ISSN:0975-9646



# Trustworthy Resource Sharing on Collaborative Cloud Computing

**T. Amith Kumar**
*Student, SCSE,*
*VIT University, Vellore, India*

**G. Priya**
*Asst. Professor (Senior), SCSE,*
*VIT University, Vellore, India*

**Dr.N.Jaisankar**
*Professor , SCSE,*
*VIT University, Vellore, India*

**Abstract-In the current generation, developments in cloud computing leads to a tremendous scope for collaborative cloud computing (CCC). In CCC, the resources are universally scattered and distributed across the globe which belong to different organizations and the resources are used to provide services to the clients. Because of the self-governing highlights of elements in CCC, the issues of reputation and resource management must be mutually communicated to guarantee the fruitful development of CCC. These issues are jointly addressed in the past but when we address these two issues jointly, it creates twofold overhead. Hence, resource and reputation management strategies are not well designed and they are not powerful. If the client selects the highest reputation node, then the other reputation nodes are neglected and there is no full utilization of resources and it doesn't meet client Qos demands. In order to overcome this, we propose a technique called Harmony. Harmony involves three stages: comprehensive resource/reputation management, selection of multi Qos-oriented resource and price-assisted resource and reputation management.**

**Key words: cloud computing, resource management, reputation management and harmony**

## INTRODUCTION

Cloud computing has become an on-demand one by which we can provide services to the clients over the internet. There are so many number of cloud service providers such as Amazon, Google, and IBM etc. These service providers charge according to the usage of storage, bandwidth and various other parameters. The service provider cannot provide the services using only one cloud and it is not possible when the clients are increasing. It also cannot provide resources to an application fully in some situations during peak time [1]. In order to provide services, the researchers need to connect multiple clouds having a virtual lab environment in order to provide super-computing capabilities to the clients in order to fully utilize the resources. Due to this features and developments in cloud computing, the demand for collaborative cloud computing (CCC) has grown.

Due to CCC, we can provide services to people where the resources belonging to different organizations are largely pooled. CCC interconnects various physical resources to empower sharing of resources in the clouds in order to provide virtual perspective of resources to its clients. This perspective is useful when a client requests resources and cloud does not have sufficient resources. It has to discover and use the resources in different clouds [2] [3].

**Importance of Resource management and reputation management:** CCC works in an extensive environment involving thousands or millions of resources which are distributed universally. It may be noticed that many clients will be using and leaving the system due to which resources utilization and availability are constantly changing [2] [4]. Due to this, resource management becomes a productive one. Because of the unique qualities in CCC, we can allocate different Qos parameters to distinctive nodes. A node may give low Qos due to machine breakdown or it is not ready to provide high Qos to spare expenses. These shortcomings are uncovered in Amazon, Google [5] and other service providers. Security has been considered as a major factor in all these service providers and also in grids [6]. Hence, resource management needs reputation management for quantifying the resource provision Qos for guiding resource selection [2] [5].

In order to guarantee the development of CCC, the issues of resource management and reputation management must be communicated jointly. It can be achieved in three errands.

1. Productively finding trustworthy resources.
2. Choosing resources from the found alternatives.
3. Completely using the resources in the system while to keep away from overloading any node.

**Existing Method:** In order to have large scale CCC, three steps must be executed in order since other incorporative methods are not suitable. Many techniques have been proposed for resource and reputation management but these two issues have been discussed separately. When we combine both resource management and reputation management in CCC, it is creating high overhead [4]. Also, the methods which were proposed in the past for resource management and reputation management are not effective and does not support for large and dynamic environment for CCC. Past resource management assign only one parameter to the node for providing resources [7], [8], [9], [10]. But we assume that node is Comprehensiveand should be separated from different resources. A node may perform well for storage services but it may not perform well for processing computations. And also previous reputation management techniques are not sufficient to give right direction in order to ensure selection of trustworthy individual resource. Likewise, reputation management needs to depend on resource management for reputation separation over numerous resources. Previous resource management techniques assume only single Qos parameter such as either security or efficiency.

When a client is given to choose from a list of service providers, it may be possible that he can choose highest service provider who provides security and neglecting other

service providers. This leads to low success rate and highest node may be overloaded with many resource requests. This ungraceful arrangement of reputation management and resource management will display conflicting behaviors and it may also influence the adequacy of both. We may face two challenges after seeing the results of single Qos parameter and its conflicting results. First and foremost is how we can jointly address the multiple Qos parameters such as efficiency, speed, price etc. And the second is how a node would be able to control its reputation and resources without being overloaded and to get profits?

**Proposed Method:** By understanding the differences between resource and reputation management, we present Harmony, a CCC platform with integrated resource and reputation management. With the help of this, we can accomplish improved and joint administration of resources and reputation management over distributed resources in CCC. Harmony empowers a node to find its wanted resources and further more discover the reputation of found resources so that client can pick resource provider by resource accessibility as well as by provider's reputation of giving the resource. Harmony can also manage the difficulties of extensive scale and dynamism in complex environment of CCC.

Harmony involves four steps. They are as follows:

1. Comprehensive resource/reputation management.
2. Multi-Qos-resource oriented selection.
3. Price-assisted resource and reputation control management.
4. Combination of these three components.

**Architecture:** The architecture is as shown as follows

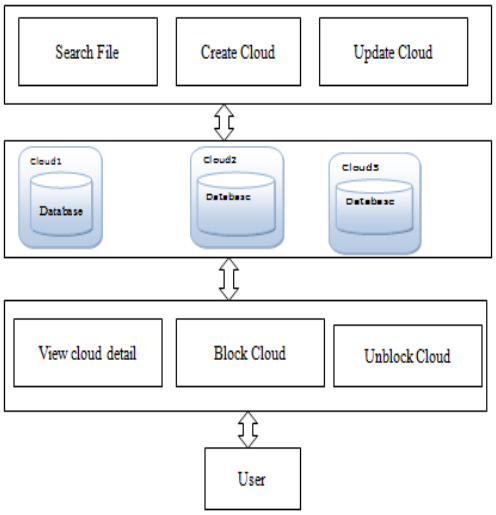


Fig 1: Architecture of proposed system

Here the user can create cloud and also he can update it. Each cloud has its own database. He can also search the file in the cloud. The user can also block and unblock the cloud on his wish. He can also transfer files between the files from the source to destination.

## HARMONY DESIGN

### Comprehensive resource/reputation management

In this, harmony uses the cycloid structure in which all the nodes are connected to one Qos parameter [12]. It can have a maximum of $n=d*2^d$ nodes where d is dimension. Each cyclic node id consists of two indices: cyclic index and cubic index. Cyclic index is the value obtained by which consistency hash value [13] modulated by d. Cubic index is obtained by dividing hash value by d. All nodes are combined into clusters. Each cluster can have a maximum of nodes from 0 to (d-1).

Harmony uses the Hilbert number [14] which indicates the geographical location of the node. We can also know the distance between the nodes by using Hilbert number. We know that every node has its own resources such as bandwidth, CPU etc. which are globally defined. It also includes IP address, memory space used by the user, location etc. We can store resource information in some directory nodes and forward those to the corresponding directory nodes when they are needed for the user [13], [15]. Similarly, in reputation management we need to store the reputation information of nodes and forward them to corresponding directory nodes [7], [8], [9], [10]. In this, resource information and reputation information should be distinguished from resource type and Qos parameters. By storing the information in the corresponding node, it becomes easier for the user for selecting resource providers. Harmony uses a cluster which stores all the information of each resource type [8]. It can also store the client feedback if it is given by him. Within each cluster, harmony also groups the physically close nodes available to the client into one, so that he can find close available resources. With the help of this, it is easier for client to find rather than searching in the whole world. It reduces cost and increases the improving resource efficiency. In the above design , each node sends its available resources and request to directory node when they are needed. The directory node stores all the information and it acts as a medium between clients and providers. In this, we have three steps: store, request and process.

In the store algorithm, we will insert the available resources provided by the node by using insert operation. Insert has two parameters: Hilbert index and resource information. Hilbert index has two parameters: first parameter is the Hilbert resource information which is used for differentiating the nodes in the cluster and second parameter is the consistency value which is used for differentiating clusters. In this, the resource id with same consistency value is stored in the same cluster and if it differs with Hilbert resource information, then it is stored in the same node in cluster.

In the request algorithm, node sends a request with its consistency value and its Hilbert number and also with its price, efficiency etc. for a resource type. Directory node stores the information of the requested node, which is close the resource requestor. In the process algorithm, the parameters which match with the node request, it gets that node in the cluster. After upon selection, it uses multi-Qos oriented resource selection algorithm. If it does not match with requests, then the directory node will send failure

response to the node. We will explain how a directory node will analyze a node in its cluster. It uses a two-way randomized method with the help of which we can achieve speed and more improvement. In two-way randomized probing method, it will generate two random ids for a node id by increasing the range of proximity of ids and gets those ids in its cluster. If the requested resource is not found, then it increases the proximity range. By increasing the range, it also helps to map resource requestors and providers to improve the efficiency. After finding the requested resource, it will use the three algorithms which are discussed above.

**Multi-Qos-Oriented resource selection**

After a root node finds the resource providers that has the reputation value, price, etc. which are matched with client requestor, the client has to choose resource providers. In order to select a resource provider, it depends on various factors such as distance, security, speed etc. Previous methods have concentrated only one single Qos-parameter such as either they may have concentrated on speed in order to have very good speed to the client or they may decide distance. They may provide security thinking security is the important one. But in this a client can select two or more parameters.

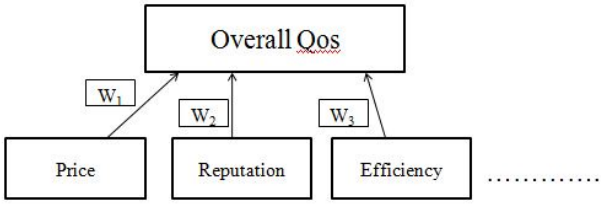


Fig 2: Neural network model for Qos attributes

The above figure shown is a neural network model for selecting resource. In order to solve the problem, harmony unifies this as a single attribute by considering all attribute values. It asks clients to give rating to each Qos parameter and overall Qos resource service. These are collected by the directory node and are stored in the server. It is not possible to determine the power of each Qos parameter on overall Qos. In order to do so, harmony uses neural network and also determines the priority of attributes. The inputs and ratings are taken as $\{Y_1, Y_2, Y_3\ldots\}$. The output is the overall Qos of a provider. Here the weight of influence of each attribute of Qos $\{W_1, W_2, W_3\ldots\}$. These weights determine the influence on overall Qos instead on single Qos. Here the activation function is Qos $X= \sum W_i*Y_i$ . It also keeps weights $W_i$ up-to date by taking the clients ratings into account. The directory node also takes the influence of each Qos parameter instead of taking the normal influence of overall Qos. To do this, it temporarily changes the weights in the neural network model. The directory node determines the node with overall high Qos value and it also takes clients priority Qos parameters in choosing the resources.

**Price-Assisted resource/reputation management**

After determining the nodes which provide high Qos attributes, harmony uses resource trading model for providing resources to the client with high Qos [12], [13], [14], [15]. In this model, the resource provider tells the price of the resources and they are charged according to the one unit per resource. Resources with higher demand, higher reputation and low load have income higher than other nodes. Hence it is better to provide higher demand to every node in order to avoid overloading of the nodes.

Previous methods have always choosen highest node as a server. When many clients always choose highest node, it will have higher income but it will overloaded. The lowest reputation node will have lower income but is not overloaded. The highest node is effective when it has unlimited resources, but it is not true in CCC in which each node has limited resources. Here in this, we will make sure that no node is overloaded by fine-tuning adjustment of price parameter. By doing this, all nodes can have higher income and reputation without being overloaded. We define a load factor f=l/c where l= amount of resource provided by a node to others and c=total amount of resources of a node. When a node is providing resources, it should ensure f<1. If f>1, it means that a node is overloaded. When it is overloaded, it decreases the price and gives chance to another nodes [10].

This technique allows the prevention of uncooperative behaviour, fully utilization of resources and it also ensures that all nodes are having higher reputation value without being overloaded.

## RESULTS AND ANALYSIS

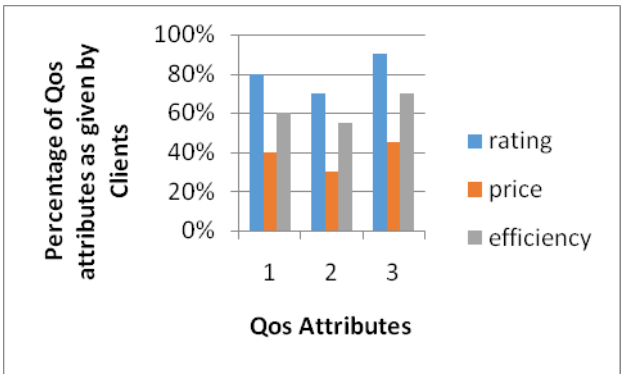


Fig 3: Percentage of Qos attributes

Fig 3 shows the percentage of Qos attributes as given by the clients after using the proposed technique.

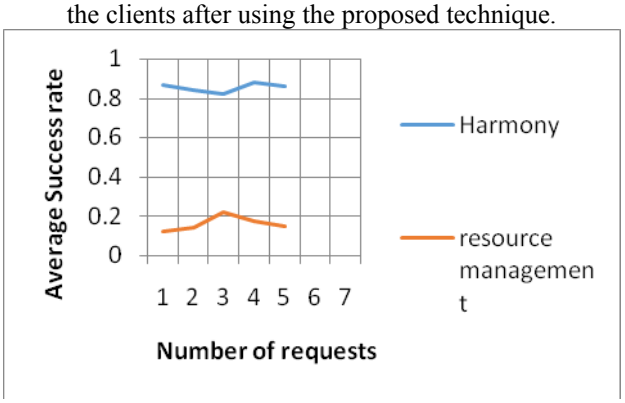


Fig 4: Average success rate

Fig 4 shows the average success rate using the harmony technique than the resource management technique

## CONCLUSION

In this paper, we propose a integrated technique for CCC in cloud computing called harmony. It has three components which is used for mutual interactions between clouds for resource sharing. In Comprehensive resource/reputation management, we collect and provide information about available resources and reputations of providers for providing the types of resources. In multi-Qos-resource oriented selection, client select resource provider that offers overall high Qos. In price-assisted resource/reputation control, it helps all the nodes to offer high Qos without being overloaded. It allows in full utilization of resources and having high income. These three components work together to enhance the efficiency and reliability of resource sharing in collaborative cloud computing.

## REFERENCES

[1] Haiying shen and Guoxin Liu "An Efficient and Trustworthy resource sharing platform for collaborative cloud computing", IEEE Transactions on Parallel and Distributed Systems, Vol 25, No. 4, April 2014

[2] P. Suresh Kumar, P. Sateesh Kumar, and S. Ramachandram, "Recent Trust Models In Grid," J. Theoretical and Applied Information Technology, vol. 26, pp. 64-68, 2011.

[3] J. Li, B. Li, Z. Du, and L. Meng, "CloudVO: Building a Secure Virtual Organization for Multiple Clouds Collaboration," Proc. 11th ACIS Int'l Conf. Software Eng. Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010.

[4] C. Liu, B.T. Loo, and Y. Mao, "Declarative Automated Cloud Resource Orchestration," Proc. Second ACM Symp. Cloud Computing (SOCC '11), 2011.

[5] C. Liu, Y. Mao, J.E. Van der Merwe, and M.F. Fernandez, "Cloud Resource Orchestration: A Data Centric Approach," Proc. Conf. Innovative Data Systems Research (CIDR), 2011.

[6] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Management," Proc. IEEE Int'l Conf. Dependable, Autonomic and Secure Computing (DASC), 2009.

[7] IBM Red Boo. Fundamentals of Grid Computing, Technical Report REDP-3613-00 2000.

[8] L. Xiong and L. Liu, "Peertrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," IEEE Trans. Knowledge and Data Eng., vol. 16, no. 7, pp. 843-857, July 2004.

[9] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks," Proc. World Wide Web Conf., 2005.

[10] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," IEEETrans. Parallel and Distributed Systems, vol. 18, no. 4, pp. 460-473, 2008.

[11] R. Zhou and K. Hwang, "Gossip-Based Reputation Management for Unstructured Peer-to-Peer Networks," IEEE Trans. Knowledge and Data Eng., 2007.

[12] H. Shen and G. Liu, "Harmony: Integrated Resource and Reputation Management for Large-Scale Distributed Systems," Proc. 20th Int'l Conf. Computer Comm. and Networks (ICCCN), 2011.

[13] H. Shen, Y. Zhu, and W. Li, "Efficient and Locality-Aware Resource Management in Wide-Area Distributed Systems," Proc. Int'l Conf. Networking, Architecture, and Storage, 2008.

[14] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," Proc.ACM SIGCOMM, 2003.

[15] M. Cai, M. Frank, and P. Szekely, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," Proc. Fourth Int'l Workshop Grid Computing, 2004.