International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

# An Efficient Model for Prediction Based Optimization in Mobile Cloud Task Offloading in TORO

**G. Amirthayogam[1*], B. Thanikaivel[2], G.Renuga Devi[3], J. Venkatarangan[4], M. Sujaritha[5], N. Kumaran[6]**

**Submitted**: 19/01/2024 **Revised**: 28/02/2024 **Accepted**: 05/03/2024

***Abstract:*** Mobile Cloud Computing (MCC) is a booming field with the high usage of smart devices to overcome the on-demand resource availability to improve performance. The humans are migrating faster for their work but due to the portable mobile devices cause resource migrating problem which degrades the performance of MCC. The smart devices are capable of operating various kinds of day-to-day applications such E-Commerce, Banking, Education Healthcare etc. In this paper, Task Offloading and Resource Optimization (TORO) architecture is proposed to handle the migration problem by optimizing the resource with supporting operations such as resource demand prediction, cloudlet resource discovery with reliability, task partitioning, task scheduling and task offloading. The implementation and evaluation are carried by simulating the proposed TORO architecture and comparing with the existing FDCO algorithm and mCloud model. Further, the evaluation results depict the proposed TORO architecture executes the tasks faster with multiple operations integrated together to provide better performance when compared with the existing FDCO algorithm and mCloud model.

***Keywords:*** *On-demand prediction, Resource discovery, Reliable Resource, context-awareness, Partitioning, Scheduling, Task offloading, Resource allocation, Resource optimization*

## 1. Introduction

Mobile Cloud Computing (MCC) is the new technology which took the world under the people's hand with smart devices such as watches, mobiles, laptop, and tablets etc. Smart devices are capable of operating various kinds of day-to-day applications such E-Commerce, Banking, Education Healthcare etc. The smart devices also called as mobile devices which are highly portable in nature and limited with computing resources. Further, to execute the complex application in mobile device it uses the MCC technology [1] which dynamic in nature to overcome the resource limitation as shown in Fig.1 where user or client request the service from local mobile cloud called cloudlet, if the resource availability in cloudlet is not enough it uses the public cloud resource.

Moreover, providing resources to users is not a matter in MCC but it should not degrade the execution performance of task [2] by unreliable resource and delay caused during the resource allocation to the tasks. So, this paper we consider the performance degradation problem in MCC during resource allocation and optimization process. Further we also addressed the additional challenges in various operations like task offloading, task partitioning, scheduling task and resource reallocation to make the resource optimization an effective in MCC.

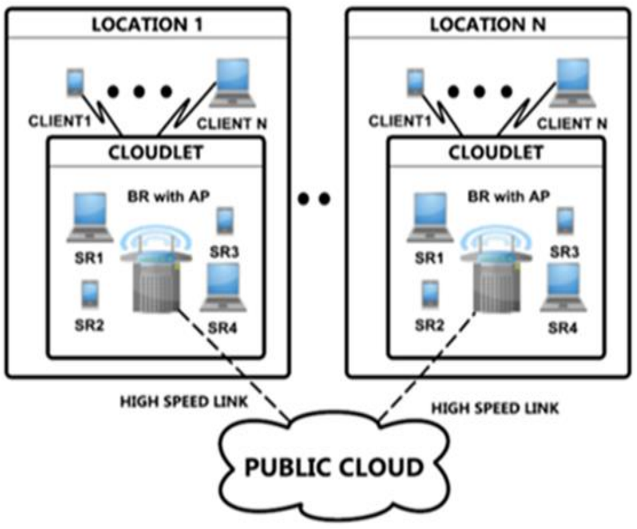

**Fig. 1** General representation of MCC

[1]*Associate Professor, Department of Information Technology, Department of Information Technology, Hindustan Institute of Technology and Science (Deemed to be University), Chennai, Tamil Nadu, India-603103.* ***Email: amir.yogam@gmail.com***
[2]*Assistant Professor, Department of Information Technology, Hindustan Institute of Technology and Science (Deemed to be University), Chennai, Tamil Nadu, India-603103.* ***Email: thanikaicse@gmail.com***
[3]*Assistant Professor, Department of Computer Science and Engineering,, P.S.R.Engineering College,Sivakasi, Tamil Nadu, India. Email:renugag0304@gmail.com*
[4]*Assistant professor, Department of Computer Science and Design, St Martin's Engineering College, Secunderbad, Telangana, India-500100,* ***Email: venkatarangan1986@gmail.com***
[5]*Professor, Sri Krishna College of Engineering and Technology, Kuniyamuthur, Tamil Nadu 641008, India.* ***Email: sujaritham@skcet.ac.in***
[6]*Assistant professor, Department of Mathematics, Vel Tech Rangarajan Dr.Sagunthala R &D Institute of science and technology, Avadi, Chennai - 600062.****Email: nkumaran@veltech.edu.in***

Further MCC resource allocation operation processes the tasks from various users are offloaded for computation where offloading process need to identify the reliable resources. Moreover, identifying the reliable resource and taking the decision [3] is a challenging factor which affects the performance of MCC. The size of task need to be considered during task offloading process due to the bandwidth limitation and traffic overhead where the big size task got delayed and small size task is offloaded faster [4]. Therefore, fixing task size for offload process is a challenging factor which reduces the transmission delay by monitoring the mobile cloud network also needs to be addressed for effective task offloading. Next, continuous execution of the tasks [5] is needed for computing environment which reduces the overall execution time and make resource utilization an effective one, which is to be addressed for performance improvement in MCC.

The virtual resource required by task need to be formed as collateral which provide the computing environment called Virtual Machine (VM). The temporary formation of VM is challenging thing which is handled by hypervisor [6]. The hypervisor is the heart of the MCC which identifies the suitable resource to form VM for a task execution; it is like mapping of resource to the task. The challenge faced by hypervisor is to take decision for identifying the VM resources which is to be mapped for tasks to provide high performance MCC. Moreover, the allocated resource to tasks also be migrated and become unavailable, for such case the allocated resources are also need to be monitored which is a challenging operation. Thus, we handle the reallocation process for migrated and unavailable resources which is mapped to task which is challenging operation for resource optimization. The handling resource migration and mobile user migration is also a challenging operation which is to be monitored continuously. Thus, the MCC performance improvements by resource allocation and optimization are complex during task offloading which are the inter linked process addressed in this paper by proposed Task Offloading and Resource Optimization (TORO) architecture.

In this paper we contribute, we integrate the task offloading, resource allocation and optimization process in TORO architecture to reduce the complexity of task execution process and the literatures about the existing works are analysed and reviews are discussed in Sect. 2. In Section.3 the proposed TORO architecture. The mathematical model and algorithms are developed in Sect. 4. The implementation and evaluation are shown in Section 5. Finally, Section. 6 concludes the proposed research work.

## 2.Literature Review

Smart Virtual Machine [7] offloads the task in cloud environment which improves the efficiency of offloading by applying decision model to select the better performance level. But collecting the context information in dynamic environment changes frequently and unreliable, needs more updates to take better decision for optimized resource allocation. Hermes et al., in [8] proposed an application task tree graph offloading technique which constructs the subset of problem to reduce the complexity. The handling of subset problem for time-sensitive application needs reliability and fastness. A cooperative runtime offloading decision algorithm [9, 10] optimize the task offloading decision making process by applying the machine learning technique combined with genetic algorithm in cooperative manner to reduce resources utilization and time. Moreover, decision making using context information moves frequently which is complex to be handled. Mobile Offloading System [11] is a mobility-aware with centralized network controller with seamless offloading operations in wireless network. The distributed of network controller in offloading process can be applied more effectively in mobile cloud network. A fast hybrid multi-site computation offloading [12] technique with Optimized Multi-site Branch & Bound algorithm provides optimal solution by reduced search space during convergence and the most suited nearby partitioning of application in timely manner is provided by Optimized Multi-site PSO algorithm [34]. Further authors stated that decision making along with available bandwidth will be designed.

Agent-based MCC framework [13] coordinates the tasks and mobile devices to generate result faster. Further, Dynamic Programming after filtering algorithm in agent optimizes the offloading process by taking decision to response quickly. Further it is stated that offloading the complicated tasks based on relationship will be designed. A heterogeneity-aware task allocation algorithm [14] uses MAC controller to take offload decision to minimize the execution time in mobile ad hoc cloud. Moreover, analyzing the MAC information with unreliable context information is not easy in dynamic environment. A fully distributed computation offloading algorithm [15] takes multi-user decision to minimize execution time by identifying the Nash Equilibrium Point without any information exchange in homogenous distributed cloud network. The authors stated that new algorithm will be designed to take offloading decision in heterogeneous cloud environment with minimum information exchange. Full offloading method [16] improves the performance of task computation for small sized task and big size task where the offloading decision is taken based on priority and completion time. But minimizing the communication cost of the big size task is complicated which is to handled. Advanced decision model [17] integrates the RED algorithm to take better decision in congested network where request is dropped based on the moving average weight in offloading process to effectively utilize the resource. In fast migrating environment calculating the moving average weight is tedious and the advance predication technique can improve the effectiveness. Network Cloud Mapping [18] offloads the task with required QoS level in heterogeneous cloud environment to improve the performance and efficiency of computation by minimize the resource and communication cost. The decision making process during offloading process is complex and can be reduced. Context Sensitive Model for Offloading System (CoSMOS) [19] takes decision by considering the execution time and energy consumption during task offloading to improve performance, resource usage and energy consumption. Further authors planned to enhance the CoSMOS with more parameters to take decision in wide range of environment.

A Software Defined Task Offloading (SDTO) [20] optimize the task offloading process by transforming the offloading process into two sub-problems such as task placement and resource allocation problem where the task duration and computational resource utilizations are minimized. Moreover, dynamic computation environment decision need to be taken frequently with varying distributed computing resource where SDTO task placement and resource allocation are handled separately which lacks in reliability and performance. A data driven MEC management optimization framework [21] optimizes the task placement by choosing the computing edge to minimize the computation cost and improve QoS. The optimization in task placement is carried by collecting the information from all mobile edges and orchestration process in dynamic environment is complex. A better method can be developed to collect computation and communication resource information with frequent updating and orchestrated. A multi objective genetic algorithm [22] handles the resource scheduling problem by using K-mean clustering method combined with genetic algorithm by considering priority, energy consumption and load balancing. The resource required by the user is estimated by the Radial Basis Function Neural Network. Authors stated that the multi objective genetic algorithm will be enhanced to improve the performance for large scale problem with complex constraint. A cross-layer resource allocation model [23] handles the resource allocation problem jointly with elastic service scaling and beam formation with remote radio head (RRH) considered as a mixed integer nonlinear programming for optimization with combinatorial NP-hard. The low complexity shaping and pruning algorithm developed in the model identifies the RRH set for user request. Moreover, pruning technique can be further enhanced to improve the QoS of the cloud network with low complexity. Mobile Edge Computing- Wireless Power Transmission [24] optimizes the task offloading process with multi-antenna AP to minimize the latency using Time Division Multiple Access for multi-user. But the processing capacity or computation resource availability should meet the number of user requests received in Mobile edge with dropping the request in the intermediate state of computation. Further predicting the number requests received on time and making the resource available without request drop in offloading process is complex.

An Edge Computing IoT (ECIoT) model [25] improves the performance of task computation by applying the lyapunov dynamic stochastic optimization by decomposing the problem into three sub-problems. The stochastic optimization process combines the neural network for executing the decomposed problem with better decision to minimize response time. The run time problem classifying is a tedious process which is to be handled. A User Level Online Offloading framework (ULOOF) [26] takes decision by predicting the execution time and bandwidth of the offloading task. The authors stated that accuracy of prediction [35] can be improved by applying the supervised learning technique for the mobility behaviour of the user. A Student Project Allocation model [27] finds the better match between user and resources to provide close optimal performance by minimizing the latency of task offloaded in distributed environment. Further optimal solution can generated for worst case offloaded tasks. The task offloading technique [28] minimizes the execution time with two algorithms: Minimum Offloading Time for Mobile device (MOTM) and Minimum Execution Time for Cloud data centre (METC). The MOTM selects the offloading link and METC selects the physical resource as per application requested. The classification of application is better choice but the classification of task for single application and multiple applications makes computation parallel thereby performance of MCC will be improved. Cloudlet assisted ad-hoc mobile cloud model [29] jointly optimizes the resource utilization and QoS. The stackelberg game formulated finds the optimal solution based on the current context information. Thus, the context information of the node in MCC needs to be updated frequently to take better offloading decision.

A hybrid genetic algorithm - Ant Colony Optimization algorithm [30] minimizes the mean completion time along with Queue based Decision Maker algorithm which reduces the drop rate and balances the cloudlet load with maximize the resource utilization of task offloading process. A Genetic Algorithm for Hybrid Granularity Resource Optimization algorithm [31] minimizes the computation cost using chromosome where relationship between tasks are found and scheduled to execute in timely manner. Further, optimization can be improved by implementing the parallelization technique to execute the related task. A decision theoretical approach [32] improves the throughput and reduces the latency using the channel state information (CSI) problem in MCC. The rate allocation scheme is applied during task offloading to decide the transmission rate of user requested by applying the decision-making technique. Further, the Wifi channel transmission is found to be effective and reduces the computation cost. A resource scheduling method [33] uses particle swarm optimization (PSO) algorithm to find the optimal solution using oriented matrix with fast convergence speed which improves the performance of MCC by offloading the task with requested time delay. Further, PSO algorithm makes proximity optimal solution which can be improved further to utilize the resource effectively.

From the literatures discussed, the performance improvement in MCC can be made by integrated way of handling task offloading, resource allocation and optimization. The optimization of resource can be done by reallocation with continuous monitoring of allocated resources and tasks. Further, the context information about the mobile cloud network and user are collected to efficiently carry the resource optimization process. The task execution can also help the resource optimization process by parallelizing the task to be executed in sequential order.

.

## 3. Proposed Methodology

In this paper, the proposed Task Offloading and Resource Optimization TORO architecture represented in Fig. 2 contains three layers where each layer perform a specific set of operations which overcomes the resource management complexity in MCC. The layers along with the operational components are discussed below.

### *3.1 Mobile Cloud Users (MCU)*

MCU are peoples who use mobile devices with various applications to carry the daily tasks such as online ticket booking, Mobile banking, Email, Online data storage space, Geographical Positioning System (GPS) and Image processing. The applications listed are few but MCU makes the mobile devices to operator all kinds of applications with limited resource capacity. So, MCU requests the MCC system to efficient handle all kind of application execution.

### *3.2 Multiple Request Handler (MRH)*

In MCC, MRH receives requests from the mobile cloud users and arrange them to provide mobile cloud service where this N numbers of services are needed to be provided on-demand. Further, the requests are forwarded to the Cloudlet Resource Discovery Procedure (CRDP) as discussed in Algorithm 1 which checks the resource availability of the requested service and intimates them to arrangement the resources in advance.

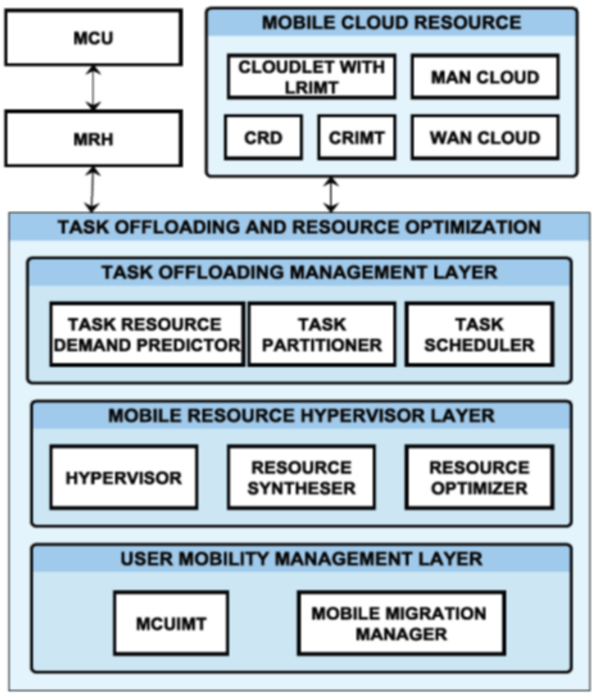

**Fig 1:** Task Offloading and Resource Optimization Architecture

*3.3 Mobile Cloudlet Resource (MCR)*

MCRL contains MCC resources which are discovered dynamically and user to provide resources on-demand. In MCRL, cloudlet is a main element which has local pool of resources, Metropolitan Area Network (MAN) which is private cloud where the resource are available in multiple location of single management, Wide Area Network (WAN) which is public cloud network where resource are available globally by connecting multi resource management service. The resource for the user requests are allocated from MCRL which is based on the availability and providing uninterrupted service using MAN and WAN.

*3.3.1 Cloudlet*

The local mobile computing resource are pooled together to form a Cloudlet which is connected and accessed by Access Point (AP). Each cloudlet contain Base Resource (BR) and Additional Shared Resource (ASR) where the BR is fixed which can be upgraded later. The ASR is dynamically collected resources from different resource provider with the particular range of cloudlet distance which is discussed in Algorithm 1. In each cloudlet, the proposed TORO architecture maintains a Local Resource Index Management Table (LRIMT) data structure which makes resource to access in distributed manner, stores the cloudlet resource information which are newly discovered and forwarding the migrated resource information from one LRIMT to other LRIMT. The update is also carried in the Centre Resource Index Management Table (CRIMT) which accesses the resource in centralized manner for carrying better optimization in resource reallocation.

*3.3.2 MAN Cloud*

MAN cloud is a private cloud in different geographical area where clouds are formed by interconnecting multiple local cloud resources. The MAN cloud resource is used only when the local cloud resource are unavailable or in overloaded condition. But the MAN cloud resources are costlier and degrade the MCC performance due to long distance and mostly not considered for resource allocation.

*3.3.3 WAN Cloud*

WAN cloud is a public cloud which is available globally with interconnection of giant cloud systems. The cloudlets and MAN cloud resource are not sufficient to provide the service WAN cloud resource is used. The WAN cloud resources are more costly and degrade the MCC performance it is not considered mostly in the proposed TORO architecture.

*3.3.4 CRIMT*

CRIMT data structure interconnects all LRIMT in cloudlets to make centralized access of resources and further reallocate the resource in fast migrating environment. The CRIMT monitors the resources with allocated task and its movement where the task execution stages are monitored. When the final stage of task execution is identified, CRIMT will not take of care of further resource reallocation step for that task thereby concentrating the other task to execute in optimized way.

## 3.4 Task Management Layer (TML)

TML is a part of TORO architecture as shown in Fig.2. TML perform three operations such as task offloading with Task Resource Demand Predictor (TRDP), Task partitioning and Task scheduling. The components of TML along with its operations are discussed as follow.

*3.4.1 TRDP*

The scheduled request from MRH is forwarded for offloading process where the task from MRH first received by TRDP. Here, decision is taken using TRDP where the requested resource by task is available in the MCR if not available initiate the dynamic resource discovery process using Algorithm 1.

*3.4.2. Task scheduler*

Task scheduler receives all partitioned task in group and scheduled them in the multi-level queue based on task completion time where each level of queue handle different GoT.

***3.5 Mobile Hypervisor Layer (MHL)***

MHL handles the Resource allocation is an important process in TORO architecture. Further MHL decomposed into three operations such as hypervisor (or) mapping, VM synthesizer and VM Optimizer.

*3.5.1 Hypervisor*

Hypervisor is an important component in TORO architecture where decision is taken to allocate which resource to task and freeze that resource as discussed in Algorithm 5. The decision is taken by considering proximity between task (X, Y, Z) and resource (X, Y, Z). Further, resource identified to map the task is collateral in to VM.

*3.5.2. VM Synthesizer*

VM synthesizer analyzes the allocated resource to VM whether the proximity between tasks is high. The resource collateral with VM is checked for proximity using proximity checker which is discussed in Algorithm 2. Thus for the resources proximity checker identify which resource is to be reallocated and that resource detail along with VM is repeated.

*3.5.3 VM Optimizer*

VM Optimizer performs resource reallocation where the VM resources reported by it pauses the task execution by the VM synthesizer as discussed in Algorithm 6. Further, during reallocation process VM optimizer reallocate the non-proximitized resource where to newly proximitized resource and restart the task execution of the GoT or BoT.

### *3.6 Mobility Management Layer (MML)*

MML perform two operations such as MCU information collection and storing MCU information. The components of MML are described as follow.

*3.6.1 Mobile Cloud User Information Management Table (MCUIMT)*

MCUIMT has data structure to store MCU information which is collected frequently in dynamic environment which also includes the newly modified resource requirement of task.

*3.6.2 Mobile Migration Manager (MMM)*

MMM monitors the MCU movement and also gets the information about the new requirement for already executing task to reallocate the resources as discussed in Algorithm 6. The continuous monitoring is carried for all users who have provide the mobile cloud service and frequent information are collected for a period of time around for 30 sec.

## 4. Mathematical model and Algorithms

### 4.1. Resource Discovery Process (RDP)

The users (U) and their requests (R) are paired as P (U, R) = {p ($u_1$, $r_1$), p ($u_2$, $r_2$)….. p ($u_N$, $r_K$)} in MCC. In Proposed TORO architecture, local cloudlet C is collected from different independent cloudlet resource {$C_1$, $C_2$, $C_3$….$C_\infty$} which ranges from 1 to ∞ which depends on the situation. Further, TRDP is carried by checking resource availability (R_Avail) as shown in Eq. (1) to project (∏) R_Avail for each $R_i$ with function *f* in C. Eq. (2) represents the function *f* applied in Eq. (1) where (∃) denotes existence of cloudlet resources (C_Res) in C for $R_i$. The R_Avail has two case shown in Eq. (3) – (4) where the Eq. (3) specify the satisfaction of resource requirement (R_Req) of $R_i$ for computation, further the resource migration process for dynamic resource migration optimization is initiated using DRMP Algorithm 1 which is described in 4.2. The Eq. (4) specifies the insufficient R_Req for $R_i$ and to make enough resource availability in dynamic environment it uses cloudlet resource discover process (CRDP) which is discussed in Algorithm 1.

$$R\_\text{Avail} \Leftarrow \left(\prod_{Re\,s} Ri \xrightarrow{f} \sum_{i=1}^{\infty} Ci\right) \quad (1)$$

$$f\colon (\exists C\_\text{Res}i \rightarrow \textstyle\sum_{i=1}^{\infty} Ci) \quad (2)$$

Case 1: Resource Requirement satisfied.

$$(Rj.R\_\text{Req} \approx R\_\text{Avail}) \rightarrow DRMP() \quad (3)$$

Case 2: Resource Requirement not satisfied.

$$(Rj.R\_\text{Req} \neq R\_\text{Avail}) \rightarrow CRDP() \quad (4)$$

In cloudlet resource discovery process (CRDP), the Eq. (5) represents available C_Res from C in the proposed TORO architecture. Moreover, the CRDP with expand and shrink state where expand state dynamically discovers the ASR within center point CP= 0 to range r=50 meters as shown in Eq. (6). The Eq. (7) denotes the shrink state where the unreliable resource is released from $C_i$ within CP= 0 to r=50 meters to effectively manage the resources.

$$C\_\text{Res} \leftarrow \textstyle\sum_{i=1}^{\infty} C_i$$

$$\text{Expand state: } C_i \leftarrow \int_{r=0}^{50}\left(BR_i + \textstyle\sum_{j=1}^{\infty} ASR_j\right) \quad (6)$$

$$\text{Shrink state: } C_i \leftarrow \int_{r=0}^{50}\left(BR_i - \textstyle\sum_{j=1}^{M} ASR_j\right) \quad (7)$$

### 4.5 Task offloading

The task offloading operation is mentioned in the Eqs. (8) - (10). Eq. (8) denotes the resource allocation process in the form of VM where proximity $\sigma_{(p\text{-}res)}$ of $C_j$ thereby $R_i$ is mapped to

$$C_j..VM_i = \left\{\sigma_{(p-res)} C_j / R_i \xrightarrow{f} C_j\right\}, \quad (8)$$

Eq. (9) mention the single group offloading of similar task S-task$_{Gi}$ to $VM_i$ where execution ($E_j$) takes place. Eq. (10) represents non-similar task offloading NS-task to $VM_j$ for execution ($E_p$) take place.

$$E_j = \left\{S\_task_{G_i} \rightarrow VM_i\right\}, \quad (9)$$

$$E_k = \left\{NS\_task \rightarrow VM_j\right\}. \quad (10)$$

**Algorithm 1: Task offloading process.**

```
Input: VM= {VM1, VM2...VMn}, NS_task, S_task
Output: Execution time (ET_i), OVM= {OVM1, OVM2...OVMk},
Begin Task Offloading process
Get (S_task_i) <-PQueue1 (S_task)
Get (NS_task_i) <-PQueue2(NS_task)
For NS_task_j j=0 to sizeOf (NS_task)
Get (p_res) <-pair (NS_j,C_j)
VM_j<- using p_res_j
Map (VM_j, NS_task_j)
E_j<- ResourceOptimization(run(NS_task_j))
End
For S_task_k k=0 to sizeOf(S_task)
Get (p_res) <-pair (S_k,C_k)
If ((G_id of S_task_k-1)! = (G_id of S_task_k))
VM_k<- using p_res_k
Map (VM_k, S_task_k)
 E_k<- Optimization(run(S_task_k))
Else
Map (VM_k, S_task_k)
E_k<- Optimization(run(S_task_k))
EndIf
```

```
End
End Task Offloading Process
```

Algorithm 5 is developed using Eqs (8-10) where the execution of $S\_task_k$ and $NS\_task_j$ are separately processed in the queue. The p_res for $task_i(x, y)$ position is identified and paired $P(S_i, C_k)$. VM for $task_k$ is created using the p_res of $C_k$. The S_task will be executed sequentially based on same group id (G_id) in same VM and parallel execution is carried by another VM for S_task with different G_id. Moreover, simultaneously parallel execution of NS_task is done. The execution time (E) noted for all $AT_i$. The task offloading in TORO architecture using algorithm 4 executes the tasks sequentially and parallel.

**4.6 Resource optimization process**

Resource optimization for allocated resource in TORO architecture is performed by Eqs. (11-14). Eq. (11) denotes the VM optimization $Opt(VM_i)$ in $VM_i$ where the performance degrades to the required QoS level ($R_i.QoS$). The optimization of VM is done using three cases; in case1, the performance degradation is due to the addition resource requirement of VM for task execution as given in Eq. (12) for optimized Opt_VM. In case 2, optimization for intra cloudlet is performed using Eq. (13) where the performance degradation happens due to mobility of the $VM_i$ resource and the reallocation done to reform Opt_VM. In case 3, optimization for inter cloudlet is performed as mentioned in Eq. (14) where the performance degradation happens due to the mobility of $VM_i$ resource and handover operation is done for $VM_i$ reallocation to different $C_j$.

$$Opt(VM) = \{VM_i \to Opt\_VM_i / P(VM) < R_i.QoS\}. \quad (11)$$

Case 1:

$$P(VM) = \{Opt\_VM_i \leftarrow VM_i + SR\}. \quad (12)$$

Case 2:

$$P(VM) = \{Opt\_VM_i \leftarrow Re\,a\,llocate(VM_i)\}. \quad (13)$$

Case 3:

$$P(VM) = \{Opt\_VM_i \leftarrow Handover(VM_i)\}. \quad (14)$$

Algorithm 6 is developed based on the Eqs. (25-28) where the hypervisor monitors the VM by checking three conditions: additional resource required to execute the task in specified VM and the proximity value (PV) between allocated resource with VM for intra cloud and handle over of VM to inter cloud. In the first condition, it allocates ASR to $VM_i$. In the second condition, it reallocates the $VM_i$ resource. In third condition, handover process is carried out due the user (U) migration to different location after offloading of resources.

**Algorithm 2: Resource Optimization process**

```
Input: single $VM_i$
Output: $Opt\_VM_i$
Begin Optimization
If (VM.performance < $R_i$.QoS_requested level)
If ($VM_i$ requires ASR)
$Opt\_VM_i \leftarrow VM_i + ASR_j$
Elseif ((current $PV_i$ of $VM_i$) > (initial $PV_i$ $VM_i$))
$Opt\_VM_i \leftarrow Reallocate(VM_i)$
//Intracloudlet communication
Elseif ((current $PV_i$ of $VM_i$ & $U_i$)> Threshold_PV
$VM_j$ <- Handover($VM_i$ )
//Intercloudlet communication
EndIf
EndIf
Begin Reallocate
If (($R_k$->$VM_k$)&&($VM_K$->$C_i$))
Foreach j=1 to M
If ($C_i$.cp<=$VM_k$.$ASR_j$(X,Y,Z)<$C_i$.r)
$C_i \leftarrow ASR_j(X,Y,Z)$
Elseif ($C_i$.cp < $VM_k$.$ASR_j$(X,Y,Z) <= $C_i$.r)
$ASR_j$= new ASR
$C_i \leftarrow ASR_j(X,Y,Z)$
Endif
End
Endif
End Reallocate
Begin Handover
If((($C_i$.r<$U_j$(X,Y,Z)<$C_{i+N}$.cp))
MCUIMT $\leftarrow$ ($U_j$(X,Y,Z) near to $C_{i+N}$.LRIMT )
$C_{i+N}.VM_j \leftarrow Reallocate(R_j$ to $C_{i+N})$
End Handover
End Optimization
```

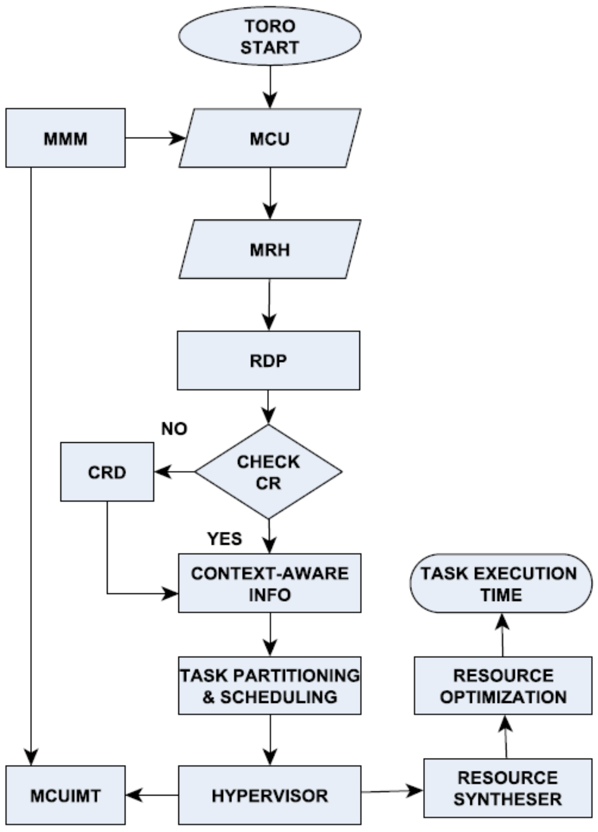

**Fig. 2** TORO Flowchart

The proposed TORO, existing FDCO [15] and mCloud [40] model are simulated with CloudExp simulator [39] for performance comparison. The proposed TORO is simulated as per the data flow shown in Figure 3. The MCU have multiple users who request services from MCC to handle the MCR. After receiving the requests from MCR, the RDP is initially processed by checking the resource availability of requests by Algorithm 1.

| Parameters | Value |
|---|---|
| C_id | Auto |
| N_id | Auto |
| P_core | 2-8 |
| P_MIPS(GB) | 1-3 |
| RAM(GB) | 2-8 |
| Storage(GB) | 8-512 |
| C_range (Meters) | 50(Appro) |
| BW(GB) | 1-2 |
| EL (%) | 0-100% (60 is threshold) |
| SS (level) | 1-6 (3 is threshold) |

**Table 1** LRIMT

In TRDP process, if the resources are sufficiently available it calls the RMM procedure for reliable resource maintenance and if resources are insufficient CRDP procedure is called to discover on-demand resource for requests. The cloudlet parameters as shown in LRIMT Table 1 which collected during discover resources process. The C_id uniquely represents the cloudlet with pool of resource, N_id mentioned denotes the mobile node which provides ASR. The node with energy level above 60% and the signal strength above 3 levels are considered as reliable ASR. The number of processing cores is depicted as P_core and P_MIPS represents the total number of instructions handled. Computational memory is denoted by RAM. The secondary memory to store the data is mentioned as Storage. The C_range denotes the cloudlet covering area. BW represents data rate for task offloading.

| Parameters | Value |
|---|---|
| N_id | Auto |
| P_core | Variable |
| P_MIPS(GB) | Variable |
| RAM(GB) | Variable |
| Storage(GB) | Variable |
| N (x, y, z) | Dynamic |
| BW(GB) | 1-2 |
| EL (%) | 0-100% (60 is threshold) |
| SS (level) | 1-6 (3 is threshold) |

**Table 2** Mobile node

Table2 represents the mobile node parameters which provide ASR. The N_id uniquely identify the mobile node, P_core denotes the number of execution units, P_MIPS represents the number of instructions per cycle, RAM denotes the memory size and Storage represents the secondary memory. The mobile node parameters are variable based on the individual configuration. Further, N (x, y, z) represents the mobile nodes current location which is monitored by MMM process and detailed information is maintained in the MCUIMT. The BW represents the transmission data rate in network. The RMM process monitors the discovered resources for migration by Algorithm 2 to make effective and efficient way of resource allocation and helps the resource optimization by communicating between cloudlets with updating the locality information in LRIMT shown in Table 1 and CRIMT shown in Table 3.

Table 3 parameters are T_Cloudlet which mention the total number of cloudlet available in the TORO system, C_id uniquely denotes the individual cloudlet. T_Node represents the total number of mobile node who provides ASR. The T_P_core depicts the total number of CPU core and T_P_MIPS represent the number of instruction handled. The total computational memory is denoted by RAM.

| Parameters | Value |
|---|---|
| T_Cloudlet | Auto |
| C_id | Auto |
| T_Node | Auto |
| T_P_core | 2-8 |
| T_P_MIPS(GB) | 1-3 |
| T_RAM(GB) | 2-8 |
| T_Storage(GB) | 8-512 |
| T_C_range (Meters) | 50(Appro) |
| T_BW(GB) | 1-2 |

**Table 3** CRIMT

The Storage denotes the capacity of secondary memory. The T_C_range specifies the range of cloudlet area covered. T_BW represents available data rate. Total resource availability on the TORO system cloudlets is accessed using this information. The DRCR process described in Algorithm 3 is implemented for provisioning reliable ASR where the energy level (EL) and signal strength (SS) is to be above the threshold level as described in Table 2. After making the resource availability for the demanded requests, each request task is analyzed by Algorithm 4 for task scheduling and prioritization where IFS value for each request is generated to partition the task in to subtasks. Further, subtasks are grouped by application and operational similarity where similar application tasks are scheduled in S_task queue and tasks without application similarity are separately scheduled in NS_task queue. The S_task and NS_task are multilevel queue, based on the completion time tasks are scheduled in ascending order.

| **Parameters** | **Value** |
|---|---|
| Image (GB) | 10 |
| RAM(GB) | 15 |
| P_MIPS(MHz) | 250 |
| BW(GB) | 1 |
| P_core | 4 |

**Table 4** NEXCUS5 VM

After scheduling the NS_tasks and S_tasks separately in multilevel queue, the resource allocation and offloading process is implemented using Algorithm 5. Moreover, sub-tasks are offloaded in sequence from scheduler to hypervisor for resource allocation where the hypervisor identify the resource which is close to the user task by creating temporary virtual machine (VM) based on the proximity between the user and resource, VM image used is Nexcus5. Table 4 describe the Nexcus 5 parameters where image represent the storage capacity, RAM denotes the primary memory, P_MIPS represent the number of instructions executed in a single cycle frequency, P_core denotes the number of execution unit in VM and finally the BW represents the data transfer rate.

| Parameters | Value |
|---|---|
| C_id | Auto |
| G_id | Auto |
| T_id | Auto |
| VM_id | Auto |
| VM(X, Y,Z) | Dynamic |
| T (X, Y,Z) | Dynamic |

**Table 5*:*** Mapping

Table 5 represents the VM and task mapping parameters. The C_id identify the unique cloudlet. The G_id represents the group id where S_task are mapped to VM, if need parallel execution in done in another VM to execute the tasks in parallel and sequential manner in allocated VM. The S-task and NS-task are uniquely identified using T_id. The NS_task are mapped to separate VM and executed in parallel. The VM_id is used to identify the allocated resource to the tasks. The NS_task and S_task are executed in parallel with different VMs. Moreover, allocated resource to tasks and users are monitored by MMM for periodically update the proximity and reliability information about users maintained in MCUIMT. The resource syntheser takes decision for optimization by checking the proximity distance between resources and users with the help of MCUIMT information. Then the resource syntheser send the command to resource optimizer which is implemented using Algorithm 2 to reallocate the allocated resource of VM with considering the cases specified in the Eq. (11) – (15) where the proximity distance between the user and resource are low. Finally the executed task time is recorded for analyzing the performance of MCC by TORO architecture.The existing FDCO algorithm and mCloud model execution data flow in are also simulated. The cloudlet is created as of in proposed TORO architecture except ASR operation. The request from the application is forwarded and decision is taken by decision engine to select the effective resource type such as 3G, Bluetooth and wifi to offload the task to VM. Further, offloaded information are stored in the database by the task manager for evaluation.

## 5.Result and Discussion

The evaluation of the proposed TORO architecture is compared with the existing FDCO algorithm and mCloud model which shows the performance improvement with the result generated for four different workloads as given in Table 6. Here, BTH denotes the big tasks which need high computation, BTL denotes big tasks which need low computation, STH represents small tasks which require high computation and STL represents small task which needs low computation. Further, the image size and MIPS of four workloads differs in size. The workloads are executed separately in the simulation environment and results are compared by plotting graphs. In graph, the execution time and resource utilization cost for tasks are analyzed for proposed TORO architecture and the existing FDCO and mCloud model.

| S.No | Workloads | No. of tasks | Image | MIPS |
|---|---|---|---|---|
| 1 | BTH | 500 | 3000 | 30 |
| 2 | BTL | 500 | 3750 | 17 |
| 3 | STH | 500 | 650 | 24 |
| 4 | STL | 500 | 725 | 6 |

**Table 6** Workload

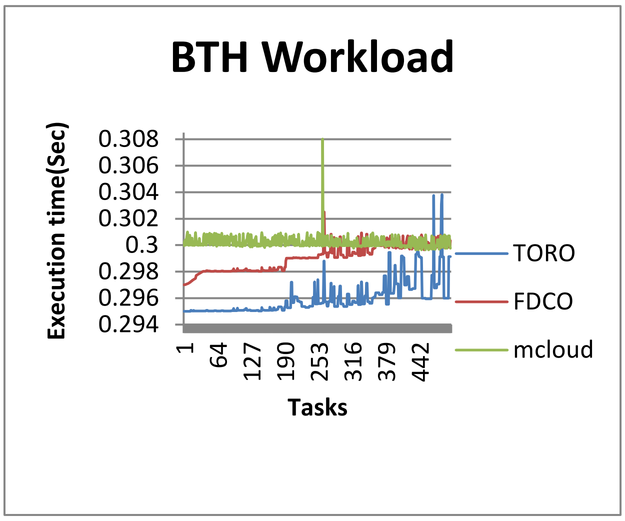

**Figure4** BTH workloads in TORO, FDCO and mCloud

Figure 4 represent the execution time for the workload BTH. The graph plotted shows the execution time of the proposed TORO architecture where tasks are executed fast when compared to existing FDCO and mCloud model. The overall optimization is applied in proposed TORO architecture to compute workload faster in dynamic environment.

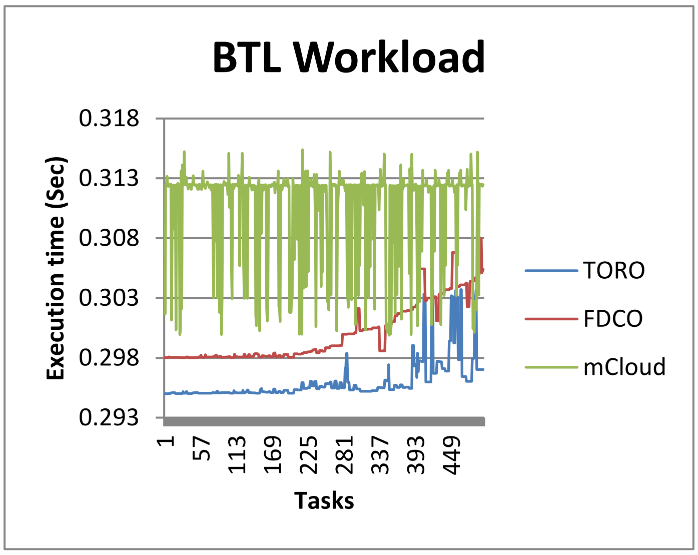

**Figure 5** BTL workloads in TORO, FDCO and mClo

Figure 5 describe the execution time for the workload BTL. The graph plotted depicts the execution time of proposed TORO architecture in which tasks are executed fast compared to existing FDCO and mCloud model. The overall optimization in the proposed TORO architecture made the workload to compute faster in dynamic environment.

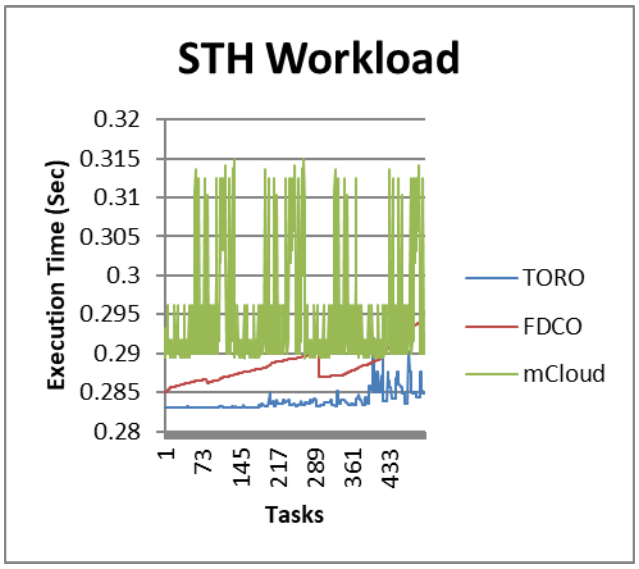

**Figure 6** STH workloads in TORO, FDCO and mCloud

Figure 6 represent the execution time for the workload STH. The plotted graph shows the execution time of proposed TORO architecture where the tasks are executed fast when compared with the existing FDCO and mCloud model. The overall optimization in the operations of proposed TORO architecture made the workload to compute faster in dynamic environment.

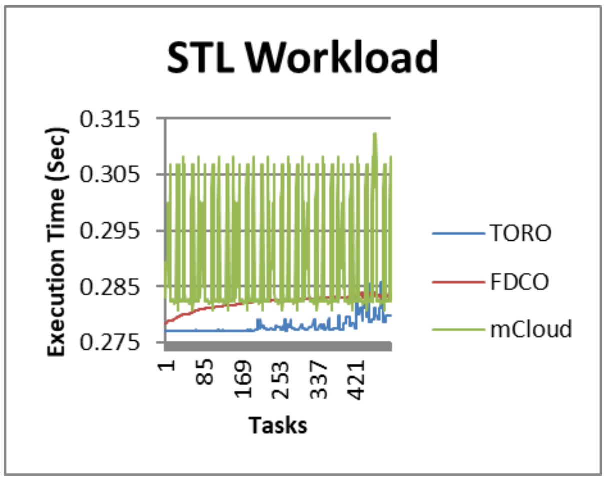

**Figure 7** STL workloads in TORO, FDCO and mCloud

Figure 7 illustrate the execution time for the workload STL. The graph plotted shows the execution time of proposed TORO architecture in which the tasks are executed fast when compared to the existing FDCO and mCloud model. The overall optimization done in the operations of proposed TORO architecture made the workload to compute faster in dynamic environment.

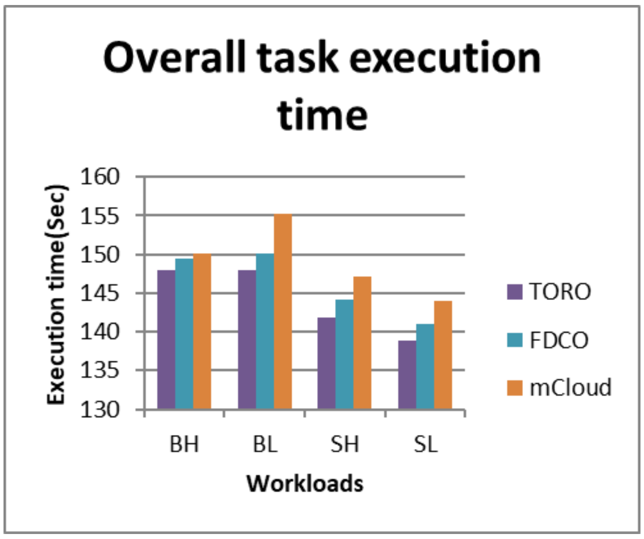

**Figure 8** Performance comparisons between TORO, FDCO and mCloud

The workloads executed in proposed TORO architecture and the existing FDCO algorithm and mCloud model are compared based on the execution time to show better performance which is represented in Figure 8. The execution time of the workload BTH shown for the proposed TORO architecture is 147.95 sec, the existing FDCO algorithm have 149.5 sec and mCloud model have 150.18 sec which shows that the proposed TORO architecture execute faster . The BTL computed in proposed TORO architecture have 147.99 sec, the existing FDCO have 150.03 sec and mCloud model have the execution time of 155.2 sec where the proposed TORO architecture compute faster. The STH computed in proposed TORO architecture have 141.9 sec, the existing FDCO have 144.22 sec and mCloud model have execution time 147.2 sec which indicates that the proposed TORO architecture execute faster. The workload STL computed in proposed TORO architecture have 138.9 sec, the existing FDCO have 141.08 sec and mCloud model have the execution time of 143.93 sec which shows that the proposed TORO architecture compute faster. Thus, the evaluation carried with four different workloads shows that the proposed TORO architecture compute the task faster and improved the performance of MCC in optimized and integrated way.

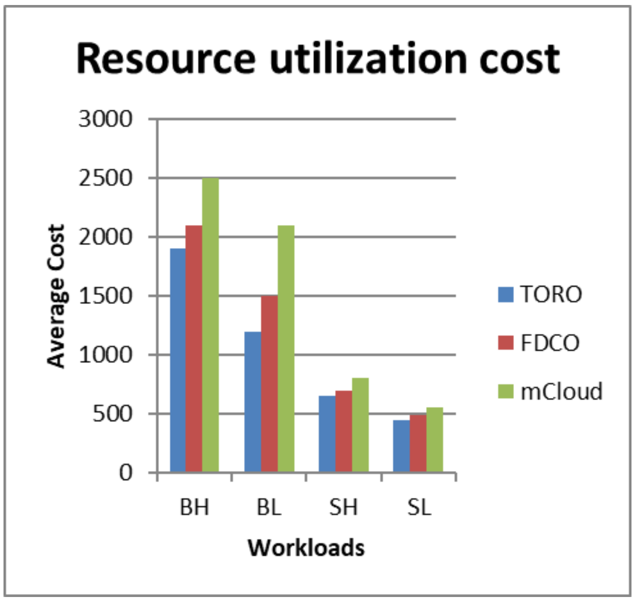

**Figure 9** Cost comparisons between TORO, FDCO and mCloud

The workloads computed in proposed TORO architecture and the existing FDCO and mCloud model are compared based on the overall resource usage and its cost is shown in Figure 9. The average cost of the workload BTH shown for the proposed TORO architecture is 1900, existing FDCO algorithm is 2100 and mCloud model is 2500 where the proposed TORO architecture computes the workload is cost effective when compared to existing FDCO and mCloud model. The BTL computed in proposed TORO architecture have cost around 1200, the existing FDCO have 1500 and mCloud model have cost around 2100 which indicate that the proposed TORO architecture show better cost. The STH computed in proposed TORO architecture have cost around 650, the existing FDCO have 700 and mCloud model have 800 which highlight that the proposed TORO architecture have better resource utilization cost. The workload STL computed in proposed TORO architecture has 450, the existing FDCO have 490 and mCloud model have 550 which specify that the proposed TORO architecture have better cost when compared to other two models. Thus from the analysis made with four different workloads proves that the proposed TORO architecture is cost effective to compute the task faster in MCC by integrated optimization.

## 5. Conclusion and Future work

In this paper, we proposed TORO architecture which optimization the various operations integrated together to the performance of MCC. The operations included are resource demand prediction, reliable cloudlet resource, task scheduling and partitioning, task offloading and optimization. The TORO architecture is simulated and evaluated with the FDCO algorithm and mCloud model. The evaluation result for various workloads shows that the proposed TORO architecture outperforms by executing tasks faster in cost effective manner with overall performance improvement when compared to the FDCO algorithm and mCloud model. In future, the proposed TORO architecture will be used to design a real-time mobile cloud operating system in heterogeneous environment.

## References

[1] B. Varghese, R. Buyya, Next generation cloud computing: New trends and research directions, Future Generation Computer Systems, 79 (2018) 849-861. https://doi.org/10.1016/j.future.2017.09.020.

[2] Jaiswal, A.S., Thakare, V.M. and Sherekar, S.S. (2015) ‘Study and analysis of architecture components of cloudlets in MCC’, International Journal of Electronics, Communication and Soft Computing Science and Engineering (IJECSCSE), pp. 376 – 382.

[3] Chunlin, L., Jing, Z. and Youlong, L. (2018) Cloud-based mobile service provisioning for system performance optimization, International Journal of Ad Hoc and Ubiquitous Computing, Vol. 29, No. 3, pp. 193 – 207. doi: 10.1504/IJAHUC.2018.095476

[4] X. Lyu, H. Tian, C. Sengul, P. Zhang, Multiuser joint task offloading and resource optimization in proximate clouds, IEEE Transactions on Vehicular Technology, 66 (4) (2017) 3435 – 3447. https://doi.org/10.1109/tvt.2016.2593486.

[5] K. Manbir, K. Kiranbir, K. Lohit, An Efficient Resource Provisioning Technique in Inter-Cloud Using Peer-to-Peer Approach, American Journal of Engineering and Applied Sciences, 10 (2) (2017) 529–539. https://doi.org/10.3844/ajeassp.2017.529.539.

[6] H. Wen, L. Yang, Z. Wang, ParGen: A Parallel Method for Partitioning Data Stream Applications in Mobile Edge Computing, IEEE Access, 6 (2018) 5037-5048. https://doi.org/10.1109/access.2017.2776358.

[7] Y. Son and Y. Lee, “Offloading Method for Efficient Use of Local Computational Resources in Mobile Location-Based Services Using Clouds,” Mobile Information Systems, vol. 2017, pp. 1–9, 2017. https://doi.org/10.1155/2017/1856329

[8] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, “Hermes: Latency Optimal Task Assignment for Resource-constrained Mobile Computing,” IEEE Transactions on Mobile Computing, vol. 16, no. 11, pp. 3056–3069, Nov. 2017. https://doi.org/10.1109/tmc.2017.2679712

[9] X. Jin, Z. Wang, and W. Hua, “Cooperative Runtime Offloading Decision Algorithm for Mobile Cloud Computing,” Mobile Information Systems, vol. 2019, pp. 1–17, Sep. 2019. https://doi.org/10.1155/2019/8049804

[10] B. Li, Y. Pei, H. Wu, and B. Shen, “Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds,” The Journal of Supercomputing, vol. 71, no. 8, pp. 3009–3036, Apr. 2015. https://doi.org/10.1007/s11227-015-1425-9

[11] W. Junior, A. França, K. Dias, and J. N. de Souza, “Supporting mobility-aware computational offloading in mobile cloud environment,” Journal of Network and Computer Applications, vol. 94, pp. 93–108, Sep. 2017. https://doi.org/10.1016/j.jnca.2017.07.008

[12] M. Goudarzi, M. Zamani, and A. T. Haghighat, “A fast hybrid multi-site computation offloading for mobile cloud computing,” Journal of Network and Computer Applications, vol. 80, pp. 219–231, Feb. 2017. https://doi.org/10.1016/j.jnca.2016.12.031

[13] Z. Kuang, S. Guo, J. Liu, and Y. Yang, “A quick-response framework for multi-user computation offloading in mobile cloud computing,” Future Generation Computer Systems, vol. 81, pp. 166–176, Apr. 2018. https://doi.org/10.1016/j.future.2017.10.034

[14] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, “Heterogeneity-Aware Task Allocation in Mobile Ad Hoc Cloud,” IEEE Access, vol. 5, pp. 1779–1795, 2017. https://doi.org/10.1109/access.2017.2669080

[15] H. Cao and J. Cai, “Distributed Multiuser Computation Offloading for Cloudlet-Based Mobile Cloud Computing: A Game-Theoretic Machine Learning Approach,” IEEE Transactions on Vehicular Technology, vol. 67, no. 1, pp. 752–764, Jan. 2018. https://doi.org/10.1109/tvt.2017.2740724

[16] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, “Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing,” IEEE Wireless Communications Letters, vol. 6, no. 6, pp. 774–777, Dec. 2017. https://doi.org/10.1109/lwc.2017.2740927

[17] Z. Yin, H. Chen, and F. Hu, “An advanced decision model enabling two-way initiative offloading in edge computing,” Future Generation Computer Systems, vol. 90, pp. 39–48, Jan. 2019. https://doi.org/10.1016/j.future.2018.07.031

[18] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje, “On the optimal allocation of virtual resources in cloud computing networks,” IEEE Transactions on Computers, vol. 62, no. 6, pp. 1060–1071, Jun. 2013. https://doi.org/10.1109/tc.2013.31

[19] F. A. Nakahara and D. M. Beder, “A context-aware and self-adaptive offloading decision support model for mobile cloud computing system,” Journal of Ambient Intelligence and Humanized Computing, vol. 9, no. 5, pp. 1561–1572, Apr. 2018. https://doi.org/10.1007/s12652-018-0790-7

[20] M. Chen and Y. Hao, “Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network,” IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 587–597, Mar. 2018. https://doi.org/10.1109/jsac.2018.2815360

[21] A. Ceselli, M. Fiore, M. Premoli, and S. Secci, “Optimized assignment patterns in Mobile Edge Cloud networks,” Computers & Operations Research, vol. 106, pp. 246–259, Jun. 2019. https://doi.org/10.1016/j.cor.2018.02.022

[22] G. Peng, H. Wang, J. Dong, and H. Zhang, “Knowledge-Based Resource Allocation for Collaborative Simulation Development in a Multi-Tenant Cloud Computing Environment,” IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 306–317, Mar. 2018. https://doi.org/10.1109/tsc.2016.2518161

[23] J. Tang, W. P. Tay, and T. Q. S. Quek, “Cross-Layer Resource Allocation With Elastic Service Scaling in Cloud Radio Access Network,” IEEE Transactions on Wireless Communications, vol. 14, no. 9, pp. 5068–5081, Sep. 2015. https://doi.org/10.1109/twc.2015.2432023

[24] F. Wang, J. Xu, X. Wang, and S. Cui, “Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems,” IEEE Transactions on Wireless Communications, vol. 17, no. 3, pp. 1784–1797, Mar. 2018. https://doi.org/10.1109/twc.2017.2785305

[25] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, M. K. Khan, M. Ni, and G. Zhu, “Joint Admission Control and Resource Allocation in Edge Computing for Internet of Things,” IEEE Network, vol. 32, no. 1, pp. 72–79, Jan. 2018. https://doi.org/10.1109/mnet.2018.1700163

[26] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, “ULOOF: A User Level Online Offloading Framework for Mobile Edge Computing,” IEEE Transactions on Mobile Computing, vol. 17, no. 11, pp. 2660–2674, Nov. 2018. https://doi.org/10.1109/tmc.2018.2815015

[27] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, “Joint Radio and Computational Resource Allocation in IoT Fog Computing,” IEEE Transactions on Vehicular Technology, vol. 67, no. 8, pp. 7475–7484, Aug. 2018. https://doi.org/10.1109/tvt.2018.2820838

[28] F.-H. Tseng, H.-H. Cho, K.-D. Chang, J.-C. Li, and T. K. Shih, “Application-oriented offloading in heterogeneous networks for mobile cloud computing,” Enterprise Information Systems, vol. 12, no. 4, pp. 398–413, Feb. 2017. https://doi.org/10.1080/17517575.2017.1287432

[29] X. Guo, L. Liu, Z. Chang, and T. Ristaniemi, “Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds,” Wireless Networks, vol. 24, no. 1, pp. 79–88, Jun. 2016. https://doi.org/10.1007/s11276-016-1322-z

[30] S. Rashidi and S. Sharifian, “A hybrid heuristic queue based algorithm for task assignment in mobile cloud,” Future Generation Computer Systems, vol. 68, pp. 331–345, Mar. 2017. https://doi.org/10.1016/j.future.2016.10.014

[31] J. Li and C. Wu, “Aviation Logistics Mobile Internet Cloud Computing Optimization,” International Journal of u- and e-Service, Science and Technology, vol. 9, no. 7, pp. 369–380, Jul. 2016. https://doi.org/10.14257/ijunesst.2016.9.7.37

[32] Y. Cai, F. R. Yu, and S. Bu, “Dynamic Operations of Cloud Radio Access Networks (C-RAN) for Mobile Cloud Computing Systems,” IEEE Transactions on Vehicular Technology, vol. 65, no. 3, pp. 1536–1548, Mar. 2016. https://doi.org/10.1109/tvt.2015.2411739

[33] Y. Lin, “Based on Particle Swarm Optimization Algorithm of Cloud Computing Resource Scheduling in Mobile Internet,” International Journal of Grid and Distributed Computing, vol. 9, no. 6, pp. 25–34, Jun. 2016. https://doi.org/10.14257/ijgdc.2016.9.6.03

[34] Joshi, P., Rathnamma, M.V., Srujan Raju, K., Pawar, U. (2021). Miss Rate Estimation (MRE) an Novel Approach Toward L2 Cache Partitioning Algorithm’s for Multicore System. In: Satapathy, S., Bhateja, V., Janakiramaiah, B., Chen, YW. (eds) Intelligent System Design. Advances in Intelligent Systems and Computing, vol 1171. Springer, Singapore. https://doi.org/10.1007/978-981-15-5400-1_58

[35] Jeyakanth, Krishnan, Perumalsamy Venkatakrishnan, and Chinnasamy Chitra. "Optimized channel prediction and auction-based channel allocation for personal cognitive networks." International Journal of Communication Systems 36, no. 3 (2023): e5391.