SPPS: A SCALABLE P2P-BASED PROXIMITY-AWARE MULTI-RESOURCE DISCOVERY SCHEME FOR GRIDS

Haiying Shen and Ze Li
Department of Computer Science and Computer Engineering
University of Arkansas, Fayetteville, AR 72701
{hshen, zxl008}@uark.edu

ABSTRACT

Grids are emerging as a novel approach of employing distributed computational and storage resources to solve large-scale problems in science, engineering, and commerce. Distributed Hash Table (DHT) middleware overlay has been applied to grids as a middleware for providing scalable multi-resource discovery. However, direct DHT overlay adoption breaks the physical locality relationship between nodes, making it difficult to discover physically close resources to requesters. Moreover, to achieve multi-resource discovery, some approaches relying on multiple DHTs need high DHT maintenance overhead and other approaches lead to imbalanced load distribution, resulting in low scalability. This paper presents a Scalable P2P-based Proximity-aware multi-resource discovery Scheme (SPPS). It collects the resource information of physically close nodes together, and maps resource requests from requesters to the resource information pool of its physically close nodes. In addition, it relies on a single DHT and achieves balanced resource discovery load distribution, enhancing the system scalability. Simulation results demonstrate the effectiveness of SPPS in proximity-awareness, overhead reduction, and balanced load distribution in comparison with other approaches.

1 INTRODUCTION

The popularity of the Internet as well as the availability of powerful computers and high-speed network technologies have led to what are popularly known as grids. Grid computing is a form of distributed computing whereby a "super and virtual computer" is composed of a cluster of networked, loosely-coupled computers, acting in concert to perform very large tasks. This technology has been applied to computationally-intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back-office data processing in support of e-commerce and web services [1]. Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed for solving large-scale computational and data intensive problems in science, engineering, and commerce. Thus, a resource discovery scheme is needed to help a resource consumer to locate requested resources in resource providers.

Different from other distributed systems, grid environment has its own distinguishing features characterized by proximity and heterogeneity. By proximity, we mean the logical proximity abstraction derived from grids does not necessarily match the physical proximity information in reality. In the environment, heterogeneous computational resources spread across geographically distributed areas world wide. The resources such as storage space and CPU are dynamic, and nodes can enter or leave the system unpredictably. Traditional resource discovery approaches relying on centralized or hierarchical policies [9, 8, 12, 6, 7] cannot tolerate such an environment. In the resource discovery approaches relying on centralized policies, all nodes report their available resources to a central grid node. When a node needs resources, it resorts to the central grid node for the information of resource providers who have its required resources. Since the central grid node needs to store all the information of available resources in the grid system, and needs to process the resource requests from all the nodes in the system, it could easily become a bottleneck and is unable to efficiently process the resource requests, leading to low performance of the grid system. In the resource discovery approaches relying on hierarchical policies, all nodes are formed into a hierarchical structure with a number of levels. A node can ask for the information of

978-1-4244-2677-5/08/$25.00

available resources from the nodes in the above level. The hierarchical structure based approaches avoid the centralized bottleneck problem by distributing resource discovery load among nodes. However, the structure needs to be maintained in the situation of node joins and departures. If the structure is not fixed in time, the efficiency of resource discovery would be adversely affected.

As a successful model that achieves higher scalability and efficiency in distributed systems, Distributed Hash Table (DHT) middleware overlay facilitates the resource discovery in large-scale grid environment [18, 11, 2, 23, 4, 13, 26]. DHT middleware overlay is an important class of the peer-to-peer (P2P) overlay networks, which is a distributed system without any centralized control or hierarchical organization. In the overlay networks, each node has equal functionality. DHT overlay networks map files to the nodes of a network based on a consistent hashing function [10]. To use a DHT middleware overlay for resource discovery in a grid system, all grid nodes are organized into a DHT overlay. The information of an available resource is regarded as a file. The information of available resources are distributed among the nodes by the DHT file allocation policy. A request for a resource is regarded as a request for a file, and is routed to the node which has the information of the required resource by the DHT data location policy, which takes $O(\log n)$ hops. Therefore, a DHT middleware overlay maps the resource providers and consumers in a completely distributed manner with high scalability and efficiency.

However, direct DHT adoption breaks the physical locality relationship of nodes in the underlying IP-level topology. That is, two nodes which are close in the DHT middleware overlay are not necessarily close nodes in the underlying IP-level topology. Since resource sharing and communication among physically close nodes enhance resource discovery efficiency, it is desirable that DHT middleware can map the resource providers and resource consumers that are physically close to each other. Proximity aside, achieving multi-resource discovery remains another challenge. Multi-resource discovery refers to locating resources that are described by a set of resources. For example, a user may need resources described in operation system name, CPU and free memory. Most current DHT-based approaches for multi-resource discovery are not sufficiently scalable and efficient. Multiple-DHT-based approaches rely on multiple DHTs with each DHT responsible for a type of resource [4]. It generates a DHT for each of the resources. Thus, if there are numerous resources, the multiple-DHT-based approaches need to build many DHT middleware overlays. It comes at the cost of high overhead for the maintenance of multiple DHT. In addition, load balance is a critical factor that affects the efficiency of resource discovery approach.

A highly scalable and efficient multi-resource discovery scheme is needed driven by the tremendous advances in grids. To meet the requirements, we propose a Scalable P2P-based Proximity-aware multi-resource discovery Scheme (SPPS), which is built on a single DHT structure. By taking advantage of the hierarchical cluster structure of the DHT, SPPS provides proximity-aware resource discovery by mapping physically close resource requesters and providers. Moreover, SPPS achieves not only multi-resource discovery on a single DHT but also balanced distribution of resource discovery overhead.

The rest of this paper is structured as follows. Section 2 presents a concise review of representative resource discovery approaches for grids. Section 3 introduces SPPS, focusing on its architecture and algorithms. Section 4 shows the performance of SPPS in comparison with other representative approaches in terms of a variety of metrics. Section 5 concludes this paper with remarks on possible future work.

2 RELATED WORK

Over the past years, the immerse popularity of grids has produced a significant stimulus to grid resource discovery approaches. There have been numerous approaches for resource discovery in grids, such as Condor-G [9], Globus toolkit [8], Condor [12], Entropia [6], AppLes [7]. However, relying on centralized or hierarchical policies, these systems have limitation in a large-scale dynamic multi-domain environment with variation of resource availability. Some middlewares such as that in [17] with necessary mechanism are not sufficient by themselves to manage large-scale grid systems with dynamic heterogeneous computer resources. Their broadcasting or flooding strategies are not as efficient as request forwarding in a dynamic environment.

To cope with these problems, more and more grids resort to DHT middleware overlay for resource discovery. DHTs [14, 24, 16, 27, 22] is an important class of the peer-to-peer overlay networks that map keys to the nodes of a network based on a consistent hashing function [10]. Multiple-DHT-based approaches adopt one DHT for each resource, and process multi-resource queries in parallel in corresponding DHTs [4]. Mercury is a scalable resource discovery protocol for routing multi-attribute range-based queries. It can support for multiple attributes and explicit load balancing. Mercury incorporates techniques to support random sampling of nodes within the system. Random sampling enables a number of lightweight approaches to performing load balancing, node count estimation and query selectivity

estimation. In addition to providing high query-routing performance, Mercury provides a range-based query primitive. However, depending on multiple DHTs for multi-resource discovery leads to high structure maintenance overhead. SOMO [26] is a highly scalable, efficient and robust infrastructure for resource management in DHT overlay networks. SOMO performs resource management by relying on a tree structure. It does so by gathering and disseminating system metadata in $O(\log n)$ time with a self-organizing and self-healing data overlay.

MAAN [11] is a Multi-Attribute Addressable Network that extends Chord to support multi-attribute and range queries for grid information services. MAAN addresses range queries by mapping attribute values to the Chord identifier space via uniform locality preserving hashing. It uses an iterative or single attribute dominated query routing algorithm to resolve multi-attribute based queries. To facilitate efficient queries on a range of keys, Andrzejak and Xu proposed a CAN-based approach for scalable, efficient range queries for grid information services [2]. The authors proposed a number of range query strategies and investigated their efficiency. The approach also enhances the routing aspects of current DHT-systems so that frequently changing data can be handled efficiently. SWORD [13] is a scalable resource discovery service for wide-area distributed systems. SWORD has a technique for efficient handling of multi-attribute range queries that describe application resource requirements. It has an integrated mechanism for scalably measuring and querying inter-node attributes without requiring $O(n^2)$ time and space. SWORD also has a mechanism for users to encode a restricted form of utility function indicating how the system should filter candidate nodes when more are available than the user needs, and an optimizer that performs this node selection based on per-node and inter-node characteristics. For scalable resource monitoring and discovery in Grids, Cai and Hwang [5] proposed a scalable Grid monitoring architecture that builds distributed aggregation trees (DAT) on a structured P2P network. By leveraging Chord topology and routing mechanisms, the DAT trees are implicitly constructed from native Chord routing paths without membership maintenance. LORM [21] realizes multi-attribute resource discovery with low overhead based on Cycloid DHT [22]. Most of these work focused on range queries but failed to take proximity feature into account to match physically close resource requesters and providers to achieve high efficiency. SPPS can be complemented by these works to achieve range resource queries, while in turn can complement these works to realize proximity-aware resource discovery.

On the other hand, most DHT-based approaches focused on organizing resource information in DHT structure based on individual resource attribute. Some other approaches focus on weaving all attributes of a resource into one or a certain number of IDs, and then map the resource information to a DHT [18, 19]. Our previous work SEMM provides a preliminary study of exploiting scalable and efficient multi-resource discovery approaches in grids [20]. Relying on Cycloid, SEMM groups physically close nodes into a cluster, and redistributes the resource information within a cluster. When a node needs resource, it queries the nodes in its own cluster. However, nodes are not evenly distributed in a wide area in practice. SEMM will lead to a situation in which nodes are not evenly distributed among clusters. Some clusters may have many nodes, while other clusters may have only a few nodes. Unbalanced node distribution will decrease the efficiency of SEMM. The work presented in this paper is motivated by the lessons learned from SEMM. It distinguishes itself from SEMM by the elimination of the need to build a proximity-aware DHT overlay network, and balanced load distribution. It aims to collect the resource information of physically close nodes without relying on a proximity-aware DHT, and at the same time achieves load balance among the nodes.

3 SCALABLE P2P-BASED PROXIMITY-AWARE MULTI-RESOURCE DISCOVERY

3.1 DHT MIDDLEWARE CONSTRUCTION

SPPS is developed based on Cycloid DHT [22]. We first briefly describe Cycloid DHT middleware overlay followed by a high-level view of SPPS architecture. Cycloid is a lookup efficient constant-degree overlay with $n{=}d \cdot 2^d$ nodes, where d is dimension. It achieves a time complexity of O(d) per lookup request by using O(1) neighbors per node. 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 and $a_{d-1}a_{d-2}.....a_0$ is a cubical index. 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. The node with the largest k in a cluster is called the *primary node* of the nodes at the cluster. All clusters are ordered by their cubical indices mod 2^d on a large cycle. The Cycloid DHT assigns keys onto its ID space by the use of a consistent hashing function. For a given key or a node, its cyclic index is set to the hash value of the key or IP address modulated by d, and the cubical index is set to the hash value divided by d. A key will be assigned to a node whose ID is closest to its ID. Briefly, the cubical index represents the cluster that a node or an object locates, and the cyclic index represents its position in a

cluster. The overlay network provides two main functions: `Insert(key,object)` and `Lookup(key)` to store an object to a node responsible for the key and to retrieve the object. For more information about Cycloid, please refer to [22].

SPPS builds an original Cycloid DHT overlay above a grid system to achieve proximity-aware and multi-resource discovery. Unlike most resource discovery approaches that depend on multiple DHTs for multi-resource discovery, SPPS relies on a single DHT with constant maintenance overhead. In SEMM, physically close nodes are in one cluster. Since nodes are not evenly distributed in a grid system, some clusters may have many number of nodes while other clusters may have only a few nodes. Thus, it will lead to imbalanced distribution of load caused by resource discovery. Unlike SEMM, SPPS does not need to build a proximity-aware DHT middleware overlay. By balanced distribution of nodes in each cluster relying on the original Cycloid DHT, SPPS leads to more balanced load distribution than SEMM. Taking advantage of the hierarchical cluster structure of Cycloid structure and `Insert(key,object)` function, SPPS gathers information of resources in close proximity in a cluster, and further distributes the information to different nodes in the cluster based on resource type. It relies on `Lookup(key)` function for multi-resource discovery. Thus, SPPS provides proximity-aware and multi-resource discovery with a single DHT base and balanced overhead distribution.

3.2 RESOURCE REPORTING AND SEARCHING

Before we present the details of SPPS, let's introduce a landmarking method to represent node closeness on the network by indices. Landmark clustering has been widely adopted to generate proximity information [15, 25]. It is based on the intuition that nodes close to each other are likely to have similar distances to a few selected landmark nodes, although details may vary from system to system. In DHTs, the landmark nodes can be selected by overlay itself or the network. We assume m landmark nodes that are randomly scattered in the Internet. Each node measures its physical distances to the m landmarks, and uses the vector of distances $< d_1, d_2, ..., d_m >$ as its coordinate in Cartesian space. Two physically close nodes will have similar vectors. We use space-filling curves [3], such as Hilbert curve [25], to map m-dimensional landmark vectors to real numbers, such that the closeness relationship among the nodes is preserved. We call this number *Hilbert number* of the node, denoted by $\mathcal{H}$. $\mathcal{H}$ indicates the physical closeness of nodes on the Internet.

Usually, the resources required by applications are described by specifying a set of resources such as available CPU and memory. It has posed a challenge for a resource discovery mechanism to effectively locate resources across widely dispersed domains based on a list of predefined attributes. Moreover, the resource discovery mechanism should discovery resources physically close to the resource requester. In the following, we introduce how SPPS deals with the challenges based on the Cycloid middleware overlay.

We define *resource information*, represented by I_r, as the information of available resources and resource requests. Basically, SPPS groups the information of physically close resources to clusters, and further divides the information into different categories based on resource types, and then assigns different nodes in the cluster responsible for different categories. In DHT overlay networks, the objects with the same key will be stored in the same node. Based on this principle and node ID determination policy, SPPS lets node i compute the consistent hash value of its resource r, denoted by H_r, and use $(H_r, \mathcal{H}_i)$ to represent resource ID, denoted by ID_r. The node uses the DHT function $\texttt{Insert}(ID_r, I_r)$ to store resource information to a node in the system. The I_r repository node is called *directory node*. A directory node periodically conducts resource scheduling between resource providers and requesters. Note that in SEMM, a node reports its I_r to its own cluster, and the I_r of the nodes in the same cluster is redistributed among themselves. Unlike SEMM, in SPPS, a node's I_r is not necessarily reported to its own cluster. Rather, its I_r is routed to a node whose logical ID is the closest to $ID_r = (H_r,\mathcal{H}_i)$. That it, the I_r first reaches a cluster whose logical ID is the closest to the $\mathcal{H}_i$, and then arrives at a node in the cluster whose cyclic index in its ID is the closest to the H_r. Therefore, from the view of the entire system, the resource information with the same ID_r will be pooled in the same node, and the resource information with the same $\mathcal{H}_i$ in its ID will be in the same cluster.

As a result, the information of the same type of resources in physically close nodes will be stored in the same directory node. Nodes in the same cluster are responsible for the resource information of physically nodes, and different nodes in one cluster are responsible for different types of resources. Furthermore, resources of I_r stored in nearby clusters to node i are located physically close to the nodes whose I_r is stored in node i.

When node i queries for different resources, it sends out a request $\texttt{Lookup}(H_r, \mathcal{H}_i)$ for each resource r. Based on the routing algorithm, the request will arrive at a node whose cubical index is the closest to the $\mathcal{H}_i$, and cyclic index is the closest to the H_r. That is, each request will be forwarded

to the directory node which may have the information of resources that conform to the requirements in the request and physically close to the requester. Thus, taking advantage of the Cycloid's functions, SPPS maps the physically close resource consumers and providers.

After receiving the resource request, the directory node first checks its own directory for the resources. If it has no requested information, it probes nodes in other clusters. It was indicated that resources of I_r stored in nearby clusters to node i are located physically close to the nodes whose I_r is stored in node i. In other words, the logically close nodes in the Cycloid DHT overlay store the information of physically close resources. Therefore, probing logically close nodes means looking for resources physically close to the resource requester. We call the directory node which has the closest ID to node j's ID in its succeeding cluster as node j's succeeding directory node, and call the directory node which has the closest ID to node j's ID in its preceding cluster as node j's preceding directory node. In the neighbor probing, a node probes its preceding and succeeding directory nodes, which will further probe their preceding and succeeding directory nodes respectively, and so on until the resource information is found or node j is reached.

Though nodes are not evenly distributed in the wide area of the system in practice, they are distributed in balance in the logical Cylcoid DHT overlay. Recall that physically close nodes will report their resource information to the same cluster. If there are many nodes in a local area and they report their resource information to the same cluster, the nodes in the cluster will be overloaded. On the other hand, if there are only a few nodes in a local area, the nodes in the cluster that these local-area nodes report their resource information to will be lightly loaded. Thus, a challenge is how to balance the load distribution between the lower-loaded cluster and higher-loaded cluster. SPPS takes advantage of the neighbor probing to move the load of higher-loaded nodes to lower-loaded nodes. If a node cannot find the required resources in its directory, it will probe its succeeding directory node and preceding directory node. Thus, if a node transfers its resource information to its succeeding and preceding directory nodes, the information sill can be reached by the subsequent probing algorithm.

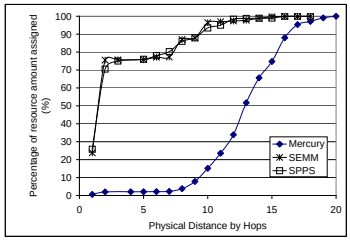
Specifically, a node contacts its succeeding and preceding directory nodes periodically. If the node's load is less than its succeeding directory and preceding directory, it moves part of its resource information to them. Its preceding and succeeding directory nodes conduct the same operation. That is, if they have more load than their preceding directory node and succeeding directory node, they will move their partial resource information to their preceding or succeeding directory nodes respectively. Therefore, if a node cannot find required resources in its own directory, it still can reach its transferred resource information by the neighbor probing. This load balancing algorithm can be regarded as resource information handover from higher-loaded nodes to lower-loaded nodes along the clusters, until all load caused by resource discovery is evenly distributed among nodes. The information transfer can be conducted by piggybacking the routing message. Thus, it will not bring about much extra cost.

4 PERFORMANCE EVALUATION

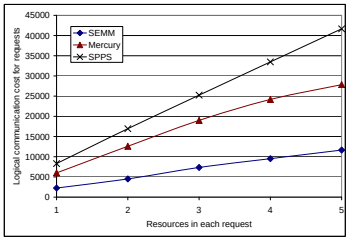
We designed and implemented a simulator for evaluation of SPPS. We compared SPPS with Mercury [4] and SEMM [20]. Mercury uses multiple DHTs and lets each DHT responsible for one resource. SEMM builds a proximity-aware Cycloid overlay on gird for resource discovery. In contrast, SPPS does not need to build a proximity-aware DHT overlay. It still is able to collect the resource information of physically close nodes together. More importantly, it can achieve balanced load distribution through balanced node distribution in each cluster. We assumed that there are 11 types of resources, and used Bounded Pareto distribution function to generate the resource amount owned and requested by a node. In the experiment, we generated 1000 requests, and ranged the number of resources in a resource request from 1 to 5 with step size of 1.

4.1 PROXIMITY-AWARE RESOURCE DISCOVERY

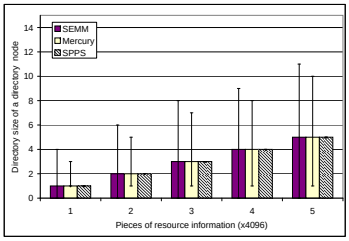
In this experiment, we randomly generated 5000 resource requests, and recorded the distance between the resource provider and requester of each request. Figure 1(a) shows the CDF of the percentage of allocated resources versus the distances in different resource discovery approaches. We can see that SPPS and SEMM exhibit similar performance. They are able to locate 97% of total resource requested within 11 hops, while Mercury locates only about 15% within 10 hops. Almost all allocated resources are located within 15 hops from requesters in SPPS and SEMM, while 19 hops in Mercury. The results show that as SEMM, SPPS can locate most resources within short distances from requesters but Mercury allocate most resource in long distances. The more resources are located in shorter distances, the higher proximity-aware performance of a resource discovery mechanism. The results indicate that the performance of SPPS mechanism is comparable to SEMM, and



(a) CDF of allocated resource



(b) Logical communication cost



(c) Directory size

Figure 1. Efficiency of different resource discovery approaches.

they are better than Mercury in terms of discovering resources physically close to resource requesters.

4.2 OVERHEAD OF RESOURCE DISCOVERY

We define logical communication cost as the product of message size and logical path length in hops of the message travelled. It is assumed that the size of a message is 1 unit. Figure 1(b) plots the logical communication cost versus the types of resources in a request for resource requesting. In the experiment, resource searching stops once requested resources are discovered. We can see that SPPS incurs higher logical communication cost than Mercury, and SEMM generates lower logical communication cost than Mercury. SEMM builds a proximity-aware Cycloid overlay, in which physically close nodes are grouped in a cluster. Without being routed in the entire system, a message takes much less hops. As a result, SEMM generates much less logical communication cost than Mercury and SPPS. In contrast, SPPS builds an original Cycloid overlay, in which physically close nodes are not necessarily in the same cluster, and all nodes are distributed in balance. The side-effect is that a resource reporting message needs to be routed in the system-wide scale. The lookup path length is $O(d)$ in Cycloid, and is $O(\log n)$ in Chord. Since in Cycloid, each node keeps constant 7 neighbors which is less than 11 in Chord, the average path length of Chord is shorter than Cycloid. It is shown in Cycloid [22]. Therefore, SPPS leads to higher path length and higher logical communication cost than Mercury. A request with m resources needs m lookups, which amplifies the difference of lookup cost between Mercury, SEMM and SPPS.

4.3 RESOURCE DISCOVERY LOAD DISTRIBUTION

We ranged the piece number of I_r of available resources reported by a node from 1 to 5 with step size of 1, and measured the average and the 1st and 99th percentiles of directory sizes. We assumed that all nodes have the same capacity for handling resource requests. Figure 1(c) plots the measured results versus the total piece number of resource information. Two observations can be made from the figure. First, the average size of three approaches are the same. This is because all approaches have the same total number of I_r pieces and the same number of nodes in the system. Second, SEMM exhibits larger variance than Mercury and SPPS, and SPPS exhibits the least variance. Mercury uses one DHT for each resource, and classifies resource information based on value/attribute in each DHT, which helps to distribute resource information in balance. On the other hand, by taking advantage of the hierarchical structure of Cycloid, SPPS lets different clusters responsible for resource information in the cluster and allocates information to nodes based on different resource type. Cycloid's more balanced key load distribution helps SPPS achieve balanced information distribution. In addition, SPPS's load balancing algorithm further helps it to achieve balanced information distribution. Though SEMM also relies on Cycloid structure and Cycloid's key distribution algorithm, it leads to unbalanced node distribution by grouping physically close nodes in a cluster. Some nodes may not be assigned resource information, while others may be assigned much more resource information. Therefore, Mercury and SPPS can achieve more balanced distribution of load due to resource information maintenance and resource scheduling operation.

5 CONCLUSIONS

Rapid development of grids requires a scalable and efficient resource discovery approach for its high performance. Most previous multi-resource discovery approaches either depend on multiple DHTs with each DHT responsible for one resource or incur load imbalance, leading to high maintenance and inefficiency. This paper presents a Scalable P2P-based Proximity-aware multi-resource discovery scheme (SPPS), which is built on a hierarchical DHT. By taking advantage of the hierarchical cluster structure, SPPS maps physically resource requesters and providers to achieve proximity-aware resource discovery. Also, SPPS

relies on a single DHT with low overhead, and meanwhile achieves balanced load distribution. Simulation results show the superiority of SPPS in comparison with other approaches. In our future work, we plan to explore methods to reduce the logical communication cost of SPPS.

Acknowledgements

This research was supported in part by U.S. NSF grants CNS-0834592 and CNS-0832109.

References

[1] Grid computing. http://en.wikipedia.org/wiki/Grid_computing.

[2] A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *Proc. the 2nd Int. Conf. on Peer-to-Peer Computing (P2P)*, pages 33–40, 2002.

[3] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmaier. Space filling curves and their use in geometric data structure. *Theoretical Computer Science*, 181(1):3–15, 1997.

[4] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting scalable multi-attribute range queries. In *Proc. of ACM SIGCOMM*, pages 353–366, 2004.

[5] M. Cai and K. Hwang. Distributed aggregation algorithms with load-balancing for scalable grid resource monitoring. In *Proc. of International Parallel and Distributed Processing Symposium (IPDPS)*, 2007.

[6] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: architecture and performance of an enterprise desktop grid system. *Journal of Parallel and Distributed Computing*, 63(5), May 2003.

[7] F. B. et. al. Adaptive computing on the grid using apples. *IEEE Transactions on Parallel and Distributed Systems*, 14(4), Apr. 2003.

[8] I. Foster and C. Kesselman. Globus: a metacomputing infrastructure toolkit. *Int. J. High Performance Computing Applications*, 2:115–128, 1997.

[9] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-g: a computation management agent for multiinstitutional grids. In *Proc. 10th IEEE Symposium on High Performance Distributed Computing*, 2001.

[10] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and P. R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 1997.

[11] J. C. M. Cai, M. Frank and P. Szekely. Maan: A multi-attribute addressable network for grid information services. *Journal of Grid Computing*, 2004. An early version appeared in Proc. of GRID'03.

[12] M. Mutka and M. Livny. Scheduling remote processing capacity in a workstation-processing bank computing system. In *Proc. of the 7th International Conference of Distributed Computing Systems*, September 1987.

[13] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Scalable wide-area resource discovery. Technical Report TR CSD04-1334, EECS Department, Univ. of California, Berkeley, 2004.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 329–350, 2001.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2002.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[17] R. Schantz, J. P. Loyall, C. Rodrigues, D. Schemidt, Y. Krisnamurthy, and I. Pyarali. Flexible and adaptive QoS control for distributed real-time and embedded middleware. In *Proc. of ACM/IFIP/USENIX International Middleware Conference*, 2003.

[18] C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. In *Proc. 12th Int. Symp. on High-Performance Distributed Computing (HPDC)*, pages 226–235, 2003.

[19] H. Shen. Pird: P2p-based intelligent resource discovery in internet-based distributed systems. *ournal of Parallel and Distributed Computing (JPDC)*, 2008.

[20] H. Shen. SEMM: Scalable and Efficient Multi-Resource Management in Grids. In *Proc. of the 2008 International Conference on Grid Computing and Applications (GCA)*, 2008.

[21] H. Shen, A. Apon, and C. Xu. LORM: Supporting Low-Overhead P2P-based Range-Query and Multi-Attribute Resource Management in Grids. In *Proc. of ICPADS*, 2007.

[22] H. Shen, C. Xu, and G. Chen. Cycloid: A scalable constant-degree p2p overlay network. *Performance Evaluation*, 63(3):195–216, 2006. An early version appeared in Proc. of International Parallel and Distributed Processing Symposium (IPDPS), 2004.

[23] D. Spence and T. Harris. Xenosearch: Distributed resource discovery in the XenoServer open platform. In *Proc. the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, pages 216–225, 2003.

[24] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 1(1):17–32, 2003.

[25] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, 2003.

[26] Z. Zhang, S.-M. Shi, and J. Zhu. Somo: Self-organized metadata overlay for resource management in P2P dht. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[27] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: An Infrastructure for Fault-tolerant wide-area location and routing. *IEEE Journal on Selected Areas in Communications*, 12(1):41–53, 2004.