OPTIMALITY ANALYSIS OF QUERY PROCESSING USING VARIOUS CACHING APPROACHES - A REVIEW
P.Mohankumar*1, Balamurugan2

1,2Department of Information Technology, VIT University Vellore, Tamilnadu, India.
Email: mkr_karai@yahoo.co.in

Received on 22-04-2016
Accepted on 18-05-2016

Abstract

Any sort of Internet and database application efficiency depends on good query base. In order to improve processing performance the cache utilization play a vital role by acting as auxiliary unit and provide associate services to processor. Various existing techniques about cache optimal utilization for the past two to three decades discussed about only the indexing, architectural view, address mapping mechanism, few concentrated cache only on needy base, In general as information retrieval among user applications as sample pharmacy or hospital relevant data in timely need plays high importance. Considering this aspect in order to find a way to improve efficiency among query processing in this paper a detailed survey and investigation was made about the cache utilization in query processing at various levels and come out with the resultant as how the semantic cache enhanced as CoopSc( cooperative semantic database architecture) and how it can be collaborated in a cooperative way intelligently in large scale applications over p2p, distributed and cloud environments in an optimal way.

Keywords: Semantic cache, Cooperative, Intelligence, Optimization, Pharmacy.

Introduction:

Any sort of Internet and database application efficiency depends on good query base. In general as information retrieval among user applications as sample pharmacy or hospital relevant data in timely need plays high importance. In order to improve processing performance the cache utilization play a vital role by acting as auxiliary unit and provide associate services to processor Caching is extremely important to improve performances in many computing systems. The main ideas of the caching proposition relies on the use of partially pre-calculated queries handled by cache services and on the definition of a logical network of cache services which can cooperate together to enhance data access. The idea of query
Caching is to reuse computed results of previously asked queries in further query processing. The performance of the caching strategies proposed so far has been studied by means of extensive experimentation using queries submitted by actual users of search engines. Few on mathematical analyses are also not effective because of the strong dependency on user behavior. Further studies have focused on cache hits rather than on query throughput. But we limit to this ancestor issue, as basic criteria and provide road map by concentrating from semantic caching onwards towards intelligence techniques and how several caches utilized in parallel as well as cooperative manner during execution of the query to improve the performance, minimize response time as well as providing user convenience. Semantic caching is a dynamic caching strategy which deals with not only exact but also inexact similar queries, as the local cache has the capability of analyzing the incoming queries. Basic caching mechanisms is tabulated above in table 1 which is detailed in as well as shown how these can be used for query processing and extended with heterogeneous queries as metasearching approach. Caching is not a performance panacea, however to incorporate caching, the DBMS must include a cache consistency protocol. The cache consistency protocol might actually increase the load on the server and/or the client due to its overhead and potential pitfall, which depends on the concurrency control scheme used, is the late discovery of data conflicts. Thus, the potential consequences of adding caching to a client-server DBMS deserve investigation stated in and presented a range of lock based cache consistency algorithms that result from recognizing that cache consistency is simply a variant of the replicated data management problem. Firstly we see the architectural issues.

Table 1: Various caching mechanism.

<table>
<thead>
<tr>
<th>Caching mechanism</th>
<th>Role</th>
<th>Pros and cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page caching</td>
<td>Individual relevant pages are cached.</td>
<td>Widely used in database applications. lead to long delays in the answer delivery</td>
</tr>
<tr>
<td>Tuple caching</td>
<td>Individual relevant tuples are cached</td>
<td>High level flexibility</td>
</tr>
<tr>
<td>Proxy caching</td>
<td>Maintains the set of recently accessed Web pages.</td>
<td>Make use of tuple caching and reuse a page from cache each time its URL is asked by a client.</td>
</tr>
<tr>
<td>Semantic caching</td>
<td>Cache as a collection of semantic regions.</td>
<td>Overcome the drawbacks of page and tuple caching.</td>
</tr>
<tr>
<td>Multi-source cache</td>
<td>Caching results from multiple Web sources.</td>
<td>Better overall performance. deal with structural and semantically heterogeneity inside a region</td>
</tr>
<tr>
<td>Reference point caching</td>
<td>Caching IP addresses to avoid DNS lookups at clients, and caching information about documents to avoid setting up new connections.</td>
<td>Search engines that return IP addresses to speed up search sessions, and caching at regional information servers that goes beyond the capabilities of today’s Proxy caching.</td>
</tr>
</tbody>
</table>
Architectural Issues

Resolution of various issue related to cooperate caching:

A key to achieving high performance and scalability in client-server database systems is to effectively utilize the computational and storage resources of the client machines. So data shipping concept arise. In a data-shipping architecture, query processing is performed largely at the clients, and copies of data are brought on-demand from servers to be processed at the clients. When caching is incorporated into a data-shipping architecture, servers are used primarily to service cache misses, and thus, client-server interaction is typically fault-driven. That is, clients request specific data items from the server when such items cannot be located in the local cache. An alternative method on using associative access, data items are not specified directly, but are selected and grouped dynamically based on their data values. The suggested associative accesses forego the data-shipping architecture in favor of a query-shipping approach, where requests are sent from clients to servers using a higher-level query specification. But not preferred mostly in Client server applications. As extension Semantic caching is a technique that integrates support for associative access into an architecture based on data-shipping. There are three main factors that impact the relative performance of the architectures they are data granularity, remainder queries vs. faulting, and cache replacement policy which is detailed in. Semantic caching exploits the semantic locality of the queries by caching a set of semantically associated results, instead of are used in conventional caching. The semantic caching can be tuples or pages which particularly effective in improving performance when a series of semantically associated queries are asked. Thus, the results may likely overlap or contain one another. Most semantic caching schemes in client-server architectures are based on the assumption that all participating components are full-edged database systems. Further most of the semantic caching is based on the following key ideas since querying capabilities of web sources are weaker than those of queries from end users, query translation and capability mapping are necessary in semantic caching. With an efficient method to locate the best matched query from the set of candidates, semantic caching for web sources can significantly improve system performance. Semantic knowledge can be used to transform a cache miss in a conventional caching to a cache hit. As extension existing semantic caching schemes cannot be directly applied. A method to use semantic knowledge, acquired from the data, to avoid unnecessary access to web sources by transforming the cache miss to the cache hit is presented. An intelligent semantic caching scheme suitable for web sources is presented which takes care of the difference between...
the query capabilities of an end user system and web sources. This focused way to use knowledge for providing cooperative capabilities such as conceptual and approximate web query answering, knowledge-based semantic caching, and web triggering with fuzzy threshold conditions. As resultant CoWeb (Cooperative Web Database)[4] arisen. Alternatively another as, clients are allowed to share cached query results in a cooperative matter, named CoopSC (Cooperative Semantic Cache[10] which allows clients to register queries for which they have cached results and also to search for queries stored in the collaborative cache that subsume or overlap new queries for which they want the result. Similar CoopNet[8], a cooperative network architecture, where clients cooperate in order to improve the overall network performance. It is described how CoopNet is used for solving Web flash crow scalability problems. We analyze all pros and cons existing in these systems and propose resultant a new model towards intelligence techniques and strive for user convenience as overall performance.

Table 2: Cooperative Semantic Caching Approach.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Data Model</th>
<th>Query Types</th>
<th>Cache Hit Types</th>
<th>Resolution Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wigan</td>
<td>Relational</td>
<td>Range selections</td>
<td>Strict</td>
<td>Centralized tracker</td>
</tr>
<tr>
<td>XPath Index Cache</td>
<td>XML</td>
<td>XPath (no predicates)</td>
<td>Strict</td>
<td>Distributed index</td>
</tr>
<tr>
<td>Adaptive Caching Service</td>
<td>Gedeon</td>
<td>Point queries</td>
<td>Strict</td>
<td>Flooding</td>
</tr>
<tr>
<td>Coop Semantic Cache</td>
<td>Relational</td>
<td>Range selections</td>
<td>Strict, Partial</td>
<td>Distributed index</td>
</tr>
</tbody>
</table>

Cache Replacement Issues in a Cooperative approach:

Caching is a technique used to store popular documents closer to the user. A lot of studies are focus on developing a better caching algorithm to improve the choice of item to replace, and simultaneously, building up techniques to model access behavior, prefetching data and produced different caching policies to optimize a specific performance and to automate policy parameter tuning. Caching is effectively for data with infrequent changes. Static Caching stores the content of a webpage which does not change and no need to request the same information repeatedly. This supports congestion control. Dynamic caching which will store the updated version. These two technologies provide solution for diminishing latency and congestions. Prefetching is an intelligent the performance enhancement of mobile web caching based on the client behavior and the event specification is explained.[11] Cache replacement policy plays a significant role
in response time reduction by selecting suitable subset of items for eviction from the cache. Here we highlight this issue relevant to Cooperative caching as well its extension as cooperative semantic caching. Cache replacement algorithm greatly improves the effectiveness of the cache by selecting suitable subset of data for caching. Various algorithms and its issues were tabulated in below. All the specified algorithms are detailed \(^1\). Most of the replacement algorithms mentioned above uses cache hit ratio as the performance metric. In cooperative caching coordinated cache replacement is more effective than local replacement since the replacement decision is made by considering, apart from these various other parameters is concerned with respect to performance behalf based on cache replacement such as Cache admission control – Value based admission control can be incorporated to minimize the number of replacements. Cache replacement based on Quality of Service (QOS) parameters can be explored. The data items that have their Time to Live (TTL) expired can be removed as the data becomes stale and cannot be used. So periodical checks can be done to delete the data items with TTL expired.

Table 3: Comparison of cache replacement policies for cooperative caching.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters Considered</th>
<th>Eviction</th>
<th>Performance measure</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>Last access time</td>
<td>Least access time</td>
<td>Cache hit Ratio</td>
<td>Simple to implement temporal based</td>
<td>Non uniformity in data is not considered</td>
</tr>
<tr>
<td>LRU Min</td>
<td>Last Access time, size</td>
<td>Lowest access time, time, size greater than incoming data</td>
<td>Average query delay</td>
<td>Cache hit ratio of smaller sized objects is increased.</td>
<td>Not suitable for large caches.</td>
</tr>
<tr>
<td>SXO</td>
<td>Size and access frequency</td>
<td>Largest size, low access frequency</td>
<td>Cache hit ratio</td>
<td>Parameters used are easily available.</td>
<td>Recency of data is not considered</td>
</tr>
<tr>
<td>LUV</td>
<td>Access probability, size, coherence and distance.</td>
<td>Low access probability, bigger size, low consistency, lowest distance</td>
<td>Byte hit ratio, average query latency, message overhead</td>
<td>Considers all the temporal data.</td>
<td>Distance parameter may vary and is not considered.</td>
</tr>
<tr>
<td>TDS</td>
<td>Distance and access frequency</td>
<td>Low access rate and lowest distance, Low access rate and lowest distance</td>
<td>Success rate, cache hit ratio</td>
<td>Value of distance is updated.</td>
<td>Doesn’t consider recency of data item.</td>
</tr>
</tbody>
</table>
Distributed Index Issues:

Figure1  Distributed hash table form.

This section describes the distributed structure that is used for indexing semantic regions. DHT provides the information look up service for P2P applications. Moreover it plays vital role in our case not only similar in whenever the system need to scale, system have heterogeneous nodes, system need self-organization, nodes fail often, the economies of scale favor decentralization, system tolerate security risks due to decentralization, need content addressability. Thus we give importance to DHTS.DHTs provide flat-application independent naming, many apps/services can co-exist on one DHT. Caching results of previous queries can substantially reduce the communication and computational cost of processing them in a distributing setting. Caching here use the query descriptor i.e. semantic information related to query data. All details discussed in the above session here we highlight about how the semantics is defined and how it is shared during execution in a peer to peer client server based environment cooperatively. In 20 suggest two ways of sharing cache content in terms of the degree of cooperation among the participating peers. One loosely-coupled approach (Index Cache), in which each peer locally caches the results of its own queries. A distributed index is built on top of these local caches to facilitate sharing. Each query consults the index to locate any peer whose cached results may be used in answering it. The other tightly-coupled approach, (Data Cache), where each peer is assigned a particular part of the
cache data space. In the Index Cache, the overall content of the cooperative cache is influenced by the local workload of each peer, in the Data Cache; the overall content is affected by the aggregated workload over all peers. Both the Data Cache and the Index Cache are semantic caches so indexing must be based on the queries that describe their content. Preferably prefix based index that takes advantage of queries. To solve the problem of distributing a data access structure recently a variety of approaches have been developed, that all try to achieve the same goal, namely performing searches not only with low latency but also by consuming only little network bandwidth. But we concentrate on cost base issue as how to locate and transfer data among the peers forming the cooperative cache instead of locating and transferring data from remote data sources. To quantify ,the ratio \( Q = \frac{BC}{BI} \), where BC is the amount of data transferred for answering queries from the cooperative cache and BI the corresponding amount of data transferred when the same queries are answered from remote hosts. Some of the p2p comparisons is given in below table which is detailed in.\(^{21}\)

Peers share information according the semantic proximity between peers and between shared files to self-organism into clustered groups. Traffic Balancing on Co-Operative Proxy Caching for Peer to Peer Networks in \(^{22}\) using cooperation among caches that belong to different autonomous systems Independent caches are handled (ASs), and cooperation among caches deployed within the same AS and proved Significant improvement in byte hit rate can achieved using cooperative caching, Architecture with Protection Mechanism Simple object replacement policies are sufficient to, achieve gain.

**Table 4: Comparison of P2P solutions.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Search Paradigm</th>
<th>Overlay maintenance costs</th>
<th>Search Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gnutella</td>
<td>Breadth-first on search graph</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Chord</td>
<td>Implicit binary search trees</td>
<td>O(log n)</td>
<td>O(log n)</td>
</tr>
<tr>
<td>CAN</td>
<td>d-dimensional space</td>
<td>O(d)</td>
<td>O(d ( n^{1/d} ))</td>
</tr>
</tbody>
</table>

The overhead imposed by cooperative caching comparison is negligible It proves that it is cost effect with improved performance and mainly focuses in congestion control. Various issues related to cooperative database such as presuppositions, misconceptions, in tensional query answering, user modeling, query relaxation, and associative query answering were discussed and a new approach for indexing a structured approach to query relaxation via Type
Abstraction Hierarchy (TAH) and a case-based reasoning approach to provide associative query answering is provided in.\textsuperscript{23}. TAHs are user and context-sensitive and can be generated automatically from data sources for both numerical and non-numerical attributes. Therefore such an approach for query relaxation can scale to large database systems. Further they developed mapping set of cooperative operators for relaxation and relaxation control was presented in which these operators were extended to SQL to form a cooperative SQL (CoSQL). Information about documents to avoid setting up new connections. Enable new services such as search engines that return IP addresses to speed up search sessions, and caching at regional information servers that goes beyond the capabilities of today’s proxy caching is detailed in.\textsuperscript{6}. In\textsuperscript{24} an architecture for the efficient delivery of documents in a distributed system that integrates semantic caching and cooperation mechanisms in a consistent way. To reduce web access latencies, a new paradigm for caching at the reference point of a document. Caching IP addresses to avoid DNS lookups at clients, and caching monitoring of previous user requests, on a limited knowledge of the semantic properties of the documents in the form of keywords, and on a knowledge of the semantic links between some of the keywords further they found out the three potential factors of improvement of the integrated system (local proxy cooperation, intermediate proxy cooperation, semantic caching) do indeed all have a positive impact on the efficiency of requests processing. Three caching strategies based in which clients cooperatively cache broadcast data items which shorten the average response time for data access by replacing cached items based on their access frequencies, the network topology, and the time remaining until each item is broadcast next so called push based in detailed in.\textsuperscript{25} A key advantage of the “push-based” delivery mechanism is its higher throughput for data access in distributed systems with a large number of clients, since the absence of communication contention among the clients requesting data means that they can efficiently share the bandwidth.

**Query Processing Issues:**

Query processing is the core activity for any application in database system. Whatever may be the issues discussed and solutions provided all cover the part of the applications only. Here we concentrate the major issue that concerned with the query related such as query definition, semantic mapping, and query execution and how its relevance is indexed and accessed prior and post execution and gets integrated as complete query. Particularly to answer queries using cooperative semantic cache scheme in client server computing environments based on cooperative semantic caching.
Caching keeps previous query results in client’s local semantic cache is that the mobile clients maintain both the semantic descriptions and associated answers of previous queries in the cache. If a client’s cache can answer anew query, it needn’t communicating with the server. If the query can only be partially answered, this query should be trimmed into two parts: probe query and remainder query. memory, and enables client to use it again in the future accesses (without communications with the server), with which can improve system performance, reduce communication quantity and support the frequent disconnected computing environment. In 26 the query definition is given and inverted index is proposed to store the semantic prefix. Relevance during execution they use the ranking mechanism to estimate the exactness of the query. A query \( q = t_0; t_1, t_d, 1g \) is a set of terms (words). In caching problems for long sequence of queries \( Q = q_0; q_{1n}; \) where \( q_i = t_i; 0; t_1; \ldots; t_i \) An inverted index \( I \) for the collection consists of a set of inverted lists \( Iw_0; Iw_1; \ldots; Iw_1 \) where list \( Iw \) contains a posting for each occurrence of word \( w \). Each posting contains the ID of the document where the word occurs, the (byte or approximate) position within the document, and possibly information about the context (in a title, in large or bold font, in an anchor text) in which the word occurs. Formally, a ranking function is a function \( F \) that, given a query \( q = t_0; t_1 \) to \( t_d1g \), assigns to each document \( D \) a score \( F(D; q) \). The system then returns the k documents with the highest score. One popular class of ranking functions is the cosine measure. The problem of semantic queries in general is viewed How-To-Ask and the How-To-Answer problems respectively. The parameters the type of relation; the ontological distance; and the lexical distance are considered for semantic identification and given definition for semantic query and appropriate answer as by 27 Semantic Query.

A semantic query \( Q \) is a 5-tuple \( \{S, m, sr, o, ld\} \) where:– \( S \) is a structure; \( m \) is a node in \( S; \)– \( sr < \) is a semantic relation; \( o \) is the ontological distance; \( l \) is the lexical distance: Semantically Appropriate Answer. Let \( M \) be a mapping between a source structure \( SA \) and a target structure \( SB \), and let \( Q \) be a query. The semantically appropriate answer to \( Q \) is the set of nodes \( n \) 2 \( SB \) such that \( n \) is related to \( m \) through the mapping \( r_M, \) for some values of \( id \). Furthermore, \( n \) must be at the appropriate ontological and lexical distance from \( m \). As stated in 4 a semantic query matching technique is proposed as efficient and effective way to resolve semantic matching approach. The matching such as exact, partial, best containing match and best-contained match which use semantic knowledge base approach to process the query. Every approach strive for various parameters to reach the level of satisfiability.
Discussions:

The goal of this paper is to bring out the enhancement of general existing semantic caching mechanism by enabling clients to share their local semantic caches in a cooperative matter. That is how the semantic cache as CoopSc (cooperative semantic cache) can be collaborated in a cooperative way intelligently in large scale such as p2p, distributed and cloud environments in an optimal way in order to process user queries. Here we dig from root to leaves and suggested a fruitful way for user utilization. The basic issue from cache utilization in database query execution from client-server architecture to semantic caching and further as cooperative semantic caching in p2p and cloud level existing pros and cons is analyzed and specified. All the caching techniques, cache replacement policy algorithmic approaches with respect to performance measure are tabled. Query formulation mechanisms, such as how to ask and how to answer approaches answering query using semantic cache and cooperative semantic cache is identified and specified. How to access the distributed clients for fetching the current query relevance and how integrate it as a single query output after execution as well how to update the obtained output with already existing result in case of remote and remainder queries also identified. When to use the distributed index and how its hashed using what values also specified.

We come with various pros and cons of the existing system and proposed a problem of identifying why can’t the cooperative semantic cache for server also instead clients alone and why not with heterogeneous database and why not intelligence inference for assisting and assuring user query in a convenient way with conclusion. Further coopsem not only for p2p environment but also for web and cloud environments with TAH query relaxation using multi dimension and multimedia applications presently as Cobase with extension of ubiquitous and recommendation technologies as future research work.

Conclusion:

This paper basically discuss with the semantic caching issues utilized in database query processing and with extension by enabling clients to share their local semantic caches in a cooperative matter. That is how the semantic cache as CoopSc (cooperative semantic cache) can be collaborated in a cooperative way intelligently in large scale such as p2p, distributed and cloud environments in an optimal way in order to process user queries. The new approach to answer queries using cooperative semantic cache mechanism in computing environments which reduces query response time and increase throughput of database management server, because server will only receive queries that cache couldn’t
answer. The complete survey is made regarding the extension of semantic cache as CoopSc from root level to leave level and conclude a fruit level for utilizing the cooperative semantic cache for query processing and extending it for using the cooperation both in client and server and instead of database one can use for cloud, web. This will be extreme use in case of rigorous application area such as pharmacy and hospital environments. Further CoopSc not only for p2p environment but also for web and cloud environments with TAH query relaxation using multi dimension and multimedia applications presently as Cobase and with extension of ubiquitous and recommendation technologies as future research work.

References:

1. Laurent d’Orazio, Claudia Roncancio Semantic caching in large scale querying systems Laboratoire d’Informatique de Grenoble 681 rue de la Passerelle BP. 72 38402
2. Xiaohui Long, Torsten Suel Three Level Caching for Efficient Query Processing in Large Web Search Engines_ WWW 2005, May, Chiba, Japan. ACM 1595930469/05
3. Heiner Stucken schmidt Vrije Amsterdam Similarity-Based Query Caching de Boolean.
5. Boris Chidlovskii, Claudia Roncancio Marie-Luise Schneider Semantic Cache Mechanism for Heterogeneous Web Querying tutorial Xerox Research Centre Europe, Grenoble Laboratory, France
7. Michael J. Carey, Michael J. Franklin ; Data Caching Tradeoffs in Client-Server DBMS Architectures ACM 0-89791-425-2/91 /0005


19. Preetha Theresa Joy1 and K. Poulose Cache Replacement Policies for Cooperative Caching in Mobile Ad hoc Networks Cochin University of Science and Technology,Kochi, Kerala, India


21. Distributed Hash Tables Parallel and Distributed Computing Spring 2011. anda@cse.usf.edu


24. David Coquil and Harald Kosch semantic and cooperative document delivery over distributed systems.:
   lionel.brunie@insa-lyon.fr.

25. Takahiro Hara cooperative caching by mobile clients in push based information systems. CIKM’02, November
   4–9, McLean, Virginia, USA. 2002.

26. Xiaohui Long ThreeLevel Caching for Efficient Query Processing in Large Web Search Engines Copyright is held

27. P. Bouquet, G. Kuper, M. Scoz, S. Zanobini Asking and answering semantic queries EU funded project
   WIKEF;(Virtual Information and Knowledge Environment Framework).

**Corresponding Author:**

P.Mohankumar*,

**Email:** mkr_karai@yahoo.co.in