I.J. Information Technology and Computer Science, **2013, 04, 40-46**
Published Online March 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.04.05

Performance Evaluation of Index Schemes for Semantic Cache

Sheneela Naz
Centre of Research in Networks & Telecom, M. A. Jinnah University Islamabad, Pakistan
[1]shahneela.cs@gmail.com

Muhammad Naeem
Department of Computer Science, M. A. Jinnah University, Islamabad, Pakistan
naeems.naeem@gmail.com

Amir Qayyum
Centre of Research in Networks & Telecom, M. A. Jinnah University, Islamabad, Pakistan
aqayyum@ieee.org

***Abstract*—** In last decade of computing, a growing popularity for semantic cache schemes has been observed. Numerous techniques have been proposed for improved performance of semantic cache. Semantic cache is aimed towards reducing the network traffic load with the ability to address some queries without contacting the server. Semantic cache overcomes the limitations of previous page-caching and tuple-cache techniques. A prime concern of semantic cache includes query processing (query response-time) and cache management. The efficiency of semantic cache can be improved by using semantic indexing sachems. Semantic indexing sachems are segment based and hierarchical semantic indexing scheme. Both these sachems can improve the efficiency of query processing and cache management. In this paper, we have performed the evaluation of index schemes for Semantic Cache through the experimental study. Evaluations of these schemes are available in literature but experimental study is not available. Through this experiment, we have highlighted which technique is most suited in what kind of scenario.

***Index Terms*—** Semantic Cache, Hierarchical Indexing Cache, Client-Server, Query Processing, Semantic Indexing, Semantic Query Processing, Indexing Scheme For Semantic Cache

I. Introduction

Peer-to-peer (P2P) file-sharing system is popular from last few years. It is distributed application architecture, in which peers have equal capabilities and privileged to access and shares the information and resources directly with each other. Peer-to-peer (P2P) file-sharing system is highly utilizing the network bandwidth as compared to other network traffic such as web browsing, e-mail etc. So, the ratio of peer-to-peer network traffic is constantly increased as compared to the total Internet traffic [1], in ISP domain. This throttle P2P file-sharing traffic has negative consequences [2] such as

- Higher chances of congestion because it increases the load at Internet backbone
- Increased cost on Internet Service Providers (ISPs) [3]

A possible solution to all these issues is to use peer-to-peer caching. In peer-to-peer caching, contents are temporarily stored in cache and flowing into an ISP's network. Peer-to-peer caching technique satisfies the user query directly from the cache, if requested contents are available in cache otherwise it pass the request on to a remote peer-to-peer user.

Traditional caching systems are tuple caching and page caching. These caching systems have some disadvantages such as high network communication cost, space overheads and difficult parallelism in query processing [3]. Moreover, semantic information is not stored in cache about cached items.

So the concept of semantic cache overcomes these problems. Semantic cache represents the cached data with semantic descriptions and results of previous queries [4-5]. In semantic cache, when the user query is raised then firstly it is checked with semantic cache. If semantic cache satisfies user query then the communication with the server is not required. If semantic cache partially answered the query then split the user query in probe and remainder query. Probe query is processed at cache and remainder query is processed at the server. So, query processing in this way reduce the network traffic load. The advantage of semantic cache is that it reduces the network traffic, ability to answer some queries without contacting the server while improving the response time [6].

Semantic cache contains dynamically defined tuples. Semantic cache technique can be organized as two ways as illustrated by the Fig. 1. First is cache model and second one is query processing. Cache model can be further categories as semantic query caching and semantic region caching [4-5, 7].

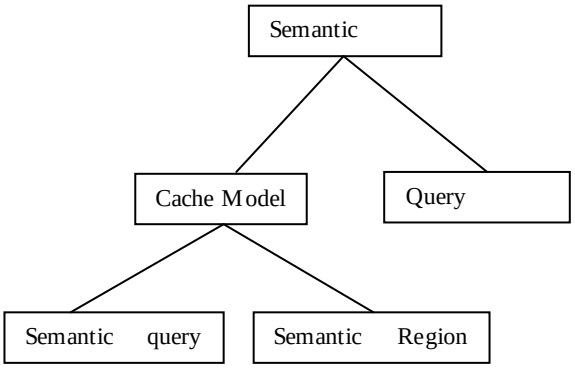

Fig. 1: Semantic Cache Taxonomy

In semantic query caching, use the results of old queries to answer new queries [7-8]. The answers of the queries are stored in relational table and label it with its relevant query. Semantic region is defined as the collection of semantically related tuples. So, semantic region caching is an extension of the semantic query caching with semantic regions [4].

Query processing is to find the relationship between a query and a semantic fragment. There are number of techniques for query processing [3] [5-6] such as LDAP (Lightweight Directory Access Protocol) [9]. The objective of this paper is to perform the evaluation of indexing schemes for semantic cache.

The remaining paper is organized into seven more sections for an easy follow up the study for our intended readers. In section two, we have briefly discussed the relevant techniques found in literature and have also shown; how our approach is novel in comparison to the existing techniques. We have devoted one complete section (three) for semantic cache for our shrewd readers because acquiring a comprehensive knowledge about the topic in discussion is useful for understanding of the general topic of semantic caches and its issues. Moreover, we also devoted another section (four) for matured ideas with hierarchical concepts as this section is also a closely related concept of this study. In section five and six, we come up the experimental detail carried out to validate our proposed idea. Section five is devoted for explanation into performance evaluation while section six describes the experimental model. The last two sections (seven and eight) contain the final conclusion of the whole of the study while we also deliver some insights for future work in this domain of study.

II. Literature Review

Semantic cache is compared with the traditional approaches such as tuple caching and page caching strategies [4], [10]. The results have shown that semantic computing reduces the network load as compared to traditional approaches [4], [10]. Semantic cache has been used in client-server system and peer-to-peer system. In client-server system, there are several techniques proposed by Shaul et al. [4] and Luo Li et al. [11], both of them address the issue of long query response time in context of improving the response time.

Ingo et al. [10] illustrated the use of semantic cache in schema based pee-to-peer infrastructure. They implemented the *MiniCon* algorithm. They presented the efficient mechanism to determine the maximally contained rewriting of a certain query based on the stored views. The simple mechanism answering queries using cache is to replace out with answer queries with materialized view. In this mechanism, a given query is divided into two parts: cache answerable and cache non-answerable. Cache answerable is related to answering the queries from cache and cache non-answerable query is forwarded to backbone peers. If full query is satisfied from cache then it is not required to transfer the query towards backbone peers. In this way, it reduces the communication network traffic, resulting which the prime advantage of this mechanism is to reduce the network traffic load in pee-to-peer network environment but the query rewriting process increase its complexity.

Parke et al. [8] extended the Semantic Query Caching (SQC) [7] technique in two different directions. First, it used the semantic query caching technique into heterogeneous database environments. They argued that traditional page-caching and tuple-caching are not found suitable in heterogonous environment because it does not understand what tuple-based or page-based caching in different databases. Secondly, this work logically enhanced the semantic query caching technique and provides the query optimization technique. This technique provides the following

- *Improvement in overall query response time*
- *Fault tolerance*
- *Answer set pipelining*

Parke et al. [4] proposed the new query optimization framework for heterogeneous environment. This optimization technique used the view for answering queries. The proposed technique is called intentional query optimization (IQO). It handles the complex, expensive queries and query rewriting. Query rewriting technique is used to optimize the query evaluation. Query evaluation cost is reduced by syntactically optimization. Query rewriting techniques utilized the

- *Semantic query caches*
- *Materialized views*
- *Semantic knowledge about the database domain*

This framework introduces the disconnected queries. The evaluation cost of disconnected queries is less than query itself.

Query processing using semantic cache is examined in [3-7], [11]. The first concept of answering queries by semantic cache was introduced in [10] and it defined the relationship between cache and queries. Parke et al. [7] discussed the following problems such as deciding while answers are in cache, extracting answers from cache, accessing semantic overlap / semantic independence and evaluating semantic remainders.

Shaul et al. [12] investigated the client side caching model and replacement in a client-server database system. They identified the essential factors of semantic cache which can enhance or mitigate the performance of semantic cache while comparing the performance of semantic cache approach with the tradition approaches such as page caching and tuple caching. It introduced the concept of query splitting. After splitting, query has two parts probe query and remainder query. Keller et al. [5] also introduced that same idea. However in both of papers, the details of query processing strategies are not defined. It only encompasses the simple selection queries instead of the complex queries on the single relation. Semantic cache information is represented in the form of regions which contains semantically related tuples. Semantic region are dynamically adjusted according to the requirements of current query. The objective of cache replacement policy is to reduce the communication between the client and server. Adaptive replacement policies are maintained for semantic region.

Luo et al. [11] consider two issues about semantic cache.

- Development of caching strategies
- Development of replacement algorithm

In caching strategies Luo et al. [11] cached both projection results of queries and conditional attributes. This caching strategy optimize storage space requirement. For region replacement, it introduced the PRAG (profit based replacement algorithm with aging counter) algorithm. It also considers the simple selection query on single relation rather than complex queries like join query.

Bashir et al. [13-14] defined the 4-Level hierarchical semantic indexing technique for Semantic Cache. This technique is based on semantic storage system. It extracts the semantics of given query and then stored these semantics. In 4-Level hierarchical semantic indexing technique, 4-Level matching is performed. At first two levels Boolean matching is performed and in next two levels approximate matching is performed. Approximate matching is used hashing technique to generate the hashing code of column name and condition. So fast query matching is performed using this technique and it also reduced the query response time.

III. Semantic Cache

Semantic cache is used in distributed environment because it reduced network traffic and improved response time. Semantic cache maintains result of previous queries and their semantic descriptions.

When the user poses the query, query is matched with cache. If whole users query is exist in cache then there is no communication with server. If user query is partially exist in cache then user query is divided into two queries. One is probe query which is satisfied through the cache and second is server.

The relationship of query and cache is determined through these questions such as

- Whether a query can be totally answered
- How much it can be answered
- What data are missing

According to these questions there are number of query cache strategies such as [3]

1. Contained query

2. Horizontal partitioning query

3. Vertical partitioning query

4. Hybrid partitioning query

5. Non-contained query

All these types of queries are defined in Fig. 2.

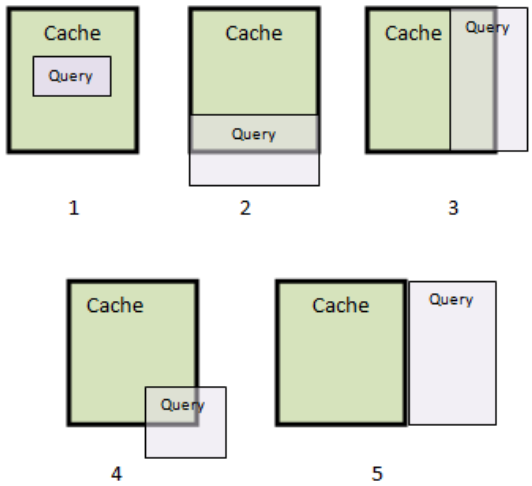

Fig. 2: Relationship between queries and cache

IV. Indexing Schemes

There are two indexing schemes for semantic cache management and query processing strategies. These include semantic segment cache and hierarchical semantic cache. It would be useful if we describe each of them in detail with some example to deliver a relevant set of idea for our proposed technique. These are as below.

4.1 Semantic Segment Cache

Semantic cache is managed as the collection of semantic segments. Semantic segments contain information about multiple tuples. Each segment contents are defined through the constraint formula. Each formula describes the segment location in semantic cache.

Table 1 shows the semantic cache index which contains the number of segments. Column s defined the segment indexes, sname defined the segment name, satt defined segment attributes and spre defined segment selection predicate.

Table 1: Semantic cache index

Table - dbo.cache_index | Summary

s	sname	satt	spre
s1	Mul	act,tme,S,R,tp,pktsize,fid	50>tme<250
s2	h_split	snd,rec,pktid,seqid	150>tme<250
s3	r_cache	act,tme,S,R,tp,pktsize,fid	50>tme<150
s4	v_h_cache	act,tme,S,R,tp,pktsize,fid,snd,rec,pktid,seqid	150>tme<250
s5	cs	act,tme,S,R,tp,pktsize,fid	150>tme<250
NULL	*NULL*	*NULL*	*NULL*

When users pose query, query first compared with cached segments sequentially. If first segment contained the partially results of a query then remainder query is compared with the next segments until the last segment. After the last segment, if there exist the remainder query whose results does not exist in cache then this remainder query is forwarded to server. So, the processing of this scheme is sequentially.

4.2 Hierarchical Semantic Cache

Hierarchical semantic cache is also another type of indexing schemes. In this scheme, conditional query processing is performed rather than sequential processing.

When a query is subjected to a database system, this query is divided into a remainder query and a probe query. The probe query is concerned with the retrieval of data from the cache, where as remainder query is restricted to be sent over the databases to pull out the data which is unavailable in the cache. Here the issue arises how to sustain the organization of the cache as optimized. This issue is handled with dynamic splitting and merging of the relevant semantic regions in perspective of the asked queries.

In cache, query semantics are stored in hierarchical form. When user posed the query then the semantics of user query is matched with the semantics of stored data in cache.

User Query:

$$select\ act, snd, tp, tme\ \ from\ dbl.multicast$$
$$where\ tme > 70\ and\ tme \leq 200;$$

The following comparisons are performed between user query and cache data.

First compare the database name if comparison is true then compare the relation name if relation name is true then compare the attributes and predicate of this relation.

It maintained the top-down hierarchy. At any level if comparison is false then there is no need to process remaining exhaustive processing and then query is forwarded to server.

V. Performance Evaluation

We examined the performance of indexing schemes for Semantic Cache through an experimental study. Index schemes used in this experiment are categorized into segment based and hierarchical indexing based schemes. Semantic cache is managed through the semantic segments which are based on constraint formulas such as (*σ time > 150 AND time < 250 Multicast)*. Our set of experiments was performed by using Microsoft.net in C sharp environment and MS SQL server 2005 in client-server model. In this setup, the following user's queries are executed which are defined in table 2.

Table 2: User's queries

Queries	User Posed Queries
Q1	select Action, S, R, type, ptksize, fid, time, snd, rec, pktid, seqid from SHAHNEELA…aa where time >= 150 and time <= 250;
Q2	select Action, S, R, pktsize, time from SHAHNEELA…aa where time >=150 and time <=250;
Q3	select pktsize, time, snd, rcv from SHAHNEELA…aa where time >=150 and time <=250;
Q4	select type, pktsize, fid, time, snd, rcv, pktid, seqid from SHAHNEELA…aa where time >=75 and time <=150;
Q5	select R, type, pktsize, fid, time, snd, rcv, pktid, seqid from SHAHNEELA…aa where time >=0;
Q6	select S, R, type, pktsize, time from cache_reg where time >=75 and time <=150;
Q7	select Action, S, R, type, pktsize, fid, time, snd, rcv, pktid, seqid from SHAHNEELA…aa where time >=0;

When user's query posed to cache then there exist different cache segment strategies which are maintained in cache according to each query.

Example

Initially when query1is posed from the users query set then there exists three possibilities of segments. Fig. 3 shows the different segment strategies which are maintained in cache. Green boxes show the probe query and gray boxes shows the remainder query.

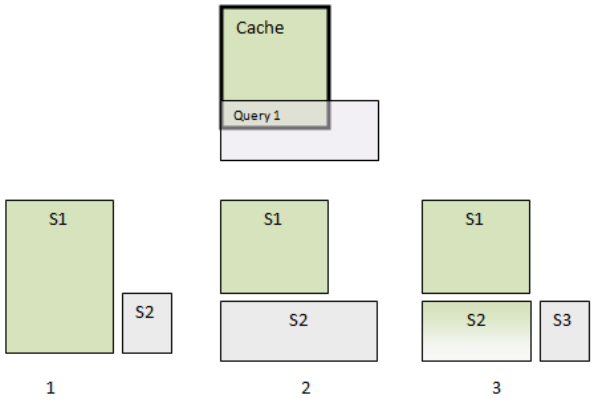

Fig. 3: Different Segment Strategies

After that cache is maintained; which is shown in Fig. 4. After each query execution semantic cache is updated.

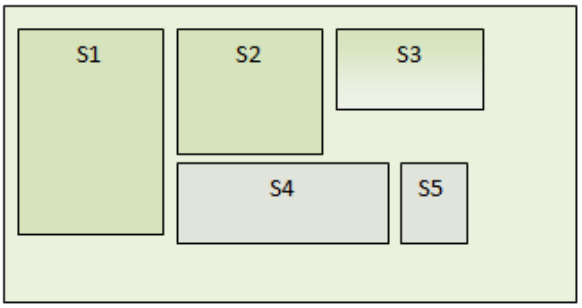

Fig. 4: Semantic Cache with Different Segments

In this section, we shall elaborate the experimental results along with their analysis.

VI. Experimental Model

In this study experimental model is defined as a client, a server and network connection which connects client and server. Database is stored and maintained at server side and it acts as a database server. While client maintains the query set and user queries are posted through client graphical interface window. We exploited the linked server feature of SQL Server to connect to the remote server running on a wired network. In order to simulate the real network traffic, the client machine was on a wireless network while the remote server was harbored on wired network. This makes sense of involving and exploiting all features of real network including congestion, packet delay/drop, variable size of TCP window, IP routing, switching etc. Table 3 illustrates the result obtained from the experiment performed on semantic cache indexing. A careful examination of the result shown by table 2 indicates that as the number of regions is increasing for the complex queries, the difference in response time for input queries is significantly increasing. This highlights the fact that as the regions serve as local cache, they are a useful depository to prevent the input query to invoke at the remote server. This revelation is quite natural but the interesting and important catch in this result set is that as the number of cache is increased the processing time for splitting the input query into region and remote queries is not increasing abruptly? In fact, we examined that for up to large number of regions, the processing time for splitting up the input query into local, remote server and mix query remain almost constant. The explanation of this lies in the fact that the processing time remains quite unchangeable for a very large number of regions. We have examined the processing time for one thousand regions and this was not increased more than 2%.

In hierarchical indexing schemes, initial comparison is very efficient at first two levels in comparison to segment indexing schemes. However in remaining two levels which include attribute comparison and predicate comparison suffer from an exhaustive search. If compared match is found then it yields reduced response time as compared to segment indexing schemes. If it does not exist in cache then query is transmitted to server for execution. This results in consumption of the same processing time.

Table 3: Results semantic segment cache index

Scenario	Queries	Regions	Runtime (millsec)		
			Local	Remote	Mix
1	7	3	30	310	70
	10	5	50	512	90
	12	7	68	745	114
2	7	3	34	230	69
	10	5	57	558	99
	12	7	67	881	121

VII. Discussion

In other words, if we compared the response time of both indexing schemes; say that segment indexing scheme take 100 millisecond query execution time with in cache searching and hierarchical indexing scheme consume approximately 40 millisecond query execution time with in cache searching. So the response time in hierarchical indexing scheme is less than as compared to segment indexing scheme.

The extension to our performance evaluation is related to the implementation of the graph based indexing scheme. Such scheme theoretically is believed to be superior in performance as argued by Ahmad et al. [14]. However there is a direct need to perform experiment to critically evaluate at which number of regions the real gain of graph based indexing scheme emanate.

VIII. Conclusion

Performance of segment index scheme and hierarchical indexing scheme was analyzed with respect to query processing response time. The results of this experimental study show that the performance of hierarchical indexing scheme is superior as compared to segment index scheme; but only in those cases where the number of regions exceeds. However in cases of small number of regions, segment index scheme performs well. In future, we will be further improve our heuristics by supporting runtime scheduling and also considering the end user's QOS (quality of service) for multiple users.

References

[1] Hamada, T., Chujo, K., Chujo, T., and Yang, X. (2004). Peer-to-peer traffic in metro networks: analysis, modeling and policies [C]. IEEE/IFIP Network Operations & Management Symposium (NOMS 2004).

[2] Saleh O, Hefeeda M. Modeling and caching of peer-to-peer traffic [C]. in Proc. of ICNP'06, Santa Barbara, CA, Nov. 2006.

[3] Qun Ren, Margaret H. Dunham, and Vijay Kumar, Semantic Caching and Query Processing [J]. IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No.1, 2003.

[4] Shaul Dar, Michael J. Franklin, Björn Thór Jónsson, Divesh Srivastava, Michael Tan, Semantic Data Caching and Replacement [C]. In Proc. of: Intl. Conf. on Very Large Databases (VLDB), Bombay, India, pages 330-341, September 1886.

[5] Keller A M, Basu J. A Predicate-Based Caching Scheme for Client-Server Database Architectures [J]. VLDB J., vol. 5, no. 2, pp. 35-47, Apr. 1886.

[6] Kami Makki, Matthew Rockey, Query Visualization for Query Trimming in Semantic Caching [C]. 209 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, 2010.

[7] Godfrey, P., and Gryz, J,. Answering queries by semantic caches [C]. In Proceedings of the 9th DEXA (Florence, Italy, 1888).

[8] Parke Godfrey and Jarek Gryz, Semantic Query Caching for Heterogeneous Databases [C]. In Proceedings of the 4th KRDB Workshop , Athens, Greece, August 1887, pp. 6.1-6.6.

[9] Cluet S, Kapitskaia O, Srivastava D. Using LDAP Directory Caches [C]. Proc. Symp. Principles of Database Systems, 1999.

[10] Brunkhorst I, Dhraief H. Semantic Caching in Schema-Based P2PNetworks [C]. In Proceedings of the Third International Workshop on Databases, Information Systems and Peer-to-Peer Computing, 2005.

[11] Luo Li, Birgitta K ̈onig-Ries, N. P., and Makki, K., Strategies for semantic caching [C]. In DEXA '01: Proceedings of the 11th International Conference on Database and Expert Systems Applications (2001), vol. 2103 of Lecture Notes in Computer Science, Springer, pp. 88–96.

[12] Karagiannis T, Rodriguez P, Papagiannaki K. Should internet service providers fear peer-assisted content distribution? [C]. In Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05), Berkeley, CA, USA, Oct. 2005, pp. 63–76.

[13] Bashir M F, Qadir M A. HiSIS: 4–Level Hierarchical Semantic Indexing for Efficient Content Matching over Semantic Cache [C]. INMIC, IEEE, Islamabad, Pakistan, pp. 211-214,2006.

[14] Ahmad M, Qadir M A. Sanaullah, M., An Efficient Query Matching Algorithm for Relational Data Semantic Cache [C]. 2nd IEEE conference on computer, control and communication, IC409, 2009.

Authors' Profiles

Shahneela Naz: Research scholar at department of computer science, M. A. Jinnah University Islamabad Pakistan. Her research area includes Clustering Classification, Semantic engineering, Text retrieval, Adaptive Real-time Video Multicast Techniques, Layered Video Multicast, Multimedia protocols, Routing and Multimedia Streaming.

Muhammad Naeem: Research scholar at department of computer science, M. A. Jinnah University Islamabad Pakistan. His research area is machine learning, semantic computing, text retrieval and data mining.

Amir Qayyum: received his Bachelors in Electrical Engineering from U.E.T., Lahore, Pakistan in 1991,then M.S. in Computer Engineering from E.S.I.M., France in 1995, then D.E.A. from University of Paris-Sud, France in 1996 and then Doctorate from University of Paris-Sud, France in 2000. He is currently a professor at M. A. Jinnah University, Islamabad and also the head of the Center of Research in Networks and Telecommunication (CoReNeT). His current research interests include mobility management framework for mobile devices, QoS and scalable multicast in IPv6 networks, MANETs and VANETs, VoIP and network security. Currently he is Dean of Faculty of Engineering at Mohammad Ali Jinnah University Islamabad Pakistan

How to cite this paper: Sheneela Naz, Muhammad Naeem, Amir Qayyum,"Performance Evaluation of Index Schemes for Semantic Cache", International Journal of Information Technology and Computer Science(IJITCS), vol.5, no.4, pp.40-46, 2013.DOI: 10.5815/ijitcs.2013.04.05

