Locality-Preserving Clustering and Discovery of Wide-Area Grid Resources

Haiying Shen
Dept. of Computer Science and Computer Engineering
University of Arkansas, Fayetteville, AR 72701
hshen@uark.edu

Kai Hwang
Dept. of Electrical Engineering and Computer Science
University of Southern California, Los Angeles, CA 90089
kaihwang@usc.edu

Abstract

In large-scale computational or P2P Grids, discovery of heterogeneous resources as a working group is crucial to achieving scalable performance. This paper presents a hierarchical cycloid overlay (HCO) architecture with resource clustering and discovery algorithms for efficient and robust resource discovery in wide-area distributed Grid systems. We establish program/data locality by clustering resources based on their physical proximity and functional matching with user applications. We further develop randomized probing and cluster-token forwarding algorithms. The novelty of the HCO scheme lies in low overhead, fast speed and dynamism resilience in multi-resource discovery. The paper presents the HCO framework, new performance metrics, and simulation experimental results. This HCO scheme compares favorably with other resource management methods in static and dynamic Grid applications. In particular, it supports efficient resource clustering, reduces communications cost, and enhances resource discovery success rate in promoting large-scale distributed supercomputing applications.

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 Grid computing. Grid computing leverages a high-degree of resource sharing in a distributed network environment. It enables the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices. Grid computing benefits a variety of applications such as collaborative engineering, data exploration, high-throughput computing, and distributed supercomputing.

Overlay networks based on distributed hash table (DHT) have been suggested to manage large-scale Grid resources [16]. DHT overlay networks [18], [10], [23], [15] map files to the nodes in a network based on a consistent hashing function [7]. Most of the DHT overlays require $O(\log n)$ hops per lookup request with $O(\log n)$ neighbors per node, where n is the network size. A computing resource is always described by a *resource type (i.e. functionality)* such as CPU and memory, and *resource attribute* indicating the quantity and special application requirement. To use a DHT overlay for resource discovery in a Grid system, all Grid nodes are organized into a DHT overlay. The descriptors of available resources are regarded as files and are distributed among the nodes. Resource queries are regarded as file lookups and are routed to the nodes having the descriptors of the required resources. Therefore, DHT overlays map the resource providers and consumers in Grids in a distributed manner with high scalability.

In a wide-area Grid system, resource sharing and communication among physically close nodes enhance application efficiency. In addition, we need to solve the problem of increasing complexity in using heterogeneous resources in a Grid system. Different resources, such as CPU and memory, are always jointly requested and used. Resource clustering based on functional matching with the demands of user application facilitates a user's resource discovery. We use *program/data locality* to represent the phenomenon in which resources are proactively clustered so that a node can always locate physically close resources satisfying required functionalities in its neighborhood on the overlay. It is desirable to develop a resource management scheme that is able to preserve the program/data locality.

However, the adoption of DHT overlays in most current resource management schemes [3], [4], [8], [11], [1], [17], [22], [12] cannot preserve program/data locality. First, direct DHT construction on a Grid system breaks the physical proximity relationship of nodes in the underlying IP-level topology. That is, two nodes which are close in the DHT overlay are not necessarily close nodes in the underlying IP-level topology. Second, a node may not locate its required multiple resources in its neighborhood on the overlay. If a node needs m resources, at least m lookup messages are needed, each of which traverses $O(\log n)$ hops in the system-wide scope.

Moreover, most DHT-based solutions of multi-resource management have limited scalability and fault tolerance. Some solutions apply multiple DHT overlays to manage specific resource groups [3]. This may result in significant increase in management overhead and low dynamism-resilience since node joins and departures lead to update of multiple DHT overlays. Others apply a single DHT over with one node responsible for one type of resources [4], [8], [1], [5], [19]. They store resource descriptors in a few nodes, resulting in imbalanced distribution of resource management workload and loss of many descriptors in dynamism.

We desire to have a Grid resource management scheme that is program/data locality-preserving (locality-preserving in short) and highly scalable and dynamism-resilient. To meet this demand, this paper extends from previous work [15], [14], [12] on locality-preserving and dynamism-resilient Grid resource clustering and discovery. We present a new hierarchical cycloid overlay (HCO) architecture with resource clustering and discovery algorithms for efficient and scalable resource discovery. We establish program/data locality by clustering resources based on their physical proximity and functional matching with user applications. We further develop randomized probing and cluster-token forwarding algorithms. The novelty of the HCO scheme lies in its low-overhead, fast and dynamism-resilient multi-resource discovery. Within the authors' knowledge, the HCO scheme is the first work that preserves the program/data locality in Grid resource management for high scalability and efficiency.

The rest of this paper is structured as follows. Section 2 presents a concise review of representative resource management approaches for Grids. Section 3 specifies the HCO architecture and applications. Section 4 presents the locality-preserving properties of HCO, the resource discovery and clustering algorithms, the algorithm to deal with dynamism, and the randomized probing and cluster-token forwarding algorithms. Section 5 reports the simulation experimental results in both static and dynamic network environments. The final section concludes with a summary of contributions and discussions on further research work needed.

2. Related Work

DHT overlay networks [18], [10], [23], [15] have been suggested to manage large-scale Grid resources. Mercury [3] is a resource discovery protocol for routing multi-resource range-based queries. It can also support explicit load balancing. To support multi-resource queries, Mercury uses multiple DHT overlays. It uses one DHT for each resource, and processes multi-resource queries in parallel in corresponding DHT overlays. However, depending on multiple DHT overlays leads to high overhead for DHT maintenance. SOMO [22] is a 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.

One group of approaches [4], [8], [1], [5], [19] organize all Grid resources into one DHT overlay, and assign all descriptors of one resource to one node. MAAN [4] is a Multi-Attribute Addressable Network that extends Chord to support multi-resource 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-resource based queries. To facilitate efficient queries on a range, Andrzejak and Xu proposed a CAN-based approach for grid information services [1]. They proposed a number of range query strategies and investigated their efficiency. The approach also enhances the routing aspects of current DHT-systems. SWORD [8] is a resource discovery service for wide-area distributed systems. It locates a set of machines matching user-specified constraints on both static and dynamic node characteristics. It has a technique for efficient handling of multi-resource range queries that describe application resource requirements. For scalable resource monitoring and discovery in Grids, Cai and Hwang [5] proposed a 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. Most of the single DHT-based approaches assign one node to be responsible for all descriptors of one resource, leading to imbalanced distribution of workload. If one of the nodes fails in dynamism, many descriptors will be lost at a time.

Most DHT-based approaches focus on organizing resource descriptors in DHT overlay based on individual resource type. Some approaches focus on weaving all attributes of a resource into one or a certain number of IDs for resource clustering in a DHT overlay [11], [12]. Schmidt and Parashar [11] proposed to use Hibert space filling curve and Shen [12] proposed to use a locality sensitive hashing function to map a multi-dimensional information space into a one dimensional DHT identifier space.

However, few of the current approaches have devoted to realizing the program/data locality to facilitate low-overhead and quick resource discovery. The HCO scheme proposed in this paper is novel in that it establishes program/data locality which enables users to discover their required and physically close resources from their nearby nodes on the DHT overlay. Unlike most existing approaches, HCO is based on a single DHT for multi-resource management. It distributes workload of resource descriptor maintenance and resource query processing in balance. Our previous work SEMM [13] provides a preliminary study of exploiting a hierarchical cycloid overlay for resource management. HCO distinguishes itself by addressing detailed issues in resource discovery and proposing strategies for locality-preserving, dynamism-resilience and efficient resource discovery. These features contribute to the high scalability and efficiency characteristics of HCO in Grid resource management. Comprehensive simulation results confirm the high performance of HCO.

3. Hierarchical Cycloid Overlay Network

We present below the architecture and processing layers of HCO. This is a DHT-based hierarchy for locality-preserving Grid resource clustering and discovery. HCO is built by extending from the cycloid overlay proposed in [15]. Cycloid is a lookup efficient overlay network generalized form the *cube-connected cycles* (CCC) [9]. A d-dimensional cycloid is built with at most $n{=}d \cdot 2^d$ nodes. Like CCC, a cycloid has a constant node degree equals to its dimension d.

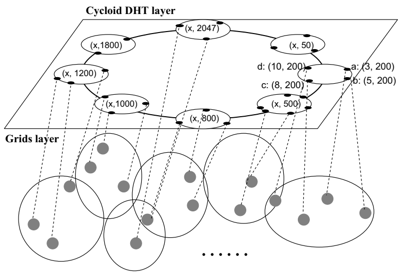


Figure 1. *A HCO network built on top of Grid resource clusters. A 3-dimensional cycloid overlay is shown to manage at most 8 clusters (only 6 are shown). Nodes at different cycles in cycloid manage different resource clusters in order to preserve program/data locality in distributed Grid computing (cluster mapping shown by dashed lines).*

The upper part of Figure 1 shows a 3-dimensional cycloid. In general, it takes at most $O(d)$ steps to lookup a file. 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 $\in [0, d-1]$, and the cubical index is a binary number $\in [0, 2^d - 1]$. The nodes with the same cubical indices are ordered by their $k \ mod \ d$ on a small cycle called a *cluster*. The node with the largest cyclic index in a cluster is called the *primary node* of the cluster. All clusters are ordered by their cubical indices $mod \ 2^d$ on a large cycle. The cubical index represents the cluster that a node or an object locates, and the cyclic index represents its position within a cluster.

An object is assigned to a node whose ID is closest to its ID. The overlay network provides two main functions: `Insert(ID,object)` stores an object to a node responsible for the ID, `Lookup(ID)` retrieves the object through DHT-based searching. Each node maintains a routing table recording its neighbors in the overlay network for object lookups. Most properties of cycloid can be found in [15].

In Figure 1, the resource nodes are shown at the bottom Grid layer. Various nodes are grouped into different clusters based on their physical proximity. All resource discovery operations are conducted in the cycloid layer in a distributed manner. The logical distance between node i and node j in an overlay is measured by $|ID_i - ID_j|$. One major challenge in resource clustering is to relate logical proximity to physical proximity of resource nodes. A landmark clustering is adopted to generate proximity information [20]. We assume m landmark nodes that are randomly scattered in the Grid. Each node measures its physical distances to the landmark nodes. A vector of distances $< d_1, d_2, ..., d_m >$ is used to perform clustering. Two physically close nodes have similar vectors. We use space-filling Hilbert curve [20], [2] to map each m-dimensional landmark vectors to a real number. The number is called the Hilbert number of a node denoted by $\mathcal{H}$. Its purpose is to preserve the physical proximity among the nodes selected for the same cluster in HCO.

The HCO architecture builds a topology-aware cycloid architecture on a Grid. Specifically, each node generates its ID $(H, \mathcal{H})$, where H is the consistent hash value of its IP address. Recall that the first index indicates node positions within a cluster and the second index differentiates clusters. Therefore, physically close nodes with the same Hilbert number are in the same cluster, and those with similar Hilbert number are in nearby clusters. To build each node's routing table, HCO uses proximity-neighbor selection technique [6]. That is, to choose a node for a routing table entry among a number of satisfying nodes, a node selects the physically nearest node. As a result, a topology-aware cycloid is constructed, in which the logical proximity abstraction derived from overlay matches the physical proximity information in reality.

4. Locality-Preserving Grid Resource Management

Using a flat DHT, resource descriptors are gathered in different repository nodes based on resource type. Multi-resource query with m resources leads to m messages being routed in the system-wide scope, leading to high overhead. In addition, it does not gather the descriptors of physically close resources together, which otherwise can help locate physically close resources more efficiently. This has posed a challenge to achieving program/data locality, by which a node can locate physically close resources for its multi-resource query by only probing its nearby nodes in the DHT overlay.

We propose locality-preserving resource clustering and discovery using HCO. The idea is to map functional resources in the same physical cluster to logically close nodes to satisfy specific application demands. Taking advantage of the hierarchical cluster structure of HCO, the `Insert(ID,object)` function is used to gather resource descriptors within a cluster and distribute the descriptors to all nodes in the cluster based on resource type. With this clustering algorithm, the descriptors of physically close nodes are grouped in the same cluster. In addition, the logical distance between a node and a cluster on the HCO reflects the physical distance between the node and the resources whose descriptors are in the cluster. In other words, the closer logical distance between a cluster and a node, the closer physical distance between the resources grouped in the cluster and the node. This facilitates a node to locate physically close resources by probing its nearby nodes in increasing proximity. Within a cluster, resource descriptors are further grouped according to resource type. This supports a node to discover resources based on its various demands.

Successful clustering leads to fast resource discovery for various Grid applications. The `Lookup(ID)` function is used to discover multiple resources. Thus, HCO achieves program/data locality by supporting proximity-aware and multi-resource discovery. The resources are discovered with a balanced workload and reduced overhead. HCO employs randomized probing algorithm and other strategies to deal with dynamism. The resource discovery efficiency is further enhanced by the cluster-token forwarding algorithm.

4.1. Locality-Preserving Resource Clustering

In general, resources required by a Grid application is specified by a set of resources such as CPU, memory, bandwidth, I/O subsystem, etc. An effective resource discovery algorithm locates resources across a wide area based on a list of predefined attributes. We specify each resource in node i by a resource descriptor D_r, consisting of the following 4-tuple:

$$Resource\ Descriptor\ D_r =< RF, ID, RA, IP >,$$

where RF, ID and RA are the resource functionality, identification and attribute. IP refers to the IP address of the resource owner or requester. Resource requests and reports are represented by such descriptors. For clarity, all node indices are omitted in the descriptors.

HCO divides resource descriptors into categories based on resource functionality, and assigns different nodes in a cluster for different categories. In a DHT overlay, the objects with the same ID are stored in the same node. Based on this object assignment policy, HCO computes the consistent hash value H_r of a resource r, and uses $ID_r = (H_r, \mathcal{H}_i)$ to represent the ID of resource r in node i. Each node applies `Insert(`ID_r`,`D_r`)` to periodically store the descriptors of its available resources in a node, which is called *directory node*. As a result, the descriptors of the physically close resources with the same functionality are stored in the same directory node. Different nodes in a cluster are responsible for resources with different functionalities. Furthermore, resources in the directories stored in nearby clusters are located in physically close nodes.

We use *a directory node's resource* to represent the resource r whose D_r is stored in the directory node. The logical distances between node i and a number of directory nodes represent the physical distances between node i and the directory nodes' resources. Therefore, if a node has resource options in a number of directory nodes, it should choose the resource in the logically closest directory node in the overlay. Theorem 4.1 shows the feature of HCO.

Theorem 4.1: If nodes j and k are directory nodes of resource requested by node i, and $ID_i \leqslant ID_j < ID_k$ or $ID_i \geqslant ID_j > ID_k$, then directory node j's resources are physically closer to node i than directory node k's resources.

Proof: If nodes j and k are directory nodes of a specific resource, nodes j and k must be in different clusters. In HCO, the logical proximity abstraction derived from overlay matches the physical proximity information in reality. Therefore, if $ID_i \leqslant ID_j < ID_k$ or $ID_i \geqslant ID_j > ID_k$, node j is physically closer to node i than node k. A node reports its resource descriptors to a node in its cluster, so directory node j's resources are physically closer to node i than directory node k's resources. ■ □

The load balancing algorithm in [14] can be further adopted to achieve more balanced descriptor distribution between the directory nodes. Since this is not the focus of this paper, we do not present the details of the load balancing.

4.2. Locality-Preserving Resource Discovery

When node i queries for multiple resources, it sends a request `Lookup`$(H_r, \mathcal{H}_i)$ for each resource r. Each request will be forwarded to its directory node in node i's cluster. If the directory node has no requested descriptor, it probes nodes in nearby clusters. Theorem 4.1 indicates that the resources of directory nodes in closer clusters are physically closer to the requester. Hence, a node should probe its logically close neighbors in order to locate physically close resources.

We present the successor and predecessor clusters of node j's cluster as *sucCluster(j)* and *preCluster(j)*, respectively. Firstly, a node probes the directory nodes in these clusters simultaneously. Then, the directory nodes in *sucCluster(sucCluster(j))* and *preCluster(preCluster(j))* are probed. This process is repeated until the desired resource descriptors are found. However, such sequential probing is not robust enough to handle dynamism where nodes join and leave the system continually.

We incorporate the algorithm in [14] and develope *proximity-aware randomized probing algorithm (PRP)* to resolve the problem. In the PRP algorithm, a node first applies sequential probing. If no response is received during a predefined time period, the node randomly chooses two nodes in an increasing range of proximity and repeats the probing process. Since resource descriptors are allocated to different nodes in a cluster based on resource functionality, the probed nodes should be the directory nodes of the requested resource. Based on the resource clustering algorithm, we know that the probed nodes should have the closest cyclic ID to the probing node. Therefore, the probing node can reach them by targeting an ID composed of its cyclic ID and a randomized cubical ID chosen in an increasing proximity.

4.3. An Example of the HCO Algorithms

Figure 2 shows an example of using HCO for locality-preserving resource clustering and discovery. Based on the ID determination policy in HCO, physically close nodes are grouped in a cluster, and the resource descriptors of all nodes in each cluster is distributed among the nodes based on resource functionality. Physically close nodes $< a, b, c, d >$ are mapped to the same cluster, and other two groups of physically close nodes $< e, f, g, h >$ and $< i, j, k, l >$ are mapped to other two clusters. In each cluster, the resource descriptors are further grouped based on resource type. For instance, the memory descriptors of nodes $< a, b, c, d >$ are stored in node a, and the disk descriptors of these nodes are stored in node b.

The requests on memory and disk resources of nodes $< a, b, c, d >$ will also be forwarded to nodes a and b, respectively. Specifically, when a node in a cluster needs memory and disk resources, it applies `Lookup(3,200)` for memory and `Lookup(5,200)` for disk space. The requests are forwarded to nodes a and b respectively. These nodes check their own directories for the descriptors of the requested resource. If they cannot find the descriptors, they probe nodes in other clusters using the PRP algorithm. For example, node b (5,200) probes nodes by targeting (5,199) and (5,201). If it does not receive reply within a predefined time period, it randomly generates two cubical indices within 100 proximity range. Suppose the two randomized number is 150 and 250, the node probes nodes by targeting (5,150) and (5,250). If the requested resource is still not found, the node increases the proximity range and repeats the same process.

4.4. Dynamism-Resilient Resource Management

In addition to exploiting the physical proximity of the network nodes to minimize operation cost, an effective resource management scheme should also work for Grids in a dynamic environment. For example, a node departure generates outdated resource descriptors, or a failed directory node makes the resource descriptors unavailable. HCO uses cycloid self-organization mechanism to cope with these problems. Specifically, nodes transfer descriptors when joining or leaving the system.

When node i joins in the system, it reports its resources via `Insert(`$(H_r, \mathcal{H}_i)$`,`D_r`)`, and receives the descriptors in its responsible ID region from its neighbors. When a node departs from the system, it transfers descriptors to its neighbors. For example, if node $(2, 200)$ joins the system in Figure 2, then the descriptors in the range $(0, 200)$ and $(2, 200)$ are transferred from node $(3, 200)$ to node $(2, 200)$. If node $(3, 200)$ leaves, it transfers its descriptors to node $(10, 200)$ or $(5, 200)$ based on the ID closeness. If node $(3, 200)$ is the only node in its cluster, it transfers its descriptors to its closest node in its closest cluster. The consistent hashing for key assignment protocol requires simple re-assignment of resource descriptors.

HCO resorts to periodical resource reporting to avoid useful descriptors from being lost in the clustering and discovery process. If a directory node has failed, its resource descriptors are lost. Within T, the lost resource descriptors will be reported to a new directory node. To prevent the descriptor space from being flooded with outdated descriptors, the directory nodes execute garbage collection periodically. Consequently, instead of relying on specific nodes for resource descriptors, HCO always stores a resource descriptor in a directory node, and the `Lookup(`$H_r, \mathcal{H}_i$`)` requests can always be forwarded to the node.

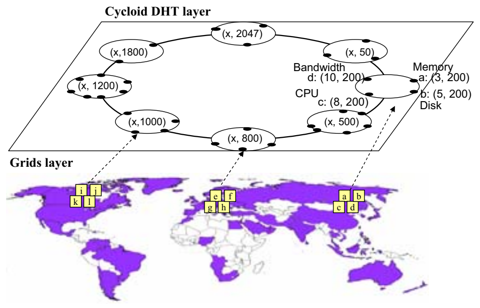


Figure 2. *Example use of the HCO network for global-scale Grid resource clustering and discovery. Grid resource clusters: $< a, b, c, d >$, $< e, f, g, h >$ and $< i, j, k, l >$ are created and managed by overlay nodes at three cycles in the cycloid hierarchy. Only partial connecting edges of the HCO overlay are shown.*

4.5. Cluster-Token Forwarding Algorithm

We introduce cluster-token forwarding algorithm to further enhance the efficiency of the HCO scheme. Like most multi-resource management approaches, HCO uses m lookups for a query of m resources. Based on cycloid routing algorithm, all lookup message are firstly routed within a cluster sequentially. Thus, rather than using m lookups, a node can combine the m lookups into one lookup message to be sequentially routed within a cluster. Moreover, since a node with available m resources needs m `Insert()` messages for resource clustering, which are routed in the same manner as the `Lookup()` messages, the two kinds of messages can be integrated. Furthermore, since the messages of all nodes in a cluster are routed in the same manner and the nodes need to report their available resources periodically, the messages for resource clustering and discovery of all the nodes can be combined.

Based on the observation, the cluster-token forwarding algorithm accumulates the messages of available resources and resource requests of all nodes in one cluster. In a nut shell, the primary node in each cluster periodically generates a token which circulates along its cluster. Each node receiving the token inserts the resource descriptors of its available resources and resource requests into the token, absorbs the descriptors of available resource and resolves the resource requests in the token that are in its responsibility.

For example, if primary node i needs multiple resources represented by $r_1, r_2, ..., r_{m1}$, and it has available resources represented by $\delta r_1, \delta r_2, ..., \delta r_{m2}$, it generates the IDs of the resources $ID_{r_1}, ..., ID_{r_{m1}}$ and $ID_{\delta r_1}, ..., ID_{\delta r_{m2}}$, in which

$$ID_{r_{\tilde{m}}} = (H_{r_{\tilde{m}}}, \mathcal{H}_i)(1 \leqslant \tilde{m} \leqslant m1 + m2).$$

The resource descriptors are ordered by H_r in the form of

$$< D_1, D_2, \cdots, D_{m1+m2} > .$$

Next, node i sends the token to its successor j.

Based on the HCO resource clustering algorithm, a node is the directory node of the resources whose H_r satisfies

$$ID_{pre.cyc} \leqslant H_r \leqslant ID_{suc.cyc},$$

where $ID_{pre.cyc}$ and $ID_{suc.cyc}$ represent the cyclic index of the node's predecessor and successor. Therefore, in order to avoid unnecessary checking, node j only needs to check the D_r in the token satisfying this condition. For those D_r of requests, if node j has resource descriptors of the required resources, it sends the resource locations to the requesters, and removes the D_r from the token. For those D_r of available resources, node j absorbs the D_r. Afterwards, it adds its own D_r of available resources and requests to the token, and forwards the token to its successor. This process is repeated until the primary node receives the token back, which means that the token completes one circulation.

At this time, the token has no D_r of available resources and the D_r left are for resource requests. The primary node uses PRP algorithm to forward the token to another cluster. At the cluster, this process is repeated until the token is empty, i.e. all requested resources are discovered. Therefore, in the algorithm, only one message is generated periodically. Combining a number of messages into a single message for forwarding within a cluster and between clusters significantly reduces cost.

5. Performance Evaluation

We designed and implemented a simulator in Java for evaluation of the HCO scheme with the cluster-token forwarding algorithm. The dimension of the cycloid simulated is 11. Thus, the DHT overlay network can accommodate 4096 nodes. We compare the performance of HCO with MAAN [4], Mercury [3], and SWORD [8]. To be comparable, we used Chord for attribute hub in Mercury and SWORD. The experimental results show advantages in using HCO over the competing overlays for the same purpose.

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. This distribution reflects the real world where there are available resources that vary by different orders of magnitude. We generated 1000 requests. The number of resources in a request ranges from 1 to 5 with step size of 1. We use HCO/o to represent HCO without the cluster-token forwarding algorithm. We used a transit-stub topology generated by GT-ITM [21] with approximately 5,000 nodes.

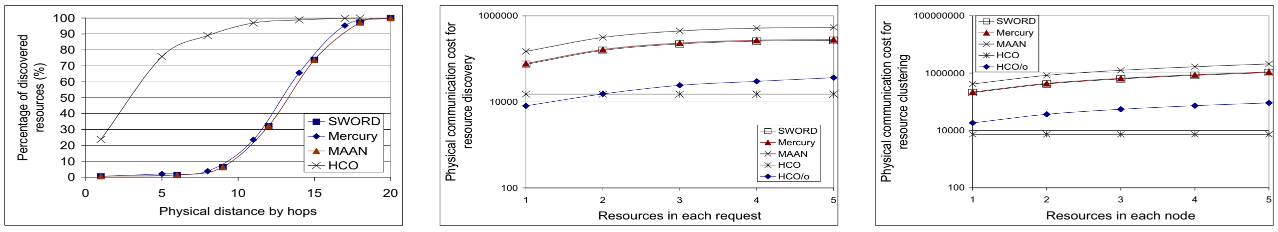
We evaluate the effectiveness of the resource management approaches in the following metrics:

(1) *Cumulative distribution function (CDF) of the percentage of discovered resources.* This reflects the effectiveness of a resource discovery algorithm to discover requested resources physically close to requesters.
(2) *Physical communication cost.* The communication cost is directly related with message size and physical routing path length. we use the product of these two factors to represent the communication cost. It is assumed that the size of a resource request message or response message is 1 unit.
(3) *Resource request success rate.* This is the percent of resource requests that arrive at their directory nodes successfully. Due to dynamic environment, some requests may fail to reach their directory nodes. This metric represents the capability of a resource management scheme to deal with dynamism.
(4) *The number of visited nodes.* Node dynamism may lead to more visited nodes in the probing phase in HCO. We use the metric to reflect the impact of dynamism on the efficiency of a resource management scheme.

5.1. Cost in Grid Resource Management

In this experiment, we randomly generated 5000 resource requests, and recorded the distance between the resource provider and requester of each request. Figure 3(a) shows the CDF of the percentage of allocated resources against the physical hop distance. We can see that HCO is able to locate 97% of total resource requested within 11 hops, while others locate only about 15% within 10 hops. Almost all resources are located within 15 hops from requesters in HCO, while 19 hops in others. The results show that HCO can locate most resources within short distances from requesters, but others locate most resource in long distances. The more resources are discovered in shorter distances, the higher efficiency of Grid applications. The results confirm the unique locality-preserving feature of HCO to enable users to locate physically close resources.

The communication cost also plays an important role in resource management efficiency. Figure 3(b) and (c) plot



(a) CDF of discovered resources

(b) Communication cost in resource discovery

(c) Communication cost in resource clustering

Figure 3. Communication cost of different resource management schemes.

the physical communication cost for resource discovery and clustering versus the number of resources in a query, respectively. From these figures, we can see that the cost of each scheme increases with the number of resources in a query. The cost of MANN grows dramatically faster than others, while HCO and HCO/o only have a marginal increase. Recall that MAAN needs two messages for each resource discovery and resource reporting, one is for resource type and the other for resource's attribute, leading to much higher communication cost. The results illustrate that HCO and HCO/o incur much less physical communication cost than others by arranging nodes to contact their physically close nodes. The results also show that the cluster-token forwarding algorithm is effective in reducing communication cost. SWORD and Mercury do not consider the proximity in the resource management process, so they generate much higher cost than HCO and HCO/o.

5.2. Performance in a Dynamic Environment

In this experiment, we run each trial of the simulation for 20T simulated seconds, where T is a parameterized resource clustering period which was set to 60 seconds. We ranged node arrival/departure time rate from 0.1 to 0.5 with 0.1 step size, and generated 5000 resource requests randomly. For instance, there were one node join and one node departure/failure every 2.5 seconds at rate 0.4. The resource join/departure rate was modelled by a Poisson process with a rate of 0.4 as in [15]; the resource type in requests are randomly distributed.

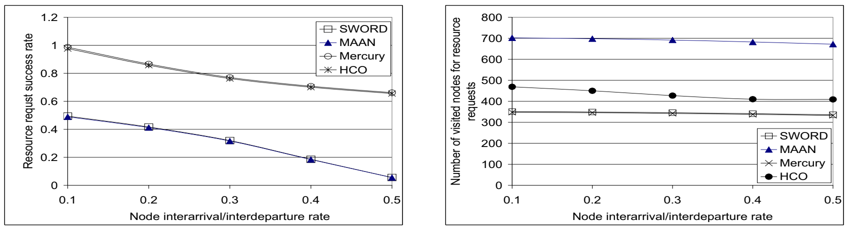
Figure 4(a) plots the resource request success rates against node arrival/departure rate. The success rate decreases monotonically in all resource management schemes. MAAN and SWORD incur lower success rates than HCO and Mercury. Because of dynamism, some requests may be lost. More requests fail to arrive at their destinations successfully when the node arrival/departure rate increases, leading to decrease of success rate. HCO and Mercury distribute resource descriptors among all nodes in the system, while MAAN and SWORD mainly depend on 11 nodes, so any departure in the 11 nodes will result in the loss of a high volume of resource descriptors, resulting in sharp drop-off of success rate.

The efficiency performance in dynamism is also reflected in the number of visited nodes for resource requests. Figure 4(b) shows the number of visited nodes for requests. MAAN has to visit more nodes than others due to its doubled lookups per query. SWORD and Mercury visit fewer nodes than HCO. This is expected because HCO has an extra probing phase to probe nodes for resources. In a dynamic environment, the number of nodes probed in the probing phase will be more than that of static environment due to node joins and departures. These results verify the superior performance of Mercury and HCO, compared with MAAN and SWORD in handling network dynamism.

6. Conclusions

Rapid development of Grids demands a scalable and efficient resource management scheme to sustain distributed performance in a dynamic wide-area environment. The major contributions of this work are summarized below: (a) This paper presents a HCO by extending the cycloid DHT overlay. The HCO pools physically close resource together in logically-close clusters. (b) We have developed locality-preserving algorithms to enable users to dynamically discover physically close resources with required functionalities in their neighborhood. Most previous schemes fail to preserve the locality and require users to discover resources in the system-wide scope. (c) The HCO scheme uses a single large DHT overlay with low overhead. It achieves balanced workload distribution and resilience to resource failure. Most previous schemes use multiple DHT-based overlays causing high overhead or one DHT overlay causing workload imbalance. Both of those are more suitable for static Grid configurations with limited applications. (d) The cluster-token forwarding algorithm enhances system efficiency. Simulation results reported demonstrate the superiority of using HCO in Grid reconfiguration for large-scale and dynamic applications.

The proposed framework is still under intensive system and middleware development. For further research, we encourage continued effort be conducted in the following aspects: (1) Prototyping of the proposed HCO network for Grid



(a) Request success rate (b) Visited nodes

Figure 4. Effectiveness of resource management schemes in dynamism.

resource management. (2) Developing benchmark programs to test the efficiency and validate the claimed advantages. (3) Apply virtual machine techniques [16] to extend the HCO model to secure and safeguard Grid applications. (4) Integrate P2P and Grid technologies with machine virtualization techniques for global-scale Internet applications. These four areas post wide open problems that are crucial to promote large-scale distributed computing in the future.

Acknowledgment

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] 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.

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

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

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

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Topology-aware routing in structured peer-to-peer overlay networks. *Future Directions in Distributed Computing*, 2002.

[7] 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.

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

[9] F. P. Preparata and J. Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *CACM*, 24(5):300–309, 1981.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware*, 2001.

[11] C. Schmidt and M. Parashar. Flexible information discovery in decentralized distributed systems. In *Proc. of HPDC*, pages 226–235, 2003.

[12] H. Shen. PIRD: P2P-based Intelligent Resource Discovery in Internet-based Distributed Systems Corresponding. *JPDC*, 2008. An early version appeared in ICDCS'08.

[13] H. Shen. SEMM: Scalable and Efficient Multi-Resource Management in Grids. In *Proc. of GCA*, 2008.

[14] H. Shen and C. Xu. Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. *TPDS*, 2007.

[15] H. Shen, C. Xu, and G. Chen. Cycloid: A Scalable Constant-Degree P2P Overlay Network. *Performance Evaluation*, 63(3):195–216, 2006.

[16] J. Smith and R. Nair. Virtual Machines,Virtual Machines: Versatile Platforms for Systems and Processes. *Morgan Kaufmann Publisher*, 2005.

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

[18] 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.

[19] S. Suri, C. Töth, and Y. Zhou. Uncoordinated load balancing and congestion games in P2P systems. In *Proc. of P2P*, 2004.

[20] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proc. of INFOCOM*, 2003.

[21] E. Zegura, K. Calvert, and et al. How to model an Internetwork. In *Proc. of INFOCOM*, 1996.

[22] Z. Zhang, S.-M. Shi, and J. Zhu. Somo: Self-organized metadata overlay for resource management in P2P DHT. In *Proc. of IPTPS*, 2003.

[23] 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. *J-SAC*, 12(1):41–53, 2004.