Research Journal of Recent Sciences _____ ISSN 2277-2502
Vol. 3**(IVC-2014),** 1-6 **(2014)** *Res. J. Recent. Sci.*

An Analytical Model for Dynamic Resource Allocation Framework in Cloud Environment

Kumar N. and Agarwal S.
Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, UP, INDIA

Available online at: www.isca.in, www.isca.me
Received 22rd June 2014, revised 15th August 2014, accepted 7th September 2014

Abstract

Cloud computing has emerged as the most popular paradigm for on-demand, pay-per-use model of computing. The software, platform and infrastructure as a service model will become the most popular mode of getting computing resources by common users. There has been growing research interest in managing the cloud of resources so as to achieve optimum utilization of resources along with desired quality of service. In the present scenario there is much scope of research in mapping users' request to appropriate servers in cloud computing environment. In this paper, the authors propose an analytical model that maps dynamic users' request to physical servers in the cloud that is based on a fixed charge mutli-index transportation problem. Thus a multi-index transportation Problem Cloud Resource Scheduler (MTPCRS) mechanism with mathematical formulation is developed along with a numerical example. A Multi- Indexed Cloud Resource Scheduling Algorithm (MICRSA) is also given in order to calculate the total cost of processing the service requests. With the help of sequence diagram and business process diagram it is shown that the model is simple to implement and produces an efficient and cost effective resource allocation plan for satisfying users' requests.

Keywords: Muti-index transportation problem, multi-index cost table, resource allocation, services, virtual machine.

Introduction

A lot of research has been done for resources management in Cloud environment so as to fairly divide available servers as the requirements of the user's request. As users' requests are dynamic in nature, several virtual machines are instantiated on the servers to handle the requests[1,2]. The numbers of virtual machines vary from server to server as requests are processed from time to time. The objective in virtual machine allocation is to accept as much requests as possible[3]. The common theme in proposed work in literature is to design an algorithm which has optimal performance and gives highest satisfaction to the users. User's satisfaction is often measured as to what extent the services are in compliance with the terms written in Service Level Agreement (SLA). Cloud services, which can be generally categorized as: Software as Service (SaaS), Platform as a Service (PaaS) and hardware resource or Infrastructure as a service (IaaS), require heavy investments in terms of infrastructure and management policies. Regardless of the way of admission control and scheduling and allocation of resources, all cloud service providers aim to maximize their profits. In some research, the problem of resource allocation has been modeled as Multi-Knapsack problem (MKP) while others have suggested several extensions to MKP. Several other cost effective admission control and scheduling algorithms have been proposed to maximize XaaS provider's profit and customer satisfaction level. Genetic algorithm based scheduling policies have difficulty dealing with 'deceptive' fitness functions, honey bee algorithms need a lot of time to get trained and react on the situation[4]. In addition, the given approaches also have high complexity due to which they often depart from attaining maximum achievable performance[5]. They fail to fulfill the three-fold objective of the XaaS providers. First, these providers are expected to maximize the number of accepted users, second, they should minimize total cost without compromising with their own profit or customer's satisfaction and third, whatever algorithm they employ to schedule task to Virtual Machines (VMs), they must avoid the wasteful usage of energy to fulfill the above said requirements[6].

In the literature an efficient dynamic scheduling algorithm has been proposed which is based on Transportation Model of Linear Programming Problem Solving Techniques[7,8,9]. The reliability of the system is shown to increase while minimizing the total turnaround time. However the algorithm has several assumptions which still make it unsuitable for real time cloud applications. One such assumption is that the resources available in the cluster of the cloud have the same capacity. Also there is no concern for efficient usage of power or minimizing energy consumption. Also, the analysis of given methods reveal that the method requires high computation overhead[10].

In this paper we propose a model of Cloud Scheduler based on Multi-index Transportation Problem (MTP), which is an extension of Transportation Problem (TP) itself. In literature, several researchers have suggested the applicability of MTP in scheduling resources where objective functions are conflicting in nature[11,12]. However, the contribution towards the use of

MTP, to model power-aware Cloud Scheduler[13], has been scarce in the present scenario[14].

Background

The Transportation Problem, (also called Hitchcock Problem and denoted by TP) is one of the classic problems in operation research and a special type of linear programming problem. The objective is to determine the optimal shipment from a given set of origins 'm' to a given set of destinations 'n' in such a way as to minimize the total transportation cost 'z'. The problem is constrained by known upper limits on the supply at the various origins and by the necessity to satisfy the known demand at each destination. The TP model has been used in various areas for scheduling the shipment of physical commodities to the users. One limitation of the classical transportation model is that it assumes per unit cost for each origin destination pair is known a priori. Mathematically a TP is given as-

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

Subject to constraints-

$$\sum_{j=1}^{m} x_{ij} = a_j, i = 1...n$$

$$\sum_{i=1}^{n} x_{ij} = b_i, j = 1...m$$

$$x_{ij} >= 0, \forall i, j$$

The traditional model doesn't take in consideration the type of carrier, the various commodities to be transported, the different characteristics of the carrier and other factors which also can influence the total cost of transportation. Also, it cannot be assumed that carriers will be able to serve every origin-destination pair for which they are the least-cost carrier because of capacity constraints on the various carriers. Consequently, it is impossible to assign, a priori, the appropriate per unit transportation costs necessary to use classic transportation problem in modeling of real time transportation.

There are major differences between classical transportation problems and cloud resource scheduler given as under. i. Tasks once assigned to be executed on a particular node may later be switched on to some other node in the cloud. ii. Sometimes instances of the same task may be executing on several nodes in a cloud. iii. The exact cost of assigning tasks to nodes may not be known apriori.

All these arguments show that there is a lack of the standard transportation model for cloud resource scheduling and allocation or cloud resource management as mentioned above. In this paper we propose more indices to build realistic and dynamic cloud resource management model.

The proposed system model

The architecture of the proposed resource allocation framework is depicted in Figure 1. We call it a Multi-index Transportation Problem Cloud Resource Scheduler (MTPCRS) as it is modeled by extending the classical transportation problem.

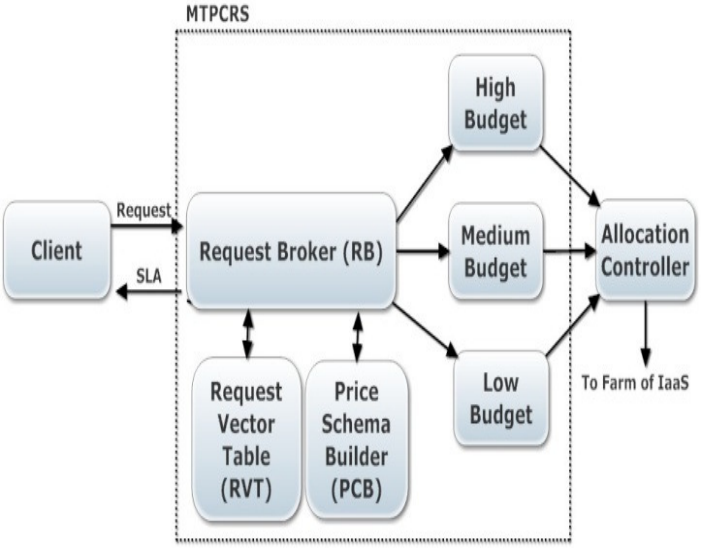

Figure-1
The System Model

The function and significance of each component are described below

The Request Broker (RB): The RB acts as an interface between the user and the system. User generated requests are admitted by the RB and the resource requirements are stored in a special table called as Request Vector Table (RVT). With the help of individual service vectors the total budget 'B' for servicing all the submitted requests is calculated. The RVT is an important information reository and becomes the blueprint for establishing service level agreement between the user and the resource provider. The resource requirements are characterized by the following parameters: i. Duration of time for which resource is required (T_p). ii. Power consumption costs incurred by the physical server to service the request (C_p). iii. Required processor capacity (Z). iv. Application execution costs (T_z). v. Number of Cores (S). vi. Service deadline (S_d)

Price Schema Builder (PSB): With help of the request vector the PSB computes the budget (B_i) required to execute the i^{th} request. The budget (B_i) is calculated using the equation

$$B_i = \sum_{S} \left(T_p\left(C_p + Z * T_z\right)\right)$$

When the budgets of the individual requests are calculated, they are classified as High, Medium and Low budget requests.

This is done in order to ensure that the high budget requests are given the highest priority while the low budget requests are given the lowest priority. Clearly the overall budget for servicing all admitted requests is given by.

$$B_{total} = \sum B_i$$

B_{total} gives a measure of stipulated cost incurred in servicing the admitted requests. The processing cost, F, is calculated by the proposed scheduling framework that prepares a resource allocation plan by formulating multi-index transportation table. The mathematical formulation for computing F is given in the next section.

Mathematical Formulation

It is assumed that the resource provider has m servers, n admitted requests and l virtual machines on each server. Each server has a specific processing capacity with lowest processing capacity servers forming the top layer of the cube as shown in figure 2. The successive layers of servers are arranged in increasing order of processing capacity. The minimum resource allocation plan is then obtained by constructing and solving Multi-index Cost Table (MCT). The formulation of MCT is described below.

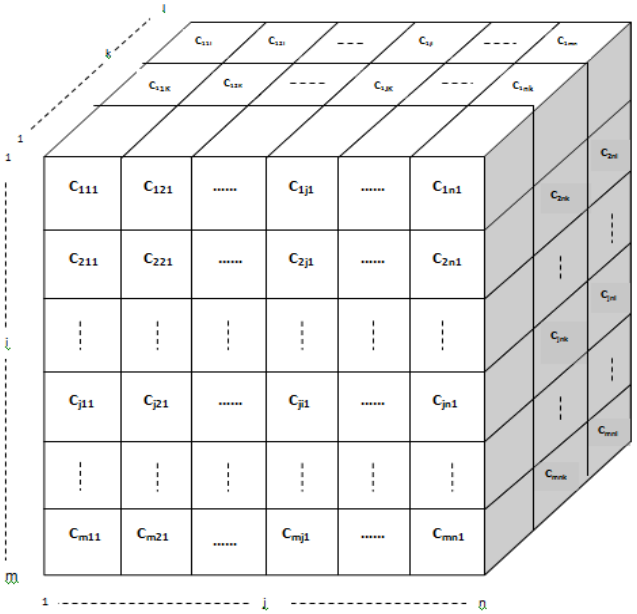

Figure-2
The cubical view of the different servers

i ε {1, 2, . . ., m}:- The variable denoting request (destination); j ε {1, 2, . . ., n}:- The variable denoting server (source); k ε {1, 2, . . ., l}:- The variable denoting virtual machine with a given processing capacity; a_j = Total CPU capacity of server j; b_i = Total CPU capacity requirement of request i; c_{ijk} = Per unit cost of executing i^{th} request when it is executed on k^{th} VM on j^{th} server; x_{ijk} = The amount of CPU capacity being utilized by i^{th} request when it is executed on k^{th} VM on j^{th} server; f_{ijk} = The fixed cost of processing i^{th} request executed on k^{th} server of j^{th} cluster; The fixed cost is subject to the class which the request belongs to (HB/LB/MB); w_{ijk}:- The cost incurred for instantiating a VM of type k, when i^{th} request is to be allocated on j^{th} server and to be processed by VM k, but VM k is not active.

Thus we arrive at following objective function:

$$\text{Minimize } F = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{l} (c_{ijk}\, x_{ijk} + f_{ijk} + w_{ijk}) \qquad (1)$$

Subject to-

$$\sum_{j=1}^{n}\sum_{k=1}^{l} xijk = ai, i = 1,2,...m$$

$$\sum_{i=1}^{m}\sum_{k=1}^{l} xijk = bj, j = 1,2,...n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} xijk = ck, k = 1,2,...l$$

The total profit earned by the XaaS provider:
Profit = *Btotal-F*
The aim of MTPCRS is to maximize the total profit minimizing F.

Algorithm: MTP Cloud Resource Scheduler

Step 1. Construct MCT for m requests, n servers, l processing capacity VMs, in such a way that the variable kε1 to l is arranged in from low to high processing capacity.
Step 2. Mark the cells MCT[ijk] as '-' where the VM has not been instantiated as yet.
Step 3. Allocate as much requests as possible to the current plane. i.e. to the VMs of current lowest capacity, in the following manner. Find MCT[ijk] for which c[ijk] is minimum. (If a tie occurs, choose the column with highest available capacity; this saves power by using underutilized servers). Allocate as much requests as possible to this cell. Case (i): If it is possible to assign each request with a corresponding server go to step 4. Case (ii): Move to the next k-plane. Mark all previous allocations and furnish the remaining requests in the same manner as in step 3.
Step 4. Calculate total processing cost and total profit.

Numerical Example: Now we illustrate the proposed method by the following numerical example. Consider the following MCT in which three requests I1, I2, I3 are to be mapped to four servers J1, J2, J3, J4. Each server has two types of VMs K1 and K2 and K1< K2. The unit cost c_{ijk} is marked in each cell. The instantiation cost w_{ijk} is also marked for those servers which are not instantiated as yet and a new VM needs to be provisioned. After applying Steps 2 and 3 from the proposed algo, we get the following table:

Table-1
Resource Allocation for request at different Servers

	J1	J2	J3	J4	
I1	10 12	2 15	20	11	15
I2	12 5	7	1 20	20 15	25
I3	4 10	14	16 10	18	10
	5	15	15	15	

Resource requirements by requests

Resource availability at clusters
(Capacity= K 1)

After following the steps 2 and 3, we see that the requirement at I2 is not fully satisfied and the availability at J3 is not fully utilized. Therefore we move onto the next plane K2.

Table-2
Resource Allocation for request at different Servers

	J1	J2	J3	J4	
I1	20 12	2 15	40	22	15
I2	12 5	14	2 5	20 15	25
I3	8 10	28	16 10	36	10
	5	15	15	15	

Resource requirements by requests

Resource availability at clusters
(Capacity = K2)

Now all requirements and availability are balanced, we calculate total processing cost: Total fixed cost (Assumed) = 100
Total processing cost= 2*15 + 12*5 + 2*5+ 20*15 +16 *10 + 100 + 20 = 680

The Business Process Diagram of the Model
In order to describe the workflow and process communication the business process diagram (BPD) of the resource allocation model is developed and depicted in figure-3. The figure indicates that simple a workflow is a major strength of the proposed model.

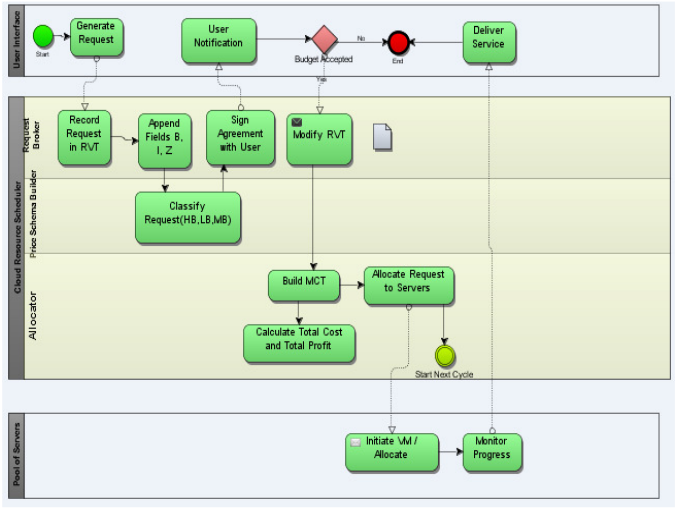

Figure-3
Business Process Diagram

There are three main pools in the BPD viz., the user interface pool, the cloud resource scheduler pool and the server pool. The division of pools into lanes and processes are as follows:

User Interface Pool: This pool has a single lane that contains the following processes: i. The Generate Request process: This process is responsible for monitoring and handling request generation from the cloud users. It gathers resource requirements of the submitted requests and converts it into a format that can be stored in the Request Vector Table (RVT) of the scheduler. ii. The User Notification Process: The function of this process is to act as an interface between the user and the cloud resource scheduler. The process ensures that the service requirements are met as per service level agreement and user gives the approval for final delivery of the service. iii. The Deliver Service Process: This process monitors the final delivery of the services to the cloud user by channelizing all communications between the cloud server and the cloud user. iv. The Start Event represents the starting point where the user enters the system. v. The End Event represents the ending point where either the user's request has been fulfilled or the user left the system due to non-acceptance of the budget proposed by the scheduler.

Cloud Resource Scheduler Pool: This pool comprises of three lanes that represent the main components of the cloud resource scheduler. The process description is given as under: i. The Request Broker Lane: This lane implements the functions performed by the request broker. ii. The Record Request process: This process records request requirements into Request Vector Table and also gives this information to the processes in the next lane. 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. iii. The Append Fields (B, I, Z): The budget B for servicing a request, any previous links to the request and required

processing capacity Z, are added to the request vector table by this process so that final agreement can be prepared. iv. The Sign Agreement Process: This process enables the user to confirm the final agreement by providing an interface via User Notification process. v. The Modify RVT Process: The SLA is agreed after a number of negotiations. This process makes the final modifications in the request vector table so that request classification can be done.

This Price Schema Builder Lane: This lane implements the functions of Price Schema Builder. i . The Classify Request Process: The admitted requests are classified as high, low and medium budget requests so that they can be serviced in a prioritized manner.

The Allocator Lane: This lane implements processes for request allocation. i. The Build MCT Process: In order to map requests to servers initial cost table is built by this process. ii. Calculate Total Cost and Total Profit Process: Using the initial cost table and applying resource allocation algorithm the total processing cost and total profit are calculated by this process. iii. Allocate Requests to Servers Process: This process monitors the final mapping of the requests to the servers according to the resource allocation schedule.

The Pool of Servers: A pool with one lane that represents the farm of cloud servers. i. Initiate VM / Allocate Process: This process monitors execution of requests on a virtual machine of the scheduled server. It manages instantiation and deployment of request on that virtual machine. ii. The Monitor Progress Process: This process supervises the processed results and monitors the delivery of the results to the user by means of Deliver Service Process in the User Interface Lane.

Sequence Diagram: The sequence diagram shown in figure 4 is used to develop the resource scheduling algorithm

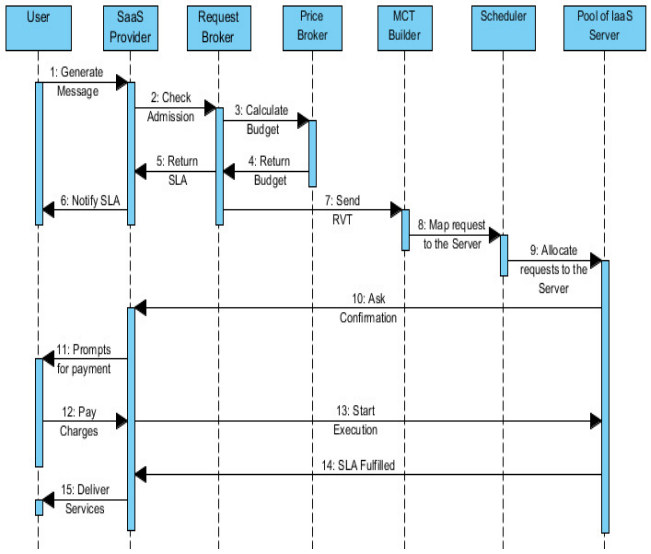

Figure- 4
Sequence Diagram

Event 1: User generates request to the resource provider. Event 2: The resource provider forwards the request to Request Broker to check admissibility of the request. Event 3: The Request Broker asks the Price Builder to calculate the budget. Event 4: The Price Builder returns the calculated budget to the Request Broker. Event 5: The Request Broker prepares the SLA and forwards it to the resource provider for the user's approval. Event 6: The resource provider notifies this SLA to the user. Event 7: The Request Broker sends the Request Vector table to the MCT builder. Event 8: The MCT Builder initializes values of the MCT and maps requests to the available servers and forwards the allocation plan to the Scheduler. Event 9: The Scheduler passes on the allocation schedule to the physical Pool of IaaS servers. Event 10: The IaaS servers send message to resource provider to ask for confirmation before executing requests. Event 11: The resource provider prompts for payment of charges from the user. Event 12: The user pays service charges to the resource provider. Event 13: Starts the execution of requests on the servers. Event 14: The IaaS servers notify the resource provider that the requests are mapped as per SLA. Event 15: Required services are delivered to the user.

Conclusion

A number of extensions to classical transportation problem have been proposed in the literature. However they have not been utilized for resource allocation purpose in cloud environment. In this paper the authors presented a multi-index transportation model based cloud resource scheduling algorithm that minimizes the total cost of processing all the requests on the physical servers running multiple virtual machines. The algorithm assigns requests to VMs on different servers by optimizing power requirements and performance both. The analytical results show that this model can be implemented using simple computational steps. The future work comprises the investigation and implementation of integrated resource allocation and resource consolidation mechanisms in the proposed framework.

References

1. Anton B., Jemal A. and Buyya R., Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing, *Journal of Future Generation Computer Systems,* 755–768**, (2012)**

2. Anton B. and Buyya R., 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, **(2012)**

3. Johan T., Montero Rubén S., Moreno-Vozmediano Rafael, Llorente Ignacio M., Cloud Brokering Mechanisms For Optimized Placement Of Virtual Machines Across

Multiple Providers, *Future Generation Computer Systems,* **28,** 358–367, **(2012)**

4. Mahmoodi K.R., Nejad S.S. and Ershadi M., 'Expert Systems and Artificial Intelligence Capabilities Empower Strategic Decisions: A Case study, *Res. J. Recent Sci.,* **3(1),** 116-121, **(2014)**

5. Liu B., Uncertain Programing, Wiley, New York, (**1999**)

6. Buyya R., Garg S.K. and Calheiros R.N., 'SLA Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture and Solutions, Proceedings of the International Conference on Cloud and Service Computing, IEEE, Australia, 1-10, **(2011)**

7. Yang L. and Feng Y., A Bicriteria Solid Transportation Problem With Fixed Charge Under Stochastic Environment, *Applied Mathematical Modelling,* **31,** 2668-2683, **(2007)**

8. Senthil K. and Balasubramanie P., 'Dynamic Scheduling for Cloud Reliability using Transportation Problem, *Journal of Computer Science',* **8(10),** 1615-1626, **(2012)**

9. Aneja Y.P. and Nair K.P.K., 'Bicriteria Transportation Problems, *Management Science,* **25,** 73-78, **(1979)**

10. Movahedi M.M, 'A Statistical Method for Designing and analyzing tolerances of Unidentified Distributions, *Res. J. Recent Sci.,* **2(11),** 55-64, **(2013)**

11. Bautista L. and Abran A., Design of A Performance Measurement Framework For Cloud Computing, *J. Software Eng. Appli.,* **5,** 69-75, **(2012)**

12. Marston S., Li Z., Subhajyoti B., Zhang J. and Ghalsasi A., Cloud Computing-The business perspective, *Decision Support Systems,* **51,** 176–189 **(2011)**

13. Nathuji R. and Schwan K., Virtual Power: Coordinated Power Management in Virtualized Enterprise Systems, *ACM SIGOPS Operating Systems Review,* **41(6),** 265–278, **(2007)**

14. Mahmoodi K.R., Nejad S.S. and Ershadi M., Expert Systems and Artificial Intelligence Capabilities Empower Strategic Decisions: A Case study, Res. *J. Recent Sci.,* **3(1),** 116-121, **(2014)**