Collaborative Caching for Distributed Mobile Database Query Processing

Mohamed Ahmed Elfaki[1], Hamidah Ibrahim[2], Ali Mamat Mohamed Othman

Department of Computer Science
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
43400 UPM Serdang
Selangar D.E., Malaysia
[1]mohamedelfaki6@hotmail.com, [2]hamidah@fsktm.upm.edu.my

ABSTRACT

As mobile applications are becoming increasingly widespread, the need for developing schemes to improve the performance and reliability for mobile database management increase. In addition, caching data in a wireless mobile computer adds an advantage to mobile applications by reducing the bandwidth requirement due to battery power limitation when disconnection occurred. In this paper, we investigate and propose an approach that supports distributed processing of a collaborative caching technique to enhance the process of continuous queries in mobile database environment in terms of average delay and hit ratio. Since our approach deals with continuous query, we propose a framework for improving the process of continuous query mechanism that caches and shares the query's result among the nodes within the same cluster or with external database (neighbors cluster or main data server).

Keywords
Mobile database, Continuous query, Collaborative caching.

1.0 INTRODUCTION

Nowadays the advances of mobile and wireless communication technologies are playing very important role for running many types of business, because they make the process of posting and retrieving information very easy, interim of updating, modifying, deleting, etc. More significantly they are changing the way we live and do business. For instance, mobile users declare that hospital paramedic unit needs a powerful facilities system that can allow the medical staff to access the medical history of the victims, information regarding to their location and where the information came from. In addition, they need the capability of quickly locating and contacting medical personnel nearest to accident site. Therefore, mobile computing equipment can hand held this service by composing a database.

However, most of current approaches for cache coherence in mobile environments have not been designed to be used in cooperative applications such as monitoring the moving objects to detect their position, to decide where the nearest data sources to that particular object, beside that the response time to locate the requested data is another factor which should be put under consideration.
Therefore, current studies are focusing more to propose searching techniques that can minimize the response time to process the query (Sardadi, Mohd, Jupri, & Daman, 2008). In addition, none of the existing caching technique is able to monitor continuous queries with fully decentralized distributed mobile database (Berkenbrock & Hirata, 2008). Besides that, a number of researchers have made an intensive study on caching technology, and proposed some caching strategies for mobile query database system from the viewpoint of distributed mobile database (LI J., LI Y., Thai & Jianzhong). Thus, this paper investigates and proposes an approach that supports distributed processing of a collaborative caching technique to enhance the process of continuous queries in mobile database environment.

The rest of the paper is organized as follows. In section 2.0 the key concepts that are related to mobile query are presented. Section 3.0 gives literature review of existing collaborative caching approaches. Section 4.0 presents our proposed approach. Our framework is explained in section 5.0. Conclusion is presented in section 6.0.

2.0 CONCEPTS OF MOBILE QUERY

Mobile database query is categorized into two types of query which are fixed objects and moving objects. The following paragraphs elaborate and describe the sub-categorizes of each of them as illustrated in Figure 1. However, focus is given more on the continuous query which is the main focus of this paper.

For fixed object, localization of mobile units is one of the major concerns of researchers being interested in mobility (Marsit, Hameurlain, Mammeri & Morvan, 2005). For example when mobile user submits a query, the system should be able to locate it. Therefore, certain types of queries may involve mobile client location in implicit or explicit manner such as Location Dependent Query (LDQ), Non Location Related Query (NLRQ), and Location Aware Query (LAQ).

Furthermore, Moving Object Database Query (MODQ) is another type of mobile database queries. This type of query consists of all queries that are issued by mobile or fixed terminals and querying moving object database (database which represents for moving object e.g. plan or vehicles). Basically the position of moving object is the keyword of this type of query.

Spatio-Temporal Query is one of moving object query which combines space dimension with time dimension. Therefore it does not only focus on the location of an object but also on its position in a given time (trajectory). There are two types of spatio query, one concerns trajectories that describe a time history of the object movement. The second one focuses on the current position of the moving object and possibly its future position. Continuously changing data (position) raised many research problems at several levels for the spatio-temporal queries. Thus, becomes one of the major interests research currently (Marsit et al., 2005). Due to that a new type of moving object is introduced called Continuous Query which has two types of queries, which are spatio-temporal continuous query and location dependent continuous query. The last one is being tackled by many techniques to improve its performance, such as semantic caching, proactive caching, and collaborative caching.

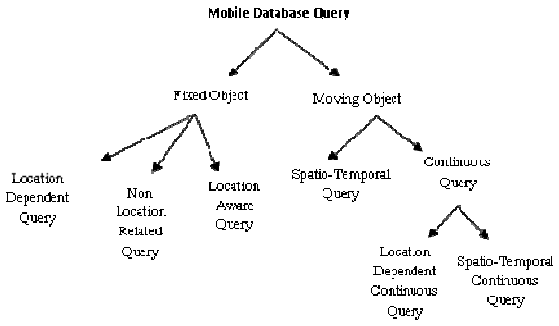

Figure 1: The Taxonomy of Mobile Query

Semantic caching is a type of caching techniques that is used to improve the system performance especially when disconnection occurred. This type of caching is crucial to Location Dependent Data (LDD) application. Therefore, by keeping a cache in mobile unit, part of new query result could be obtained locally. By doing so the system performance is improved and the wireless traffic network is reduced as well (Ren & Dunham, 2000).

Proactive caching model is a kind of caching technique that supports all types of spatial queries on smart mobile clients. It achieves outstanding performance compared to different caches such as traditional page caching or semantic caching in terms of cache hit rate and bandwidth savings (Hu, 2005).

Collaborative caching is developed to share the cache content among a number of peers. Furthermore collaborative caching has three ways of caching the data, such as bellow:
CacheData, is one of collaborative caching techniques that is used to cache data. In this scheme the nodes cache a passing by data item locally when it finds the particular data is popular, where many nodes requests for it (Yin & Cao, 2006).

CachePath, it provides the hop count information between the source and destination. By caching the data path for each data item, bandwidth and query delay can be reduced since the data can be obtained through less number of hops (Yin & Cao, 2006).

HybridCache, this type of collaborative cache is taking advantage of the CacheData and CachePath, based on some criteria's such as the data item size and time to leave, TT.

3.0 RELATED WORKS

This section of the paper, discusses the previous works related to continuous query mobile database. It focuses more on collaborative caching environment for improving or enhancing the performance of continuous query processing in mobile database computing.

Distributing continuous range processing query on moving objects (continuous query) is one of the related works of this research. This type of queries is mainly processed on the mobile device side, which is able to achieve real-time updates with less server load.

For caching data and query matters, Hu (2005) developed a proactive caching model as general framework to support all types of spatial queries on smart mobile clients. This framework achieves outstanding performance compared to different caches such as traditional page caching or semantic caching in terms of cache hit rate and bandwidth savings. Another feature of this approach is monitoring continuous spatial queries, which every smart mobile client can detect their own location and decide their own location updates.

Lillis and Pitoura (2008) covered the area of cooperative caching system in XML documents that allows sharing the cache content among a number of peers. There are two fundamental ways for sharing cache content in

terms of the level of cooperation among the participating peers which are, IndexCache where each peer caches its own queries that result of local process, where the distributed index is built on top of these local caches to facilitate sharing (each query checks the index to locate any peer whose cached results may be used in answering the query). Meanwhile in DataCache each peer is assigned a particular part of the cache data space (Lillis & Pitoura, 2008). In general, when a new query is requested, the system will check the cache to determine whether the query can be answered by the cached results of some previous queries.

Yin and Cao (2006) designed a cooperative caching technique to enhance data access in ad hoc networks. Their technique has two basic cooperative caching schemes, CacheData which is used to cache data, and PathData which caches the data path. A hybrid approach which can improve the performance further by taking the advantage of both schemes (CacheData and CachePath) while avoiding their weaknesses. CacheData, in this scheme the nodes cache a passing by data item locally when it finds the particular data is popular, where many nodes request for it. In addition, a node does not cache data if all requests for data are from the same node that to save space. The second scheme is CachePath, it provides the hop count information between the source and destination. By caching the data path for each data item, bandwidth and query delay can be reduced since the data can be obtained through less number of hops (Yin & Cao, 2006). However, recording the map between data items and caching nodes increases routing overhead. So optimization technique is used to reduce the records, for example a node does not need to record the path information of all passing by data if this node is near to the center data. As a result, CachePath performs better in situations where small cache size is involved, on the other hand; CacheData performs better in situations where large cache size exists. HybridCache is taking advantage of the CacheData and CachePath, based on some criteria such as the data item size and time to leave, TT.

Haojun et al. (2006) proposed a leveraging system to compute the capacities of mobile devices for continuous range query processing (Haojun, Roger & Wei, 2006). They contributed a distributed server infrastructure that divided the entire region into a set of service zones and cooperatively handle requests of any continuous range queries.

Chi-Yin et al.(2005) introduced a new type of caching known as distribute group based cooperative caching in a mobile broadcast environment, to increase data availability in mobile environment. The main objective of this study is to distribute group-based cooperative caching scheme, called tightly-coupled group (TCG). It consists of two types of cooperative cache management protocol which are *cooperative cache management admission control* and *cooperative cache replacement* (Chow, Leong, **&** Chan, 2005). The first cache is used to encounter a local cache miss to send request to its peers. If some peers turn in the required data item to the Mobile Host (MH) and the local cache is not full, the MH caches the data item, no matter whether it is returned by a peer in the same TCG or not. However, after peer sends the required data to requesting MH and if they belong to the same TCG, the peer must update the last access timestamp of data item to have a longer time-to-live in the global cache. In cooperative cache replacement, MH replaces the least valuable data item that is related to the MH itself and other members in the same TCG. This replacement satisfies three useful properties. First, the most valuable data is always retained or kept in the local cache. Second, in local cache the data which has not been accessed for a long period will be replaced eventually. Third, in a global cache, a data item which "spawns" replica is first replaced in order to increase the effective cache size.

Yu et al. (2009) proposed a new type of caching to improve data availability and access efficiency using collaborating local resources of mobile nodes. There are two basic problems of cooperative caching, which are cache resolution and cache management. Cooperative caching (COOP) explores data resource that includes less communication overhead by utilizing cooperation zones and hop-by-hop resolution. For management, COOP increases the effective capacity of cooperative caches by minimizing caching duplications within cooperation zone and accommodating more data varieties. The cooperation of caching nodes is twofold. First, caching node can answer the data requests from other nodes. Second, caching node stored the data not only on behalf of its own needs, but also based on other node's needs. Cache resolution addresses how to resolve a data request with minimal cost of time, energy, and bandwidth. Cooperative caching emphasizes on cache resolution is to answer how nodes can cooperate among each other in resolving data requests to improve the average performance. Hop-by-hop resolution is used to check along the path whether a node has the requested data, the request gets resolved before reaching the server. In addition, Cocktail resolution is used when the local cache misses and the request fails in its cooperation zone. Therefore hop-by-hop resolution is used to resolve the request along the forwarding path. Two metrics are adopted to measure the performance which are average response delay and average energy cost per request, which reflect the time efficiency and energy efficiency (Du, Gupta, & Varsamopoulos, 2009).

4.0 THE PROPOSED APPROACH

To overcome the limitation of mobile database processing, a continuous query optimization needs to be achieved by having a decentralized collaborative mobile database. Therefore a collaborative caching is one of the techniques that we will use to reduce the response time of any requested query and improve the performance of

mobile continuous query. Our proposed framework is designed to improve collaborative caching technique management to serve the query in minimum response time and increase the hit ratio for a requested query to be answered locally.

Figure 2 illustrates the evaluation and reevaluation of the query when a mobile client submits a query to get data item. The query is checked and evaluated at the local cache first to check whether the requested query can be answered locally (line 1) or not. If the result of the query is found, the algorithm will check the validity of the result interim of accurate and up to date data that can be achieved by comparing the result to the original copy in the data source (neighbor's nodes or main data server) using the data item ID.

In case, if the data item is not found in the local cache, the requested query for the desire data item will be checked based on its status[1] (priority) (line 5). If the requested query is urgent, the requested query is forwarded directly to the data center or main server to avoid any delay to return the result to the requested mobile host. Once the mobile host received the data item will send messages to its neighbor's nodes for updating matter (line 6.c). On the other hand, if the request data item is not urgent and it is not found in the local cache (line 7), the query is forwarded to the neighbors node within the same cache cluster (lines 8). Lastly, if the data item (dx) is not found in the group clusters of requested MH, the requested query is forwarded to another cluster or data server (DS) to get the desired data item (lines 11, 12, 13).

Begin
1. When a mobile host (MH), requests a data item (*dx*:), it first checks dx in its own local cache;
2. If a copy of *dx* exists in its cache and valid, then
3. Return *dx*;
4. Else
5. Check the requested query status (priority)
6. If the requested query is urgent query;
   a. Forward the requested query directly to the data server (DS)
   b. DS return the result to MH,
   c. Save the data items ID in history table and update other nodes
7. Else // query status is not urgent
8. Requests dx from neighbor's node within the same cluster.
9. If *dx* exists and still valid then
10. Return *dx*;
11. Else // data is not found in the cluster
12. Requests *dx* through other CC/DS;
13. Return dx
14. Repeat line 6.c
15. End

Figure 2: *Continuous Query Evaluation and Reevaluation*

[1] The status of the query is defined and formatted, whereby in each query packet there is a mark showing its status.

5.0 THE PROPOSED FRAMEWORK

Figure 3 illustrates the flow process of our framework when a mobile client submits a query to get data item. The requested query is processed in three stages, local node caches, cluster nodes (inter cluster) or across clusters (intra cluster). The query is checked at the local cache first to see if it has the data item. The checking of the requested data is based on data item ID, where each node has its own node index table for storing the data item. If the result of the query is found, the validity of the data will be checked.

Furthermore, if the data item is not found in the local cache (local miss), the node directs the query request to neighbors node within the same cluster. The requested query is served in a cluster and across clusters based on query status (popular data) and frequency access.

On the other hand, the query is checked at the local cache first. If local cache missed, the query is forwarded to neighbor's nodes in the same cluster then in other cluster to get the data item that if the status of the query is normal (means not urgent). Once the requested node received the desired data item, information is updated in the local cache. Meanwhile, the neighbor's node and cluster manager[2] will update their index table based on the priority or needs of the data for future request.

This framework is distinct from other works by adding two functionalities; first a new strategy to guide the node requester by formatting a query based on its status (priority) to indicate what process that needs to be used to serve the request in optimal time.

Second, the new collaborative caching strategy of the way nodes cooperate and share the information message to neighboring nodes within the same cluster and with other clusters group. Thus, this way of collaborative is expected to enhance the data retrieval performance, reduce the bandwidth cost, and increase the hit rate locally.

In addition, each cluster has a manager which is coordinated the cluster address (CA) for all nodes. The cluster manager is on charged for checking the priority and urgently of the requested query that is been accessed or served, beside that the cluster manager is decided which data should be replaced or dropped from the cache history table (index table).

Based on the current works that we have reviewed, we can conclude that the majority of the approaches achieved good performance, but still an optimal collaborative caching need to be modeled to improve the existing collaborative caching management. Thus, by

[2] The cluster manager will be selected based on certain criteria such as the space size, sufficient resources, and longest expected time to stay in network.

proposing a new strategy of collaborative caching, performance enhancement is expected for continuous query process. Hence, this paper attempts to overcome the limitation of previous works by enhancing the performance of the continuous query (collaborative cache management) with respect to response time, storage space, data consistency, low bandwidth, and reliability.

6.0 CONCLUSION

In this paper, we first discussed the issues of mobile queries and the related works that have been developed to improve the performance of mobile queries in general and continuous query process particularly.

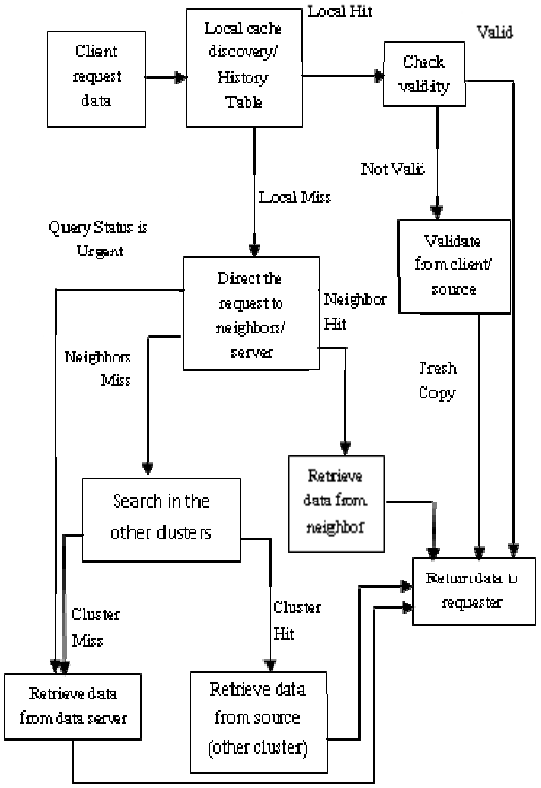

Figure 3: a Client Request Data using Collaborative Caching Strategy

Furthermore, cooperative caching can offer benefits in the performance of delay-tolerant networks (Pitkanen & Ott, 2007). Beyond the realm of networking, there are still open optimization problems to be addressed, such as the size of the cooperation zone, the (distributed) caching structure and organization, and fairness in cache access and management

Therefore the expected results of collaborative caching technique of this research, will improve the performance of collaborative caching management in order to enhance continuous query process for mobile database environment by reducing energy and bandwidth consumption, increase hit ratio, and improve data availability.

REFERENCES

Berkenbrock, C. D., & Hirata, C. M. (2008). Supporting Coherence in Mobile Cooperative Systems. *The Seventh IEEE International Symposium in Proceeding of Network Computing and Applications, Brazil*, pp. 240-243.

Chow, C.Y., Leong, H. V., **&** Chan, A. T.S. (2005). Distribute Group Based Cooperative Caching in a Mobile Broadcast Environment. *The 6th International Conference in Proceedings of Mobile Data Management, Ayia Napa, Cyprus*, pp. 97–106.

Du, Y., Gupta, S. K., & Varsamopoulos, G. (2009). Improving on Demand Data Access Efficiency in MANETs with Cooperative Caching. *The Journal of the Ad Hoc Networks, 7*(3), 579-598.

Hu, H. (2005). *Spatial and Continuous Spatial Queries on Smart Mobile Clients*. Ph.D. Thesis, The Hong Kong University of Science Technology.

Haojun, W., Roger, Z., & Wei, S. K. (2006). Distributing Continuous Range Processing on Moving Objects. *Proceedings in the 17th DEXA Conference, 4080/2006*, pp. 655-665.

LI, J., LI, Y., Thai, M., & Jianzhong, J. (2005). Data Caching and Query Processing in MANETs. *The Journal of Pervasive Computing and Communications, 1*(3), 169-178.

Lillis, K., & Pitoura, E. (2008). Cooperative Xpath Caching. *The ACM SIGMOD International Conference in Proceedings of Management of Data*, 327-338.

Marsit, N., Hameurlain, A., Mammeri, Z., & Morvan, F. (2005). Query Processing in Mobile Environments: A Survey and Open Problems. *The first International Conference in Proceeding of Distributed Frameworks for Multimedia Applications*, pp.150–157.

Pitkanen, M. J., & Ott, J. (2007). Redundancy and Distributed Caching in Mobile Delay Tolerant Networking. *The 2nd ACM/IEEE International Workshop in Proceedings of Mobility in the Evolving Internet Architecture*, pp. 579-59.

Ren, Q., & Dunham, M. (2000). Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. *The 6th Annual International Conference in Proceedings of Mobile Computing and Networking*, pp. *210–221.*

Sardadi, M. M., Mohd, M. S., Jupri, Z., & Daman, D. Daut, (2008). Choosing R-tree or Quadtree Spatial Data Indexing in one Oracle Spatial Database System to Make Faster Showing Geographical Map in Mobile Geographical Information System Technology. *The Journal of World Academy of Science, Engineering and Technology 46,*

University Technology of Malaysia, 36, pp. 249-257.

Yin, L., & Cao, G. (2006). Supporting Cooperative Caching in Ad Hoc Networks. *The Journal of the National Science Foundation, IEEE Transactions on Mobile Computing, Volume 5*(1), *pp. 77-89.*