when it comes to cost constraints, it's better to execute activities together rather than the whole duties. When compared to remote and communal environments, the local environment takes nearly three times longer to finish. Any kind of task can be completed in less time in the cloud because of the efficient resources available there. Consequently, the cloud environment has a shorter completion time than the remote environment.

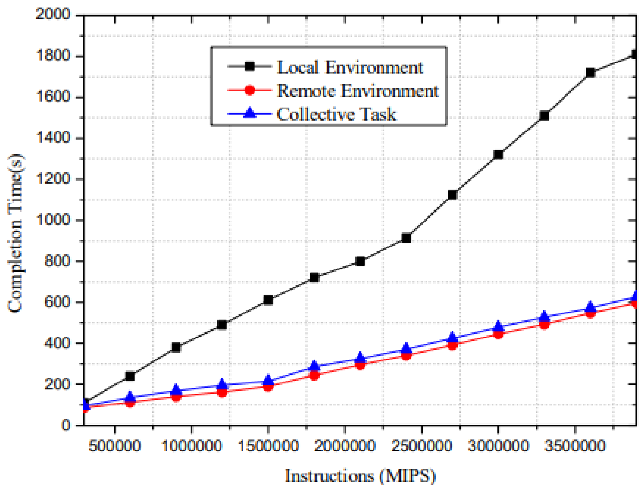

Figure 4. Task Completion Time over the Number of Instructions

The proposed Ternary Decision Making algorithm (TDM) & existing method are compared in Figure 5, which is sourced from Lin et al. (2015). The identical conditions that were used to test the original algorithm are also used to test this TDM algorithm. The suggested approach outperforms the TDM in terms of completion time. This is because the TDM does not support the proposed model's communal task execution. Consequently, the suggested model had a lower computation time compared to the TDM.

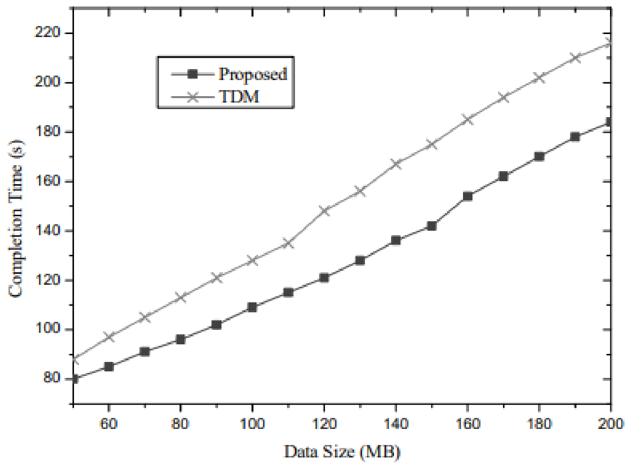

Figure 5. Task Completion Time of Proposed Method and TDM

ENERGY CONSUPTION OF THE PROPOSED MODEL

Kumar (2010) found that CPUs in the local area consume an average of 0.9W of power at 500 MIPS. With regard to data rate & task instructions, Figures 6 and 7 comparison the task's energy consumption in the local environment with that in the remote environment, as well as in both settings. The values in Table 2 are used to test the network's energy usage, where λ is the data size in kilobytes.

Table 2: Configuration Parameters for Three Networks' Energy Consumption During Upload & Download

Energy consumption	3g	LTE	Wi-Fi
Scanning energy (J)	NA	NA	3.9
Transfer energy download (J)	$0.025(\lambda\times2^8)$	$1.01(\lambda\times2^8)$	$0.007(\lambda\times2^8)$
Transfer energy Upload(J)	$0.030(\lambda\times2^8)$	$1.06(\lambda\times2^8)$	$0.009(\lambda\times2^8)$
Ramp energy (J)	3.5	4.2	NA
Tail energy (J/s)	0.62×12.5	0.74×32.5	NA
Maintenance (J/s)	0.02	1.26	0.05

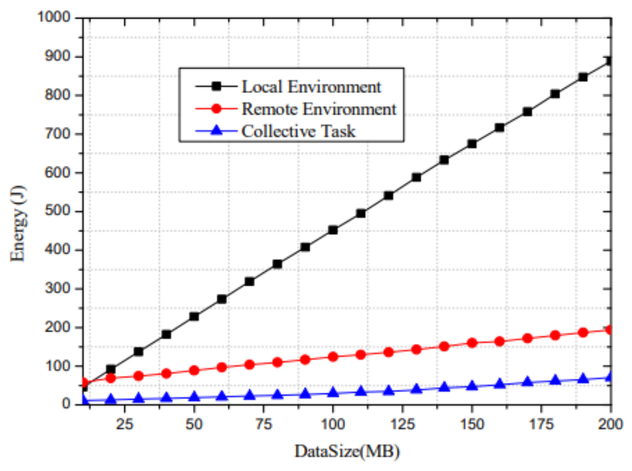

Figure 6. Energy Consumption Vs Increasing Data Size

The energy consumption of tasks is illustrated in Figure 7 with respect to the number of instructions, and in Figure 6 with respect to the rising data size. The energy savings from the group job execution relative to both contexts are clearly seen in both figures. Because it is always changing, the collective task execution uses less energy. The distant environment takes data as input and provides data as output to the local environment. Consequently, energy usage will be higher in remote areas. When compared to fully executing tasks in a remote setting, the flexibility of using a dynamic offloading decision algorithm to offload tasks during collective task execution is superior. When compared to a distant environment, the average energy usage of a group working on a job is three times lower.

In Figure 8, we can see the energy usage of the Ternary decision making (TDM) method and the proposed model. The suggested model has lower energy consumption than the TDM. The suggested model carried out the operations in both the local and remote settings. The local environment's resource consumption is decreased as a result of work being offloaded to the remote environment. Consequently, saving energy is achieved by the execution of tasks in both local and remote environments in simultaneously. In the end, the suggested approach uses less power than the TDM while still supporting simultaneous job execution.

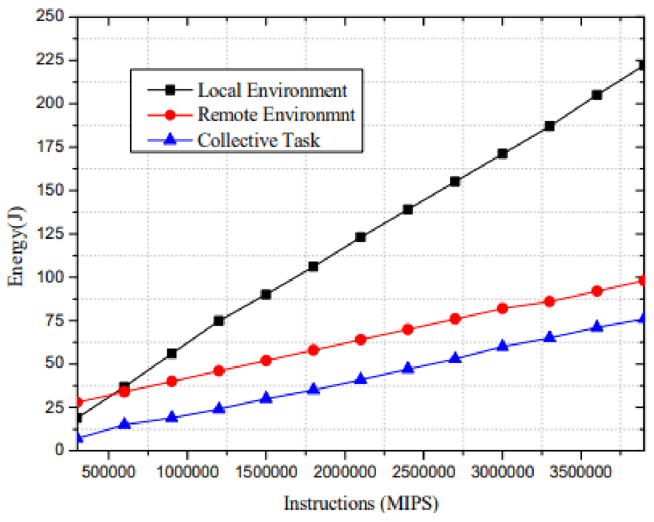

Figure 7. Energy Consumption Vs the Number of Instructions

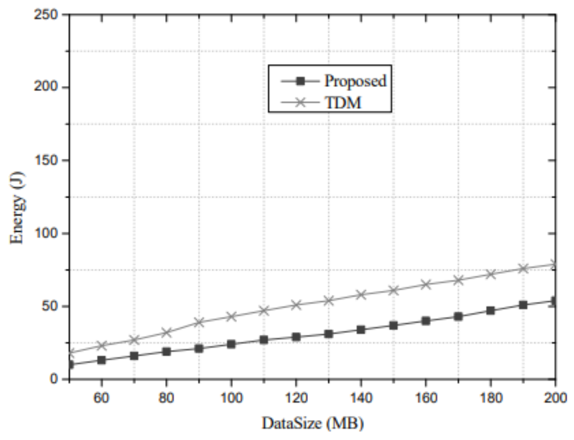

Figure 8. Energy Consumption of Proposed Model and TDM

CONCLUSION

This article presented a new system design for MCC's dynamic task offloading that would lower task energy usage. The program takes into account the task's energy consumption in three different environments: the local, the remote, and the network. In addition, we suggest a collaborative task execution strategy to reduce the total energy consumption of the tasks that have been submitted. In addition, the simulation results demonstrate that the suggested design is effective for group job execution in terms of reducing energy consumption in both the local & remote environments, albeit at the cost of increased overall completion time.

REFERENCES

1. Ali, M., J. Zain, M.F. Zolkipli and G. Badshah (2015). Mobile cloud computing & mobile's battery efficiency approaches: A

Review. Journal of Theoretical and Applied Information Technology, Vol. 79, No. 1, pp. 153.

2. Boukerche, A., Guan, S., & Grande, R. E. D. (2019). Sustainable offloading in mobile cloud computing: algorithmic design and implementation. *ACM Computing Surveys (CSUR)*, *52*(1), 1-37.

3. Cui, Y., Ma, X., Wang, H., Stojmenovic, I., & Liu, J. (2013). A survey of energy efficient wireless transmission and modeling in mobile cloud computing. *Mobile Networks and Applications*, *18*(1), 148-155.

4. Mayo, R.N., and P. Ranganathan (2003). Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down. In Proceeding of the International workshop on Power-Aware Computer Systems, pp. 26-39.

5. Nagaraju, D. and V. Saritha. Energy-aware dynamic task offloading and collective task execution in Mobile Cloud Computing. Communicated to Journal of Supercomputing. (Scopus Indexed).

6. Ou, S., K. Yang and J. Zhang (2007). An effective offloading middleware for pervasive services on mobile devices. Pervasive and Mobile Computing, Vol. 3, No. 4, pp. 362-385.

7. Qureshi, S.S., T. Ahmad and K. Rafique (2011). Mobile cloud computing as future for mobile applications-implementation methods and challenging issues. In Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, pp. 467-471.

8. Rahman, M., Gao, J., & Tsai, W. T. (2013, March). Energy saving in mobile cloud computing. In *2013 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 285-291). IEEE.

9. Saad, S. M., & Nandedkar, S. (2014). Energy efficient mobile cloud computing. *International Journal of Computer Science and Information Technologies*, *56*(5), 1757-1771.

10. Tang, C., Xiao, S., Wei, X., Hao, M., & Chen, W. (2018, January). Energy Efficient and Deadline Satisfied Task Scheduling in Mobile Cloud Computing. In 2018 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 198-205).IEEE.

11. Wang Z., Pang X., Chen Y., Shao H., Wang Q., Wu L., Chen H. and Qi H. (2019), "Privacy-preserving Crowd-sourced Statistical Data Publishing with an Untrusted Server", IEEE Transactions on Mobile Computing, 18(6), pp. 1356-1367, 2019.

12. Wang, K., Yang, K., &Magurawalage, C. S. (2016). Joint energy minimization and resource allocation in C-RAN with the mobile cloud. IEEE Transactions on Cloud Computing, 6(3), 760-770

13. Xie J, Dan L, Yin L, Sun Z, Xiao Y, 2015, 'An energy-optimal scheduling for collaborative execution in mobile cloud computing', In: 2015 International Conference and Workshop on Computing and Communication (IEMCON), IEEE, pp. 1–6.

14. Xiong Y., Huang S., Wu M., She J and Jiang K. (2019), "A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing", IEEE Transactions on Cloud Computing, 7(3), pp. 597-610, 2019.

15. Xu B, Peng Z, Xiao F, Gates AM, Yu J-P, 2015, 'Dynamic deployment of virtual machines in cloud computing using multi-objective optimization', Soft Computing, Vol. 19, No.8,pp.2265–2273.

16. Xu X., Dou W., Zhang X and Chen, J. (2016), "EnReal: An Energy-Aware Resource Allocation Method for Scientific Workflow Executions in Cloud Environment", IEEE Transactions on Cloud Computing, 4(2), pp. 166-179, 2016.

Corresponding Author

Vivek Kumar Shukla*

Research Scholar, Shri Krishna University, Chhatarpur, Madhya Pradesh, India

Email: vivekshukla.it@gmail.com