OPTIMIZATION OF SMART MOBILE DEVICE WORK TIME USING COMPOSITE OFFLOAD DECISION ALGORITHMIC APPROACH

Sridhar S K

[1]Research Scholar , RajaRajeswari College of Engineering, Bengaluru.
Assistant Professor, Department of Computer Science and Engineering,
Ballari Institute of Technology and Management, Ballari,
Affliated to Visvesvaraya Technological University,
Belagavi, Karnataka, India
sridhar@bitm.edu.in

Dr. S. Jeyabalan

[2]Professor, Department of Computer Science and Engineering,
RajaRajeswari College of Engineering, Bengaluru.
Affliated to Visvesvaraya Technological University,
Belagavi, Karnataka, India
amutharajj@rrce.org

Dr. Amutharaj J

[3]Professor & Head, Department of Information Science and Engineering,
RajaRajeswari College of Engineering, Bengaluru.
Affliated to Visvesvaraya Technological University,
Belagavi, Karnataka, India
amutharajj@rrce.org

Abstract

Today, Most of the popular mobile devices are well equipped with Wi-Fi access mechanism with powerful hardware components and software applications installed by which they are virtually capable of acting as server to some of possible request from neighbor mobile client with in the short range of network. The proposed work aims to combine the intelligence with powerful smart mobile device capability to relish the quality of the mobile computing service by proposing a novel design of proper composite offload decision algorithm (CODA) which integrates the workload meta data with local device parameters to make a composite decision, either to offload data over Wi-Fi to another registered device active in local mobile cloud (LMC) or perform the local execution on current device. The LMC need not necessarily use the internet to execute the request from client device. The LMC is an inter connection of devices where each device can play the role of client or server depending on their battery strength and device status ability to offer or receive an execution service. The implementation results achieved are evident to show the minimized device battery life usage and progressive increase in the overall work time of a mobile that participate in the CODA framework.

***Keywords*:** Local mobile cloud, Client side look up cache, Report generation, ID3 offload decision, Server side look up cache, Direct & Indirect Result Transfer, Mean battery life, Mean execution time measure, Increased device work time.

1. The Main Text

The rapid advancement in mobile device communications and machine learning has paved the way to identify research gaps in the Local mobile cloud computing (LMCC) model. The current model lacks efficient content caching mechanism at client end; there is no proper co-operation among multiple edge servers. The necessity of deep learning for data caching in edge node communication is required to reason out the cache prediction problem and acquire the ability to predict the popularity of content cache data [9]. The integration of

unsupervised learning, deep learning, and optimization algorithms to edge caching is a catalyst to maximize the data access efficiency [16]. Efficient network traffic can be regulated and operational network is maintained by combination of LTE and Wireless LAN using Software Defined Network controller's intelligent approach with dwell time prediction [7]. The utilization of both cellular and operator owned Wi-Fi resources simultaneously have impact on user decision and bid can achieve runtime offload [6]. The mobile device task waiting time and the service time on the edge servers, response time and transmission time are to be considered for data offload mechanism [8]. With the continuous improvement in the design and development of artificial intelligence approach, more tasks can be allotted to servers with increased mobility and heavy computing capabilities, there by transmission energy consumption can be reduced [10]. Wi-Fi data offloading algorithm based on network traffic awareness algorithm can greatly increase user satisfaction [4]. Artificial Intelligence ensures accurate decisions, quick adaptability and required portability by reducing the margin of error probability [11]. An effective estimation of task execution time and its service node location enables optimized data offload [14]. The favoring output by artificial intelligence with a bundle of experiences, and powerful models, can automate the decision-making process to improvise accuracy of mobile data offload decision, which suits all stages of the offloading process [11]. The potential local mobile clouds constituted from large set of mobile computing systems are the future for seamless computation [21]. The combination of multimedia quality adaptation and content sharing mechanisms can save energy at mobile client devices [20]. Computational intensive applications can take the advantage of several offloading techniques as compared to the multimedia application where massive data has to be moved to the remote server for execution that takes more communication time [19]. The machine training required to estimate model that predict the execution costs through static analysis can synthesize a decision model that eliminates the non essential components from active servers [13]. The technology that enables hybrid applications to be built with code transfer that run on heterogeneous operating systems can reduce work time required from developers, as the same code is run on a both mobile device and cloud[15]. These related work encouraged us to incorporate intelligence in content caching, offload decision making and server device selection for quick and efficient task execution thereby increasing the overall mobile work time.

2. Problem Statement

To optimize the overall smart mobile device work time by incorporating the content caching mechanism at client end, scheduling low residual energy mobile client devices on server for execution using an intelligent learning algorithm to automate the data offload decision on client side, maintaining minimal number of edge computing nodes to save local device energy and communication overhead.

3. Proposed Plan

The entire proposed plan begins with compute-intensive workload selection on mobile client device. Once the user submits workload to CODA, it follows a sequence of procedural activity to make an appropriate data offload decision to increase the overall mobile device work time. The CODA considers several considerations such as, mobile client device status, wireless network connection status, selection of mobile server and its device status information. The device profiling gathers the status information of mobile device in terms of current battery life, processor core count, physical & logical memory available for use, current processor load and wireless connection status for communication. The CODA assures security aspect of LMC with pre registration process which collects all the necessary details of mobile device owner who wish to participate in Wi-Fi offload scheme. The CODA application framework is made up of following 8 different modules as shown in Table 1.

1	Compute Workload Selection Module - [CWS]
2	Client Side Cache Search Module - [CSCS]
3	Report Generation Module - [RG]
4	ID3 Decision Tree Module - [DT]
5	Server Side Cache Search Module – [SSCS]
6	Idle Device Selection Module – [IDS]
7	Direct Result Transfer Module – [DRT]
8	Indirect Result Transfer Module - [IRT]

Table 1. Modules of CODA Framework

The Highlight of proposed CODA application framework is the implementation of content caching at client and server side, usage of supervised learning algorithm to manage popular data on cache and hence make efficient data offload decision to relish the capability of smart mobiles.

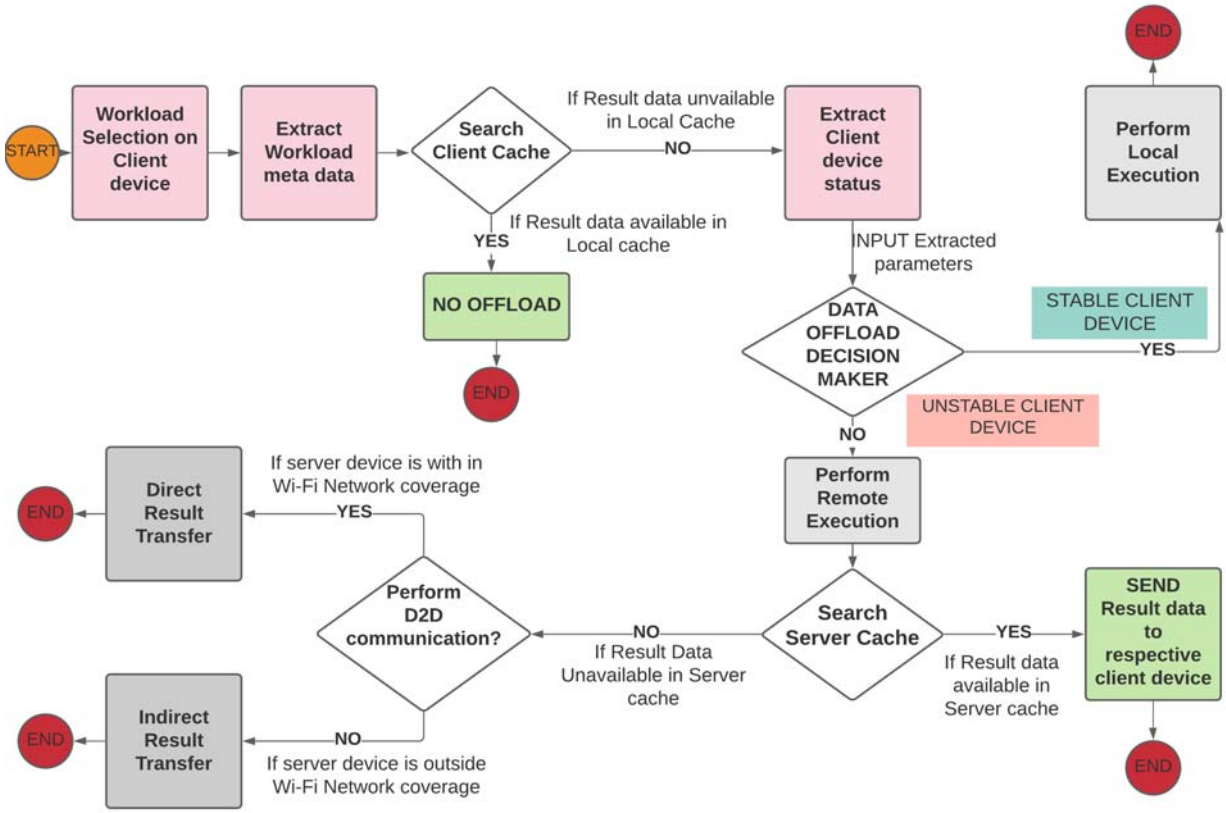

Fig. 1. Proposed CODA Architecture

4. Mathematical Model

A comparative analysis has been carried out between ANN back propagation, ID3 decision tree and Naïve Bayes classifier machine learning algorithms. On performing around 2000 iterations with random Train and Test Samples, it is observed that the mean accuracy value attainment is at 89%, 92.91% and 35% respectively.

Since the research data input is factored data set, the ID3 decision-making algorithm mathematical model is preferred to make an optimized data offload decision to increase the mobile device work time. The work time is a measure of the device active period and capability of task execution.

4.1. *Entropy*

It is a measure of the amount of variability in the dataset S.

$$E = \left[-\left(\frac{Pd}{Pd+Nd}\right) * \log2\left(\frac{Pd}{Pd+Nd}\right)\right] + \left[-\left(\frac{Nd}{Pd+Nd}\right) * \log2\left(\frac{Nd}{Pd+Nd}\right)\right] \quad (1)$$

Where 'P_d' represent number of Positive instances, 'N_d' represent number of Negative instances and 'P_d+ N_d' represent total samples in the given training data set S.

4.2. *Information Gain*

It is used to decide the ordering of attributes in the decision tree.

$$IG = \text{Entropy (parent)} - [\text{Average Entropy (children)}] \quad (2)$$

The attribute with minimum entropy or maximum information gain is treated as best attribute and can be the root node of the decision tree. ID3 algorithm is able to provide several measurements such as accuracy, recall, confusion matrix and precision for a given input data set.

4.3. *Turnaround time (TAT) measure of a client device:*

The Turnaround time is the time taken from workload submission to its execution for a client device. The formulae to evaluate the TAT, we have the following:

When the selected server device is within the wireless network coverage area,

$$Mclient_{TAT} := LCT + SCT + OffloadTime + RET + RRT \quad (3)$$

When the selected server device is out of the wireless network coverage area,

$$Mclient_{TAT} := LCT + SCT + OffloadTime + RET + RRT + CUT \quad (4)$$

In Eq. (3) and (4), LCT is Local Cache Search Time, SCT is Server Cache Search Time, Offload Time is time taken to transfer workload from client to LMC server, RET is remote execution time, RRT is remote server response time taken to respond to process the input workload, CUT is the cloud upload time in case remote device is out of wireless network coverage area. On experimental evaluation, it is found that the Client device turnaround time (TAT) is slightly more when the server device is out of wireless network coverage area, since cloud upload and download operations are incorporated. But as a part of CODA framework these data transfer time seems negligible and provides overall increased work time for mobile device after task offload.

5. Proposed Algorithm

5.1. *Proposed Prime Algorithm*

Algorithm: Composite Offload Decision Algorithm

Input: Mobile device parameter data set.

Output: Offload decision.

```
While (Workload Selection == True):
    If (Pre-Check-Up-1 _CSCSM) == True:
      Then "Result found in Client side cache: No Offload"
            (Quick Work Done & Time Saved)
    Else:
      Generate CODA Report by Parameter extraction (RG)
      Apply ID3 decision tree algorithm on coda report (DT)
      If (ID3_Offload decision) == True:
        If (Pre-Check-Up-2_SSCSM) == True:
          Then "Result found in LMC server cache: No D2D"
                (Quick work done & time saved)
        Else:
          Select an idle device with good battery - (IDS)
          If (Device is within network coverage):
            Then invoke direct result transfer - (DRT)
          Else:
            Invoke indirect result transfer - (IRT)
      Else:
        The client is stable & local execution is feasible
```

Algorithm 1. Prime CODA algorithm

5.2. *Elements of Proposed Prime Algorithm*

The CODA has four major design modules namely:

5.2.1. *Client side cache search (CSCS) module:*

When a user makes up his/her mind to execute a particular computation workload and submits it to a CODA, it primarily performs the user registration and then authentication through login credentials. Only on successful authentication, the user is allowed use benefits of CODA application. The CSCS Module gets activated on

selection of computation workload to check whether the desired result linked with workload, is already available on the local cache or not. If available, no execution required. Otherwise, either local or remote execution may be needed. It is also responsible to manage client side cache table by applying the local cache data replacement policy whose goal is to retain the result of previously offloaded files to save work time of the device. It uses locally executed file first (LEFF) policy and then if required it replaces file with larger X-factor value. The Algorithm 1.1 results in better data offload decision on every next attempt. The CSCS algorithm is as follows,

Algorithm: Client Side Cache Search
Input: Workload Parameter data
Output: Workload Result available/unavailable

BEGIN
Check *Client Side Cache Table* for Matched Event with Specified Workload.
If *similar Event registered* in the Table:
Then *Corresponding Result is Available* and *NO OFFLOAD* required.

If *No similar Event registered* in the Table:
Invoke *CODA Report Generation module*
Record the*Remote Execution Time on Offload.*
Record and Link the output file received to input workload.
Update Client Side Cache Table for TASK.
END

Algorithm 1.1. Client side cache search algorithm

File Name	File Size (Bytes)	Prev. Offloaded?	RET (Sec)	LET (Sec)	X factor	Result
CI2.py	512	NO	0.0	16.49	31.04	Victim
CI4.py	586	NO	0.0	20.34	28.81	Victim
CI8.py	610	YES	26.28	0	11.60	Retain
LC.py	1026	YES	33.05	0	15.52	Retain
HC.txt	5086	YES	300.66	0	8.45	Retain

Table 2. Client side cache attributes for replacement policy

The cache replacement decision is based on the X-factor value, which is evaluated as follows,

$$X - Factor = FileSize/[(2 * RET) + (1 * LET)] \quad (5)$$

Where, RET and LET represent remote execution time and local execution time in seconds. The victim indicates the file and its location to be replaced with new popular file accessed for computation. Higher the X-Factor value, there is a more chance of being a victim of replacement.

5.2.2. *Report generation (RG) module:*
The RG module is activated on no similar event encounter in client side cache search and begins the collection of all the necessary details of workload to be executed and device status information using device-profiling method. The algorithm for parameter extraction from selected workload to be executed and current client device is as follows:

Algorithm: Report Generation
Input: Workload and Client device feature
Output: Profiled workload and system data display

BEGIN
Extract *FILE_NAME, FILE_TYPE* and FILE_SIZE to run.
Extract CURRENT_BATTERY_PERCENTAGE of the local device.
Extract *CPU_COUNT* (Physical and Logical) of the local device.
Extract *PHYSICAL AND VIRTUAL MEMORY (%)* availability of the Local device.
Extract *DISK STORAGE (%)availability* the Local device.
Extract *Wi-Fi NETWORK STATUS* of Connection.
Invoke ID3 decision Tree Module.
END

Algorithm 1.2. Device status report generation algorithm

The work flow of report generation includes the participant registration and login authentication. On success the report generation module is applied as directed by local cache on selection of compute intensive workload to produce the analysis report which act as input to make an efficient data offload decision through machine learning approach. The mandatory user registration and login provides safety and security in terms of critical data communication.

5.2.3. *ID3 Decision Tree Module:*
Once the list of profiled parameter values are obtained in relation with submitted workload and the mobile client device status. The preferred ML algorithm for to make offload decision is:

Algorithm: ID3 Decision Tree
Input: Generated Report from RT Module
Output: Decision on Local or Remote or No Execution necessity for selected workload.
BEGIN
Calculate *entropy* for data of parameter extraction report.
For every data feature
Calculate *entropy* for all its possible categorical values.
Calculate *information gain* for the data feature.
Bring out the feature with *maximum information gain*.
Iterate until the desired data offload decision tree obtained
END

Algorithm 1.3. ID3 decision tree algorithm

The CODA Report contains 7 prime input features extracted from client mobile device system:

FSIZE	CUR_BAT	CPU_CORE	CPU_PER_AVAIL
0/1	L/M/H	D/Q/O	0/1
PHY_MEM	**PHY_DISK**	**WIFI_SIG**	**PHY_MEM**
0/1	0/1	0/1	0/1

Table 3. Input attribute list for ID3 decision tree algorithm

Where, The FSIZE has two possible input values namely '0' to indicate small size file (< 1MB) and '1' to indicate large size file (> 1MB). The CUR_BAT represent current battery status which has 3 possible input values namely L/M//H indicating Low (0>L<=30%), Medium (30>L<=60) and High (60<L<=100) battery status in percentage. The CPU_CORE represent number of client processor core information with 3 possible input values namely D/Q/O indicating Dual (2 cores), Quad (4 cores) or Octa (8 cores). The CPU_PER, PHY_MEM, PHY_DISK represent Processor, Memory and Disk storage availability in percentage with two possible inputs namely '0' to indicate less than or equal to 50% availability and '1' to indicate more than 50% availability. The Final and crucial attribute on deciding data offload is Wi-Fi which has two possible input values namely '0' to indicate poor network signal and '1' to indicate strong network signal respectively. The total training samples required can be evaluated as 2*2*2*2*2*3*3 = 288 samples. Once training is completed with sample Table 4, a report generation module can be applied on any mobile device with different configuration and specification to extract seven major attributes.

FSIZE	CUR_BAT	CPU_CORE	CPU_PER	PHY_MEM	PHY_DISK	WIFI_SIG	DOD
0	L	Dual	1	1	1	0	0
0	L	Dual	0	0	1	0	0
0	M	Quad	1	1	1	0	0
0	H	Octa	1	1	0	0	0
0	H	Quad	1	0	1	0	0
1	L	Dual	0	1	1	0	0
0	L	Dual	1	0	0	1	1
1	L	Quad	1	0	0	1	1
1	H	Dual	1	1	0	1	0
1	M	Octa	1	0	1	1	0
1	M	Quad	1	1	0	1	0
0	M	Dual	0	0	0	1	1
1	H	Octa	1	1	1	1	0
1	M	Octa	0	0	0	1	1

Table 4. Normalized Training sample input data set for ID3 decision algorithm

Based on training, ID3 accepts 7 input attributes, performs analysis and produces decision on whether to offload the data to another device for execution or not. This decision-making capability increases the overall mobile work time and reduces the regular battery usage on specific workload. The default threshold values are initial set values, which represent minimum battery life, minimum memory requirement, acceptable file size and minimum processor load for the selected compute intensive workload to get executed locally in the mobile device. Here it is assumed the 50% of the available hardware capacity. The acceptable file size is about 1MB.
The precise ID3 decision tree output is as follows:

{'WIFI_SIG': {0: 0, 1: {'CUR_BAT': {'H': {'PHY_MEM': {0: {'CPU_CORE': {'Dual': 1, 'Octa': {'FSIZE': {0: 0, 1: {'CPU_PER': {0: 1, 1: 0}}}}, 'Quad': {'CPU_PER': {0: 1, 1: {'PHY_DISK': {0: 1, 1: 0}}}}}, 1: {'FSIZE': {0: 0, 1: {'CPU_CORE': {'Dual': {'CPU_PER': {0: 0, 1: {'PHY_DISK': {0: 0, 1: 1}}}}, 'Octa': 0, 'Quad': {'CPU_PER': {0: {'PHY_DISK': {0: 1, 1: 0}}, 1: 0}}}}}}}}, 'L': {'CPU_CORE': {'Dual': 1, 'Octa': {'FSIZE': {0: {'CPU_PER': {0: 1, 1: {'PHY_MEM': {0: {'PHY_DISK': {0: 1, 1: 0}}, 1: 0}}}}, 1: 1}}, 'Quad': 1}}, 'M': {'PHY_MEM': {0: {'CPU_PER': {0: 1, 1: {'PHY_DISK': {0: 1, 1: {'CPU_CORE': {'Dual': 1, 'Octa': 0, 'Quad': {'FSIZE': {0: 0, 1: 1}}}}}}}}, 1: {'CPU_PER': {0: {'CPU_CORE': {'Dual': {'FSIZE': {0: {'PHY_DISK': {0: 1, 1: 0}}, 1: 1}}, 'Octa': 0, 'Quad': {'FSIZE': {0: 0, 1: {'PHY_DISK': {0: 1, 1: 0}}}}}}, 1: 0}}}}}}}}}

5.2.4. *Server Side Cache Search Module:*
Based on ID3 algorithm offload decision, the server side cache search Algorithm 1.4 gets activated to find the match for a client selected workload. If match found, then the respective output file along with overall transfer time is sent back to client device. Otherwise, an idle device with strong battery status, which is present within the network coverage, is opted for service execution followed by direct result transfer mechanism. If opted mobile device is out of coverage for any reason, then it is the responsibility of the opted device user to upload the output file to cloud using an LTE or any other network and inform the same to the server system. The server is responsible to update the LMC cache with most popular data based on the number of access from different registered mobile devices that work in LMC.

Algorithm: Server Side Cache Search

Input: Generated Report from RT Module

Output: Decision on Data availability in server cache or Device selection for remote execution.

```
BEGIN
  Check Server Side Cache Table for Matched Event with Specified Workload.
  If similar Event registered in the Table:
    Then "Result found in LMC server cache: No D2D"
    END
  If No similar Event registered in the Table:
    Then Select an idle device with good battery (IDS)
      If (Device is within network coverage):
        Then invoke direct result transfer - (DRT)
        Return the Output file with Total Offload Time.
      Else:
        Invoke indirect result transfer - (IRT)
```

Record the *Remote Execution Time on Offload.*
Return the *Output file* with *remote run time.*
Update *Server Side Cache Table for TASK.*
END

Algorithm 1.4. Server side cache search algorithm

FILE_NAME	FILE_SIZE	Remote ET	X-factor	Decision
C1-2.py	512	16.49	15.52	Victim
CI-4.py	586	20.34	14.40	Victim
CI-8.py	610	26.28	23.21	Retain
LC.py	1026	33.05	31.05	Retain
HC.py	5086	300.66	16.91	Retain

Table 5. Server side cache attributes for replacement policy

The server cache data replacement decision is based on the X-factor value, which is evaluated as follows,

$$X - Factor = FileSize/(2 * RET) \quad (6)$$

Smaller the X-Factor value, there is a more chance of being a victim of replacement.

6. Implementation

The experimental Setup involved 80 mobile devices which includes smart phones powered by android v9 operating system with following specification: 4 GB RAM, Octa-core 2.01GHz processor, Kernel v4, 64GB internal storage and battery 4000mAh each and laptop computers powered by 64 bit windows 10 operating system having specification as follows:

1. Intel Core i3 @ 2.20GHz, 4GB RAM and 2.29Ah Battery.
2. Intel Core i5 @ 2.30GHz, 8 GB RAM and 2.29Ah Battery.
3. Intel Core i7 @ 2.30GHz, 16 GB RAM and 3.3Ah Battery

The Python programming language is used to develop a CODA mobile application using several special packages such as kivy, psutil, os, time and so on. The Python version 3.8 interpreter is used execute python application with pycharm IDE. The Buildozer tool is used to convert python kivy application into android application and then run on real android device. The real devices were connected over Wi-Fi without internet and an exchange of data is performed using special network packages such as scikit, accuracy metrics, socket, plyer and wifi available in python.

Suppose the User Chosen File: D:\\Files\sj.py		
FEATURE	**VALUE**	**NORMAL-RANGE**
File Size	3.24 MB	0 - 1 MB
System Battery	61 %	30 – 60 – 100 %
System CPU Count	6	02 – 04 - 08
System CPU Available	97.1	50 – 100 %
System Virtual Memory	44.6	50 – 100 %
System Disk Space	99.7%	50 – 100 %
System Network Signal	Excellent	-67 to -50 dBm

Table 6. Client device real time resource usages monitor report

The successful Wi-Fi connection between data off loader and data receiver is established. The short range data offload mechanism implementation has been carried out for different task files that keep CPU busy at different number of cores to compute run time. The combination of compute intensive and data intensive task are executed for 1000 rounds to get the better outcome of real experiment conducted. The visualization of results and their meaningful representation is interpreted in the next section.

7. Results and Discussion

On experimenting with three compute intensive code files namely CI-2.py, CI-4.py and CI-8.py of size in KB which can keep 2, 4 and 8 processor cores at maximum load, it is observed that mobile client device consume more battery life on local execution as shown in Table 7 and Figure 2.

File_Name	File_Size	On Local Device	On Wi-Fi Offload to LMC
CI-2.py	512B	0.68%	0.13%
CI-4.py	586B	0.78%	0.16%
CI-8.py	610B	0.82%	0.17%

Table 7. Real time Mean battery life spent on client device

Instead, if the files are offloaded to another registered mobile device in LMC via Wi-Fi for processing will reduce the burden on CPU of mobile client and its battery power consumption is low by keeping offload time at negligible quantity as less than 0.02 seconds as depicted in table 8 and 9 respectively.

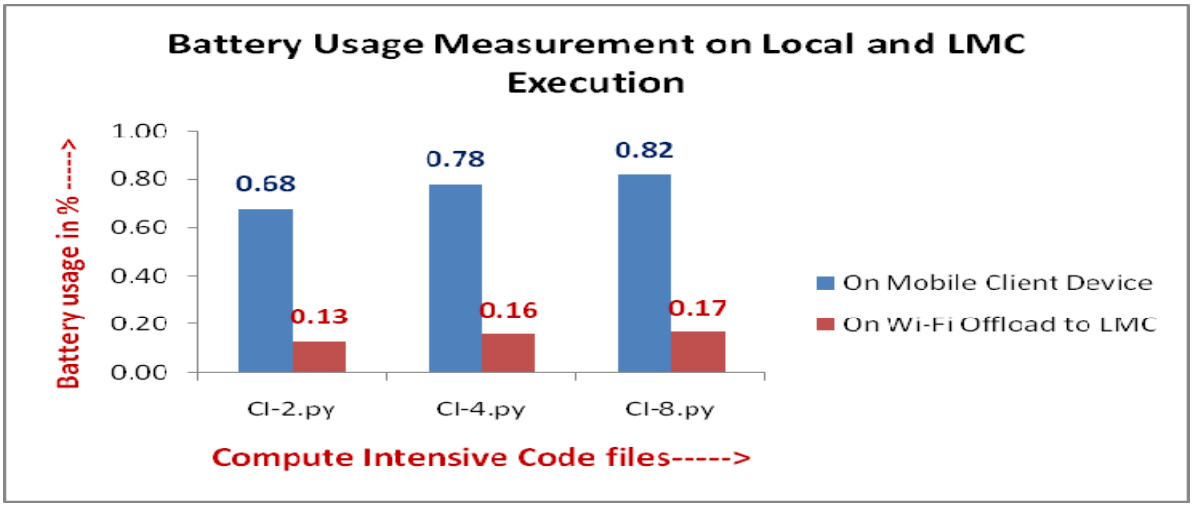

Fig. 2. Battery Usage on Local and LMC Execution

File_Name	File_Size	Link Speed-in Mbps	Mean DTT in (seconds)
CI-2.py	512B	96	0.016 s
CI-4.py	586B	96	0.018 s
CI-8.py	610B	96	0.02 s

Table 8. Real time mean data transfer measure (DTT) for compute intensive files

MEAN REAL EXECUTION TIME MEASURE			
File_Name	File_Size	Local CPU Time	On Wi-Fi Offload to LMC
CI-2.py	512B	16.49 s	16.51 s
CI-4.py	586B	20.32 s	20.34 s
CI-8.py	610B	26.28 s	26.30 s

Table 9. Real time mean execution time measure for compute intensive files

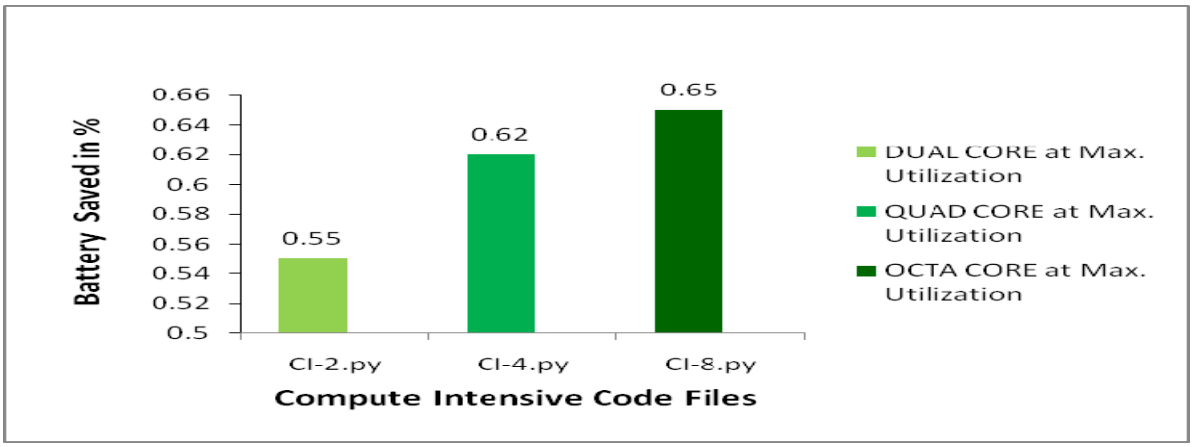

Fig. 3. Battery Conservation on Wi-Fi Offload

The battery power saved on Wi-Fi data Offload mechanism in LMC of each compute intensive files is at 0.55, 0.62 and 0.65 percent computed with overall real time discharge measure during offload between the physical mobile devices as depicted in Figure 3.

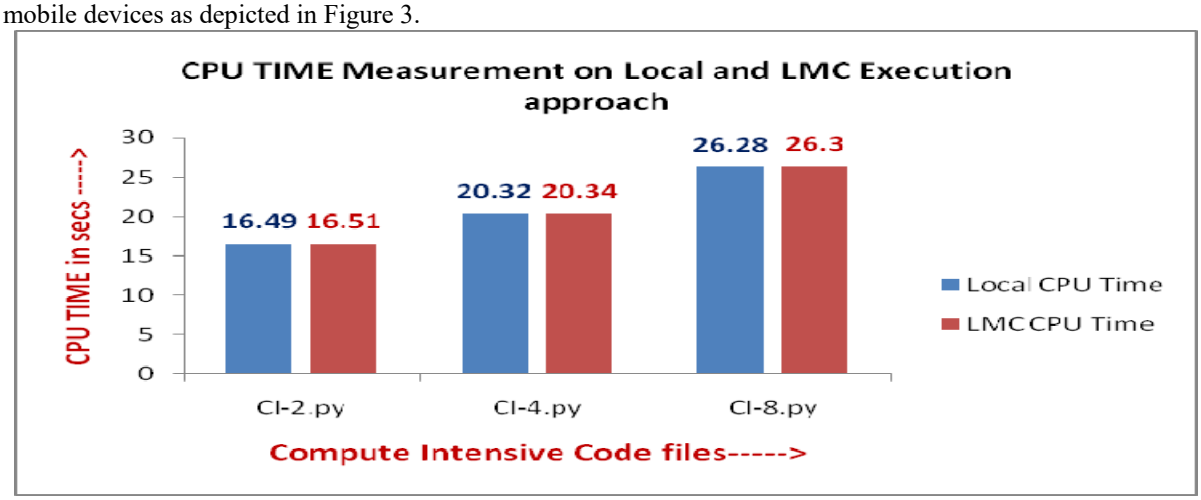

Fig. 4. CPU run time comparison graph

From Figure 4, it is evident that the compute intensive workload will normally make use of more processor time there by discharging more battery power of the device in which it gets executed. Therefore offloading them to high-end mobile device with good battery strength is quite capable of executing this code and delivers the processed data as result to requested mobile client device. Since the data transfer time of compute intensive files is low, they can minimize power consumption on client device and is appropriate to make a Wi-Fi data offload decision to LMC by considering several critical factors of devices, which is graphed for better communication.

File_Name	File_Size	Link Speed-in Mbps	Data Transfer Time in seconds
DI-1.mp4	59.91MB	96	12.6 s
DI.-2mp4	258.91MB	96	84.38 s
DI-3.mp4	504.58MB	96	168.85 s
DI-4.mp4	1146.88MB	96	366.46 s

Table 10. Real time mean data transfer time measure for data intensive files

On Wi-Fi Offloading the four data intensive files of varying size from 59.91 MB to 1.12 GB where each of them running at 96 Mbps Link speed on network has taken data transfer time ranging from 12.6 seconds to 366.46 second (around 6 minutes) as shown in Table 10. It is observed that the Wi-Fi Offload mechanism works better on compute intensive files which are reasonably medium size files leading to increased battery conservation which in turn leads to increase the overall mobile device work time. The benefit of using Wi-Fi network is low building cost and no worry of charges and helpful to reduce the load in cellular network. Wi-Fi is well integrated with 4G ecosystem and can reduce burden of fast data traffic growth to attract new users with additional services. The Wi-Fi data offload can be user initiated or automatic depending on the data sensitivity and level of confidentiality in view of security aspects.

8. Conclusion

The proposed work has been implemented to adopt a composite offload decision algorithm (CODA) framework. It is evident that there is a mark able reduction in the overall time taken to execute compute intensive applications in local mobile cloud (LMC) and send back results to the requested mobile device client. It is noticeable with the results that if the data computation time value is larger compared to the data communication time value then the data offloading will definitely consume less energy. It is utmost useful when the compute intensive workload execution makes local mobile device client to consume more time and energy. Moreover any mobile device registered can act as both client and server depending on its device status information. Content caching at client and server end, offload decision making capability managed by supervised learning algorithms and `data transfer mechanisms are the key factors the proposed framework. The real time experiment can be extended with improvised machine learning intelligence approach to make devices to learn on own with data set recorded on every process transaction and improvise the Wi-Fi data offload decision-making capability by updating the look up table periodically. Optimal number of nodes can be added to this real time experiment to graph the efficiency of CODA in saving battery power and increasing overall work ability of smart mobile devices.

Acknowledgement

My sincere gratitude to Dr. Amutharaj J for being my research supervisor, compassionate and he is the one who continuously encourage and directs to gain a deeper knowledge at the research findings and upgrade to next progressive level by setting up naïve objectives of professional research and career in scope.

References

[1] Wenhao Fan, Junting Han, Le Yao, Fan Wu, Yuan'an Liu, "Latency-energy optimization for joint WiFi and cellular offloading in mobile edge computing networks", Computer Networks, Volume 181, 2020, 107570, ISSN1389-1286, https://doi.org/10.1016/j.comnet.2020.107570.

[2] NuntanutBhooanusas, Sok-Ian Sou, Kai-Chun Cheng, "Satisfaction-based Dynamic Bandwidth Reallocation for multipath mobile data offloading, Computer Networks", Volume 185, 2021, 107594, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2020.107594.

[3] Thomas Rausch, Alexander Rashed, SchahramDustdar, Optimized container scheduling for data-intensive serverless edge computing, Future Generation Computer Systems, Volume 114, 2021, Pages 259-271, ISSN 0167-739X, https://doi.org/10.1016/j.future.2020.07.017.

[4] Shen Han, "Congestion-aware WiFi offload algorithm for 5G heterogeneous wireless networks", Computer Communications, Volume 164, 2020, Pages 69-76, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2020.10.006.

[5] TasnimAbar, AbderrezakRachedi, Asma ben Letaifa, Philippe Fabian, Sadok el Asmi, "FellowMe Cache: Fog Computing approach to enhance (QoE) in Internet of Vehicles", Future Generation Computer Systems, Volume 113, 2020, Pages 170-182, ISSN 0167-739X, https://doi.org/10.1016/j.future.2020.06.026.

[6] Yi Zhao, Ke Xu, YifengZhong, Xiang-Yang Li, Ning Wang, Hui Su, Meng Shen, Ziwei Li, "Incentive mechanisms for mobile data offloading through operator-owned WiFi access points", Computer Networks, Volume 174, 2020, 107226, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2020.107226.

[7] SudhaAnbalagan, Dhananjay Kumar, Mercy Faustina J, Gunasekaran Raja, Waleed Ejaz, Ali Kashif Bashir, "SDN-assisted efficient LTE-WiFi aggregation in next generation IoT networks", Future Generation Computer Systems, Volume 107, 2020, Pages 898-908, ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.12.013.

[8] Samah A. Zakaryia, Safaa A. Ahmed, Mohamed K. Hussein, "Evolutionary offloading in an edge environment", Egyptian Informatics Journal, 2020, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2020.09.003.

[9] Wang, Yantong; Friderikos, Vasilis. 2020. "A Survey of Deep Learning for Data Caching in Edge Network" *Informatics* 7, no. 4: 43. https://doi.org/10.3390/informatics7040043.

[10] GonçaloCarvalho, Bruno Cabral, Vasco Pereira, Jorge Bernardino, "Computation offloading in Edge Computing environments using Artificial Intelligence techniques", Engineering Applications of Artificial Intelligence, Volume 95, 2020, 103840, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2020.103840.

[11] Yuan Cheng, "Edge caching and computing in 5G for mobile augmented reality and haptic internet", Computer Communications, Volume 158, 2020, Pages 24-31, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2020.04.054.

[12] Sridhar S K, Dr. Amutharaj J, Dr. S. Vijayanand, "Survey: Enhancing Energy Proficiency in Smart Mobile Devices using Composite Offload Decision Alogorithms", Journal of Critical Reviews, Vol. 7, Issue 04, 2020, pp. 2671-2682, ISSN:2394-5125.

[13] Xing Chen, Jiaqing Chen, Bichun Liu, Yun Ma, Ying Zhang, HaoZhong, "AndroidOff:Offloading android application based on cost estimation", Journal of Systems and Software, Volume 158, 2019, 110418, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2019.110418.

[14] Wenda Tang, Xuan Zhao, WajidRafique, Lianyong Qi, Wanchun Dou, Qiang Ni, "An offloading method using decentralized P2P-enabled mobile edge servers in edge computing", Journal of Systems Architecture, Volume 94, 2019, Pages 1-13, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2019.02.001.

[15] Nawrocki, Piotr &Śnieżyński, Bartłomiej&Słojewski, Hubert. (2019). "Adaptable mobile cloud computing environment with code transfer based on machine learning". Pervasive and Mobile Computing. 57. 49-63. 10.1016/j.pmcj.2019.05.001.

[16] Z. Chang, L. Lei, Z. Zhou, S. Mao and T. Ristaniemi, "Learn to Cache: Machine Learning for Network Edge Caching in the Big Data Era," in *IEEE Wireless Communications*, vol. 25, no. 3, pp. 28-35, JUNE 2018, doi: 10.1109/MWC.2018.1700317.

[17] C. Zhong, M. C. Gursoy and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," 2018 52nd Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, 2018, pp. 1-6, doi: 10.1109/CISS.2018.8362276.

[18] Khadija Akherfi, Micheal Gerndt, Hamid Harroud, "Mobile cloud computing for computation offloading: Issues and challenges", Open access, Applied Computing and Informatics (Dec-2016), www.sciencedirect.com.http://dx.doi.org/10.1016/j.aci.2016.11.002, page No. 1-16.

[19] Farhan Azmat Ali, Pieter Simoens, Tim Verbelen, Piet Demeester, Bart Dhoedt, "Mobile device power models for energy efficient dynamic offloading at runtime", Journal of Systems and Software, Volume 113, 2016, Pages 173-187, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2015.11.042.

[20] Ruiqi Ding, Cristina HavaMuntean, Gabriel-Miro Muntean, "Energy-efficient device-differentiated cooperative adaptive multimedia delivery solution in wireless networks", Journal of Network and Computer Applications, Volume 58, 2015, Pages 194-207, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2015.09.006.

[21] Niroshinie Fernando, Seng W. Loke, WennyRahayu, "Mobile cloud computing: A survey", Future Generation Computer Systems, Volume 29, Issue 1, 2013, Pages 84-106, ISSN 0167-739X, https://doi.org/10.1016/j.future.2012.05.023.