Efficient and Locality-aware Resource Management in Wide-area Distributed Systems

Haiying Shen, Wing-Ning Li[3]
Dept. of Computer Science and Engineering
University of Arkansas, Fayetteville, AR 72701
{hshen,wingning}@uark.edu

Yingwu Zhu[2]
Dept. of Computer Science & Software Engineering
Seattle University, Seattle, WA 98122
zhuy@seattleu.edu

Abstract

Wide-area distributed systems such as data sharing, computational grids, and multimedia are increasingly being deployed in a large-scale, heterogeneous and dynamic distributed environment with geographically scattered resources. However, most current resource management approaches are unable to simultaneously deal with the characteristics of such an environment. This paper presents an efficient DHT-based locality-aware resource management mechanism. Taking advantage of a DHT's hierarchical structure, it uses a single DHT to achieve multi-resource management with low overhead. Moreover, it has high capability to handle the characteristics of distributed systems. Simulation results demonstrate the effectiveness of the mechanism in comparison with other resource management algorithms. The mechanism performs no worse than existing locality-aware approaches and exhibits high resilience to dynamism. It also reduces the overhead of the locality-aware algorithms due to the elimination of unnecessary communications by shrinking probing scope. In addition, it yields significant improvements in the efficiency of resource discovery.

1 Introduction

Advancements in technology over the past decade are leading to a promising future for computing, where globally-scattered computing resources will be collectively pooled and used in a cooperative manner to achieve unprecedented petascale supercomputing capabilities. Wide-area distributed systems, such as grid and peer-to-peer (P2P) infrastructures, interconnect computers, clusters, storage systems, instruments, and available infrastructure of large scientific computing centers to make possible the sharing of the resources such as CPU time, storage, memory, network bandwidth, and data. Wide-area distributed applications, such as data sharing, computational grids, navigation systems, multi-party video conferencing, multimedia and telecommunications, have been widely used in scientific, engineering and commercial areas. For example, the average simultaneous users of global P2P distributed applications currently are already over 9 million [1] out of approximate 1 billion documented Internet users. Consequently, resource management has become a critical part for high performance wide-area distributed systems. Although resource management on parallel and distributed systems has been investigated extensively in the past, they are limited to cooperative and dedicated environments.

The rapid development of wide-area distributed systems has posed challenges in resource management due to their complex environments. Specifically, they are characterized by: (1) *Large scale.* Distributed systems connect millions of resources by high-speed Internet. (2) *Heterogeneity.* With the increasing emergence of diversified end devices on the Internet equipped with various computing, networking, and storage capabilities, the heterogeneity of resources in a distributed system is pervasive. (3) *Dynamism.* Participant nodes and resource availability are continuously changing. Nodes can enter or leave the system unpredictably. Resources such as available CPU time, available memory can become unavailable at any time and their values vary significantly over time. (4) *Locality.* Heterogeneous computational resources such as personal computers, clusters and online instruments may belong to different communities, and they spread across geographically distributed areas world wide.

In such a complex environment, traditional resource management approaches relying on centralized, hierarchical, or decentralized consensus based policies [10, 2, 9, 15, 12, 7, 8, 16] are insufficient in dealing with the above characteristics. Current middlewares provide necessary mechanism to meet the end-to-end QoS requirements of the appli-

cations, but they are not sufficient by themselves to manage large-scale distributed systems with dynamic heterogeneous computer resources [22]. There is an ever-increasing need for an efficient resource management mechanism for such a complex environment to deal with these characteristics.

Distributed Hash Tables (DHTs) is an important class of the overlay networks that map keys to the nodes of a network based on a consistent hashing function. It has emerged as a new paradigm for solving large-scale resource management. They enable nodes to contribute their resources while being autonomous, and let much otherwise unused resources be harvested for the development of science, engineering, and business. However, some DHT-based approaches neglect resource heterogeneity by assuming there is only one bottleneck resource [27, 13, 18, 28, 33, 14, 24], while other multi-resource management approaches neglect locality [6, 3, 26, 5, 6, 3, 17, 11, 29, 23] To meet all the requirements with focus on low cost and high efficiency, we propose a single DHT-based Efficient and LOcality-Aware resource Management mechanism (Eloam) that can deal with the characteristics with its high capabilities in terms of scalability, efficiency, reliability, and self-organization. Eloam efficiently supports large-scale distributed systems to harness heterogeneous resources of a vast number of individual computers across geographically distributed areas for high performance. It maps physically close resource requesters and providers, and further enhances the efficiency of existing locality-aware resource management approaches.

The rest of this paper is structured as follows. Section 2 presents a concise review of representative resource management approaches for distributed systems. Section 3 describes the DHT-based multi-resource management mechanism, focusing on its resource management framework and resource management algorithms. Section 4 shows the performance of Eloam in comparison with other approaches using a variety of metrics. Section 5 concludes this paper with remarks on possible future work.

2 Related Work

There have been numerous approaches to managing resources in wide-area distributed systems. Systems such as Condor-G [10] uses the Globus toolkit [9] to integrate with a grid computing environment for resource management. A number of projects, including Condor [15], XtremWeb [12], Entropia [7], AppLes [8], and Javelin++ [16], have investigated scheduling of computations on grids. However, these systems have limitation to explore in a dynamic multi-domain environment with variation of resource availability and the presence of large-scale heterogeneity due to their centralized or hierarchical features. To cope with these problems, more and more distributed systems resort to DHT [27, 25, 19, 32, 21] middleware overlays for resource management.

One group of resource management approaches is load balancing in DHT overlay networks themselves. "Virtual server" [27, 13] is a popular approach, in which each real node runs $O(\log n)$ virtual servers, and the keys are mapped onto virtual servers so that each real node is responsible for the key ID space of different length proportional to its capacity. An alternative to ID partitioning for load balance is load reassignment [18, 28, 33, 14, 24] that move load from heavily loaded nodes to lightly loaded nodes. These methods assume there is only one bottleneck resource and neglect resource heterogeneity. In addition, most of them are not able to deal with either dynamism or locality.

DHT also has been widely adopted in wide-area distributed systems for multi-resource discovery with the consideration of resource heterogeneity. Multi-resource discovery refers to the problem of locating resources that are described by a set of attributes or characteristics (e.g., OS version, CPU speed, etc.). A number of approaches have been proposed to organize resource information in order to efficiently support multi-resource queries. Some systems use one DHT for each attribute, and process multi-attribute range queries in parallel in corresponding DHTs [6, 3, 26, 5]. Depending on multiple DHTs for multi-attribute resource management leads to high maintenance overhead for DHT structures. Another group of approaches [6, 3, 17, 11, 29] organize all resources into one DHT overlay and let a node be responsible for all information of resources with the same attribute. This approach results in load imbalance among nodes, leading to high cost for searching resource information among a huge volume of information in a single node. In addition to high overhead, most current approaches are not able to deal with both dynamism and locality features simultaneously. Recently, Shen *et al.* proposed LORM [23] resource management algorithm. It uses one DHT to realize multi-attribute resource management with low overhead. Specifically, it clusters resource information based on different resource attributes, and further groups the resource information in each cluster based on resource size range. However, it does not take locality feature into account to match physically close resource requesters and providers to achieve high efficiency.

Eloam can deal with the heterogeneity, locality and dynamism features simultaneously by clustering the resource information based on resource attribute and their owners' physical closeness, and by introducing a factor of randomness in the probing process in a range of proximity. Eloam's self-organization mechanism contributes to its reliability in dynamic environment. Thus, it has high scalability and efficiency.

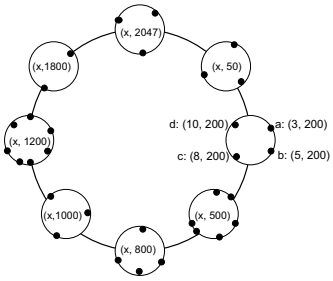

Figure 1. Cycloid node partial routing links state.

3 Efficient and Locality-aware Resource Management

3.1 Overview

Eloam resource management mechanism is built on top of Cycloid [25] DHT overlay network. Cycloid's hierarchical structure as well as flexible topology and routing algorithm facilitate to handle the challenges in resource management in wide-area distributed systems. Before we begin more detailed discussion, we briefly describe Cycloid DHT followed by a high-level view of the architecture and components of the resource management mechanism.

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, where $n{=}d \cdot 2^d$ nodes and 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 we call *cluster*. The largest cyclic index node in a cluster is called the *primary node* of the nodes at the cluster. All clusters are ordered by their cubical index mod 2^d on a large cycle. Figure 1 shows the partial routing links of a 11-dimensional Cycloid, where x indicates all possible cyclic index which ranges from 0 to 10. The links include the node's predecessor and successor in the cluster, two primary nodes of the preceding and the succeeding cycles, and one cubical neighbor and two cyclic neighbors which are not shown in the figure.

Cycloid assigns keys onto its ID space by the use of a consistent hashing function. For a given key, its cyclic index is set to its hash value modulated by d and its cubical index is set to its hash value divided by d. A key will be assigned to a node whose ID is closest to its ID. If the target node of a key's ID $(k, a_{d-1}\ldots a_1a_0)$ is a participant, the key will be mapped to the node. Otherwise, the key is assigned to the node whose ID is first numerically closest to $a_{d-1}a_{d-2}\ldots a_0$ and then numerically closest to k. For example, a key with ID (2, 150) will be assigned to node (3, 200) in the figure. For more information about Cycloid, please refer to [25].

3.2 Designs

A fundamental service of resource management is to locate resources based on predefined requirements. In the complex environment of wide-area distributed systems, a challenge for a resource management mechanism is to discover resources physically close to the resource requesters at a low cost of overhead for overall high performance of the distributed systems.

LAR [24] clusters resource information of physically close nodes and discovers resource by inter-cluster searching and then system-wide intra-cluster probing. LAR can deal with locality but assumes a single resource bottleneck. On the other hand, LORM [23] enables multiple resource management on one Cycloid DHT with each cluster responsible for a resource. But it cannot consider the locality factor in the process of resource management. Consequently, this has posed a challenge to discover resources based on both resource type and locality, and at the same time to avoid system-wide probing that comes at a high cost of node communication.

In this section, we will describe an algorithm which can deal with the locality and resource heterogeneity problems and meanwhile avoids system-wide probings. First, let us introduce a landmarking method to represent node closeness on the Internet by indices. Landmark clustering has been widely adopted to generate proximity information [20, 30]. 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 Internet. 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 $< d_1, d_2, \ldots, d_m >$ as its coordinate in Cartesian space. Two physically close nodes will have similar landmark vectors. We use space-filling curves [4], such as Hilbert curve [30], to map m-dimensional landmark vectors to real numbers, such that the closeness relationship among the points is preserved. This mapping can be regarded as filling a curve within the m-dimensional space until it completely fills the space. We partition the m-dimensional landmark space into 2^{mx} grids of equal size (where m refers to the number of landmarks and x controls the number of grids used to partition the landmark space), and number each node according to the grid into which it falls. We call this number *Hilbert number*

of a node. The Hilbert number indicates physical closeness of nodes on the Internet. The smaller the x, the larger the likelihood that two nodes will have same Hilbert number, and the coarser grain the physical proximity information.

Recall that Cycloid consists of a number of clusters, which constitute a large Cycle. We let each cluster responsible for the information of a type of resource, and divide the information among nodes within the cluster based on resource locality. In a Cycloid ID, the cubical indices differentiate clusters, while the cyclic indices indicate different node positions in a cluster. Based on the Cycloid topology, we use cubical indices to represent different resources, and use cyclic indices to represent the locality of nodes. Specially, we assign each type of resources a Cycloid ID. The ID's cubical index is the consistent hash function value of the resource's name, denoted by rescHash, and its cyclic index is the Hilbert number of the resource's owner. Since an object is assigned to the node whose ID is closest to the object in the ID space in Cycloid, if two objects have similar keys, then they will be stored in the same node or close nodes in the ID space. Based on this, we let nodes report their resource information to the system by `Insert((HilbertNum,rescHash),rescInfo)` every P seconds periodically. As a result, the information of the same type of resources will be in a same cluster. Within each cluster, the information of resources in close proximity will gather together in a node. When a node wants to query for different resources, it only needs to send request `Lookup(HilbertNum,rescHash)` for each resource. Each request for a resource will be forwarded to the node responsible for the information of the resource in close proximity. If the resource information receiver, i.e. directory node, does not have the requested resource information, it probes its neighbors in its cluster for the requested resource. The method to probe the neighbors will be described in more details in Section 3.3. Since all the information of one type of resource is in one cluster, if there's no resource information satisfying the requesters in the cluster, there's no need to search the entire system as in the LAR algorithm. Algorithm 1 shows the pseudocode of the algorithm.

Unlike existing resource management approaches, Eloam relies on a single DHT to achieve multi-resource management. It has balanced load distribution and low cost for probing.

Algorithm 1 Pseudo-code for resource information storing and retrieving conducted by a node.

```
1:  //to store resource information in local cluster
2:  get its HilbertNum
3:  get rescHash, the consistent hash value of the resource
4:  generate rescID (HilbertNum,rescHash)
5:  use DHT function Insert(rescID,recInfo) to store the resource
    information in its local cluster
6:
7:  //to request resource
8:  get its HilbertNum
9:  get rescHash, the consistent hash value of the resource
10: generate rescID (HilbertNum,rescHash)
11: use DHT function lookup(rescID) to get the resource informa-
    tion < rescName, ip_addr(i) >...
12: ask resources from node ip_addr(i)
13:
14: //when receives a resource request
15: check its rescInfo directory
16: if its rescInfo directory has rescInfo of requested resource
    then
17:   return the rescInfo
18: else
19:   while has not found the resoInfo of requested resource do
20:     probe other nodes in its cluster
21:   end while
22: end if
23: return the rescInfo
```

3.3 Dynamism-resilient Resource Management

Resource discovery in dynamic environment is a challenge in a large-scale and dynamic wide-area distributed systems. For example, a node departure generates outdated resource information, or a failed directory node makes the resource information unavailable. In addition to exploiting the physical locality of network nodes to minimize operation cost, an effective resource management algorithm should work for wide-area distributed systems with dynamic node joins and departures. Cycloid has self-organization mechanism to maintain its structure and data in a distributed manner, which helps Eloam to handle dynamism. When a node joins in the system, in addition to reporting its resources, it gets the resource information that is in its responsible ID region from its neighbors based on Cycloid's key assignment policy. When a node departs from the system, it transfers its resource information to its neighbors. Since the resource information transfer is exactly the same as file transfer in dynamism based on the key allocation policy, the resource information can be transferred along with the file to save extra communication overhead.

For node failures or departures without warning, Eloam resorts to the periodical resource information reporting by which the lost resource information is recovered. Specifically, when a node receives a resource request, if it cannot locate requested resource, it assumes that the old directory node of the resource information failed, and waits for P seconds which is the resource information reporting time period. Within P seconds, the lost resource information will be reported to its new directory. To prevent the information space from being flooded with out-

dated information left behind by node or network failure or even maliciously injected false information, nodes execute garbage collection periodically. That is, after a period of time, if a node has not received resource information from another node, it deletes the outdated information of the node. With this mechanism, instead of relying on specific nodes for resource information, resource information is always stored in a node responsible for the ID region where the information ID locates even in dynamic situation, and the `Lookup(HilbertNum,rescHash)` requests for the resource will always be forwarded to the node.

Algorithm 2 Pseudo-code for probing in resource management performed by a directory node.

performProb($< R_i$,T_i,$ip\_addr(i) >$){

```
1: success=0;
2: suc=successor
3: pre=predecessor
4: while suc!=itself and pre!=itself and !success do
5:    probe suc and pre for their ARL: ARL_suc and ARL_pre
6:    if get response from suc within predetermined period time
      then
7:       success=performAssign(<R_i, T_i, ip_addr(i)>,
         ARL_suc)
8:       suc=suc.successor;
9:    end if
10:   if get response from pre within predetermined period time
      then
11:      success=performAssign(<R_i, T_i, ip_addr(i)>,
         ARL_pre)
12:      pre=pre.predecessor;
13:   end if
14:   if no response from suc after predetermined period time
      then
15:      generate a randomized number num between
         suc.cyclicID and (suc.cyclicID+range)%d
16:      suc=(num, cubicalID)
17:   end if
18:   if no response from pre after predetermined period time
      then
19:      generate a randomized number num between
         pre.cyclicID and (pre.cyclicID-range)%d
20:      pre=(num, cubicalID)
21:   end if
22: end while
23: return success;
24: }
```

In the previous section, we mentioned that if a directory node does not have information of a requested resource, it probes its neighbors in its cluster. Since Hilbert number represents the physical closeness of nodes, the resource information of physically close nodes will gather in the same or logically close nodes in Eloam. Hence, with locality consideration, a node should probe its logically close neighbors in order to map physically close resource requesters and providers. Specifically, a node probes its successor and predecessor at the same time at first, and then the successor of its successor and the predecessor of its predecessor, and so on until the resource information is found or itself is reached. However, such sequential probing may fail due to dynamism. It is proved in [24] that 2-way randomized probing is effective in dealing with dynamism, and it could achieve faster speed over one-way probing, but $d(> 2)$-way probing may not result in much additional improvement. Based on this, the system uses a 2-way randomized probing method. Specifically, it sets a time period. If a node does not get reply within the time period after sequential probing, it assumes that the probed node has departed. It then randomly generates two cyclic ID within a increasing range of proximity in two directions, and probes the nodes with the random IDs. For example, if a node does not receive reply from node with ID (5,200), it randomly generates two cyclic ID within 2 and 8 given range 3. Suppose the two randomized number is 6 and 3, the node then probes nodes (6,200) and (3,200) at the same time. If requested resource still is not found, the node increases the proximity range and repeats the same process. Eloam introduces a factor of randomness in the probing process in a range of proximity to deal with the dynamism and locality simultaneously. As is known that randomized probing is an effective way to handle dynamism. Probing in a increasing range of proximity helps to map physically close resource requesters and providers. Hence, Eloam's probing method improves the resource management efficiency. Algorithm 2 shows the pseudocode for probing in resource management performed by a directory node.

4 Performance Evaluation

We designed and implemented a simulator in Java for evaluation of the efficient and locality-aware resource management mechanism (Eloam) based on Cycloid DHT. We compared Eloam with LAR [24] and LORM [23]. LORM maps resource type and value to two levels in hierarchical Cycloid. We modified LAR for multi-resource management. LAR conducts resource management in a cluster with physically close nodes first, and then uses system-wide randomized probing for resource searching. The number of nodes was set to 4,096. We assumed there are 100 types of resources and we used Bounded Pareto distribution function to generate the resource amount owned and requested by a node. In the experiment, the resource requesters are randomly chosen.

We use a transit-stub topologies generated by GT-ITM [31]: "ts5k-large" with approximately 5,000 nodes each. "ts5k-large" has 5 transit domains, 3 transit nodes per

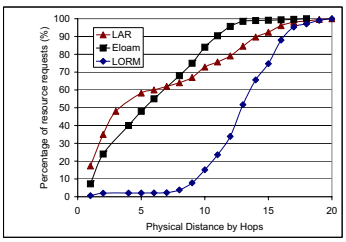

Figure 2. CDF of total resource requests distribution of different schemes.

transit domain, 5 stub domains attached to each transit node, and 60 nodes in each stub domain on average. "ts5k-large" has a larger backbone and sparser edge network (stub). It is used to represent a situation in which a DHT overlay consists of nodes from several big stub domains. To account for the fact that interdomain routes have higher latency, each interdomain hop counts as 3 hops of units of latency while each intradomain hop counts as 1 hop of unit of latency.

4.1 Locality-aware Resource Management

This section shows how Eloam helps to achieve high locality-aware performance by mapping physically close resource providers and requesters. In the experiment, we randomly generated 10,000 resource requests, and recorded the distance between the resource provider and the resource requester of each request.

Figure 2 shows the cumulative distribution function (CDF) of the percentage of allocated resources versus distance between resource requesters and providers of different resource management schemes. We can see that in "ts5k-large," Eloam and LAR are able to allocate 90% and 80% of total resources requested within 11 hops respectively, while LORM allocates only about 15% within 10 hops. Within 15 hops, almost all resources requested are allocated in Eloam and LAR, while the LORM scheme allocates only 75% of the resource. The results show that Eloam and LAR can allocate most resources within short distances from requesters while LORM allocates most resources in long distances. It demonstrates the high locality-aware performance of Eloam and LAR to locate nearby resources for the resource requesters in resource management. The figure also shows that the LAR allocates more resources within 7 hops than Eloam, while Eloam allocates more resources between 7 and 15 hops. Our experiment results show that there are 416 clusters in the Cycloid. It means that in LAR, physically close resources information is partitioned into 416 parts, while in Eloam the information is partitioned into at most 11 parts based on physically closeness. Fine-grained resource information in LAR leads to higher locality-aware performance. However, less information in one directory leads to lower possibility of successful response to required resources. To solve this problem, LAR depends on randomized probing in the entire system for requested resources. This explains why LAR does not perform as well as Eloam in distances within hop 7 to hop 15. The more resource located within shorter distances, the higher locality-aware performance of a resource management scheme with less communication cost. The results indicate that the performance of Eloam locality-aware resource management mechanism is better than LORM, and is comparable with LAR with regards to locality-aware performance.

4.2 Efficiency of Resource Management

Communication cost When a directory node does not have information of required resources, it probes other directory nodes for requested resources. Therefore, resource probing constitutes a main part of resource management overhead. The cost is directly related to message size and physical path length of the message travelled; we use the product of these two factors of all resource probings to represent the cost. It is assumed that the size of a message is one unit. In the experiment, we ranged the number of resource requests from 3,000 to 13,000, with step size of 1,000. Figure 3 plots the probing cost of Eloam, LAR and LORM in "ts5k-large" respectively. From these figures, we can see that the probing cost of LORM remains at 0, and the cost of Eloam and LAR increases with the request number. The cost of LAR grows dramatically faster than Eloam, while Eloam only has a marginally increase. The LORM scheme stores resource information of a resource in a certain range in a node. If the directory node does not have resource information of requested resources, then there is no requested resource in the system. Therefore, the directory node does not need to probe other nodes for the resource, leading to zero probing cost. In LAR scheme, the directory node probes nodes in the entire system using randomized-probing. Therefore, the directory node may contact nodes very far away from itself, and non-existing resource requested may lead to infinite probing. In addition, the fine-grained resource information in LAR contributes to more directory node probings before the requested resource is located. On the other hand, in Eloam, the directory node only needs to probe other nodes in its clus-

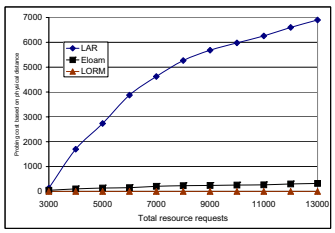

Figure 3. Physical probing cost of different resource management schemes.

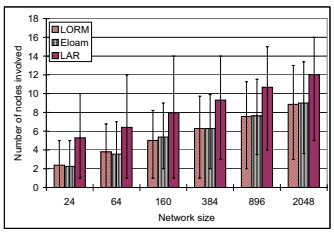

Figure 4. Number of nodes involved.

ter rather than nodes in the entire system. Consequently, its reduced probing scope helps Eloam significantly reduce the probing cost of LAR. Combined with the experiment results of locality-aware performance, the results demonstrate that Eloam achieves locality-aware resource management at a cost of marginally higher communication overhead than LORM. However, LORM has poor performance in mapping physically close resource requester and suppliers. LAR is comparable with Eloam in locality-aware performance, but it comes at a high cost of node communication overhead.

Number of nodes involved This experiment test the 1st, 99th percentiles and the average number of nodes involved in a resource location operation, which includes message routing nodes and probed nodes. We measured the metric as a function of network size with $n = d \cdot 2^d$ nodes, and the dimension d is varied from 3 to 8. The number of requests was set to the twice the number of nodes. The results are shown in Figure 4. We can see that the number of nodes of Eloam and LORM are approximately the same. However, the number of involved nodes of LAR is much higher than Eloam and LORM. Since all of them are built on Cycloid, the number of nodes for forwarding a message from a requester to the directory node should be the same between them. LAR incurs much more involved nodes is because that it probes nodes for required resources in a system-wide manner. It groups the information of different physically close resources in one cluster. Therefore, when a cluster does not have the information of a requested resource, the nodes in the other clusters should be probed, leading to high number of involved nodes. In contrast, Eloam and LORM group each type of resource in one cluster. Thus, if a cluster does not have the information of requested resource, it is not necessary to probe nodes in other clusters. The probing scope reduction from system-wide to cluster-wide helps to reduce the number of nodes involved, resulting in less overhead and high efficiency.

5 Conclusions

Rapid development of wide-area distributed systems requires an efficient resource management mechanism to deal with their characteristics of heterogeneity, large scale, dynamism and proximity. This paper presents an efficient and locality-aware resource management mechanism, which is built on a single DHT to deal with these challenges. Unlike most previous approaches which either neglect the resource heterogeneity or the locality characteristics, the mechanism is able to locate physically close multiple resources for requesters, and meanwhile deal with dynamic feature. Furthermore, it improves the efficiency and scalability of existing locality-aware resource management approaches. Simulation results show the superiority of the mechanism, in comparison with a number of other resource management approaches in terms of locality-awareness and efficiency.

We plan to further investigate the efficiency, scalability, and applicability of the proposed mechanism for low-overhead multi-resource management given a large-scale, heterogeneous, dynamic distributed environment with geographically scattered resources. Using semantic information for precise resource discovery in such an environment is a relatively new research topic. Distributed matchmaking and semantic-based routing will promise improvements in discovery precision and cost. We envision that resource management mechanisms that use DHT-based networks for scalable and reliable storage of semantic information will appear soon in distributed systems. Hence, we plan to further explore and elaborate the mechanism design to discover resources based on semantic information.

Acknowledgments

This research was supported in part by the Acxiom Corporation.

References

[1] Average simultaneous global P2P users. http://www.slyck.com/misc/p2p_history_sep-06_average.xls.

[2] D. Abramson, J. Giddy, and L. Kotler. High Performance Parametric Modelling with Nimrod/G: Killer Application for the Global Grid? In *Proc. 14th International Parallel and Distributed Processing Symposium*, 2000.

[3] 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.

[4] 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.

[5] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *Proc. of ACM SIGCOMM*, pages 353–366, 2004.

[6] M. Cai, M. Frank, and P. Szekely. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. *Grid Computing*, 2(1):3–14, 2004.

[7] A. Chien, B. Calder, S. Elbert, and K. Bhatia. Entropia: Architecture and Performance of an Enterprise Eesktop Grid System. *Journal of Parallel and Distributed Computing*, 63(5), May 2003.

[8] F. B. et. al. Adaptive Computing on the Grid Using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4), Apr. 2003.

[9] I. Foster and C. Kesselman. Globus: a Metacomputing Infrastructure Toolkit. *Int. J. High Performance Computing Applications*, 2:115–128, 1997.

[10] 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.

[11] T. Friese, B. Freisleben, S. Rusitschka, and A. Southall. A Framework for Resource Management in Peer-to-Peer Networks. In *International Conference NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World*, pages 4–21, 2002.

[12] C. Germain, V. Neri, G. Fedak, and F. Cappello. XtremWeb: Building an Experimental Platform for Global Computing. In *Proc. of the 1st IEEE/ACM International Workshop on Grid Computing (Grid)*, Dec. 2000.

[13] P. Godfrey and I. Stoica. Heterogeneity and Load Balance in Distributed Hash Tables. In *Proc. of INFOCOM*, 2005.

[14] D. R. Karger and M. Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *Proc. of IPTPS*, 2004.

[15] 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.

[16] M. O. Neary, S. P. Brydon, P. Kmiec, S. Rollins, and P. Capello. Javelin++: Scalability Issues in Global Computing. *Future Generation Computing Systems Journal*, 15(5-6):659–674, 1999.

[17] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Scalable Wide-Area Resource Discovery. Technical Report TR CSD04-1334, Univ. of California, 2004.

[18] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load Balancing in Structured P2P Systems. In *Proc. of IPTPS*, 2003.

[19] 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.

[20] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proc. of INFOCOM*, 2002.

[21] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *Proc. of the USENIX Annual Technical Conference*, 2004.

[22] R. Schantz, J. P. Loyall, C. Rodrigues, D. Schemidt, Y. Krishnamurthy, 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.

[23] 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.

[24] H. Shen and C. Xu. Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):849–862, 2007.

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

[26] 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.

[27] I. Stoica, R. Morris, D. Liben-Nowell, and et al. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 1(1):17–32, 2003.

[28] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica. Load Balancing in Dynamic Structured P2P Systems. *Performance Evaluation*, 63(3):217–240, 2006. An early version appeared in INFOCOM, 2004.

[29] S. Suri, C. Töth, and Y. Zhou. Uncoordinated Load Balancing and Congestion Games in P2P Systems. In *Proc. of the Third International Workshop on Peer-to-Peer Systems*, 2004.

[30] Z. Xu and et al. Turning Heterogeneity into an Advantage in Overlay Routing. In *Proc. of INFOCOM*, 2003.

[31] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of INFOCOM*, 1996.

[32] B. Y. Zhao, L. Huang, and et al. Tapestry: An Infrastructure for Fault-tolerant wide-area location and routing. *IEEE Journal on Selected Areas in Communications*, 12(1):41–53, 2004.

[33] Y. Zhu and Y. Hu. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 16(4), 2005.