Semantic Caching in Schema-Based P2P-Networks

Ingo Brunkhorst[1] and Hadhami Dhraief[2]

[1] L3S Research Center
Expo Plaza 1
D-30539 Hannover, Germany
brunkhorst@l3s.de
[2] Distributed Systems Institute, Knowledge Based Systems, University of Hannover
Appelstrasse 4
D- 30167 Hannover, Germany
dhraief@kbs.uni-hannover.de

Abstract. In this paper, we present the use of semantic caching in the environment of schema-based super-peer networks. Different from traditional caching, semantic caching allows the answering of queries that are not in the cache directly. The challenge of answering the queries using the cache is reduced to the problem of answering queries using materialized views. For this purpose, we implemented the MiniCon-algorithm, which delivers the maximally-contained-rewritings of a posed query based on the stored views. Using simulation and experimental results, we will show the benefit of semantic caching.

1 Introduction

P2P computing provides a very efficient way of storing and accessing distributed resources.

Our goal in the Edutella project [1] is designing and implementing a scalable schema-based P2P infrastructure for the Semantic Web. Edutella relies on the W3C metadata standards RDF[1] and RDF Schema (RDFS) and uses basic P2P primitives provided by Project JXTA[2].

Consider a group of researchers or students sharing metadata for papers they are currently discussing, or learning material they are using. Additionally, metadata repositories from universities and institutes are providing metadata for documents and courses. In this milieu where we deal with a significant number of participant peers and super-peers the network resources become more precious and the avoidance of superfluous queries and messages is crucial.

As a schema-based peer-to-peer network, Edutella constitutes an interesting application area for *semantic caching*. Applying caching techniques reduces the response time of the network and decreases the bandwidth and load for the end-user nodes. Different from traditional caching approaches, the semantic caches

[1] Resource Description Framework: http://www.w3.org/RDF/
[2] Project JXTA(TM) http://www.jxta.org/

G. Moro et al. (Eds.): DBISP2P 2005/2006, LNCS 4125, pp. 179–186, 2007.

we propose have the benefit of being able to answer queries that are not in the cache directly. Our approach tries to rewrite the query so that available entries from the cache can be used to answer the request.

In this paper we will focus on the use of semantic caching in a schema-based system. Answering the queries using the cache is reduced to the problem of answering queries using materialized views. For this purpose, we implemented the MiniCon-algorithm, which delivers the *maximally-contained rewritings* of a given posed query based on the stored views. In the next section, we discuss semantic caching in general and how it compares to other caching strategies, followed by Section 3 in which we explain our approach to using semantic caching in the schema-based Edutella network. Section 4 gives a description of the simulation we used and presents the results. Section 5 concludes with a summary and directions for further work.

2 Related Work

Semantic caching in client-server or multi-database systems has received growing interest recently. Dar et al. proposes in [2] a model for client-side caching and replacement in a client-server database system and show the performance of the semantic caching approach compared to the traditional page caching and tuple caching. They mention the easy extension of their approaches to a multiple-server architecture or P2P network, but do not investigate the specific issues related to P2P networks, such as the placement of the cache data structures or the most appropriate cache replacement policy. Moreover, they only consider selection queries on single relations. Dealing with more complex queries is, however, an important issue in the context of semantic caching, and in our approach, we do consider them. More specifically, we consider scalable algorithms for answering conjunctive queries (even with arithmetic comparisons[3]) using views. Godfrey and Gryz present in [3] a general formal framework for semantic caching. The *key idea of semantic caching* is to remember the queries in addition to the query results. Semantic caching uses dynamically defined groups of tuples. We discern two types of semantic caching [4,2]: *Semantic Query Caching (SQC)* [3] and *Semantic Region Caching* [2]. Semantic caching can be considered as a passive caching strategy in contrast to active caching, a.k.a. *replication*, where the caches are filled using automatically generated queries during the periods of low network load. While replication aims at providing high data availability, semantic caching gives the priority to the optimization of response time.

In the context of P2P systems, semantic caching has so far not been investigated. There is some work on caching in P2P networks, mainly associated with replication and focusing on cache replacement strategies. Kangasharju and Ross present in [5] a set of distributed algorithms for replication and cache replacements policies (e.g. Top-K MFR and Top-K LRU) for P2P caches assuming an underlying DHT infrastructure. They show that Top-K MFR provides near optimal performance. Whether this cache replacement policy performs also in

[3] Not implemented in our current prototype.

schema-based super-peer networks, remains to be investigated. Wierzbicki et al. compare in [6] some conventional cache replacement policies in the case of P2P traffic and show that the LRU policy performs very well. An important additional challenge for P2P networks, and in particular super-peer networks is the placement of caches. While in client-server architectures the caches are placed at the clients, in P2P networks there is a need to define the most appropriate location. In addition, interesting for our implementation are semantic caching approaches investigated in the context of *web caches*. The traditional approach of web caches consists of using dedicated machines for centralized caching of web pages. A decentralized P2P-approach has been shown in SQUIRREL [7]. However, the course of action there does not use semantic caches or investigates the cache management at the peers. Lee and Chu present in [8] caching via query matching techniques for Web databases. They also only consider disjoint conjunctive predicates, the caches are located in the wrappers.

3 Semantic Caching in Super-Peer Networks

Super-peer based P2P infrastructures are usually based on a two-phase routing architecture, which first routes queries in the super-peer backbone, and then distributes them to the peers connected to the super-peers. Our routing mechanism is based on two distributed routing indices storing information to route within the super-peer backbone and between super-peers and their respective peers. Super-peers in Edutella [9] employ routing indices which explicitly acknowledge the semantic heterogeneity of schema-based P2P networks, and therefore include schema information as well as other possible index information. The HyperCuP [10] topology is used to organize the super-peers into a structure that enables efficient and non-redundant query broadcasts. Peers connect to the super-peers in a star-like fashion, providing content and content metadata. The characteristics of the super-peer backbone are exploited for maintaining the routing indices and executing distributed query plans.

3.1 Answering Queries Using Semantic Caches

Semantic caching is simply storing queries and their results. The stored queries and their respective results are called *views*. For our approach, we assume that "only" conjunctive queries are used, which are equivalent to *select-project-join* queries. Conjunctive queries are considered as the most used query class. In order to answer an incoming query from the cache, we have to rewrite it in terms of the stored views. The major obstacle for query rewriting is the *query containment problem*. The problem is known to be NP-complete [11], even for conjunctive queries without built-in atoms. The primary source of complexity is the fact that there are an exponential number of candidate rewritings that need to be considered. However, there are algorithms which efficiently generate maximally-contained rewritings (MCR) of a conjunctive query using a set of conjunctive

views. We decided to use the MiniCon algorithm to answer incoming queries using the cached views and content, since it is one of the established well-scalable algorithms for answering queries using views.

3.2 Cache Management Strategies in Edutella

Cache Locality: In order to place the caches and to implement semantic caching in Edutella, it is expedient to look at the different peer types and their characteristics. We distinguish two types of peers in Edutella: *Super-Peers*, which maintain the backbone structure of the network, they facilitate the best network connections and highest availability, as well as high capacity for processing messages. *Peers* are connected via slow connections and sometimes are only available for short durations only. These nodes are run on end-users' computers. The super-peers seem to be the ideal place for caching, in order to attend an optimal exploitation of caching. Due to their characteristics, the super-peers offer the best requirements for caching. Super-peers are responsible for query routing according to routing indices and query processing as well.

Cache Replacement Strategies: The cache replacement policy defines a strategy for replacement of items when additional space is needed in the cache. The caches that are placed on the super-peers must be maintained according to this strategy. The strategy is used to assign values to the items in the cache, the items with the lowest value are then replaced first. In the traditional systems we distinguish between *temporal locality* [2] and *spatial locality*. Temporal locality expresses the concept that items that have been referenced recently are likely to be referenced again in the near future. A "hard and fast" policy does not exist. Each of the replacement policies has both advantages and disadvantages. The choice of a certain replacement policy depends on the use case. The most known replacement strategies are FIFO - First In First Out, Random, *LRU - Least Recently Used* and LFU - Least Frequently Used. In our approach we only use the LRU-Policy, a strategy that is based on the temporal locality principle, i.e. the items which have not been used for the longest time are replaced.

Cache Misses Issues: If a query cannot be completely answered using the local cache, the query is forwarded to the neighboring super-peers according to the HyperCuP protocol. This technique of handling cache misses is referred to as *faulting*. The query and the obtained results are then cached at all super-peers on the path from the requester to the provider.

3.3 Answering Queries in Edutella Using MiniCon

Queries in Edutella are formulated in QEL[4], a query language based on datalog. The MiniCon algorithm [12] aims at finding the *maximally-contained rewriting* (MCR) of a conjunctive query using a set of conjunctive views. It starts similar to the bucket algorithm [13], i.e. in the first step it maps each query subgoal

[4] QEL Query Exchange Language: `http://edutella.jxta.org/spec/qel.html`

Table 1. Example MiniCon Descriptions (MCD)

V	h	φ	G
$V_1(A, B)$	$A \to A$, $B \to B$	$A \to X$, $B \to Y$	g_1, g_2, g_3
$V_2(C)$	$C \to C$	$X \to C$	g_1,g_2
$V_3(Z)$	$Z \to Z$	$Y \to Z$	g_3
$V_4(D, E, D)$	$D \to D$, $E \to E$, $F \to D$	$X \to D$, $Y \to E$, $X \to F$	g_1, g_2, g_3

to a view subgoal and determines if there is a partial mapping from the query subgoal to the view subgoal. Let's consider the following query:

$Q_3(X, Y) : -\text{hasSubject}(X, \text{“Edutella”}), \text{hasLanguage}(Y, \text{“de”}), \text{hasRating}(X, \text{“good”}),$

which can be separated into the subgoals $g_1(X) : -\text{hasSubject}(X, \text{“Edutella”})$, $g_2(Y) : -\text{hasLanguage}(Y, \text{“de”})$ and $g_3(X) : -\text{hasRating}(X, \text{“good”})$,

and the views:

$$V_1(A, B) : - \text{hasSubject}(A, \text{“Edutella”}), \text{hasLanguage}(B, \text{“de”}), \text{hasRating}(A, \text{“good”})$$
$$V_2(C) : - \text{hasSubject}(C, \text{“Edutella”}), \text{hasRating}(C, \text{“good”})$$
$$V_3(Z) : - \text{hasLanguage}(Z, \text{“de”})$$
$$V_4(D, E, F) : - \text{hasSubject}(D, \text{“Edutella”}), \text{hasLanguage}(E, \text{“de”}), \text{hasRating}(F, \text{“good”})$$

Once the partial mappings are found, the algorithm focuses on variables rather than on subgoals. The subgoal g of a query Q is mapped to a subgoal g_i of a view V according to specific rules (for details see [12]). The set of such query subgoals that have to be mapped to subgoals from one view (and the mapping information) is called a *MiniCon Description* (MCD).

The MCDs (generalized buckets) that only overlap on distinguished view variables are combined. Given a query Q, a set of views V and the set of MCDs C for Q over the views in V, the only combinations of MCDs that result in non-redundant rewritings of Q are of the form $C_1, C_2, ..., C_l$, where

$$\text{Subgoals}(Q) = \text{Goals}(C_1) \cup \text{Goals}(C_2) \cup \ldots$$
$$\forall i \neq j, \quad \text{Goals}(C_i) \cap \text{Goals}(C_j) = \emptyset.$$

Table 1 shows the corresponding MiniCon descriptions for the Query Q_3 and the Views V_1 to V_4 as presented above: The column h is representing a homomorphic mapping of variables from the head, and φ is showing the mapping of the variables from the query to the variables from the view. All combinations of the MCDs where the elements from G contain all the subgoals of the Query Q are valid rewritings of the query, in this example V_1, V_4, and the combination of V_2 and V_3.

4 Simulation

4.1 Experimental Setup

In addition to implementing and testing the Semantic Caching in the Edutella network, we simulated a larger network using a discrete simulation framework [14].

We extended the existing simulation framework [15] with an implementation for a semantic cache based on the MiniCon algorithm. Queries are routed using the HyperCuP protocols. For the simulation of caching strategies we only use a topology consisting of randomly distributed peers and super-peers. In the Edutella network we distinguish between *provider* and *consumer* peers. Provider peers are providing data to the other peers and can be queried for metadata, consumer peers are used to send queries to the network and retrieve the results.

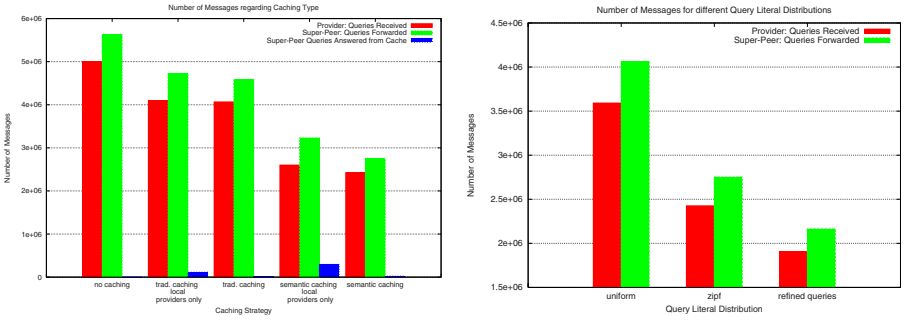
Assumptions: Provider and Consumer peers join and leave the network on a regular basis. Provider peers are considered more stable than consumer peers, and stay in the network for a longer duration. Super-Peers are considered well maintained and highly available, they remain integrated into the topology the whole time. Content is not simulated in the network, instead each provider peer has a fixed probability of 10 percent for generating a response for a received query. We assume that there won't be a fixed schema, only that the set of schema properties and values is identifiable and consists of arbitrary many named elements. Each of the query literals represents a combination of a schema property and a value, e.g. hasSubject(X, "Edutella"). We assume a Zipf-like distribution of schemas because it is a validated law on distributions [15] in many empirical studies and recent research has shown that it holds for the internet as well. Another assumption is that quite often new posed queries are refinements of previous queries, due to the fact that the initial query sometimes doesn't give the wanted results.

Hypothesis: Caching implemented in the super-peer backbone of the P2P network should reduce the amount of messages routed to the data providing peers. With semantic caching the benefit should be larger than with classic caching algorithms, since not only the exact matching cache entries are used to answer the request. The distribution of query literals should also have an effect on the hit/miss ratio of the cache, especially if new queries are refinements of already sent queries.

Experiments: For the experiments we simulated a network of 500 provider, 1000 consumer and 64 static super-peers. The Zipf-distribution used for this simulation consisted of 48 properties and 16 values for constructing query literals. Each query consisted of up to three different literals.

Different caching strategies: The first simulation compares five different caching strategies, including a network without caching. For classic and semantic caching, two different configurations were used. In the first setup only data from providers connected to the corresponding super-peer is cached, messages in the super-peer backbone are transmitted without caching. In the second approach all messages on a super-peer are cached.

Influence of query property/value distribution: To show the influence of the type of distribution of query literals, a simulation was performed with three different configurations. For the first run the properties and values of query literals were uniformly distributed. The second and third simulation uses the same



(a) Caching strategies (b) Query distributions

Fig. 1. Simulation Results

Zipf distribution as in the first experiment. Additionally, in the third simulation a new query is always generated by extending a previously sent query.

4.2 Results

Figure 1a shows the total (accumulated) number of messages in the network after the consumers sent 10000 queries. Every caching strategy reduces the amount of messages transmitted in the system. Without caching the providers alone had to process approx. 5.5×10^6 messages to answer the posed queries. While traditional caching in the super-peers reduces the load for the provider to approx. 4.5×10^6 messages, the use of semantic caching further reduces the amount of messages to 2.7×10^6 (for more details see [16]). Figure 1b shows the results for three different query literal distributions. Semantic caching works best for Zipf distributions, and where new queries are refinements of previously posed queries.

5 Summary and Future Work

Our simulations and experiments have shown, that semantic caching can be used to optimize routing in our schema-based P2P infrastructure. Especially for networks, where a large number of queries is similar, i.e. distributed according to Zipf's law, the semantic caching approach improves the efficiency more then classic caching does. By implementing the MiniCon algorithm we have an efficient way to find the maximally-contained rewriting of a given query using the views contained in the caches. An interesting extension would be a more efficient handling of cache misses using *remainder queries*. Instead of forwarding the query in case answering from the cache is not possible, the query is checked if it can be split (rewritten) in a "cache answerable" and a "cache non-answerable" part. The "cache non-answerable" part, called remainder query is forwarded to the other super-peers in the backbone. If the query can completely be answered from the cache, then we obtain a null remainder query, i.e., no communication with other

super-peers is needed. The advantage of this approach is a lower network load. However, the query rewriting implies an increased complexity.

References

1. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmr, M., Risch, T.: EDUTELLA: A P2P Networking Infrastructure based on RDF. In: Proc. 11th WWW Conference, Hawaii, USA (2002)
2. Dar, S., Franklin, M.J., Jónsson, B., Srivastava, D., Tan, M.: Semantic data caching and replacement. In: Proc. 22th VLDB, Morgan Kaufmann Publishers Inc. (1996) 330–341
3. Godfrey, P., Gryz, J.: Answering queries by semantic caches. In: Proc. 10th DEXA, Florence, Italy (1999)
4. Luo Li, Birgitta König-Ries, N.P., Makki, K.: Strategies for semantic caching. In: Proc. 12th DEXA. Volume 2113 of Lecture Notes in Computer Science., Springer (2001) 99–106
5. Kangasharju, J., Ross, K.W., Turner, D.A.: Adaptive replication and replacement in P2P caches. Technical Report EURECOM+1102, Institut Eurecom, France (2002)
6. Wierzbicki, A., Leibowitz, N., Ripeanu, M., Wozniak, R.: Cache replacement policies revisited: The case of p2p traffic. In: 4th. GP2P Workshop, Chicago, Il. (2004)
7. Iyer, S., Rowstron, A., Druschel, P.: Squirrel: A decentralized peer-to-peer web cache (2002)
8. Lee, D., Chu, W.W.: Towards intelligent semantic caching for web sources. J. Intell. Inf. Syst. **17** (2001) 23–45
9. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A.: Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In: Proc. 12th WWW Conference, Budapest, Hungary (2003)
10. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In: International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy (2002)
11. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal **10** (2001) 270–294
12. Pottinger, R., Levy, A.Y.: A scalable algorithm for answering queries using views. In: VLDB '00: Proc. 26th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (2000) 484–495
13. Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: VLDB '96: Proc. 22th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (1996) 251–262
14. Cowie, J., Liu, H., Liu, J., Nicol, D.M., Ogielski, A.T.: Towards realistic million-node internet simulation. In: PDPTA. (1999) 2129–2135
15. Siberski, W., Thaden, U.: A simulation framework for schema-based query routing in p2p-networks. In: Proc. Workshop on P2P Computing and Databases, EDBT, Heraklion, Greece (2004)
16. Brunkhorst, I., Dhraief, H.: Semantic caching in schema-based p2p-networks. Technical report, L3S Research Center (2005) `http://www.l3s.de/php/publikation.php`.