Performance Analysis of DHT Algorithms for Range-Query and Multi-Attribute Resource Discovery in Grids

Haiying Shen
Dept. of Electrical and Computer Engineering
Clemson University
Clemson, SC 29634, USA
shenh@clemson.edu

Cheng-Zhong Xu
Dept. of Electrical & Computer Engineering
Wayne State University
Detroit, MI 48202, USA
czxu@wayne.edu

***Abstract*—Resource discovery is critical to the usability and accessibility of grid computing systems. Distributed Hash Table (DHT) has been applied to grid systems as a distributed mechanism for providing scalable range-query and multi-attribute resource discovery. Multi-DHT-based approaches depend on multiple DHT networks with each network responsible for a single attribute. Single-DHT-based approaches keep the resource information of all attributes in a single node. Both classes of approaches lead to high overhead. Recently, we proposed a heuristic Low-Overhead Range-query Multi-attribute DHT-based resource discovery approach (LORM). It relies on a single hierarchical DHT network and distributes resource information among nodes in balance by taking advantage of the hierarchical structure. We demonstrated its effectiveness and efficiency via simulation. In this paper, we analyze the performance of the LORM approach rigorously by comparing it with other multi-DHT-based and single-DHT-based approaches with respect to their overhead and efficiency. The analytical results are consistent with simulation results. The results prove the superiority of the LORM approach in theory.**

***Keywords*-Resource discovery, Grids, Peer-to-peer system, Distributed hash table**

I. Introduction

Grid systems integrate computers, clusters, storage systems and instruments to provide a highly available infrastructure for large scientific computing centers. They make possible the sharing of existing resources such as CPU time, storage, equipment, data, and software applications. In many grid systems, resources are highly dynamic and vary significantly over time such as available CPU time and memory. Because of this variability, scalable and efficient resource discovery is critical to providing usability and accessibility for large-scale grid computing systems.

The resources required by applications are often described by a set of attributes such as available computing power and memory with range for each attribute (multi-attribute range queries). A fundamental service of resource discovery is to locate resources across multiple administrative domains according to the attribute inputs. Recently, Distributed Hash Table (DHT) architecture has been applied to grid systems for large-scale and dynamic resource discovery due to its high scalability and dynamism-resilience features. DHT can efficiently route messages to the unique owner of any given object. Because of its single deterministic object location, the object can be either resource attribute or value. Therefore, it is a challenge to realize resource discovery with both range-query and multi-attribute features.

Most DHT-based approaches can be classified into multi-DHT-based, single-DHT-based centralized and single-DHT-based decentralized approaches. Multi-DHT-based approaches support multi-attribute range queries by relying on multiple DHT networks with each network responsible for a single attribute [1, 2, 7, 13]. To locate resources specified by several attributes and ranges, each query for a resource is presented to the appropriate DHT and then the results are concatenated in a database-like "join" operation. Single-DHT-based centralized approaches [6] keep the resource information of all values for a specific attribute in a single node. Single-DHT-based decentralized approach [3, 4] is based on one DHT and provides range searching. It maps resource attribute and value of a resource separately to one DHT, and processes a query by searching them separately.

To reduce overhead and enhance efficiency of resource discovery in grids, we recently proposed a DHT-based resource discovery approach with features of Low-Overhead, Range-query and Multi-attribute (LORM) [9]. Unlike the three types of approaches mentioned, LORM is built on a single DHT called Cycloid [10]. LORM arranges each node to be responsible for the information of a specific attribute within a value range by taking advantage of the hierarchical structure of Cycloid. We demonstrated its effectiveness and efficiency via simulation. In this paper, we analyze the performance of the LORM approach rigorously and compare it with multi-DHT-based and single-DHT-based approaches with respect to their overhead and efficiency. The analytical results are consistent with the simulation results. The results prove the superiority of the LORM approach in theory.

The remainder of this paper is structured as follows. Section II describes a review of representative DHT-based resource discovery approaches for grid systems. Section III presents an overview of the DHT-based LORM resource discovery approach and theoretical analysis of its features. Section IV presents the performance analysis of LORM

in comparison to other representative resource discovery methods. Section V presents the performance comparison between LORM and other approaches with regards to the consistency between analysis results and simulation results. Section VI concludes the paper and provides remarks on possible future work.

II. DHT-based Resource Discovery Approaches

As a successful model that achieves scalability, robustness, and deterministic data location, DHT has been widely adopted for resource discovery in grids [1, 2, 7, 13, 6, 3, 4, 9, 11, 8]. Since the resources required by applications are often described in the form of multi-attribute range queries, two important issues investigated recently are range queries and multi-attribute resource discovery. Range queries look for resources specified by a range of attribute values (e.g., a CPU with speed from 1.2GHz to 3.2GHz). Since there is only one unique key for each data in DHTs, either the value or the attribute can be regarded as the key to distribute the resource information to nodes. Therefore, DHTs cannot be simply applied to grids for both multi-attribute query and range query. This problem has posed a challenge to support both range queries and multi-attribute queries.

Current approaches to achieve multi-attribute range-query can be generally classified into three groups: (1) *Multi-DHT-based approach* that adopts one DHT for each attribute, and processes multi-attribute range queries in parallel in corresponding DHTs [1, 2, 7, 13]. In this approach, multiple DHTs for multiple attributes need to be maintained, and the key in each DHT functions is used as the index for resource value for range queries. To locate resources specified by several attributes, the approach uses multi-attribute queries and presents each query for a resource to the appropriate DHT and then concatenates the results in a database-like "join" operation. (2) *Single-DHT-based centralized approach* that pools together resource information of all values for a specific resource attribute in a single node [6]. In this approach, the key in the DHT functions as the index for resource attribute. (3) *Single-DHT-based decentralized approach* that separately maps the resource attribute and value in a resource information to a single DHT, and processes a query by searching them separately [3, 4]. There are other methods [11, 8] that focus on multi-attribute resource discovery without considering range.

Cycloid [10] has a hierarchical structure and is featured by high scalability with constant maintenance overhead and balanced key load distribution. LORM [9] takes advantage of the Cycloid's hierarchical structure to use two indices to represent resource attribute and value for resource information, and to distribute overhead for resource discovery and information maintenance among nodes in balance. Therefore, it only relies a single DHT to realize multi-attribute range-query resource discovery with low overhead.

III. Overview and Properties of LORM

Cycloid [10] is a lookup efficient constant-degree overlay with $n{=}d \cdot 2^d$ nodes, where d is 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 and $a_{d-1}a_{d-2}.....a_0$ is a cubical index. The cyclic index is an integer, ranging from 0 to $d-1$ and the cubical index is a binary number between 0 and 2^d-1. The nodes with the same cubical index are ordered by their cyclic index mod d on a small cycle, which is called *cluster*. All clusters are ordered by their cubical index mod 2^d on a large cycle. Figure 2 shows the partial routing links of a Cycloid. The Cycloid DHT assigns keys onto its ID space by a consistent hashing function [5]. A key will be assigned to a node whose ID is closest to its ID. Cycloid has APIs including `Insert(key,object)`, `Lookup(key)`. LORM relies on a single Cycloid with constant maintenance overhead.

Resource	Hash
CPU	50
Memory	150
Disk	450
External memory	700
Software package	1000
Web service	1200
Bandwidth	1850
Database	2000

Figure 1. Resources hash values.

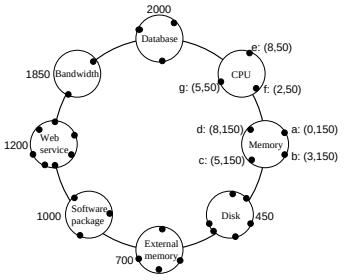


Figure 2. Resource information allocation in LORM.

A computing resource has a specific value for each attribute, for example, "OS=Linux", "CPU=1000MHz", and "Free memory=1024MB". Without loss of generality, we assume that each resource is described by a set of attributes with globally known types denoted by a, and values/ranges or string description denoted by π_a. For example, "Free memory$\geqslant$ 2MB" or "OS=Linux". We define *resource information* as information of available resources and resource queries, denoted by $rescInfo$. Resource information of a resource requester j is represented in a set of 3-tuple representation: $< a, \pi_a, ip\_addr(j) >$. The available resource information of node i is represented in the form of $< a, \delta\pi_a, ip\_addr(i) >$, in which $ip\_addr(i)$ denotes the IP address of node i, and $\delta\pi_a$ is the π_a of its available resource. Usually, the operation in resource discovery is to pool together information of available resources in a number of *directory nodes*, and direct resource requests to those nodes.

A Cycloid consists of a number of clusters, which together constitute a large cycle. LORM lets each cluster be responsible for the information of a attribute, and distributes the information between nodes within the cluster based on resource value/range or string description. Recall that in a Cycloid ID, the cubical indices differentiate clusters, and the

cyclic indices indicate different node positions in a cluster. LORM uses cubical indices to represent a, and uses cyclic indices to represent π_a. Consistent hash function, denoted by H, is used to generate the hash value of attributes. Locality preserving hashing function [3], denoted by $\mathcal{H}$, is used to generate the hash value of attribute value. Thus, the ID of a resource $rescID$=($\mathcal{H}_{\pi_a}, H_a$).

A node reports its available resources to the system periodically via interface `Insert(rescID,rescInfo)`. Therefore, the information of the same attribute will be mapped to the same cluster. We call the node or the ID of the node as the *root* of the $rescID$ or $rescInfo$. Within each cluster, each node is responsible for the information of a resource whose cyclic index falls into the ID space sector it supervises. For example, for the resources and their hashed values listed in Figure 1, the resource information will be stored in nodes as illustrated in Figure 2. In the clusters responsible for memory and CPU, each node is responsible for the resource information in its range. For instance, the resource information with resource ID ($\mathcal{H}_2$,H_{mem})=(2,150) will be routed to and stored in node a.

Proposition 3.1: In LORM, given a range query $[\pi_1, \pi_2]$ for a resource where $\pi_{min} \leqslant \pi_1 \leqslant \pi_2 \leqslant \pi_{max}$, a node that contains attribute value π within $[\pi_1, \pi_2]$ must have an ID that satisfies root($\mathcal{H}_{\pi_2}$,H_a) $\geqslant$ ID $\geqslant$ root($\mathcal{H}_{\pi_1}$,H_a).

Proof: In LORM with $n = d \cdot 2^d$ nodes, a node reports its resource information using the Cycloid interface `Insert((`$\mathcal{H}_{\delta\pi_a}, H_a$`),rescInfo)`. Attribute a with value $\delta\pi_a$ will be stored in root($\mathcal{H}_{\delta\pi_a}$,$H_a$) whose ID is the closest to ($\mathcal{H}_{\delta\pi_a}$,$H_a$). According to the locality preserving hashing, because $\pi_1 \leqslant \pi \leqslant \pi_2$, the resource information of value v will be stored in node i that satisfies to the condition $root(\mathcal{H}(\pi_1)) \leqslant i \leqslant root(\mathcal{H}(\pi_2))$. ■

A node uses `Lookup(rescID)` to query for resources, and the query is routed to the directory node for the desired resource. A multi-attribute query is composed of a set of sub-queries on each attribute, which are processed in parallel. For example, when a node k needs a multiple-attribute resource, say 1.8GHz CPU and 2GB memory, it sends requests
`Lookup(`$\mathcal{H}_{1.8}, H_{cpu}, <cpu, \pi_{cpu}, ip\_addr(k)>$`)` and
`Lookup(`$\mathcal{H}_2, U_{mem}, <mem, \pi_{mem}, ip\_addr(k)>$`)`,
which will be resolved in parallel. The queries will arrive at node a and node e, which reply to the requester node k with the requested resource information $< mem, \delta\pi_{mem}, ip\_addr(i) >$ $where$ $\delta\pi_{mem} = 2$ and $< CPU, \delta\pi_{CPU}, ip\_addr(j) >$ $where$ $\delta\pi_{CPU} = 1.8$. The requester node then concatenates the results in a database-like "join" operation based on ip_addr. The results are the nodes that have desired resource by the requester. For range queries such as "CPU$\geqslant$1.8GHz" and "Free memory$\geqslant$2GB", in addition to responding with satisfied resource information in their own directories, node a and e forward the resource queries to their immediate successors in their own clusters. The successors repeat the same process. This process is repeated until a successor has no satisfied resource information. If the requested resource range is less than a value, then nodes forward queries to their predecessors. If the queries have lower and upper bounds such as "1GHZ$\leqslant$CPU$\leqslant$1.8GHz" and "1GB$\leqslant$memory$\leqslant$2GB", the queries will be forwarded in both directions. Cycloid has a self-organization mechanism to maintain its structure and stored objects, which helps LORM to handle dynamism with node joins and departures.

IV. Comparative Study and Analysis

We use Mercury [2], SWORD [6], MAAN [3] as representatives of multiple-DHT-based, single-DHT-based centralized and single-DHT-based decentralized classes, and analyze LORM in comparison with the approaches. LORM maps resource attribute and value or string description to two levels of a hierarchical Cycloid DHT. Mercury uses multiple DHTs with one DHT responsible for each attribute and maps resource value to each DHT. SWORD maps resource information including both attribute and value in a flat DHT, and MAAN maps attribute and value separately to a flat DHT. To be comparable, we use Chord for attribute hubs in Mercury, and we replace Bamboo DHT with Chord in SWORD.

In Mercury, for higher efficiency of resource query, a node within one of the hubs can hold the data record while the other hubs can hold a pointer to the node. This strategy can also be applied to other methods. To make the different methods be comparable, we don't consider this strategy in the comparative study. We analyze their performance in terms of structure maintenance overhead, resource information maintenance overhead, and the efficiency of resource discovery. In the analysis, we use "attribute value" to represent the locality preserving hash value of both attribute value and attribute string description. We use *directory size* to represent the number of resource information pieces in a directory.

A. Maintenance Overhead

Theorem 4.1: In a grid system with n nodes and m resource attributes, with high probability[1], LORM can improve the structure maintenance overhead of multiple-DHT-based methods (e.g. Mercury) by no less than m times.

Proof: LORM is based on Cycloid, in which each node is responsible for maintaining $d \leq \log(n)$ neighbors. In multiple-DHT-based methods such as Mercury, each node is responsible for maintaining $\log(n)$ neighbors for each DHT of one resource. Therefore, each node has $m\log(n)$ neighbors. The structure maintenance overhead that can be saved is $\frac{m\log(n)}{d} \geq \frac{m\log(n)}{\log(n)} = m$ times. ■

[1]An event happens with high probability (w.h.p.) when it occurs with probability $1 - O(n^{-1})$.

Theorem 4.2: In a grid system, the total number of resource information pieces in single-DHT-based decentralized resource discovery methods (e.g. MAAN) is twice of those in LORM, single-DHT-based centralized methods (e.g. SWORD) and multi-DHT-based methods (e.g. Mercury).

Proof: For each piece of resource information, MAAN splits its a and π_a, and stores the two pieces of information separately, while LORM, single-DHT-based centralized (e.g. SWORD) and multi-DHT-based methods (e.g. Mercury) only store one information piece. Therefore, the size of the total resource information of MAAN is twice of others. ■

Theorem 4.3: In a grid system with n nodes and m resource attributes, with the assumption that each type of resource attribute has k pieces of resource information and its values are uniformly distributed, w.h.p., LORM can reduce the number of resource information pieces in a directory node in the single-DHT-based decentralized resource discovery methods (e.g. MAAN) by $d(1 + \frac{m}{n})$ times.

Proof: For k pieces of resource information of a resource attribute, MAAN splits the attribute and value. k pieces are stored in the same node, and the other k pieces are uniformly distributed among the n nodes based on the value. A directory node has a total of $k + m \cdot \frac{k}{n}$ pieces. LORM does not split the information, and all resource information of a particular resource attribute is in a cluster with d nodes. With the uniform distribution assumption, each node is responsible for k/d pieces of resource information. Therefore, LORM can reduce the total size of resource information in a directory node in MAAN by $\frac{k+m\cdot\frac{k}{n}}{k/d} = d(1 + \frac{m}{n})$ times. ■

Theorem 4.4: In a grid system with n nodes and m resource attributes, with the assumption that each type of resource attribute has k pieces of resource information and its values are uniformly distributed, w.h.p., LORM can reduce the resource information size in a directory node in single-DHT-based centralized methods (e.g. SWORD) by d times.

Proof: In LORM, all resource information of a particular resource attribute is in a cluster with d nodes. With the uniform distribution assumption, each node has k/d pieces of resource information. In SWORD, all resource information of a particular resource attribute is in a single node. Thus, LORM can reduce the resource information size in a directory node in SWORD by $\frac{k}{k/d} = d$ times. ■

Theorem 4.5: For any set of n nodes and m resource attributes, with the assumption that each type of resource attribute has k pieces of resource information and its values are uniformly distributed, w.h.p., multi-DHT-based methods (e.g. Mercury) can achieve more balanced resource information distribution than LORM by $\frac{n}{dm}$ times.

Proof: In LORM, all resource information of a particular resource attribute is in a cluster with d nodes. Since the resource values are uniformly distributed, each node is responsible for k/d pieces of resource information. In Mercury, for one attribute, a node is responsible for $\frac{k}{n}$ pieces of resource information. Given m resource attributes, each node is responsible for $\frac{mk}{n}$ pieces of resource information. Thus, Mercury can achieve more balanced resource information distribution than LORM by $\frac{k/d}{\frac{mk}{n}} = \frac{n}{dm}$ times. ■

Theorem 4.6: Multi-DHT-based methods (e.g. Mercury) and LORM achieve more balanced resource information distribution than the single-DHT-based decentralized resource discovery methods (e.g. MAAN) and single-DHT-based centralized methods (e.g. SWORD).

Proof: Theorems 4.3 and 4.4 show that LORM achieves more balanced resource information distribution than the single-DHT-based decentralized (e.g. MAAN) and centralized methods (e.g. SWORD). Theorem 4.5 shows that multi-DHT-based methods (e.g. Mercury) achieves more balanced distribution than LORM. Therefore, multi-DHT-based methods and LORM achieve more balanced resource information distribution than the single-DHT-based decentralized and centralized methods. ■

B. Efficiency of Resource Discovery

Theorem 4.7: To discover resources for an m-attribute non-range resource query in an n-node network, w.h.p., LORM can reduce the total number of contacted nodes of single-DHT-based decentralized resource discovery methods (e.g. MAAN) by $\frac{\log n}{d}$ times.

Proof: For each non-range resource query, LORM needs one DHT lookup, while MAAN needs two DHT lookups for each attribute: attribute name and value. For an m-attribute resource query, LORM needs m DHT lookups, and MAAN needs $2m$ DHT lookups. Hence, for one resource query, the number of lookups in MAAN is $\frac{2m}{m}$=2 times of LORM. On the average case, one lookup needs $\log n/2$ hops in Chord [12] and d hops in Cycloid [10]. Thus, LORM can reduce the total number of contacted nodes of MAAN by $\frac{2\log n/2}{d} = \frac{\log n}{d}$ times. ■

Theorem 4.8: To discover resources for an m-attribute non-range resource query in an n-node network, w.h.p., multi-DHT-based methods (e.g. Mercury) and single-DHT-based centralized methods (e.g. SWORD) can reduce the total number of contacted nodes of single-DHT-based decentralized resource discovery methods (e.g. MAAN) by twice.

Proof: For each non-range resource query, the multi-DHT-based methods (e.g. Mercury) and single-DHT-based centralized methods (e.g. SWORD) need one DHT lookup, while MAAN needs two DHT lookups for each attribute. For an m-attribute resource query, the former methods need m DHT lookups, and MAAN needs $2m$ DHT lookups. Hence, the former methods can reduce the number of contacted nodes in MAAN by twice. ■

When a range resource query is routed to its root, the root node checks its directory for the range query. Then, in SWORD, the resource searching stops; in Mercury and MAAN, the node forwards the query to its successor or

predecessor according to their closeness to the queried range; in LORM, the node forwards the query to its successor or predecessor in its cluster according to their closeness to the queried range. The nodes receiving the query will repeat the process. We call the nodes that receive a resource query and check their directories for the queried resource as *visited nodes* of the resource query.

Theorem 4.9: In an n-node network, w.h.p., LORM can reduce at least $\frac{m(n-d)}{4}$ visited nodes to discover required resource for an m-attribute range resource query in system-wide range resource discovery methods (e.g. MAAN and Mercury), and SWORD can reduce $\frac{md}{4}$ visited nodes in LORM, in the average case.

Proof: In Mercury, for a range query, the number of contacted nodes needed is $\frac{n}{2}$ in the worst case. Thus, on the average case, the total number of visited nodes for an m-attribute resource range query on the average case is $m(1+\frac{n}{4})$. MAAN has two lookups for each attribute query. m-attribute resource query needs $m(2+\frac{n}{4})$ hops. In LORM, the nodes needed to be visited for a resource query is $m(1+\frac{d}{4})$ on the average case. Therefore, LORM can reduce at least $m(1+\frac{n}{4})-m(1+\frac{d}{4})=\frac{m(n-d)}{4}$ visited nodes for an m-attribute resource query in system-wide range resource discovery methods such as MAAN and Mercury. Because SWORD doesn't need to forward query for range query. It reduces $m(1+\frac{d}{4})-m=\frac{md}{4}$ contacted nodes in LORM. ■

Theorem 4.10: In an n-node network, w.h.p., LORM can reduce at least mn contacted nodes to discover required resource for an m-attribute resource range query in system-wide range resource discovery methods (e.g. MAAN and Mercury), in the worst case.

Proof: Mercury uses one DHT for each attribute. For each attribute in a resource requester, it needs $\log n$ hops for the request to reach its root node [12]. After that, n nodes need to be probed for the range query, in the worst case, because all resource information of the attribute spreads over the n nodes. Therefore, for each attribute query, $\log n + n$ hops are needed in the worst case. For a resource query with m attributes, Mercury needs to contact $m(\log n + n)$ nodes. MAAN has two lookups for each attribute query since it lookups attribute name and value or string description separately. Only one lookup needs system-wide probing on n nodes. Consequently, each query of an attribute needs $(2\log n + n)$ hops, and m-attribute resource query needs $m(2\log n+n) > m(\log n+n)$ hops in MAAN. The nodes need to contact in LORM for a resource query is $m \cdot d \leq m \cdot \log n$ in the worst case. Therefore, LORM can reduce at least $m(\log n + n) - m \cdot \log n = mn$ contacted nodes for an m-attribute resource query in system-wide range resource discovery methods, in the worse case. ■

V. Performance Comparison

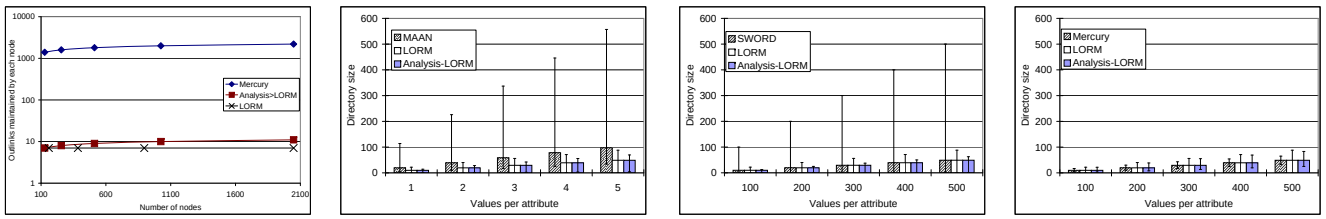
This section presents the performance evaluation of LORM in average case in comparison with Mercury [2], SWORD [6], MAAN [3]. To be comparable, we used Chord for Mercury and SWORD. The dimension was set to 8 in Cycloid and 11 in Chord, and each DHT had 2048 nodes. We assumed there were $m = 200$ resource attributes, and each attribute had $k = 500$ values. We used Bounded Pareto distribution function to generate resource values owned by a node and requested by a node. The resource attributes in a node resource request were randomly generated.

A. Maintenance Overhead

In DHT overlays, each node needs to maintain a number of neighbors (outlinks) which constitute a large part of the DHT overlay maintenance overhead. Theorem 4.1 shows that LORM can reduce the DHT maintenance overhead of Mercury by no less than m times. We use "Analysis>LORM" to represent the experiment results of Mercury divided by $m = 200$. Figure 3(a) plots the number of outlinks maintained by each node in Mercury, "Analysis>LORM" and LORM versus network size. From the figure, we can see that the number of outlinks per node in LORM is less than that of "Analysis>LORM". The experiment results are consistent with Theorem 4.1. Recall that Mercury has multiple DHTs with each DHT responsible for one resource attribute, such that each node has a total number of outlinks equals to the product of routing table size and the number of DHTs. As a result, each node in Mercury maintains dramatically more outlinks than that of LORM.

In addition to the outlinks, a directory node needs to maintain resource information. It is desirable to distribute the information among nodes uniformly so that the information maintenance overhead as well as resource discovery load can be distributed among nodes to avoid bottlenecks. The average directory size are the total number of resource information pieces divided by the total number of nodes.

Figure 3(b) plots the experiment results of the average and the 1st and 99th percentiles of directory size per node in MAAN and LORM. It also plots the analysis results of LORM based on MAAN according to Theorems 4.2 and 4.3. The analysis results of the 1st and 99th percentiles are the experiment results of MAAN divided by the factor of $d(1+\frac{m}{n}) = 8 \times (1+\frac{200}{2048}) = 8.78$, and the analysis results of the average directory size are the experiment results of MAAN divided by 2. We can see that the experiment results of the average directory size of LORM match the analysis results. Recall that MAAN separates resource attribute and value or string description of a piece of resource information, and stores the information separately. Therefore, MAAN doubles the total resource information size and needs information maintenance overhead twice as high as others. The experiment results of the 1st and 99th percentiles are close to the analysis results. The experiment results of the 99th percentile of LORM are slightly higher than the analysis results. This is because that the resource values are randomly



(a) Average number of outlinks (Theorem 4.1) (b) MAAN versus LORM (Theorem 4.3) (c) SWORD versus LORM (Theorem 4.4) (d) Mercury versus LORM (Theorem 4.5)

Figure 3. Overhead in different resource discovery approaches.

chosen in the experiment and are not completely uniformly distributed. In addition, all resource information of the same attribute is collected in one cluster. It is very likely that a node has much more resource information than other nodes in the same cluster.

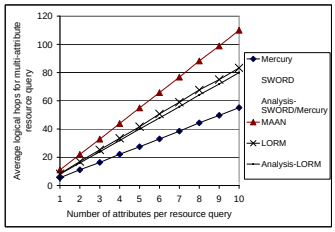
Theorem 4.4 proved that LORM can reduce the directory size of SWORD by d times. Figure 3(c) plots the experiment results of the average and the 1st and 99th percentiles of directory size per node in SWORD and LORM. It also plots the analysis results of LORM based on the experiment results of SWORD. The analysis results of the 1st and 99th percentiles are the experiment results of SWORD divided by the factor of d, and the analysis results of the average directory size equal to the experiment results of SWORD according to Theorem 4.2. We can see that the experiment results of the average directory size of LORM match the analysis results in Theorem 4.2. The experiment results of the 99th percentile are only slightly higher than the analysis results in Theorem 4.4. This is due to the reason that the attribute values are randomly distributed among d nodes, and some nodes have more resource information than others with random distribution.

Theorem 4.5 proved that Mercury can achieve more balanced resource information distribution than LORM by $\frac{n}{dm}$ times. Figure 3(d) plots the experiment results of the average and the 1st and 99th percentiles of directory size per node in Mercury and LORM. It also plots the analysis results of LORM based on Mercury. The analysis results of the 1st and 99th percentiles are the experiment results of Mercury multiplied and divided by the factor of $\frac{n}{dm} = \frac{2048}{8\times 200} = 1.28$ respectively, and the analysis results of average directory size are the experiment results of Mercury according to Theorem 4.2. We can see that the experiment results of the average directory size of LORM match the analysis results in Theorem 4.2. The experiment results of the 99th percentile are also only slightly higher than the analysis results due to the same reason observed in Figure 3(b) and (c). In addition, the results of the 1st percentile are lower than the analysis results in Theorem 4.4. This is because when attribute values are randomly selected, some values may not be chosen and hence some nodes in a cluster in LORM may not be assigned resource information.

In general, the experiment results of LORM mach the analysis results. From Figures 3(b), (c) and (d), we can observe that MAAN and SWORD exhibit significantly larger variance of directory size than Mercury and LORM. MAAN and SWORD distribute resource information to directory nodes based on resource attribute. As there are 200 resource attributes, the information is accumulated in 200 nodes among 2048 nodes, leading to large variance of directory size. On the other hand, Mercury uses one DHT for each attribute, and classifies resource information based on value in each DHT. The widespread information distribution helps to distribute resource information uniformly. LORM arranges different Cycloid clusters to be responsible for resource information based on resource attribute and allocates information to a node based on its range, leading to more balanced information distribution. Therefore, Mercury and LORM can achieve more balanced distribution of load due to resource information maintenance and resource discovery operation. This result is in agreement with Theorem 4.6.

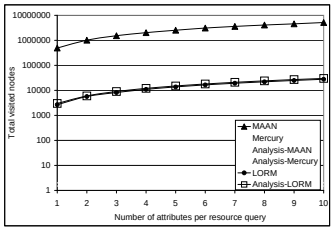
B. Efficiency of Resource Discovery

For a non-range query, Theorem 4.7 shows that LORM can reduce the total number of contacted nodes of MAAN by $\frac{\log n}{d} = \frac{11}{8}$ times; Theorem 4.8 shows that Mercury and SWORD can reduce the total number of contacted nodes of MAAN by twice. We varied the number of attributes in a query from 1 to 10 with step size of 1. The logical hop metric is measured by the number of hops traversed by a query in a resource search. We randomly chose 100 nodes and let each node send 10 resource queries. Figures 4(a) and (b) show the experiment results of the average and total logical hops for multi-attribute resource queries versus the number of attributes in a resource query in different approaches. The figures also plot the analysis results of LORM and SWORD/Mercury based on the experiment results of MAAN denoted by "Analysis-LORM" and "Analysis-SWORD/Mercury", which are the experiment results of MAAN divided by $\frac{11}{8}$ and 2 respectively. Because the difference between Mercury, SWORD and "Analysis-SWORD/Mercury" is no more than 0.3 in Figure 4(a), and their difference is no more than 800 in Figure 4(b), these curves are completely overlapped. In order to make the figure clear, we only draw the curve of Mercury and use it to represent SWORD and "Analysis-SWORD/Mercury". From

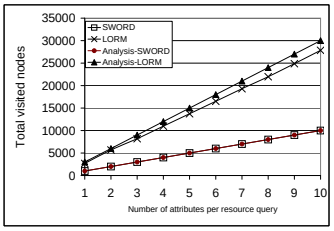


(a) Average number of contacted nodes (Theorems 4.7 and 4.8)

Figure 4. The number of hops for query routing in different resource discovery approaches.



(a) Visited nodes of MAAN/Mercury and LORM (Theorem 4.9)



(b) Visited nodes of SWORD and LORM (Theorem 4.9)

Figure 5. Searching cost in different resource discovery approaches.

the figure, we can see that the experiment results of LORM is very close to the analysis results, and the experiment results of SWORD/Mercury exactly match the analysis results in Theorem 4.7 and 4.8.

Comparing the different methods, we can see that MAAN generates the highest number of contacted hops and SWORD/Mercury produce the least number of contacted hops. This is because MAAN needs two lookups for resource attribute and value, and others only need one lookup. The reason that LORM has higher number of hops than Mercury and SWORD is due to their time complexity of lookups. Chord has a time complexity of $O(\log n)$ per query, and Cycloid has a time complexity of $O(d)$ per query. These results are consistent with the file lookup path length in [10]. The figures also show that the average and total number of logical hops increases as the number of attributes in each resource request grows. This is because a node needs to send out multiple queries for multiple attributes.

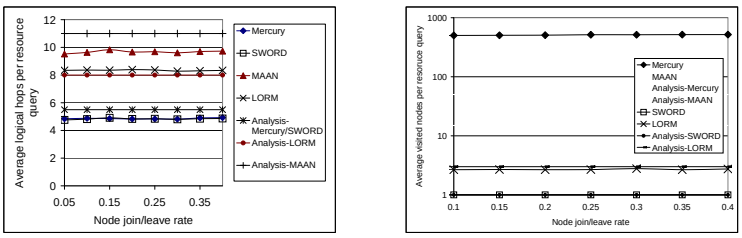
For a range resource query, a root node needs to probe other nodes after receiving a query. We use the number of visited nodes to represent resource searching efficiency. The proof in Theorem 4.9 shows that to query an m-attribute resource with range requirement in an n-node network, the total number of visited nodes is $m(1+\frac{n}{4})$ in Mercury, $m(2+\frac{n}{4})$ in MAAN, $m(1+\frac{d}{4})$ in LORM and m in SWORD. Based on the analysis, we calculated the number of visited nodes for one query. It is $m(1+\frac{n}{4}) = 513m$ in Mercury, $m(2+\frac{n}{4}) = 514m$ in MAAN, $m(1+\frac{d}{4}) = 3m$ in LORM, and m in SWORD. The total number of visited nodes for 1000 queries is the product of the result and 1000. Figure 5(a) and Figure 5(b) plot the experiment results and analysis results of the number of visited nodes versus the number of attributes per query. In Figure 5(a), the results of MAAN, Mercury, "Analysis-MAAN" and "Analysis-Mercury" are completely overlapped because their values differ no more than 70000. Therefore, in order to make the figure clear, we only show the results of MAAN to represent itself and the others. In the figure, the y axis is shown in logarithmic scale and MAAN and Mercury have a very large number of visited nodes. From Figure 5(b), we can see that the experiment results of LORM are a little lower than its analysis results.

The experiment results of other methods are consistent with their analysis results. Mercury and MAAN visit tremendously more nodes than SWORD and LORM. Recall that Mercury and MAAN accumulate resource information based on attribute value, which spreads along the entire DHT ID space. They need to probe nodes along a very large ID space. In contrast, SWORD accumulates resource information based on node attribute. All information of a particular attribute is in one directory node, and no node needs to be probed. Therefore, a resource query for an m-attribute resource needs m visited nodes, and its experiment results are exactly matches the analysis results. LORM stores resource information of a specific attribute name in a cluster, and only the nodes in the cluster should be probed. This limits the node probing scope to a cluster rather than the entire system. As a result, SWORD and LORM incur much less cost for a range query than Mercury and MAAN.

C. Performance in a Highly Dynamic Environment

This section evaluates the efficiency of the LORM in a highly dynamic environment in comparison with other approaches. In this experiment, the resource join/departure rate R was modelled as a Poisson process as in [12]. For example, there is one resource join and one resource departure every 2.5 seconds with R=0.4. We varied R from 0.1 to 0.5, with step size of 0.1. We set the number of total resource requests to 10000. Experiment results show that there were no failures in all test cases. Figure 6(a) shows the experiment and analysis results of the average number of logical hops for a non-range resource query as R changes. The analysis results are drawn from the proof of Theorem 4.7 and 4.8. Compared with the logical hop evaluation in Figure 4, we can see that the measured number of hops in a dynamic environment is very close to that in a static environment and does not change with the rate R. We can also observe that the analysis results are slightly higher than the experiment results. This is because that the lookup path is reduced sometimes due to the node departures. The results are also consistent with Theorem 4.7 and 4.8 in terms of the number of contacted hops for a resource query.

Figure 6(b) shows the analysis results and experiment results of the average visited nodes per range resource query. The results of Mercury, MAAN, "Analysis-Mercury"



(a) Average number of contacted nodes (Theorem 4.7 and 4.8)

(b) The number of visited nodes (Theorem 4.9)

Figure 6. Efficiency of different resource discovery approaches in churn.

and "Analysis-Mercury" are overlapped since their results differ no more than 30. Thus, we only draw the result of Mercury. The analysis results are drawn from the proof of Theorem 4.9. First, we can see that the experiment results are consistent with the analysis results. Because Mercury and MAAN are system-wide range resource discovery methods that distribute resource information in all nodes based on attribute values, they incur much more node communication for a resource query due to system-wide probing. Second, the results are consistent with the results in Figure 5 in static situation. Thus, dynamism generates little adverse effect on the efficiency of resource querying. In conclusion, the experiment results confirm that LORM can effectively resolve resource queries in a dynamic environment.

VI. Conclusions

Resource discovery is a critical issue for grid systems in which applications are composed of hardware and software resources. Previous resource discovery approaches either depend on multiple DHTs with each DHT responsible for a resource or rely on one DHT by pooling resource information of an attribute in a single node, leading to high maintenance overhead or inefficiency due to load imbalance. Recently, we proposed a Low-Overhead Range-query Multi-resource discovery approach (LORM). LORM relies on a single DHT with constant maintenance overhead to achieve range-query multi-attribute resource discovery with low overhead. In this paper, we analytically study the performance of LORM in comparison with the previous resource discovery methods with regards to their structure maintenance overhead, resource information maintenance overhead and resource searching efficiency. We also show the consistency of the analysis results with the experiment results. Analytical results show the superiority of LORM in comparison with other representative approaches in terms of overhead cost and efficiency of range-query and multi-attribute resource discovery. We plan to further explore and elaborate upon the LORM design to discover resources based on semantic information.

Acknowledgment

The authors are grateful to the anonymous reviewers for their valuable comments and suggestions. This research was supported in part by U.S. NSF grants CNS-0834592 and CNS-0832109.

References

[1] A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *Proc. of P2P*, 2002.

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

[3] M. Cai, M. Frank, and et al. MAAN: A multi-attribute addressable network for grid information services. *Journal of Grid Computing*, 2004.

[4] M. Cai and K. Hwang. Distributed aggregation algorithms with load-balancing for scalable grid resource monitoring. In *Proc. of IPDPS*, 2007.

[5] D. Karger and et al. 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.

[6] D. Oppenheimer, J. Albrecht, and et al. Scalable wide-area resource discovery. Technical Report TR CSD04-1334, EECS Department, Univ. of California, Berkeley, 2004.

[7] S. Ratnasamy, J. M. Hellerstein, and S. Shenker. Range queries over DHTs. Technical Report IRB-TR-03-009, Intel Corporation, 2003.

[8] H. Shen. A P2P-based Intelligent Resource Discovery Mechanism in Internet-based Distributed Systems. *JPDC*, 2008.

[9] 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.

[10] H. Shen, C. Xu, and G. Chen. Cycloid: A scalable constant-degree P2P overlay network. *Performance Evaluation*, 63(3):195–216, 2006.

[11] D. Spence and T. Harris. Xenosearch: Distributed resource discovery in the XenoServer open platform. In *Proc. of HPDC*, pages 216–225, 2003.

[12] I. Stoica, R. Morris, and et al. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM TON*, 11(1):17–32, 2003.

[13] 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, Institute on System Architecture, CoreGRID - Network of Excellence, 2006.