AIP Conference Proceedings

Develop a decision-making algorithm based on cloud storage

Cite as: AIP Conference Proceedings **2776**, 040002 (2023); https://doi.org/10.1063/5.0137262
Published Online: 12 April 2023

Haeder Talib Mahde Alahmar

Develop a Decision-Making Algorithm Based on Cloud Storage

Haeder Talib Mahde Alahmar[a)]

Technical college Al-Mussaib, Al-Furat Al-Awsat Technical University,Babylon, Iraq.

[a)]Corresponding author: haeder.ahmar@atu.edu.iq

Abstract. Despite many developments and improvements in mobile devices, it still suffers from its limited capabilities such as short battery life, limited processing capabilities, and small storage spaces that can be overcome by canceling the implementation locally and attributing it to the cloud, in this research all cases have been applied to methods Implementation in mobile applications, which can be either Java methods, (C++) methods, or CUDA methods that require calculations to be performed on the cloud server's graphics processor (GPU).We have been working on proposing and developing an algorithm for decision-making in attributing implementation before starting it, according to the network status, as well as developing an application using (Android Studio) accompanying this research, through which the best methods of implementation can be determined and thus it becomes possible to make the final decision in determining a clear mechanism for implementation and choosing between two states, either the implementation is local or on the cloud server. The results are set for each possible implementation case (locally - remotely), based on the decision-making algorithm by determining the implementation time taking into account the mobile device CPU speed, network performance, application program characteristics, and cloud server efficiency. It is important to rely in this research on taking into account the CPU speed of the mobile device, network performance, application program characteristics, and the efficiency of the cloud server to reach the best results that were reviewed in this research with the use of different platforms to choose the best one in accomplishing our work. The actual results of the applied algorithm proved a significant saving in implementation time with the increase in the request to perform tasks on the cloud server and with the increasing complexity of the issue.

Keywords. Moving Cloud Computing (MCC), Decision-Making, platform (MAUI).

INTRODUCTION

Cloud computing mainly allows IT departments to focus on their business and projects rather than caring for their data centers [1]. Cloud computing expresses a new concept that aims to provide computing resources as services in a rapid manner, upon request, and to pay as peruse. The cloud computing model is presented in three models of cloud connection, as shown in Figure 1.:

- Infrastructure as a (laaS) Service.
- The platform as a (PaaS) service.
- Software as a (SaaS) Service.

1st International Conference on Achieving the Sustainable Development Goals
AIP Conf. Proc. 2776, 040002-1–040002-16; https://doi.org/10.1063/5.0137262
Published by AIP Publishing. 978-0-7354-4441-6/$30.00

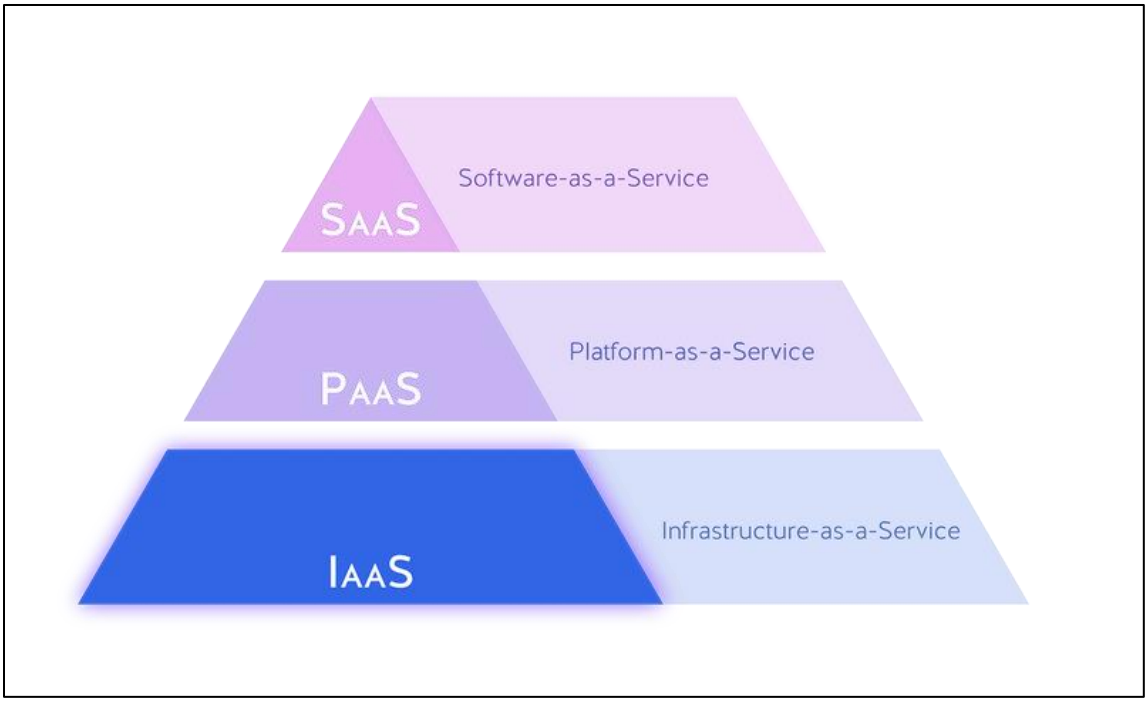

FIGURE 1. Main cloud delivery models

According to what the researcher presented in [2], the global annual growth rate of cloud computing in 2016 was distributed as follows:

- Infrastructure as a Service (laaS) (41%).
- The platform as a service (PaaS) (26.6%)
- Software as a service (SaaS) at (17.4%)

The simplest form of mobile computing (MCC) refers to the infrastructure in which data is stored and data is processed outside the mobile device. Mobile cloud applications transfer computing power and data storage away from mobile devices to powerful and centralized computing platforms in the cloud, which are then accessed via wireless.

The user's preferences for computing have changed over the past few years due to another reason: developments and improvements in mobile computing technologies. The importance of (MCC) has been presented in many reports and studies that confirmed the impact of (MCG) on customers and companies. According to a recent study conducted by (ABI Research), more than 250 million cloud services companies used mobile devices in 2018, and this will push the MCC return to 6.5.$ billion [3]. Smartphone use has increased rapidly in various areas, including organizations, information systems management, games, e-learning, entertainment, games, and healthcare. Although predictions that mobile devices will control future computing devices, these devices with their applications are still restricted by some restrictions such as battery life, processor capabilities, and memory capacity [4]. Modern mobile devices have sufficient resources such as fast processors, large memory, etc. However, this is still not sufficient to carry out tasks that require many calculations such as natural language processing, image recognition [5], and decision-making. Mobile devices provide less computer power compared to computers, as the tasks that require many calculations put heavy loads on battery power.

RELEVANT STUDIES

here is a lot of research presented in the field of cloud computing that was mainly aimed at enhancing computing readings for resource-limited mobile devices by providing access to cloud infrastructure, software, and computing services.

for example, Amazon Web services are used to protect and store customer personal data via (Simple Storage Service-S3) [6].

as well Many platforms allow remote processing of tasks on cloud servers that require processing large amounts of data.

and Cloud Implementation Stadium platforms (ASM) [7] have shown that relying on remote implementation to protect user accounts helped reduce the energy consumption cost of mobile devices by 33%, and application response time by 45% [8].

Most research that transfers tasks from mobile devices (Smart Mobile Device - SMD) to a remote server divides the work into three basic steps:

- Application segmentation: This step is very important for mobile computing in Its dependence on external sources. The app includes both remote and non-executable components, which means that components to be kept on the mobile device must be challenged and components migrated to the cloud server.
- Setup: In this step, all required procedures for remote executable components are prepared to enable their use in mobile applications. This includes testing a remote server, deciphering and installing code, and transferring data for remote execution.
- Decision-making: This step is the final step before beginning the remote implementation of the executable components. In this step, the final decision is made on whether or not to implement remotely, which largely depends on the context of the current implementation. The decision is made at run time, as more accurate information must be available. For example, the mobile device may not have a wireless connection and therefore the power consumption for data transmission for remote execution is very high. The researchers focused on how to divide the application into executable parts on the (Off-loadable) and (Un off-loadable) parts, as the decision-making process during the implementation of the application did not receive sufficient attention.

There are two basic phases in the reference processing to cloud before starting remote implementation: Divide the code into parts that are (off-loadable) and some of which are (Un off-loadable).

- Transfer the downloadable parts to the cloud server.

To reduce the computational burden of mobile devices, researchers [9] introduced a dynamic implementation algorithm based on dynamic programming (DPOA) to find the optimal fragmentation of mobile application components and implement them either on the mobile device or on the cloud server and achieve the fastest possible implementation time, Taking into account the mobile device CPU speed, network performance, application program properties, and cloud server efficiency, this DPOA algorithm has found the best ways to remotely execute parts of the application that lead to reduced implementation time, which mobile applications have been It is due through simulation on the MATLAB program but fails when increasing the mathematical complexity of the application. The researchers focused on [10] on assigning implementation to the cloud in mobile cloud computing, as they tried to determine the components that should be assigned to the cloud so that the application can be completed at the lowest cost of implementation and that for a specific set of computational components that make up a mobile app. The researchers in this research classify the issue of cloud support as a matter of example, and then they proposed a reference algorithm to get the best solutions for implementing mobile applications, the first algorithm was proposed mobile phone applications that consist of components linked together, while the second algorithm was proposed for applications General made up of parts with unknown reliability between them. In this research, the efficiency of the proposed algorithms was evaluated based on numerical experiments.

The researchers [11] introduced a new reference algorithm called (DPH), as they used a random distribution and Hamming standard to find the best attribution solution tasks and implement them in the shortest possible time. This algorithm relies on assigning as many tasks as possible to the cloud when the network transmission bandwidth is high, which improves the overall execution time of all tasks and reduces the power usage of the mobile device. The algorithm can find very good solutions with reduced computational complexity. In this research, the researchers used a new and innovative method to fill the dynamic programming joints to avoid unnecessary calculations, thus reducing the calculation time compared to other algorithms. The DPH algorithm is scalable to be able to deal with larger attribution problems without losing arithmetic efficiency. The performance evaluation shows that the proposed algorithm (DPH) can achieve the minimum energy while meeting the imposed application time constraints, and can find the decision to assign is almost perfect within a few iterations.

To address the challenges posed by the strict restrictions on application completion time, researchers in [12] have introduced a dynamic, energy-efficient attribution and resource scheduling (eDor) policy to reduce energy consumption and reduce overall implementation time. First, the researchers drafted the question (eDors) within the framework of the issue of reducing energy cost (EEC) to a minimum, while achieving the requirements of the tasks that are being implemented and adherence to the deadline for completion. To solve the problem of examples and obtain the best solutions, the researchers then proposed an algorithm (eDors) consisting of three sub-algorithms:

achieving the optimal reference, monitoring the clock frequency, and allocating the transmission energy. The researchers found that the method of achieving the cloud reference does not depend only on the burden imposed by the task It also depends on the time of execution of the tasks preceding it, the clock frequency, and the transmit power of the mobile device. Experimental results and in a real test have proven that the (eDors) algorithm can effectively reduce the energy cost (EEG) by setting the clock frequency (CPU) to a Perfect way on a mobile device.

Researchers in [13] have introduced a platform (MAUI), this platform allows to attribute the tasks that consume energy in mobile devices to the infrastructure in the cloud and thus reduce the burden on the programmer as possible. The previous methods of these problems largely depended on the programmer splitting the application, and therefore each whole process required a complete (VM). During operation, the (MAUI) platform decides which methods to implement remotely, relying on it on methods and technologies that achieve the best possible energy savings in light of the current connection restrictions for the mobile device. At the end of their research, the researchers showed that the (MAUI) platform can do many tasks, including:

- A resource-based facial recognition application that consumes less energy.
- The application of the (arcade) game is sensitive to response time, which leads to double the refresh rate.
- A language translation application that transcends the boundaries of the smartphone environment by implementing unsupported components remotely.

But in this research, we focus on the second section by proposing an algorithm that decides how to implement (locally or remotely) and apply it within a wireless network concerning a specific set of computational components that constitute an application, by making the appropriate decision in implementation Remote or local implementation.

RESEARCH PROBLEM

Research in the field of outsourcing to support mobile cloud computing focused mainly on how to assign tasks to the cloud as well as determine what is being attributed by splitting the components of the mobile application into parts that are assignable and parts not attributable, but the biggest problem lies in decision-making for local implementation or remote implementation Which did not receive much interest in the research. Current implementation attribution schemes require either a compilation or a special modification of the code of the different applications, which makes it difficult to publish them in practice since most of the research was based on simulating using MATLAB without actually applying it or through numerical experiments that are not included in actual applications.

RESEARCH OBJECTIVE

This research aims to develop a mobile application through which to determine the best methods of implementation and thus it becomes possible to make the final decision in determining a clear mechanism for implementation and choosing between two cases, whether the implementation is local or on the cloud server located on a virtual machine (VM) in a home network consisting of A mobile device and a laptop with a mechanism for handling the calculations required from the application. The decision-making process will take place through the development of an algorithm and its use in solving actual problems, not simulating with (MATLAB). The results will be developed for every possible implementation case by specifying the execution time taking into account the mobile device CPU speed, network performance, application program characteristics, and cloud server efficiency.

CODE OFFLOADING

Remote implementation techniques have been studied since the computer network was developed. These technologies aim to transfer code that requires complex calculations from resource-limited computers to resources on remote servers to speed up the calculation process and reduce power consumption on devices with weak batteries. Figure.2 shows a general concept of code transference. Initially, in the general transference model, the instructions that require complicated calculations are determined in the mobile applications, then they are presented through the decision-making process to transfer them and implement them on the cloud or keep them on the mobile

device to be implemented, based on the goal of the mobile cloud enhancement service (e.g. energy saving). Implementation is outsourced to remote computing resources by relying on various types of technologies available.

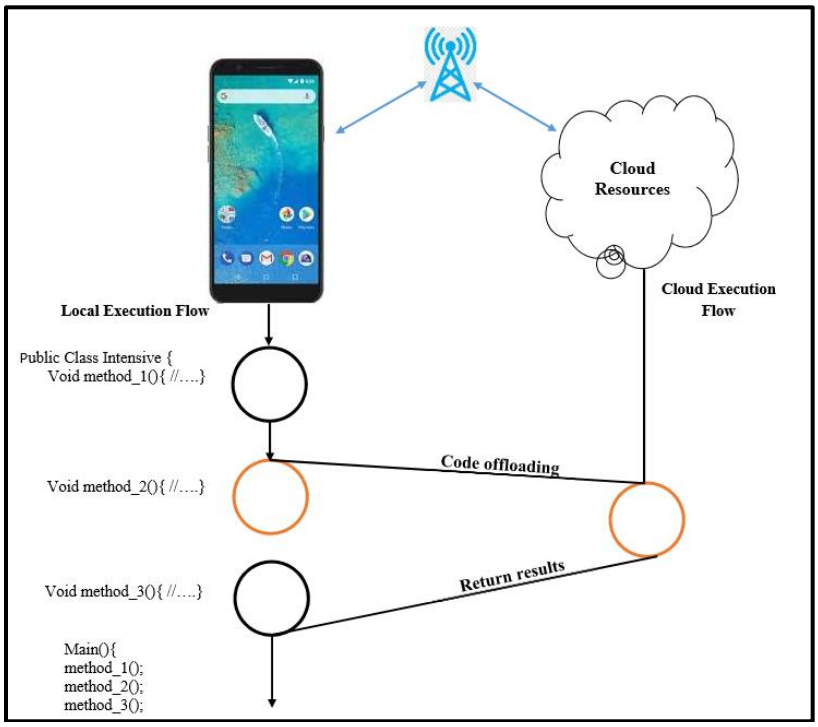

FIGURE 2. Concept of attributing code implementation

Various implementation-based techniques assume that all application code is executed locally on mobile devices, enabling mobile devices to migrate to resource-rich computing devices, and thus mobile applications become more flexible in terms of outsourcing. The main disadvantage of this implementation attribution process is that the code requires developers to split the code and define the instructions to attribute the implementation to the cloud, which is an unnecessary task and may impose unnecessary expenditures on mobile devices. Various methods of migrating instructions have been proposed and applied by proposing different software models for mobile applications. These methods can be categorized into four main categories: (Partitioned Offloading), (VM Migration), (Mobile Agent-Based), and (Replication Based- Offloading).

FACTORS AFFECTING HOW TO MAKE A DECISION

Factors affecting how remote decision-making can be categorized into several different classifications such as the contents of mobile applications, the mobile cloud environment, device characteristics, and user preferences are shown as follows:

- The contents of the mobile applications.
- The mobile cloud environment
- Hardware properties
- User preferences.

THE CLOUD APPLIED IN THIS RESEARCH

To achieve the cloud attribution, two main techniques were relied upon: displacement (VM) and programming using the Java language, as well as the network analytical model was adopted and a statistical principle was adopted and network status monitoring to make the appropriate decision when implementing the application dynamically according to the proposed algorithm.

A special registration mechanism has been adopted that allows mobile devices to find and connect to nearby devices automatically and therefore use the required resources, after that the operating system, which works on each mobile device takes into account many values that represent the status of the connected device such as the local status and task requirements and the status of Virtual machines on the cloud to which it is communicating to decide whether tasks must be performed locally or remotely. Round algorithms, acceptance control policies, service level agreements, and licensing policies are used to serve multiple applications that are accelerated efficiently on heterogeneous cloud infrastructures [14].

The application being worked on enables highly demanding applications to the central processing unit (CPU) or graphics processing unit (GPU) through physical or virtual machines with high capabilities or resources, and dedicated scheduling policies were used through which a decision could be made to use Acceleration implementation service by assigning code execution to more capable devices when this is considered necessary or useful. An application was built consisting of some non-assignable tasks to the cloud (i.e., it must be performed locally) and a set of assignable (N) tasks. Local tasks include those that deal directly with user interaction, access to local (Input / Output) devices, or access to specific information on the mobile device, so local tasks on the mobile device must be handled locally, in some scenarios we can combine all local tasks into one task In this research, an application was built that contains a set of non-assignable tasks and three tasks that are assigned to the cloud according to three scenarios, that is, the division of tasks into two sections has been determined in advance without the use of a partitioning algorithm.

Figure 3. illustrates the main components that were activated to obtain the results accompanying this research, which can be illustrated as follows:

- Agent Device: It is the device with low power (mobile phone, for example), which will speed up the implementation by attributing the implementation of the code, in our scenario in this research, it will be a phone that works with (Android) version + 4.1.
- Agent Library: It is an Android library. Attribution support for code required by Android applications)).
- Application: It is the Android application that will be accelerated by the platform. This application includes the driver component (AL) as a built-in library for attributing remote code execution.
- Virtual machine (VM): It is a virtual device that runs on virtual programs and is installed on an operating system similar to the operating system installed on the first component (UD). In our scenario, we will hit (Android-x86) version 4.8 using Virtual Box.
- Server: This component represents the cloud server that carries out the tasks entrusted to it and that was built in the form of an application (Android) running on the aforementioned (VM) device and is responsible for implementing the code that was assigned to download to the cloud. The cloud built here can be classified as a public cloud.

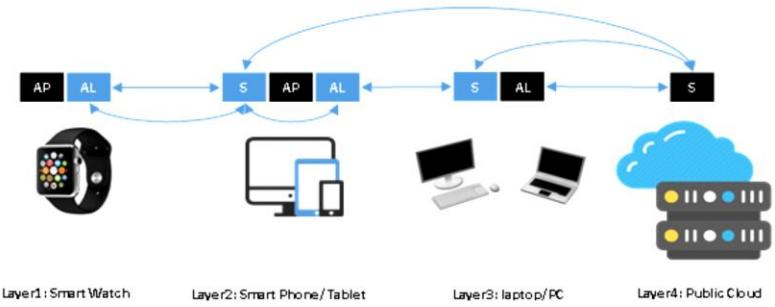

FIGURE 3. The main components of the working mechanism used in this research

THE WORKING MECHANISM USED IN THIS RESEARCH

Figure 4. shows a blueprint for the working mechanism used in this research, considering that the method used to choose the method of implementation is dynamic based on the proposed decision-making algorithm.

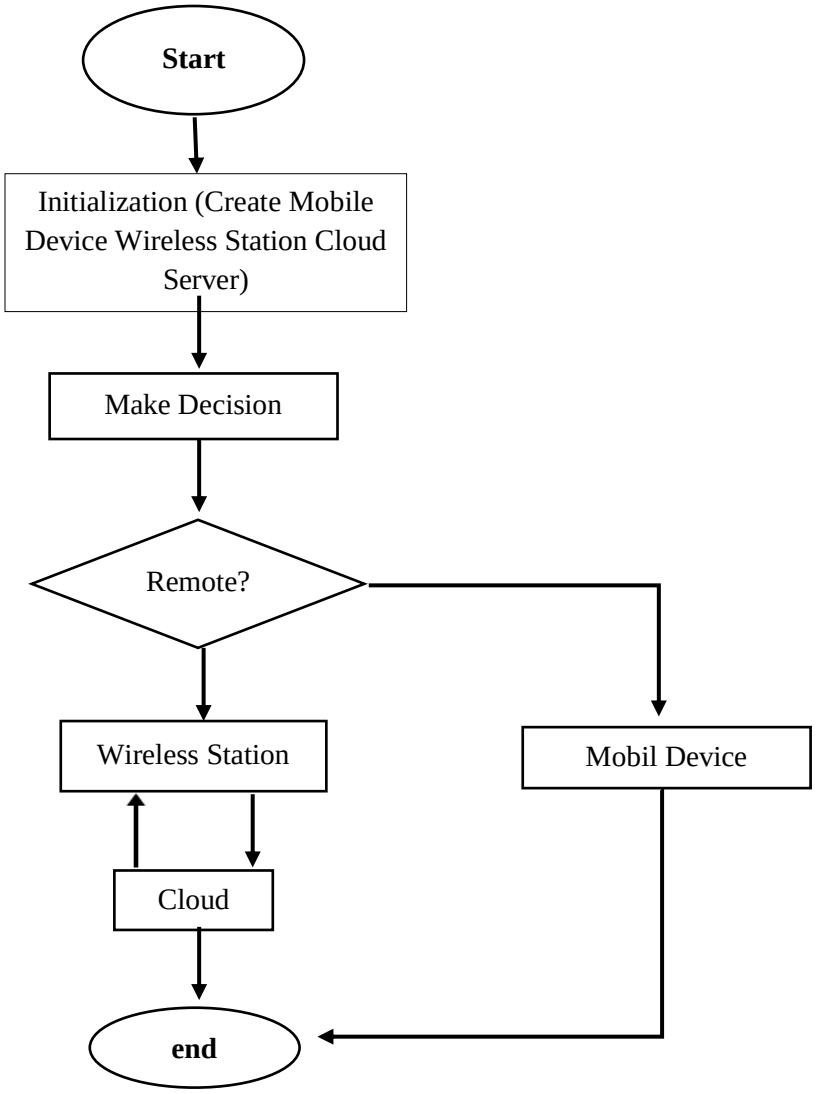

FIGURE 4. Mechanism of work used in this research

The unit responsible for identifying the characteristics of the mobile device (Device profiler) and the application analyzer calculates the cost of the method when operating on the mobile device according to the CPU speed and the complexity of the application program. The Network Profiler unit calculates the cost of data transmission for each method that must be assigned, taking into account the network properties and the amount of data transmitted from each method. Using all the information provided, the decision-making algorithm calculates and decides to assign implementation to the mobile application before starting to implement it.

Figure 5. shows a box diagram of the units that were built to help the algorithm make the appropriate decision regarding the most appropriate method of implementation.

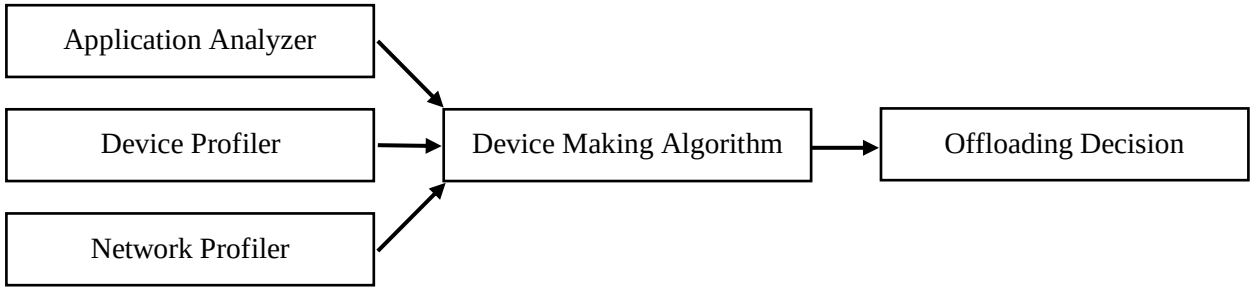

FIGURE 5. A box diagram of the units assisting in decision making

THE DESIGNED NETWORK MODEL

Considering that the application on the mobile device includes (N) a separate mission from each other. The tasks to be performed include two types of tasks:

- Executable tasks on the cloud server: Three separate tasks that can be assigned to the cloud, according to three possible scenarios.
- Non-executable tasks on the cloud: The application consists of (M) a task that must be performed locally as shown in Figure 6.

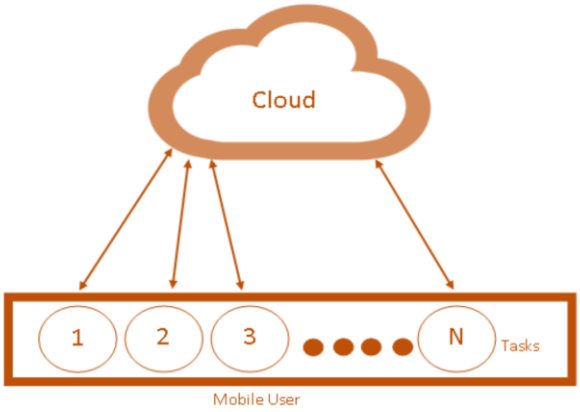

FIGURE 6. The designed check models

We assumed that a wireless (WIFI) network is available for the mobile device, but the network bandwidth can change dynamically. Wireless interference and network congestion usually change the network transmission bandwidth dynamically. The mobile device needs to determine whether each local task must be addressed or assigned to its cloud implementation, according to the current wireless network transmission bandwidth.

After the time it takes to transfer a task between a mobile device and the cloud over a wireless link is an important problem because there are full execution time restrictions for all tasks, so the applied algorithm must take into account the current bandwidth of the wireless network when making the decision.

Figure 7. shows a box diagram of the local network that was built using a computer, a mobile device, and a router.

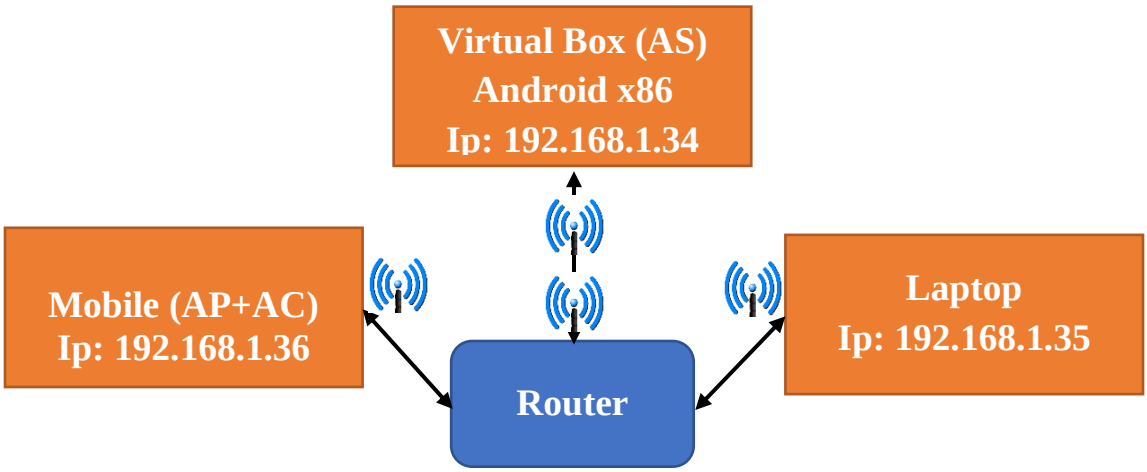

FIGURE 7. A local network box diagram applied in this paper

FORMULATE THE ISSUE

The mobile application consists of a set of tasks. Assuming that all methods can be implemented on the mobile device, and some of them can be executed on the cloud server, the cost of one method, when implemented on the mobile device, differs from its cost when implemented on the cloud server in terms of execution time and power consumption.

For two consecutive methods, there will be additional time transferring data and energy cost to the mobile device if the current task changes the way it is executed from the way it works in the previous method, and if both successive methods are implemented in the same method, there is no additional time or extra cost. The transfer time and power cost of the mobile device vary depending on the amount of data to be transmitted and the status of the wireless network.

(D_i) represents the cost of performing the task (i) on the mobile device, while (C_i) is the cost of implementation in the cloud, (CS_i) is the cost of sending mission data (i) over the network, which is calculated by multiplying the cost for each broadcast by the size of the method. (P_i) is the place where the method is performed:

- 1: On the mobile device
- 0: On a cloud server.

The cost mentioned here is the time taken to implement the relevant method. The issue can be formulated as aiming to obtain the minimum cost in the system, which can be calculated as follows:

$$\sum_{i}^{N} (P_i \times D_i + (1 - P_i) \times C_i + |P_i + P_{i-1}| \times CS_i) \quad (1)$$

Given that (N) is the number of tasks that can be assigned to the cloud. From equation (1) you can find that the transferring cost is calculated only if the current method of implementation differs from the previous method.

The cost of data transfer (CS_i) can be calculated with the Network Identification Unit (Network Profiler). The network speed can be obtained by measuring the period when sending a certain number of data as in [15]. It is necessary to measure network properties in a short period as wireless and computing environments change rapidly.

METHODS OF REMOTE IMPLEMENTATION FOLLOWED IN THIS RESEARCH

Three remote implementation scenarios were followed in this research to obtain the results of the importance of remote implementation and to compare the three scenarios with the corresponding local implementation in each of the different scenarios.

- Attribution of Java implementation methods to the cloud: This scenario is the simplest case of implementation attribution in portable computing, dealing with the remote implementation of Java methods. In this scenario, the issue of Ministers (N - Queens) will be resolved after a variable of Ministers. The issue of (N - Queens) involves the task of arranging (N) a minister on a chessboard with dimensions (NxN) so that no minister can attack another minister on the board. The implementation that has been implemented is based on the regression algorithm, whose implementation will be assigned to the cloud. The user can change the number of ministers in the application from 4 to 8, and thus the issue will become more difficult and time-complicated, in the event of local implementation, the component (AL) will be responsible for finding the solution on the device or remotely on (VM). The results will be displayed in terms of the number of local executions, the number of executions remotely, and the average duration of executions in both cases is set in milliseconds.
- Attribution of the implementation of the followers written in (C / C ++) and included in Android applications: (Android) allows developers to include software codes in (C / C ++) in the various applications to increase the performance of tasks or to allow the reuse of instructions Software. (Java) methods can call followers of (C++) by (JNI) library. In this paper, an application that simply returns only a text string that was implemented in (C ++) has been included and was included as a native application library (AP). Results will be displayed in terms of the number of local executions, the number of executions remotely, and the average duration of executions in both cases is estimated in milliseconds.
- The assignment of graphics processing tasks (CUDA) included in Android applications: This scenario is complex in terms of implementation, which includes methods that must be implemented using (GPU) as is the case in different graphics applications. When the local CUDA method is executed, if the client does not

have a GPU, operations from the client machine will be assigned to the server (AS), which it performs on the physical GPU of the device on the cloud, when the remote method calls (CUDA) are executed.

THE PROPOSED ALGORITHM FOR DECISION MAKING IN THE LOCAL OR REMOTE IMPLEMENTATION

The focus of the researchers in this field was on suggesting algorithms to divide the application, the parts are subject to remote implementation or local development without paying attention to the necessary decision-making algorithms to determine the implementation mechanism for the task according to the network status and actual calculation requirements in terms of the amount of data to be transferred on the network.

To make decisions in remote implementation or local implementation, and if the application is relied upon to make the decision dynamically, an algorithm has been proposed and developed based on the algorithms applied in [16] [17]. This algorithm decides for remote implementation or local implementation. In this research, some steps have been added to the original algorithm that preserves the time spent for each of the previous operations of the application, whether when implemented locally or remotely, that is, it maintains an archive of previous operations and times Consumed for each implementation thereof. The steps for the modified algorithm are as follows:

- Creating the variables for the local implementation (number of execution times - execution time (Tl_i).
- Creating the variables for remote execution (number of executions - execution time (Tr_i)).
- Knowing the previous method of implementation and verifying that it was done locally or remotely.
- If the number of remote execution times until now is equal to zero and the current connection is made good, assign execution to the cloud.
- In the event of local execution, (Tl_i) will be calculated according to equation (1), and then the mean value (meanDurLocal) will be calculated for the time spent during implementation locally, which was obtained from previous operations, as more weights are given to the last operations.
- In the case of remote implementation, (Tr_i) will be calculated according to equation (1), and then the mean value (meanDurRemote) will be calculated as follows:
  - The first mean value (meanDurRemote l) is equal to the mean value obtained during local execution.
  - The second mean value (meanDurRemote 2) This value is calculated by modifying the record (RD) that contains three entries representing the time taken for each of the three fastest runs to date with greater weight given to the fastest operation (RD [0])

$$meanDurRemote2 = \frac{\frac{RD[2] + RD[1]}{2} + RD[0]}{2} \quad (2)$$

  - Calculated

$$meanDurRemote = \frac{meanDurRemote1 + meanDurRemote2}{2} \quad (3)$$

  - Modify records [2] RD, [1] RD, [0] RD) after obtaining the current implementation time.
- Deciding on the remote implementation in case the remote implementation is faster than the local implementation.
- Return to step 3 when the task is executed again, given that the number of implementation times park is only ten times).

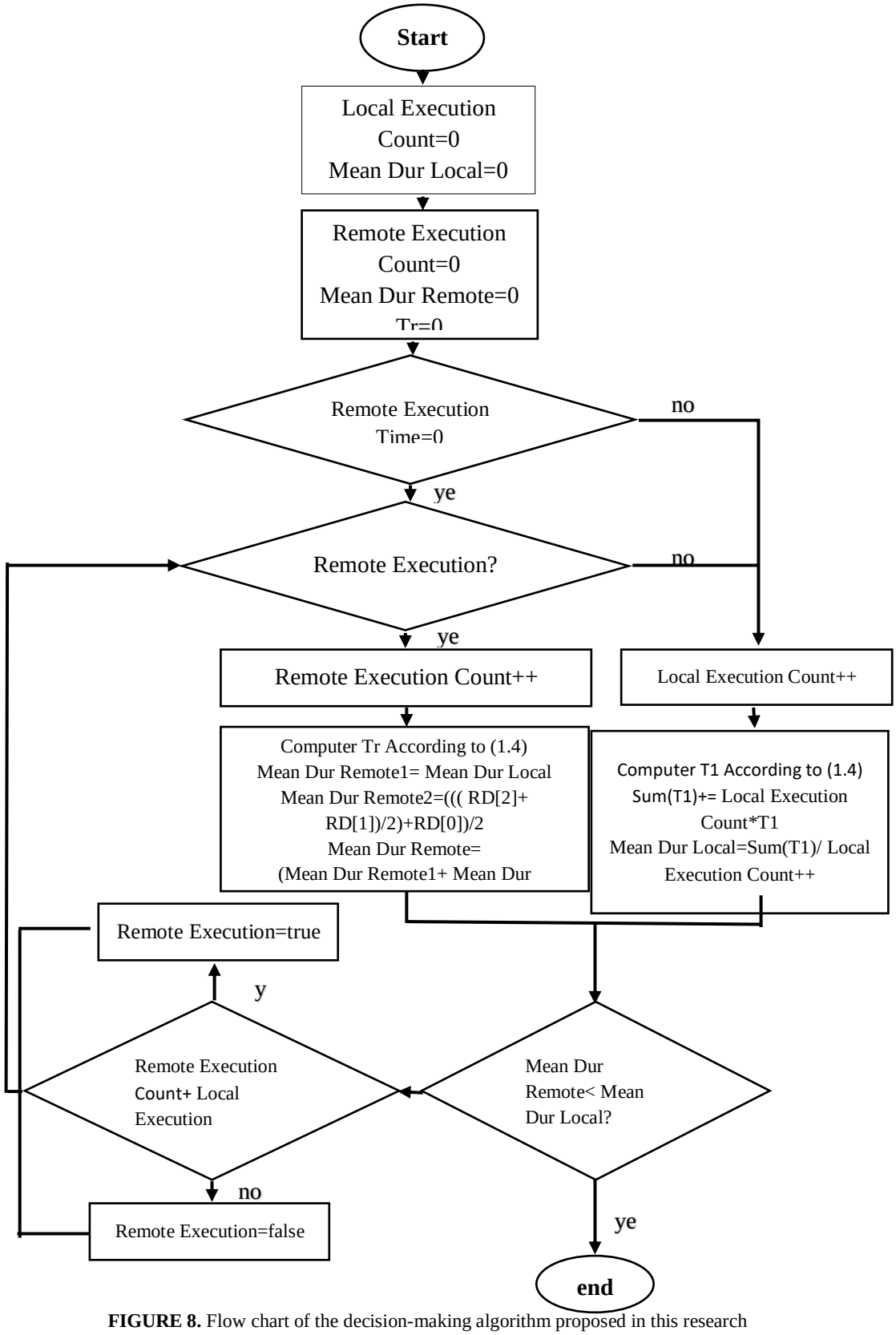

FIGURE 8. Flow chart of the decision-making algorithm proposed in this research

Figure 9. shows the original algorithm before modification [18], as the original algorithm depends only on comparing the allowed delay time (T_{delay}) according to the user's desire with the local implementation time on the mobile device. The fundamental amendment that was made in this research lies by canceling the data transfer time when the task is executed in the same way between two consecutive processes, for example (remote - remote). As for the original algorithm, it adds the data transmission time with each new operation, and the modified algorithm maintains an archive. In previous operations [19], it helps in determining (T_{delay}) based on previous operations, and therefore decision-making and speed up implementation without taking into account calculating the amount of energy consumed due to the lack of appropriate software packages that help in the consumption of energy.

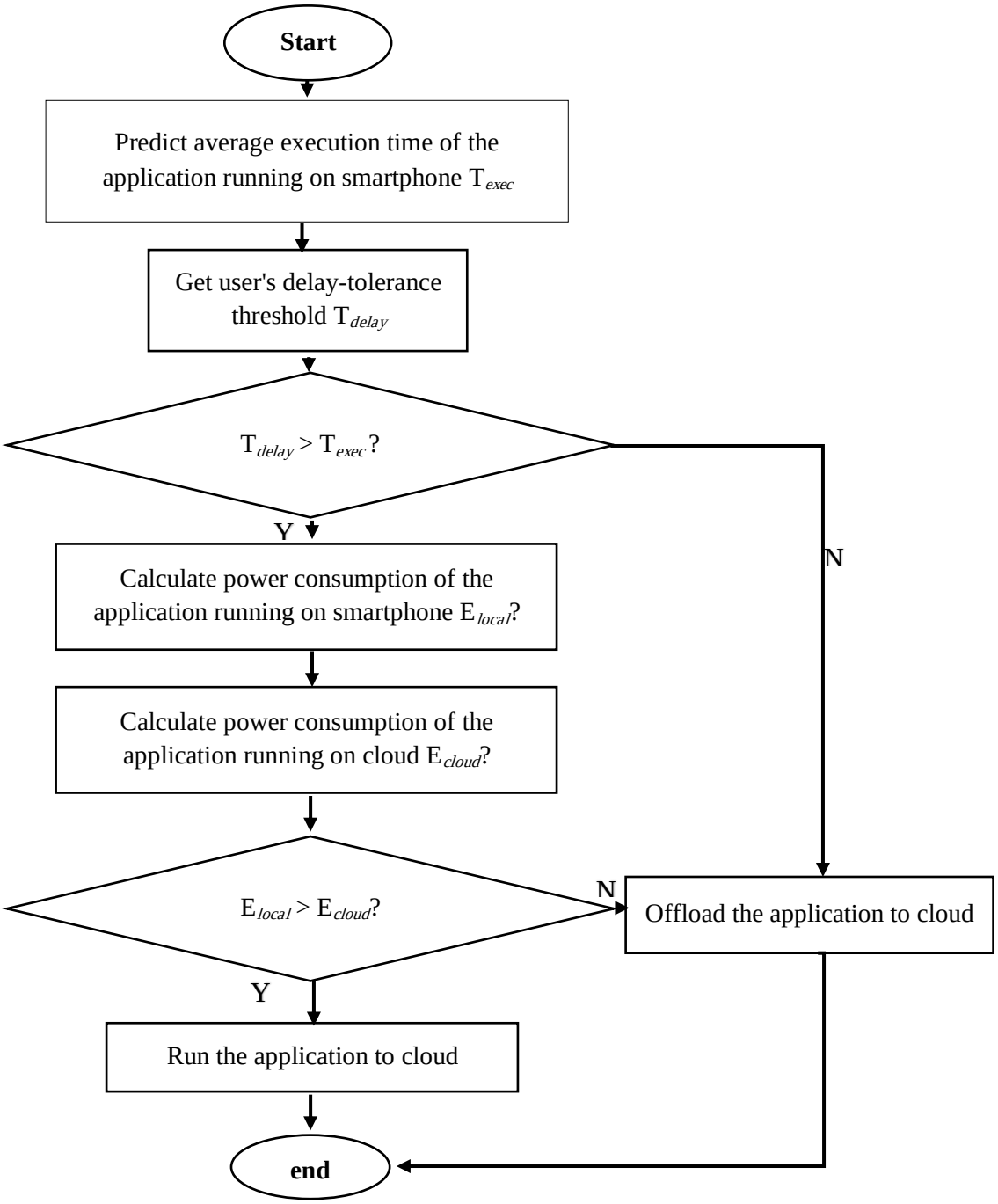

FIGURE 9. Flow chart of the original decision-making algorithm

THE RESULTS OBTAINED UPON IMPLEMENTATION BASED ON THE MODIFIED ALGORITHM

The results obtained will be displayed according to the above three scenarios, and each of the three scenarios will be implemented according to the three strategies:

- Implementation locally.
- Remote implementation.
- Take into account the total implementation time.

The implementation was carried out in each of the three scenarios, considering that implementation is ten times locally, then implementation ten times by distance, then implementation ten times, depending on the algorithm.

The First Scenario: Assigning Implementation of Java Methods to the Cloud

Table 1. shows the average operating time of the solutions obtained when calling the Java method responsible for finding solutions for ten consecutive runs according to the existing implementation cases.

TABLE 1. the implementation times obtained according to the modified algorithm - the first scenario

Misters No.	Solution No.	Average operating time locally(ms)	Operating time, on average, remotely(ms)	Implement dynamically according to the modified algorithm	Time-saving ratio
4	2	21.80	85.40	15.39	29.4%
5	10	148.62	105.18	89.90	39.51%
6	4	2480.71	139.48	135.72	94.53%
7	40	51513.24	1142.03	1161.65	97.74%
8	92	1221922.77	26072.42	26237.26	97.85%

We note from Table 1. that the difference between local implementation and remote implementation increases with the increasing complexity of the issue through an increase from the ministers as shown in Figure 10. We also note from Table 1. that when the issue is implemented ten consecutive times in each of the different cases, the implementation according to the decision-making algorithm helps to accelerate the implementation of the application.

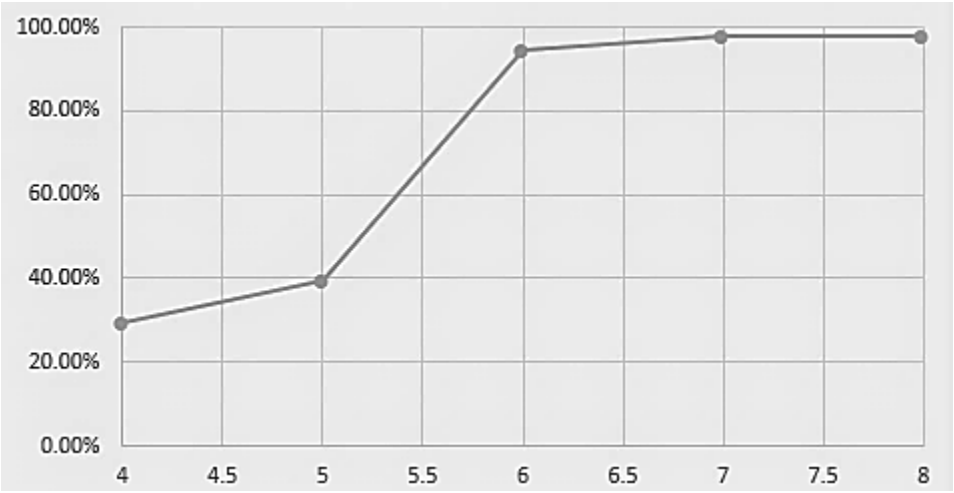

FIGURE 10. Time-saving ratios according to the first scenario

With the increase in the number of ministers, we notice that the calculations are increasing and therefore the application tends to request the implementation of the special method for solving the issue (NQueens) on the cloud server and thus the average implementation time is closer to the time of remote execution.

The Second Scenario: Attribution of the Implementation of the Children of the Written Language (C + + / C), Which Is Included in Android Applications.

Table 2. shows the average operating time of the result of implementing one of the (C++) methods for ten consecutive runs according to the existing implementation cases.

TABLE 2. The implementation times obtained according to the modified algorithm - the second scenario

Misters No.	Average operating time locally(ms)	Operating time, on average, remotely(ms)	Implement dynamically according to the modified algorithm
1	6.25	133.35	6.66
2	6.47	112.34	6.79
3	6.44	104.05	6.56
4	6.21	99.69	6.73
5	6.2	92.49	6.59
6	6.40	83.23	6.40
7	6.32	87.83	6.35
8	6.35	84.72	6.38
9	6.47	82.18	6.49
10	6.41	80.86	6.47

We note from Table 2. that the local execution time is smaller than the time required for remote implementation because the volume of data that needs to be transferred is very small and there is a lost time each time to set up the connection and delay time on the network and therefore the time savings ratio is zero to implementation the local. When implementing the previous method ten times in a row and dynamically, we notice that the algorithm chooses to implement locally in all the runs that are being implemented.

The Third Scenario: Assigning Tasks (CUDA) Included in Android Applications

Table 3. shows the average operating time of the result, the process of multiplying the two arrays on the GPU for ten consecutive runs according to the existing implementation states.

TABLE 3. The implementation times obtained according to the modified algorithm - the third scenario

Misters No.	Average operating time locally(ms)	Operating time, on average, remotely(ms)	Implement dynamically according to the modified algorithm
1	3128.44	240.24	237.87
2	3122.79	257.77	207.60
3	3125.63	241.33	201.20
4	3127.78	233.51	204.85
5	3128.59	238.72	199.65
6	3128.41	237.13	203.06
7	3136.08	238.06	205.57
8	3137.29	236.10	201.81
9	3147.56	230.23	197.69
10	3143.40	231.33	195.70

We notice from the previous table that the remote operating time is smaller than the time required for local execution because the calculations here are more complicated. When implementing the previous method ten times in a row and dynamically, we notice that the algorithm chose to implement remotely in all the operations that are being implemented and the time-saving ratio for local implementation after ten consecutive runs (93. 77%).

COMPARE THE RESULTS WITH THE ORIGINAL DECISION-MAKING ALGORITHM

When implementing locally, the same implementation times obtained will be obtained in competition with simple differences that do not affect the results. Therefore, in this section, the results obtained upon implementation will be dynamically inserted according to the modified algorithm obtained in Table 1. And Table 2. and Table 3. according to the first scenario (the issue of the eight ministers) and the results obtained according to the original algorithm and the comparison of results as shown in Joule (4) on the understanding that (Tdelay = 1000).

TABLE 4. implementation times obtained according to the first scenario

Misters No	Solution No	Implement dynamically according to the basic algorithm	Implement dynamically according to the modified algorithm
4	2	85.40	15.39
5	10	90.91	89.90
6	4	142.12	135.72
7	40	51513.24	1161.65
8	92	1221922.77	26237.26

CONCLUSION

The main difference between the two algorithms lies in the fact that the modified algorithm does not calculate the cost of data transmission when the implementation method is not different, for example (remote - remote). As for the original algorithm, the cost of transport will be calculated at each new implementation. We note that the original algorithm performs remote execution when the matter size is small as in the case of (4 ministers) because it compares with the input threshold from the user while the modified algorithm performs local execution, and therefore we note that there is a time savings of (70.01 ms). When increasing the complexity of the problem, we find that the original algorithm will reach an implementation time that is greater than the threshold entered by the user, and therefore the decision, in this case, will be local compliance, and from here we note the big difference in the time of exemption between the two cases.

ACKNOWLEDGMENTS

I extend my thanks and love to everyone who helped me in this research so that it would be fruitful in the path of science and the future in sustainable development.

In particular, my colleagues at: Al-Furat Al-Awsat Technical University - Technical College Al Mussaib/ Department of Electrical Power Technology Engineering.

REFERENCES

1. Alahmar, Haeder Talib Mahde. "Random Task Scheduler Algorithms as a Comparison and Access to the Best to Use in Real Time. International Journal of Scientific & Engineering Research. 4, April 2019 , 10, pp. 522-527.
2. Compare the Color Digital Image with JPEG 2000 and JPEG with Re-Size by Developing new Algorithm. Annals of the Romanian Society for Cell Biology. 06, May 08, 2021, 25, pp. 10697 - 10718.
3. Speech Recognition by Improving the Performance of Algorithms Used in Discrimination. International Journal of Computer Science & Information Technology. February 1, 2019, Vol. 1, pp. 19-29.
4. Amazon. Overview of Amazon Web Services AWS Whitepaper. [book auth.] AWS. Overview of Amazon Web Services . West Seattle : Amazon Web Services, 2019, p. 78.
5. Butler, Brandon. Gartner: Cloud putting crimp in traditional software, hardware sales. Network World. 13 JUL, 2012.

6. CloneCloud: elastic execution between mobile device and cloud. Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, Ashwin Patti. Salzburg Austria : Association for Computing Machinery, 2011. EuroSys '11: Sixth EuroSys Conference 2011.
7. MAUI: making smartphones last longer with code offload. Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, Paramvir Bahl. 2010. MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services. pp. 49–62.
8. Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Han Qi. Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. The Journal of Supercomputing. August 2, 2013, pp. 84-103.
9. Feng Xia, Fangwei Ding, Jie Li, Xiangjie Kong, Laurence T. Yang & Jianhua Ma. Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. Springer, Inf Syst Front. 1, October 20, 2013, 29, pp. 95-111.
10. Fernando A. M. Trinta, Masum Z. Hasan,Jose N. de Souza. Enhancing Offloading Systems with Smart Decisions, Adaptive Monitoring, and Mobility Support. hindawi. Apr 21, 2019, pp. 1-18.
11. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. Guo, Songtao, et al. San Francisco, CA, USA : IEEE, 2016. IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications. pp. 1-9.
12. Irshad Ahmed Abbasi, Alwalid Bashier Gism Elseed, Sikandar Ali , Zahid Anwar, Qasim Rajpoot, and Maria Riaz. TAMEC: Trusted Augmented Mobile Execution on Cloud . Hindawi. Mar 08, 2021, pp. 1-8.
13. Juan Fang, Jiamei Shi, Shuaibing Lu , Mengyuan Zhang and Zhiyuan Ye. Efficient computation offloading strategies for mobile cloud computing. Micromachines. 12 2021, pp. 2-17.
14. Kumar, Karthik, and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? Computer. 2010, pp. 51-56.
15. Ms.Gayathri M R, Prof K. Srinivas. A Survey on Mobile Cloud Computing Architecture, Applications and Challenges. International Journal of Scientific Research Engineering & Technology. September 2014, pp. 1013-1021.
16. Muhammad Shiraz, Abdullah Gani, Rashid Hafeez Khokhar, Rajkumar Buyya,. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. IEEE Communications Surveys & Tutorials . November 29, 2012, pp. 1294 - 1313.
17. A dynamic programming offloading algorithm for mobile cloud computing. Shahzad, Haleh and Szymanski, Ted H. Vancouver, BC, Canada : IEEE, 2016. 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). pp. 1-5.
18. An Effective Dynamic Programming Offloading Algorithm in Mobile Cloud Computing System. Yanchen Liu, Myung J. Lee. Istanbul, Turkey : IEEE, 6-9 April 2014. 2014 IEEE Wireless Communications and Networking Conference (WCNC). pp. 1868-1873.
19. Alahmar, Dr.Haeder Talib Mahde. Edge Detection Methods in Scientific Cellular Photographs by Means of Analytical Assessment. Asian Journal of Information Technology. 5, 2020, pp. 95 - 101.