Caching for Improved Retrieval in Peer-to-Peer Networks

Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, Uwe Thaden

L3S Research Center and University of Hannover
Expo Plaza 1, 30539 Hannover
{balke, nejdl, siberski, thaden}@l3s.de

Abstract: In modern information systems the dominant retrieval paradigms have shifted from exact matching towards retrieving a list of the most relevant objects. This is because users usually are not satisfied by arbitrarily large and unordered answer sets, but rather interested in small sets of ”best” answers. All popular ranking approaches need collection-wide information. This poses a challenge in the P2P context where each peer has only local knowledge. We discuss how to deal with collection-wide information in P2P networks and how to optimize query distribution based on query result analysis, and present evaluation results.

1. Introduction

Peer-to-peer networks have become valuable infrastructures for sharing information in a wide variety of applications. The information shared can range from simple media files annotated (and subsequently retrieved) with meta-data (e.g. simple file sharing applications like mp3 sharing in Napster) to compound text or even multimedia documents, whose content has to be fully indexed. With such more complex documents and full-text or content-based indexes recently also the dominant retrieval paradigms have shifted from simple exact matching of (meta-data) attributes, also called query bindings, towards retrieving the most relevant objects in a ranked retrieval model. For instance techniques imported from IR research like probabilistic or vector space models can help to rank even complex full-texts and allow for cooperative retrieval.

However, with the distributed nature of peer-to-peer networks, complex retrieval processes are hard to facilitate. Standard techniques for ranked retrieval models generally build on centralized servers where efficient technologies (like e.g. inverted file indexes) can be used to improve retrieval performance. Early retrieval in peer-to-peer focussed on the flooding of queries through the network, where a query was forwarded to all adjacent peers within a certain radius of hops. However, since flooding always needs to contact all peers to answer queries correctly, the retrieval is rather inefficient and does not scale, see e.g. [Rit01]. Facilitating advanced retrieval in peer-to-peer networks needs more efficient techniques like distributed hash tables that index semantic information for each peer and forward queries only to those peers, which can contribute to the result set. But also these techniques have drawbacks: the constant updating of centralized or distributed global indexes draws a lot of performance from the network especially in networks with very limited bandwidths or higher volatilities.

The L3S Research Center Hannover is focussed on developing information systems, e-learning and knowledge technologies and innovative concepts and infrastructures for education. Especially its research in Semantic web technologies, peer-to-peer systems and cooperative techniques for information systems and information services promises to open up traditional database collections, digital libraries, and Web-based collections with a maximum of usability. Moreover, also the information service provisioning process is supported by these techniques and the development of adequate business models. In this paper we will show some of these techniques to investigate the problems of designing indexing schemes for structured peer-to-peer environments with a view towards advanced retrieval paradigms. We will consider highly efficient local indexing schemes that nevertheless deliver almost all relevant results. We will also discuss the problem of integrating (and updating) necessary collection-wide information in local query evaluation schemes, like e.g. inverted document frequencies, in local indexes and present first results. And finally we will close with an outlook on open problems on the way towards efficient and semantically meaningful retrieval in peer-to-peer networks.

2. Building Blocks for Efficient Retrieval in P2P Networks

2.1. Network Topologies

Usually a peer-to-peer network is an unstructured set of peers that are connected within a certain network topology. However, in typical real world P2P-networks peers often considerably vary in bandwidth and computing power. Exploiting these different capabilities leads to an efficient network architecture, where a small subset of peers, called super-peers, takes over specific responsibilities and thus create a structure for the network. Super-peer-based networks can provide better scalability than unstructured networks, and are able to support sophisticated indexing and routing [YG03]. Queries can then be forwarded efficiently through a high bandwidth super-peer backbone. Though usually also low bandwidth peers will be connected to that backbone, these peers are never used for forwarding queries or index information, but can fully dedicate their limited capabilities to answer queries. In our network we arrange the super-peers in the HyperCuP topology which is capable of organizing super-peers into a recursive graph structure, e.g. a hypercube. Super-peers form an effective backbone and can join and leave the network with very little overhead. Having N super-peers in a network, the HyperCuP ensures a maximal path length of $\log_2 N$ [SSD+02].

2.2. Distributed Ranking

Indexing schemes in databases usually provide a central index that grows with the collection size. Every time a new object is inserted into or deleted from the database the index has to be updated. Though in principle this approach is also applicable in peer-to-peer systems, in large networks the traffic needed for updating may use up a considerable amount of bandwidth. Hence, for efficiency reasons large peer-to-peer systems need to evaluate query predicates *locally* at each peer. Local rankings can then

provide the desired cooperative retrieval behavior avoiding empty or too large result sets. But as known from e.g. IR applications (e.g. inverted document frequencies), local computations of rankings typically need some collection-wide information at each peer to do globally correct score computations. In the following we will give an overview of a) how to *compute* this information, b) where to *store* it, and c) how to *distribute* it.

One key to success is the observation that in most information sharing applications requests for e.g. Web data are Zipf-distributed. That means only a small number of data is requested often, whereas the majority of data objects are requested rather rarely. Thus, for 'correct enough' query results generally no complete inverted index is needed to process a query, but an adequate caching strategy will pay off in the majority of requests [NST+04]. Considering the collection-wide information we can further observe that complex measures usually can be split into the collection-wide part and a locally computable part. So typically it suffices to globally provide only a very small part of the entire information needed to score objects. For instance for the well-known TFxIDF measure the document frequency for each query term and the document count has to be provided globally, whereas the term frequency can be computed locally.

a) Computation Generally complex measures for scoring documents with respect to a query will contain a locally computable part for which each individual peer will provide its own index. To score objects completely now also the necessary collection-wide information has to be transferred to the local peer. This information is, however, not stored at the local peer (in which case it would have to be updated every time there's a change, even if there are no queries needing this information, and -even worse- the respective peer is not relevant for answering this query anyway meaning that though it might have matching documents, it is not able to contribute to the top-scored documents of the entire network), but attached to the request by the querying super-peer. For the upkeep of correct collection-wide information peers responding to a query generally do not only deliver their matching documents, but also each peer adds its local data necessary to compute (and update) the collection-wide information at the querying super-peer. On the way back to the originating super-peer, this data is aggregated piece by piece. Thus, the originating super-peer gets everything it needs to compute the complete aggregate, and can store the computed result in its collection-wide information index for later use. Based on the origin information of each top-scored document, the super-peer also can create or update the respective entries in its routing index.

b) Storage To store all information needed for retrieval and exploit each peers capabilities, we structure our network by a super-peer-based network approach as described above. Powerful super-peers form a network backbone for routing queries , and each individual content provider peer is directly connected to one of them. To efficiently answer queries we introduce two index types, containing collection-wide information and routing information respectively. Responsibility for the maintenance of these indexes is assigned to the super-peers only. To notice changes in a network's collection with respect to the collections of individual peers, the leaving of peers and the joining of new peers, we use timestamps on all index entries. Depending on the average volatility of the network each index entry expires after a certain time span. Hence (within a certain probability) no outdated information is used for retrieval. The more

stable the information distribution over the network and the less volatile the network (i.e. the joining and leaving of peers), the longer these expiry time spans can be chosen. The exact choice for these parameters however, always depends on the kind of application.

c) Distribution For the distribution we need both of our index types. In our application [NST+04] a query generally consists of a conjunction of keywords that are searched for in a distributed document collection's content. Before distributing it, the querying super-peer generally enhances its requests by adding all necessary collection-wide information from its local collection-wide information index. If a query has been asked before the adequate values are used. If any of the keywords is not yet in the index, an approximation (based on statistical information generated by periodical flooding queries performed in times of low network usage) is provided. With this information each queried peer is enabled to compute scores for its objects locally (we show how to do the complete rankings on top of the scorings in [BNS+05]). To know to which specific peers a query has to be routed, we need our second index type: the routing index. If the routing index already contains a non-expired entry for the query, we send it only to the peers listed. If there is no such entry we have to flood the query, but within this flooding can collect all necessary information to update routing and collection-wide information indexes at the super-peer. Due to the Zipf-distribution of requests in almost all practical applications our indexes will need some time to be built up (i.e. at the beginning almost all queries have to be flooded), but over time can achieve significant savings over the general flooding of queries.

2.3. Evaluation

For our experiments we used the P2P simulation framework described in [ST04]. To get a good impression how our approach woks in practical applications like news dissemination, we used LA Times articles from the TREC-CD 5, which were distributed randomly among the peers. This is obviously the worst case for our approach: when documents are semantically clustered on peers, the number of addressed peers can be essentially reduced (see next subsection). For the queries, we selected terms from the documents randomly, assuming a Zipf-shaped query frequency distribution.

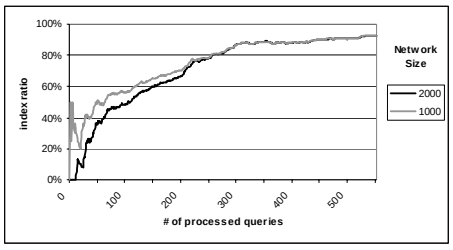

Figure 1: Indexed Queries

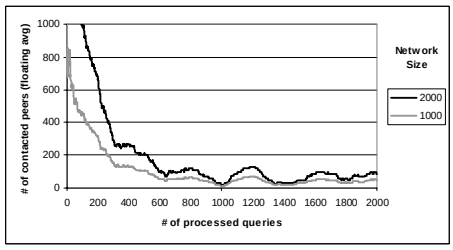

Figure 2: Contacted Peers per Query

Fig. 1 shows how many of the randomly asked queries were already found in the indexes (index hits as floating average with an interval of 200). As stated above the index need

some time for the initial build up, but then the index coverage increases quickly due to the Zipf-distribution of queries. After 550 queries more than 90% of all queries are already found in the indexes. The rapid build up of the index corresponds to a significant reduction of contacted peers (Fig. 2), because after initial flooding of queries a considerable number of queries can then be routed directly to only those peers that have contributed top-scored documents in the recent past. Our indexing scheme thus promises to be applicable for most practical environments and applications. We can thus state that given Zipf-distributed requests and allowing for a limited error-rate of the result set caused by the networks volatility (see [BNS+05] for a detailed investigation of this issue) caching strategies can help to essentially improve the scalability of peer-to-peer applications.

2.4. Extension to Categorized Content

As stated above we ran our performance experiments in a practical environment, though choosing the worst case for caching strategies, i.e. randomly distributed content in a volatile network.. In most real world applications, however, documents show at least some semantical clustering. For instance we can assume that a content provider interested in a certain topic will usually offer some more related documents, e.g. there may be dedicated servers for, say, computer science papers. Often documents within such servers are even already classified according to a given taxonomy. For example, newspaper articles are always arranged based on topics like politics, economics, sports, etc. To further improve our caching strategies we can exploit such a classification by enhancing the query with a category specifying the requested topic.

From the users point of view the category specification can be used as additional information to disambiguate query terms. For our cooperative retrieval paradigm we can use this information (together with a suitable domain-specific ontology) to avoid empty result sets by relaxing query terms in over-specified queries. If a query term does not result in any matches, instead of entirely dropping it, we can generalize it along the ontology and thus get to semantically more relevant results, while avoiding under-specified queries that might lead to the retrieval of too many result objects. Additionally, we can exploit this information to further reduce the number of peers (and thus documents) which must be searched and ranked to answer a query. For this purpose we introduce a additional routing index, for each category containing all peers that can provide documents for this category. Since this information can be extracted from query responses (or initial floodings),too, we can easily upkeep this third index at the super-peers and instantaneously prune large parts of the network for answering queries.

For the cooperative retrieval at each individual peer, first only documents in the specified category are taken into account. If the category was inappropriately chosen by the user (or the collection simply does not contain any or contains too many matching documents, which cannot be known a-priori to the user), this may result in too few or no matches; in this case delivering (additional) matches from similar categories improves user satisfaction. Therefore, we use query relaxation, first to subcategories and then to super-categories. This process is repeated until the peer has a desired number of results

or the taxonomy root is reached as shown in [BNS+05b]. The introduction of the category routing index results in a further significant decrease in the average amount of messages needed for query distribution and thus also improves scalability.

3. Summary and Outlook

In this paper we discussed in brief the impact of local indexing schemes to improve retrieval in peer-to-peer networks. Experiments in practical applications show that a distribution of queries and subsequent local evaluation can essentially increase the retrieval efficiency and thus has great impact on the scalability of retrieval over structured P2P networks. Since the main obstacle for local scorings are complex scoring functions involving collection-wide information, we also discussed the question how to integrate collection-wide information into our local evaluation scheme and showed that by enhancing requests with only the pieces of information needed we can again decrease the necessary communication in the network, in contrast to the traffic caused by the (often unnecessary) upkeeping of central indexes providing such information.

Our future work will focus on more ways to get 'correct enough' estimations of collection-wide information and thus a further improvement of our techniques. We will look into samplings of a representative set of documents to directly derive collection-wide information. We expect sampling techniques to be also effective for finding malicious peers that try to post irrelevant documents with high scorings into result sets. In the long run, our aim is to develop techniques for distributed Web search engines that can integrate the results of local crawlings in a hierarchical fashion into Web searches without the need of huge central indexes like used by today's Web search engines.

4. References

[BNS+05] Balke, W.-T., Nejdl, W.; Siberski, W.; Thaden, U.: Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks. In Proc. of International Conference on Data Engineering (ICDE'05), Tokyo, Japan, 2005.

[BNS+05b] Balke, W.-T., Nejdl, W.; Siberski, W.; Thaden, U.: Here is the News – Distributed Document Retrieval based on Classification and Content. *Submitted for publication.*

[NST+04] Nejdl, W.; Siberski, W.; Thaden, U.; Balke, W.-T.: Top-k Query Evaluation for Schema-based Peer-to-Peer Networks. In Proc. of International Semantic Web Conference (ISWC'04), Hiroshima, Japan, 2004.

[Rit01] Ritter, J.: Why Gnutella Can't Scale. Technical Report Darkridge, Inc., 2001. (www.darkridge.com/\~jpr5/doc/gnutella.html)

[SSD+02] Schlosser, M.;Sintek, M.; Decker, S.; Nejdl, W.: HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks. In Proc. Of International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy, 2002.

[ST04] Siberski, W.; Thaden, U.: A Simulation Framework for Schema-based Query Routing in P2P Networks. In Proc. of International Workshop on Peer-to-Peer Computing & Databases (P2P&DB'04), Heraklion, Greece, 2004.

[YG03] B. Yang, H. Garcia-Molina: Designing a super-peer network. In Proc. of International Conference on Data Engineering (ICDE'03), Bangalore, India, 2003.