Journal of Intelligent & Fuzzy Systems 40 (2021) 787–797
DOI:10.3233/JIFS-200799
IOS Press

Cloud resource optimization based on poisson linear deep gradient learning for mobile cloud computing

G. Saravanan[a,*] and N. Yuvaraj[b]
[a]*Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Erode, Tamil Nadu, India*
[b]*Department of Information Technology, J.K.K. Nattraja College of Engineering and Technology, Natarajapuram, Komarapalayam, Tamil Nadu, India*

Abstract. Mobile Cloud Computing (MCC) addresses the drawbacks of Mobile Users (MU) where the in-depth evaluation of mobile applications is transferred to a centralized cloud via a wireless medium to reduce load, therefore optimizing resources. In this paper, we consider the resource (i.e., bandwidth and memory) allocation problem to support mobile applications in a MCC environment. In such an environment, Mobile Cloud Service Providers (MCSPs) form a coalition to create a resource pool to share their resources with the Mobile Cloud Users. To enhance the welfare of the MCSPs, a method for optimal resource allocation to the mobile users called, Poisson Linear Deep Resource Allocation (PL-DRA) is designed. For resource allocation between mobile users, we formulate and solve optimization models to acquire an optimal number of application instances while meeting the requirements of mobile users. For optimal application instances, the Poisson Distributed Queuing model is designed. The distributed resource management is designed as a multithreaded model where parallel computation is provided. Next, a Linear Gradient Deep Resource Allocation (LG-DRA) model is designed based on the constraints, bandwidth, and memory to allocate mobile user instances. This model combines the advantage of both decision making (i.e. Linear Programming) and perception ability (i.e. Deep Resource Allocation). Besides, a Stochastic Gradient Learning is utilized to address mobile user scalability. The simulation results show that the Poisson queuing strategy based on the improved Deep Learning algorithm has better performance in response time, response overhead, and energy consumption than other algorithms.

Keywords: Mobile cloud computing, mobile cloud service providers, mobile cloud users, poisson linear, stochastic, deep neural resource allocation

1. Introduction

In a mobile computing environment, the integration of cloud and mobile services also referred to as MCC enhances the transmission effectiveness of the data and the potentiality involved in processing. One of the key issues to be addressed in MCC is resource management that determines the QoS of mobile cloud services and the usage of cloud resources. However, due to the vital and vague scenes in MCC, resource optimization is still uncertain. Poisson distribution is a discrete probability distribution that presents the probability of a number of events occurring in a fixed period of time. The Poisson distribution is generally employed when the occurrences are independent. Poisson distribution arises when a number of events

*Corresponding author. G. Saravanan, Assistant Professor, Department of Computer Science and Engineering, Erode Sengunthar Engineering College, Erode-638057, Tamil Nadu, India. E-mail: gsaravanan.pacet@gmail.com.

are counted across time or over the area. The probability of two or more occurrences of the number of event in a very tiny time interval is negligible.

A novel Elasticity Debt Analytics was proposed in [1] for resource provisioning in MCC. The method used a game theoretic approach for ensuring the quality of service. With the objective of improving resource utilization, utility-based elasticity dept and profit quantification were also investigated via the hidden Markov model. The problem was formulated as an elasticity debt quantification game, with the aid of an incentive mechanism.

With this incentive, mechanism elasticity debt was predicted and therefore resulting in the optimization of cloud resource provisioning. Due to this, the method allocates the mobile device user requests to high elasticity debt-level services. As a result, elasticity debt minimization was facilitated for MCC environments. However, the optimization time involved in the resource being utilized was not focused. To address this issue, in this work, along with the allocation of resources in an optimal manner, first, Poisson queuing based cloud user entry is formulated so that the resource optimization time can also be minimized considerably.

In MCC, the reservation and allocation of resources in the preliminary stage can notably minimize the Cloud Service Providers (CSP) total provisioning cost. However, the undetermined attributes of users' in mobility necessitate the requirement of resources that in turn make the reservation and allocation uncertain. As far as MCC is concerned, the mobile application QoS is determined based on both the resources utilized in the radio propagation network and the resources required for virtual machine processing. Hence, a joint allocation of these two types of resources is said to be required.

In [2], RRA with uncertainty over mobile user demand, a robust optimization model was designed. Besides, constant relative risk aversion functions were also established to extract the needs and want of the mobile users'. With this, allocations between RRs and VMRs were done based on the resource requirements of mobile applications. Followed by which a powerful Collective Resource Provision and Assignment mechanism was also designed for optimization of radio propagation network and virtual machine processing. Due to this, the total foundation cost in addition to the resource utilization was found to be efficient. Despite improvement observed in provisioning cost, the overhead incurred in resource utilization was not focused. To address this issue, in this work, a Linear Rationalized Deep Neural Resource allocation is designed that not only considers optimal resource allocation but also the overhead incurred.

Although equilibrium and logarithmic model has various advantages, there are also some shortcomings like other intelligent algorithms. For example, it is easy to fall into local optimum and sometimes difficult to attain the quality of service (QoS), like reducing response time and overhead. To further improve the performance of this algorithm, Poisson Queuing, Linear Gradient Deep Resource Allocation, and Stochastic Gradient are incorporated into learning in this paper. The Poisson Queuing helps to follow the exponential distribution and balance the mobile cloud user incoming processes (i.e., incoming requests). The Linear Gradient Deep Resource Allocation along with Stochastic Gradient ensures a smooth occurrence of optimal resource allocation solution via optimal decision making and high perception ability factors. Based on the above improvement aspects, our Linear Gradient Deep Resource Allocation algorithm in PL-DRA when applied to the allocation of data or information based on resource optimization in MCC environment, helps to maintain the balance between response time and overhead.

In this paper, four major contributions are as follows:

(1) To support the precise requirement in a complex MCC environment, the resource allocation problem of mobile cloud user instances (i.e. data or information request) is formalized. It provides a theoretical foundation for mobile cloud user instance allocation based on mobile cloud user demands.
(2) To cope with a potentially increased number of cloud servers, a distributed resource controller model, called, Poisson Distributed Resource Queuing has been developed. Each gradient is responsible for modeling performance and request of a single user while exchanging information with other gradients to harmonize resource allocation.
(3) To compensate for the shortcomings of equilibrium and logarithmic model, an improved deep learning algorithm that employs linear and stochastic gradient operations satisfying decision making and perception ability is developed. It reduces the overhead incurred in an instance.

(4) To verify the effectiveness and efficiency of our novel resource optimization algorithm, based on various examples, comprehensive experimental evaluation is demonstrated. It can provide the superiority of the PL-DRA method compared with the state-of-the-art methods.

The structure of the remainder of the paper is as follows. Section 2 introduces the related work. Section 3 provides the system model along with the problem formulated and detailed resource optimization method based on improved Deep Learning. Section 4 illustrates the experimental settings and comprehensive results. Section 5 summarizes the paper.

2. Related works

In the past few years, considerable development has been made in MCC environment. However, due to aspects such as inadequate computing and storage potentiality of mobile cloud users and the restricted batter volume by hardware design, only a few simple tasks are said to be performed.

Artificial Intelligence (AI) technology was applied in [3] based on the data size of the computation task to minimize the total task delay with increasing data. However, the resource usage or running cost in the cloud was not taken into consideration. To address this issue, multi-tenancy and asynchronous processing was introduced in [4], therefore contributing to constrained resource demand. Performance and energy efficiency was enhanced in [5] using non-convex optimization based on Weighted Minimum Mean Square Error (WMMSE).

As the notion of integrating the potentialities of mobile devices and cloud computing is becoming progressively well received, the most prerequisite query occurs in optimal scheduling of services between devices and the cloud. A service selection optimization algorithm based on supervised learning was presented in [6] for reducing energy consumption with significant cloud service executions. However, the response time was not reduced due to an imbalanced load. To address this issue, a fast non-dominated sorting genetic algorithm [7] was designed. That not only reduced the response time but also improved network resource utilization significantly. Yet another QoS-aware service provisioning scheme was presented in [8] therefore contributing towards queuing time.

In recent years with the proliferation of mobile devices and cloud computing environment has resulted in various powerful mobile applications. Besides most of these mobile applications specifically necessitate rigorous estimation and high processing, that are found to be not compatible with mobile devices due to their resource limitation.

Graph partitioning algorithms were designed in [9] with the objective of minimizing the required bandwidth. Yet another meta-heuristic model based on Greedy Randomized Adaptive Search Procedure (GRASP) was designed in [10] to improve the performance along with the QoS. Furthermore, an AES cryptographic technique was proposed in [11] to determine optimal computation offloading and reduce the computation and communication cost.

Cloud computing environment failures simultaneously affect several mobile cloud users and therefore decreasing the number of available computing resources. Besides, a bad scheduling mechanism can also reduce the physical machines expected makespan and the idle time concurrently. To address these issues, optimized scheduling of scientific workflows on the cloud was presented in [12] for virtual machine allocation and task scheduling effectively.

Yet another multi-objective resource optimization model based on energy consumption, time consumption, and the cost was presented in [13] to identify the optimal offloading strategy. A meta-heuristic approach maximizing demand satisfaction and minimizing cost was presented in [14] using the fuzzy linguistic method. However, with fixed length job requests with dynamic situations being arisen having variable lengths, optimization was not ensured. To focus on this issue, a varied length algorithm close to the optimum with the minimum gap was presented in [15]. This in turn contributed towards strong system stability and low congestion.

To reduce both cost and makespan, a novel Directional and Nonlocal Convergent Particle Swarm Optimization (DNCPSO) mechanism were presented in [16]. Yet another graph partitioning heuristic and delta synchronization model were designed in [17] to enhance the quality dimensions such as latency and bandwidth consumption.

A resource cooperative provision mode was designed in [18] using two different aggregation models. It is not only ensuring flexible resource management but also ensured dynamic resource allocation. However, the focus was not oriented towards price attribution. To provide a solution to this issue, the mean variance optimization algorithm was

designed in [19] to obtain optimized cloud resource allocation. A survey focusing on service adoption and provision in MCC was proposed in [20].

These above improved Deep Learning algorithms have been developed in various ways that in turn effectively achieve significant performance. However, there are only very few researches about lowering resource usage on the cloud side for MCC environment. Many applications not only minimize the total delay but also consider the accuracy factors and real-time requirements of certain specific tasks. Thus, this paper will first settle the issue of queuing in a mobile cloud computing environment, which can take full advantage of queuing. Then we will develop our novel Linear Gradient Deep Resource Allocation algorithm from two aspects (i.e. bandwidth and memory) to deal with the application in an adaptive fashion and address the issues effectively and efficiently.

3. Methodology

In this section, we first introduce the system model. Then, we introduce the specific problem background of this paper. Followed by which the design of Poisson Linear Deep Resource Allocation (PL-DRA) is given.

3.1. System model

The system model comprises of three entities, Mobile Cloud Data Owners '$MCDO = MCDO_1, MCDO_2, \ldots, MCDO_n$', Cloud Service Providers '$CSP = CSP_1, CSP_2, \ldots, CSP_n$' via Cloud Servers '$CS = CS_1, CS_2, \ldots, CS_n$' and Mobile Cloud Users '$MCU = MCU_1, MCU_2, \ldots, MCU_n$' respectively. The topology of the cloud service providing framework based on mobile cloud computing is depicted in Fig. 1.

3.2. Problem formulation

In this work, all computing devices in the MCC are represented by a graph as '$G = (D, E, G)$'. Here, 'D' represents a cloud service provider holding cloud servers. Next, 'E' represents the communication constraints between the Mobile Cloud Data Owners '$MCDO$' and the cloud service provider 'CSP' (i.e. service provisioning scheme). Here, the constraints represent the classification of Mobile Cloud Data Owners into segments, corporate, premium, and basic users [1] respectively. Finally, 'G' represents the

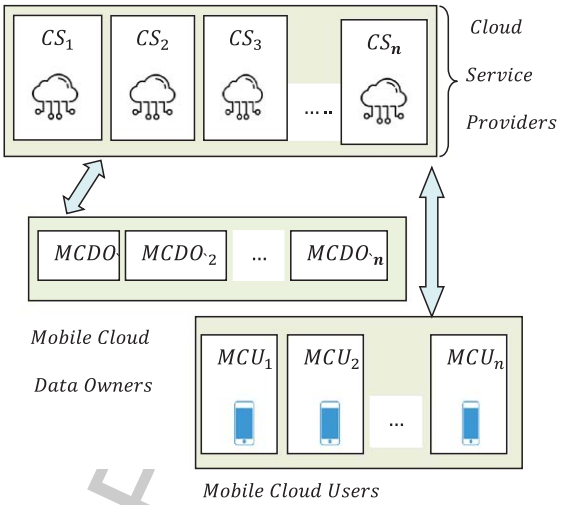

Fig. 1. Topology of service provisioning framework based on Mobile Cloud Computing.

Mobile Cloud Users. With the above graphical construction, the problem remains in minimizing the response time, overhead, and energy consumption while fetching the data or information of the user via optimal resource allocation. The resource allocation is carried out by the service provider based on the available resources (i.e. bandwidth and memory) and requirements of the users. Here, the objective remains lowering the response time, overhead, and energy consumption by optimizing the resource usage on the cloud side for MCC. With the above system model and the problem being formulated, the PL-DRA method is elaborated in the following section.

3.3. Poisson distributed resource queuing

In order to evaluate the PL-DRA method, a Poisson Distributed Resource Queuing (PDRQ) described in the MCC framework. In the PDRQ model, the mobile cloud user is denoted as '$MCU = (MCU_1, MCU_2, \ldots, MCU_n)$' as the set of '$MCUs$' in the cloud server with '$R = (r_1, r_2, \ldots, r_n)$' their available free resource, where 'r_j' is assigned in terms of 'CPU', 'RAM' and '$StorageCapacity$' respectively, with '$r_j = (\alpha_{cj}, \beta_{cj}, \gamma_{cj})$'. Also, we represent '$CS = (CS_1, CS_2, .., CS_n)$' as the set of '$CSs$' to be allocated to '$MCU$' where '$ar_i = (\alpha_{ri}, \beta_{ri}, \gamma_{ri})$' representing the amount of resources required for running the cloud server.

A distributed resource controller is introduced to provide subsistence for a probably substantial number of cloud servers. The single Poisson Queu-

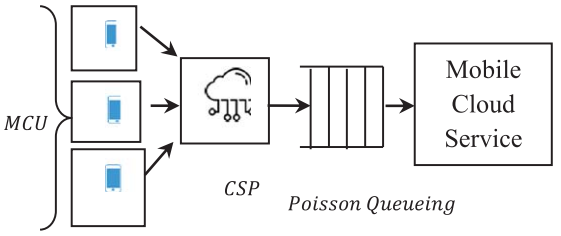

Fig. 2. Hypothetical representation of Poisson Optimal Resource Queuing.

ing model is split into several prototypes, each of them being in charge of the performance of a single cloud server. Each of the distributed resource controllers optimizes its mean queue length deep resource allocation model and exchanges information with other resource controllers to harmonize resource allocations. In the distributed version, the resource controller is split into several resource controllers, each accountable for resource allocation of a single cloud server. Figure 2 depicts the Hypothetical representation of Poisson Optimal Resource Queuing.

As depicted in Fig. 2, a service provider utilizes resources to fetch the data or information as requested by multiple users'. The service provider requires to obtain the request from the user with the assumption that the user requests arrive at the service provider based on Poisson distribution, the arrival rate being 'λ'. The service time of the service provider before the user data reaches the cloud in our work discharges the with a service rate of 'μ' respectively. With the assumptions of this arrival rate and service rate, the pseudo-code representation of Poisson Optimal Resource Queuing is given below.

Algorithm 1 Poisson Optimal Resource Queuing

Input: Mobile Cloud Data Owners '$MCDO = MCDO_1, MCDO_2, \ldots, MCDO_n$', Cloud Service Providers '$CSP = CSP_1, CSP_2, \ldots, CSP_n$' Cloud Servers '$CS = CS_1, CS_2, \ldots, CS_n$' Mobile Cloud Users '$MCU = MCU_1, MCU_2, \ldots, MCU_n$',

Output: Optimal Resource Queuing

1: Initialize resource allocation start time
2: Initialize resource allocation completion time
3: Begin
4: For each Mobile Cloud Data Owners '$MCDO$' with Cloud Servers 'CS' and Mobile Cloud Users 'MCU'
5: With the problem formulated in (1)
6: Measure makespan using (2)
7: Obtain constraint using (3)
8: End for
9: End

From the above algorithm, exceptionally, the queuing service and mobile cloud service represent different services to be provided to the users. On one hand, the queuing service comprises forwarding user requests (i.e. want of data or information) to the service provider and optimal resource usage on the cloud side to respond to the user on the other hand. Next, the cloud service here represents the service that processes the user request and optimally allocates the services of the user. Then, the POR Queuing is expressed as given below.

$$Problem : F = \left\{ DAST\,(\mu) + DACT\left(\frac{\lambda}{\mu - \lambda}\right) + MS \right\} \tag{1}$$

$$MS = DACT{-}DAST \tag{2}$$

$$Subjectto : \frac{1}{\mu - \lambda} + ET_{MCU_i} \leq ET \tag{3}$$

From the above Equations (1), '$DAST$' represents the data allocation start time, '$DACT$' represents the data allocation completion time with 'MS' representing the makespan (i.e. the difference between the data allocation completion time and the data allocation start time respectively) and '$\frac{\lambda}{\mu-\lambda}$' denoting the mean queue length in the cloud server.

Next, from the Equation (3), '$\frac{1}{\mu-\lambda}$' corresponds to the mean queue length time per unit user in a cloud server, with 'ET' representing the average execution time for a resource to be allocated and 'ET_{MCU_i}' representing the execution time for a resource to be allocated to a single user. The above problem is based on the mean queue length time per unit user in the cloud server subject to mean queue length in a cloud server. At first, optimal users obtain in the cloud server for resource requests and response time is significantly reduced.

3.4. Linear Gradient Deep Resource Allocation

With the measured mean queue length via Poisson distribution, the data or information to be allocated by the server amongst the user is explained in this section.

Also, bandwidth and memory optimization problems are studied. The resource allocation framework

based on Gradient Deep Learning (GDL) is given. To start with a Utility Optimizer 'UOP' using Linear Programming (LP) model for data allocation to users from the resource pool created by the service providers is designed. The pseudo-code representation of Linear Gradient Deep Resource Allocation is given below.

Algorithm 2 Linear Gradient Deep Resource Allocation

Input: Mobile Cloud Data Owners '$MCDO = MCDO_1, MCDO_2, \ldots, MCDO_n$', Cloud Service Providers '$CSP = CSP_1, CSP_2, \ldots, CSP_n$' Cloud Servers '$CS = CS_1, CS_2, \ldots, CS_n$' Mobile Cloud Users '$MCU = MCU_1, MCU_2, \ldots, MCU_n$',

Output: Optimal resource allocation

1: **Initialize** number of application instances
2: **Begin**
3: **For** each Mobile Cloud Data Owners '$MCDO$' with Cloud Servers 'CS' and Mobile Cloud Users 'MCU'//measure utility optimizer
4: Obtain bandwidth constraint using (5)
5: Obtain memory constraint using (6)
6: Measure *ithvector*' of the training process using (7)
7: Update request (i.e. mobile cloud user) based on sigmoid activation using (8)
8: Fine-tune request based on Stochastic Gradient Learning using (9)
9: **End for**
10: **End**

As given in the above algorithm, to start with, the objective function is to determine the number of application instances (i.e. users) is mathematically expressed as given below.

$$\sum_{u \in MCU} \sum_{a \in app} \sum_{c \in CS} MAX\left(fun_{u,a,c}\right) \quad (4)$$

From the above Equation (4), '$fun_{u,a,c}$' represents the number of application instances from users 'u' using application 'a' connecting to the cloud server 'c' respectively. Here, the example application refers to the requirement of a specific product '*book*' to be purchased from a specific cite '*openlibrary*'. With the above, application instances, a 'UOP', the optimal resource allocation strategy for maximizing the bandwidth and minimizing memory is expressed as given below.

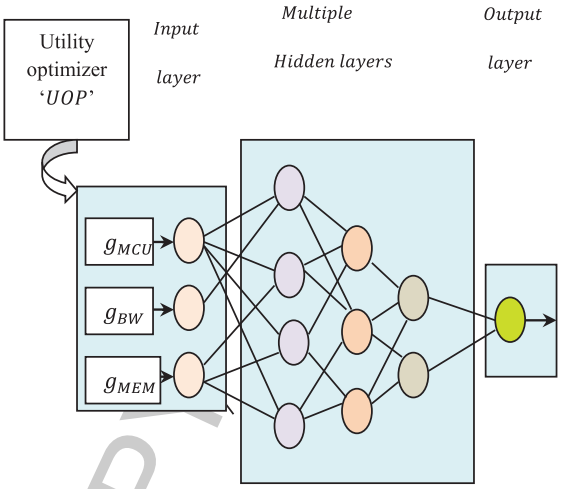

Fig. 3. Resource allocation framework based on Linear Deep Neural Network.

$$\sum_{u \in MCU} \sum_{a \in app} \sum_{c \in CS} fun_{u,a,c} R^{BW} \leq A_c^{BW} \quad (5)$$

From the above equation, the constraint in (5) ensures that the total amount of required bandwidth to support '$fun_{u,a,c}$' instances must be less than or equal to the available bandwidth in the resource pool. Here, 'R^{BW}' represents the bandwidth 'BW' required 'R' per instance of application 'a' and 'A_c^{BW}' represents the available bandwidth at cloud server 'c'.

$$\sum_{u \in MCU} \sum_{a \in app} \sum_{c \in CS} fun_{u,a,c} R^{MEM} \leq A_c^{MEM} \quad (6)$$

Besides, the constraint in (6) ensures that the total amount of required memory to support '$fun_{u,a,c}$' instances must be less than or equal to the available cloud servers' memory. Here, 'R^{MEM}' represents the memory utilization required per instance of application 'a' and 'A_c^{MEM}' represents the available memory at cloud servers 'c' respectively. Considering uncertainties in the system's parameters, we next present a Linear Deep Neural Resource Allocation (L-DNRA). By using the L-DNRA model, the above-said resources (bandwidth and memory) are said to be utilized in an optimized manner on the cloud side. Hence the user instances are said to be fetched with minimum time and overhead. Figure 3 shows the Resource allocation framework based on Linear Deep Neural Resource Allocation (L-DNRA).

As shown in Fig. 3, to start with an 'UOP' uses the Linear Deep Neural Network model with the objective of searching, for resource allocations that reach

maximum utility. The training method for obtaining characteristic parameters is obtained via the Linear Programming model. The mapping from the problem parameter 'c' and the initial value 'a_0' to the final output 'a_f', where 'f' denotes the feasible solution. It is accurately measured using deep neural networks that possess 'n' hidden units and an activated function based on sigmoid. The feasible solution refers to the optimization of resources (i.e. bandwidth and memory) with which the data or information requested as per the user is optimally assigned by the server.

Based on the linear programming model, it is inferred that a deterministic algorithm that iteratively updates continuous mappings between input and output units is said to be learned via a deep neural network. By using an L-DNRA that the optimal CPU computing power for maximizing the resource allocation of MCC is a continuous mapping between the users and service providers. L-DNRA is used to attain the CPU computing power allocation strategies.

Thus the PLS-DNRA method is designed according to the Linear Programming model. A resource allocation framework is based on L-DNRA to optimize the performance of MCC of the user and service provider of the MCC environment. It comprises three layers, namely, the input layer, multiple hidden layers, and finally the output layer. The inputs are the instantaneous user, bandwidth, and the storage capacity that possess continuous probability density functions. Then, the '*ithvector*' of the training process is mathematically expressed as given below.

$$a_i = \left[g^i_{MCU}, g^i_{BW}, g^i_{MEM}\right] \tag{7}$$

The output is the optimal resource being utilized for data or information allocation of the MCC environment on the cloud server-side. In this work, the output is the optimal data being allocated by the cloud server for maximizing the user request or requirement of the MCC environment.

However, with the increase in the number of users request from the cloud server for its resources, thus expanding the scope of the cloud server, the excessive energy consumption occurs. To reduce the energy consumption in resources being used for data or information to be allocated between users, the PL-DRA work uses Stochastic Gradient Learning (SGL). This is mathematically expressed as given below.

$$\theta = \theta - \eta * \nabla_\theta * J\left(\theta; a^{(i)}; b^{(i)}\right) * S$$

$$S = \frac{1}{1 + e^{-MCU}} \tag{9}$$

From the above Equation (8), $'\theta'$ represents the weight vector, 'η' is the step size, J is the function of gradient learning, a and b are the input and output. 'S' represents the sigmoid function. By selecting SGL, only a single sample is used to perform each iteration instead of one million samples for completion of one iteration. In this manner, the sample or the user request is randomly shuffled and selected for performing the iteration, therefore minimizing the computation overhead.

4. Experimental evaluation

The experimental evaluation of the proposed PL-DRA method and existing methods Elasticity Debt Analytics [1] and RRA [2] are implemented using Java language. The experimental evaluation is performed using Amazon EC2 big dataset taken from Personal Cloud Datasets http://cloudspaces.eu/results/datasets. The data in MCC environment are processed by the resource optimized data center (i.e. service provider). To conduct a fair comparison between the PL-DRA and existing methods, varying mobile cloud users data collected at different time intervals in the range of 50 to 500 are considered.

Experimental evaluation of the PL-DRA method is compared with the existing methods with different factors such as response time, response overhead, and energy consumption with respect to the number of users. The experimental results are discussed in the next section with different parameters. With the user's data considered as input, three different metrics are evaluated using the PL-DRA method and existing methods, Elasticity Debt Analytics [1] and RRA [2]. To ensure a fine-grained and fair comparison, similar input data are used as test data for both PL-DRA and existing methods. The three metrics are measured using the formulations given below in the results and discussion section.

5. Discussion

Results of the PL-DRA method and two existing methods Elasticity Debt Analytics [1] and RRA [2] are discussed in this section. The various performance metrics such as response time, response overhead,

and energy consumption are discussed with respect to a different number of mobile cloud users data selected in the range of 50 to 500 collected at different time intervals. The performance of the PL-DRA method and existing methods are discussed with help of table values and graphical representation.

5.1. Scenario 1: Response time

Response time refers to the set of processes and methods to match available resources (i.e. bandwidth and memory) with the needs of the users to achieve established goals. In other words, allocation comprises of achieving the desired results within the stipulated time with minimum utilization of the resources themselves. The requirement to optimize resources is specifically evident when the service providers have to address several users' requests at the same time and hence exceeding the resources currently available. The response time here refers to the time consumed in allocating the desired users' data or information at a specific time instance and is expressed as given below.

$$R_{time} = \sum_{i=1}^{n} MCU_i * Time\,(Data_{alloc}) \qquad (10)$$

From the above equation (9), the response time 'R_{time}' is measured based on the mobile cloud user requests 'MCU_i' placed for certain data or information from the cloud server and the time consumed in allocating the data or information '$Time\,(Data_{alloc})$' respectively. It is measured in terms of milliseconds (ms). Table 1 is given to compare the response time required to assign the user request via optimal resource allocation. The table shows the response time generated by the existing Elasticity Debt Analytics [1], RRA [2] with that required by the PL-DRA method. It is seen from Table 1 that the optimal resource allocation based on linear deep neural significantly decreases the response time, irrespective of the number of users. This verifies the efficiency of our resource allocation method in terms of the response time and shows the potentials for achieving in MCC environment.

When the number of users' data or information increases, there increases in response time, since is it running at the same speed as the cloud and the other curves are all ascending. Ascending curves show that the response time increases as the number of users' data or information requests from the cloud server via service provider increases and it reaches its maximum on the top curve. The contribution of our PDRQ model finds a suitable performance-response trade-off through mean queue length and aids potentially large number of cloud servers by a distributed resource controller. With a distributed resource controller, the incoming request (made by the user) is being controlled by the controller based on the arrival and the service rate. With this balancing factor, the controller maintains a complete balance between the user and the service provider. However, in the case of [1], dynamic QoS was not focused and in [2] though total resource provisioning cost was reduced using robust optimization theory, response time was not focused. But, in the case of proposed work, a fair flow between the two entities, user request, and resource allocation response, is followed via optimal resource allocation being performed in the cloud server. Hence the time consumed in allocating the desired users' data or information. This reduces the response time using the PL-DRA method by 33% when compared to [1] and 54% compared to [2] respectively.

Table 1
Comparison of the response time for optimal resource allocation

Number of mobile cloud users	Response time (ms)		
	PL-DRA	Elasticity Debt Analytics	RRA
50	0.90	1.460	1.950
100	0.975	1.685	2.655
150	1.165	1.875	2.725
200	1.250	1.915	2.895
250	1.395	2.024	3.195
300	1.455	2.155	3.330
350	1.765	2.353	3.525
400	1.955	2.530	3.785
450	1.995	2.820	3.925
500	2.015	3.345	4.105

5.2. Scenario2: Response overhead

The second parameter measured in the PL-DRA method is the response overhead or the computational cost incurred while allocating the data or information made by the users. The response overhead refers to the processes and methods to match available resources (i.e. bandwidth and memory). In other words, response overhead consists of attaining the desired results within the stipulated overhead with minimum resource utilization. The response overhead refers to the memory consumed in allocating the required data or information to the number of users and is expressed as given below.

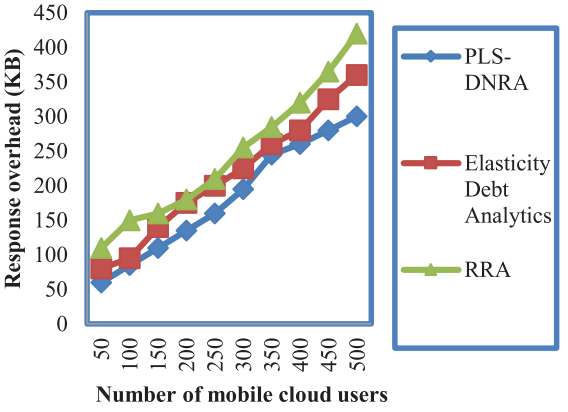

Fig. 4. Performance graph of response overhead.

$$R_{overhead} = \sum_{i=1}^{n} MCU_i * Mem\,(Data_{alloc}) \quad (11)$$

From the above Equation (10), the response overhead '$R_{overhead}$' is measured based on the user requests 'MCU_i' placed for certain data or information from the cloud server and the memory required to allocate the information or data with the available resources '$Mem\,(Data_{alloc})$' respectively. It is measured in terms of kilobytes (KB).

Figure 4 shows the response overhead between the number of the user by using the existing Elasticity Debt Analytics [1], RRA [2] and that achieved by using the PL-DRA method. The results are obtained by sampling the results from 500 different users. It can be seen that the response overhead increases as the users' requests increases.

Figure 4 is a figure of response overhead generated by each method or the computational cost incurred while data allocation is made by the users. With this simulation results, it is inferred that the response overhead based on the PL-DRA method achieves good results in latency and network usage, due to the optimal resources being allocated in the cloud server via the Linear Gradient Deep Resource Allocation algorithm. Through Deep Resource Allocation by considering bandwidth and memory constraint the data or information requested by the user is allocated by the cloud server using the Linear Gradient Deep Resource Allocation algorithm. On the other hand, by applying utility-driven elasticity debt and profit quantification approach as in [1], scalability and optimal resource provisioning is ensured but without considering the overhead. In a similar manner in [2], by using the logarithmic function, the total resource provisioning cost of cloud service providers is reduced, compromising the overhead. With the application of the linear programming model, the optimum use of resources is said to be made at the server-side. Besides, it also assists in re-evaluating the resource optimization for changing conditions based on the changes of user requests. Due to this, the response overhead is found to be comparatively lesser using the PL-DRA method by 15% compared to [1] and 28% compared to [2] respectively.

5.3. Scenario 3: Energy consumption

One of the most significant issues faced in the mobile cloud computing environment is the optimization of energy utilization. Besides, servers consume a major proportion of energy in MCC environments and hence their energy consumption varies correspondingly with their utilization. This energy consumption also differs from the computation type taking place on the mobile cloud server e.g., data retrieval and data processing. Hence, the energy consumption is measured as given below.

$$E_T = \sum_{i=1}^{n} MCU_i * \left[E_{Storage} + E_{Computation} + E_{Communication}\right] \quad (12)$$

From the above equation (11), the total energy consumption 'E_T', is the summation of the energy consumption of storage resources '$E_{Storage}$', the energy consumption of computation resources '$E_{Computation}$' and the energy consumption of communication resources '$E_{Communication}$' respectively. It is measured in terms of joules 'J'. Table 2 is provided to compare the energy being consumed while the data is assigned to the corresponding user. The table shows the energy consumption rate for the state-of-the-art methods, Elasticity Debt Analytics [1], RRA [2], and the PL-DRA method. From Table 2 that the energy consumption is significantly less when compared to [1] and [2].

To evaluate the optimization model proposed by us in terms of energy consumption, the energy consumed is compared to the results simulated. It is inferred that the energy consumption is not proportionately in the increasing trend. This is because of the reason that with the requirement of users' requests, the utilization of the resources also changes. Hence, the proportionate increase in terms of energy consumption requests based on Stochastic Gradi-

Table 2
Comparison of the energy consumption for optimal resource allocation

Number of mobile cloud users	Energy consumption (J)		
	PL-DRA	Elasticity Debt Analytics	RRA
50	4.920	6.750	8.475
100	6.245	9.425	10.725
150	8.885	10.165	13.655
200	9.385	12.560	15.985
255	10.375	14.755	18.425
300	11.365	15.825	20.825
350	12.475	17.655	23.995
400	12.925	19.565	25.725
450	13.565	21.465	28.855
500	14.725	24.875	30.535

ent Learning, continuous mappings among input and output units is learned via a deep neural network. However, in [1] though energy consumed in resource scheduling was focused via the Nash equilibrium approach, a tradeoff between energy consumption and overhead was observed and in [2] energy consumption between cloud and mobile devices was not focused. With this, the energy does not arrive. Simulations conducted with the application of the Fine-tune consumption are found to be significantly reduced using the PL-DRA method by 29% compared to [1] and 45% compared to [2] respectively.

6. Conclusion

In this article, the main focus remains in the optimal allocation of resources (i.e. bandwidth and memory) in MCC on the cloud side whenever placed with the user request of certain data or information from the user. We use Poisson Distributed Resource Queuing to narrate the collective impact of arrival rate and service rate of the incoming requests on the users' data or information provisioning. We have carried out the distributed resource controller as a multi-threaded request where each controller is run by one thread in counterpart with other controllers. Furthermore, a robust Linear Gradient Deep Resource Allocation algorithm is proposed. With the advantage of self-learning and high perception ability, optimal resource allocation is ensured. Therefore reducing time and overhead of user requests on certain data or information based on a deep network platform. Next, the energy being consumed for allocating the users' request is also said to be reduced via Stochastic Gradient Learning. The comprehensive evaluations show that the PL-DRA method performs significant resource allocation by reducing the response time, overhead, and energy consumption.

7. Ethical statements

- Yes, All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.
- Yes, The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript
- The article does not contain any studies with human participants performed by any of the authors. (Or) Ethical approval: This article does not contain any studies with animals performed by any of the authors. (Or) Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] G. Skourletopoulos, C.X. Mavromoustakis, G. Mastorakis, J.M. Batalla, H. Song, J.N. Sahalos and E. Pallis, Elasticity Debt Analytics Exploitation for Green Mobile Cloud Computing: An Equilibrium Model, *IEEE Transactions on Green Communications and Networking* **3**(1), Mar 2019.

[2] Y. Li, J. Liu, B. Cao1 and C. Wang, Joint Optimization of Radio and Virtual Machine Resources with Uncertain User Demands in Mobile Cloud Computing, *IEEE Transactions on Multimedia* **20**(9), Sept. 2018.

[3] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami and M.S. Hossain, Intelligent task prediction and computation offloading based on mobile-edge cloud computing, *Future Generation Computer Systems*, Elsevier, Sep 2019.

[4] P. Nawrocki and W. Reszelewski, Resource usage optimization in Mobile Cloud Computing, *Computer Communications*, Elsevier, Dec 2016.

[5] K. Wang, K. Yang and C.S. Magurawalage, Joint Energy Minimization and Resource Allocation in C-RAN with Mobile Cloud, *IEEE Transactions on Cloud Computing* **6**(3), July–Sept. 1 2018.

[6] P. Nawrocki and B. Sniezynski, Autonomous Context-Based Service Optimization in Mobile Cloud Computing, *Journal of Grid Computing*, Springer, Jul 2017.

[7] L. Chunlin, W.Y. Ping, T. Hengliang and L. Youlong, Dynamic multi objective optimized replica placement and migration strategies for SaaS applications in edge cloud, *Future Generation Computer Systems*, Elsevier, May 2019.

[8] Y. Luo, K. Yang, Q. Tang, J. Zhang, P. Li and S. Qiu, An optimal data service providing framework in cloud radio access network, *EURASIP Journal on Wireless Communications and Networking*, Springer, Jun 2016.

[9] T. Verbelen, T. Stevens, F.D. Turck and B. Dhoedt, Graph partitioning algorithms for optimizing software deployment in mobile cloud computing, *Future Generation Computer Systems*, Elsevier, July 2012.

[10] F. Palmieri, U. Fiore, S. Ricciardi and A. Castiglione, GRASP-based resource re-optimization for effective big data access in federated clouds, *Future Generation Computer Systems*, Elsevier, Jan 2015.

[11] I.A. Elgendy, W. Zhang, Y.-C. Tian and K. Li, Resource allocation and computation offloading with data security for mobile edge computing, *Future Generation Computer Systems*, Elsevier, May 2019.

[12] D. Oliveira, A. Brinkmann, N. Rosa and P. Macie, Performability Evaluation and Optimization of Workflow Applications in Cloud Environments, *Journal of Grid Computing*, Springer, Jan 2019.

[13] K. Peng, M. Zhu, Y. Zhang, L. Liu, J. Zhang, V.C.M. Leung and L. Zheng, An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing, *EURASIP Journal on Wireless Communications and Networking*, Springer, May 2019.

[14] M. Ficco, C. Esposito, F. Palmieri and A. Castiglion, A coral-reefs and Game Theory-based approach for optimizing elastic cloud resource allocation, *Future Generation Computer Systems*, Elsevier, May 2016.

[15] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang and S. Ou, Dynamic Resource Scheduling in Mobile EdgeCloud with Cloud Radio Access Network, *IEEE Transactions on Parallel and Distributed Systems* **29**(11), Nov. 1 2018.

[16] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A.S. Sani, D. Yuan and Y. Yang, A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment, *Future Generation Computer Systems*, Elsevier, Mar 2019.

[17] H. Baraki, A. Jah, S. Jakob, C. Schwarzbach, M. Fax and K. Geihs, Optimizing Applications for Mobile Cloud Computing Through MOCCAA, *Journal of Grid Computing*, Springer, Aug 2019.

[18] N. Zhang, X. Yang, M. Zhang and Y. Sun, Crowd-Funding: A New Resource Cooperation Mode for Mobile Cloud Computing, *PLOS ONE* | DOI:10.1371/journal.pone.0167657 December 28, 2016.

[19] C. Gao, X. Wang and M. Huang, A Cloud Resource Allocation Mechanism Based on Mean-Variance Optimization and Double Multi-Attribution Auction, *Network and Parallel Computing*, Springer, May 2013.

[20] K. Peng, V.C.M. Leung, X. Xu, L. Zheng, J. Wang and Q. Huang, A Survey on Mobile Edge Computing: Focusing on Service Adoption and Provision, *Hindawi Wireless Communications and Mobile Computing*, Oct 2018.

[21] R. Gracia-Tinedo, Y. Tian, J. Sampé, H. Harkous, J. Lenton, P. García-López, M. Sánchez-Artigas and M. Vukolic, Dissecting UbuntuOne: Autopsy of a Global-scale Personal Cloud Back-end, ACM Internet Measurement Conference (IMC'15), Tokyo, October 2015.