CTO: Concept Tree Based Semantic Overlay for Pure Peer-to-Peer Information Retrieval

Jianming Lv
Institute of Computing Technology, CAS
No.6 Kexueyuan South Road
Beijing,100080, China
lujianming@software.ict.ac.cn

Xueqi Cheng
Institute of Computing Technology, CAS
No.6 Kexueyuan South Road
Beijing,100080, China
cxq@ict.ac.cn

ABSTRACT

Inspired by how search behavior works in human society, we propose CTO, a self-organized semantic overlay based on concept tree for P2P IR infrastructure, which is efficient for full text search in pure P2P environment without any central control or powerful peer as hub node. Especially, CTO performs very well on searching the unpopular resources shared by a few peers. In our experiment, while searching for the scarce documents shared by the peers, CTO achieves about 80% recall rate when the search covers less than 5% peers in the overlay. The search latency of CTO is also very low, which is controlled in the range about 5~12 hops.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Search process;
C.2.4 [Computer-Communication Networks]: Distributed Systems

General Terms
Algorithms, Performance, Experimentation, Design

Keywords
P2P IR, Concept tree, Peer clustering, Keyword Co-occurrences

1. INTRODUCTION

How to intelligently organize information in intranets of large organizations, and efficiently search for relevant documents, persons, or communities, has become a more and more challenging problem as a result of information explosion. Compared with traditional centralized infrastructures, recently developed P2P IR (Peer-to-Peer IR) methods are more scalable, cost-effective and much easier to deploy and maintain. It is also efficient for searching popular information, but it usually fails to search scarce resources such as documents only shared by a few peers.

Especially, in unstructured P2P overlay, peers are guided to join the network without obeying any strict rules. The topology is loosely constructed and looks much more like random graph. A lot of researches [1][2][3][4][5] have been presented in this field about how to search efficiently in the overly. They can be roughly classified in to 2 main categories: blind search and informed search. In a blind search, peers do not maintain any information about the data location and forward the queries only according to the network topology. The blind search includes the basic flooding algorithm in Gnutella [1], modified random BFS [2], iterative deepening [3], k-walker random walk [4] and so on. On the other hand, in the informed search, peers keep some data location records and some forwarded message history to improve the search efficiency, such as the intelligent search [2], local indices based search [3], routing indices based search [5] and so on. Unstructured P2P is very easy to deploy and maintain. It is also efficient for searching popular information, but it usually fails to search scarce resources such as documents only shared by a few peers.

More recently, some researches [6][7][8] presented hybrid P2P overlay to improve the search efficiency of unstructured P2P. In hybrid P2P, peers are classified into ultra-peers and leaf-peers. Ultra-peers are the powerful ones with higher computing performance and bandwidth available. They act as the proxies of the leaf-peers and guide the overly construction and query message routing. Although hybrid P2P reduces the average search latency of unstructured P2P, it is still not efficient to locate the less popular resource in large scale P2P environment.

The most effective P2P overlay is structured P2P developed in recent years, which is very good at exact keys searching [9][10][11][12][13][14]. Given a target key, the search in structured P2P overlay takes about log (N) hops to locate the destination peers which store the data matching the key. The maintenance of structured network is relatively higher than the others mentioned above, and it increases linearly along with the count of the published keys in the overlay. In the P2P full text search scenario, it needs very high maintenance cost for structured P2P to support keyword-based searching.

Among all of the three overlays above, unstructured P2P is the simplest and easiest model to deploy and maintain in the large scale completely decentralized environment. But it is still a great challenge to locate the less popular resources efficiently in unstructured P2P overlay.

Inspired by how search behavior works in human society, we propose CTO, a self-organized semantic overlay based on concept
2. CTO: ARCHITECTURE

The idea of CTO is inspired by search behavior in human society. For a given question, people do not ask others randomly but make some intelligent decisions. Everybody holds a classification tree of personal knowledge in the mind, and he/she knows which domain the question belongs to. When one wants some information, he/she usually queries someone else who knows better about the corresponding domain.

Similarly, in CTO we build the classification tree of knowledge, the concept tree, which contains all of the possible terms in dictionary as its leaves, and use the non-leaf tree nodes to denote concepts like domains in human society. The Concept tree is illustrated in Figure 1. CTO organizes the overlay according to the concepts owned by each peer, and routes queries to the peers who match them or their upper level medium concepts. We will show the detail in the following sections.

In order to construct the tree, we choose to deploy hierarchical k-means clustering on the terms in dictionary. In this paper, we use the Reuters corpus (Vol.1) including 109,500 news articles to form the dictionary. The descriptor of each term is simply its document vector transformed from its news articles to form the dictionary. The descriptor of each hierarchical k-means clustering on the terms in dictionary. In order to construct the tree, we choose to deploy

2.1 Semantic Overlay

In order to construct the tree, we choose to deploy hierarchical k-means clustering on the terms in dictionary. In this paper, we use the Reuters corpus (Vol.1) including 109,500 news articles to form the dictionary. The descriptor of each term is simply its document vector transformed from its document posting list in the sample corpus, and each item of the vector is binary denoting whether or not the term appears in the corresponding document. By clustering the terms, the concept tree holds the co-occurrences information, in which the terms belong to the same sub tree tend to appear in one document together and express similar topics.

In CTO, every peer holds the same predefined concept tree as a global knowledge to understand what concepts he and his neighbors have and which concepts a coming query belongs to. For clarity, we define $CT$ as the global concept tree, and $d$ as the average number of sub-nodes in $CT$, $L$ as the depth of $CT$, $n_k$ as a concept of level $k$ in the concept tree ($0 \leq k < L$), $Hi$ as the concept collection that peer $i$ has, $Oi$ as the concept collection the peer $i$ does not have.

The overlay of CTO is organized similar with human society. In usual life, people search through friendship, and they prefer to ask those that they are familiar with and are also good at the domain their questions relate to. More people in different fields the one knows, more possibly his/her questions will be answered soon. Similarly, in order to support efficient search in P2P environment, it is necessary to build the ‘friendship’ between peers and choose neighbors for each peer to cover as many concepts as possible. However, in real world, a peer cannot hold too many connections for its capacity limitation. To overcome this restriction, we define the priority of the concepts. The concepts of higher level should be covered with higher priority, which helps the peer to have a broader view of the concept tree and know better how to forward a search. The priority of a level $k$ concept is defined as: $P(S_i) = d^{L-k}$. With the priority definition, we define the weighted coverage of two concept collections as: $Cov(S_1, S_2) = \{ \sum P(t) | t \in S_1 \land t \in S_2 \}$. Then we can define how well a peer $j$ covers another peer $i$’s concept collections as: $Lcov(i, j) = Cov(Hi, Hj)$. Also, we can define how well peer $j$ covers the concepts that peer $i$ has not: $NCov(i, j) = Cov(Oi, Hj)$.

When a new peer $i$ joins the overlay, it randomly selects a peer $j$ in the network as proxy and sends $Hi$ and $Oi$ to it. Received this request, peer $j$ checks its neighbors and picks out the peers with highest $Lcov$ ($ij$) and the ones with highest $NCov$ ($ij$) to form a peers list, and sends the list back to peer $i$. Then peer $j$ forwards $Hi$ and $Oi$ to one of its neighbors, who will do the similar thing and sends back a peers list to peer $i$ too. The iteration runs until the forwarding hops reach a TTL limit.

Received the peers lists, peer $i$ picks out a peer $m$ having highest $LCov$ value with itself, then subtract the concepts covered by peer $m$ from $Hi$, and goes on to pick out another peer with highest $LCov$ from the left ones in the lists. The procedure is looped until the selected peers reach a predefined limit. We take these peers to form the local neighbor table of peer $i$, which hold the neighbors with similar interest with peer $i$. In the same way, we can pick out the peers with highest $NCov$ to form the remote neighbor table containing the peers who feel interested in something different from peer $i$.

Some one may possibly argue that it may spend a lot of bandwidth to send out $Hi$ and $Oi$. This is really a big problem for some weak peers. But it is easy to compress the message according to $CT$. We can use the top $\theta$ level concepts in $Hi$ and $Oi$ to represent them and ignore the lower level ones. In the experiment, $\theta$ is 5, $L$ is 8 and $d$ is 8. Under this setting, the message size of $Hi$ and $Oi$ is about 0.5k, which is acceptable for P2P application.

2.2 Concepts Guided Routing

Upon receiving a query, the peer checks its local inverted index of documents to match it. If matched, it sends back the result to the query originator. Then it selects some neighbors to forward the query until the forwarding hops reach the TTL limit. When doing forwarding, the peer knows the concepts neighbors both in the local neighbor table and remote neighbor table by $Cov(S_n, Hi)$ (where $Hi$ is a neighbor’s concept collection). The peer then forwards the query to the top $K*Cov_{max}$ ranked neighbors where $K$ is a constant and $Cov_{max}$ is the maximum $Cov(S_n, Hi)$ value of all the neighbors.
3. EVALUATION

We evaluate CTO over the Reuters corpus (Vol.1), which is comprised of news articles including text and author fields. We deem each author to be a peer and associate documents written by the author to the corresponding peer. This leads to 109,500 documents shared by 2,368 peers. We compose a query set with 100 queries by randomly choosing 100 documents from the corpus and choosing 3 terms uniformly at random from each document vector to form a query. The answer for a query is comprised of all documents containing all terms in the query. In order to test the searching performance of CTO towards the scarce resources as well as the popular ones, we classify the queries into Q1~Q4 subsets according to their popularity (calculated as how many documents in the corpus match the query). The average popularity of each subset is listed in Table 1. We initiate each query in CTO from randomly chosen 200 peers and calculate the average recall rate which is illustrated in Figure 2.

<table>
<thead>
<tr>
<th>Query Set</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popularity</td>
<td>2.6</td>
<td>22.6</td>
<td>46.8</td>
<td>55.6</td>
</tr>
<tr>
<td>Search Latency (hops)</td>
<td>12.4</td>
<td>8.5</td>
<td>6.1</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Table 1: Query sets with different popularity

In Figure 2, we plot recall vs query processing cost (how many peers covered during searching). CTO with concept guided routing achieves closely average 50% recall rate when covers only 5% peers. From Table 1, we can see that the search latency is also very low. It is interesting to find that CTO is very efficient to locate the less popular resource. For Q1 query set, CTO achieves almost 80% recall rate when only covers 5% peers, and the latency is about 12 hops.

4. CONCLUSIONS

CTO with concept guided routing is efficient for full text content search in pure P2P environment without any central control and powerful peers. Especially, it is quit efficient in searching less popular resources shared by a few peers. In our experiment, while searching for the scarce documents shared by the peers, CTO achieves about 80% recall rate when the search covers less than 5% peers in the overlay. The search latency of CTO is also very low, which is controlled in the range about 5~12 hops. In the future, we will test the scalability of CTO on some larger corpus. Moreover, we will also give some further theoretical analysis about CTO in the following research.

5. ACKNOWLEDGMENTS

This research is supported by the 973 National Basic Research Program of China (2004CB318109& 2007CB311100). We would like to thank the anonymous reviewers for their constructive comments on the earlier version of this paper.

6. REFERENCES


