Improving the Lookup Performance of Chord Network by Hashing Landmark Clusters

Yun-Shuai Yu
Department of Electrical Engineering, National Cheng Kung University, Taiwan, R. O. C.
yu@hpds.ee.ncku.edu.tw

Yu-Ben Miao
Information & Communication Research Labs, Industrial Technology Research Institute
Taiwan, R. O. C.
miao@itri.org.tw

Ce-Kuen Shieh
Department of Electrical Engineering, National Cheng Kung University, Taiwan, R. O. C.
shieh@ee.ncku.edu.tw

Abstract - DHTs are efficient peer-to-peer systems which can locate objects within an bounded amount of overlay hops. Originally, those systems don't exploit network proximity in the underlying Internet and lead to high latency when searching a target. Recently, some approaches, such as *Random Landmarking* (RLM) and *Lookup-Parasitic Random Sampling* (LPRS), have been suggested to build topology-aware overlay to improve the lookup efficiency. Although these approaches help the nodes in P2P system to be aware of the underlying network, they do a limited improvement because of their proximity neighbor selection. In this paper, a *hashing landmark clusters* (HLC) method is proposed to improve the performance of Chord by improving the accuracy of proximity mapping between overlay network and physical network. The analysis demonstrates that our approach can reduce the lookup latency better than RLM & LPRS can.

1. INTRODUCTION

The increasing popularity of Peer-to-Peer applications poses many challenges for architects to design an efficient and scalable P2P overlay networks. Distributed Hash Table (DHT) [1-4] can help to build a structured P2P system. One of the most famous DHT-based systems is Chord [1]. Nowadays, many applications, such as CFS [11] and P2P-SIP [7-8], are implemented by Chord. Although Chord can guarantee a bound on the number of overlay routing hops that have to be taken to discover a resource, it doesn't take the underlying network topology into consideration. One hop in the overlay may result in one or more hops in the physical network. This uncertainty makes the routing optimization difficult and harmfully affect the performance of the Chord overlay networks.

Many approaches, such as *Random Landmarking* (RLM) [5] and *Lookup-Parasitic Random Sampling* (LPRS) [6], are proposed to solve the above problem. Although these approaches help the nodes in Chord system to be aware of the underlying network, they do a limited improvement because of their proximity neighbor selection. In this paper, a *hashing landmark clusters* (HLC) method is proposed to improve the lookup performance of Chord by improving the accuracy of proximity mapping between overlay network and physical network. The lookup performance is determined by the *hops* and the *hop-reach* [6], where the hops are the overlay routing hop counts and the hop-reach is the average network latency incurred by each hop. Since Chord has optimized the hops, HLC focuses the efforts on reducing the hop-reach. The idea behind HLC is to help each Chord nodes have as more fingers pointing to proximity neighbors as possible. Compared with existing solutions, HLC can improve the performance by *13% to 35%*.

The remainder of this paper is organized as follows. Section 2 introduces Chord briefly and discusses related works. Section 3 describes the design of HLC. Finally, we evaluate the performance of HLC in Section 4 and conclude in Section 5.

2. BACKGROUND AND RELATED WORK

Chord is a ring-based DHT for structured P2P systems. In an N-node Chord, every node stores at most Log$N$ entries in its finger table to point to other peers. The i-th entry in the table of a node whose ID is n contains the pointer to the first node, s, that succeeds n by at least 2^{i-1} on the ring, where i is between 1 and log$N$. When this node n wishes to lookup the node that is responsible for key k, it will searches its finger table for a node j whose ID immediately precedes k, and passes the lookup message to j. j then recursively repeats the same operation. At each step, the overlay space that needed to be searched is at most half of the one in the previous step. Therefore, Chord can guarantee the number of overlay hops for a lookup is at most log$N$ in an N-node network.

However, each step of the lookup process may induce long latency on the physical network and hence degrades the performance of Chord. RLM [5] make use of the landmark technique [9,10] to make the overlay network topology congruent with the underlying physical topology. The nodes close to each other will form clusters on the Chord ring. Thus, the first few fingers of a Chord node can have low network latency since they point to the nodes in the same cluster it locates. Nonetheless, other fingers do not point to proximity nodes.

In contrast with RLM, LPRS [6] breaks one invariant of Chord to allow a node n to choose its i-th finger to refer to the physically nearest among all nodes whose ID is between $n+2^{i-1}$ and $n+2^i$. The success of this design depends on the number of nodes that are eligible for one finger. Because the last few fingers have large *finger intervals*, they should have more candidates and the proximity of chosen fingers can be

0-7803-9746-0/06/$20.00©2006 IEEE

expected. But for the first few fingers, the limited number of candidates makes these fingers difficult to point to proximity nodes and may degrade the performance of Chord.

3. DESIGNS

In this section, we present the details of HLC as it applies to improve the lookup performance of Chord network. We first give the definition of landmark keys. The nodes that want to take over those keys will help other nodes construct a topology-aware overlay. We then describe an enhanced joining procedure, which can assist the joining node to get an ID near its physically close nodes. Last, HLC amends node's fingers to point to proximity neighbors to lower the hop-reach.

3.1 Landmark node configuration

First, a small group of node IDs is defined to be the *landmark keys*. The landmark keys denote the set $S = \{\bigcup_{i=0}^{m-1} \left\lfloor \frac{Ni}{m} \right\rfloor\}$. N is the overlay network size, and m is a configuration parameter to determine the number of landmark keys. The nodes responsible for the landmark keys are called *landmark nodes* and will assist other nodes to find their appropriate positions in the overlay. The nodes physically close to the landmark node with ID *Ni/m* will pick up the IDs lies in the interval (Ni/m, N(i+1)/m). This range is defined as the *la*ndmark cluster.

The nodes responsible for the landmark keys are not a set of fixed servers. If the landmark nodes failed or left the Chord network, their successors will take over their landmark keys automatically. This design can save the cost to build and maintain the servers. But since the nodes are not expected to know the landmark nodes in advance, each node has to make lookup requests to find the network addresses of all the existing landmark nodes. To decrease the amount of lookup messages, the landmark nodes can periodically exchange the information of their positions. Once the joining node finds a landmark node, it gets the information of all other landmark ones.

3.2 Node ID assignment

The HLC joining procedure is shown in Fig. 1. Before joining the Chord network, the node has to lookup for all the landmark keys. If there is no landmark node at current Chord network, the joining node will be a landmark node and take over all the landmark keys. Otherwise, the joining node does measurements to determine the network latencies to all the existing landmark nodes. If the network latency between the joining node and a landmark node is lower than a preconfigured threshold, *th*, it means that the joining node is very close to the landmark one. Hence the node would assign itself an ID to join the cluster that succeeds the nearest landmark node.

If all the measurements are higher than the threshold, it means the joining node is far away from the existing landmark nodes. At this time, the node will check the amount of available landmark keys. It is determined by whether a landmark node has more than one landmark key or not. When someone does have two or more landmark keys, the joining node can be a new landmark node by asking a landmark key from the existing landmark node. But if there is no available key, the node has to join the landmark cluster of the closet landmark node, no matter the latency is higher than the threshold.

The policy to choose a landmark key is an important design of HLC. When deciding to become a landmark node, the joining node determines its node ID according to the hash function used by traditional Chord. Then it picks up one available landmark key that is closest to the hashed ID. It seems that the selected landmark key is determined by the hash function. Since every landmark cluster succeeds its corresponding landmark node, the position of the landmark cluster seems to be derived from the hash function, too. This characteristic can bring benefits in the third step: proximity finger selection.

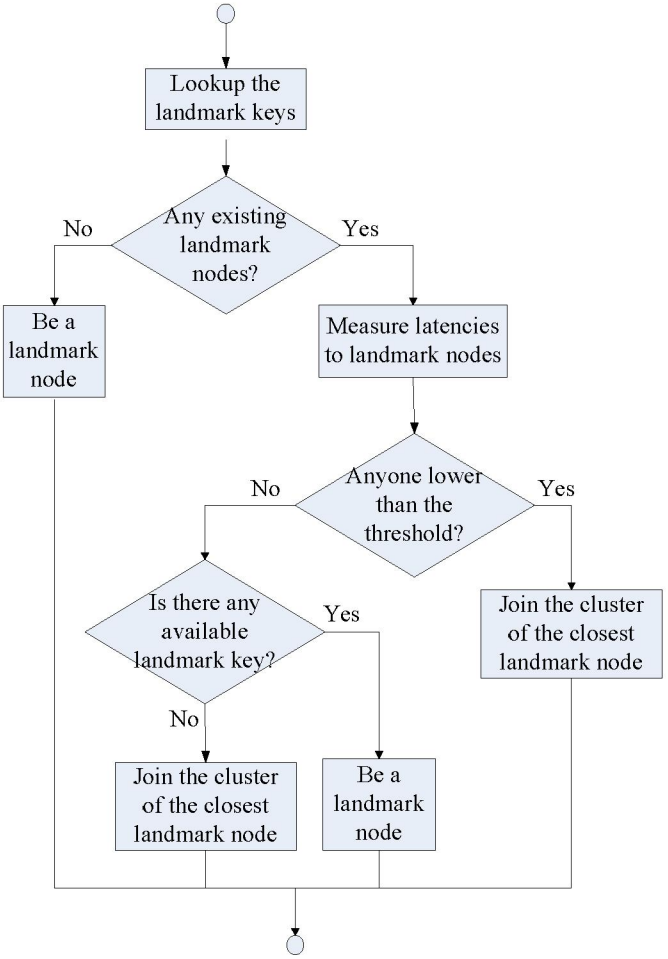

Fig. 1. HLC joining procedure

3.3 *Proximity finger selection*

After joining the Chord network, the first few fingers of each node can point to the nodes in the cluster it locates and thus achieve short hop-reach. But their higher few fingers still point to topologically distant nodes. When the node finds the hop-reach of its finger is greater than a threshold, *th2*, the node perceives that it points to a far landmark cluster. Thus, the node will randomly sample a small number of nodes from the range of the finger, measures the network latency to each sampled node, and select the one with smallest latency as the finger for the range. It has been proven in [6] that only log$N$ samples can dramatically improve the latency.

Recall that HLC uses the hash function to determine the positions of the landmark clusters. The hash function has an interesting feature: The clusters close to each other in the physical network will not gathered at a small region in the overlay space. It is shown in Fig. 2 and Fig. 3. Originally, two fingers of node A will point to distant nodes D and E due to the design of Chord. But in HLC, A will use the proximity finger selection procedure to select node B and C as its fingers. Because the clusters that B and C locate in are near the one that A resides in, the network latencies between them are still acceptable. *In sum, HLC can not only help each Chord node have the first few fingers point to the nodes in the cluster it is in but also make its last few fingers point to the nearby clusters.*

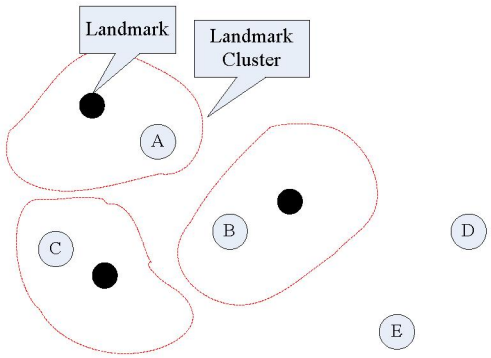

Fig. 2. Landmark clustering in physical space

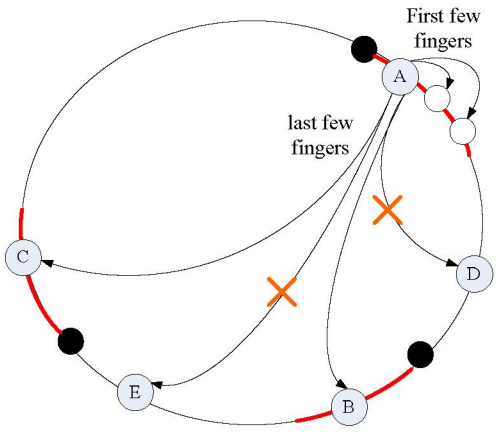

Fig. 3. Landmark clustering in overlay network

4. SIMULATIONS AND PERFORMANCE EVALUATION

The most challenge of HLC is to determine the number of landmark keys to trade off the *locality* for the extra *network overhead*. The locality means the average latency between each pair of nodes in the same cluster. Recall that in section 3.1, when m is too small, the nodes far away from each other may be in the same cluster. Otherwise, if m is too large, a tremendous amount of traffic will be generated to estimate the network latencies between the joining nodes and the landmark nodes. In this section, we determine the HLC's parameters by the results of simulations and then evaluate its performance through analysis.

4.1 *HLC's parameters determination*

To determine HLC's parameters, we first investigate the locality for different underlying physical networks. 10,000, 60,000, and 300,000 participating nodes with randomly, uniformly distributed plane topologies are considered. The geographic distance is used as an approximation for the propagation latency between two nodes. The maximum network latency between the nodes with the largest geographic distance is set as 500ms. We, then, use different number of landmark keys to exam the locality. Fig. 4 shows the locality of the 3 network sizes. When there are 128 landmark keys, the network has converged to its eventual locality value of slightly above 30ms. Thus we determine the parameter m to be 128.

To determine the threshold *th*, we measures the average network latency between each node and its closet landmark node. In the condition that m equals 128, the network latency from a joining node to its closet landmark node is about 28ms. Therefore we set the parameter *th* mentioned in section 3.2 as 30ms. We also set the parameter *th2* appeared in section 3.3 as 100ms. The simulation result shows that there are 7 nearby clusters for a node if *th2* equals 100ms.

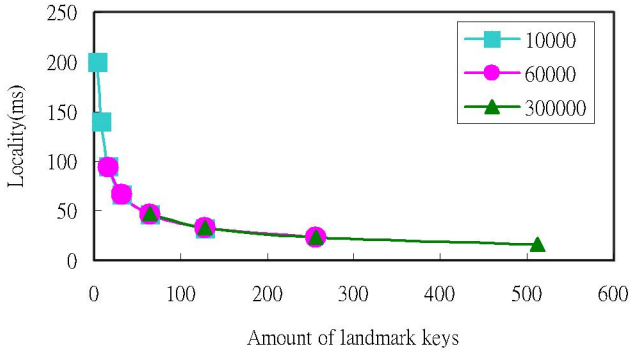

Fig. 4. The locality of landmark cluster

4.2 Performance evaluation and comparison

Then, we evaluate the performance of HLC using the above network model with 300,000 nodes and the finger table size is 19. The focus of HLC is on reducing the hop-reach to improve the lookup latency. Hence, we weight the network latency of each finger to be the performance metric. To clarify the comparison, we classify the fingers with different latencies into 3 groups:

- ***Local cluster finger***: the fingers pointing to the nodes in the same cluster it locates. Through the simulation, the average network latency of those fingers is 33ms.
- ***Nearby cluster finger***: the fingers pointing to the nodes in the nearby clusters. Through the simulation, the average network latency of those fingers is 65ms.
- ***Far cluster finger***: the fingers pointing to the nodes in the far clusters. Through the simulation, the average network latency of those fingers is 282ms.

First, because m is set as 128, there are average 2343 nodes in a landmark cluster. Then, the average number of local cluster fingers of the HLC nodes is *10.26*. Second, the positions of the nearby clusters for a node are determined by the hash function. We expect the hash function is a random number generator with uniform distribution. Thus, the probability p for a cluster to be in the k^{th} entry in the finger table of another node is the ratio between the range of the k^{th} finger and the overall Chord space. If there were y clusters in the real world, the probability of that at least one cluster mapped into the k^{th} finger range of another node will, then, be $1-(1-p)^y$. Since there are about 7 nearby clusters in HLC, the expected number of nearby cluster finger is *1.68*. Last, HLC nodes have *7.06* far cluster fingers.

RLM also has *10.26* local cluster fingers if it runs in the same environment with HLC. However, it has no nearby cluster fingers. In the case of LPRS, the probabilities of the nearby nodes to become the finger of another node can also be calculated by the formula, $1-(1-p)^y$. Since there are 2343 nearby nodes, the last *5.25* fingers of LPRS nodes should have network latency lower than 33ms. The nodes with latency from 33ms to 100ms can be the last few fingers of another LPRS node, too. Those fingers are similar to the nearby cluster fingers of HLC. The expected value of them is 6.46. But since the last 5.25 fingers have pointed to the nearby nodes, LPRS has only *1.21* nearby cluster fingers.

Table 1. The comparison of lookup performance for all approaches.

	Chord	RLM	LPRS	HLC
Local cluster finger	0	10.26	5.25	10.26
Nearby cluster finger	0	0	1.21	1.68
Far cluster finger	19	8.74	12.54	7.06
Average lookup latency (ms)	5358	2803	3788	2439

Table 1 shows the lookup latency for all the solutions when the lookup process takes 15 overlay hops. Compared with RLM, an improvement of 13% is gained. Even more, the latency of HLC is only 2/3 of the one of LPRS.

5. CONCLUSION

In this paper, we propose an approach HLC to improve the performance of Chord overlay network. It achieves the proximity by means of landmark node configuration, node ID assignment and proximity finger selection. Consequently, the average lookup latency for Chord can be reduced significantly. In a 300,000-node Chord, our simulation results show that HLC performs 13% and 35% better than RLM and LPRS, respectively. The HLC is suit to but not limited to the Chord system. The current authors are working on applying HLC to different P2P systems and will publish the achievements in the near future.

6. REFERENCES

[1] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. *Chord: A Peer-to-peer Lookup Service for Internet Applications.* ACM SIGCOMM, 2001.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-addressable Network.* ACM SIGCOMM, 2001.

[3] A. Rowstron and P. Druschel. *Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems.* International Conference on Distributed Systems Platforms (Middleware), Nov 2001.

[4] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. *Tapestry: An Infrastructure for Fault-resilient Wide-area Location andRouting.* Tech. Report UCB/CSD-01-1141, U.C. Berkeley, April 2001.

[5] R. Winter, T. Zahn, and J. Schiller. *Random Landmarking in Mobile, Topology-Aware Peer-to-Peer Networks.* Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04), 2004.

[6] H. Zhang, A. Goel, and R. Govindan. *Incrementally Improving Lookup Latency in Distributed Hash Table.* ACM SIGMETRICS, 2003.

[7] K. Singh and H. Schulzrinne. *Peer-to-Peer Internet Telephony using SIP.* New York Metro Area Networking Workshop, City University of New York, New York, NY, Sep 2004.

[8] D. Bryan, B. Lowekamp, and C. Jennings. *SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System.* AAA-IDEA, 2005.

[9] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, *Topologically-Aware Overlay Construction and Server Selection*, in Proceeding of IEEE Infocomm, 2002.

[10] Z. Xu, C. Tang, and Z. Zhang. *Building Topology-Aware Overlays using Global Soft-State.* In ICDSC'2003, May 2003.

[11] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, *Wide-area cooperative storage with CFS*, in the Proceedings of the 18th SOSP, 2001.