# Game Theoretic Resource Allocation in Cloud Computing

Srinivasa K G, Sharath Kumar K, Shashank Kaushik U, Srinidhi S, Vignesh Shenvi, Kushagra Mishra
Department of CSE
M S Ramaiah Institute of Technology
Bangalore, India
{srinivasakg, kademarsharath, kaushik2061, amnidhi007, vighnesh.mr, kushagra.om}@gmail.com

***ABSTRACT:*** *Considering the proliferation in the number of cloud users on an everyday basis, the task of resource provisioning in order to support all these users becomes a challenging problem. When resource allocation is non-optimal, users may facehigh costs or performance issues. So, in order to maximize profit and resource utilization while satisfying all client requests, it is essential for Cloud Service Providers to come up with ways to allocate resources adaptively for diverse conditions. This is a constrained optimization problem. Each client that submits a request to the cloud has its own best interests in mind. But each of these clients competes with other clients in the quest to obtain required quantum of resources. Hence, every client is a participant in this competition. So, a preliminary analysis of the problem reveals that it can be modeled as a game between clients. A game theoretic modeling of this problem provides us an ability to find an optimal resource allocation by employing game theoretic concepts. Resource allocation problems are NP-Hard, involving VM allocation and migration within and possibly, among data centers. Owing to the dynamic nature and number of requests, static methods fail to surmount race conditions. Using a Min-Max Game approach, we propose an algorithm that can overcome the problems mentioned. We propose to employ a utility maximization approach to solve the resource provisioning and allocation problem. We implement a new factor into the game called the utility factor which considers the time and budget constraints of every user. Resources are provisioned for tasks having the highest utility for the corresponding resource.*

**Keywords:** Cloud Computing, Game Theory, Utility based, Fair Allocation, Cloud Architectures

**Received:** 10 November 2013, Revised 14 December 2013, Accepted 20 December 2013

## 1. Introduction

### 1.1 General Introduction

Cloud and cloud based services are gaining popularity at a very fast pace. Users these days want portability and accessibility at the same time. Cloud services help them to access data from any device at any time. For example, Evernote is a very popular application with more than 10 million downloads. Some of its features include taking notes, setting reminders etc. It is available for all devices, be it your PC, mobile or tablet. The data can be edited and accessed from any of the devices at any time.

Stats collected on public IT cloud service point out that the worldwide revenue exceeds $21.5 billion in 2010 and is expected to reach $72.9 billion in 2015, representing a compound annual growth rate (CAGR) of 27.6%. It is four times the projected growth for the worldwide IT market as a whole (6.7%) [1].

Cloud is not an independent phenomenon in the IT industry but acts as the core part of the bigger picture of the industry. The boom in cloud technology is complimented by the growth of portable devices, growing availability of wireless broadband and scores of big data tools available.

The soaring popularity of cloud computing in scalable computing and data store are due to its shared nature of computing resources that are distributed around the globe. Cloud-based infrastructures automate the management of similar resources as a single entity and provide services like computational and storage facilities to the end users. Cloud-based services integrate distributed resources around the world into a coherent computing platform.With this ever increasing demand and scale, Resource Allocation/Provisioning is a problem that holds immense rewards for efficient solutions and strategies.

### 1.2 Problem Statement

The problem involves modeling, mathematically, the interactions between requests and available resources and developing a strategy to provision available resources to tasks such that the Cloud Service Provider obtains a maximal profit while also optimally satisfying resource requests of the tasks. Therefore, it is an optimization problem. The goal is to find such an optimal allocation.

## 2. Related Work

Cloud computing is an on-demand service and owing to this data and computational resources can be acquired or freed on-demand. Consumers rely on cloud service providers for uninterrupted and scalable solutions to meet their needs. Service providers strive to maintain the quality of service and try to optimize it according to the users need. Game theoretic solutions are being explored by the cloud community [9, 10, 11, 12, 13, 14, 15,16].

Chonho, et al. [13] proposed an algorithm called Nudge that allows applications to adapt their locations and resource allocation to the environmental conditions in a cloud. In another work, Wei,G., et al. [14] considered a quality of service resource allocation problem. In this problem the consumers tried to solve a complicated parallel computing problem. The solution consisted of two steps; first to solve the problem independently and then using an evolutionary mechanism to take into account optimization and fairness. Rajkumar, et al. [15] also used a game theoretic method to schedule related computational cloud computing services. Fei, et al [16] used a Bayesian Nash Equilibrium Allocation algorithm to solve resource allocation problem in cloud computing. The experiments carried out in the work show that cloud users can receive Nash equilibrium allocation solutions by gambling stage by stage and the resource will converge to the optimal price.Self-organized Mobile Ad-hoc Networks (MANNES) are P2P networks put more emphasis on selfish behavior of entities [17, 18, 19, 20, 23, 24, 25, 26]. Each node is specified as a player is under the authority of a self interested user in such networks.

## 3. Proposed Algorithm

The working of the allocation method consists of mainly three stages. Users and brokers act as the requesters; Physical machines are the actual resources which are virtualized to hide the original properties. Hence, there are a certain number of virtual machines created in the datacenter. The second stage is the SLA resource allocator which does the accounting and pricing of the resources and finally dispatches the virtual machines.

**Input:** $M$, the set of all requests. $N$, the set of all resources.

**Output:** Allocation Matrix indicating the requests and resources provisioning as determined by utility maximization method.

### 3.1 Algorithm

#### 3.1.1 Utility Calculation

The utility value from which the matrix is formed is calculated using the formula:

$$U_{ij} = \frac{1}{(w_m * \gamma_j * (\lambda_i / \kappa_j)) + w_t * \left(\frac{\gamma_j}{\kappa_j}\right)} \tag{1}$$

where $w_m$ = Weightage for money.

$\gamma_j$ = Cost per seconds.

$\lambda$ = Length of each request.

$w_t$ = Weightage for time.

$\kappa_j$ = Capacity of each request.

**3.1.2 Calculation of Local Maximums**

Local maximum of utility values is calculated using the formula:

$$LocalMax = max_{ij}\,(U_i)\ \forall_i \in M \tag{2}$$

Where $U_i$ = Utility value for the task.

**3.1.3 Calculation of Global Maximums**

$$GlobalMax = \begin{cases} max_i\,(max_j\,(U_i)),\ U_j \in LocalMax \\ max_i\,(R_i),\ U_j \notin LocalMax \end{cases}\quad \forall_i \in M \text{ and } j \in N \tag{3}$$

Where $max_i$ indicates the maximum value over all tasks.

$max_j$ indicates the maximum value over all resources

$R_j$ is the revised utility value.

**3.2 Steps**

• Classify the requests into reservation queue and general game queue. This is done in the initial stage of service level agreement.

• Choose the game queue as the problem to be solved.

• Invoke the first round game and find utility values for every request - resource pair based on weightage to budget and deadline constraints.

• Calculate local and global maximum values and populate the 2-D array where the row numbers represent requests and column numbers represent resources.

• Allocate the first resource to the request with maximum utility (global max).

• If there are multiple requests with same utility factor for a resource, put them into wait - queue of the same resource.

• Continue the iteration and allocation to all resources.

• Print the allocation matrix and utility values for allocated VM-cloudlet pair.

• If there are any more requests, then start another round of game. Else, exit.

**3.3 Calculating the Utility Value**

$$U_{ij} = \frac{1}{(w_m * \gamma_j * (\lambda_i / \kappa_j)) + w_t * \left(\frac{\gamma_j}{\kappa_j}\right)}$$

A solution of the scheduling problem is a non-negative matrix m rows, one for each task, and m columns, one for each resource.

The entry $t_{ij}$ is task allocated to resource $t_j$. Let $t_i$ represent the ith row of matrix a. Then allocation vector $a_i$ satisfies $a_{ij} \in a_i$ and $a_{ij} = k_i$.

$w$ and $e$ denote the weights of completion time and expense, respectively. Therefore, the $U_i(a_i)$ denote the utility of task $S$. We consider equal weightage for expense and the time. So the weightage coefficient $w_t$ and $w_e$ will be equal. Since we are not dealing with the communication of subtasks in the current algorithm, the total time is the value calculated by the following formula.

$$Execution\ time = Total\ length\ /\ capacity\ in\ mips. \tag{4}$$

Where *mips* = Million instructions per second.

| **Algorithm Name** | **Utility Maximization** |
|---|---|
| Input: | Cloudlets: Request makers who desire to get the resources (Virtual Machines), their demand for number of execution or processing units coupled with weightage to bud get and deadline constraints. |
| Output: | Allocation matrix representing allocated VM-request pair and corresponding utility fac tor values. |
| 1. Classify (Req) → (General (Req), Reservation(Req)) | |
| 2. Choose Game (Queue) → problem to solve. Choose the set of requests to be considered in the game. | |
| 3. find Utility_value (Req) Resource pair based on weightage to budget and deadline constraints.Calculate utility value of each task by using the formula 1.1 Invoke_Game (req) The game will start by taking the utility values of all the tasks and form a utilization matrix for further calculation. | |
| 4. Calculate local and global maximum values: Maxarray ← Globalmax (Localmax ( (Utility_value)) Populate the 2-D array where the row numbers represent requests and column numbers represent resources. Local and Global maximums are calculated using the formula 1.2 and 1.3. | |
| 5. Allocate the first resource to the request with maximum utility (global max): Allocatted ← Req (Globalmax). | |
| 6. If there are multiple requests with same utility factor for a resource, put them into wait - queue of the same resource: Vmqueue ← Push (Req) | |
| 7. Continue the iteration and allocation to all resources. | |
| 8. Print the allocation matrix and utility values for allocated VM-cloudlet pair. | |
| 9. If there are any more requests, then start another round of game. Else, exit. | |

## 4. Motivation

In this section, we provide results of an experiment that corroborate our main hypothesis: existing resource allocation policies with non-optimal strategies can cause clients to incur unnecessarily higher costs as compared to when the resource allocation policy is optimal. This servers as our preliminary motivation to carry out the work and come up with a new novel approach to optimize the method. The tabulations made in the results section shows requests from clients of diverse characteristics and demands. Sub optimal allocations made by load shared model or the first come first serve model in contrast to optimal allocations made by our Utilization Maximization model are reflected in the same.

We demonstrate the effect of resource contention in a cloud simulation environment by performing various experiments on the Cloudism. We run computationally highly intensive and less intensive custom jobs using data generated by a front end random writer. We perform different types of experiments using different input data sizes and processing entity demands. This helps us understand the effect of contention in the network.

In this paper, we address the problem of resource allocation via utilization maximization model using Game Theory.

## 5. Cloud Resource Allocation

In this section, we define and present Cloud Resource Allocation Games (CRAGs). CRAGs model the resource allocation in a cloud and capture the provider-client interactions. We introduce the concept of Utilization factor and propose mechanisms which ensure that the cost incurred by the system at highest utility is close to the global optimum. In this paper, we only consider a static scenario in which a set of clients submit their jobs to a cloud as a batch. We leave modeling of the dynamics that can exist in practice to future work.

### 5.1 Modeling resource allocation in clouds

In our model, the clients of a cloud are modeled as the players in a CRAG. A client's strategy is represented by the client's resource allocation. We make the following assumptions about the various entities involved in a cloud:

• The cloud hosts different types of resource (e.g., CPU) and all clients are concerned with the usage of cheapest resource which satisfies its demands. This general case with multiple types of resources helps tabulate results obtained in real cloud environment.

• Each client acts selfishly and the cloud provider tries to optimize the total utility of the clients while ensuring efficient resource usage. Although cloud providers would be interested in maximizing their revenue, they would not want to charge their clients unnecessarily high prices in order to retain them. This follows from the fact that cloud computing is a competitive market with a choice of multiple providers for clients.

• Clients are charged based on per CPU time consumed. This implies that the amount of money a client has to pay the cloud is directly proportional to the amount of time the requested amount of resources is used by the client. This assumes a fixed number of processing entities requested by a client at a time.

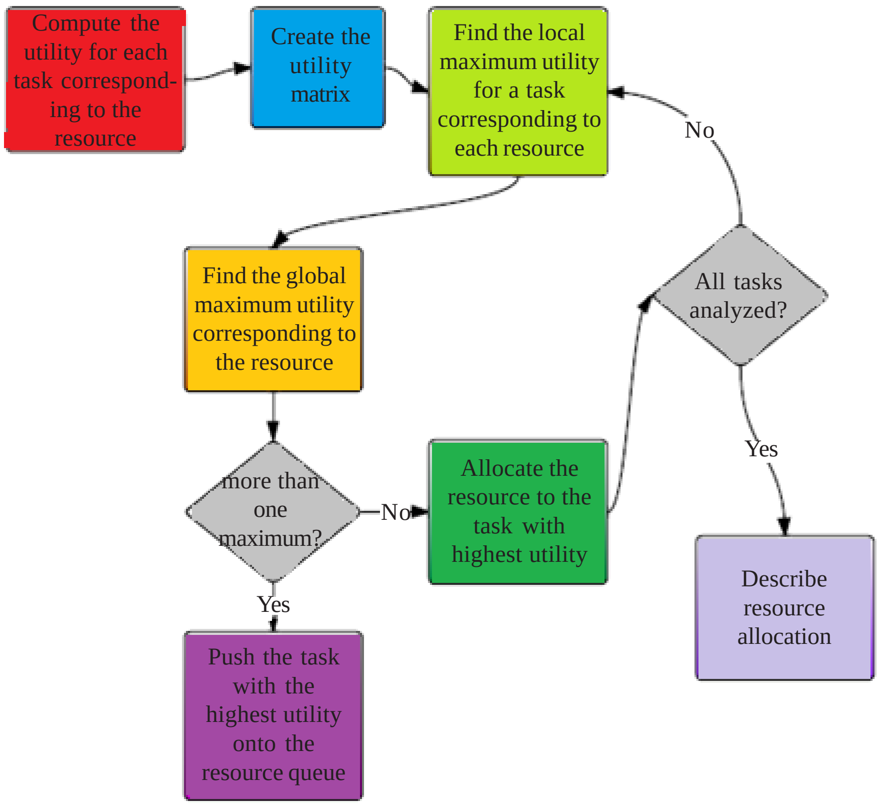
• The cloud may or may not have sufficient amount of resources to accommodate the resources requested by all the clients. So, a queue exists to contain a limited number of requests which could potentially maximize the profit with a high utilization factor computed based on its characteristics and constraints coupled with the cloud resources and their characteristics.

### 5.2 CRAGs in Practice

In the earlier subsection we have shown how the resourceallocation in clouds can be modeled using CRAGs. In this section, we show how these mechanisms can be adopted in practice. The resource allocation based on the utility maximization takes place in rounds. At the start of each round, all the clients submit the jobs that have to be run in that round, to the cloud. The actual set of clients can vary across rounds. The provider first calculates its resource allocation vector by deriving a solution to min max problem. This results in allocation of resources in best-fit manner to cloudlets in the cloudlet pool. This satisfies all the conditions outlined earlier and maximizes the profit of the resource provider while also making allocation to handle service requests fairly, in spite of selfish strategies employed by selfish players.The description of the system design is shown in figure 1.

## 6. System Design

Here, we provision resources for requests (used interchangeably with "*task*") based on the utility value calculated by employ ing the proposed algorithm and then finding the suitable task for which to provision the available resources. The process starts with the creation of virtual machines. All the virtual machines are created in one or more datacenters. The status of these virtual machines is tracked continuously. After the virtual machines are created, the broker decides which requests to schedule first. The broker performs the task of choosing the request with highest expected utility from the entire set of waiting requests for which to allocate the requested resources.



Compute the utility for each task corresponding to the resource
Create the utility matrix
Find the local maximum utility for a task corresponding to each resource
No
Find the global maximum utility corresponding to the resource
All tasks analyzed?
more than one maximum?
No
Allocate the resource to the task with highest utility
Yes
Describe resource allocation
Yes
Push the task with the highest utility onto the resource queue

After this initial phase, the actual process of the algorithm starts. It starts with the computation of utility value for each pair of task and resource. Thus, we construct a utility matrix with rows representing the tasks and columns representing the resource (i.e VM). We then consider the problem of expected utility maximization as a min-max problem. In this case, since we want to implement a fair allocation policy and want to strike a balance between fair allocations and maximize revenue generated for Cloud Service Providers, we must first determine the local maximum utility value for each task. We must then identify the global maximum utility value for each resource from among these local maxima.

To accomplish this, we use an evolutionary mechanism that changes the strategies chosen in the initial optimal allocation as determined by the local maxima. It does so by modifying the initial solution according to the global maxima. This evolutionary optimization technique ensures that each resource is scheduled to execute the task for which it has the highest utility which in turn means that the task will be executed faster and that it has bid for the resource at a higher price, thereby, guaranteeing maximized returns for the cloud service provider.

It finds the maximum utility in each column. After the *LocalMax* calculation, we compute global max of utility once again, where we check the maximum of maximums. That is for each column's maximum utility we again check for the maximum utility in each row. If we find only one max for one resource then allocate the virtual machine with that id to that cloudlet with the corresponding id. Repeat the procedure until all the tasks are analyzed. If there are multiple maximum utility values, then push the line up the tasks with equivalent utility values to the resource queue/waitlist. Finally, the resources are fairly allocated to all the tasks according to their utility value. Figure 2 shows the sequence of operations between the three major entities Consumer, Broker and the Datacenter.

The requests obtained by the users handled by the broker and the availability of corresponding virtual machines are checked by

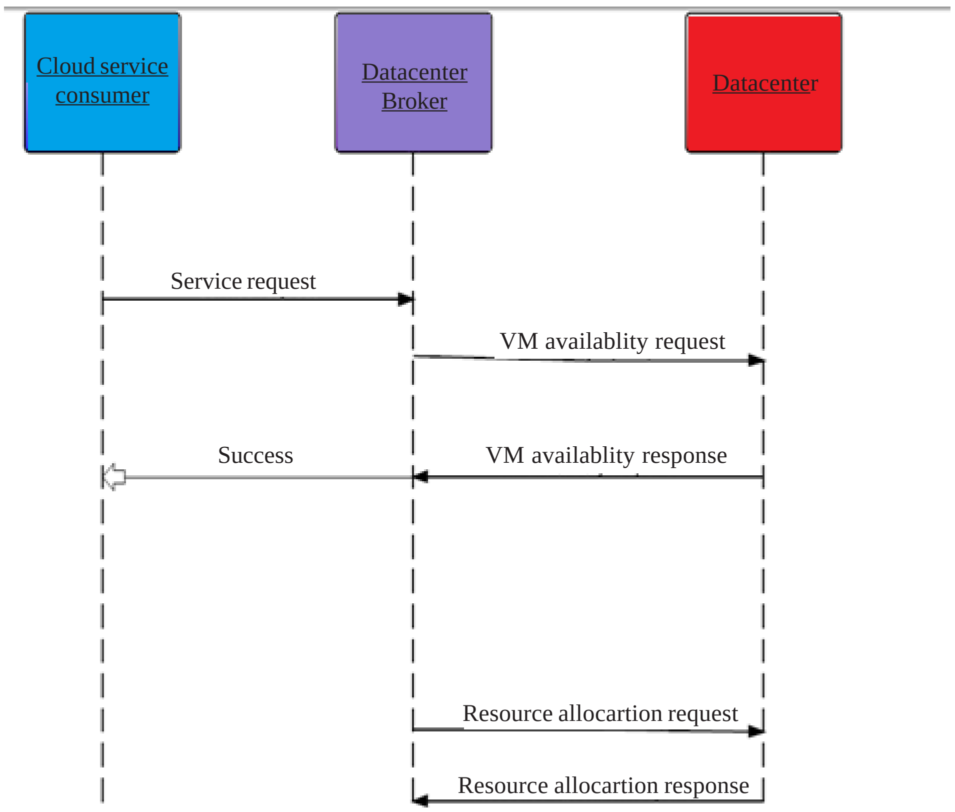


Figure 2. Sequence Diagram

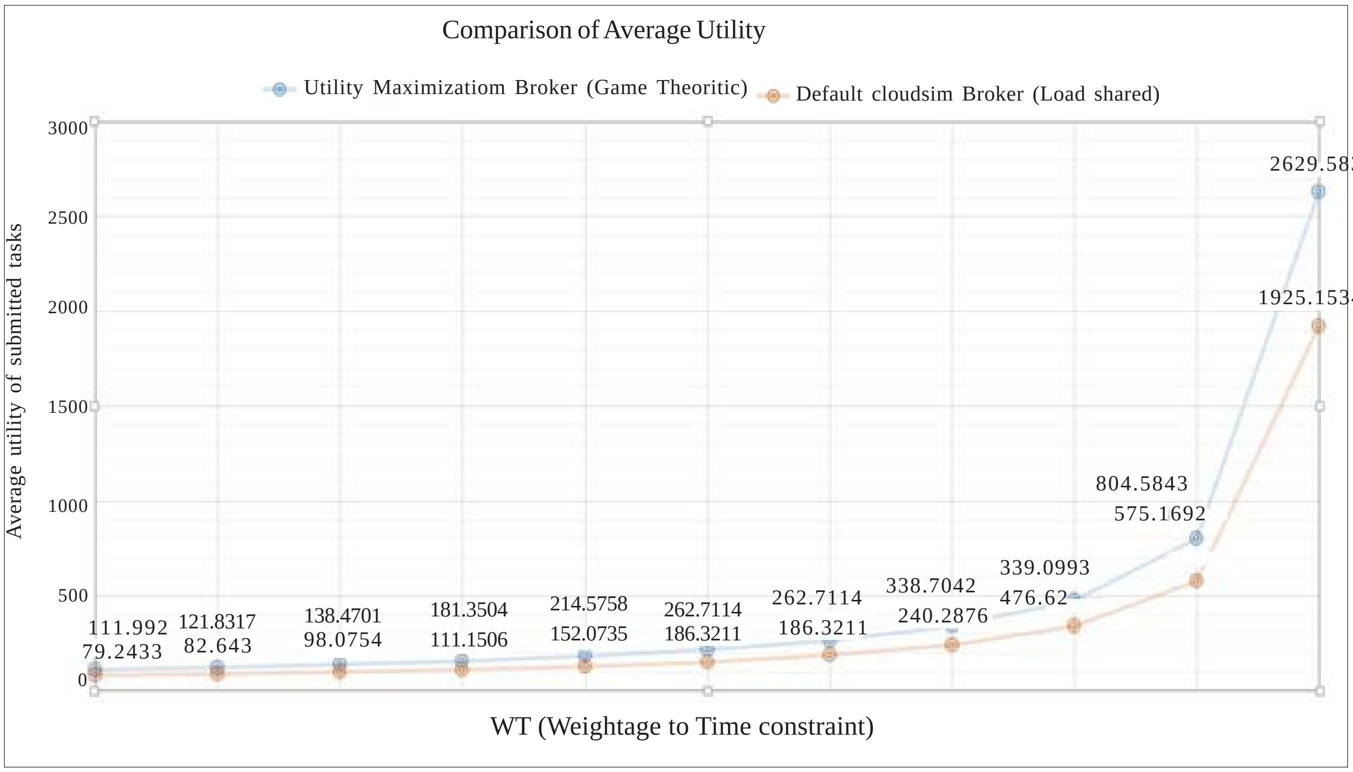


Figure 3. Graphical Representation of Obtained Improvement

| Wt/wm | Average Utility value | | Improvement Factor |
|---|---|---|---|
| | Utility (Game Theoretic Approach) | Utility (Default Cloudsim Approach) | |
| 0.0/1.0 | 111.992 | 79.2493 | 1.413161 |
| 0.1/0.9 | 123.8317 | 87.643 | 1.41291 |
| 0.2/0.8 | 138.4709 | 98.0254 | 1.412602 |
| 0.3/0.7 | 157.0358 | 111.1986 | 1.41221 |
| 0.4/0.6 | 181.3504 | 128.4625 | 1.411699 |
| 0.5/0.5 | 214.5758 | 152.0735 | 1.411001 |
| 0.6/0.4 | 262.7114 | 186.3211 | 1.409993 |
| 0.7/0.3 | 338.7042 | 240.4876 | 1.408406 |
| 0.8/0.2 | 476.62 | 339.0993 | 1.405547 |
| 0.9/0.1 | 804.5843 | 575.1692 | 1.398865 |
| 1.0/0.0 | 2629.5804 | 1925.1514 | 1.365908 |

Table 1. Graphical Representation of Obtained Improvement

sending a request to datacenter. Upon availability the datacenter sends a response to datacenter broker. Then the resource allocation request is made to the datacenter where the algorithm is employed and allocation is made. Finally a successful allocation message is displayed to the user and resource is provided to that consumer.

## 7. Implementation

This implementation provides:

• support for modeling and simulation of large scale Cloud computing data centers

• support for modeling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines

• support for modeling and simulation of energy-aware computational resources

• support for modeling and simulation of data center network topologies and message-passing applications

• support for modeling and simulation of federated clouds

• support for dynamic insertion of simulation elements, stop and resume of simulation

• support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines

## 8. Results Obtained

The figure 3 depicts the resulting improvement in the game theoretic approach to resource provisioning against the default load shared approach adopted in Cloudsim's DatacenterBroker package. We measure Average Utility value that is approximately 140% of the value seen in Cloudsim approach and in FCFS approach. We have seen an even higher improvement over the standard FCFS approach. Results are tabulated in table 1.

Our main objective of taking up this project was to maintain a fair and equitable environment towards resource provisioning and allocation. Through our approach, the cloud consumer's requirements are taken into account and it is in the best interest of the cloud consumer as well as the cloud service provider since profits/benefits are maximized to either party. All consumers share equal chances of obtaining the resource in demand as our approach involves allocation of resources to tasks taking the least time for execution. Observed improvement over existing approach is > 40%. Implementing this at the Cloud Service Provider's end will result in dramatically increased revenue to the CSP as well as efficient execution of requests in the event of large number of requests for a limited number of available resources given time and budget constraints. Also, all customers are assigned resources that are best suited for the specified constraints and computational nature of the task.

We compare the execution time of the Utility Maximization approach and the Load Shared approach for tasks of both varying and uniform load characteristics. We notice that the reduction in execution time, and hence, the improvement in efficiency, is non-uniform over the different tasks. Using the approach suggested in this paper, The execution time could be reduced to approx. 27% in the best case and approx. 50% in most cases. In some cases, there is no reduction observed. This happens routinely in tasks which are so resource hungry that they do not share resources. Therefore, we note that the worst case of the proposed approach is not worse than the default approach.

**8.1 Qualitative improvements**

Qualitative characteristics refer to qualities or properties of cloud computing, rather than specific technological requirements. One qualitative feature can be realized in multiple ways depending on different providers. The proposed approach forecasts

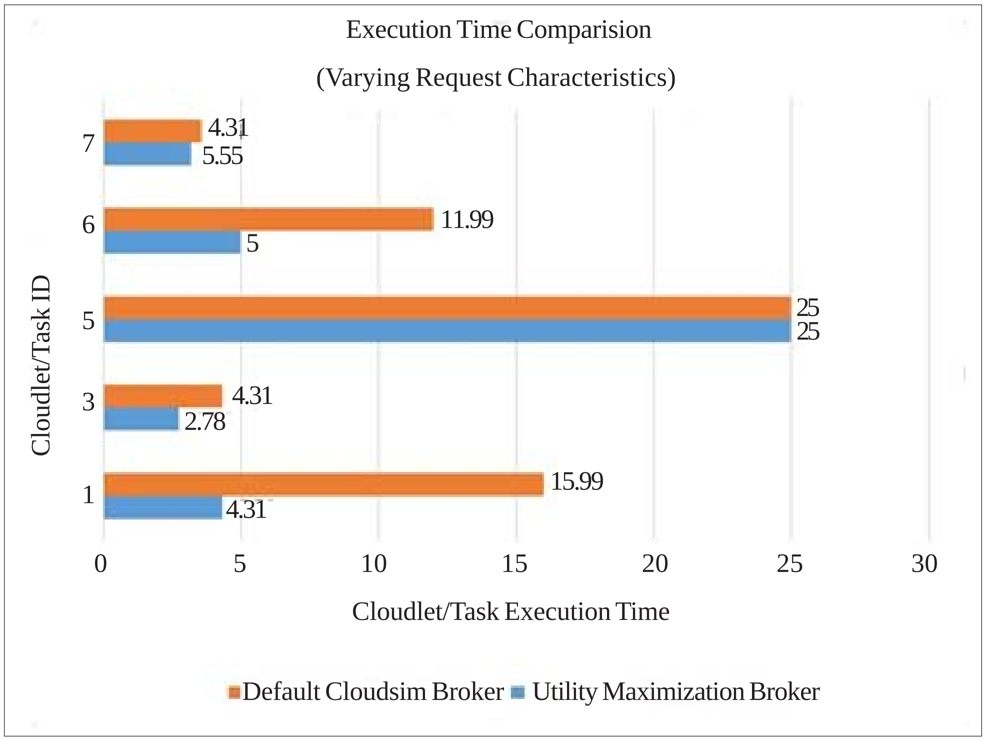


Figure 4. Results

improvement in terms of Elasticity.

• Elasticity means that the provision of services is elastic and adaptable, which allows the users to request the service near real-time without engineering for peak loads. The services are measured in fine-grain, so that the amount of offering can nearly perfectly match the consumer's usage. As it has been illustrated in the resulting graphs, such elasticity will assist in fair allocation of resources for heterogeneous and homogenous requests alike.

• Availability refers to a relevant capability that satisfies specific requirements of the outsourced services by a cloud broker. QoS metrics like response time and throughput must be guaranteed, so as to meet advanced quality guarantees of cloud users. Since execution time on all available virtual machines is reduced to optimal condition using our strategy, processing entities become available sooner for upcoming requests and can handle more number of requests than in a non-optimal resource allocation environment.

## 8.2 Economic Improvements

• **Energy Efficiency:** As it has been illustrated in the two graphs, processing times are reduced by approximately 27% and about 34% correspondingly for homogeneous and heterogeneous requests.

Considering dynamic nature of the cloud services and demands, heterogeneous requests out number homogeneous requests. This best illustrates the best-fit processing time reduction improvement and thereby energy-efficiency resulting from our Utilization-Maximization strategy in actual commercial cloud environment.

• **Reduced Operational Expenditure:** Pricing on a utility maximization strategy basis is fine-grained with usage based options, so cloud customers need not worry about paying for what has not been their share of operational service. Operational Expenditure is greatly reduced.

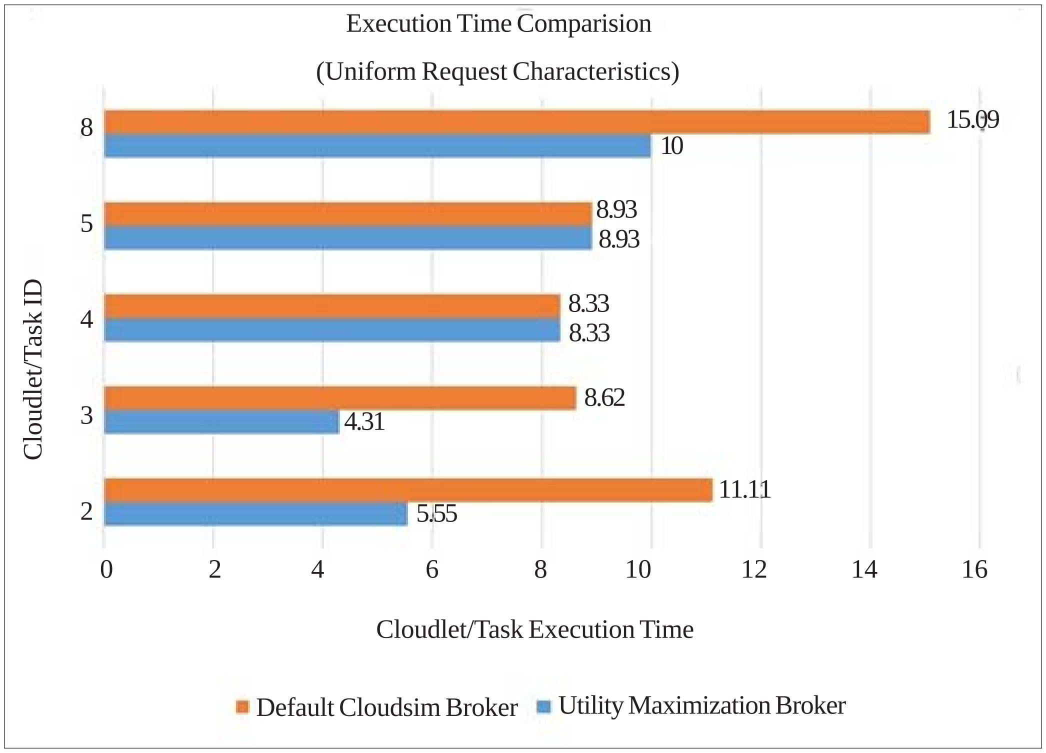


Figure 5. Results

## 9. Conclusion

The Utilization Maximation [UM] model is proposed in this paper and the model focuses on the resource allocation problem in cloud computing. Firstly, we made use of Cloudsim platform to simulate different entities involved in cloud resource allocation environment and the interactions and procedures between involved entities. Then, the algorithms of clients and resource brokers are proposed. Specifically, in order to find and minimize the unfairness due to the participation of selfish players, we introduce a utilization calculation method in the resource broker's algorithm. The user broker creates an allocation matrix taking into account the utility factors of each request-resource pair. Then, based on the allocation matrix local and global maximas, the broker allocates available resources to requests with highest utility factor. Simulation results show that, the UM algorithm allocation is better in maximizing overall utility factor against the load shared model and the FCFS model [First come first serve]. It can be compared to the best-fit strategy algorithms employed at kernel level in operating systems. This alleviates the overall execution time of available virtual machines (resources) and thereby QoS is partly assured. Selfish players do not get undue advantage since the allocations are dependant on utility factor. What's more, we analyzed the performance based on the simulation results and the computational load of the UM model is superior to other models. It is worthwhile to note that the UM model is much more applicable than some basic allocation models. However, the network delay, fraud user, reliability of re-sources problems are not considered in this paper. Thus, how to make the model realistic, fulfill the complete QoS requirements of users and improve the resource scheduling algorithms form the next step of our work.

## References

[1] http://www.idc.com/prodserv/idc_cloud.jsp

[2] Qiufen Xia, Weifeng Sun, Zichuan Xu, Mingchu Li. (2010). A Novel Grid Resource Scheduling Model Based on Extended Second Price Sealed Auction, 3rd International Symposium on Parallel Architectures, Algorithms and Programming, Dalian, p. 305-310.

[3] Lei Yao, Guanzhong Dai, Huixiang Zhang, Shuai Ren, Yun Niu. (2008). A novel algorithm for task scheduling in grid computing based on game theory, The 10th IEEE International Conference on High Performance Computing and Communications, p. 282-287.

[4] Maheswaran, R. T., Basar, T. (2003). Nash equilibrium and decentralized negotiation in auctioning divisible resources, Group Decision and Negotiation, p. 361-395.

[5] Kwok, Y. K., Song, S. S., Hwang, K. (2005). Selfish grid computing: Game-theoretic modeling and NAS performance results, *In*: Proceedings of CCGrid, p. 349- 356.

[6] Bredin, J., Kotz, D., Rus, D., Maheswaran, R. T., Imer, C., Basar T. (2003). Computational markets to regulate mobile-agent systems, Autonomous Agents and Multi-Agent Systems, p. 235-263, 2003.

[7] LI Zhijie, Cheng Chuntian, Huang Xuefei, Li Xin. (2006). A sequential game-based resource allocation strategy in grid environment, *Journal of Software*, p. 2373-2383. (in Chinese with English abstract).

[8] Abramson, D., Buyya, R., Giddy, J. (2002). A computational economy for grid computing and its implementation in the nimrod-G resource broker, Future Generation Computer Systems, p. 1061?1074.

[9] Amazon Elastic Compute Cloud (EC2). http://aws.amazon.com/ec2.

[10] Microsoft Azure Services Platform. http://www.microsoft.com/azure/default.mspx.

[11] Rackspace Mosso. http://www.mosso.com/.

[12] Ristenpart, T., Tromer, E., Shacham, H., Savage, S. (2006). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds, *In*: Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, p. 199-212.

[13] Chonho Lee, Junichi Suzuki, Athanasios Vasilakos, Yuji Yamamoto, Katsuya Oba. (2010). An evolutionary game theoretic approach to adaptive and stable application deployment in clouds, *In*: Proceeding of the 2nd workshop on bio-inspired algorithms for distributed systems, ACM, New York, p. 29-38.

[14] Guiyi Wei, Athanasios Vasilakos, Yao Zheng, Naixue Xiong. (2010). A game-theoretic method of fair resource allocation for cloud computing services, *The Journal of Supercomputing*, 54, p. 252-269.

[15] Rajkumar Buyya, Manzur Murshed. (2002). GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing, Concurrency and Computation: Practice and Experience, p. 1175-1220.

[16] Fei Teng, Frédéric Magoulès. (2010). A New Game Theoretical Resource Allocation Algorithm for Cloud Computing, Advances in Grid and Pervasive Computing, p. 321-330.

[17] Korilis,Y. A., Varvarigou. T. A., Ahuja, S. R. (1998). Incentive-compatible pricing strategies in non-cooperative networks, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, 2, p. 439-446, 29 Mar-2 Apr.

[18] Marbach, P. (2001). Pricing differentiated services networks: bursty traffic, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, 2, p.650-658.

[19] Ranganathan, K., Ripeanu, M., Sarin, A., Foster. I. (2004). Incentive mechanisms for large collaborative resource sharing, luster Computing and the Grid, p. 1- 8, April.

[20] Volper, D. E., Oh, J. C., Jung, M. (2004). GameMosix: Gametheoretic middleware for CPU sharing in untrusted P2P eEnvironment, *Parallel and Distributed Computing and Systems* (SPDCS).

[21] Micah Adler, Rakesh Kumar, Keith Ross, Dan Rubenstein, David Turner, David D. Yao. (2004). Optimal peer selection in a free-market peer-resource economy, *In*: Proceedings of the Second Workshop on Economics of Peer-to-Peer Systems.

[22] Luzi Anderegg, Stephan Eidenbenz. (2003). Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents, *In*: Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03), New York, p. 245-259.

[23] Milgrom, Paul. (1989). Auctions and Bidding: A Primer, *Journal of Economic Perspectives*, American Economic Association, p. 3-22.

[24] Buyya, R., Abramson, D., Giddy, J. (2001). A case for economy grid architecture for service-oriented grid computing, *In*: Proceedings of the 10th IEEE Int'l Heterogeneous Computing Workshop, IEEE Computer Society, Washington, p. 776-790.

[25] Guiyi Wei, Vasilakos. A.V., Naixue Xiong. (2000). Scheduling parallel cloud computing services: an evolutional game", 1st International Conference on Information Science and Engineering, p. 376-379, December.

[26] Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998). Biological sequence analysis: Probabilistic models of proteins and nucleic acids, Cambridge, in Cambridge University Press.

[27] http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/ html_dev/main.html