Harmony: Integrated Resource and Reputation Management for Large-Scale Distributed Systems

Haiying Shen and Guoxin Liu
Department of Electrical and Computer Engineering
Clemson University, Clemson, SC 29631
Email: {shenh, guoxinl}@clemson.edu

***Abstract*—Advancements in technology over the past decade are leading to a promising future for large-scale distributed systems, where globally-scattered distributed resources are collectively pooled and used in a cooperative manner to achieve unprecedented petascale supercomputing capabilities. The issues of resource management (resMgt) and reputation management (repMgt) need to be addressed in order to ensure the successful deployment of large-scale distributed systems. However, these two issues have typically been addressed separately, despite the significant interdependencies between them: resMgt needs repMgt to provide a cooperative environment for resource sharing, and in turn facilitates repMgt to evaluate multi-faceted node reputations for providing different resources. Current repMgt methods provide a single reputation value for each node in providing all types of resources. However, a node willing to provide one resource may not be willing to provide another resource. In addition, current repMgt methods often guide node selection policy to select the highest-reputed nodes, which may overload these nodes. Also, few works exploited node reputation in resource selection in order to fully and fairly utilize resources in the system and to meet users' diverse QoS demands. We propose a system called Harmony that integrates resMgt and repMgt in a harmonious manner. Harmony incorporates two key innovations: integrated multi-faceted resource/reputation management and multi-QoS-oriented resource selection. The trace data we collected from an online trading platform confirms the importance of multi-faceted reputation and potential problems with highest-reputed node selection. Trace-driven experiments performed on PlanetLab show that Harmony outperforms existing resMgt and repMgt in terms of the success rate, service delay, and efficiency.**

I. Introduction

Advancements in technology over the past decade are leading to a promising future for large-scale distributed systems, where globally-scattered distributed resources are collectively pooled and used in a cooperative manner to achieve unprecedented petascale supercomputing capabilities. A large-scale distributed system interconnects geographically distributed computing and storage resources to make possible the sharing of the resources. Rapid development of large-scale distributed systems has stimulated a proliferation of applications such as BitTorrent [1] data sharing and the OGSA [2] computational grid.

Large-scale distributed systems applications are likely to become increasingly prevalent. The applications operate in a large-scale, inherently dynamic environment involving resources across disparate administrative domains. This environment makes efficient resource sharing a non-trivial task. Ubiquitous users without preexisting reputation relationships in the system also pose a challenge for cooperative resource sharing. For example, if *a user needs the resources "OS name=Linux", "CPU speed=1000MHz", and "Memory=1024MB" for a computing task, how can it efficiently choose trustworthy resources?* While many technologies are important to achieving the objective of efficient and trustworthy resource sharing, perhaps two of the most essential issues to address are resource management (resMgt) and reputation management (repMgt). ResMgt involves resource discovery and allocation for high system efficiency. 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 nodes for high system reliability and security.

However, these two issues have typically been addressed separately, despite the significant interdependencies between them; resMgt needs repMgt to provide a cooperative environment for collaborative resource sharing and in turn facilitates repMgt to evaluate multi-faceted node reputations for providing various resources. Simply building and combining individual resMgt and repMgt systems in large-scale distributed systems will generate doubled, prohibitively high overhead.

Dependency of repMgt on resMgt. Previous repMgt systems [3]–[7] 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. For example, a person trusts a doctor for giving advice on medical issues but not on financial ones. Similarly, a node performing well with processing services does not mean it also performs well with storage services. Thus, previous repMgt systems are not effective enough to provide correct guidance for trustworthy individual resource selection. RepMgt needs to rely on resMgt for reputation differentiation across multiple resources.

Dependency of resMgt on repMgt. Since each of the nodes belongs to different authorities or individuals and may be constrained by computing resources, a selfish node may not be willing to provide resources if it won't receive benefits in return. It was reported that over 70% of nodes in a file sharing system contributed nothing in return [8]. In addition, malicious nodes may launch attacks such as spoofing attacks by lying about their available resources to attract requesters (i.e., clients) but offering low-level resources. Non-cooperative behavior is an obstacle to the wide acceptance of resource sharing in large-scale distributed systems. Therefore, resMgt [9]–

[15] needs to depend on repMgt for a cooperative and secure environment.

Given a number of resource providers (i.e., servers), the efficiency-oriented resMgt policy would choose the one with the highest available resource, while the trust-oriented repMgt policy would choose the one with the highest reputation. The former may lead to low service success rate while the latter will overload the node by many resource requests. The overloaded node then cannot provide high Quality of Service (QoS) and subsequently receives low reputation. The drawbacks of both resource selection policies pose a challenge: how to jointly consider multiple QoS demands, including reputation, efficiency, and available resource amount, in resource selection.

By identifying and understanding these interdependencies and challenges, this paper introduces Harmony, a system built on a large-scale distributed system with harmoniously integrated resMgt and repMgt (res/repMgt). It can achieve enhanced and joint management of resources and reputation across a distributed system. The contributions of this work can be summarized as below:

(1) *Integrated multi-faceted resource/reputation management.* 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 repository node. In this way, it enables nodes to simultaneously access the information and reputation of available individual resources.

(2) *Multi-QoS-oriented resource selection.* Harmony enables a client to perform resource selection with joint consideration of diverse QoS requirements, such as reputation, efficiency, distance, and price, with different priorities.

(3) *Real trace study and trace-driven experiments on PlanetLab [16].* We analyzed the transaction and feedback rating data we collected from an online trading platform (Zol [17]), and verified the importance of multi-faceted reputations and the potential problem with the highest-reputed node selection policy. Some sellers offer high QoS in providing one type of merchandise but offer low QoS in providing another type. Also, buyers tend to buy merchandise from high-reputed sellers. Trace-driven experimental results show the superior performance of Harmony in comparison with current resMgt and repMgt systems, and the effectiveness of its components.

This work is the first to integrate repMgt with resMgt for multi-faceted node reputation evaluation to provide correct guidance for resource selection. The rest of this paper is structured as follows. Section 2 details the design of Harmony. Section 3 analyzes our crawled trace data and presents the trace-driven experimental results obtained by running a Harmony prototype on PlanetLab, in comparison with current resMgt and repMgt approaches. Section 4 presents a concise review of representative reputation systems and economic marketing approaches. Finally, Section 5 summarizes this paper.

II. Related Works

For high scalability and efficiency, many resource management systems based on DHTs have been proposed for grid resource management [9]–[13], [18]. Some efforts depend on multiple DHTs with each DHT responsible for one resource [9]–[11]. Other efforts depend on a single DHT overlay [12], [13], [18]. The former generates high maintenance overhead due to multiple DHTs, while the latter methods may generate bottlenecks because one node is responsible for all resource information of one resource. HCO [14] and PIRD [15], rely on the hierarchical structure of the Cycloid [19] DHT for resource management. HCO clusters resources so that a node can always locate physically close multiple resources in its neighborhood. PIRD weaves multiple resources to one index for multi-resource discovery.

In other resource management works, Erdil *et al.* [20] studied the performance characteristics of epidemic and pairwise gossiping, and showed that pairwise gossiping protocols work best when resource distribution on the grid is uniform. Dai *et al.* [21] modeled grid resource availability by considering both the failures of resource management servers and the length limitation of request queues. iShare [22] relies on a DHT to facilitate the sharing of diverse resources located in different administrative domains over the Internet. Yan *et al.* [23] incorporated peers' behavioral rankings into the resource allocation to provide differentiated services.

Many reputation systems have been proposed that aim to improve scalability or accuracy of reputation calculation. These works include PeerTrust [3], TrustGuard [4], PowerTrust [5], GossipTrust [6], and those in [4], [7]. For scalability improvement, they use a DHT to map the reputation reports on a node and reputation queries on the node. The systems also provide different methods to calculate a node's reputation in order to more accurately reflect node trustworthiness. Zhang *et al.* [24] presented a reputation system built upon the multivariate Bayesian inference theory for reliable service selection. All of these reputation systems give a node one global reputation value; thus, are not effective enough to provide correct guidance for trustworthy individual resource selection. Harmony's concept of multi-faceted reputation shares similarity with that in Wang *et al.* [25]'s work. However, their work focuses on presenting different trust aspects of one file offering service and combining them for a global trust, while Harmony focuses on the reputation differentiation of different resources.

III. The Design of Harmony

A. Integrated Multi-Faceted Resource/Reputation Management

Harmony leverages the Cycloid hierarchically structured P2P overlay [19] for its substrate. Cycloid has at most $n{=}d \cdot 2^d$ nodes, where d is its dimension. Each Cycloid node is represented by a pair of indices $(k, a_{d-1}a_{d-2}\ldots a_0)$, where k is a cyclic index $\in [0, d{-}1]$ and $a_{d-1}a_{d-2}.....a_0$ is a cubical index $\in [0, 2^d - 1]$. For a given key or node IP address, its cyclic index is its consistent hash value [26] modulated by d

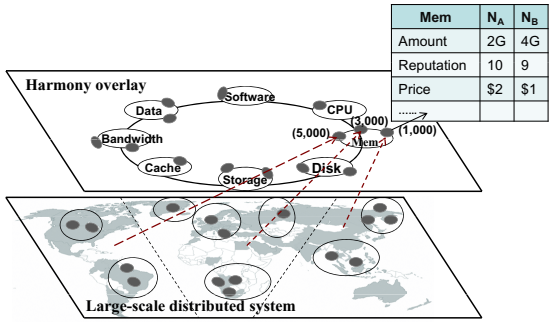


Fig. 1. Harmony infrastructure.

and its cubical index is the hash value divided by d. All nodes are grouped into different clusters, which are identified by $a_{d-1}a_{d-2}.....a_0$. Within a cluster, the nodes are differentiated by k. Cycloid assigns a key to a node whose ID is closest to the 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. It achieves a time complexity of $O(\log n)$ per lookup request by using 7 neighbors per node, where n is the number of nodes.

We assume that resource types, such as CPU, bandwidth and memory, are globally defined and known by every node. The resource information (denoted by I_r) includes the resource owner's IP address, the resource type, the available amount, the resource physical location, the price, etc. As shown in Figure 1, Harmony proactively collects all resource information I_r of each resource type in a cluster. Thus, to search for one resource, a node only needs to probe nodes in one cluster rather than executing system-wide probing.

Mapping each individual resource to one cluster enables Harmony to deal with the challenge of reputation differentiation. Similarly, Harmony distinguishes the reputation feedback of a resource provider by resource types and maps the reputation feedback of one resource type to the corresponding resource cluster. For example, the reputation feedback of a node's "Mem" resource provision is mapped to the cluster for the "Mem" resource. As a result, for each specific resource type, Harmony pools together the resource information on available resources and the reputation of resource providers to facilitate resource searching. According to the reputation and resource information, a resource requester can select a resource provider.

To enable users to find physically close resources within a cluster, Harmony further groups the resource information of physically close nodes into one node, called a *directory node*. Directory nodes periodically collect I_r and requests, and function as matchmakers between resource requesters and providers. Harmony adopts the hash-based proximity clustering approach introduced in our previous work [27] that relies on DHT functions to collect the information of physically close nodes together. The clustering approach introduces the *Hilbert number* ($\mathcal{H}$) that represents a node's geographical location so that the distances between nodes can be inferred from the distance between the Hilbert numbers. Physically close nodes have closer $\mathcal{H}$ values.

With the design described above, for resource reporting, node i reports its available resource r by $Insert((\mathcal{H}_i, H_r),I_r)$, where H_r is the consistent value of resource r's name. Based on the Cycloid key assignment policy, information for the same resource type will be collected in the same cluster, and that information is further distributed among the nodes within the cluster based on resource proximity. In Figure 1, for example, the messages for the memory resource of physically close nodes are forwarded to the same node in the "Mem" cluster. To query multiple resources, node i sends out $Lookup(\mathcal{H}_i,H_r)$ requests along with the desired amount, time period, reputation, price, etc., with one request for each resource type. Each request is forwarded to the resource's directory node, which has the information on the requested resource that is physically close to the requester. If the directory node has no I_r satisfying the requests (i.e., the amount and reputation are higher than the requested values and the price is lower than the requested value), it randomly probes its neighbor nodes in its cluster in an increasing range of proximity. After the directory node finds an I_r satisfying the request, it uses the *multi-QoS-oriented resource selection* algorithm (Section III-B) to select the best server(s) for the client.

Upon receiving the satisfying I_r, the client further chooses the servers with the highest reputation and the lowest resource price, and then randomly chooses one to ask for the resource. After the client finishes using the requested resource, it reports its reputation feedback on the server's resource provision service using $Insert((\mathcal{H}_s, H_r), R)$, where s represents the server and R denotes the reputation. According to the Cycloid key assignment policy, the feedback receiver is exactly the directory node for resource r of server s. Therefore, all reputation feedback of server s for providing the "Mem" resource is collected in this directory node. All resource information on "Mem" for server s is also collected in this directory node. The directory node periodically calculates node s's reputation value on providing the "Mem" resource based on feedback using a reputation calculation method [3]–[7].

P2P's self-organization mechanism helps Harmony to handle node dynamism. In the mechanism, nodes update their neighbors periodically and transfer resource and reputation information based on the DHT key assignment policy when joining or leaving. Also, before a node departs from the system or after a node joins the system, it notifies the directory nodes of its resources. Thus, a node's resource information is always stored in its directory nodes even in dynamism, and $Lookup(ID)$ requests can always be forwarded to the directory nodes.

B. Multi-QoS-oriented Resource Selection

After a directory node locates the providers that have the required reputation, available amount, and price, it needs to choose a 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. We call these factors QoS demands (or attributes). A challenge here is how to consider individual or combined QoS attributes

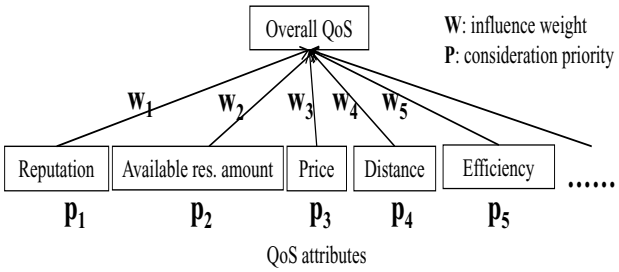
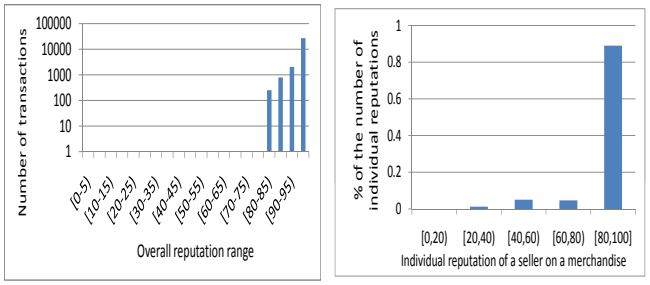


Fig. 2. A neural network model for resource selection.



(a) Total transactions vs. reputation. (b) Distribution of individual reputation.

Fig. 3. Trace data analytical results.

with a user's desired priorities of the attributes in provider selection.

Harmony solves this problem by unifying all attribute values and a client's consideration priority of different attributes to a profit 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. It requires nodes to give ratings for each QoS attribute and for overall QoS in addition to a reputation for the server. As with the reputation feedback, the QoS ratings are collected at the directory node for the provided resource of the server. 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 [28] to find out the influence weight of each attribute on the overall QoS value, and further considers users' attribute consideration priority.

Each directory node builds a neural network model as shown in Figure 2. Its inputs are the ratings of QoS attributes such as reputation, available resource amount, distance price and so on, and its output is the overall QoS value for a provider as a combined effect of the different attributes. The training process of the neural network is the process of determining the weight of influence of each attribute on the QoS. Suppose the attribute influence weights are $\{w_1, w_2, \cdots, w_5\}$ after training. These weights reflect the normal influence of different QoS attributes on the overall QoS.

Individual nodes have their own priorities when considering the QoS attributes. A requester sends its consideration priority weight $p_i (\sum p_i = 1)$ along with its resource request. After the directory node locates the satisfied resource providers, it calculates the profit value for each server option by considering both the normal influence of QoS attributes on the overall QoS and the requester's consideration priority. To do this, the directory node changes w in the neural network model: $w_i = p_i w_i$, as shown in Figure 2. It then inputs each server's attribute values $\{A_1, A_2, \cdots, A_5\}$ to the neural network. The output of the neural network is the profit. Finally, the directory node determines the server(s) with the highest profit value. Each directory node trains its neural network model periodically using its newly collected QoS ratings.

IV. Trace Data Analysis and Performance Evaluation

A. Trace Data Analysis and Experiment Settings

Zol, like Amazon and eBay, is a top online e-commerce platform in China. We collected trace data of 30,049 transaction records from Zol covering the period from 9/20/2006 to 6/26/2010. Each record represents 52 transactions on average. We chose Zol because neither Amazon nor eBay provides the historical rating record for each transaction. Zol provides the rating scores [0,100] in five QoS attributes of each transaction: 1) price; 2) distance; 3) quality; 4) service; 5) efficiency. We calculated the average of the five ratings as the seller's reputation for a transaction. Figure 3(a) shows the number of transactions versus the seller's overall reputation. We see that clients tend to choose higher-reputed nodes. In this case, if the resource a node possesses is limited, the highest-reputed nodes can easily become overloaded.

We then searched 60 merchandise keywords in these transactions and found 2231 transactions. We classified the transactions of the same merchandise of a seller into a group. We calculated the average reputation of each group as the reputation of the seller for the merchandise (we call this individual reputation). Finally, we derived the individual reputations of each product, the overall reputation of the seller and the number of transactions for 50 sellers. We found that the lowest overall reputation of these sellers is 89. Figure 3(b) shows the distribution of the individual reputations of these sellers. We find that 90% of individual reputations of these sellers are in the range of [80,100], 4.4% are in [60,80], and 5.1% are in [40,60]. The results imply that even highly-reputed sellers sometimes offer low QoS for certain merchandise, which supports our claim that resource reputation should be multi-faceted. When choosing a resource supplier, we need to refer to the reputation of the suppliers in providing the resource rather than the overall reputation.

To validate the design of Harmony, we implemented a prototype on PlanetLab and conducted trace-driven experiments. We regard sellers as nodes, regard ratings as reputation values, and regard products in transactions as resources. Since around 234 nodes in the PlanetLab can be stably accessed, we set the number of nodes in the system to 234. We also set the number of resource types for each node to 12, which should be approximately the number of individual resources. From the

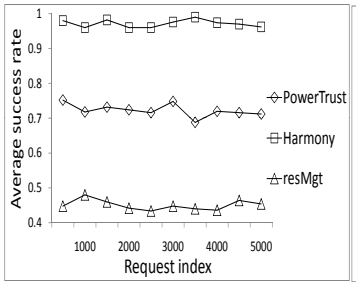
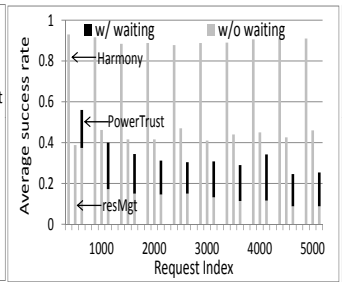
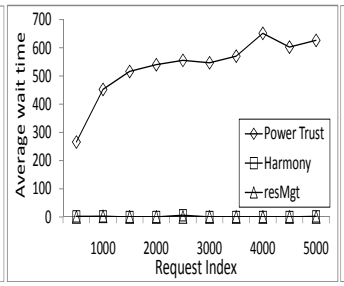


Fig. 4. The need for reputation management in resource management.



(a) Ave. success rate



(b) Ave. waiting time

Fig. 5. The need for resource management in reputation management.

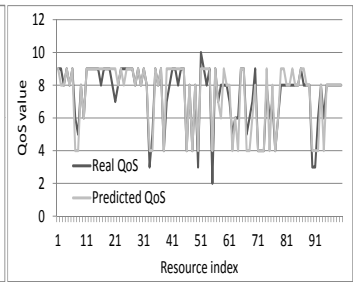


Fig. 6. The comparison of predicted and real QoS.

processed trace data, we identified 13 sellers who have more than 3 products and chose 3 products for each of the sellers. We multiplies the data of the 3 products of each seller by 4 for the data of its 12 products, and multiplied the data of 13 sellers by 18 and mapped the data to 234 nodes. We normalized the reputation values from [0,100] to [0,10] by dividing the original reputation by 10. The lowest overall reputation of these sellers is 8.9 in the trace data. Since real large-scale distributed systems should have high-, medium- and low-reputed nodes, we generated synthetic data for 78 nodes with overall reputations randomly chosen from [1,3] and [4,6], respectively. Their individual reputations equal their overall reputations. On PlanetLab, we selected 21 nodes (7 from the US, 7 from Europe and 7 from Asia) as landmark nodes for calculating node Hilbert numbers. We randomly chose 8 nodes in America, Europe and Asia as requesters. In the experiments, unless otherwise specified, each requester sends one request every 10 seconds for a resource randomly chosen from the 12 types. A resource provider with an individual reputation of t for a resource has a $t/10$ probability to provide this resource. We reported the 10 average values of each group of 500 requests over time as the experimental results.

In order to show the importance of integrating resMgt and repMgt, we first evaluated Harmony in comparison with Harmony without repMgt (denoted by resMgt) and the PowerTrust repMgt system [5]. To make the methods comparable, we use Harmony's structure and resource discovery algorithms for PowerTrust. These methods are only different in resource selection. After locating the providers satisfying a resource request, Harmony chooses the lightly loaded provider with the highest individual reputation, PowerTrust chooses the resource owner with the highest overall reputation, and resMgt randomly chooses a resource provider among lightly loaded nodes. We also evaluated Harmony with other resMgt methods that rely on a single DHT and multiple DHTs in order to show Harmony's higher efficiency in the environment of large-scale distributed systems.

B. Effectiveness of Multi-Faceted Reputation Management

1) The need for reputation management in resource management: ResMgt needs repMgt to provide a cooperative environment for resource sharing. Without repMgt, a node cannot know which resources are trustworthy. Also, resMgt in turn facilitates repMgt to evaluate multi-faceted node reputations for providing different resources. We assumed that nodes do not drop requests when overloaded and will process the requests later on in order to see the sole effect of reputation management. Figure 4 shows the average success rate of each system, which is measured by the ratio of successfully resolved resource requests over total requests. We see that Harmony achieves an over 96% success rate, while PowerTrust achieves around 73% and resMgt achieves around 44%. ResMgt selects a resource without consideration of reputation and may choose a resource provider with a low reputation for the requested resource, leading to a low success rate. PowerTrust always selects the highest-overall-reputed provider. As the Zol trace data shows, a node with a high overall reputation may not have a high individual reputation in providing a specific product. In large-scale distributed systems, a node with a high overall reputation for one resource may provide low QoS for another resource due to its unwillingness or overloaded status. Therefore, with its multi-faceted reputation management, Harmony dramatically improves the success rate of PowerTrust that always chooses the supplier with the highest individual reputation. The experimental results confirm the effectiveness of multi-faceted reputation management and its importance to resource management in guiding trustworthy resource selection.

2) The need for resource management in reputation management: Without resMgt, repMgt cannot tell whether a resource supplier has sufficient available resources. In this experiment, each node maintains a waiting queue. If a resource provider is overloaded when it receives a request, it puts the request into its waiting queue. The requests that have been in the queue for more than 1000s are dropped. We use the Pareto distribution [29] in determining a node's capacity for a resource type, a request's requested amount, and a time period. This distribution reflects the real world situation. We set the shape parameter to 2, and the scale parameters to 100, 40 and 200, respectively.

Figure 5(a) shows the success rates of Harmony, PowerTrust and resMgt. 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 PowerTrust generates a large number of delayed successful requests, while the others generate no delayed

requests. This result is consistent with Figure 5(b), which shows that PowerTrust generates high delay for a request, while Harmony and resMgt have little or no delay. This is because PowerTrust always chooses the overall highest-reputed nodes as the resource providers without considering node load status. Harmony relies on the lightly loaded individual highest-reputed node, while resMgt randomly chooses a lightly loaded node. Therefore, the highest-overall-reputed node in PowerTrust receives too many requests, causing many to wait in the queue.

C. Multi-QoS-oriented Resource Selection

In this experiment, we used 95×12 transaction records for 95 sellers and 12 products for each seller. We used 40×12 for training and the remaining 55×12 for testing. Due to a lack of trace data, we regard the individual reputation of a node as its QoS, and regard a node's overall reputation as its reputation in Harmony. The inputs of the neural network model include the QoS attributes price, distance, service, quality and efficiency in each transaction and the seller's overall reputation. The output of the model is the seller's individual reputation. Because the real trace does not have the users' consideration priority, we assume that the 6 QoS attributes have the same priorities. Figure 6 shows the predicted QoS and the real QoS for 100 resource requests. We see that the two curves are almost overlapping, thus demonstrating the effectiveness and accuracy of the neural network model in predicting the QoS in resource selection.

V. Conclusion

In this paper, we propose an efficient integrated resource/reputation management system called Harmony. Recognizing the interdependencies between resource management and reputation management, Harmony incorporates two innovative components to enhance their mutual interactions for high system QoS. *The integrated resource/reputation management* component efficiently and effectively collects and provides information about available resources and node reputations for each resource type. *The multi-QoS-oriented resource selection* component helps requesters choose resource providers that offer the highest QoS as measured by the requesters' priority considerations of multiple QoS attributes. The components collaborate together to enhance the efficiency and reliability of cooperative sharing of globally-scattered distributed resources in large-scale distributed systems to achieve petascale supercomputing. *Trace-driven experiments* on PlanetLab verified the effectiveness of the different Harmony components and the superior performance of Harmony in comparison with a resource management system and a reputation management system. The experimental results also show that Harmony achieves high scalability, balanced load, locality-awareness and dynamism-resilience in a large-scale distributed system environment.

Acknowledgment

This research was supported in part by U.S. NSF grants OCI-1064230, CNS-1049947, CNS-1025652, CNS-1025649, CNS-1057530 and CNS-0917056, Microsoft Research Faculty Fellowship 8300751, and Sandia National Laboratories grant 10002282.

References

[1] Bittorrent. http://en.wikipedia.org/wiki/Bittorrent.
[2] Towards Open Grid Services Architecture. http://www.globus.org/ogsa/.
[3] L. Xiong and L. Liu. Peertrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *IEEE TKDE*, 16(7):843–857, 2004.
[4] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proc. of World Wide Web Conference*, 2005.
[5] R. Zhou and K. Hwang. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE TPDS*, 2008.
[6] R. Zhou and K. Hwang. Gossip-based reputation management for unstructured peer-to-peer networks. *IEEE Transactions on Knowledge and Data Engineering*, 2007.
[7] A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proc. of P2P*, 2003.
[8] B. W. O'Hearn. Experiences Deploying a Large-Scale Emergent Network. In *Proc. of IPTPS*, 2002.
[9] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *Proc. of ACM SIGCOMM*, 2004.
[10] D. Talia, P. Trunfio, J. Zeng, and M. Högqvist. A DHT-based Peer-to-Peer Framework for Resource Discovery in Grids. Technical Report TR-0048, Univ. of California, 2006.
[11] S. Ratnasamy, J. M. Hellerstein, and S. Shenker. Range Queries over DHTs. Technical Report IRB-TR-03-009, Intel Corporation, 2003.
[12] M. Cai, M. Frank, and P. Szekely. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. *Grid Computing*, 2004.
[13] M. Cai and K. Hwang. Distributed Aggregation Algorithms with Load-Balancing for Scalable Grid Resource Monitoring. In *Proc. of IPDPS*, 2007.
[14] H. Shen and K. Hwang. Locality-Preserving Clustering and Discovery of Wide-Area Grid Resources. In *Proc. of ICDCS*, 2009.
[15] H. Shen. A P2P-based Intelligent Resource Discovery Mechanism in Internet-based Distributed Systems. *JPDC*, 2008.
[16] Planetlab. http://www.planet-lab.org/.
[17] Zhongguancun online. http://www.zol.com.cn/.
[18] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Scalable Wide-Area Resource Discovery. Technical Report TR CSD04-1334, Univ. of California, 2004.
[19] H. Shen, C. Xu, and G. Chen. Cycloid: A Scalable Constant-Degree P2P Overlay Network. *Performance Evaluation*, 63(3):195–216, 2006.
[20] D. C. Erdil and M. J. Lewis. Grid resource scheduling with gossiping protocols. In *Proc. of P2P*, 2007.
[21] Y.-S. Dai, M. Xie, and K.-L. Poh. Availability Modeling and Cost Optimization for the Grid Resource Management System. *IEEE TSMC, Part A: Systems and Humans*, 38(1):170–179, 2008.
[22] Seyong Lee, Xiaojuan Ren, and R. Eigenmann. Efficient content search in ishare, a p2p based internet-sharing system. In *Proc. of IPDPS*, 2008.
[23] A. El-Atawy Y. Yan and E. Al-Shaer. Ranking-based optimal resource allocation in peer-to-peer networks. In *Proc. of ICDCS*, 2007.
[24] Y. Zhang and Y. Fang. A Fine-Grained Reputation System for Reliable Service Selection in Peer-to-Peer Networks. *IEEE TPDS*, 2007.
[25] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *Proc. of P2P*, September 2003.
[26] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proc. of STOC*, pages 654–663, 1997.
[27] H. Shen and C. Xu. Hash-based proximity clustering for efficient load balancing in heterogeneous DHT networks. *JPDC*, 2008.
[28] S. Haykin, editor. *Neural Networks: A Comprehensive Foundataion*. Second Edition, Prentice-Hall Publisher, 1999.
[29] X. Zhang, Y. Qu, and L. Xiao. Improving distributed workload performance by sharing both cpu and memory resources. In *Proc. of ICDCS*, pages 233–241, 2000.