Indian Journal of Science and Technology, *Vol 6(11), 5474–5479, November 2013*

Performance Analysis of Cache Consistency Maintenance in Mobile Environment using Agent Technique

G. Shanmugarathinam[1*] and K. Vivekanandan[2]

[1]Research Scholar (external mode) in Bharathiar University, Coimbatore, Tamil Nadu, India; metshanmugam@gmail.com

[2]Professor, Bharathiar University, Coimbatore, Tamil Nadu, India; vivekbsmed@gmail.com

Abstract

This paper illustrates three Agents Based Techniques namely, Thread Based Cache Consistency Model, Multiple Server Cache Consistency Model, and Cluster Based Cache Consistency Models used to solve the problems of cache consistency maintenance and also compared with existing method. The drawbacks of existing method are latency, excessive bandwidth wastage, query delay, server overloads as well as network traffic. In the Thread Based Cache Consistency Model, the log is maintained in both server and client, it is easy for server as well as client to make their data object reliable. Agents at server and client make an internal request response operation(s) to access or update data object. It is easy to update the cached data and as well to maintain consistency between client and server. The Multiple Server Cache Consistency Models where the node is designed as a middle server (master client cache) between server and clients, some of the client data cache is placed in a middle server. The middle server is near the client, as result the communication cost and energy consumption is comparatively less. It reduces server's workload. The Cluster Based Cache Consistency Model designed as a Cluster head node between the server and the client using agent technique, effectively reduces the query latency. These three Agent Based Models are implemented in Ns2 and their performance was analysed and compared with existing method. Agent Based Technique resulted in better performance in relation to update delay, through various numbers of nodes and update rate.

Keywords: Mobile Database, Wireless Networks Database Cache, Threads Agent, Low Bandwidth Networks.

1. Introduction

In a mobile database processing, data caching at mobile clients is an important and effective technique for improving the performance of wireless data dissemination system variable data size, data updates, limited client resource and frequent client disconnections make cache management a challenge issue. Object caching is often used to improve the performance of mobile application. When data at the server changes, the client hosts must be made aware of this fact in order to invalidate their cache, otherwise the host would continue to answer queries with the cached values returning incorrect data. Although a number of studies have been made in this subject, few researchers focused on cache consistency maintenance in mobile database. We analysis the existing system problem, so we focus on Agent based technique used in cache consistency in mobile environment. As illustrated in Figure 1, we used the NS2 tool for simulation and the performance were analysed.

2. Related Work

2.1 Updated Invalidation Report (UIR) Based Cache Techniques

The Invalidation Report based cache management is an attractive approach for mobile environments. In this approach the server periodically broadcasts an IR in which the changed data items are indicated. Since IR arrive

**Corresponding author:*
G. Shanmugarathinam (metshanmugam@gmail.com)

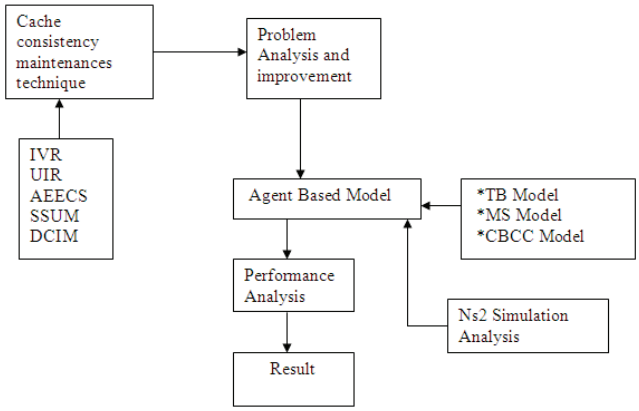

Figure 1. Framework of agent based technique.

periodically, client can go to sleep most of the time and wake up when the IR is received. It has some drawbacks such as long query latency and low hit ratio. There is a long latency problem with a UIR [1] (Updated Invalidation Report) based approach, where a small fraction of the essential information related to cache invalidation is replicated several times within an IR interval, and hence the client can answer a query without waiting until the next IR. If there is a cache miss the client still needs to wait for the data to be delivered.

2.2 Adaptive Energy Efficient Cache Invalidation Scheme

This method [2] is used to reduce the bandwidth requirement, the server transmits in one of the three modes slow, fast and super-fast. The mode is selected based on thresholds specified for time and the number of clients requesting updated objects. The drawback of this method is when the mode changes to slow, the client has to wait for a long time to utilize cache data during invalidation report and waits for query replay from the server.

2.3 Smart Server Update Mechanism

SSUM [3] the server sends data updates to the Cache Node (CN) request mobile host that desires a data item send its request to its nearest Query Directory (QD). If this QD finds the query in its cache, it forwards the request to the CN caching the items, which in turn sends the item to the requesting mobile host. Otherwise, it forwards, it to its nearest QD, If the request traverses all QDs without being found, a miss occurs and it gets forwarded to the server which sends the data item to the request mobile host. The drawback in this method is latency and more bandwidth is wasted.

2.4 Distributed Cache Invalidation Method for Maintaining Cache Consistency in Wireless Mobile Networks (DCIM)

This paper proposes [4] a client-based cache consistency scheme that is implemented for caching data items in MANETs, namely COACS, where special nodes cache the queries and the addresses of the nodes that store the responses to these queries. IT is a pull-based algorithm that implements adaptive TTL, piggybacking and prefetching, Cached data items are assigned adaptive TTL values that correspond to their update rates at the data source, where items with expired TTL values are grouped in validation requests to the data source to refresh them, whereas, unexpired ones but with high request rates are perfected from the server. The shortcoming in this approach is that the client demands wait for the server replay for the Invalidation report, Query delay and more latency.

3. Agent Based Technique

The client server model becomes one of the main ideas of mobile computing. Client server agent architecture that supports database application on mobile host, when mobile hosts are disconnected, transactions are processed locally at the mobile hosts. When mobile hosts reconnect to the database server, synchronization processes are carried out to reconcile information. Server agent maintains log of each client cache data. Whenever a client updates the data base, the server sends the updating information to other client using server agent log information. Table 1 provides the summary of Agent technique in mobile environment.

Server Agent process

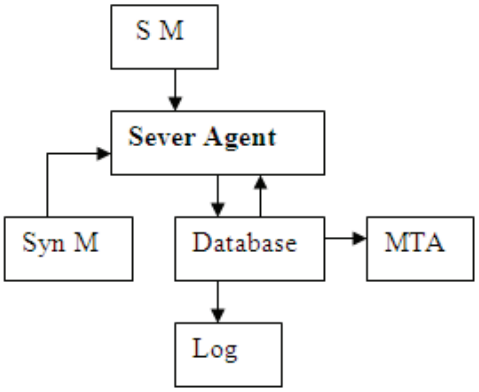

Figure 2. Server agent process.

Client Agent Process

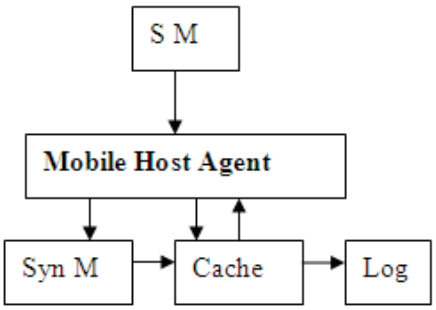

Figure 3. Client agent process.

Key Terms

SM = Security Module
Syn M = Synchronization Module
MTA = Migrating Thread Agent

Server Agent

Agent at server maintains as well as keeps on monitoring the frequent broadcast values and frequent client cache access. Whenever a value is read/written to server, it has to be updated and broadcasted. During this process, server agent maintains a log which holds information about broadcast values, information of mobile client who needs updated value as shown in Figure 2.

Migration Thread Agent

Whenever write operation is performed by a mobile client to the server, a special thread called "Migration Thread Agent" is activated, which keeps monitoring which client is performing the write operation to the server. It maintains a write log of both, client & server. It also seeks whether the updated data object is required by any other mobile clients. If so, it finds certain client(s) and it begins to synchronize in order to keep data object at cache as more reliable as shown in Figure 2.

Client Agent

Agent at client maintains as well as keeps on monitoring the frequent broadcast values and agent at server. Whenever a value is read or written to server, it will be updated to server. Now the updated value will be broadcasted to the requested mobile client. Agent at the client maintains a Thread Log which holds information such as broadcast values, broad cast time, log IDc as shown in Figure 3.

3.1 Agent Based Three Models

3.1.1 Thread Based Cache Consistency Model

This model is compared with Existing Invalidation Report algorithm. A drawback of invalidation message imposes a high processing of load on clients. Thread Based Cache Consistency model lies in the following four key features:

(i) Use of the log at mobile user's cache maintains the data consistency
(ii) Use of Thread Agent (TA) at both client and server
(iii) Use of log at server
(iv) Use of Migration Thread Agent at server.

Table 1. Summary of agent technique in mobile environment

Components	Client	Server
Client Agent, Server Agent	To listen to the response, to synchronize the updated data with server agent.	To listen to the request , to update the database and synchronize with Client Agent.
SM- Security module	To authenticate user-identity with the appropriate server.	To verify user ID and to keep track of the client host.
Cache	To check and maintain the frequently accessed data in local cache.	To check the valid data cached in Mobile Client.
MTA-Migration Threads Agent	To update the data in the local cache	When the data becomes invalid in mobile cache; it starts updating data with server and client
Sync. Module	To work together with current data using the synchronization module with server agent.	To keep valid data in client cache using synchronization module process in client agent
Log	To track the current data in local cache using log-time and date.	To track the data in the database using log-time and date.

Since log is maintained in both server and client, it is easy for the server as well as the client to make their data object reliable. Thread Agents at server and client make an internal request response operation(s) to access or update data object. It is easy to update the cached data, easy to maintain consistency between client and server.

3.1.2 Multiple Servers - Model

In this model, the node is designed as a middle server (master client cache) between server and clients. Some of the client data cache is placed in a middle server, which is near the client. In mobile environments, a Mobile Cache is found in one of the two states: (i) Awake or (ii) Sleep. If a Mobile Client is awake, an internal request is shared between Thread Agent at Middle server and Thread Agent at Client to ensure that data object is updated. If there are any updations, the SynchM of server synchronizes with SynchM of middle server and client in order to make data object valid. The data objects of a Mobile Client in the sleep state are unaffected, until it wakes up. When a mobile client wakes up, a new Thread is created which holds last accessed log; this log is passed to the Middle server. On receiving the log, it compares with previous log maintained by it. If it is an invalid data cache, the thread listener of middle server starts the synchronization with client for updating data. So, the communication cost and energy consumption is very less and it is easy to update the cached data and to maintain consistency. It reduces workload in the server. This technique saves wireless bandwidth and reduces the network traffic.

3.1.3 Cluster Based Cache Consistency Model

We designed the cluster based caching technique in a mobile environment. This technique improves data accessibility and reduces the query latency in a mobile environment. In this model, we designed a Cluster Head Node in between the server and the client, using agent technique. Cluster head selected is nearest to the center of the grid and full battery power among other nodes in the cluster. Agent technique is easy to maintain cache consistency in the server, cluster head and the client node. So, it reduced the query latency compared with previous DCIM model.

4. Performance Evaluation by Simulation

The performance of an individual mobile client and server is measured in terms of Throughput (total number of interactions) and Latency (average interaction completion time). The thread based model does not require broadcasting of an Invalidation Report. At the client side, Agent analyses cache data with server agent. Thread Based Algorithm has significantly better performance than Invalidation Report approaches.

Multiple Server Model is compared with the existing system, Smart Server Update Mechanism (SSUM). In SSUM model, due to latency being high, more bandwidth is used, the client is made to wait idle for the server to reply. In Multiple Server Model, when the number of servers is increased, the percentage of blocking can be computed using agent based technology. Hence we get average latency time and less update delay compared to the SSUM algorithm.

Cluster Based Cache Consistency Model is compared with the existing system - Distributed Cache Invalidation Method (DCIM). In Cluster Based Cache Consistency Model, it is found that cost for communication and energy consumption is very low and easy to maintain the cache consistency. This increases the local cache hit ratio, avoiding every request sent to server because the cluster head answers the query, so that the network traffic is reduced and update delay is reduced.

The Ns2 is used to simulate the mobile environment concept. The simulation time is 2000 seconds, network size 400x400m, wireless bandwidth 2 Mb/s. Node speed 2m/s, size of data items 10 kb and channel capacity of each mobile host has 1 Mbps. The MAC protocol uses 802.11. We take a number of nodes 10-100, Data update rate 5-50.

Table 2. Simulation result shows that, the existing method - SSUM, DCIM model is more update delay when compared with Agent based of two models MS, CBCC model and varies with number of nodes and data update rate.

Figure 4 and Figure 5 explain that SSUM, DCIM model update delay is more when compared with MS, CBCC model of Agent Technique which varies with number of nodes, data update rate.

5. Conclusion

In this paper, we have analyzed the performance of existing methods with three agent based system. In Thread Based Cache Consistency Model, it is easy to update the cached data and also to maintain consistency when compared with invalidation report. Multiple Server Cache Consistency Model when compared with SSUM Method

Table 2. Update delay

Number of Nodes	10	20	30	40	50	60	70	80	90	100
CBCC	0.04	0.04	0.07	0.06	0.07	0.09	0.1	0.11	0.12	0.12
M S	0.05	0.06	0.06	0.07	0.08	0.1	0.11	0.11	0.12	0.13
DCIM	0.06	0.06	0.08	0.07	0.08	0.1	0.12	0.12	0.13	0.14
SSUM	0.05	0.07	0.08	0.09	0.1	0.11	0.12	0.13	0.14	0.15
Data Update Rate	**5**	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
CBCC	0.06	0.11	0.14	0.18	0.26	0.21	0.25	0.35	0.32	0.36
M S	0.08	0.11	0.16	0.16	0.25	0.23	0.26	0.39	0.31	0.36
DCIM	0.1	0.13	0.18	0.27	0.3	0.39	0.41	0.43	0.42	0.46
SSUM	0.1	0.15	0.23	0.33	0.29	0.48	0.54	0.61	0.59	0.64

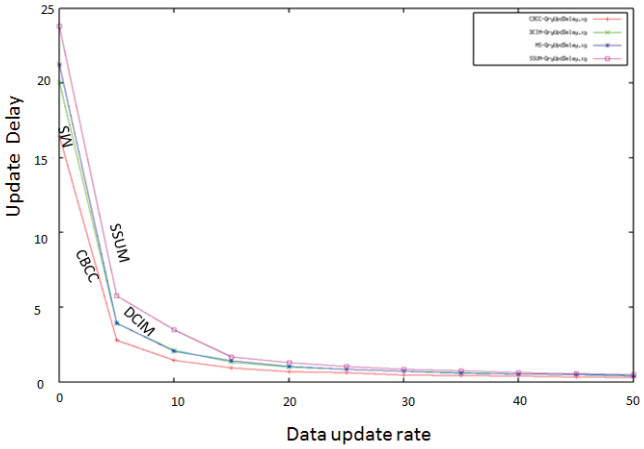

Figure 4. Update delay varies with data update rate.

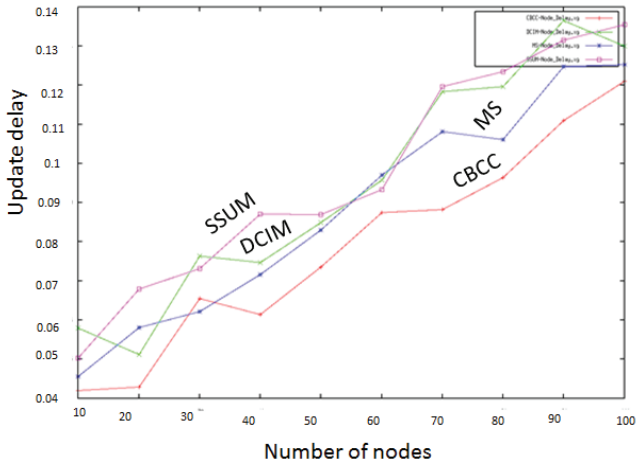

Figure 5. Update delays varies with number of nodes.

Technique, shows that wireless bandwidth is saved and the network traffic is reduced. The Cluster Based Model compared with DCIM Model reduces the update delay. Simulation results show that, the Agent Based Three Techniques gives a significantly better performance than earlier approaches.

6. Reference

1. Duvvuri V, Shenoy P et al. (2003). Adaptive leases: a strong consistency mechanism for the world wide web, IEEE Transaction of Knowledge Data Engineering, vol 15(4), 1266–1276.
2. Madhukar A, and Alhajj R (2006). An adaptive energy efficient cache invalidation scheme for mobile databases, Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France.
3. Mershad K, and Artail H (2010). Smart server update mechanism for maintaining cache consistency in mobile environments, IEEE Transaction on Mobile Computing, vol 9(6), 778–795.
4. Fawaz K, and Artail H (2013). Distributed cache invalidation method for maintaining cache consistency in wireless mobile networks, IEEE Transaction on Mobile Computing, vol 12(4), 680–693.
5. Shen H, Kumar M et al. (2004). Energy efficient caching and prefetching with data consistency in mobile distributed systems, Proceedings of the 18th International Parallel and Distributed Processing Symposium.
6. Huang Y, Cao J et al. (2010). Flexible cache consistency Maintence over Wireless Adhoc Networks, IEEE Transaction on Parallel and Distributed System, vol 21(8), 1150–1161.
7. Chang C-Y, and Chen M-S (2002). Exploring aggregate effect with weighted transcoding graphs for efficient cache

replacement in transcoding proxies, Proceedings of the 18th International Conference on Data Engineering. 383–392.
8. Hartung E F, Horn U et al. (2001). Streaming technology in 3G mobile communication systems, Computer, vol 34(9), 46–52.
9. Wang Z, Kumar M et al. (2006). Dynamic cache consistency schemes for wireless cellular networks, IEEE Transactions on Wireless Communications, vol 5, No. 1, 366–376.
10. Imielinski T, and Badrinath B R (2002). Wireless graffiti–data, data everywhere matters, Proceedings of 28th International Conference on Very Large Data Bases.
11. Kahol A, Khurana S et al. (2001). A strategy to manage cache consistency in a distributed disconnected wireless environment, IEEE Transactions on Parallel and Distributed Systems, vol 12(7), 686–700.
12. Yeung M K H, and Kwok Y-K (2009). Wireless cache invalidation schemes with link adaptation and downlink traffic, IEEE Transactions on Mobile Computing, vol 4(1), 68–83.
13. Tolia N, and Wolbach A (2003). Improving mobile database access over wide-area networks without degrading consistency, Proceedings of MobiSys '07 Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, 71–84.