DUP: Dynamic-tree Based Update Propagation in Peer-to-Peer Networks*

Liangzhong Yin and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
E-mail: {yin, gcao}@cse.psu.edu

Abstract

In peer-to-peer networks, indices are used to map data id to nodes that host the data. The performance of data access can be improved by actively pushing indices to interested nodes. This paper proposes the Dynamic-tree based Update Propagation (DUP) scheme, which builds the update propagation tree to facilitate the propagation of indices. Because the update propagation tree only involves nodes that are essential for update propagation, the overhead of DUP is very small and the query latency is significantly reduced.

1 Introduction

In peer-to-peer networks, a data object can be searched by its name, usually called *Key*. This paper focuses on the structured peer-to-peer network [2], where the network relies on a hash function to map the key to a virtual space. Each node in the network is responsible for part of this virtual space. It maintains the $(key, value)$ pair for all keys that fall into its responsible area. The $value$ in the pair indicates the nodes that host the data corresponding to the key. A node is the *Authority Node* of the (key, value) pairs it maintains.

Data is inserted or removed from nodes in the network from time to time, and nodes may join or leave the network at any time. When such a change happens, the node that hosts the data should inform the authority node. It also needs to send *keep-alive* messages periodically to the authority node to deal with node failures. The authority node needs to update the index, i.e., the (key, value) pair, whenever it receives update messages or considers the node hosting the data is dead because it did not receive the keep-alive message from the node for a specific amount of time.

When a node needs to locate a data object, its request is routed along the well-defined search path of the structured peer-to-peer network, called the *index search tree* from the current node to the authority node. The search finishes when the request reaches a node that has the mapping information for that object. This can either be an intermediate node that caches the mapping information or the authority node. An index that indicates the address of the node hosting the data is then sent back to the requesting node, which retrieves the data from the hosting node as indicated by the $value$ in the (key, value) pair.

To reduce the index query latency, the index can be cached by intermediate nodes along the query path so that intermediate nodes can serve queries for the index later [1, 2]. To maintain cache consistency, invalidation-based approach can be used, where the authority node keeps track of all nodes that cache the data, and sends invalidation messages to them when the data is changed. However, the invalidation-based approach can not be directly applied to peer-to-peer networks due to scalability issues. Since there are huge number of nodes in peer-to-peer networks, it is impossible for the node that maintains the mapping information to keep track of all nodes caching the index. Therefore, the task of tracking caching nodes and pushing updates should be distributed in the network. Further, because the index size is very small, to do cache invalidation, the updated index should be sent so that caching nodes need not request for the updated index again. Following these ideas, we propose a *Dynamic-tree based Update Propagation* (DUP) scheme. In DUP, on top of the existing index search tree, a dynamic update propagation tree is constructed. This propagation tree contains only those nodes that are either interested in the index or essential for propagating the updates. By propagating the updates along the tree, the index query cost is reduced and the performance is improved.

2 Dynamic-tree based Update Propagation

The CUP Update Propagation Scheme: When the index is updated by the authority node, such update should be propagated to the nodes that cache the index to reduce the query latency. Roussopoulos and Baker [1] proposed a Controlled Update Propagation (CUP) scheme which actively pushes the updated indices to interested nodes along the index search tree. In this scheme, each node needs to record the interests of its neighboring nodes in the index search tree and push updated index to them when necessary. Based on the benefit and the overhead of pushing the updates, each

*This work was supported in part by the National Science Foundation (CAREER CCR-0092770 and ITR-0219711).

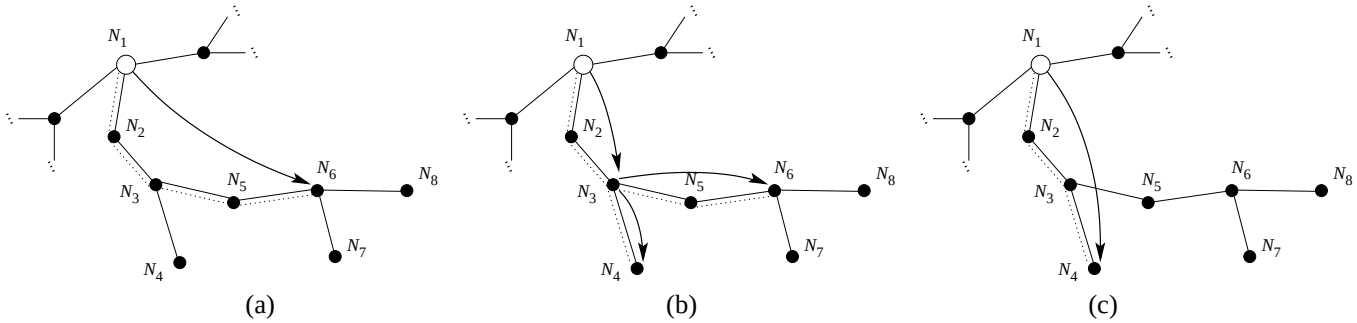

Figure 1. An evolving dynamic update propagation tree. Nodes linked by arrows are in the DUP tree. Nodes linked by dotted lines are in the virtual path.

node determines whether to push the index update further down the tree. As a result, an index is pushed hop-by-hop following the index search tree to reach interested nodes.

Figure 1 (c) can be used as an example to show how CUP works. Node N_1 is the authority node for key K. Suppose N_6 is the only node that is interested in the index update, the index is still pushed through the path $N_2 \rightarrow N_3 \rightarrow N_5 \rightarrow N_6$. If intermediate nodes decide to stop forwarding the index, N_6 will be cut off from the update information. This incurs long delay and high cost when N_6 needs to access the index.

Dynamic-tree based Update Propagation: CUP has performance limitations since it pushes the update along the query path. Intermediate nodes along the path receive the updated index even if they do not need it. To efficiently propagate the index updates, we propose a Dynamic-tree based Update Propagation (DUP) scheme.

The idea of DUP can be explained by Figure 1. Suppose only N_6 is interested in the index. When an update happens in N_1, N_1 pushes it directly to N_6. As the peer-to-peer network is an overlay network on top of the Internet, the physical distance between N_1 and N_6 is not necessarily much longer than that between N_1 and N_2. Such direct push can significantly improve the performance. It only costs one hop to push the update. If the update is not pushed to N_6, it costs eight hops for N_6 to send the request and get the index from N_1 in PCX. Therefore, the cost is reduced by 87.5%. This direct push is illustrated by the solid arrow in Figure 1 (a). The dynamic update propagation tree (DUP tree) contains only N_1 and N_6.

Later, if N_4 is also interested in the index (see Figure 1 (b)), N_1 pushes the index to N_3, the nearest common parent of N_4 and N_6. Then N_3 is in charge of pushing the index to N_4 and N_6. The new DUP tree, which is linked by the solid arrows, contains N_1, N_3, N_4, and N_6. Compared to PCX and CUP, this scheme only costs three hops while PCX costs ten hops and CUP costs five hops to serve N_4's and N_6's queries. Our scheme performs better because it takes short-cuts when pushing the updates. In the worst case when no short-cut is available, our scheme falls back to CUP and still performs well.

If N_6 is no longer interested in the index after it joins the DUP tree, it notifies the upstream nodes in the index search tree. As a result, N_3 stops forwarding the index updates to N_6. Since N_3 has only one child left, it informs the upstream nodes that N_3 no longer needs the update. After the first upstream node in the DUP tree (N_1 in our example) catches this message, it pushes the updates directly to N_4 instead of N_3 (see Figure 1 (c).)

When a node needs to access an index that is not in the local cache, it sends a request to the root through the requester's index search path. Along the path, the first node that has a valid copy of the index serves the query by sending the index along the reverse path. When an update occurs at the root, it pushes the update to its downstream nodes in the DUP tree. Each node that receives the updated index refreshes its cache and repeats the pushing process. Finally, all nodes in the DUP tree receive the updated index and the update propagation ends.

3 Conclusions

Index update propagation in peer-to-peer networks can significantly reduce the query latency and the query cost. In this paper, an update propagation scheme called DUP has been proposed. DUP builds a dynamic update propagation tree on top of the existing index searching structure with very low cost. The tasks of tracking caching nodes and pushing updated are distributed in the network. So DUP is very scalable. By pushing updates along the DUP tree, both the query cost and the query latency can be reduced.

DUP provides a low cost platform to propagate index updates in peer-to-peer networks. The idea of DUP may be applied to more general data dissemination scenarios. We plan to extend DUP to a general data dissemination platform in overlay networks.

References

[1] M. Roussopoulos and M. Baker. CUP: Controlled update propagation in peer-to-peer networks. *Proceedings of the 2003 USENIX Annual Technical Conference*, 2003.

[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM*, 2001.