American Journal of Economics and Business Administration 3 (1): 33-39, 2011
ISSN 1945-5488


# Collaborative Caching Architecture for Continuous Query in Mobile Database

Mohamed Ahmed Elfaki, Hamidah Ibrahim, Ali Mamat and Mohamed Othman
Department of Computer Science, Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, 43400 Serdang, Selangar D. E., Malaysia

**Abstract: Problem statement:** Cooperative caching, which allows sharing and coordination of cached data among clients, is a potential technique to improve the data access performance and availability in mobile ad hoc networks. However, variable data sizes, frequent data updates, limited client resources, insufficient wireless bandwidth and client's mobility make cache management faces many challenges. **Approach:** In this study, we propose a Cluster Cache Collaborative (CCC) to reduce the average delay and increase the local cache hit ratio. Our architecture considers several factors that affect cache performance, namely: access probability, query status, distance between the requester and data source/cache, and data size. A collaborative cache management strategy, intra/inter cluster, is developed that employs Last Recently Used (LRU) and Time-To-Live (TTL) as the replacement data policy. **Results:** Our proposed architecture is being evaluated and compared to previous study' architectures, and result shows that our architecture is different from other approaches by including query status model into a collaborative caching architecture to reduce the number of hopes between the requested node and the source node. **Conclusion/Discussion:** Due to the limitations of previous study' architectures, this study attempts to enhance the performance of the continuous query (collaborative cache management) with respect to the local cache hit ratio and the average delay.

**Key words:** Mobile database, continuous query, collaborative caching

## INTRODUCTION

Mobile databases are gaining popularity and are likely to do so well into the future as portable devices become more and more popular and common. However, the growing need for monitoring applications (traffic control, ecology and environment monitoring) has forced an evolution on data processing paradigms, moving from Database Management Systems (DBMSs) to Data Stream Management Systems (DSMSs).

The unique characteristics of mobile ad hoc network lead to research challenges due to limited bandwidth, disconnection, and frequent link rate. The limited bandwidth is caused by the wireless links between mobile devices (Hajar and Mahmoud, 2010). One of the main issues is the nodes movement and the dynamic change that occurs in the network topology (Natsheh and Buragga, 2010). That is due to the lack of centralized infrastructure such as a base station or an access point, since each mobile node acts as a router, forwarding data items to other mobile nodes.

In dynamic Peers to Peers (P2P) environment, query execution and optimization faced many challenges, due to the lack of the global knowledge. The collaboration of autonomous peers is essential to fully take advantage of the resources in the system. This involves more optimization issues, in terms of coordination, locality aware peer clustering, and load balancing.

P2P query processing should be effective and fully efficient for handling a large scale of autonomous peers in dynamic and distributed networks. There are many challenges to implement the query answering functions in P2P systems. One of these challenges is peers join and leave the system anytime, anywhere, due to a purely dynamic and ad hoc network environment. Hence, the fundamental protocol should be efficient enough to handle peers and network failure. In addition, a full decentralized process should be adopted for the query processing (Weining *et al*., 2005).

Collaborative caching is particularly attractive in environments where the network is constrained in terms of bandwidth, disconnection, and storage. Collaborative caching offers several benefits since it enables efficient utilization of available resources by storing different data items and sharing them among themselves.

**Corresponding Author:** Mohamed Ahmed Elfaki, Department of Computer Science,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang,
Selangar D. E., Malaysia

Consequently, current studies are focusing more on searching techniques that can minimize the response time to process the query (Marute *et al*., 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 (Jinbao *et al*., 2005).

This study investigates the data retrieval challenges of mobile ad hoc networks and proposes a novel architecture, called Cluster Cache Collaborative (CCC) for caching that exploits LRU and Time-To-Live (TTL) values for cache replacement. The aim of CCC is to increase the caching hit ratio and reduce the average delay for requested data.

The study begins with the discussion on the key concepts of mobile query followed by the literature reviews of existing collaborative caching approaches. Then, our proposed approach is presented. Conclusion is presented in last section of this study.

**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 Fig. 1. However, focus is given more on the continuous query which is the main focus of this study.

For fixed object, localization of mobile units is one of the major concerns of researchers being interested in mobility (Marsit *et al*., 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).

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 in moving object e.g. plan or vehicles). Basically the position of moving object is the keyword of this type of query.

Spatio-Temporal queries are 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)

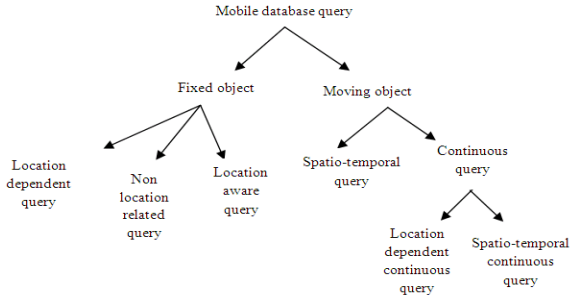


Fig. 1: The taxonomy of mobile query

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).

For moving query, a new type of query is introduced called Continuous Query which is further divided into 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.

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 (Qun and Margaret, 2000).

Proactive caching model is a kind of caching techniques 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 (Haibo *et al*., 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, in this scheme a node caches a passing by data item locally when it finds the particular data is popular, where many nodes requests for it (Liangzhong and Guohong, 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 (Liangzhong and Guohong, 2006).

- HybridCache, this type of collaborative cache is taking advantage of the CacheData and CachePath, based on some criteria such as the data item size and Time-To-Live (TTL).

**Related works:** Researchers are still exploring the current performance and future potential of 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, Haibo *et al.* (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, in which every smart mobile client can detect their own location and decide their own location updates.

Kostas and Evaggelia (2008) cover 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. 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.

Liangzhong and Guohong (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 CachePath which caches the data path and provides the hop count information between the source and destination. A hybrid approach which can improve the performance further by taking the advantage of both schemes (CacheData and CachePath) while avoiding their weaknesses. In the CacheData, a node caches 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. 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. 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 source of the requested 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-Live (TTL).

Wang *et al*. (2006) proposed a leveraging system to compute the capacities of mobile devices for continuous range query processing. They contributed a distributed server infra-structure that divides the entire region into a set of service zones and cooperatively handles requests of any continuous range queries.

Chi-Yin and Alvin (2005) introduced a new type of caching known as distributed 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. 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, the data in a local cache 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 node's data source 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 nodes' 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.

## MATERIALS AND METHODS

**The proposed system architecture:** In this study, we propose a cluster collaborative caching architecture with a distributed database management. Our architecture considers a network environment (MANET) with a number of Mobile Hosts (MHs) and Data Center (DC), whose address is known by all other mobile nodes, which can be regarded as a database server that stores all the data items needed by the whole MANET. The MHs can act either as a DC which provide the data services to other MHs if it has the requested data or as a requester of the data from other DC, or MHs. In short, our collaborative caching architecture has a number of clusters, number of mobile nodes, and data centers.

**Overview of the cluster collaborative caching:** This architecture is developed to provide a collaborative caching service which makes use of clustering mobile ad hoc network and consists of four basic components namely: cluster, index profile, information searcher, and cache management.

The cluster component is responsible for the structure and maintenance of clusters which consists of MHs that join and leave the cluster dynamically. If a new mobile host joins the cluster, the coordinator of each cluster updates its index profile when a new mobile host's notification is received. It is also possible for a new joining MH to be a coordinator, if it has some specific qualification that are more cache space, sufficient resources and longer expected time to be in the network than the current one. The new MH takes over the information from the old coordinator and sends messages to all neighboring coordinators to update their index table. For the MH which leaves the cluster, the leaving MH informs the coordinator of the cluster to update their indexed information. If the leaving MH is a coordinator, it contacts the next elected MH to take over the coordination information and that is based on the qualification as mentioned earlier.

In each cluster there are MHs with specific characteristic (sufficient sources, more space, and longer time expected to be in the network) acting as coordinator (manager) to maintain the mobile host address and the list of data items along with their Time-to-Live (TTL) within the same cluster. In addition, the cluster coordinator is also the gateway to collaborate with other clusters (across clusters) or database server. A cluster coordinator is in charge of marinating information about specific query executions. This information includes the source providers that provide the requested data and the locality information of the providers. A coordinator also is responsible to coordinate the requesters of the data to determine what data should be cached by which MH in the cluster to avoid any redundancy data (cache management component). Basically, our architecture has two types of collaborative caching: mobile hosts within the same cluster and other clusters.

The index profile component provides cross cluster information. By using the index profile, information is shared between the MHs within the same cluster, cluster to cluster, and the data server.

The information searcher component provides the location of the data item which is requested by the client. The result of the requested query can either be the original data from the data source or the cached copy in a MH. Basically, our architecture locates the data items based on the requested query status (urgent or normal). The requested query is checked at the local cache first. In case if the requested query is normal and the data item is not found locally, the requested query is forwarded to the neighborhood in the same cluster and

cluster to cluster before sending the request to the data source to reduce query delay.

The cache management component consists of three sub-components namely: admission control, whose in charge to determine what data items should be cached; cache replacement, which determines what data should be removed when there is no space for a new data to cache; and cache consistency or validation that maintains synchronization of the data items that are cached compared to the original data items in data source.

**Clustering architecture:** To overcome the limitations of mobile database query 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 are used to reduce the response time of any requested query and improve the performance of mobile continuous query. Our proposed architecture is designed to improve collaborative caching technique management to serve the query in minimum average delay and increase the local cache hit ratio.

Our architecture is divided into several clusters which are distributed based on the similarity of data item. Each cluster consists of a number of mobile hosts: one MH plays the role of a cluster head which is a coordinator and a gateway to access other clusters; and other MHs play the role of cluster members, which exist in each cluster. The number of cluster members is dynamic which is based on the frequent changes of cluster. Basically, our architecture consists of nodes that are distributed in clusters as shown in Fig. 2. The cluster is built based on interest of the requested data item. There is more than one cluster, where each cluster has several MHs, but the number of MHs within the cluster might increase/decrease depending on MHs joining or leaving the cluster (the cluster configuration is beyond the scope of this study). In addition, each MH in the cluster has a Host Address Table (HAT), table holding the data ID that it caches.

However, as our architecture classifies the status of serving the query, communication format is constructed to meet this requirement, for the queries and nodes that store the desire data. A query is divided into two status, normal query and urgent query. The query will be formatted and processed based on cluster coordinator ID which has the Cluster Address (CA) for storing the nodes ID along with their cached data item ID within the cluster; Host Address Table (HAT) for storing the data item ID based on the history access (priority); expire period or Time-to-Live (TTL) and lastly the status of the requested query to be served. Basically, the

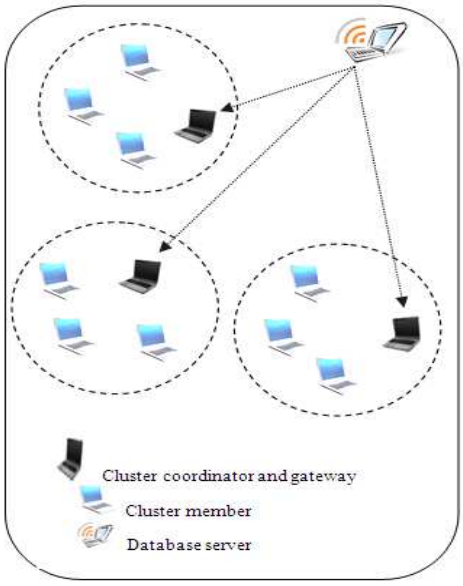


Fig. 2: Cluster cache collaborative architecture

coordinator is used to interrogate the MHs inside the cluster and supply the MHs address. HAT defines a pointer to any additional host links (collaborative MHs). The HAT manages a history link of MH caches. Therefore, by classifying the query, a collaborative caching architecture is expected to increase the hit rate and reduce the average delay.

## RESULTS

This part of the study, evaluates some of the current related architectures that we have reviewed. Based on our knowledge there is still a need to design an optimal collaborative caching architecture to improve the existing collaborative caching management. Table 1 illustrates the comparison between our architecture and the previous works, in terms of whether the architecture provides a decentralized model, whether it is being managed in cluster or transmission range for ambulated data, and whether the query directory considers the status of the query. However, the combination of all the models will create a new way of collaboration to serve a requested query among mobile hosts. For instance, decentralized model, allows the MHs to act either as a DC which provides the data item services to other MHs if it has the requested data item or as a requester of the data item from other DC, or MHs. Therefore, by build a decentralized model into our architecture will reduce the average delay and increase the local cache hit. For managing the cluster or transmissions range model, is

Table 1: Architecture evaluation

| Author's name and publish year | Decentralized model | Managing the cluster or transmission range | Component of the cluster / query directory based on query status | Item to be cached |
|---|---|---|---|---|
| Liangzhong and Guohong (2006) | × | × | × | Cachedata, cachepath, and hybrid cache |
| Weining *et al*. (2005) | √ | × | × | Cache data |
| Haibo *et al*. (2005) | × | × | × | Cached the index of the object |
| Chi-Yin and Alvin (2005) | √ | Tightly-coupled group | × | Data item |
| Yu *et al.* (2009) | √ | Transmission rang | × | Data item |
| Wang *et al*. (2006) | √ | Transmission rang | × | Query |
| Qun and Margaret (2000) | × | × | × | Query result |
| Our approach | √ | Cluster | √ | Data item/ query |

designed to control and group the mobile host based on the similarity of data item, and help to locate the requested data item with less average delay. Query directory based on query status model, plays the role of determining the kind of service for the query to be processed to avoid broadcasting the request for each mobile hosts. Our architecture is different from other approaches by considering all the above models. Thus, by proposing a new architecture of collaborative caching, performance enhancement is expected for continuous query process by including the entire models mentioned earlier, in particularly the query status model into a collaborative caching architecture to eliminate or reduce the number of hopes between the requested host and the source host.

## DISCUSSION

Based on the current works that we have reviewed, we can conclude that the majority of the approaches achieved good performance, but still optimal collaborative caching needs to be modeled to improve the existing collaborative caching management. Thus, by proposing a new architecture of collaborative caching, performance enhancement is expected for continuous query process. Hence, this study attempts to overcome the limitations of previous works by enhancing the performance of the continuous query (collaborative cache management) with respect to the local cache hit ratio and the average delay.

## CONCLUSION

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

Furthermore, cooperative caching can offer benefits in the performance of delay-tolerant networks (Mikko and Jorg, 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 increasing the local cache hit and reducing the average delay.

## REFERENCES

Chi-Yin, L.H. and C. Alvin 2005. Distribute group based cooperative caching in a mobile broadcast environment. Proceedings of the 6th International Conference on Mobile Data Management, 9th-13th May, ACM New York, ISBN: 1-59593-041-8, pp: 97-106.

Haibo, H., X. Jianliang, W. W. Sing, Z. Baihua, L. Dik, and L. Wang-Chien, 2005. Proactive caching for spatial queries in mobile environments. Proceedings of the 21st International Conference on Data Engineering (ICDE'05), 5th-8th April, IEEE Computer Society, pp: 403-414. http://doi.ieeecomputersociety.org/10.1109/ICDE,

Hajar, B. and F. Mahmoud, 2010. An algorithm for localization in vehicular Ad-Hoc networks. J. Comput. Sci., 6: 168-172. ISSN: 1549-3636

Jinbao, L., L. Yingshu and L. Jianzhong, 2005. Data caching and query. Processing in MANETs. Int. J. Pervasive Comput. Communications, 1: 169-178. DOI: 10.1108/17427370580000122

Kostas, L. and P. Evaggelia, 2008. Cooperative Xpath caching. Proceedings of the ACM International Conference on Management of Data, 9th-12th June, ACM New York, ISBN: 978-1-60558-102-6, pp: 327-338.

Liangzhong, Y. and C. Guohong, 2006. Supporting cooperative caching in Ad hoc networks. IEEE Trans. Mobile Comput., 5: 77-89. ISBN: 1536-1233.

Marsit, N., A. Ameurlain, Z. Mammeri and F. Morvan, 2005. Query processing in mobile environments: a survey and open problems. Proceedings of the 1st International Conference on Distributed Frameworks for Multimedia Applications, 6th-9th February, IOS Press Amsterdam, The Netherlands, pp: 150-157. ISBN: 0-7695-2273-4

Marute, S., M. Masserie, J.M. Shafry, D. Zahabidin and 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. Proceedings of the World Academy of Science, Engineering and Technology, 17th-19th December, pp: 249-257. ISBN: 2070-3724

Mikko, J. and O. Jörg, 2007. Redundancy and distributed caching in mobile DTNs. Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture, 27th-30th August, ACM New York, ISBN: 978-1-59593-784-8, DOI: http://doi.acm.org/10.1145/1366919.1366930.

Natsheh, E. and K. Buragga, 2010. Nodes density and broadcast management in heterogeneous environments of mobile Ad-Hoc networks. J. Comput. Sci., 6: 312-319. ISSN 1549-3636

Qun, R. and D. Margaret, 2000. Using semantic caching to manage location dependent data in mobile computing. Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, 6th-11th August, ACM New York, pp: 210-221. ISBN: 1-58113-197-6

Wang, H., R. Zimmermann and W.S. Ku, 2006. Distributing continuous range query processing on moving objects. Springer-Verlag Berlin Heidelberg, 4080: 655-665. ISBN: 978-3-540-37871-6

Weining, Q., X. Linhao and Z. Aoying, 2005. CoCache: Query Processing based on Collaborative Caching in P2P Systems. Springer Berlin, ISBN: 978-3-540-25334-1, pp: 498-510.

Yu, D., K.G. Sandeep and V. Georgios, 2009. Improving on demand data access efficiency in MANETs with cooperative caching. J. Ad Hoc Network., 7: 579-598. ISBN: 1570-87057:579-598