Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

Efficient and Secure Sharing of Resources in Harmony for Collaborative Cloud Computing

Soumya Thomas[1], Soumi C.G[2]
[1,2]*Department of Computer Science, IIET, Nellikuzhi*

Abstract- Advancements in cloud computing provides a promising future for Collaborative Cloud Computing(CCC). Distributed resources belonging to different organizations or individuals are collectively used in a cooperative manner to provide various services to cloud resources. In cloud environment resources are shared on-demand basis. Due to the various autonomous features of objects, resource management and reputation management to ensure the successful deployment of CCC. Simply combining these two systems generates double overhead. Previous resource and reputation management methods are not sufficiently efficient or effective. So introduces the concept of Harmony platform which integrates the resource and reputation management. Also provides multiple QoS oriented resource selection and price-assisted resource/reputation control. Job migration is combined with this Harmony platform to provide efficient load balancing across nodes. Load balancing can be efficiently done with the help of Hotspot detection.

Keywords- Distributed systems, reputation management, resource management, distributed hash tables, cloud computing

I. INTRODUCTION

Cloud computing is a compelling technology. In clouds, clients can dynamically allocate their resources on-demand without sophisticated deployment and management of resources. CLOUD computing has become a popular computing paradigm, in which cloud providers offer scalable resources over the Internet to customers. 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. Currently, many clouds, such as Amazon's EC2, Google's AppEngine, IBM's Blue- Cloud, and Microsoft's Azure, provide various services. Cloud customers are charged by the actual usage of computing resources, storage, and bandwidth. A single cloud may not be able to provide sufficient resources for an application (especially during a peak time). 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 [3]. Key enabling technologies for clouds include the MapReduce programming paradigm, distributed file systems, virtualization, and so forth. These techniques emphasize scalability, so clouds can be large in scale, and comprising entities can arbitrarily fail and join while maintaining system reliability.

Distributed file systems are key building blocks for cloud computing applications based on the MapReduce programming paradigm. CCC operates in a large-scale environment involving thousands or millions of resources across disparate geographically distributed areas, and it is 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 (resMgt) (i.e., resource location and resource utilization) 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. As shown in Figure:1, 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.

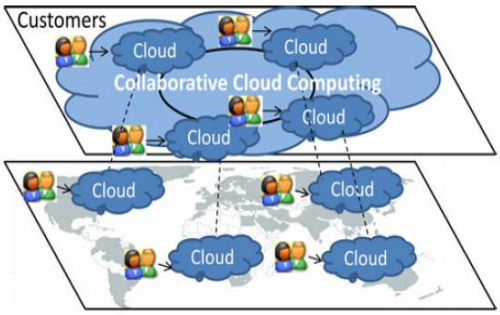

Fig. 1. Collaborative cloud computing

Thus, resMgt needs reputation management (repMgt) to measure resource provision QoS for guiding resource provider selection. As in eBay and Amazon, a repMgt system computes each node's reputation value based on evaluations from others about its performance in order to provide guidance in selecting trustworthy resources. 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.

The three tasks must be executed in a distributed manner since centralized methods are not suitable for large-scale CCC. However, though 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.

II. RELATED WORKS

This research work focus on the area collaborative cloud computing. Advancements in cloud computing are leading to a promising future for collaborative cloud computing (CCC). Collaborative Cloud Computing method uses two different types of mechanism for activating the cloud server they are resource management and reputation management technique therefore we can avoid over node of traffic congestion hence we can generate high quality of double bandwidth over node. In the previous research work a single value will be added to the reputation management so that it can represent every node with the unique ID, even though the value has been assigned to the every node the server could not able to identify resources to individual types of user's therefore QoS (Quality of Service) could not able to meet by the resources selection allocator.

Trustworthy Resource Sharing on Collaborative Cloud Computing : 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 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, 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.

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.

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 combine both resource management and reputation management in CCC, it is creating high overhead. 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.

Resource Allocation in Collaborative Cloud Based on Multi-QoS : High increase in the demand and advancements in cloud computing is giving rise to a promising future for collaborative cloud computing (CCC). Also, the previously addressed methods for these issues are not much effective and efficient. In previous research method single reputation value of each node was provided and it could not reflect the reputation of a node in providing different types of resource. In this case initially reputation is same for all node, then reputation is calculated based on the execution of the particular job on a node. Cloud computing mainly focuses on maximizing the effectiveness on the shared resources which are shared by multiple users and sometimes dynamically reallocated as per the demand. So for a successful deployment of CCC, resource management and reputation management issues have to be addressed jointly. In previous researches the above two issues have been addressed separately and simply combining these two systems may lead to dependency between the two i.e. resMgt needs repMgt for a cooperative environment for resource sharing and repMgt is required for multi-faceted node reputations for providing resources. In current methods, always a highest reputed node is selected which would lead to overloading of these nodes. . By always recommending the highest reputed nodes, the methods are failing in selecting a suitable node for resource selection to satisfy the user's diverse QoS demands. Hence a CCC platform is proposed known as Harmony which integrates both reputation and resource management in a harmonious manner.

III. PROPOSED WORK

In collaborative cloud computing (CCC), globally-scattered distributed cloud resources belonging to different organizations or individuals (i.e., entities) are collectively used in a cooperative manner to provide services. CCC operates in a large-scale environment involving thousands or millions of resources across disparate geographically distributed areas, and it is also inherently dynamic as entities may enter or leave the system and resource utilization and availability are

continuously changing. Here introduce a CCC platform, called Harmony, which integrates resource management and reputation management 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. Harmony outperforms existing resource management and reputation management systems in terms of QoS, efficiency and effectiveness.

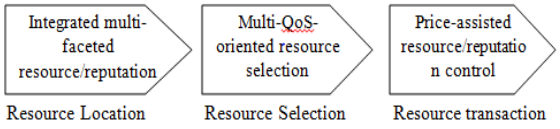

Fig. 2. Harmony components in resource market stages

By identifying and understanding the interdependencies between resMgt and repMgt, introduce Harmony, a CCC platform with harmoniously integrated resMgt and repMgt. It can achieve enhanced and joint management of resources and reputation across distributed resources in CCC. Different from the previous resMgt and repMgt methods, Harmony enables a node to locate its desired resources and also find the reputation of the located resources, so that a client can choose resource providers not only by resource availability but also by the provider's reputation of providing the resource. In addition, Harmony can deal with the challenges of large scale and dynamism in the complex environment of CCC.

Design Harmony of Integrated Multi-Faceted Res/Rep Management: Harmony leverages the Cycloid hierarchically structured P2P overlay for its substrate. It has at most n= d * 2ˆd nodes, where d is its dimension. Each Cycloid node's ID is represented by a pair of indices (k, ad-1,ad-2 . . . a0), where k€[0, d-1] is a cyclic index and ad-1,ad-2.....a0 €[2d -1]is a cubical index. For a given key or node IP address, its cyclic index is its consistent hash value modulated by d and its cubical index is the hash value divided by d. All nodes are grouped into different clusters, which are identified by ad-1, ad-2.....a0. Within a cluster, the nodes are differentiated by k. In the Cycloid key assignment policy, an object/key is assigned to the node whose ID is closest to the object/key's ID. It provides two main functions: Insert(ID, object) and Lookup(ID) to store an object in its owner node and to retrieve the object, respectively.

Multi-QoS Oriented Resource Selection: A direc- tory node locates the resource providers that have the required reputation, available amount, and price, it needs to choose provider(s) for the requester. The final QoS offered by a provider is determined by a number of factors such as efficiency, trustworthiness, distance, security and price. Call these factors QoS demands (or attributes). When choosing from a number of providers, most previous approaches rigidly consider a single QoS demand at a time. A challenge here is how to consider individual or combined QoS attributes, and a user's desired priorities of the attributes in provider selection. Harmony solves this problem by unifying all attribute values and a client's considered attribute priority into an overall QoS metric. eBay's reputation system asks users to provide feedback on different aspects such as item description and shipping charge. Similarly, Harmony utilizes a list of QoS attributes. The overall QoS is actually a result of the combined influence from the QoS attributes. However, it is not easy to detect how the different attributes influence the overall QoS. Harmony depends on a neural network to find out the influence weight of each attribute on the overall QoS value, and further considers users' attribute consideration priority.

Price-Assisted Resource/Reputation Control: Previous repMgt methods always encourage nodes to choose the highest-reputed node as the server. However, with the highest-reputed server selection policy, a high reputed server easily becomes overloaded. This server selection policy is effective when nodes have unlimited resources, but it does not hold true in CCC, where each node

has limited resources. Always choosing the highest reputed node will overload that node, which then cannot offer high-QoS and receives low ratings from others. Then, the node will have low reputation and few chances to be chosen as a server in order to earn credits and increase its reputation. Though a node in Harmony chooses a server with a record of sufficient resources in the directory node, the server still would be overloaded due to delayed updates of the resource information or many simultaneous requests. Therefore, a challenge here is how to enable a node to keep a high reputation, fully utilize its resources and avoid being overloaded. By fine-tuning its price, a node can control the calculated overall QoS, thus controlling its own load and reputation. Hence, Harmony takes advantage of the price to avoid overloading nodes and strengthen cooperative and high-QoS resource sharing. Specifically, a node adaptively adjusts its resource price according to its load in order to always remain high-reputed and avoid being overloaded while gaining the highest income. Define a load factor f=l/c, where l is the amount of a 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 > $\propto$(0.8<=$\propto$ < 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.

Combination of Three Components: The pseudo- code of the algorithm executed by node I in Harmony combining three components. In the integrated multifaceted resource/reputation manageme nt, a node periodically reports its specified resource prices along with its available resource amount. When a node needs a resource, it sends a request with its desired price, amount, time period, and provider's reputation. After receiving a request, a directory node searches its directory, and finds all providers that have qualified resources satisfying the requester's requirements.

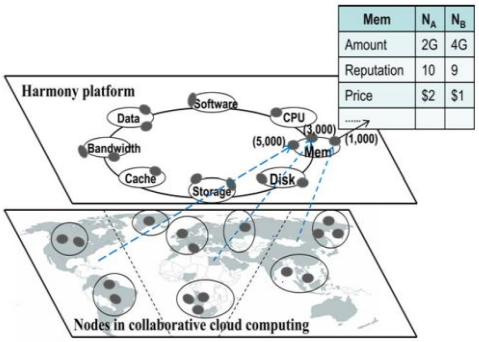

Fig. 3. The Harmony platform

If a node's reputation for a resource is increased, it has a higher probability of being selected as a resource provider. Then, it gains more opportunities to earn credits and can have enough credits to buy its desired resources. On the other hand, if a node's reputation for a resource is decreased, it has a lower probability of being selected as a resource provider.

Algorithm 1: Resource Sharing in Collaborative Cloud Computing

Input: Programs or functions that are need to be executed.

Output: Transactional details and the final output

Steps:

Step1: Initialize server and its subservers or nodes

Step2: Establish connection between subserver and servers

Step3: Client should register and login to perform the required action.

Step4: Obect/key should be assigned to each node.

Step5: Use Insert(ID,Object) to store an object in its owner node.

Step6: Use Lookup(ID) to retrieve an object in its owner node.

Step7: Call Process procedure to perform a transaction.

Step8: Process procedure select appropriate subserver node to perform the action.

Step9: Calculate the activation function QoS=$\sum_{i=1}^{n} Wi * Ai$.

Step11:Using the QoS attribute select the appropriate node for performing transaction.

Job Migration: So far, have been mainly concerned with cloud computing in which communication is limited to passing data. However, there are situations in which passing programs, sometimes even while they are being executed, simplifies the design of cloud computing. Traditionally, job migration in cloud computing took place in the form of process migration in which an entire process was moved from one machine to another. Moving a running process to a different machine is a costly and intricate task, and there had better be a good reason for doing so. That reason has always been performance. The basic idea is that overall system performance can be improved if processes are moved from heavily-loaded to lightly-loaded machines. Load is often expressed in terms of the CPU queue length or CPU utilization, but other performance indicators are used as well. Load distribution algorithms by which decisions are made concerning the allocation and redistribution of tasks with respect to a set of processors, play an important role in compute-intensive systems. Here client can communicate with the server to perform various functions as in a client-server architecture. In cloud environment, there are number of machines to perform various functions. Harmony platform with job migration module is included to perform the transactions. In this way, it enables nodes to simultaneously access the information and reputation of available individual resources.

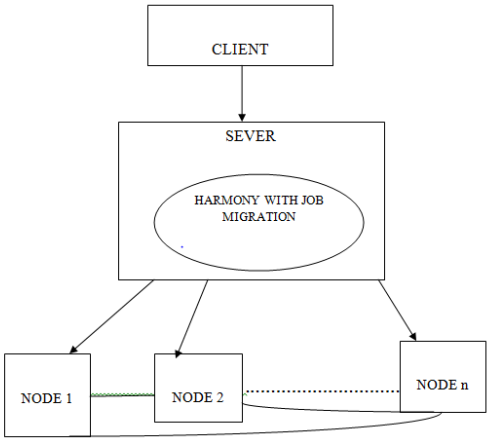

Fig. 4. Harmony with job migration

It also enables a client to perform resource selection with joint consideration of diverse QoS requirements, such as reputation, efficiency, distance, and price, with different priorities. Harmony employs a trading model for resource transactions in resource sharing and leverages the resource price to control each node's resource use and reputation. Moving a running process to a different machine is a costly and intricate task, and there had better be a good reason for doing so. That reason

has always been performance. The basic idea is that overall system performance can be improved if processes are moved from heavily-loaded to lightly-loaded machines.

Algorithm 2: Job Migration with Harmony in Collaboration Cloud Computing

Input: Programs or functions that are need to be executed and migration of job is performed if needed.

Output: Final result of the program or transaction.

Steps:

Step1: Initialize server and its subservers or nodes

Step2: Establish connection between subserver and servers

Step3: Client should register and login to perform the required action.

Step4: Client should enter the job(n) needed to run.

Step5: Identify the situations that when a migration is take place with the help of Hotspot
Step6: Calculate the threshold=avg*1.1 and check this with the current value to detect Hotspot.

Step7: Find the job scheduling by ordering the jobs according to the length and detect the Hotspot situations to find which job should migrate.

Step8: Check the fact that where to migrate a job.

Step9: Consider previous migration state and calculate the reputation and speed of the node

Step10:Based on the above attributes migration should perform.

IV. RESULTS

Performance of the system can be analyzed with the help of cloud simulator. Firstly analyze the case of wait time against the job size.

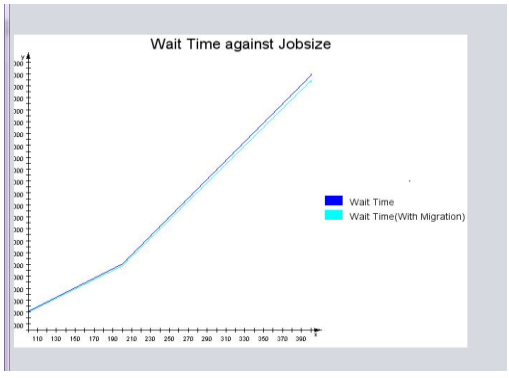

Fig. 5. Wait time against Job size

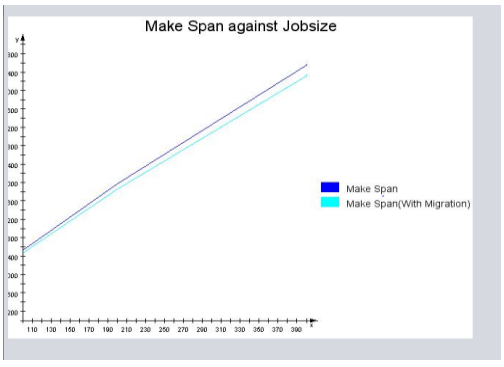

Fig. 6. Make Span against Jobsize

Next is to check the make span against the job size. Here given n jobs J_1, J_2, ..., J_n of varying sizes, which need to be scheduled on m identical machines, while trying to minimize the makespan. The makespan is the total length of the schedule (that is, when all the jobs have finished processing).

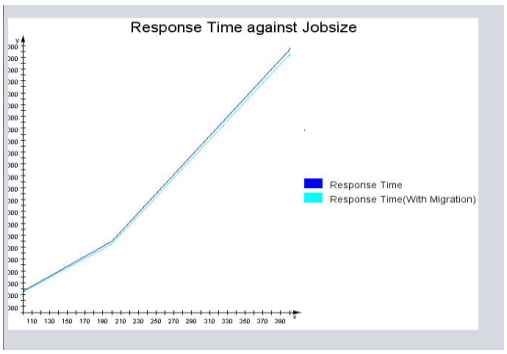

Fig. 7. Response Time against Jobsize

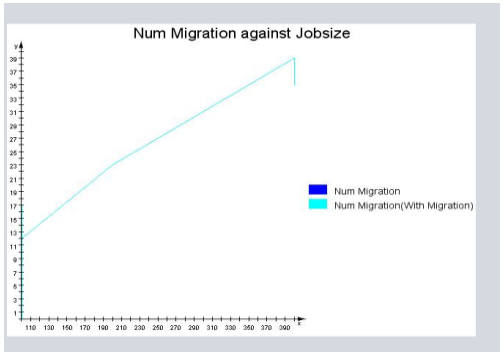

Fig. 8. Number of Migration against Jobsize

These plots show the wait times on our core systems vs. job size, for all jobs submitted, including jobs that have not yet started, and jobs that were cancelled while waiting. So the main aim is to reduce waiting time. Identifying the fact that how response time varies with the job size. Response time will normally increases with job size. Size-based schedulers have very desirable performance properties: optimal or near-optimal response time can be coupled with strong fairness. Despite this, however, such systems are rarely implemented in practical settings, because they require knowing a priori the amount of work needed to complete jobs: this assumption is difficult to satisfy in concrete systems. Next is to consider the number of migrations against the job size. Number of migration is completely depends upon the size of the job and performance of the node. Identify the last time situation with jobsize. Last time of the job is the last time at which the job is go to its final state of completion.

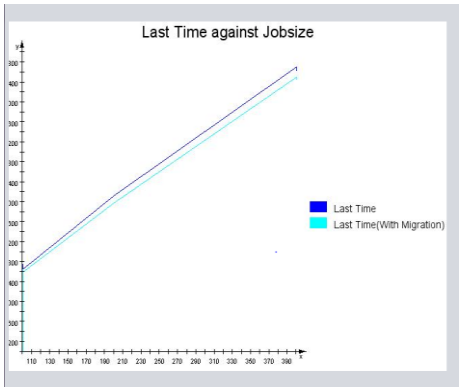

Fig. 9. Last Time against Jobsize

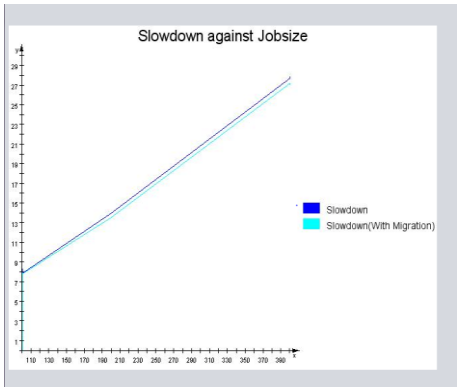

Fig. 10. Slowdown against Jobsize

Next is to check the slowdown against the jobsize. Assuming no new jobs arrive, the shortest processing-time-product(SPTP) Schedule is known to minimize the slowdown of the existing jobs.

V. CONCLUSION

Harmony recognizing the interdependencies between resource management and reputation management, which incorporates three innovative components to enhance their mutual interactions for efficient and trustworthy resource sharing among clouds. The integrated resource/ reputation management component efficiently and effectively collects and provides information about available resources and reputations of providers for providing the types of resources. The multi-QoS-oriented resource selection component helps requesters choose resource providers that offer the highest QoS

measured by the requesters' priority consideration of multiple QoS attributes. The price-assisted resource/reput- ation control component provides incentives for nodes to offer high QoS in providing resources. Job Migration is introduced for a perfect load balancing across the nodes. Here check the three conditions that when, which, where to migrate. Based on the Hotspot detection it decides when and which migrate. Hotspot is detected based on the comparison of attributes to its threshold value. Then the next is to decides where to migrate and is identified based on the previous history of the migration. Reputation of each node is calculated based on the current attributes and previous working conditions.

As a future work it is possible to add the concept of distance based on the geographical position of each node. Therefore reputation of each node is also depend upon the distance attribute.

REFERENCES

[1] A. Satsiou and L. Tassiulas, "Reputation Based Resource Allocation in P2P Systems of national Users," I EEE Trans. Parallel and Distributed Systems, vol.21, no.4, pp.466-479, Apr. 2010.
[2] C. Liu, Y. Mao, J.E. Van der Merwe,and M. Fernandez, "Cloud Resource Orchestration: A Data Centric Approach," Proc.Conf.Innovative Data Systems Research (CIDR) , 2011.
[3] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for P2P Systems,"Proc.16th ACM Symp.Parallel Algorithms and Architectures (SPAA'04), pp. 3643, June2004.
[4] D. Talia, P. Trunfio, J.Zeng, and M.H€ogqv ist, "A DHTBased P2Peer Framework for Resource Discove ry in Grids,"Technical Report TR-0048, CoreGRID-Network of Excellence, 2006.
[5] H.Shen and G. Liu, "Harmony: Integrated Resource and Reputation Management for LargeScale Distribut ed Systems," Proc.20th Int'l Conf. Computer Command Networks (ICCCN), 2011.
[6] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 4, pp. 460- 473, 2008.
[7] R. Zhou and K. Hwang, "Gossip-Based Reputation Management for Unstructured Peer-to-Peer Networks," IEEE Trans. Knowledge and Data Eng., 2007.
[8] S. Chaisiri, B.S. Lee, and D. Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," IEEE Trans. Services Computing, vol.5, no.2, pp.164-177, Apr.-June 2012.
[9] S. Di and C. Wang, "Dynamic Optimization of Multiattribute Resource Allocation in Self- Organizing Clouds," IEEE Trans. Parallel and Distributed Systems, vol.24, no. 3, pp. 464-478, Mar. 2013.
[10] S. Lee, X. Ren, and R. Eigenmann, "Efficient Content Search in iShare, A P2P Based Internet-Sharing System," Proc.IEEE Int'l Symp. Parallel and Distributed Processing(IPDPS), 2008.

