International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 10, October 2015
Available at http://internationaljournalofresearch.org

A Novel Approach to Trustworthy Resource Sharing on Collaborative Cloud Computing

Meadisetti N Srinivas Rao[1]& Sk. Jahangeer Basha [2]
[1]PG Scholar,Dept of CSE, Rao & Naidu Engineering College, Ongole,Prakasam Dist, Andhra Pradesh
[2] Associate Professor,Dept of CSE, Rao & Naidu Engineering College, Ongole,Prakasam Dist, Andhra Pradesh

Abstract—
Cloud computing has significant efficiency and cost advantages, advancements in cloud computing provides a promising future for collaborative cloud computing (ccc). In ccc globally-scattered distributed cloud resources belonging to different organizations or individuals are collectively used in a cooperative manner to provide services. In this paper, we address resource management and reputation management are the two fundamental issues. In ccc harmony platform Cloud will used to integrates res mgt and rept mgt in a harmonious manner. Harmony incorporates three key innovations: integrated multi-faceted resource/reputation management, multi-qos-oriented resource selection, and price-assisted resource/reputation control. The issues of resource management and reputation management must be jointly addressed in order to ensure the successful deployment of ccc. however, these two issues have typically been addressed separately in previous research efforts, and simply combining the two systems generates double .

Keywords— Distributed systems; Reputation management; Resource management; Distributed hash tables; Cloud computing; Multi-qos resource selection; Price control

I. INTRODUCTION

Cloud computing has become a popular computing paradigm, in which cloud providers offer scalable resources over the Internet to customers. Currently, many clouds, such as Amazon's EC2, Google's AppEngine, IBM's Blue- Cloud, and Microsoft's Azure, provide various services (e.g., storage and computing). For example, Amazon [1] (cloud provider) provides Dropbox [2] (cloud customer) the simple storage service (S3) (cloud service). Cloud customers are charged by the actual usage of computing resources storage, and bandwidth.

The demand for scalable resources in some applications has been increasing very rapidly. For example, Dropbox currently has five million users, three times the number last year. A single cloud may not be able to provide sufficient resources for an application (especially during a peak time). Also, researchers may need to build a virtual lab environment connecting multiple clouds for petascale supercomputing capabilities or for fully utilizing idle resources. Indeed, most desktop systems are underutilized in most organizations; they are idle around 95 percent of the time [3]. Thus, advancements in cloud computing are inevitably leading to a promising future for collaborative cloud computing (CCC), where globally- scattered distributed cloud resources belonging to different organizations or individuals (i.e., entities) are collectively pooled and used in a cooperative manner to provide services.
As shown in Fig. 1 below, a CCC platform interconnects physical resources to enable resource sharing between clouds, and provides a virtual view of a tremendous amount of resources to customers. This virtual organization is transparent to cloud customers. When a cloud does not have sufficient resources demanded by its customers, it finds and uses the resources in other clouds.

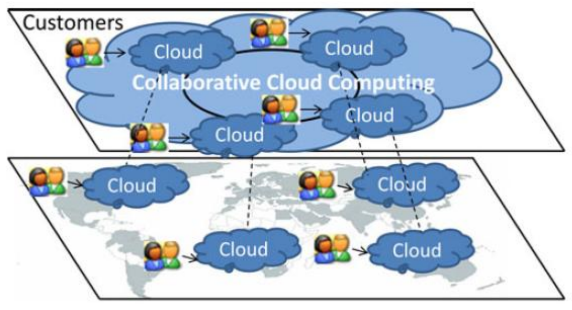


Fig:1 An example of collaborative cloud computing

Uses of Resource and Reputation Management:

A node may provide low QoS because of system problems (e.g., machines break down due to insufficient cooling) or because it is not willing to provide high QoS in order to save costs. Also, nodes may be attacked by viruses and Trojan horse programs. This weakness is revealed in all the cloud platforms built by Google, IBM, and Amazon [7], and security has been recognized as an important factor in grids (the predecessor of clouds) [8]. Thus, resMgt needs reputation management (repMgt) to measure resource provision QoS for guiding resource provider selection [4], [7]. As in eBay and Amazon, a repMgt system computes each node's reputation value based on evaluations from others about its will exhibit contradictory behaviors and significantly affect the effectiveness of both, finally leading to degraded overall performance. The results of the single-QoS-demand assumption and contradictory behaviors pose two challenges. First, in task (2), how can we jointly consider multiple\ QoS demands such as reputation, efficiency, and available resources in resource selection? Second, in task (3), how can we enable each node to actively control its reputation and resource supply so that it avoids being over overloaded while gaining high reputation and profit?

To ensure the successful deployment of CCC, the issues of resMgt and repMgt must be jointly addressed for both efficient and trustworthy resource sharing in three tasks:

1. Efficiently locating required trustworthy resources.

2. Choosing resources from the located options.

3. Fully utilizing the resources in the system while avoiding overloading any node.

II. METHODOLOGY

A cloud environment often contains large number of machines that are connected by a high-speed network. Users access sites hosted by the cloud environment through the public internet. A site is typically accessed through a URL that is translated to a network address through a global list check, such as domain name system. A demand is made on a site through the internet which either processes the request or forwards it. Clouds are shared environments where multiple cloud users utilize the same equipment. Cloud user requests the resource from service provider. Multiple cloud users can request number of cloud services concurrently. So there must be a preparation that all requirements are made available to the demanding user in powerful manner to satiate their need. Collaborative Cloud networks are shared in a best-effective manner, making it hard for both users and cloud operators to reason about how network resources are allocated.

A. Existing Methodology

Although many distributed resMgt and repMgt systems for grids have been proposed previously, and cloud resource orchestration (i.e., resource provision, configuration, utilization and decommission across a distributed set of physical resources in clouds) has been studied in recent years, these two issues have typically been addressed separately. Simply building and combining individual resMgt and repMgt systems in CCC will generate doubled, prohibitively high overhead. Moreover, most previous resMgt and repMgt approaches are not sufficiently efficient or effective in the large-scale and dynamic environment of CCC.

Previous repMgt systems neglect resource heterogeneity by assigning each node one reputation value for providing all of its resources. We claim that node reputation is multi-faceted and should be differentiated across multiple resources (e.g., CPU, bandwidth, and memory). For example, a person trusts a doctor for giving advice on medical issues but not on financial issues. Similarly, a node that performs well for computing services does not necessarily perform well for storage services. Thus, previous repMgt systems are not effective enough to provide correct guidance for trustworthy individual resource selection. So, RepMgt needs to rely on resMgt for reputation differentiation across multiple resources.

B. Proposed System Combined Multi faceted Res/Rep Management:

A CCC operates in a large-scale environment involving thousands or millions of resources across disparate geographically distributed areas, and it's also inherently dynamic as entities may enter or leave the system and resource utilization and availability are continuously changing. This environment makes efficient resource management (res Mgt) a non-trivial task. Further, due to the autonomous and individual characteristics of entities in CCC, different nodes provide different quality of service (QoS) in resource provision. A node may provide low QoS because of system problems or because it is not willing to provide high QoS in order to save costs. Relying on a distributed hash table overlay (DHT), Harmony offers multi-faceted reputation evaluation across multiple resources by indexing the resource information and the reputation of each type of resource to the same directory node. In this way, it enables nodes to simultaneously access the information and reputation of available individual resources.

Multi-QoS-Oriented Resource Selection: For a single QoS request of customers, Harmony allows a client to implement resource selection with joint consideration of various QoS requests, such as reputation, efficiency, distance, and price, with different priorities. The difficulty here is how to consider different or combined QoS attribute, and a customer's most wanted priority of the attributes in provider collection. Harmony solves this problem by joining all attribute values and a client's considered attribute priority into an overall QoS metric. Similarly, Harmony develops a list of QoS attributes. It involves nodes to give evaluations for each QoS element and overall QoS for a source package in addition to the repute for a server. As the reputation response, the QoS ratings are also collected at the directory node of the resource of the server. The overall QoS is actually a result of the joint power from the QoS elements. However, it is not easy to identify how the changed features impact the overall QoS. The various attributes related to QoS is shown in Fig 2. Harmony depends on a neural network to find out the stimulus load of each attribute on the overall QoS value, and further considers users' attribute respect priority. A neural network can be used to derive meaning from complex data, extract forms and detect trends.

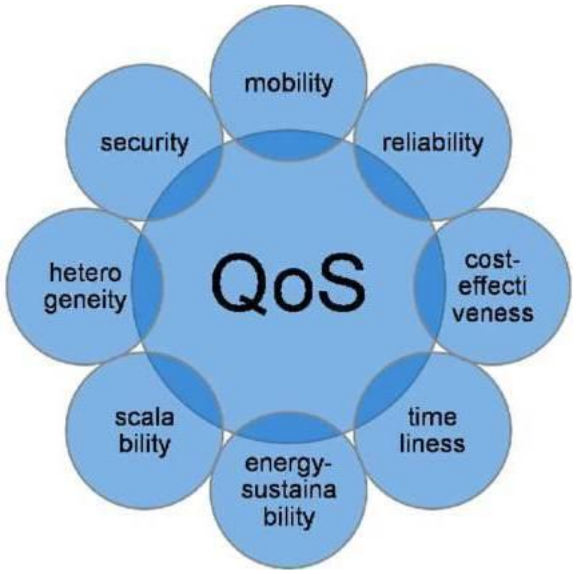


Fig.2. Quality of Services (QoS) attributes

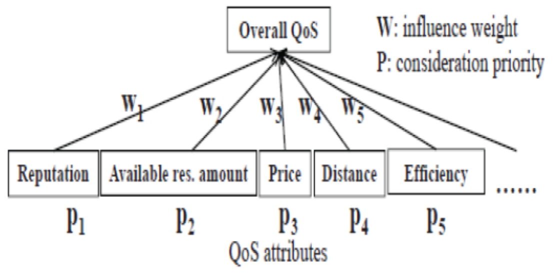


Fig.3. a neural network model for resource selection

In Harmony, each directory node builds a neural network model as shown in Fig.8. Its inputs are the ratings of each QoS attribute, denoted by {A1,A2, · · · ,A5, · · · }, and its output is the overall QoS value for a provider. The training process of the neural network is the process of determining the weight of influence of each attribute on the QoS, denoted by {w1, w2, · · · , w5, · · · }. These weights reflect the normal degree of influence from different QoS attributes on the overall QoS calculated by the collected QoS ratings from many nodes. After building the neural network, each directory node trains its neural network model periodically using its newly collected QoS ratings to keep the model and the calculated weights wi up-to-date. After the directory node locates satisfying resource providers, it calculates the overall QoS value for each server option by considering both the normal influence of QoS attributes on the overall QoS and the requester's consideration priority.as shown in Fig.3. It It then inputs each server's attribute values {A1,A2, · · · ,A5, · · · } into the neural network. The output of the neural network is the overall QoS. Finally, the directory node determines the server(s) with the highest overall QoS value. Thus, Harmony jointly takes into account the client's considered priority on different attributes and the influence weights of attributes on the final rating in server selection.

Price Assisted Res/Rep Management: In a supply matter, a store activist pays a resource provider (in the form of virtual credits) for its resource. The communications are directed in a circulated custom in Harmony. Harmony works an exchange perfect for supply communications in store allocation and controls the store price to regulator each node's reserve use and standing. It qualifies each node to adaptively regulate its supply price to make best use of its income and retain an in height character although escaping presence encumbered, in demand to totally and fairly utilize possessions in the system. A worker usually requires the price of its resources giving to the reputation value, load, and the interest of the resources. Resources with difficult reputations, lower loads, and higher demand (frequently requested) should have high prices. Therefore, in direction to earn more payment, each node is interested to afford high QoS to retain high reputation, while avoiding being overloaded. The price-assisted res/rep controller scheme checks sun helpful manners, heartens nodes to provide high QoS, and allows nodes to adaptively alter their load to offer high QoS. As a result, all properties in the system are fully and fairly utilized, nodes are not overloaded, and a node's reputation can truly reflect its QoS in offering resources without the influence of the overloaded status. To address the problem of low reputation for newly joined nodes, Harmony assigns the nodes a certain amount of starting virtual credits that can be used for building initial reputation. We define a load factor $f = l/c$, where l is the amount of resource a node has provided to others, and c is the total amount of that resource the node owns. When a node's $f > 1$, it is overloaded. A node periodically checks its f. If the node's $f > \alpha$ $(0.8 \leq \alpha < 1)$, it increases its price by one price unit to discourage requesters and avoid being overloaded. Otherwise, it decreases its price by one price unit in order to promote its resource usage to raise its own reputation and income. We choose $\alpha < 1$ rather than $\alpha = 1$ in order to avoid delayed responses to the node's overloaded status. This control method is used to deal with the overloaded and under loaded resource utilization situations. Cloud Broker or their brokers request a service anywhere around the world to the Cloud. An important notice makes the difference between Cloud consumers and users of the deployed cloud services. For example, a company deploying a Web application can be a consumer that represents different workload as per the different number of "users" using it.

C. Dynamic Priority Scheduling A dynamic scheduling algorithm is proposed with dynamic priority for the nodes based on which the virtual machines are scheduled. It schedules the virtual machines to the nodes controlled by upon their arrangement value, which varies dynamically based on their load vector. This dynamic priority approach leads to better operation of the property. Priority of a node is assigned turn to upon its space and the load factor. This algorithm hammers the true sense of balance among act and control effectiveness.

```
Algorithm scheduling priority {

Flag=0;

If (P =/0)

P1=max available resource node

If (load vector of P1<0.8)

Assign VM to P1;

If (P2 is set AND load vector of P2<0.8

AND

Swap P1 and P2;

Assign VM to P2;

Else if P2=P1

P1=current max available resource node

Assign VM to P1;

}
```

III. PERFORMANCE MEASURE

Finally we are evaluating the proposed approach with the existing approach for the Resource selection. Here we analyze and compare the performance offered by different configurations of the computing cluster. And present the evaluation comparison by the parameter metrics such as the viability, from the point of view of scalability, execution time, performance, and cost. Based on the comparison and the results from the experiment, we show that proposed approach works better than the other existing systems.

A. Performance Evaluation on Planet Lab

To validate the design of Harmony, we implemented a prototype on Planet Lab and conducted trace-driven experiments. Since we can stably access around 234 nodes in Planet Lab, we set the number of nodes in the system to 234. We created 12 resource types in our system. In order to derive node overall reputations and individual reputations for these resource types, we first identified 13 sellers from the Zol trace data with 3 merchandise types in common. We then mapped this trace data to the 234 nodes and 12 merchandise types. We normalized the reputation values from [0,100] to [0, 10]. The lowest overall reputation of these sellers is 8.9 in the trace data. To simulate an environment with high-, medium- and low-reputed nodes, we generated synthetic data for 1/3 nodes with overall and individual reputations randomly chosen from [1, 3] and [4, 6], respectively. Their individual reputations were set equal to their overall reputations. On Planet Lab, we selected 21 nodes (7 from the US, Europe and Asia, respectively) as landmark nodes for calculating node Hilbert numbers. We randomly chose 8 nodes as requesters in America, Europe, and Asia, respectively. In the experiments, unless otherwise specified, each requester sends one request every 10s for a resource randomly chosen from the 12 types.

B. Integrated Multi-Faceted Res/Rep Management

Resource management needs reputation management to provide a cooperative environment for resource sharing. Otherwise, a node cannot know which resources are trustworthy. Resource management in turn facilitates reputation management to evaluate multi-faceted node reputation in providing different resources. Therefore, resMgt may choose nodes unwilling to provide services and generate many service failures. Power Trust may choose overloaded nodes that cannot process requests successfully. By integrating both resource management and reputation management, Harmony enables joint consideration of node reputation and load status and chooses lightly loaded high-reputed nodes that can always process requests successfully. Below, we present experimental results to show the importance of integrating resource management and reputation management.

C. Trustworthy Resource Sharing

We first tested different methods when all requests are single resource requests. In order to see the effect of reputation management alone, we assumed that nodes do not drop requests when overloaded but queue the requests for processing later on. Fig.4 (a) shows the average success rate of each system, which is measured by the ratio of successfully resolved resource requests over total requests. A request is successfully resolved if the selected server has provided its requested resource. We see that Harmony achieves a success rate of over 96%, while Power Trust achieves around 73% and resMgt achieves around 44%. ResMgt selects a resource without considering reputation and may choose a resource provider with low reputation for the requested resource, leading to a low success rate. Power Trust always selects the highest overall-reputed provider. As verified by the trace, a node with a high overall reputation may provide low QoS for another resource due to either unwillingness or overloaded status. Therefore, Harmony significantly outperforms Power Trust by always selecting the supplier with the highest individual, rather than overall, reputation. Fig.4 (b) shows the average individual reputation of every group of 500 selected resource providers for the requested resources, which follows Harmony>Power Trust>resMgt. The experimental results confirm the effectiveness of multi-faceted reputation management and its importance in guiding trustworthy resource selection. We then tested different methods when all requests are multiple-resource requests. Fig.4(c) shows the average success rate of each method, with each request calling for three resources. A multi-resource request is successfully resolved only after all three resources are successfully discovered The result shows that Harmony>Power Trust>resMgt due to the same reasons as in Fig.4 (a) and Fig.4 (b). Comparing Fig.4(c) with Fig.4 (a), we observe that the average success rate of Harmony decreases about 0.06, while those of Power Trust and resMgt decrease around 0.34 and 0.35, respectively. This is because if one of the three resource suppliers has a low individual reputation, the final request failure is low. We also find that Harmony outperforms Power Trust and resMgt more significantly for multiple-resource requests because it can ensure the success rate of each of the three selected suppliers by considering multi-faceted reputations for different resources. We used the product of the individual reputations of the selected three resource providers of a request as the individual reputation result of this request. Fig. 3(d) shows the average individual reputation for each group of 500 multi-resource requests over time. The experimental result also follows Harmony>Power Trust>resMgt, which is consistent with the success rate result in Fig.4(c). Also, the result of each method in Fig.4 (d) is lower than that in Fig.3 (b). This is caused by the same reasons for the differences between Fig.4 (a) and Fig.4(c), which confirms the importance of considering individual reputation in selecting resource providers, especially for multi-resource requests.

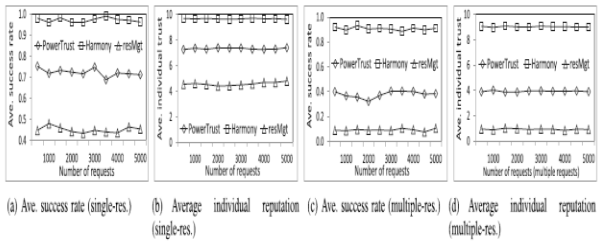


Fig.4. Trustworthy resource sharing.

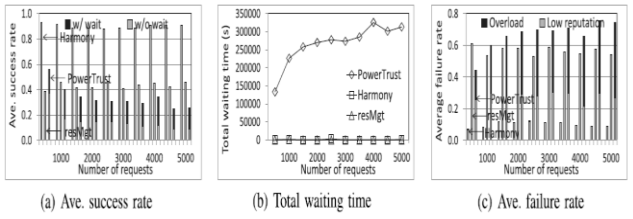


Fig.5. Efficient resource sharing.

D. Efficient Resource Sharing

Without resource management, reputation management cannot tell if a resource supplier has sufficient available resources. We then tested the importance of resource management to reputation management. In this experiment, each node maintains a waiting queue; an overloaded resource provider inserts its received request into its waiting queue. The requests staying in the queue for more than 100s are dropped. Therefore, a resource provision failure is caused by either a provider's overloaded status or its unwillingness to provide a service as reflected by its reputation. The Pareto distribution reflects the real world in which the amounts of available resources vary by different orders of magnitude. Thus, we used a Pareto distribution in determining a node's capacity for a resource type, a request's requested amount, and time period. We set the shape parameter to 2, and the scale parameters to 100, 40, and 200 for the above three parameters, respectively. We arbitrarily set the values in their reasonable ranges. Different setting values will not change the relative performance differences of the methods.

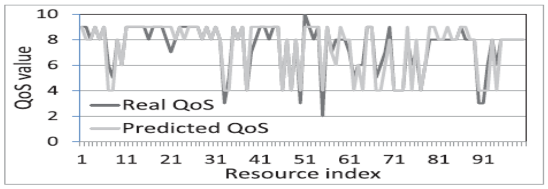


Fig.6. Predicted and real QoS values.

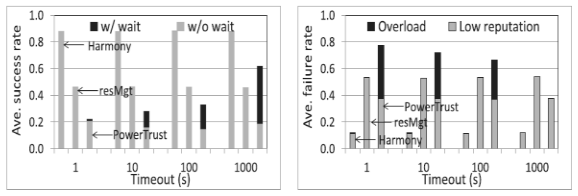
Fig.5 (a) shows the average success rates for groups of 500 requests by each method. The black color represents delayed successful requests that have waited in the queue before being processed, and the grey color represents successful requests with no delay. We see that Power Trust generates a large number of delayed successful requests, while the other methods generate no delayed requests. Fig.5 (b) shows the total waiting time for each group of 500 requests, including failed requests. We see that Power Trust generates high delay for a request, while Harmony and resMgt produce little or no delay, which is consistent with Fig.5 (a). This is because Power Trust always chooses the highest-overall-reputed nodes as resource providers without considering node load. These nodes receive too many requests, causing many to wait in the queues. Since both Harmony and resMgt select lightly loaded nodes as resource providers, they generate few delayed requests. Fig.5 (a) also shows that the success rates in most cases follow Harmony>resMgt>Power Trust. In order to further explore the trend, we show in Fig.5(c) the failure rate due to provider overload or unwillingness to serve for groups of 500 requests for each method. We see that Power Trust generates a higher failure rate due to overload than resMgt; this is because Power Trust fails to consider load in provider selection. However, resMgt has a higher failure rate due to provider unwillingness than Power Trust because resMgt only considers node load but neglects reputation. Only Harmony can constrain failures due to either cause, as it jointly considers both load and reputation.

E. Multi-QoS-oriented Resource Selection

We then evaluate the effectiveness of the multi-QoS-oriented resource selection scheme. We used 95×12 transaction records for 95 sellers, each with 12 types of merchandise. Each transaction record has 52 transactions. We used 40×12 for training and the remaining 55 × 12 for testing. We regard a node's individual reputation (rating on its transactions for a type of merchandise) as its overall QoS for the merchandise (i.e., resource) and regard its overall reputation as its reputation in Harmony. The inputs of the neural network model include the QoS attributes in each transaction (i.e., price, distance, service, quality and efficiency) and the seller's overall reputation. The output of the model is the seller's overall QoS. Because the real trace does not have users' consideration priorities, we assume that the six QoS attributes have equal priorities. Fig.6 shows the predicted overall QoS and the real overall QoS for 100 resource requests, both of which almost overlap. Their root mean square error equals 0.95, a very small value. The results show the effectiveness and accuracy of the neural network model in predicting the QoS in individual resource selection.

F. Effect of Queuing Timeout

In order to study the effect of the timeout for dropping requests on the success rate and failure rate, we measured the two metrics with varying timeout values for 5000 requests. Fig.7 (a) shows the success rates of different methods with different timeout values. We see that Harmony and resMgt generate no delayed requests since they choose lightly loaded nodes, while Power Trust generates many delayed successful requests since it does not consider node load. This result is consistent with that in Fig.5 (a). We observe that the timeout value does not affect the success rate of Harmony and resMgt. Also, as the timeout value increases, the success rate of Power Trust for delayed successful requests increases. As Harmony and resMgt always choose lightly loaded nodes, they do not need to store requests that cannot be processed in time into a queue. Power Trust heavily depends on the queue since it is very likely to choose an overloaded node. A smaller timeout introduces more request drops and hence fewer delayed successful requests, and vice versa. We also observe that the success rate follows Harmony>resMgt>Power Trust except when the timeout equals 1000s for the same reasons as in Fig.5 (a). The large timeout of 1000s allows Power Trust to store and resolve more requests, thus producing more successful requests than resMgt.



(a) Ave. success rate (b) Ave. failure rate

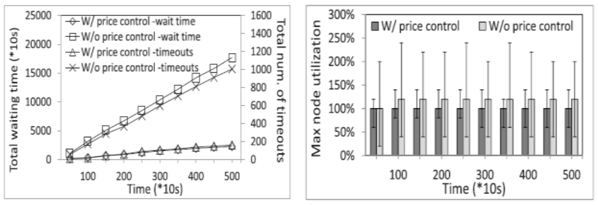
Fig.7. Effect of queuing timeout on resource sharing

Fig.7(b) shows the failure rate due to overload or unwillingness for the three methods at different timeouts. We see that the failures in Harmony and resMgt are all caused by provider unwillingness to provide services, and the failures in Power Trust are caused by both overload and unwillingness. As the timeout value increases, the failure rate caused by provider overloads decreases. A larger timeout enables more requests to be resolved and leads to few failures and vice versa. These results are consistent with those in Fig.7 (a). We also observe that the failure rate follows Harmony<resMgt<Power Trust except when the timeout equals 1000s, for the same reasons as in Fig.5(c). The large 1000s timeout allows Power Trust to store and resolve more requests, thus producing fewer failed requests than resMgt. In conclusion, Harmony and resMgt are not affected by the timeout, while Power Trust is sensitive to the timeout and a large timeout enables it to resolve more requests. Moreover, Harmony always achieves a higher success rate than the other methods since it considers both reputation and load status.

G. Price-assisted Resource/Reputation Control

We then test the performance on a system with and without the price-assisted resource/reputation control algorithm denoted by w/Price and w/price, respectively, in a heavy load situation. We randomly chose nodes from the system to generate requests. The request generating rate follows a Poisson process at a rate of 2 requests per second. We set every request to use 40 units of a resource for 200s. We chose these values so that some resource providers have insufficient available resources when requested. We set the price range to [10, 20] credits. All nodes have the same capacity of 200 and reputation of 9.

During the simulation period, no resource information is updated in the directory nodes. To choose a resource provider, a requester first identifies the providers with reputation > 8, then identifies the providers with the lowest price, and finally randomly chooses one. Nodes with reputation > 8 have probability 1 to offer the service. A resource request fails only when its provider is overloaded. If a requester receives a successful service, the resource provider's reputation is increased by 0.001. Otherwise, its reputation is decreased by 0.02. In every 10s, each node checks its load; if its load factor f > 0.8, it reduces its price by 1; otherwise, it increases its price by 1.



(a) Wait time and timeouts (b) Max. node utilization

Fig.8. Effectiveness of price-assisted control algorithm.

Also, each provider charges its clients for its offered resources and its reputation is updated in every 10s. The timeout of the waiting queue of each provider was set to 60s. We randomly determined the values of the above parameters in their reasonable ranges, and different parameter values should not change the relative performance differences
between different methods. Fig.8 (a) shows the total request waiting time in node queues and total number of request timeouts versus time. W/price generates a significantly higher total request waiting time than w/Price. Also, the request waiting time of w/price increases dramatically while that of w/Price grows only slightly as time goes on. This is because nodes in w/Price adjust their own load by price, which can effectively prevent node overload and avoid long queues.
In contrast, reputed nodes in w/price are likely to be overloaded since they have a high probability of being chosen as providers but have no strategy to adjust their load. Due to the same reason, the total number of request timeouts of w/price is much smaller than that of w/Price. Fig.8 (b) shows the median, 99th and 1st percentiles of all nodes' maximum utilizations every 500s. We see that w/Price generates smaller median values, much smaller 99th percentile values, and larger 1st percentile values than w/price. These results imply that w/Price distributes request load among servers more evenly than w/price. W/Price makes full use of all available resources while constraining node utilization within 100%; while in w/price, some nodes are overloaded or under loaded. The experimental results in both figures further verify the effectiveness of the price-based control algorithm in fully utilizing available resources and preventing overloads.

IV. CONCLUSION

Integrated resource/reputation management developed a new environment called Harmony which is a collaboration of the different clouds. Harmony is built using inter-dependencies between reputation and resources management which is used to retrieve information available on the cloud service providers; the resulted services are very much efficient and effective. The multi – QoS oriented allocate resources for it operation based on the highest level of QoS (Quality of Service) which is multiplied by QoS attributes. Price – assisted resources/ reputation control used to offer very high range of resources and also avoid over loading of the nodes. Collaborative Cloud Computing (CCC) generates the nodes which are widely and globally scattered over distributed areas and hence sharing of resources is much more reliability. Our literature paper helps to study about collaborative cloud computing along with cloud knowledge which is responsible for searching and discovering other mobile resources, connecting, maintaining, connections and communicating with external device.

V. REFERENCES

[1] Haiying Shen, Senior Member, IEEE, Guoxin Liu, Student Member, IEEE, "An Efficient and

Trustworthy Resource Sharing Platform for Collaborative Cloud Computing", IEEE Transactions on Parallel and Distributed Systems.

[2] Amazon elastic compute cloud (EC2). http://aws .amazon.com.

[3] Drop box. Available: www.dropbox.com. [4] P. Suresh Kumar, P. Sateesh Kumar, and S. Ramachandram. Recent Trust Models in Grid. JATIT, 2011.

[5] J. Li, B. Li, Z. Du, and L. Meng. CloudVO: Building a Secure Virtual Organization for Multiple Clouds Collaboration. In Proc. of SNPD, 2010.

[6] C. Liu, B. T. Loo, and Y. Mao. Declarative Automated Cloud Resource Orchestration. In Proc. of SOCC, 2011.

[7] C. Liu, Y. Mao, J. E. Van der Merwe, and M. F. Fernandez. Cloud Resource Orchestration: A Data Centric Approach. In Proc. of CIDR, 2011.

[8] K. Hwang, S. Kulkarni, and Y. Hu. Cloud Security with Virtualized Defense and Reputation-based Trust Management. In Proc. of DASC, 2009.

[9] IBM Red Boo. Fundamentals of Grid Computing. Technical Report REDP-3613-00, 2000. [10] L. Xiong and L. Liu. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. TKDE, 2004.

11] M. Srivatsa, L. Xiong, and L. Liu. Trust guard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In Proc. of World Wide Web Conference, 2005.

[12] R. Zhou and K. Hwang. Power Trust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. IEEE TPDS, 2008