LP based Adaptive Resource Management Framework in Cloud Environment

Narander Kumar[1], Shalini Agrawal[2] and Vipin Saxena[3]

[1,2,3]Department of Computer Science,
B. B. Ambedkar University(A Central University),
Lucknow, UP, India.

Abstract

Computing may be based on cloud environment in 21st century. So cloud of the resources will become most important entity in next generation of computing environment. Therefore researchers' interest is going towards resource management in the cloud environment. However, much research has been done in this line but there is more scope of research in managing scalable resource requests by the end users under the constraint of budget. In this paper, we present an analytical model for a priority based dynamic resource management framework in cloud environment and a multi-index transportation problem cloud resource scheduler (MTPCRS) mechanism with mathematical formulation and an algorithm named Multi-Indexed Cloud Resource Scheduling Algorithm (MICRSA) with state transitions, deterministic finite automation and we present different test cases to evaluate and validate the results for calculating the total cost of processing requests as well as business process diagram and sequence diagram using unified modeling language (UML). A resiliency analysis is also presented to allocate the resources according to their priority and processing cost is also optimized. Finally, the proposed mechanism will be able to calculate the processing cost, time, profit of service providers as well as customers' benefits and optimum use of cloud resources of various clusters. We use *OMNET++* simulator to simulate the proposed mechanism.

Keywords: *Multi-index Cost Table, Millions of Instructions Per Second, Multi-index Transportation Problem, Return on Investment, Service Level Agreement, UML.*

1. Introduction

In next generation of computer networks, computing may be based on cloud environment due to hand held devices and different types of applications are being egressed. Therefore, requires very strong cloud environment to complete the processing within stipulated time. Therefore very large data centers with cooling systems are required. These data centers avail oneself of enormous amount of electrical energy and responsible for huge amount of carbon dioxide emissions per day. If under utilized servers of these data centers are put to energy saving mode then a surge of requests that come for processing to these servers render the decision of power saving mode to be faulty, as the work load suddenly increase. So, most data centers run at maximum capacity regardless of incoming requests and store the requested data violating the green IT rules

Generally cloud services are categorized as: software as service, platform as service and hardware as a service. These require a huge infrastructure as well as management policies. Most service provider or data centers aim to maximize their revenue, give less attention about admission control policies, scheduling the requests and allocation of resources.

We can say that the cloud environment is emerged as IT as a product as well as service. However some techniques and scheduling algorithms are given in literature but most of them being heuristic in nature, compromised energy consumption for attaining maximum achievable performance. So there is a scope to optimize their performance in respect of resources. With the help of operations research techniques, we optimize the performance of cloud computing system towards Green IT perspectives.

In particular, we summarize the main contributions of this paper as follows:

- An operation research technique named Multi-index Transportation based resource scheduling mechanism is provided to optimum use of resources in cloud computing.
- Mathematical formulation has been given using operation research techniques for the cloud environment.
- An algorithm named Multi-Indexed Cloud Resource Scheduling Algorithm (MICRSA) has been proposed.

- For the validation of proposed algorithm, we provide State Transition Diagram using Unified Modelling Language.
- Different test cases and automaton is provided for the evaluation of mathematical formulation.
- For the sake of simplification, we have provided a sequence diagram using UML.
- To show the usefulness of proposed mechanism, we present business process diagram using UML.
- Simulation on computer system of proposed mechanism and algorithm. We use OMNET++ simulator in cloud environment and evaluate the performance. An analysis of the performance of proposed system or mechanism has been found up to mark.

The remainder of this paper is organized as follows: In **section 2**, a brief review of the related work regarding optimum use of resources using Operation Research based techniques is provided. A proposed system model for the resources allocation process through Multi-index Transportation Problem Cloud Resource Scheduler (MTPCRS) has been presented in **section 3**. The mathematical formulation of MTPCRS using a generalization of the traditional Transportation Problem has been given in **section 4**. MICRSA algorithm, state transition diagram, DFA as well as test cases with significance have been given in **section 5**. **Section 6** shows the business perspectives with business process diagram of the proposed mechanism. The sequence diagram for the resource scheduling algorithm has been given in the **section 7**. System simulation model has been given in **section 8**. We discussed simulation results in **section 9**. Finally **section 10,** conclude the paper and given the future direction for further research.

2. Related Work

Optimizations in both cloud customers and providers are the key features of service delivery industry in cloud environment. In present as well as near future hosting infrastructures will serve as the basis for real world system. Network Input/output workloads, network Input/output resources, resource sharing effectiveness and impact of collocating applications which compete for either CPU or network resources is discussed [1]. Resilient Self-Compressive Monitoring system of cloud environment for hosting infrastructures. This system reduces the cost of remote data collection through on-line data compression. It provides failure resilience to achieve robust monitoring in dynamic distributed systems [2]. A layered scheme as cloud, server and GPGPU with hardware task features scheduling and execution mechanism is proposed in [3]. An architecture [4] with multilayer access control for cloud environment to provide isolation, information flow and resource sharing control in multiple visual infrastructures as well as security guarantee to configure the resources and deploy the application on the basis of logic virtual domain terminate an application, clean up the system of internal state and prevent the future failures through software rejuvenation. An optimal rejuvenation policy to maximize the availability of resources in dynamic work-load conditions has been proposed in [5].

Content distribution solution [6], where inter and intra cloud communication resources along with traditional colored computing resources are considered over different provider networked cloud environment. Predictive acknowledge and traffic redundancy elimination system does not require maintain the status of client at server and to maintain cloud elasticity, combine client mobility and server migration. Through Traffic redundancy chunks of chains it predicts future transmission of chunks [7]. A framework to optimize the cloud and network resources [8] has proposed to reduce the cost of information storage and computation. An analytical model and general methodology is the basis of utilization, availability, waiting time as well as responsiveness is presented in [9]. A distributed virtual Machine Multiplexing Resources Allocation scheme [10] has proposed to manage and optimize decentralized resources using the proportional share model as well as optimal execution efficiency and locating qualified nodes using a protocol named as Multi-attribute range query protocol. A mapping scheme for resource requests in a shared substrate interconnecting distinct islands of computing resources is present in [11].
Some algorithm is discussed in [12,13,14]. Service Level Agreement (SLA) constraints have been proposed in [15]. Soft resource scaling technique on power management policies is given in [16,17,18]. A model [19], to maximize the system capacity and find the power requirements with the help of LP. To minimize the cost with negligible power dissipation as well as re-engineered software and cooling systems to decrease wasted power is discussed in [20].The applicability of MTP for scheduling the resources has proposed in [21, 22, 23]. Different optimization problems about the providers, the consumers and the cloud (or Network) in cloud computing paradigm has described in [24]. The Rate controlled adaptive resource co-ordination and end-to-end QoS based admission control and access provider network interaction has been given in [25, 26].

In literature, the proposed mechanism or techniques are mostly heuristic in nature and more complex. They also compromised generally optimize use

of resources for attaining maximum achievable performance. XaaS provider's deployment, high performance is prime concern. Due to lack of these attentions towards the optimization of energy consumption. So there is required a mechanism or framework which provide the optimize use of resources according to priority based domain and maximize the net earning value as well as nature friendly without violating the green IT rules and regulations.

In the present paper, we propose a framework named an Adaptive Transportation Problem based dynamic resource management with budget constraints as well as the priority based domain in cloud environment which is an agreement towards using OR perspectives in cloud environment. The proposed model is based on multi index Transportation problem (MTP) according to the priority based class requests, which is an extension of Transportation problem itself.

3. Proposed System Model

For the resources allocation process, the subscribers issue requests for applications to the SaaS provider, then SaaS provider initiates the resource allocation process through reconfiguring in its resource allocation table to accommodate the incoming requests. The resource capacity of the IaaS provider and resource usage by incoming requests are characterized through the parameter i.e. the CPU usage. From this perspective the total CPU capacity (defined in Millions of Instructions Per Second (MIPS)), available at each provider is the 'commodity' to be transported. Each incoming request acts as a 'destination' where some amount of CPU capacity is required while the IaaS providers acts as the 'sources' whose responsibility is to fulfill the requirements of the 'sources' for this 'commodity'. Under balanced condition of TP, the total CPU capacity required by requests (destination) must be equal to the total CPU capacity available at the IaaS Provider.

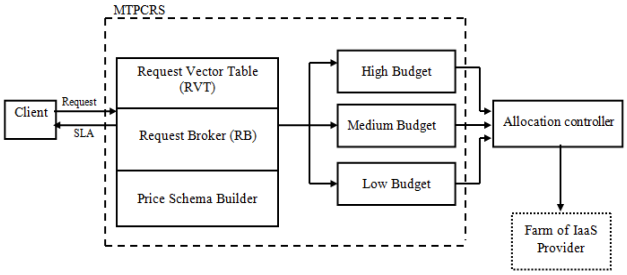


Figure 1: The System Model

The Multi-index Transportation Problem Cloud Resource Scheduler (MTPCRS) has four main components as shown in Figure 1. Each component of figure 1 is described as:

a) Request Vector Table (RVT):

Each incoming request has following parameters associated with it.

- Time (Tp) for which resource is needed.
- The cost of energy consumed by the physical server (Cp).
- CPU capacity (Z)
- Cost of running an application on a CPU of capacity Z (Tz)
- Number of Cores (S)
- Deadline (D)

All these parameters are stored in RVT.

One important significance of the RVT is that, it serves as a Service Level Agreement between the customer and the SaaS provider in case the new request is finally accepted by the SaaS provider. Hence the same table can be used for operational as well as archival purposes.

(b)The Request Broker (RB): The RB admits the new incoming request and stores it in a Request Vector Table (RVT). Apart from above discussed parameters, the RB also appends two more fields to the RVT.

- Budget (B) which is the total cost incurred by the user for executing its request.
- Last Info (I) which is a link to any previous information of the user held by the provider.

c) Price Schema Builder (PSB): Through PSB we calculate the budget B_i in order to execute the i^{th} request.

$$B_i = \sum_S (T_p (C_p + Z * T_z)) \text{ .....(1)}$$

According to the budget, the requests are divided into three classes viz. High, Medium and Low budget, according to their budget assigns priority to the requests. If the request i or customer i agrees with the budget B_i, the request is admitted and the customer waits for time quantum D otherwise the customer leaves the system and the corresponding entry is deleted from RVT.

The Total Budget of all accepted requests is calculated as:

$$B_{total} = \sum B_i \quad \text{.....(2)}$$

Once B_{total} has been calculated, the total processing cost Z is calculated through MTP scheduler by using a three index transportation table.

d) Resource Allocation Controller: After calculate the total budget (B_{total}) then allocation controller allocate the resources according to their priority which is given on the basis of budget (B_i). The mathematical formulation for computing Z (the total processing cost) is given in the next section.

4. Mathematical Formulation

We propose a mathematical model of MTPCRS using a generalization of the traditional Transportation Problem. In this model there are m clusters, n requests and p servers in each cluster. The servers with different processing capacity (MIPS) are arranged in the highest to lowest order, the cubical view given in fig 2. We have two-fold objective while allocating the resources to the requests. *First*, to allocate the resources to requests with minimum (the total) cost of processing on the servers of various clusters. *Second*, to minimize (the total) processing time of the requests, so that they can be completed within their deadlines.

Moreover, a fixed cost is incurred for every cluster which is the cost of its VM initialization and provisioning. This leads us to a Muli-Index Fixed Charge Bi-Criterion Transportation Model in order to obtain optimum resource management plan using Multi-index Cost Table (MCT). We define and explain the significance of all the indexes and variables associated with MCT.

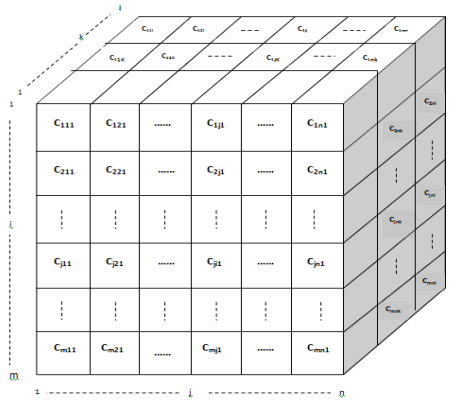


Figure 2: The cubical view of the different servers

i ε {1, 2, . . .,m}:- The index for cluster (origin);
j ε {1, 2, . . .,n}:- The index for request (destination);
k ε {1, 2, . . .,p}:-The index for server with different processing capacity(types of commodities);

x_{ijk} = Total CPU capacity utilized by j^{th} request running on the k^{th} server of i^{th} cluster (the amount of k^{th} type of commodity transported from i^{th} origin to the j^{th} destination.);

c_{ijk} = The per unit variable cost of processing j^{th} request running on the k^{th} server of i^{th} cluster (the variable cost per unit amount of k^{th} type of commodity transported from i^{th} origin to the j^{th} destination);

t_{ijk} = The time taken for processing j^{th} request running on the k^{th} server of i^{th} cluster (the time required to transport k^{th} type of commodity from the i^{th} origin to j^{th} destination) cluster j;

f_{ijk} = The fixed cost of using k^{th} server on i^{th} cluster by any of the i^{th} request (the fixed cost associated with origin i and commodity k for a destination j). $\{f_{ijk} = 0$ if $x_{ijk} = 0\}$

A_{jk} = The amount of CPU capacity of k^{th} server, utilized by j^{th} request (the total quantity of k^{th} type of commodity to be sent to the j^{th} destination);

B_{ki} = The total CPU capacity of k^{th} server available at i^{th} cluster (the total quantity of k^{th} type of commodity available at the i^{th} origin);

E_{ij} = The total CPU capacity to be allocated from the i^{th} cluster to the j^{th} request (the total quantity to be sent from i^{th} origin to the j^{th} destination

Z = Total Cost of Processing all the requests

t = Total time taken to process all the requests

P = Total Profit of the service provider

We set priorities to the requests as-
PRIORITY = 1 if the request is a high budget request
PRIORITY = 2 if the request is a medium budget
PRIORITY = 3 if the request is a low budget
(Where 1 is taken as highest priority and 3 is taken as the lowest)

We now give the mathematical model to formulate of the objective function of our model which is as follows:

Minimize

$$z = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ijk}x_{ijk} + \sum_{i=1}^{m}\sum_{k=1}^{p} F_{ik} \quad \ldots\ldots\ldots\ldots (3)$$

And

$$\text{Minimize} \quad t = \max\{t_{ijk}\} \quad \ldots\ldots\ldots\ldots (4)$$

$$\text{Subject to} \quad \sum_{i=1}^{m} x_{ijk} = A_{jk} \quad \ldots\ldots\ldots\ldots (5)$$

$$\sum_{j=1}^{n} x_{ijk} = B_{ki} \quad \ldots\ldots\ldots\ldots (6)$$

$$\sum_{k=1}^{p} x_{ijk} = E_{ij} \quad \ldots\ldots\ldots\ldots (7)$$

$$\sum_{j=1}^{n} A_{jk} = \sum_{i=1}^{m} B_{ki} \quad \text{..... (8)}$$

$$\sum_{k=1}^{p} B_{ki} = \sum_{j=1}^{n} E_{ij} \quad \text{..... (9)}$$

$$\sum_{i=1}^{m} E_{ij} = \sum_{k=1}^{p} A_{jk} \quad \text{.....(10)}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{p} A_{jk} = \sum_{k=1}^{p}\sum_{i=1}^{m} B_{kj} = \sum_{i=1}^{m}\sum_{j=1}^{n} E_{ij} \quad \text{..... (11)}$$

$$x_{ijk} >= 0, i = 1,2,...m; j = 1,2,...n; k = 1,2,...p \quad \text{.....(12)}$$

$$P = B_{total} - Z \quad \text{..... (13)}$$

5. Multi-index Cloud Resource Scheduling Algorithm (MICRSA)

We now present the algorithm to obtain a resource allocation plan to the requests which minimizes total processing cost as well as time. The variable and fixed cost model we have taken helps the cloud provider to offer several types of payment structures as per the customer's requirement.

. Input requests (n), clusters (m), and different processing capacity servers/resources (p), the number of k^{th} type of resources required by the j^{th} request (A_{jk}), the number of k^{th} type of resource available at the i^{th} cluster (B_{ki}), the number of resources to be allocated from i^{th} cluster to the j^{th} request (E_{ij}), the variable cost of processing i^{th} request on k^{th} server of j^{th} cluster (c_{ijk}),the fixed cost incurred by the i^{th} cluster for server(resource) k (F_{ik}), estimated time of processing i^{th} request on the k^{th} server of j^{th} cluster (t_{ijk}), total allocated budget of processing requests by the proposed system (Btotal), integer variables H,M,L.

2. (i) Arranged the index k ε 1 to p in non- increasing order.
(ii)Set H=0, M=0, L=0

3. Insert variable cost estimation C_{ijk}, Fixed cost estimation F_{ik} and time estimation t_{ijk}, in each cell of Multi-index Cost Table (MCT).

```
4. For i= 1 to j do
    {
      If Ri = High Budget
      {
       Set PRIORITY = 1
       H=H+1 /* count the number of high
       budget requests */
       }
      If Ri = Medium Budget
       {
       Set PRIORITY= 2
       M=M+1 /* count the number of medium
      budget requests */
       }

      If Ri = Low Budget
       {
       Set PRIORITY =3
       L=L+1  /*count  the  number  of  low
       budget requests*/
       }
     } /*End For*/
/*Repeat  allocation  for  each  type  of
      resource */
 5. while k<>0 do
      {
       while H <> O do /*Resources are
       allocated to requests in priority
       order*/
       {
       (i)   Locate MCT [i, j, k] for
             which  c[i,  j,  k]  is
             minimum.
       (ii)  If  two  cells  with  same
             minimum  cost  occur,  then
             select the one whose column
             has  maximum  number  of
             allocations.
       (iii) Allocate as much resource
             as possible to the request.
        (iv)       H=H-1
           } /*end while*/

                    while M <> O do
                    {
          (i)   Locate  MCT  [i,  j,  k]
          for  which  c[i,  j,  k]  is
          minimum.
          (ii)  If  two  cells  with  same
                minimum cost occur, then
                select  the  one  whose
                column  has  maximum
                number of allocations.
          (iii)   Allocate  as  much
                resource as possible to
                the request.
          (iv)  M=M-1
                    } /*end while*/

                    while L <> O do
                    {
          (i) Locate MCT [i, j, k] for
               which  c[i,  j,  k]  is
               minimum.
          (ii) If  two  cells  with  same
               minimum cost occur, then
               select  the  one  whose
               column  has  maximum
               number of allocations.
```

```
            (iii)Allocate as much resource
                 as    possible    to    the
                 request.
            (iv)  L=L-1
                                    }/*     end
       while */
  k=k-1
     } /*end while*/
```

6. Calculate Total Processing Cost using equation (3).
7. Calculate total time for processing all the requests using equation (4).
8. Calculate Total Profit using equation (13)

A. State Transition Diagram

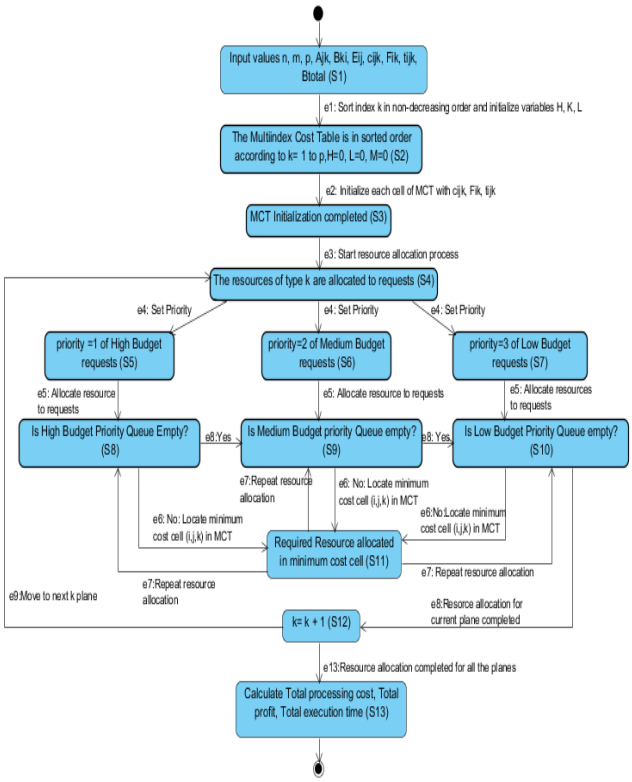


Figure3: State Transition Diagram

The state transition diagram for proposed algorithm is represents in the figure 3. The initial state (S_1), represents the point when number of requests n have been accepted by the Request Broker mentioned in the figure 1 of the system model. After the SLA has been negotiated with the user, the cloud resource scheduler will attempt to allocate resources to the requests in the most cost and time efficient manner. The numbers of states are S_1 (initial) to S_{13} (final) which are according to the algorithm steps.

B. Deterministic Finite Automaton

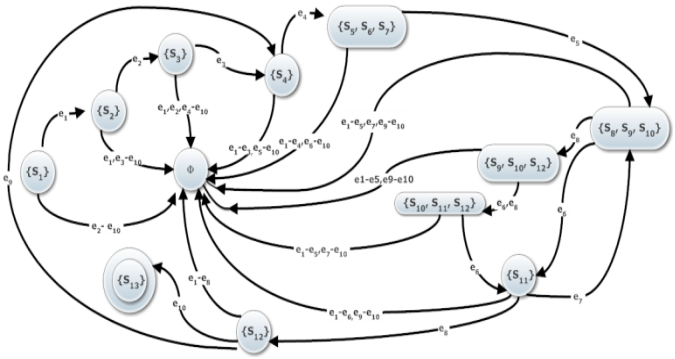


Figure 4: The Deterministic Finite Automaton

The equivalent Deterministic Finite Automaton (DFA) has been obtained as in the figure 4. Those states that are unreachable from a given state take the automaton to the state containing the null set. The figure shows that the DFA repeats the allocation process for the requests in the priority order. Hence least cost cell of MCT are occupied by the high priority (high budget) requests first as against the medium and low priority. This improves the total processing cost and helps to maximize the profit.

C. **Test Cases**

To represent the test cases, we consider that the number of requests entering the system is 4 viz R1, R2, R3, R4 (i=4). There are 3 clusters (j=3), each having 3 types of servers (k=3). Request number R1 and R2 are high priority, R3 is medium priority and R4 is low priority. In each cell (i, j, k) of the multi-index cost table, the value of c_{ijk} is given at the top left corner and t_{ijk} is given at the bottom right corner. The units allocated to the request are mentioned in brackets. We assume that B_{total} = 2000.

Significance:

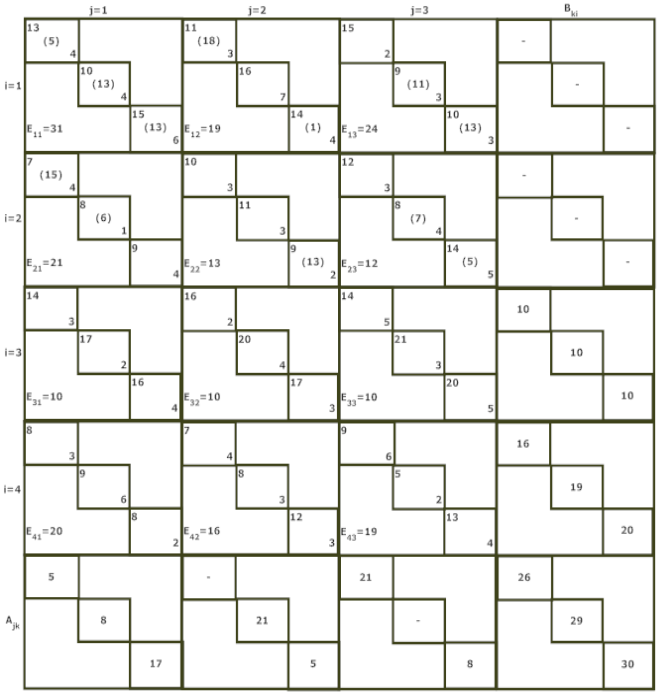
1) After transition $S_1e_1S_2e_2S_3e_3S_4e_4S_5e_5S_8e_7S_{11}e_6S_8e_8S_9$ we arrive at table A which represents the scenario when all the high budget requests have processed.

2) After transition $S_1e_1S_2e_2S_3e_3S_4e_4S_6e_5S_9e_6S_{11}e_7S_9e_8S_{10}$ we arrive at table B which represents the scenario when all the medium budget requests have processed.

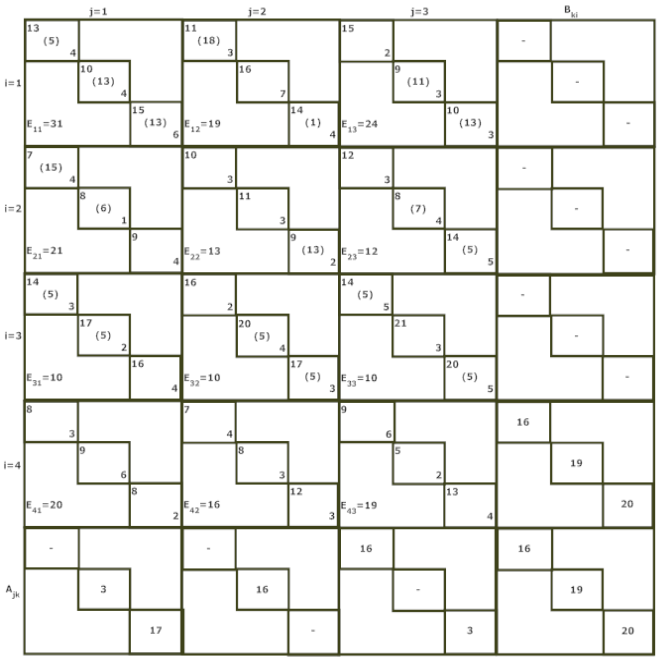
3) After transition $S_1e_1S_2e_2S_3e_3S_4e_4S_7e_5S_{10}e_6S_{11}e_7S_{10}e_8S_{12}$ we arrive at table **C** which represents the scenario when all the low budget requests have processed.

4) In order to reach the final state we take the transition $S_1e_1S_2e_2S_3e_3S_4e_4S_7e_5S_{10}e_8S_{12}e_9S_4e_4S_7e_5S_{10}e_8S_{12}e_{13}S_{13}$. Here Z, t, P are calculated. For the computation of Z, the fixed costs are given in Table D.

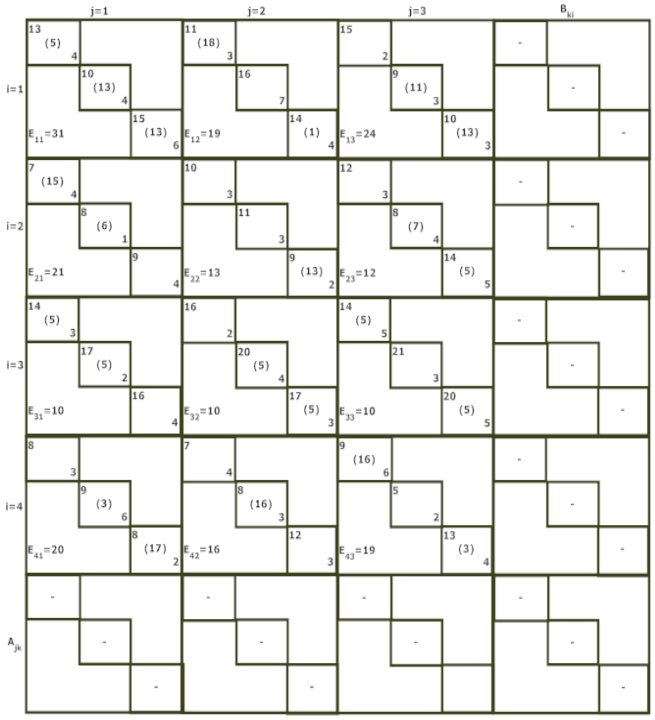
TEST CASE 1: TABLE A



TEST CASE 2: TABLE B



TEST CASE 3: TABLE C



TEST CASE 3: TABLE D

f_{111}= 15	f_{121}=21	f_{131}=16
f_{112}=32	f_{122}=10	f_{132}=23
f_{113}=25	f_{123}=16	f_{133}=10
f_{211}=5	f_{221}=12	f_{231}=20
f_{212}=41	f_{222}=34	f_{232}=11
f_{213}=17	f_{223}=23	f_{233}=15
f_{311}=10	f_{321}=15	f_{331}=16
f_{312}=9	f_{322}=11	f_{332}=26
f_{313}=28	f_{323}=21	f_{333}=5
f_{411}=24	f_{421}=10	f_{431}=18
f_{412}=23	f_{422}=15	f_{432}=12
f_{413}=10	f_{423}=32	f_{433}=12

For the sake of explanation of Test Case 1 (Table-A), i, j represents the clusters and requests respectively to the process the parameters. A cell (i, j=1) has three divisions which represents number of servers (resource types) in each cluster. In each internal division, the cost of processing the request (c_{ijk}) is mentioned on the upper left corner and the processing time (t_{ijk}) is mentioned on the lower right corner. The number of units allocated (x_{ijk}) if it is done, is mentioned in brackets at the centre. Similarly the rest of the cells of the table assigned numerical values as explained above. Column B_{ki} represents the requirement of the requests and Row A_{jk} represents the availability of the resources. Last cell of B_{ki}

and A_{jk} represents that the total availability of the resources.

For eg., within the cell(1,1) the upper left corner division represents k=1, the centre division represents k=2 and the lower right division represents k=3. E_{11} represents total amount of all resource types that can be allocated to R1 (j=1) by cluster 1(i=1) on any of its servers (k=1,2,3).

The allocation process is done iteratively such that total resource requirements are fulfilled in accordance to resource availability, until all requests are assigned their resources. In the current example the first iteration (Table A) shows the resource allocation of high priority requests. The second iteration (Table B) shows the resource allocation of medium priority requests and the final iteration (Table C) shows the allocation of low priority requests.

With the help of the final table, Table C and the fixed cost table, Table D, we compute total cost of processing all the requests (Z), by using equation (3) in section IV. The total processing time t is obtained by summing up t_{ijk} values of Table C, where allocations are made. The total profit (P) earned is obtained by using equation 13 in section IV.

Total cost of processing all the requests (Z) = 1591

The total processing time (t) = 6

The total profit earned (P) =409

6. The Business Perspective

The Cloud Computing has enhanced the 'cohesiveness' of people on the earth. Nevertheless, this benefit comes with plethora of issues which need serious consideration. Organizations increasingly discover that their substantial capital investments in Information Technology are often grossly under utilized [27]. Gartner research has shown that maintenance and housekeeping budgets are overwhelmingly larger than computing costs. There is a perpetual strive for attaining estimated Return On Investment (ROI), which is widely unfulfilled due to heavy power dissipation. In our proposed model, we have divided the incoming requests into three classes: high, medium and low, which considerably reduce bandwidth requirements. Also, the scheduler optimizes power requirements by first mapping requests to those clusters which have the highest CPU capacity available at that time. (This indicates under-utilized servers with lesser possibility of initiating new VM). The scheduler has an easy access to a large backbone of IaaS servers with built in redundancy checks.

The business process diagram shows in the figure 5, that our proposed model is simple with easy workflow and process communication. The simplicity in design reduces the complexity of implementation.

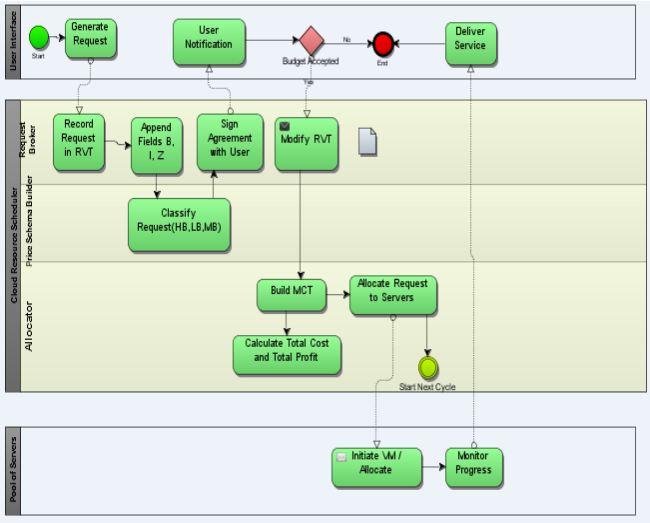


Figure5: Business Process Diagram

The major participants in the model are shown in three pools. The first pool represents the end user interface, the second represents the cloud resource scheduler and the third represents the backbone of IaaS servers. The processes in each of these pools are described below.

I) User Interface Pool: This pool has a single lane consisting of the following processes-

i) The Generate Request process: This process handles random generation of requests from the end users. It allows the users to submit the request specifications to the scheduler in a format prescribed by the SaaS provider. The process collects all requests specifications and redirects them to be recorded in the Request Vector Table (RVT) of the scheduler.

ii) The User Notification Process: Once the request has been admitted for execution, this process is responsible for all further communication between the user and the cloud resource scheduler. It forwards the SLA document to the user for final approval, so that further execution can be started.

iii) The Deliver Service Process: This process provides final results of execution of a request to the user. It acts as interface between the backbone of service providers and the users to make the desired application available to the user. It may be noted here, that a single request may be

running on several VMs in the cloud environment. Hence, this process provides abstraction to the user by collecting execution results from several places and delivering them to the final destination.

iv) *The Start Event marks* the beginning point in the model, where user enters the system.

v) *The End Event marks* the exit point where the user leaves the system, either after the fulfillment of the request or because of non-acceptance of the budget proposed by the scheduler.

***II) Cloud Resource Scheduler Pool*:** This pool has three lanes which mark the three main components of the cloud resource scheduler. The processes in each of these lanes are as under-

(a) The Request Broker Lane describes the processes for request profiling.

i) *The Record Request process:* It saves and manages incoming request entries into Request Vector Table. This process marks the origin of SLA description between the user and the service provider. It provides input the next process in the lane which appends necessary fields to the RVT.

ii) *The Append Fields (B, I, Z):* This process is responsible for revoking from archive, any previous association from the user; and if so, adding it (field I) to the RVT, so that the price schema builder could offer some discount structure in the budget, to its old customers, if any. It also adds the initial budget B and processing capacity Z to the RVT.

iii) *The Sign Agreement Process:* This process communicates with the User Notification process described above, for final agreement on budget or SLA.

iv) The Modify RVT Process: After the final budget has been calculated and request has been classified by request classifier of the next lane, this process makes final changes to the RVT. Each entry of the RVT is saved as individual copy of SLA.

(b) The Price Schema Builder Lane consists of the single process as described below:

The Classify Request Process: It handles the final classification of request as High Budget, Medium Budget or Low Budget. As shown in the business process diagram, the input to this process is Append Field (B, I, Z) and output is directed to Sign Agreement process.

(c) The Allocator Lane has processes dealing with the final mapping of user requests to the servers. The description of each process is as under-

i) *The Build MCT Process:* This process creates multi-index cost table in the form of a cube. It inserts all cost values into the cells of the cube for final selection of servers according to MTPCRSA.

ii) Calculate Total Cost and Total Profit Process calculates total processing cost and total profit using equations (1) and (2) in section III.

iii) Allocate Requests to Servers Process handles final mapping of requests to the backbone of servers in the MCT according to MTPCRSA.

III) The Pool of Servers: This is a single lane pool represents the backbone of clusters in the cloud environment. Two processes are defined in this pool-

i) Initiate VM/Allocate Process handles physical allocation of requests to clusters. In case the designated server in the assigned cluster has not been instantiated, a new VM is initiated for execution of the request.

ii) The Monitor Progress Process works as a supervisor and forwards the result of execution to the user via Deliver Service Process in the User Interface Lane.

7. Sequence Diagram

The sequence diagram shown in figure 6 is used to develop the resource scheduling algorithm. We now explain the sequence of events as shown in figure 4. The numbers in the parenthesis show the event numbers. On the first event (1), user generates request to the SaaS provider. On the second event (2), the SaaS forwards the request to Request Broker to check admissibility of the request. The Request Broker asks the Price Builder to calculate the budget on the third event (3). The Price Builder returns the calculated budget to the Request Broker on event (4). On the fifth event (5), the Request Broker prepares the SLA and forwards it to the SaaS provider for the user's approval. Next, the SaaS provider notifies this SLA to the user as shown on event (6). Once the user approves the SLA, the Request Broker sends the Request Vector table to the MCT (8), to build the initial cost table. On the eighth event (8), the MCT Builder initializes values of the MCT and maps requests to the available servers to forward it to the Scheduler. The Scheduler passes on the allocation schedule to the physical Pool of IaaS servers on the ninth event (9). The

IaaS servers send message to SaaS provider to ask for confirmation before executing requests, on event (10). In the subsequent event (11), the SaaS provider prompts for payment of charges from the user. In response to this, the user pays service charges to the SaaS provider on event (12). Following this, event (13) starts the execution of requests on the servers. On event (14), the IaaS servers finish the execution and notify the SaaS provider that the SLA has been fulfilled. The final event (15) marks the final delivery of required services to the user.

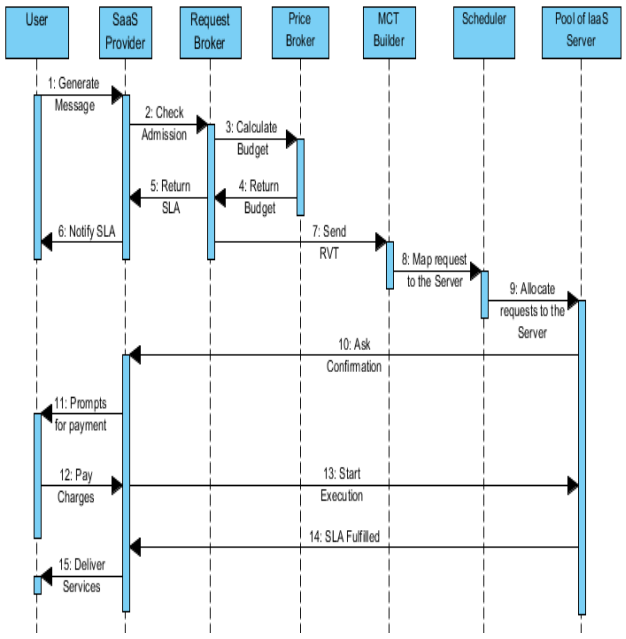


Figure 6: Sequence Diagram

8. Proposed System Simulation Model

The framework prioritizes the requests into three different budget queues according to their importance to the system, named as Budget[i] {i=0,1,2} in figure 7. Each partition or queue named budget[i] is reserved for each class of requests. Let the inter-arrival time of high, medium and low budget requests be exponential.

For the sake of explanation, we consider an allocation controller, which is provided the services according to their priority. Allocation Controller is also responsible to decide the minimum cost server among the several servers which are available in different clusters of servers. This is done by maintaining the cost table. Then the high, medium and low priority requests are mapped (according to their priority) to the respective servers of cluster dynamically. After processing the requests exit at the point named, Processed_H, Processed_M and Processed_L respectively. The proposed mechanism is capable of handling different service time.

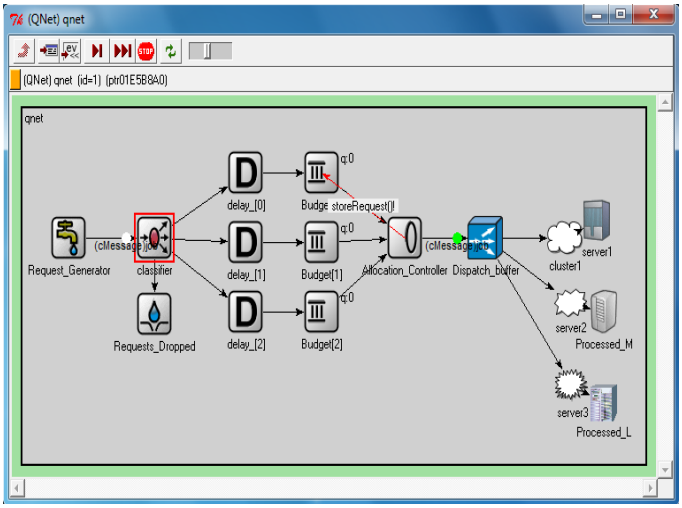


Figure 7: Simulation Model

9. Simulation Results and Discussion

Our simulation results show that:-

1) There is no dropping of requests at Rejected_Requests. Rejected_Requests is the dropping queue of the requests. Since there is no dropping of requests as from figure 6, therefore no request will be lost and no penalty would be paid by the system framework and system framework performance would be increased and it may lead to be ideal.

2) Performance of IaaS (Server1 and Server 2) evaluation (figure 7 and figure 8) shows that the high-budget and medium-budget requests are being processed according with almost no delay. (Other requests are processed exponentially with respect to their requirements). There can be situations of high traffic load where the total time of the cell can be more than the sum of maximum time of all the running requests and goes down as traffic smoothens.

3) Performance of IaaS (server 3) evaluation (figure 9) shows low-budget requests are being processed. In this scenario, the requests are processed exponentially with respect to their requirements.

4) From the analytical (model) results we calculated the processing cost of requests and provided the services in the simulation accordingly.

5) Test cases results obtained from Finite State Machine (FSM) also shows the better performance and able to solve the problem of optimize resource scheduling in cloud environment.

Thus these above discussed observations lead to a better performance of proposed analytical model for dynamic resource management framework in cloud computing environment and hence better results and smooth system performance. All the results are simulated on OmNetPP simulator.

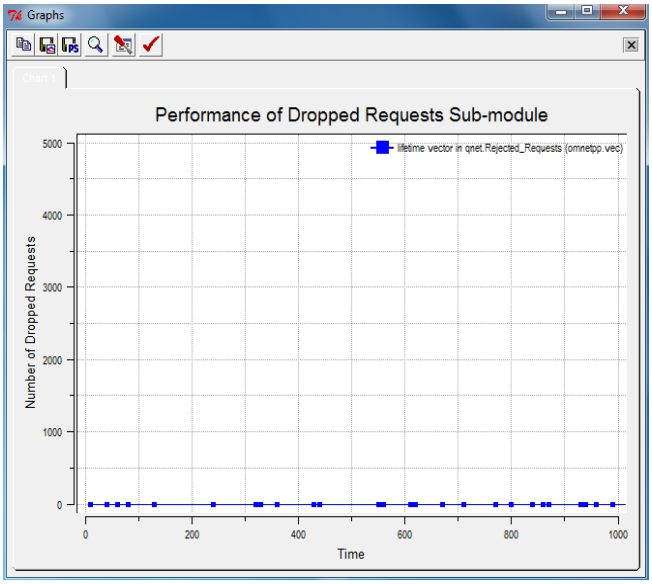


Figure 8: Performance of Dropped Requests

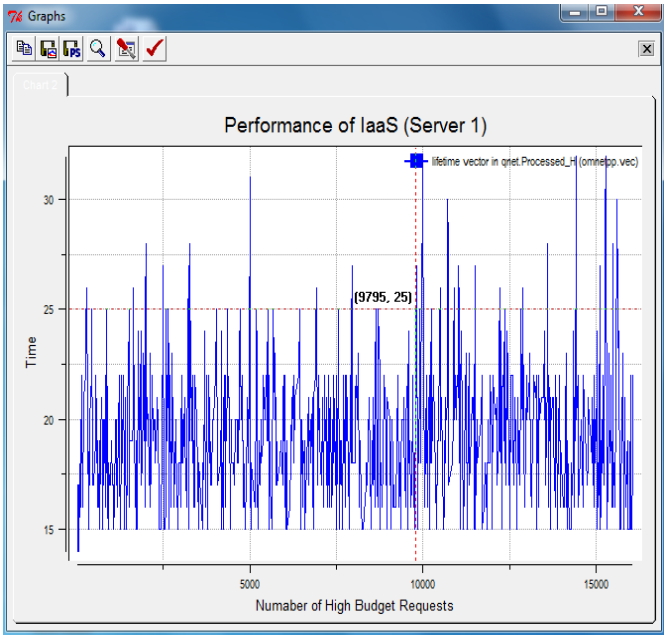


Figure 9: Performance of IaaS (Server 1)

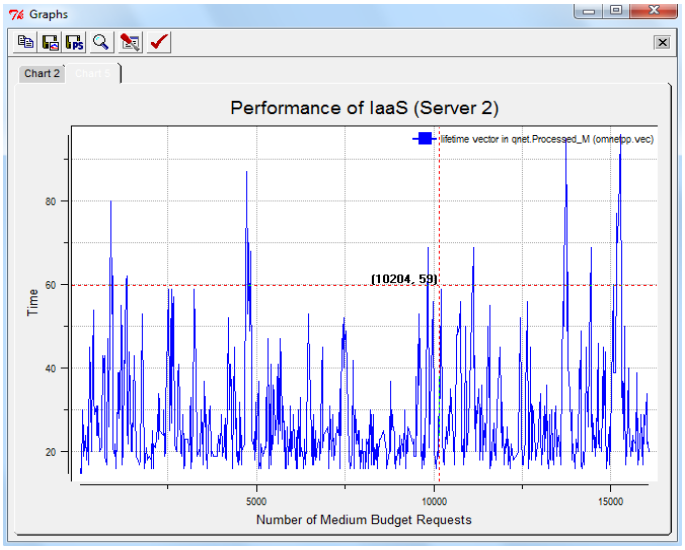


Figure 10: Performance of IaaS (Server 2)

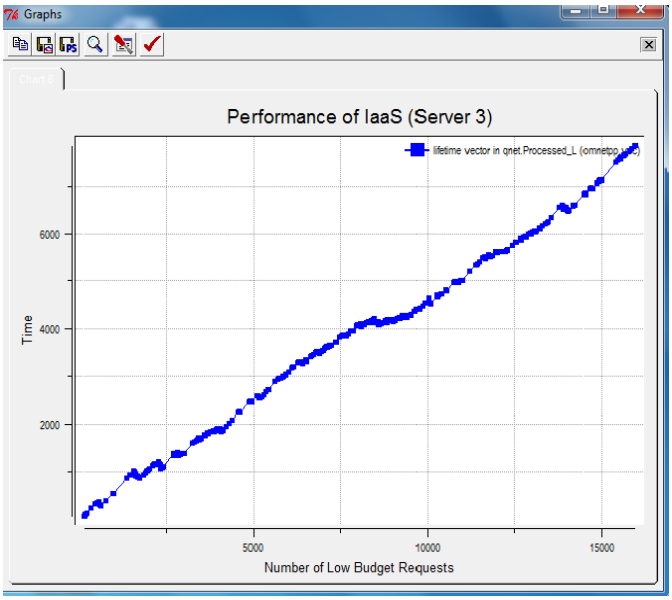


Figure 11: Performance of IaaS (Server 3)

10. Conclusion and Future Directions

The classical transportation problem based cloud resource scheduling has been suggested in literature. However, the model has a limitation of considering all resources of the same type, which may not be so in any cloud environment. We extend the basic model to build MTPCRS by introducing a third index, i.e., processing capacity of the server. We provide the mathematical formulation required to model a cloud resource scheduler using multi-index transportation problem technique. So that it is necessary to develop a dynamic and adaptive

algorithm to improve optimum resource management in order to avoid virtual machine migration and consolidation problems. The algorithm assigns requests to servers on different clusters based on request priority. A state transition diagram and deterministic finite automaton of the problem has been developed, through the test cases we evaluate the performance of the proposed mechanism. We have suggested its significance in business perspectives using business process diagram. The results of simulation show that the mechanism works well to comply with SLA and maximizes profits, both customers as well as the provider. As a future direction, it may be attempt to implement the algorithm by incorporating QoS concern on service level agreement as well as different type of computing environment using other Operations Research techniques such as GOAL Programming, Dynamic Programming and NLPP.

References :

[1] Mei, Yiduo, Liu, Ling, Pu, Xing, Sivathanu, Sankaran, Dong, Xiaoshe, "Performance Analysis of Network I/O Workloads in Virtualized Data Centers," *Services Computing, IEEE Transactions on*, vol.6, no.1, pp.48,63, First Quarter 2013. doi: 10.1109/TSC.2011.36

[2] Yongmin Tan, Venkatesh, V., Xiaohui Gu, "Resilient Self-Compressive Monitoring for Large-Scale Hosting Infrastructures," *Parallel and Distributed Systems, IEEE Transactions on*, vol.24, no.3, pp.576, 586, March 2013. doi: 10.1109/TPDS.2012.167

[3] Hu, Liang; Che, Xilong, Xie, Zhenzhen, "GPGPU cloud: A paradigm for general purpose computing," *Tsinghua Science and Technology*, vol.18, no.1, pp.22, 23, Feb. 2013. doi:10.1109/TST.2013.6449404.

[4] Qiang, W., Zou, D., Wang, S., Yang, L.T., Jin, H., Shi, L., "CloudAC: a cloud-oriented multilayer access control system for logic virtual domain," *Information Security, IET*, vol.7, no.1, pp.51, 59, March 2013. doi: 10.1049/iet-ifs.2012.0094

[5] Bruneo, D., Distefano, S., Longo, F., Puliafito, A., Scarpa, M., "Workload-Based Software Rejuvenation in Cloud Systems," *Computers, IEEE Transactions on* , vol.PP, no.99, pp.1,1, 0 doi: 10.1109/TC.2013.30

[6] Papagianni, C., Leivadeas, A., Papavassiliou, S., "A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment," *Dependable and Secure Computing, IEEE Transactions on*, vol. PP, no. 99, pp.1, 1, 0 doi: 10.1109/TDSC.2013.12

[7] Zohar, E., Cidon, I., Mokryn, O., "PACK: Prediction-Based Cloud Bandwidth and Cost Reduction System," *Networking, IEEE/ACM Transactions on*, vol. PP, no.99, pp.1,1, 0 doi: 10.1109/TNET.2013.2240010

[8] Fang, X., Yang, D., Xue, G., "Evolving Smart Grid Information Management Cloudward: A Cloud Optimization Perspective," *Smart Grid, IEEE Transactions on* , vol.4, no.1, pp.111, 119, March 2013, doi: 10.1109/TSG.2012.2230198

[9] Bruneo, D., "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol.PP, no.99, pp.1,1, 0 doi: 10.1109/TPDS.2013.67

[10] Sheng Di, Cho-Li Wang, "Dynamic Optimization of Multiattribute Resource Allocation in Self-Organizing Clouds," *Parallel and Distributed Systems, IEEE Transactions on*, vol.24, no.3, pp.464,478, March 2013. doi: 10.1109/TPDS.2012.144

[11] Papagianni, C., Leivadeas, A., Papavassiliou, S., Maglaris, V., Cervell\ó-Pastor, C., Monje, A., "On the Optimal Allocation of Virtual Resources in Cloud Computing Networks," *Computers, IEEE Transactions on* , vol.PP, no.99, pp.1,1, 0, doi: 10.1109/TC.2013.31

[12] Senthil Kumar, S.K., P. Balasubramanie, "Dynamic Scheduling for Cloud Reliability using Transportation Problem", *Journal of Computer Science* 8 (10): 1615-1626, 2012, ISSN 1549-3636.

[13] Shahaf I.Wayer and Arie Reichman, "Resource Management in Satellite Communication Systems: Heuristic Schemes and Algorithms", *Journal of Electrical and Computer Engineering*, Volume 2012, Article ID 169026.

[14] Bautista, L., A. Abran "Design of A Performance Measurement Framework For Cloud Computing", *J. Software Engg. Appli*., 5: 69-75. DOI: 10.4236/jsea.2012.52011.

[15] Buyya, R., S.K. Garg and R.N. Calheiros, "SLA Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture and Solutions", Proceedings of the *International Conference on Cloud and Service Computing, IEEE*, Australia, pp: 1-10, 2011.

[16] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya,"Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing", *Journal of Future Generation Computer Systems,* (2012), pp-755–768.

[17] Anton Beloglazov, Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers", *Concurrency And Computation: Practice And Experience,;* 24:1397–1420, Published online in *Wiley InterScience*, 2012, DOI: 10.1002/cpe.1867.

[18] R. Nathuji, K. Schwan, "Virtual Power: Coordinated Power Panagement in Virtualized Enterprise Systems", *ACM SIGOPS Operating Systems Review* 41 (6) (2007) pp-265–278.

[19] Al-Azzoni, Douglas G. Down, "Linear Programming Based Affinity Scheduling for Heterogeneous Computing Systems", *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 19, No. 12, Dec 2008, pp 1671-1682.

[20] Johan Tordsson, Rubén S. Montero, Rafael Moreno-Vozmediano, Ignacio M. Llorente, "Cloud Brokering Mechanisms For Optimized Placement Of Virtual Machines Across Multiple Providers", *Future Generation Computer Systems* 28 (2012), pp-358–367.

[21] Lixing Yang a, Yuan Feng , "A Bicriteria Solid Transportation Problem With Fixed Charge Under Stochastic Environment", *Applied Mathematical Modelling* 31 (2007), pp-2668-2683.

[22] Aneja, Y.P., and Nair, K.P.K., "Bicriteria Transportation Problems", *Management Science* 25, (1979), pp-73-78.

[23] B. Liu, "Uncertain Programing", *Wiley*, New York, 1999.

[24] Ilyas Iyoob, Emrah Zarifoglu, and A. B. Dieker, "Cloud Computing Operations Research" *Service Science serv*.1120.0038; published online before print March 1, 2013, *doi:10.1287/serv.1120.0038*

[25] N. Kumar, YDS Arya, "Rate Controllrf Adaptive Resource Coordination Framework for Wireless aware Multimedia Application", International Journal of Computer System and Network Security, Vol. 10, Issue 12, pp 99-105, 2010,

[26] R. Singh, N. Kumar, Verma, S., "End to End QoS-based Admission Control and APN Interaction Framework with Negotiation", Proceedings of ISCA, CATA-2009, New Orleans, Louisiana, USA, pp 145-150, ISCA-2009.

[27] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, Anand Ghalsasi, "Cloud Computing-The business perspective", *Decision Support Systems* 51 (2011), pp-176–189.

[25] Saurabh Saxena, Energy Saving using Cloud Computing"," International Journal of Computational Engineering and Management IJCEM, Vol. 15 issue 6, November 2012.

Authors' Profile

1. Narander Kumar received his Post Graduate Degree and Ph. D. in CS & IT, from the Department of Computer Science and Information Technology, Faculty of Engineering and Technology, M. J. P. Rohilkhand University, Bareilly, Uttar Pradesh, INDIA in 2002 and 2009, respectively. His current research interest includes Quality of Service (QoS), Computer Networks, Resource Management Mechanism, in the networks for Multimedia Applications, Performance Evaluation, Cloud Computing, Software Engineering,. Presently he is working as Assistant Professor, in the Department of Computer Science, Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, UP, INDIA.

2. Shalini Agarwal received her Master of Technology (M. Tech) in Computer Science and Engineering from Amity University, Uttar Pradesh, India in 2010. She also did her M.Sc (Computer Science) at J. K. Institute of Applied Physics and Technology, University of Allahabad, Allahabad, UP, India. She was a faculty member in the Department of Computer Science and Engineering, Sri Ram Swaroop Group of Professional Colleges, Lucknow, UP, India. Currently, she is a research scholar in the department of Computer Science at Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, UP. India. Her research area is computer networking and resource management for grid and cloud computing infrastructures.

3. Vipin Saxena is a Professor and Head, Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India. He got his M.Phil. Degree in Computer Application in 1992 & Ph.D. Degree work on Scientific Computing from University of Roorkee (renamed as Indian Institute of Technology, Roorkee, India) in 1997. He has more than 17 years of teaching experience and 20 years of research experience in the field of Scientific Computing & Software Engineering. He has published more than hundred International and National research papers and authored four books in the Computer Science field. Dr. Saxena is a life time member of Indian Science Congress