An Overview of Mobile Database Caching

Rooma Rathore (ratho005@umn.edu)
Rohini Prinja (rohinip@cs.umn.edu)

Abstract

Wireless networks have become an integral component of the modern communication infrastructure. Database applications in wireless devices are of special interest to us in this paper. They are different from traditional client-server database applications due to low bandwidth in the wireless environment and frequent disconnects from the server. Hence, efficiently caching some of the elements that are frequently required by the mobile device can save the scarce resources and give better response time for the client application. Many mechanisms have been published in research to address parts of the problem.

In this paper, we analyze various caching mechanisms for mobile devices with the emphasis on critiquing the assumptions made in various schemes. We compare these techniques based on their strengths and weaknesses. This paper presents a comprehensive summary of work done in mobile database caching area.

Keywords: Mobile computing, caching, cache replacement, cache consistency.

1 Introduction

Mobile devices like laptops, palmtops, cellular phones etc. have become ubiquitous today. These devices cater to a large variety of applications from web-browsing to mini database applications. This paper focuses on the database applications running on mobile devices. The client-server paradigm in the wireless environment differs significantly from the traditional approach because of two main reasons: frequent disconnections and low bandwidth. Also, in a mobile environment, upstream queries (i.e., from client to server) are more resource-consuming than the downstream queries (i.e., from server to client), so there is a need to reduce the number of trips made to the server. Caching data items at the client side is a solution. However, the caching techniques used for traditional database models can not be applied in this arena as it is. Additionally, mobile clients also have limited resources like power, so any of the caching schemes have to be energy efficient and support long and frequent disconnections. In the following paragraph, we look at the criteria that must be considered for designing a cache management mechanism for mobile environment.

Since, wireless bandwidth and client's battery power are the two scarcest resources; we need an efficient way to measure these resources. Packet efficiency and power efficiency respectively are used to measure bandwidth and battery power. Packet efficiency means the ability of the algorithm to minimize the total number of packets sent on wireless link. Power efficiency refers to the ability to minimize the energy spent by the client that is running the algorithm. Lastly the system performance is measured in terms of access

efficiency, which is minimizing the period of time from when a mobile computer issues a data request until the time the data item is received. Having defined the criteria for designing a cache management scheme, we look at the typical setup of wireless environment.

Basic setup (Figure 1) of a mobile environment can be thought of as consisting of a Mobile Service Station (MSS) and a number of Mobile Hosts (MH) which acts as clients. Each MSS has a physical area that it can serve, called its cell. Mobile hosts connect to MSS which in turn connects to the actual database server for answering queries [1, 2, 6]. An alternative scheme is cooperative caching [5] in which instead of one database serving 'n' mobile hosts in its cell, clients form a tightly coupled group to exploit the cache space of each other to answer their queries.

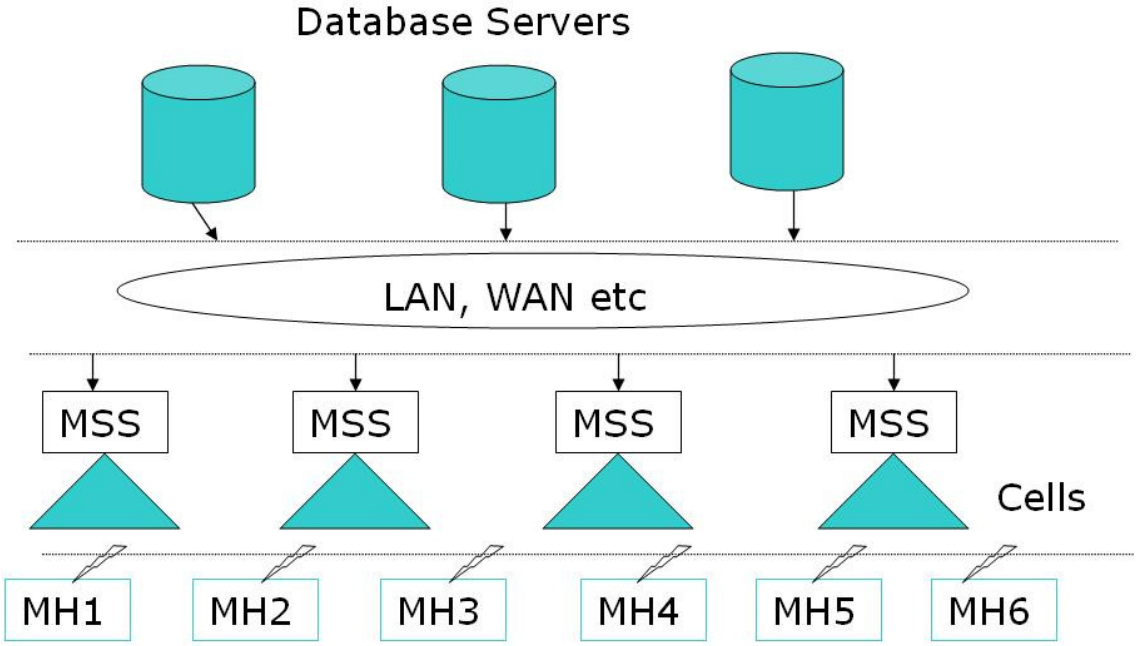

Figure 1: Basic setup of a mobile environment

Although many schemes have been proposed for caching in mobile devices, we sensed the lack of a thorough survey paper that discusses different caching schemes in entirety; right from what to cache to how to replace cached items to make way for new ones. Our paper attempts to bridge this gap in this research area. The main goal of the paper is to look at and analyze different caching schemes as a whole and compare and contrast them. In addition, since these schemes make a lot of assumptions about the underlying network on which mobile devices work, we look into these assumptions thoroughly. We have categorized our work on mobile database caching in three areas, cache granularity, cache coherence and cache replacement.

The rest of the paper is organized as follows: section 2 discusses the general overview of caching; section 3, 4 and 5 details the caching granularity, cache coherence strategies and cache replacement techniques, respectively. Finally section 6 concludes the paper.

2 Caching Overview

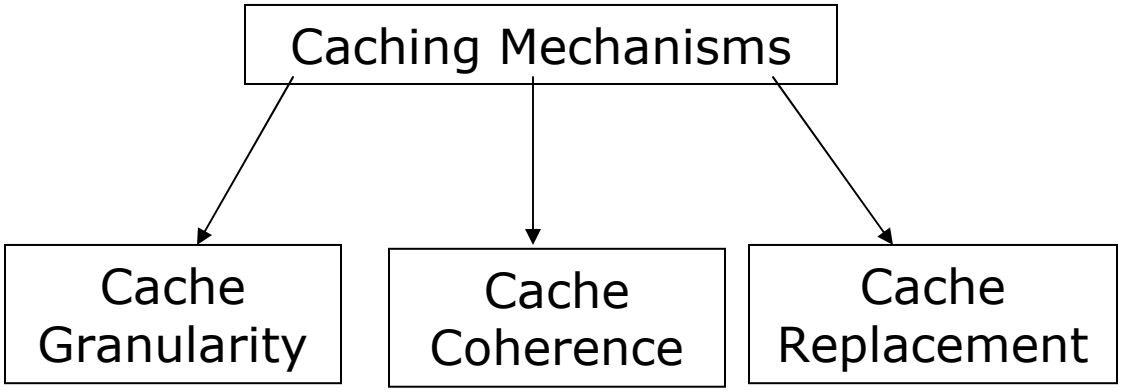

Figure 2: Caching mechanism

A caching mechanism (Figure 2) is characterized by its *caching granularity*, *cache coherence strategy* and *cache replacement policy* [6]. Caching granularity answers the basic question of what to cache; this includes the decisions made by the application to cache right kind of information so that it can optimize the cache storage. Had the storage capacity of a mobile device been unlimited, the application could have cached all the data items that it requested from the server. Since, this is not the case, intelligent decisions made by the application about what to cache, in the very beginning, affects the efficiency of caching system markedly.

After the decision has been made as to what to cache, the next step is to define a cache coherence strategy whose function is to keep the cache consistent. A number of different schemes have been proposed in this area. Majority of them are cache invalidation schemes which are based on push-based mechanism, in which a server repeatedly broadcasts/multicasts data reports whenever a specific item changes in the database. Listening to such reports, a client can decide if the cache it is keeping, is valid or not and can mark the specific items to be invalid.

When a client has to add some new data to the cache, it employs a cache replacement policy to purge old data to make way for the new data. Traditional replacement policies like LRU do not work in the mobile environment efficiently. Also these policies are page-based to gain additional cache hits for future accesses (principle of locality). However in wireless environments, this benefit can not be reaped as the typical application data accesses do not strictly obey the principle of locality of reference.

Additionally, none of the conventional caching techniques are right fit for mobile environments as they usually assume a fairly stable network, a reasonably high transmission bandwidth and a high degree of locality among database items residing within a data page at the database server. None of these attributes are present in a mobile environment since a wireless network is unreliable and has low bandwidth.

3 Caching Granularity

Caching mechanisms in conventional client-server environment are usually page-based, primarily because the server's storage is page-based and transmission bandwidth is comparable to the disk bandwidth. The overhead of transmitting one item or a page is same. Furthermore, the overhead will be paid off if any item within the transmitted page will be accessed by a client in the near future. In general, page-based caching mechanism requires a high degree of locality among the items within the page to be effective.

In contrast, mobile clients are powered by short-life batteries. Caching a page will result in wasting of energy when the degree of locality is low. Downloading a whole page instead of just the requested item will result in long query delay, so page based caching is not suitable for mobile environment. Other techniques like attribute caching etc., lowers the degree of granularity and hence are more efficient In the following sub-sections we explain these techniques and compare them.

3.1 Object Caching

The granularity level in this scheme is individual objects. Each mobile client tends to have its own set of hot objects that it accesses most frequently. Furthermore, a client might access different attributes of an object in different queries it initiates [6]. Using this scheme, whenever the server receives a request, it sends all attributes of a qualified object, x. The idea here is to prefetch all attributes of a requested object to the client that will cache the returned attributes [11].

3.2 Attribute Caching

The granularity level in this scheme is the attributes of individual objects. After the server has evaluated the query submitted by client, it returns only those attributes of the qualified objects that are requested by client. This method is very efficient if the application is going to use only the selected attributes always. For example, an application that checks for name and rate of all hotels in an area will only be interested in these attributes and might not need other attributes such as rating, vacancy etc. In such cases not returning all the attributes, saves bandwidth. Not caching all attributes saves cache space and so helps increase the cache hit ratio. On the other hand if one set of attributes are used in a query (say 'q_1') and another query (say 'q_2') requires a different set of attributes then it is better to cache the whole object.

3.3 Hybrid Caching

Hybrid caching is a mix of object and attribute caching defined above. In object caching, the database server will send all attributes of a qualified object 'x', to the client. However, often, not all attributes of 'x' will be accessed in the future. This not only wastes the transmission bandwidth, but also occupies storage for caching other more frequently accessed attributes. Hybrid caching restricts the database to prefetch only those attributes of a qualified object with a high likelihood to be accessed in the future. A prefetching threshold 'p' is defined for each attribute, and hence the attributes whose access probabilities are above 'p', will be prefetched. Experiments done in [7] suggest

that hybrid caching performs better than attribute and object caching and results in high cache hit ratio and low response time.

3.4 Semantic Caching

The idea of Semantic Caching (SC) in mobile environment stems from the fact that semantics of cache content can be exploited to make better decisions about what to cache [4]. The mobile client maintains the semantic descriptions and results of previous queries in its cache. If a new query is totally answerable from the cache, then no communication with the server is necessary [8]. If it can be partially answered then the original query is trimmed and the trimmed part is sent to the server and processed there. By processing queries this way the amount of data transferred over the wireless link can be substantially reduced. Table 1 below, compares advantages of semantic caching to traditional Page Caching (PC).

Item	**SC**	**PC**	**Reason**
Communication cost	low	high	Only the required data is transferred in SC, not whole page.
Cache space overhead	low	high	Only data satisfying previous queries are stored in SC
Parallelism in query processing	easy	difficult	Missing data is exactly determined as SC keeps semantic information
Disconnection handling	efficient	inefficient	Not only previously cached queries, but new queries with (partial) cached data can be answered in SC.
Coherency control	efficient	inefficient	Out-dated data items can be incrementally maintained in SC

Table 1: Semantic Caching vs. Page Caching in Mobile Computing

In semantic caching, cache is divided into fragments and sub-fragments. After a query is evaluated by running it on the first fragment, the remainder query can then be checked against the remaining cache and trimmed by next candidate fragment or sub-fragment. This process continues until there is no fragment that could contribute to the query result. The final remainder query is sent to the server for processing.

The main advantage of semantic caching is that by trimming queries, local cache contents can be used even if the query is not an exact match. This helps reduce the communication with the server. However, a prominent disadvantage is that it applies to only select and project queries. It can't be used with joins etc.

3.5 Comparison

Firstly, both object and attribute caching are defined for object-oriented databases [6]. Given an object-oriented database, although the decision to choose between object-based or attribute-based caching mechanism is more application specific, it is noteworthy that page-based caching granularity does not give good results in a mobile environment. To reap the benefits of both objects and attribute caching, a hybrid model can be employed. Applications like traffic information systems should employ a hybrid model as based on the location, query results can vary significantly. In addition the hybrid model can be combined with semantic caching to gain performance benefit. Semantic caching

mechanisms can also benefit applications that involve answering continuous queries [9] like finding the nearest restaurant in the vicinity of a mobile car. However, when the query content is huge, semantic caching may not give the anticipated benefit.

4 Cache Coherence Strategies

A cache coherence strategy involves cache invalidation and update schemes to invalidate and update an out-dated cached item. Conventional cache invalidation usually employs an eager and stateful approach. When an item is modified, the server immediately sends an invalidation message to all clients holding a cached copy of modified item (since because of stateful behavior, it already knows which data-item is cached at which client). An eager invalidation approach is not feasible in a mobile environment as each client connects and disconnects from the wireless network frequently. So the server might not be able to send invalidation messages to all clients that have a cached copy of the modified item.

An item in cache may become invalid for two reasons:

- The item has been updated on the server. This is also referred to as temporal-dependent invalidation [3].
- The client has moved to a different location and the results of previously cached query are no longer valid. This is also referred to as location dependent invalidation [3].

Most of the research for cache invalidation has focused on the prior part, i.e., invalidation due to server updates. Only recently the focus has shifter to the latter part.

Various invalidation techniques have been proposed over the last 20 years for caching. They can be broadly categorized into three main categories:

1. Stateful Server – The server knows the status and cache contents of all the clients. Each time a data item is updated the server sends invalidation message to all the clients that have cached the data item. This is called push-based mechanism [3]. In a mobile environment, when a mobile client moves to another cell, it has to notify the server of its new location. This method generates too much uplink traffic and is not very scalable.
2. Client Verification – The client verifies with the server the validity of its cache before any of the contents are used and requests only the required items. This method is also inefficient in our context, since it requires the client to check with server each time a new query is executed. This is called pull-based mechanism [3].
3. Stateless Server – The server doesn't know about client's location or contents of its cache. The server periodically broadcasts invalidation reports containing recently updated data items. The clients listen to these reports and determine the validity of its cache. Clients go uplink only if the items can't be found in their cache or have been invalidated.

We discuss different invalidation schemes in detail below.

4.1 Timestamp-based Schemes

D. Barbara and T. Imielinski proposed three cache invalidation schemes, namely, *Broadcasting Timestamps* (TS), *Amnesic Terminals* (AT) and *Signatures* (SIG) [12]. In the TS scheme, the server broadcasts an Invalidation Report (IR) every 'L' seconds which contains the timestamps of data items updated in the last '*w'* seconds. The client uses this report to invalidate the contents of its cache. A client disconnected more than '*w*' seconds has to purge all its cache even though some of the items in the cache might still be valid. The problem in this technique is to identify the value of '*w'*. If a low value of *w* is used then the time for which clients can be disconnected and still retain cache, is very low. If a high value is used then a large amount of data has to be broadcast as IR every 'L' seconds. This increases the length of IR and the bandwidth cost.

In the AT scheme, the IR is broadcast every 'L' seconds but only the items updated since last IR are included. The client keeps a variable 'T_l' that indicates the timestamp of last report received. If the difference between the current report timestamp 'T_i' and 'T_l' is more than 'L', (which means that client missed a report) then the entire cache is invalidated; otherwise the client uses the current report to check the validity of items in its cache. In this case the client has to invalidate its entire cache if it misses a single IR. Again problem here is choosing the period 'L'. If 'L' is too short then the time client can be disconnected and still retain the cache is too short. If a large 'L' is used, the query latency increases as the client has to wait for the next IR before it can determine whether the query can be answered locally or not.

The SIG scheme uses a set of combined signatures to verify whether an item in the cache is valid or not. Signatures are checksum computed over the value of items and combined signatures are Exclusive OR of individual checksums. In the mobile environment, before transmission, subsets of data-items, contained in each combined signature are universally agreed upon. A server periodically broadcasts a set of combined signatures at the fixed time interval 'L'. In addition a mobile client also keeps set of combined signatures for the items of interest. Then, the client compares two sets of signatures and an item is declared invalid if it belongs to 'too many' unmatched signatures. However, using this scheme there is a chance that an invalid cache item is not detected. Also the probability that valid items are marked invalid is non-zero. Hence, the main drawback of this scheme is that cache accuracy is not guaranteed and many items are invalidated because of false hits. Also, query latency is high because of a long transmit interval 'L'.

A modification of TS and AT scheme which includes cache validation on reconnection was proposed in [19]. It requires the server to maintain a window 'W' of past broadcasts. This introduces the same basic question how to choose 'W'. Also it increases the uplink traffic as the client has to contact sever after each disconnection and verify its cache validity.

To reduce the query latency with TS scheme, Updated Invalidation Report (UIR) concept was introduced in [14]. This scheme involved sending small UIRs between two big IRs. Since the UIRs did not have time stamps, there was lesser overhead to process them. To answer a query, the client needs to wait for the next IR or UIR, whichever arrives earlier.

Based on the received IR or UIR, the client determines whether its cache is still valid or not. If there is a valid cached copy of the requested data item, the client returns the data item immediately. Otherwise it sends a query request to the server. In case when client requests a data item, server enters the request into a broadcast list. After every IR, it checks the number of clients that have requested the particular item, if this number is beyond a particular threshold; server broadcasts the particular data items. This method is used to utilize the broadcast bandwidth effectively.

4.2 Adaptive Invalidation Schemes

The problem with time-stamp based algorithms is that it requires the window size 'w' which can not adapt to workload parameters. Adaptive invalidation methods are those that adapt themselves to load characteristics of the network as well as query access patterns of various clients served by the server [2].

One way to make the invalidation algorithm adaptive is to vary the speed of transmission; all the clients can not receive data at the same pace. Also there are times when the network is overloaded, and then it becomes server's responsibility to transmit data at an appropriate rate [3, 15]. The feedback mechanism can be used to find out the optimal transmit rate at a particular time instant 't'.

In Bit-Sequences (BS) approach [22], invalidation report consists of a set of bit sequences, each with corresponding timestamp. Each bit in the bit sequence corresponds to a data item. If the bit is '1' it means that the data item has been updated at the server since the timestamp specified in the bit sequence. If the bit is '0', the item has not been updated since the timestamp. The set of sequences are arranged in the hierarchical manner starting with B_n with n-bits, with number of bits in the sequence decreasing as we go down the hierarchy. In general, maximum of (n/2) bits can be set to '1' in B_n. As the bit sequence is received by client, the client invalidates the items based on the corresponding '1' values. A client will invalidate its entire cache only when more than half the bits are set to '1'. Since this algorithm does not depend upon the parameter 'w', it is more flexible. Although when the disconnection is short or rare, this scheme is overkill as it causes clients to spend more time in active mode for receiving invalidation reports.

Adaptive invalidation schemes in [23], proposed combination of TS and BS called Adaptive Invalidation Report with Fixed Window (AFW). In AFW, BS is broadcast as the next invalidation report only if there is atleast a single client which needs more history update information than the window 'w'. Otherwise, window 'w' is broadcast whenever it is possible to save the bandwidth of the broadcast channel.

In another scheme proposed in [23] called Adaptive Invalidation Report with Adjusting Window (AAW), authors argue that even if the disconnection window is barely greater than 'w', use of BS scheme wastes bandwidth, so it feasible to adjust the window 'w' in the TS algorithm. However, if the disconnection window is huge it makes sense to switch to a BS algorithm.

4.3 Asynchronous Invalidation Schemes

Khurana et al [1, 13] presented a cache maintenance scheme, called AS (Asynchronous Stateful), suitable for wireless mobile environment. This scheme integrates mobility management scheme of Mobile IP with cache maintenance scheme used in Coda file system. The basic setup here consists of a home Mobile Support Station (MSS) which maintains for each Mobile Host (MH), a data structure called Home Location Cache (HLC). This data structure stores the latest time-stamp for each data item cached by the MH. When a mobile client changes its cell, the HLC is copied from the previous cell's MSS to new cell's MSS.

The proposed scheme AS uses asynchronous invalidation reports (call-backs) to maintain cache consistency i.e. reports are broadcast by the server only when some data changes, and not periodically. In this scheme the client can continue to use its cache even after a period of disconnection from network, without the need of discarding the entire cache. This scheme has the advantage that there are no periodic broadcasts and so query latency is low. The disadvantages are that there is overhead of keeping a HLC at each MSS. There is much more uplink traffic as the MH needs to tell the MSS what items it is caching and the MH has to probe the MSS each time it wakes up to get all the invalidation messages. Since the invalidation messages are on per MH basis and not on a broadcast channel the bandwidth utilization low.

One of the assumptions made in the paper is that no message is lost due to communication failure or otherwise in the wired network i.e., whenever any data item is updated anywhere in the network, an invalidation message is sent out to all Mobile Switching Stations (MSS) via the wired network; thus, when a mobile host is roaming, it gets the invalidation message if it is not disconnected. However if the message about modification of data is lost, it can result in inconsistent cache.

4.4 Validity-scoped Invalidation Schemes

Xu et al [17] provide three invalidation methods suitable in location dependent environment. The idea is to make use of *validity information* of data items. The server delivers the validity scope analog with the data-item value to client. The coverage area is divided into cells, identified by a cell identifier (CID). The CID is periodically broadcast to all mobile clients in the corresponding cell.

In the first scheme, *Bit Vector with Compression* (BVC), the complete validity information, i.e., complete set of cells in which the data value is valid, is attached to a data item value. BVC uses a bit vector to record the valid scope. A '1' in nth bit means the value is valid in nth cell, '0' means it is not. The client uses this information to determine the validity of data item. The method assumes that there is a possibility where certain data-value is valid in a particular location but becomes invalid as soon as user changes location. The overhead of BVC is very large if there are a large number of cells.

To reduce the large overhead in BVC, a *Grouped Bit Vector Scheme* (GBVC) can be used. Under GBVC, the whole geographic area is divided into disjoint districts and all cells in a district form a group. Validity information attached to a cached data value is

represented as a vector of the form (group-id, BV). While delivering the data value to a client, the server attaches BV, since the group id can be inferred from current CID. When a mobile client checks the validity of a data item value, it listens for the current cell's ID i.e., (group-ID_c, intra-group-ID_c) and compares the group-ID_c, with one associated with the cached data. If they are not the same the data is invalid. Otherwise the client checks the intra-group-ID_c^{th} bit in the BV to determine whether the cached item is valid. The group size is crucial for the performance of this strategy. If group is too large, then overhead is still significant. If it is too small then invalidation rate becomes too high since the data value is regarded as invalid when outside of original group.

The third scheme *Implicit Scope Information* (ISI) attempts to reduce the validity information size at expense of complexity of validation procedure. The server enumerates the scope distributions of all items and numbers them sequentially. For any data item i, its valid scope is specified by a 2-tuple (SDN_i, SN_i), where SDN_i is scope distribution number and SN_i is the scope number within distribution. In this case the size of validity information is small and independent of number of cells in which the value is valid. Also many data items may share the same scope distribution, so number of scope distributions could be much smaller than the number of items in database.

All three of these methods only consider the invalidation of data item when moving from one cell to another. How to efficiently invalidate items based on updates is not in scope of these. The good thing about these methods is they use a cell based geographic distribution which matches most of the current cellular networks.

4.5 Comparison

Most of the research in the invalidation mechanism has focused on modifications to TS scheme. Time stamp based invalidation reports, are only broadcast periodically. Due to this either the client has to wait till next IR (or UIR) or send the query request to the server. This waiting result in long query delay and sending request to the server doesn't make use of cache and increases bandwidth cost. Adaptive scheme needs a good measure to determine what speed to transmit at and how often to transmit. Adaptive schemes fit well for applications that require high refresh rates for the data, for e.g.: in a stock application. Also when the application is configurable by the user, it makes more sense to employ an adaptive scheme as different users will query different items. Asynchronous scheme of [13] requires maintaining a HLC on the MSS. This causes lot of uplink traffic. Also it assumes that a MSS is available for each cell. This might not be true for all applications. Validity based scheme only deal with items being invalid on change of location and not because of update on server. To maintain the cache consistent, the cache invalidation scheme should be performed for both temporal dependent and location dependent updates.

5 Cache Replacement

Cache size is usually limited and in case of mobile clients it is much more limited. When the cache is full, to accommodate a new item into the cache, a replacement candidate needs to be identified. If there are items in the cache that have expired, they can easily be

replaced. But there might be situations where there are no or not many expired-items to provide adequate space for new item(s). In such cases a policy is needed to identify the replacement victims.

Conventional caching schemes use popular methods like least recently used (LRU), LRU-k and most recently used (MRU). However, all these schemes are page-based because of logical mapping made by database to the physical storage. The performance of replacement policies depend upon the type of queries initiated and the application environment. Due to inherent differences between mobile environment and conventional databases, conventional cache replacement policies can not be used here. Also, in a mobile environment, the location of client (i.e., mobile hosts) changes often, which means that database items that it is interested in might change over time. So the replacement policy needs to take into account the changing access patterns of queries [6]. In the following sub-sections we discuss various cache replacement policies for mobile environments.

5.1 Access Frequency-based Schemes

Leong and Si [2, 7] proposed new cache replacement strategies better suited for mobile environments. The *mean* scheme, proposed in [2], measures the access frequency of each object. Access frequency can also be measured in terms of mean inter-operation arrival duration. A cached object is replaced if it has the highest mean duration among all objects. A limitation of this scheme is that it reacts slowly to changes.

In the *window* scheme, proposed in [7], each object has its own window 'w' which stores the access time of past operations. The cached object with highest mean duration within the window is replaced. A problem for the window scheme is the amount of storage needed to maintain the windows.

Their third scheme measures the exponentially weighted moving average of durations such that recent durations have higher weights and the weights tail off as the duration becomes aged. It therefore adapts better to the changes in access patterns. This scheme is called *Exponentially Weighted Moving Average* (EWMA) scheme. Experiments suggest EWMA results in 5-10% higher cache hit ratio.

5.2 Gain-based Cache Replacement Policy

Cache hit ratio is not a good performance metric in case the cache objects are not of same size [20]. The item with larger data size should be replaced because the larger data size item has longer service time and removing larger data size item will provide more space in cache to accommodate more data items. The SAIU (Stretch Access Rate Inverse Update –frequency) replacement policy, proposed in [20], is a gain based cache replacement policy. In this scheme, a gain function is calculated for each item. The item with minimum gain value is chosen as the replacement candidate. This replacement policy takes into account the following criteria while selecting the data to be replaced:

- Low access probability
- Short data retrieval delay
- High update frequency

- Large data size for replacement

The gain function is defined as: $gain(i) = L_i . A_i / (s_i . U_i)$

Where:

- L_i : data retrieval delay on the broadcast channel for item i
- A_i : Access rate for data item i
- s_i: size of data item i
- U_i: Update frequency for data item i

Each of these parameters is computed during the running duration of the system. Unless there is sufficient history for an object to compute these parameters, this caching scheme doesn't perform well.

The Min-SAUD (Minimum Stretch integrated with Access rates, Update frequencies, and cache validation Delay) scheme [7] is an improvement over SAIU scheme as it accounts for the cost of ensuring cache consistency before each cache item is used. Hence, it adds data item cache validation delay to the four factors mentioned above. The modified cache replacement formula, using a gain function based on the five parameters, is defined as:
$gain(i) = (p_i/s_i) (b_i/(1+x_i) - v)$

Where:

- p_i: access probability of data item i
- s_i: size of data item i
- b_i: retrieval delay from the server (i.e., cache miss penalty) for data item i
- x_i: the ratio of update rate to access rate for data item i
- v: cache validation delay, i.e., access latency of an effective invalidation report

Min-SAUD employs on-demand broadcasts for data dissemination and assumes that the access latency of the cache is zero since it is negligible compared to the access latency to the server.

Both of these schemes try to minimize a metric *stretch*, which is defined as the ratio of the access latency of the request to its service time, where the service time is defined as the ratio of the item size to the broadcast bandwidth.

5.3 Target-driven Cache Replacement Policy

The algorithm devised in Min-SAUD scheme is for strong cache-consistency model and minimizes 'stretch'. Although value function used in Min-SAUD is optimal, but it does not discuss how to arrive at such an optimal value function. Each of the value functions optimizes only one value (target parameter). In target-driven cache replacement policy [24], a generalized cost function is introduced, which can be modified to get an optimal gain function corresponding to different performance parameters.

Using the generalized value functions, authors arrive at the specific functions for two targets namely: minimizing the query delay and minimizing the downlink bandwidth. However, this generalized mechanism still does not answer the question: how to decide a replacement candidate set of data items given more than one optimization criteria.

5.4 Comparison

In Access frequency based scheme, the mean scheme is not very useful because it reacts slowly to changes so it cannot take advantage of bursty nature of arrivals. The window scheme is better than mean scheme but it requires amount of storage to maintain the window and with a large window it is no different than mean scheme. The EWMA scheme perform better because it keeps knowledge about past and is adaptive to consistently achieve the best performance even at bursty arrivals.

The multiple attribute based schemes use complex formulas to find the right candidate to be replaced. The disadvantage is high computation cost of weight, the cost function dependent on metric trying to minimize and the fact that it takes a lot of history to compute the cost function correctly. Although the target-driven scheme derives the gain function based on the parameter that needs to be optimized, it still suffers from the overhead of high computation cost. This scheme, however, can be easily customized to different application needs for e.g.: for applications that require quick response time, the gain function that minimizes response time is used to pick replacement candidates.

6 Conclusion

In mobile environment a mobile client accesses a stationary database server via wireless channels that have limited bandwidth and are vulnerable to disconnections. Caching of frequently accessed data items in a client's local storage can reduce contention on narrow bandwidth wireless channel. Since mobile environment is much different than the conventional client server architecture, conventional caching techniques cannot be directly applied to it. We have given a summary of the work done in caching in mobile databases by dividing it into three parts: cache granularity, cache coherence strategy and cache replacement policy. Page-based caching granularity does not work very well in mobile databases as it wastes a lot of bandwidth. So there is a need of adjusting the granularity of caching based on the application's typical requirements for e.g.: caching at the object level is good for applications that utilize all the attributes of an object. Similarly, robust cache coherence and cache replacement algorithms have increased importance in mobile environment as there are frequent and random disconnections between server and clients. A lot of mechanisms have been proposed which depend upon invalidating the data based on timestamps, lately there has been emphasis on designing cache coherence strategies depending upon needs of set of applications like applications that depend upon location dependent data. Cache replacement policies devised for mobile databases take into consideration the access patterns to conserve the storage space which is a scarce commodity in a mobile device.

We believe that the area of cache replacement has not received much attention. Lot of work needs need to be done in this area, especially in case of location dependent data, to find better replacement policies. Work is also needed in area of cache granularity to better predict which attributes to cache and also to move beyond object-oriented databases. In the future work, we should be able to answer which combination of techniques can co-exist to meet the needs of particular set of applications and if any of these algorithms are implement in commercial mobile applications.

References

[1] Kahol, S. Khurana, S.K.S. Gupta and P.K. Srimani, "*A strategy to manage cache consistency in a distributed mobile wireless environment*", IEEE Trans. on Parallel and Distributed System, 12(7), pp 686700, 2001

[2] Hong V. Leong, Antonio Si "*On Adaptive Caching in Mobile Databases*" April 1997, Proceedings of the 1997 ACM symposium on Applied computing

[3] Alok Madhukar, Reda Alhajj "*An Adaptive Energy Efficient Cache Invalidation Scheme for Mobile Databases*" April 2006, Proceedings of the 2006 ACM symposium on Applied computing SAC '06

[4] Ken. C.K. Lee, H.V. Leong, Antonio Si "*Semantic Query Caching in Mobile Environments*", June 2005, Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access

[5] Chi-Yin Chow, Hong Va Leong, Alvin T. S. Chan "*Distributed Group-based Cooperative Caching in a Mobile Broadcast Environment*" May 2005, Proceedings of the 6th international conference on Mobile data management MDM '05

[6] Boris Y. L. Chan, Antonio Si, Hong Va Leong . "*Cache Management for Mobile Databases: Design and Evaluation.*" 1998, Proceedings of the Fourteenth International Conference on Data Engineering

[7] Jianliang Xu; Qinglong Hu; Wang-Chien Lee; Dik Lun Lee; "*Performance evaluation of an optimal cache replacement policy for wireless data dissemination*" 2004, IEEE Transactions on Knowledge and Data Engineering

[8] Quen Ren, Margret Dunham, "*Semantic Caching in Mobile computing*" (2001)

[9] Baihua Zheng, Dik Lun Lee, "*Semantic Caching in Location-Dependent Query Processing*", Lecture Notes in Computer Science (2001)

[10] D. Barbara, "*Mobile Computing and Databases—A Survey"*, IEEE Trans. Knowledge and Data Eng., vol. 11, no. 1, Jan./Feb. 1999.

[11] Kristian Kvilekval, Ambuj Singh, "*SPREE: Object Prefetching for Mobile computers*", (2004), Distributed Objects and Applications (DOA)

[12] D. Barbara, T. Imielinski, "*Sleepers And workalcholics : Caching strategies in mobile computing*" , Proc. ACM SIGMOD Int'l Conf. Management of Data, vol. 23, no. 2, May 94

[13] A. Kahol, S. Khurana, S. K. S. Gupta, P. K. Srimani, "*An Efficient Cache Maintenance Scheme for Mobile Environment*" ,(2000) International Conference on Distributed Computing Systems

[14] G. Cao, "*A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments*," IEEE Trans. Knowledge and Data Eng., vol. 15, no. 5, Sep 03.

[15] Yeung, M.K.H.; Yu-Kwong Kwok , "*Wireless cache invalidation schemes with link adaptation and downlink traffic*", IEEE Transactions on Mobile Computing, 2005, vol4 issue 1.

[16] S. Acharya, M. Franklin, and S. Zdonik, "*Dissemination-Based Data Delivery Using Broadcast Disks,*" Personal Comm., vol. 2, no. 6, Dec. 1995.

[17] J. Xu, X. Tang, D. L. Lee ," *Performance Analysis of Location-dependent Cache Invalidation Schemes for Mobile Environments*", 2002.

[18] H. Song; G. Cao, "*Cache-miss-initiated prefetch in mobile environments*", IEEE International Conference on Mobile Data Management, 2004 Page(s):370 – 381

[19] K.-L. Wu, P. S. Yu, and M.-S. Chen. "*Energy-efficient caching for wireless mobile computing*". In 20th International Conference on Data Engineering, pages 336-345, Feb. 26-March 1 1996.

[20] Xu, Q.L. Hu, and D.L. Lee, W.-C. Lee, "*SAIU: An Efficient Cache Replacement Policy for Wireless On-Demand Broadcasts*", Proc. Ninth ACM Int'l Conf. Information and Knowledge Management, pp. 46-53, Nov. 2000

[21] Chun-Hung Yuen, J.; Chan, E.; Lam, K.-Y.; Leung, H.W. "*Database and Expert Systems Applications*", 2000. Proceedings. 11th International Workshop on 4-8 Sept. 2000 Page(s):155 - 159

[22] J. Jing, A. K. Elmargarmid, S. Helal, and R. Alonso. "*Bit-Sequences: An Adaptive Cache Invalidation Method in Mobile Client/Server Environments*". ACM/Baltzer Mobile Networks and Applications, 2(2), 1997

[23] Q.L. Hu and D. L. Lee. "*Adaptive cache invalidation methods in mobile environments*". In Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing, August 1997.

[24] L. Yin, G. Cao, and Y. Cai. "*Target-Driven Cache Replacement for Mobile Environments*", Journal of Parallel and Distributed Computing, Vol. 65, pp. 583-594, 2005