How Caching Queries at Client-peers Affects the Loads of Super-peer P2P Systems

Rozlina Mohamed[1, 2], Christopher D. Buckingham[1]

[1]*School of Engineering & Applied Science, Aston University, Birmingham B4 7ET, UK*
[2]*Faculty of Computer System & Software Engineering, University Malaysia Pahang, Kuantan 25000, Malaysia*
{mohamedr, buckincd } @aston.ac.uk

Abstract

Super-peer P2P systems strike a balance between searching efficiency in centralized P2P systems and the autonomy, load balancing and robustness provided by pure P2P systems. A super-peer is a node in a super-peer P2P system that maintains the central index for the information shared by a set of peers within the same cluster. The central index handles the searching request on behalf of the connecting set of peers and also passes on the request to neighboring super-peers in order to access additional indices and peers. In this paper, we study the behavior of query answering in super-peer P2P systems with the aim of understanding the issues and tradeoffs in designing a scalable super-peer system. We focus on where to post queries in order to retrieve the result and investigate the implications for three different architectures: caching queries at the peer; caching only at the super-peer; and an ordinary P2P system without any caching facilities. We are adopting the existing equation on measuring the network cost for query answering in super-peer systems. In addition, we are adapting the same equation for super-peer system with caching facilities while answering their queries. Using these equations, the cost of query processing for these architectures is compared. The paper discusses the tradeoffs between architectures with respect to caching, highlights the effect of key parameter values on system performance, and ends by considering whether certain knowledge domains are more appropriate for particular architectures.

Keywords

Peer-to-peer, super-peer, query routing, query answering

1. Introduction

Peer-to-peer (P2P) systems have become popular as a medium of sharing huge amounts of data. In a broad sense, there are three major types of P2P systems: pure, centralized and super-peer. This classification is reflects the varying degrees of decentralization in processing tasks and sharing resources among peers in the network. Super-peer P2P system such as YouTube[2] (one of the most popular video file sharing system) strikes a balance between inherent result searching efficiency in centralized P2P system; and the autonomy, load balancing and robustness provided by pure P2P system. The super-peer P2P system is also known as super-peer network in P2P system network architecture. A super-peer is a node in a super-peer P2P system that maintains the central index for the information shared by a set of peers within the same cluster. The central index is maintained in order to serve the searching request on behalf of these peers and also interconnected super-peers. All these tasks seem to take-out the peer autonomy features of the super-peer node.

Directing our goal is to study the scalability of query answering process in the super-peer system. In this paper, we are comparing the tradeoffs parameters for the cost of query answering while considering query caching as an alternative approach to the standard super-peer system. Thus, the next section is defining three different super-peer system architectures to be compared. Section 3 presents the network cost equations for each compared architectures while Section 4 is the parameters for estimating the caching facilities. Then, in Section 5 we are presenting the simulated result as a validation for our performance analysis in Section 6. Section 7 is the conclusion of our findings.

2. Super-peer System Architectures

In this section we are comparing three different super-peer systems architecture that we named it as standard super-peer systems, query and result cached at super-peer and location-based schema caching at client-peer. We begin by describing the basic concepts of query routing in a standard schema-based super-peer network system. Then, in the following sub-sections is the description for query routing in the super-peer system with the query caching facilities.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

2.1 Standard Super-peer Systems

In this paper, the query routing for a standard super-peer system is referred to Hyper project as described in [3, 4], thus as the first architecture for further comparison discussion. Our terms super-peer and client-peer are referred to 'hyperpeer' and 'peer'in Hyper. In our research context, we consider super-peer in a standard super-peer system as a node that responsible for query routing according to routing indices. The routing index is an index of shared resources located at peers in the network. This index keeps either the actual location or direction toward the actual location of shared resources [5]. Except for P2P file-sharing system, the data sources for most of the schema-based P2P systems are provided in XML (Extensible Markup Language) [6-9] and RDF (Resource Description Framework) [10, 11] format. XML is widely known as self-describing data, robust, durable, manipulable, free format for information identification and retrofitted with external conversion layer which is fit for future conversion or mapping of shared resources. Whereas RDF is used for describing shared resources on the Web. In our research we are considering XML as a shared data format while XQuery as our query language since it has been recommended by W3C for querying the XML data[12].

Searching for resource location intended for query result in schema-based P2P system can be seen as identifying the keyword from the query message, and matched this keyword with the existing schema provided in the routing indexes. Thus, the query message is forward towards resource location that has been identified from the routing indexes. In [9, 13], the routing index is constructed as a single XML tree of peers shared resources. Meanwhile, the keyword matching process is done semantically based on schema obtained in the index in [10, 14].

Figure 1 illustrates a scenario for query routing that will be used throughout this paper. It shows that peer p_1 as the *queried peer* that sends a query message request for data 'a' and 'b'. Data 'a' is obtained by peers p_2, p_3 and p_4 while peer p_5 is the resource location for data 'b'. Figure 2 shows the sequences of query routing in standard super-peer system from the queried peer toward resource locations for the example scenario in Figure 1.

2.2 Query and Result Caching at Super-peer

We perceive caching the queries and respective results is similar to materialized views concept that has been applied in database integration systems (batini, al-mourad, karunaratna). In P2P environment, this similar concept has been adapted in [14] (zhou, xia and qian). This subsection is defining a P2P system providing a view-based query processing mechanism. We are referring to [14] as the second architecture for further comparison.

Figure 2 shows the sequence diagram of query routing scenario for example query scenario in the second architecture. Here we assume that all of the data required by client-peer p1 is obtained by super-peer sp1. Thus, no further query routing is required by the second architecture. However, if the query request for schema that is not provided by the cached, the normal query routing as in standard super-peer systems is applied.

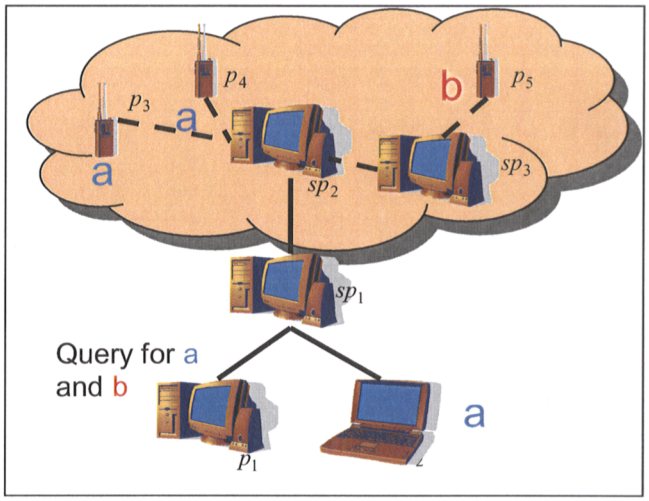

Figure 1. Example of query scenario

2.3 Location-based Schema Caching at Client-peer

Location-based schema caching is the caching strategy that only cached the routing direction information towards actual resource location. In contrast to the previous approach that cached the query together with the respective result, the caching of resource location information that has been introduced in [1](paper at kansas) [15] (lan quan et.al) would be able to reduce the cost for obtaining the cached information while the tendency of accessing an outdated result is definitely avoided. However, there is some trade-off cost required for query routing. We refer to (paper at kansas) as the third architecture for further comparison discussion. In (paper at kansas), the system is piggy-back on the standard super-peer system while the cached information that consist of resource locations information is obtained by client-peer. Therefore, normal route of query routing from client-peer may passed the query message direct to the resource location without referring to the super-peer routing indices as being done in the standard super-peer systems. The resource location information that has been cached is captured from the previous queries. In the case of queries that require for schema that is not provided by the cached, the normal routing as in standard super-peer is applied.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

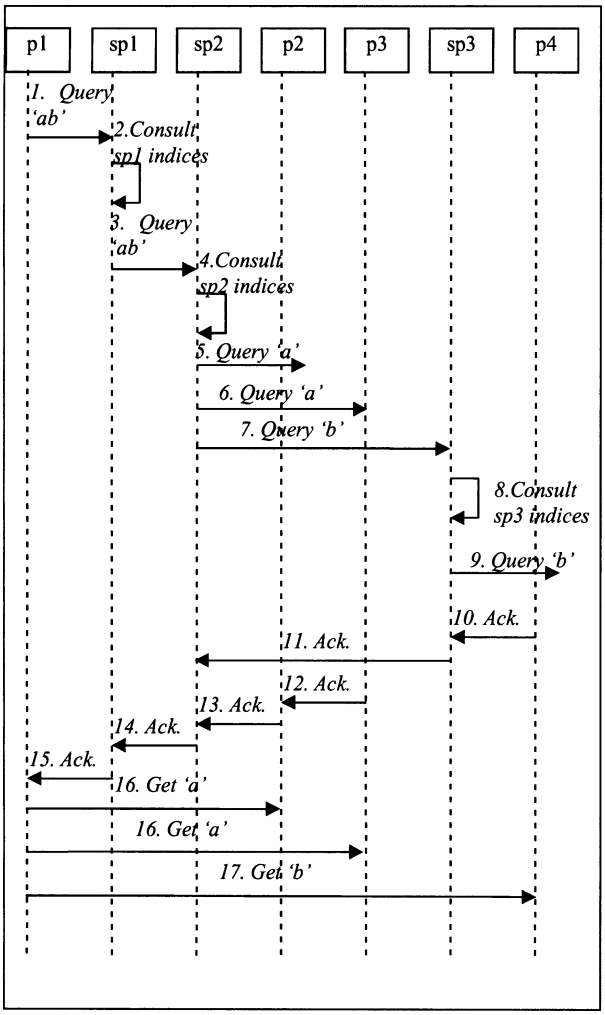

Figure 2. Sequence diagram for query routing scenario in a standard super-peer system

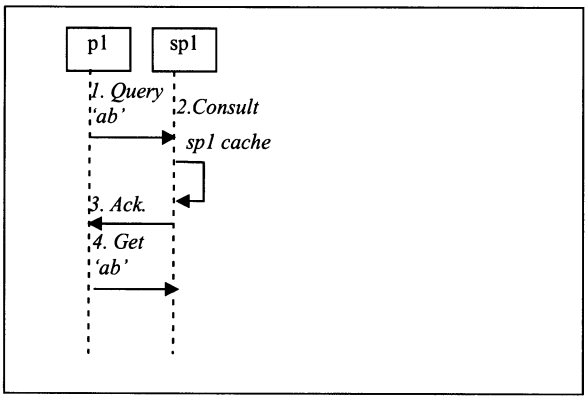

Figure 3. Sequence diagram for query routing scenario in a super-peer system with the query result caching facilities at super-peer node

3. Query answering Cost

To compare the abovementioned architectures, we need some equations that can be used to estimate the cost involved in query answering process. In this paper, we have divided the cost of query processing into two aspects. The first aspect is network cost involves in query routing while the second aspect is the cost of processing at *queried peer*. Here, we assume that the cost involved at resource locations are equivalent for all architectures. Therefore, we exclude the cost at the resource location in our paper. The following subsection discussed the cost of query answering for the standard super-peer system which is adopted from [16]. Meanwhile, the same equation has been adapted to fit the facilities provided by the query and result caching and location-based schema caching approaches.

Figure 4 shows the sequences of messages routing in the third architecture. Here we assume that required resource location is already being cached by p_1.thus, no message passing from super-peer sp1 is needed. Therefore, the messages passing are in between queried peer and the resource locations only.

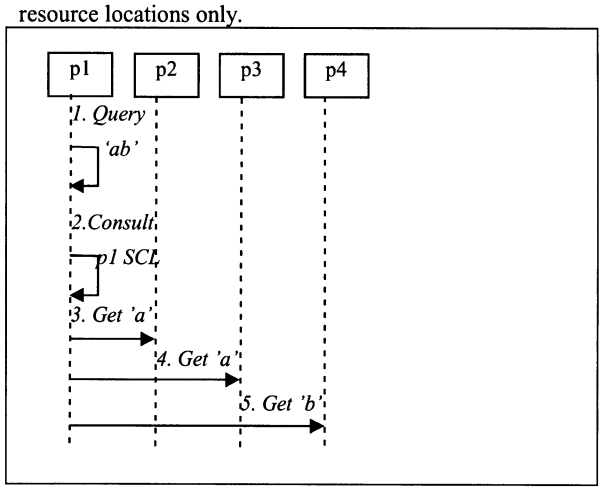

Figure 4. Sequence diagram for query routing scenario in super-peer system with SCL at client-peer node

3.1 Standard Super-peer Systems

The cost for query answering in a standard super-peer system is denoted as,

$$q_{cost} = \begin{array}{l} c_1.ExTotal\,\mathrm{Re}\,sults + \\ c_2.Ex\,\mathrm{Re}\,mote\,\mathrm{Re}\,sults + \\ c_3.ExServ \end{array} \quad (1)$$

where c_1 is the total cost that associated with finding one result, c_2 is the cost for transferring one result that has been processed by remote super-peer (which is not done by local super-peer), c_3 is the cost to startup the query answering process, *ExTotalResults* is the expected number of results for one query that has been performed, *ExRemoteResults* is the expected number of results returned by remote super-peer, which is not processed by local super-peer and *ExServ*

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

is the expected number of super-peers involved in order to get the result for the current query that is being processed.

In [16], the constant value have been assigned to c_1, c_2, and c_3, as 7778 bytes, 100000 and 2000 respectively. These parameter values have been recorded are based on their experiments in OpenNap project. OpenNap is a P2P system that clones Napster for simulating the use of 'server' in P2P environment. Napster is one of the most popular file sharing systems [17]. Here, we consider the 'server' in OpenNap as a super-peer node in the P2P systems.

The value for *ExTotalResults* is deriving from Equation (1), where the probability of x number of query successfully being processed is lead to number of query result returned. Since we do not have any running OpenNap experiment result, we assume that the *ExRemoteResults* is 17.29799335 % of *ExTotalResults* and the *ExServ* is 1.919 server. These values have been used in [16] detail calculation.

3.2 Query and Result Caching at Super-peer

The query answering cost for the super-peer system that has caching facilities at super-peer can be divided into two scenarios: (i) the query result found at the local super-peer, and (ii) the query result is not found at the local super-peer, thus require the query to be routed to other neighboring super-peer which is known as remote super-peer. Remote super-peer is a super-peer that is not directly connected to client-peer who initiates the query. In the real-world scenario, there is a possibility where the query result is not found in the network. In this case, we consider the query that has been initiated is routed up to remote super-peers. Thus the second scenario applied for this particular case. Here, we consider the cost of query answering that involves the remote super-peer(s) is same as for Equation (1). Meanwhile, the cost of query answering for the first scenario is shown as Equation (2)

$$q_{cost} = \begin{matrix} c_1.ExTotal\,\mathrm{Re}sults + \\ c_3.1 \end{matrix} \quad (2)$$

In this scenario, we consider the *ExServ* is 1 since local super-peer is the only super-peer involved in this query answering process. In Equation (2), the calculation for expected remote result is excluded since this scenario is calculated the cost for the query processing only from local super-peer.

3.3 Location-based Schema Caching at Client-peer

The cost of query answering in this architecture can be divided into three scenarios: (i) query result is found locally in schema cached list (SCL), (ii) the query result is found at local super-peer index, and (iii) the query result is found at remote super-peers index or when the query result is not found at all.

The cost of query answering for the third scenario is considered as the same the existing Equation (1), while the second scenario is consider the same as in Equation (2). Thus, the equation for the first scenario is represented in Equation (3).

$$q_{cost} = c_1.ExTotal\,\mathrm{Re}sults \quad (3)$$

In the third architecture, no super-peer involvement in query answering process therefore, *ExServ* is definitely excluded from Equation (3).

4. Cost estimation for caching facilities

In this section, we are estimating the cost of processing the query using two types of caching facilities. The first approach is caching the query and its result while in the second approach is caching the information of actual resource location (where the data being retrieved).

In this paper, we assume that the cached data is stored in a form of table. Data structure of this table is illustrated in Figure 5. We consider both approaches are using the same data structure with different types of data stored for 'Data Objects. The 'Data objects' in the first approach is the cached query with the respective results while in the second approach is used to keep data on resource location.

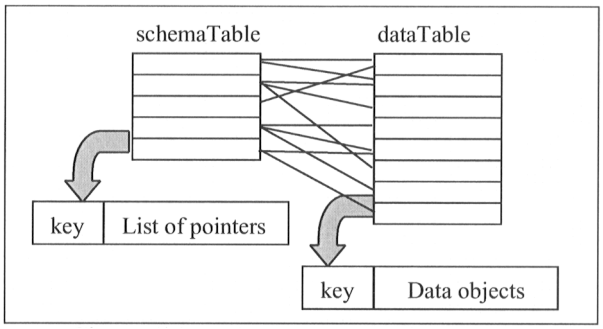

Figure 5. Illustration of data structure for caching facilities (based on [1])

4.1 Cost of maintaining the cached data for 'Query and Result Caching' approach

For the query and result caching approach, the data that has been cached is the query with the respective results. The *dataTable* size S is proportional to the number of query & the respective results that has been cached. Equation (4) is used to represent the query and result caching approach.

$$p \in \{\sum_{i=1}^{n} q_i + r_i\} \quad (4)$$

where p is a predicate which is the cached data in *dataTable* structure. Predicate p is a set of query q_i together with their respective query result r_i. Query q_i and result r_i that has been cached is assigned to respective schema in the *schemaTable* structure.

Cost of maintaining (μ_{cost}) for $q_i + r_i$ are consist of query message and query result costs. Thus, the cost of $q_i + r_i$ for each predicate are as follows,

$$q_i + r_i = qMessage_{cost} + q\,\mathrm{Re}sult_{cost} \quad (5)$$

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

$$= (WordsPerQuery * CharPerWord * BytesPerChar)$$
$$+ (WordsPerFile * CharPerWord * BytesPerChar)$$
$$= (2.4 * 5 * 2) + (10 * 5 * 2)$$
$$= 124$$

Each parameter used in equation (5) are defined in Table 1. These parameters are adapted from [16] and based on our own survey in the query processing. Equation (5) shows that the query and result caching approach require 124 bytes for each predicates being cached.

Table 1. Default values and description of each parameter used in the equations

Parameter Name	Default Value	Description
WordsPerQuery	2.4	Average keywords per query
WordsPerResult	10	Average words per query result
CharPerWord	5	Average characters per word
CharForPort	4	Estimation characters to describe a port
CharForIP	13	Estimation characters to describe an IP address
CharForPath	25	Estimation characters to describe a path of data
BytesPerChar	2	Number of bytes required for a character in java

4.2 Cost of maintaining the cached data for 'SCL' approach

In contrast to the previous sub-section approach, our SCL used the cached (*dataTable* structure) list to cache the resource location information. Thus, predicate p in SCL elements are,

$$p \in \{\sum_{i=1}^{n} loc_i\} \quad (6)$$

where p is a predicate which in *dataTable* structure that consist of information about where data i is located (loc_i). Each loc_i is also assign to particular schema in *schemaTable* structure.

Cost of maintaining (μ_{cost}) for loc_i are consist for each predicate are as follows,

$$loc_i = (CharForPort + CharForIP + CharForPath) * BytesPerChar \quad (7)$$
$$= (4 + 13 + 25) * 2$$
$$= 84$$

Parameters used in Equation (7) are defined in Table 1. Equation (7) shows that the SCL caching approach require 84 bytes for each predicates being cached.

5. Simulation

With the intention to test our proposed pre-processing query routing using SCL in the super-peer network and to validate the equations and their parameter that has been discussed throughout this paper, we have extended an existing simulator that has been used in [15]. This simulator has been written in Java and it is initially based on pure P2P network with the routing direction towards the resource location has been cached. Our simulation is focused on network load instead of calculating the peers' load to accomplish the query answering task. However, the query contents and the results are not simulated in this simulation.

5.1 Experimental setup

The routing queries behavior is based on three different super-peer architectures that we have compared. Here, each client-peer is directly connected to a single super-peer. For the experiments, we constructed 900 peer nodes where each super-peer is allocated to four client-peers. For the architecture with the caching facilities, the nodes that are responsible for obtaining the cache have obtained 50 cached data. This cache data is used for query routing purpose. Each query has 8 TTL (*time-to-live*) values.

5.2 Experiments

The experiment that has been setup is to compare the number of contacted peers and the number of messages generated in the network. The comparison that has been setup is between super-peer networks without any caching facilities, caching facilities is located at super-peer nodes and caching facilities is allocated at client-peer nodes. Figure 6 show the percentage of accumulated contacted peers to the total number of querying peers for query result retrieval. Figure 7 shows the percentage of generated messages in the network in order to answer the query versus the number of querying peers.

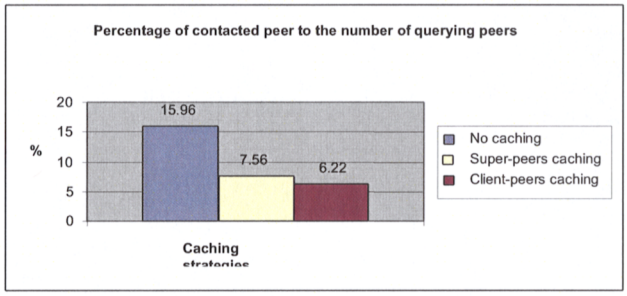

Figure 6. Percentage of contacted peers for query result retrieval

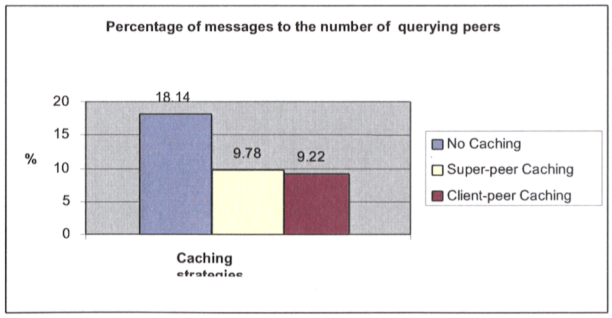

Figure 7. Percentage of messages in the network

Based on the simulation results on Figure 6 and 7, it is proven that caching facilities are capable to reduce up to 50% of contacted peers and messages generated in the network in answering the queries where the cached data is fully utilized. Furthermore, by allocating the cached data at the client-peers would be able to give more reduction for the number of contacted peers and messages generated. Therefore, allocating the caching facilities at the client-peers is more efficient for reducing the network cost to answer the query in the super-peer network.

6. Performance evaluation

In this section, we are comparing the cost of query answering to evaluate the super-peer architectures. We begin by comparing the network cost for answering the queries. Then, we illustrate the probability query processing cost while utilizing the caching facilities on answering the queries. Finally, we are presenting the cost of maintaining the caching facilities as the tradeoffs for eliminating the cost of answering queries.

6.1 Network cost for query answering

Figure 8 shows the comparison of network cost (q_{cost}) for 1000 query result that has been processed using three super-peer architecture that has been compared in Section 2 and 3.

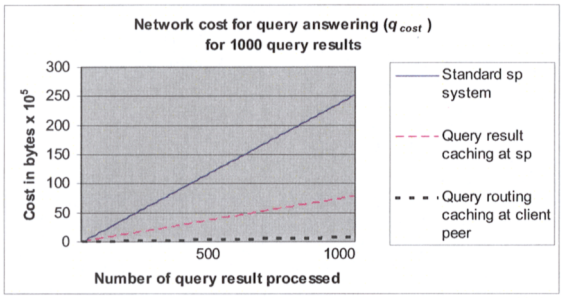

Figure 8. Network cost for query answering

This figure shows the tremendous reduction in network cost for query answering while implementing the caching facilities especially for the SCL usage at client-peers.

6.2 Query answering cost for query answering with caching facilities

Figure 9 show the query processing cost for each caching strategies that we have compared. In this figure we are illustrating the percentage of using the caching facilities that affected the cost of query processing. The figure illustrate that the more caching facilities is used, we can get more reduction in query processing cost. Furthermore, implementing SCL at client-peer to cache the routing information instead of query and result caching at the super-peer would be able to gives more reduction in query processing cost.

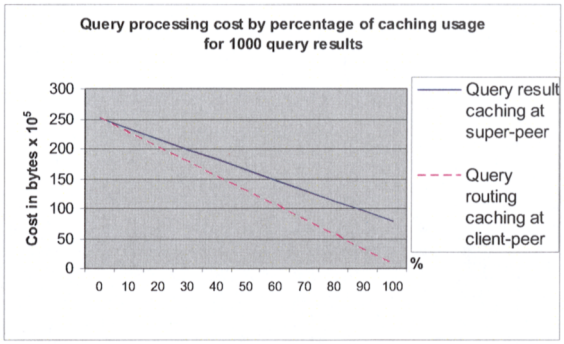

Figure 9.Cost for processing 1000 query result by percentage of caching facilities usage

Figure 8 is a graph that has been plotted to compare the maintenance cost (μ_{cost}) between the abovementioned approaches. This figure show that caching the resource direction for each predicate in the *dataTable structure* required less cost compared to caching the query and result.

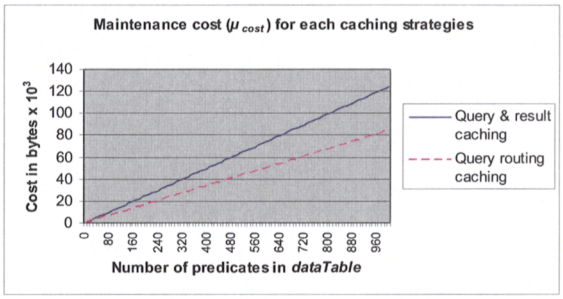

Figure 8. Maintenance cost of each caching strategies for 1000 predicates in *dataTable* structure

7. Conclusion

In this paper, we studied the behavior and performance of query answering processing in three different architectures that are based on super-peer systems. We are adopting the existing equation on measuring the network cost for query answering in super-peer systems. Then, we are adapting same equation for measuring the network cost of various architectures. In addition we are identifying parameters involves in query answering with caching facilities. Finally, we evaluated and compared the cost of each strategy. A summary of our findings for the three different super-peer architectures with two caching strategies are as follows:

- In our opinion, the super-peer system with the SC:L is the best architecture for the P2P system since it require least network cost for the query answering process. Certainly, this architecture requires an additional cost of maintaining the cached information. However, the tremendous cost reduction has been trade-offs with the small amount of memory require for maintaining the resource location information.
- The super-peer system with the query and result caching facilities has a good potential when the super-peer nodes is more stable and the memory is cheaper for obtaining the

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

cache. This is because, the super-peer in this architecture has to more responsibility compared to the standard super-peer node.

The standard super-peer system is the best architecture for a P2P system since it requires least amount of entire load for query answering. There is no doubt that the standard super-peer does not require any additional cost for caching facilities. However, the fact that the super-peer node is a single-point of failure for its cluster may affect the query answering process.

8. References

1. Rozlina, M. and D. Christopher, Buckingham. *Pre-processing for Improved Query Routing in Super-peer P2P Systems*. in *2008 IEEE Region 5 Technical, Professional, and Student Conference* 2008. Kansas City, USA: To appear.
2. YouTube, L. *YouTube*. 2008 [Available from: *http://www.youtube.com/*]
3. Calvanese, D., et al., *Hyper: A Framework for Peer-to-Peer Data Integration on Grids.* Lecture Notes in Computer Science, 2004. **3226/2004**(Semantics of a Networked World): p. 144-157.
4. Diego, C., et al., *Logical foundations of peer-to-peer data integration*, in *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2004, ACM: Paris, France.
5. Crespo, A. and H. Garcia-Molina. *Routing Indices for Peer-to-Peer Systems*. in *22 nd International Conference on Distributed Computing Systems (ICDCS'02)*. 2002. Washington, DC, USA: IEEE Computer Society.
6. Fegaras, L., He, W., Das, G., Levine D. *XML Query Routing in Structured P2P Systems*. in *DBISP2P2P Workshop in 32nd International Conference on Very Large Data Bases (VLDB)*. 2006. Seoul, Korea.
7. Igor, T., et al., *The Piazza peer data management project.* SIGMOD Rec., 2003. **32**(3): p. 47-52.
8. Boyd, M., et al., *AutoMed: A BAV Data Integration System for Heterogeneous Data Sources* Lecture Notes in Computer Science, 2004. **3084/2004**(Advanced Information Systems Engineering): p. 82-97.
9. He, W., Fegaras, L., Levine, D. *Indexing and Searching XML Documents Based on Content and Structure Synopses*. in *24th British National Conference on Databased (BNCOD)*. 2007. Glasgow UK: Springer.
10. 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 *12th international conference on World Wide Web*. 2003. Budapest, Hungary: ACM.
11. Kokkinidis, G. and V. Christophides, *Semantic Query Routing and Processing in P2P Database Systems: The ICS-FORTH SQPeer Middleware* Lecture Notes in Computer Science, 2005. **3268/2005**(Current Trends in Database Technology EDBT 2004 Workshops): p. 486-495.
12. W3C (2007) *XQuery 1.0: An XML Query Language*. **Volume**,
13. Sartiani, C., Manghi, P., Ghelli, G., Conforti, G. *XPeer : A Self-Organizing XML P2P Database System* in *9th International Conference on Extending Database Technology (EDBT), PhD Workshop*. 2004. Heraklion, Greece: Springer Berlin / Heidelberg.
14. Brunkhorst, I., Dhraief, H., *Semantic Caching in Schema-Based P2P Network.* Lecture Notes in Computer Science, 2007. **4125/2007**: p. 179-186.
15. Doulkeridis, C., Norvag, K., Vazirgiannis, M. *Schema Caching for Improved XML Query Processing in P2P Systems*. in *6th IEEE International Conference on Peer-to-Peer Computing*. 2006. AUEB, Greece: IEEE Xplore.
16. Yang, B. and H. Garcia-Molina. *Comparing Hybrid Peer-to-Peer Systems*. Technical Report, Stanford University 2001 [Available from: http://dbpubs.stanford.edu/pub/2000-35.
17. Wikipedia, *Napster*, *http://en.wikipedia.org/wiki/Napster*. 2008.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE