A New Model for Location-Dependent Semantic Cache Based on Pre-Defined Regions

Heloise Manica
Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis - SC, Brasil
Heloise@inf.ufsc.br

and

Murilo S. de Camargo
Departamento de Informática – Universidade de Brasília (UNB)
Brasília - DF, Brasil
murilo@cic.unb.br

and

Ricardo R. Ciferri, Cristina D. A. Ciferri
Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá - PR, Brasil
rrciferri@uol.com.br

Abstract

Mobile Computing is an emerging paradigm that provides to mobile clients the capability of accessing information anywhere and anytime. Data Management in this paradigm poses new challenging problems to the database community. New research problems include management of location dependent data. Location-Dependent Services (LDS) is an emergent application that allows new types of queries such as location-dependent queries and continuous queries. Data caching plays a key role in data management due to its ability to improve system performance and availability limitations. However, data cached in LDS can become obsolete when a mobile client moves from a location to a new one. The spatial property of location-dependent data opens up new challenges and opportunities for data caching research. The cache management requires more than the traditional solutions because mobility and location must be addressed. In this paper, we first provide a review of the existing approaches for data cache management in location dependent systems. Secondly, we propose a new model for location-dependent semantic cache. For this model, we present a new cache organization based on pre-defined regions, an improved cache replacement policy called ASCR and a strategy to build new semantic segments for LDS.
Keywords: Database**,** Cache Management, Mobile Computing, Semantic Cache.

1. INTRODUCTION

Advances in mobile computing and wireless networks provide flexibility in accessing and manipulating information. Database servers and mobile clients communicate via wireless networks without restricting users to specific locations. While this technology is revolutionary in its approach to managing data, many issues remain to be addressed. Traditional approaches may not perform well and more effective solutions by taking into account the impacts of mobile computing must be developed.

Data caching plays a key role in data management due to its ability to improve system performance and availability limitations. Undoubtedly, caching is advantageous to such environment since it helps to save power consumed with server communication. If queries results can be obtained locally, server communication is not required. Besides that,

wireless links are relatively unreliable and limited. This implies that the traffic should be reduced to improve system performance.

Another relevant issue is the client disconnections that can be voluntary or not. The cache technique helps to deal with disconnections since mobile users are still able to work using the cached data when the network is unreachable.

Mobility raises new requirements for data management. The transaction model, query processing mechanism, consistency control and recovery for the traditional distributed systems have to be adapted to be used in mobile environments [9].

Mobility also makes possible the development of new classes of applications such as Mobile Location-Dependent Information Services (LDIS). Nowadays, these applications are becoming popular in the mobile computing environment. Through these services mobile clients can access location sensitive data such as local traffic report, hotel and restaurant information, emergency services, etc. These access data, called Location Dependent Data (LDD), are related to their geographical position, and the queries that issue for them are named Location-Dependent Queries (LDQ). LDQ usually originates from moving objects, whose locations determine the results of these queries. "Find the closest restaurant within 10 miles" is an example of a LDQ. Answering this type of query requires the client's current position. The client's geographical position is provided by a Location Service or GPS-solution.

Similarly, the caching technique is also crucial for LDD applications. Moreover, data cached in LDIS can become obsolete when a mobile client moves from a location to a new one. Then, the cache management requires more than the traditional solutions because mobility and location must be addressed. There are important works in cache management for LDD ([2], [10], [11], [12]). However, how to choose an effective caching model still needs further study. The spatial property of location-dependent data opens up new challenges and opportunities for data caching research.

There are two issues involved in client cache management: cache invalidation that maintains data consistency between the client cache and the server; cache replacement policy that determines which data should be deleted from the cache when it does not have enough free space to accommodate a new item [12]. In recent years, cache consistency has been extensively studied; however most of the previous works have studied the cache invalidation problem incurred by data updates (temporal invalidation).

Since clients are not fixed equipments, the data stored in client's cache may be obsolete not only due to updates performed on data items but also when the client moves from one location to another. Spatial invalidation occurs when the data values stored in the client's cache become invalid because the client has moved to a new location area.

The maintenance of a valid cache when the client moves is called Location-Dependent Cache Invalidation and a data item can have different value depending on its location [12]. When the cache is full, a cache replacement policy has to be used to identify and delete the data that is most unlikely to be used again. In LDIS, traditional policies such as LRU and LFU are not suitable and must be adapted.

To better explore the spatial property, the semantic cache model has been studied for mobile computing environment ([1], [4], [7], [8]). This significant model and its advantages for mobile computing are discussed in section 4.

A comprehensive discussion and comparison of caching strategies for general mobile computing systems can be found in [5]. In a previous paper, we have presented a wide review in the cache management area. This paper extends our previous work to address caching issues for LDIS. The main contributions of our work are summarized as follows.

We first provide a review of the existing approaches for data cache management in location dependent systems. The review provides a basis for identifying strengths and weaknesses of individual methodologies, as well as general guidelines for future improvements and extensions.

Secondly, we propose a new model for location-dependent semantic cache. For this model, we present a new cache organization based on pre-defined regions, an improved cache replacement policy called ASCR and a strategy to build new semantic segments for LDIS.

The remaining of this paper is organized as follows. Sections 2 and 3 present location-dependent cache invalidation and replacement strategies proposed in the literature for the traditional page cache model. Another important solution, the semantic cache model is presented in section 4. In section 5 we propose a new semantic cache management based on geographic information that improves the query processing and the cache hit. Finally, section 5 discuss our proposed model and concludes this paper.

2. LOCATION-DEPENDENT CACHE INVALIDATION STRATEGIES

In a location-dependent information system, a data item can show different values for different geographical locations. That is, the answer to a query depends on the location where the query originates. This section and section 3 present location-dependent cache management strategies that are suitable for the physical cache model. In the physical data storage model, the MU cache contents are copies (tuple or page) of data items from the server. Page caching is a traditional approach and widely used in client-server systems [3].

Otherwise, another important model is the logical cache where arbitrary query answers are stored in the client's cache. Different from the physical model, the data is retrieved from the server using queries. This requires more processing capability at the server, but only the required data is transmitted over the wireless link. This type of cache model is described on section 4.

Following, we discuss location-dependent invalidation strategies. A common way to perform location-dependent cache invalidation is to attach validity information (named valid scope or valid area) to the data values in the cache. The valid scope of an item value is defined as the geographic area in which the item value is valid. There are different forms to relate the data value with its valid area. The form to represent the valid scope depends on the location model employed. In the symbolic model the valid scope is represented by a set of logical identifications (IDs), e.g. the ID of a cell in a cellular communication system. A geometric model represents a valid scope by geographical coordinates of the area, e.g. a polygonal.

Efficient cache invalidation methods are critical to the whole system performance. The attached information provides a way to check the validity of cached data with respect to a certain location without communicating with the server. Besides, the invalidation information can be used by cache replacement policies.

In [11] the validity information of a data item is the set of cells (symbolic model) within the item is valid. It is assumed that a client cache is logically organized and each cache entry contains a data item ID, attached validity information if any, and a pointer pointing to the real data.

The server delivers the valid scope along with a data item value to a MU. The client caches the data as well as its valid scope for later validity checking. For validity information organization [11] proposed three methods: Bit Vector (BV), Grouped Bit Vector with Compression (GBVC), and Implicit Scope Information (ISI).

The method BVC considers that each cell has identification (CID) and it uses a bit vector to record scope information. The bit vector length is equal to the number of cells in the system and all data in cache is associated to a bit vector to record the valid scope. This way, the value 1 (one) in the nth bit means that the data item value is valid in the nth cell while 0 (zero) means that it is invalid.

Whenever a data item value is required for location-dependent validation, the client listens to the broadcast for the current cell's ID, and uses it to examine the cached bit vector of the data item value. When the system is very large, the overhead is large.

To remedy the overhead in BVC, the GBVC only store information about cells that are adjacent or near MU current location. The model proposes the division of the wide geographical area of the system into groups and intra-groups. The cell ID consists of two parts: group ID and subgroups ID.

When a mobile client checks the validity of a data item value, it listens for the current cell's ID, i.e., (group-IDc, intra-group-IDc), and compares the group-IDc with the one associated with the cached data. If they are not the same, the data is invalid. Otherwise, the client checks the intra-group-IDcth bit in the bit vector to determine whether the cached data is valid.

The Implicit Scope Information model divides the database into multiple logic sections. Data items with the same valid area are placed at the same section. The data item in cache will have the format {Di, SDNi and SNi}, where Di is the data item value, SDNi is the section number, and SNi the data number inside the section (scope number).

In [12] is described tree schemes for representing valid scopes: Polygonal Endpoints (PE), Approximate Circle (AC) and Caching-Efficiency-Based (CEB).

The PE strategy records all endpoints of the polygon representing the valid scope of a cached data item. When the number of endpoints is large, this technique will consume a large portion of the wireless bandwidth and space for caching the valid scope in the client. The advantage is the complete knowledge of the valid scopes.

Another alternative is the utilization of an approximate circle (valid area circle) inserted inside the original polygon (v1' in figure 1). Thus, the valid area will be the approximate area defined by the center and radius of the circle. A problem occurs when the polygon shape is thin and long. In this case, the approximating error will be high and the cache can consider valid data as invalid if the query is outside the circle.

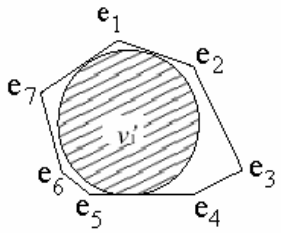

Figure 1: Possible Valid Areas. From [12]

Caching-Efficiency-Based is a generic method for balancing the overhead and the precision of the valid scopes to be attached. CEB generates a candidate valid scopes set, and then select the best one.

3. LOCATION-DEPENDENT CACHE REPLACEMENT STRATEGIES

Due to limitations of the client cache size, it is impossible to hold all the accessed data items in the cache. As a result, cache replacement algorithms are another important issue to be handled.

The MU location changes open up new research issues in replacement policies. In LDIS, the cache replacement policy must consider other factors besides the traditional factor "access probability". Policies such as LRU, LFU, and LRU-K are not suitable for this application.

When mobile client moves around, other factors must be considered such as valid scope, distance and direction. Valid scope represents the geometric area in which the data value is valid. A common way to perform location-dependent cache invalidation is to attach the valid scopes to the data values returned to the client [10]. The larger the valid scope area of the data, the higher the probability that the client requests this data.

In LDIS, the server answers queries according to the client's location, then the distance is an important factor to be considered. When the valid scope of a data value is far away from the current client's location, this data will have a lower chance to become useful. The computation of the distance between client's location and the data valid area can change according to the kind of application. In a rural zone for example, we can use the Euclidean distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. However, in an urban area this formula is not suitable because the MU can move through the streets with buildings or other obstacles.

Another used factor is the direction. It can be used first to eliminate from the cache the data that are in the opposite direction of the client's movement. To use this factor a mobility model is required to represent the locations and moving behaviors for mobile users.

Next we briefly present some cache replacement policies for location-dependent systems. These methods were proposed for physical cache model (page or tuple cache).

The Probability Area (PA) proposed in [12] defines the data items to be replaced according to a cost function defined as the product of the access probability of a data item and its attached valid scope. The cost function of a value j of item i is: $C_{i,j} = Pi \cdot A(v_{i,j})$, where Pi is the access probability of the item i and A(v'i,j) is the area of the attached valid scope for a value j of an item i. When the data replacement is performed, this policy selects the data with the least cost (s).

In the Probability Area Inverse Distance (PAID) policy [12], the cost function of a value j of an item i is given by C i,j = Pi . A(v'i,j) / D(v'i,j) where Pi and A(v'i,j) is defined in the same way as above, and D(v'i,j) is the distance between the current location and the valid area v'i,j. When data replacement is carried out, PAID ejects the data value (s) with the least cost (s).

To consider the movement direction, the authors in [12] take extensions of the model PAID: PAID-U (Probability Area Inverse Distance - Unidirectional) and PAID-D (Probability Area Inverse Distance - Directional). In PAID-D, the distance is calculated considering the client's current direction of movement. PAID-D keeps the data that are in the direction of the client's movement. On the other hand, in PAID-U the distance is computed regardless of the current direction of the client's movement.

In urban area, it is more difficult to find out buildings such as post office and hotel, is more general than issuing in rural area. Consequently, good services cannot be provided for users without considering the characteristics of urban area [13]. The Manhattan Distance policy was proposed by Jung et al. for location dependent queries in urban area. The distances in urban zones are given by |(x1-x2)|+|(y1-y2)|. The proposed algorithm computes the Cache Replacement Score (CRS) based on the MD computation. It chooses the victims according to CRS by the current location of the MU. Thus, the victims who are farthest from the MU will be replaced first.

4. SEMANTIC CACHING

As discussed before, the client cache technique plays an important role in mobile computing. Recent studies show that the semantic cache model (SC) is an attractive approach for mobile computing ([1], [4], [7], [8]).

The SC idea is to maintain in the client's cache both the semantic descriptions and associated answers for previous queries. The semantic information is very useful to organize and to manage the client's cache.

Affinity refers to the kind of relationship between the data items in the cache. This relationship can be temporal (temporal locality) or semantic (semantic locality). Temporal locality means that items referenced recently are likely to

be referenced again in a near future. Semantic locality means that if an item has been referenced, other items with the same semantic function (for instance, the nearest) are also likely to be referenced.

Semantic Cache is very useful in applications where data items do not change frequently and, hence, clients can cache data items and use them to serve queries locally. However, if the data are frequently updated, caching may not be helpful. In this case, broadcasting the data on the air may be a good solution.

The query processor uses the semantic descriptions to determine which data is available in the cache and which ones will have to be requested to the server. The semantic description is also utilized in the definition of the cache replacement policy, not requiring any additional attached information to each tuple as in traditional cache management systems.

In a basic approach, the semantic segment is represented by the set (SR, SA, SP, and SC), where SR and SA define the relation and the involved attributes. SP denotes the selection condition and SC represents the result (pointer for the result pages). Each semantic segment is associated with a pointer pointing to the first page in the cache memory.

The answer of a query Q can be totally contained in a segment S, partially contained or it cannot be answered for S. If Q can be partially answered by S, it is divided in two parts: probe query, which represents the portion of Q satisfied by S and remainder query, which represents the portion of Q not found in S. This procedure is called query trimming.

After the query splitting by the first segment, the next candidate segment will divide again the remaining query. This process continues until there are no more candidate segments or the reminder query becomes empty. At the end, if the reminder query is not empty it is sent to the database server to be processed. When the server returns the answer, the whole result is composed by all probe and reminder queries. A detailed study of semantic cache management in mobile computing can be found in [8]. The authors explore the semantic caching query processing strategies.

4.1 Semantic Caching for LDIS

Semantic caching is by nature an ideal cache scheme for location-dependent applications due to the following reasons [9]. First, semantic cache is built on the semantic locality among queries, which just fits the LDD applications. Secondly, continuous queries can be incrementally processed by semantic caching. Thirdly, since a semantic cache organizes data by semantic information, it makes cache management more flexible. A page or tuple cache cannot offer such functionality, since the cached data is not associated with any semantic meanings.

With semantic information, different strategies can be developed. If the user provides information about your schedule and route, it is possible to define the exact data items that the user will need later. However, when user-provided information is not unavailable, strategies need to be carefully designed to manage the location-dependent cache.

Most semantic cache invalidation and replacement strategies are built on temporal locality. LRU and MRU are examples of temporal locality invalidation strategies. For LDIS, the semantic locality is more appropriate.

[1] were the first to use semantic distance function for replacement policy. They utilize the Manhattan distance function to calculate the distance of each semantic region from the user's current location. Their proposed algorithm discard regions according to the values computed.

[6] extend the work of Dar et al. to investigate ways in which semantic caching can be used to manage location-dependent data. Their work includes a formal model to represent moving objects and propose strategies of applying semantic caching to location dependent applications. In this model, the semantic locality is highly related with the moving behavior of mobile users.

For cache replacement, Ren and Dunham [6] propose a semantic cache replacement policy called FAR (Furthest Away Replacement). FAR chooses replacement victims according to the user's status. The future location is calculated based on user's current location, moving speed and direction. Then, the segments are classified in two sets: the first one with segments that are in the direction of the movement, and the second with segments that are out of the direction. The victims are always selected from the out-direction set. When the out-direction set is empty, the most distant segments of the in-direction set are replaced. In the next section, we describe our proposed semantic cache model and cache management strategies.

5. LDD Semantic Cache Based on Pre-Defined Regions

The cache technique is only useful when the data cached is used to answer queries. If there is a cache hit, the client can serve the query locally; otherwise, it is necessary to send an uplink request to the server. The higher the cache hit ratio, the higher the local data availability, the less the uplink cost and battery consumption, and the less downlink cost

for query responses of fixed sizes [12]. So, query delay, bandwidth utilization and power consumption are related to the cache hit ratio.

There are techniques which can improve the cache hit ratio. In the prefetch technique, the clients prefetch data that may be used in the near future. Other important strategy to increase the cache hit ratio is the cache replacement policy. Mobility motivates the development of new cache replacement strategies built around location and movement. In a mobile environment, the location-dependent data cache must be replaced taking into account the impact of mobility, since the cache contents are required to move as the mobile unit moves to a new location.

A problem in LDIS is that mobile clients may have different movement patterns. As time passes by, the MU can change its moving behavior. Data distance may or may not affect cache performance, depending on the mobile client's movement and query patterns [12].

Since mobility pattern may change over time, the ideal cache replacement politics for LDD systems varies; it will depend on the client mobility model. Motivated by these needs we propose ASCR (Adaptive Semantic Cache Replacement Approach), a new method for semantic cache replacement.

This research assumes that a MU always moves in 2-dimensional space. To simplify the problem, we consider only location dependent queries on single relations and the supported operator in cache are selections. In the next subsections we describe our proposed model.

5.1 Model Organization

Location dependent queries are constructed with two kinds of predicates: traditional location unrelated predicates and location related ones. For example, the query "Give the hotels in Brasília with price < 100" is a location unrelated predicate, otherwise, the query "Give the hotels within 20 km from my actual position" is a location related. Our work considers the location related ones.

A location dependent query is defined as follows. Given a database D = {Ri}, a location dependent query Q is a tuple (QR, QP, QMBR, QC) where QR∈D represents the relation issued, QP a location predicate, QMBR the minimum bounding rectangle that represents the geographic area that is going to be queried and QC represent the query results.

In semantic cache model, the key structure is the semantic segments (or regions). We organize the answers received from the server in sets named segments. Basically, each segment has a set of tuples, a constraint formula that describes the tuples grouped together within the segment, and a replacement value. Each tuple in cache is associated with only one segment.

Our location dependent semantic model is basically composed by three main parts: the content, the index and the group structure. The content part consists of the queries results. The index part maintains the semantic as well as physical storage information for every cached segment. The index is represented by (S, SR, SP, SMBR, SG, SC, Sts) where S stands by the segment identifier, SR the database relation involved in the query, SP the select condition, SMBR the minimum bounding rectangle (IX=[X1, X2] and Iy[Y1, Y2]) that represents the geographic area the segment, SC the first page address of the segment content, Sts a timestamp, and SG the group. The group structure maintains the set of segments with related geographic position (table 2).

This model differs from other semantic cache schemes in the following aspects. First, we further propose to maintain spatial information, the rectangle that represents the geographic area of the data in the segment. This spatial information will be very useful in cache admission, replacement and query processing. Second, we propose an extra structure named "groups" that represents a geographic region previously defined in the system. In addition to the basic components listed above, other items can be kept for maintenance use.

The semantic cache index is more clearly illustrated through the following example 1. Let's consider tree database relations: Restaurant (Rno, Rname, Rtype, Rx, Ry), Hotel (Hno, Hname, Hvac, Hx, Hy) and Drugstore (Dno, Dname, Dschedule, Dx, Dy). Suppose that a MU issues queries on his path from different positions. The MU position is represented by M(x,y) and the following queries are issued on T1, T2, T3 respectively:

T1 - UM(40, 50): "Give me all hotels within 20 km"
T2 - UM(20, 40): "Give me all restaurants within 5 km and type is Chinese "
T3 - UM(30, 25): "Give me all Drugstores within 10 km"

Suppose that the client cache is empty and these queries results are cached. The index and group structures are formed as demonstrated in table1 and table2.

S	S_R	S_P	S_{MBR}	S_C	S_{ts}	S_G
S_1	Hotel	$(UM_x - 20 \leq Hx \leq UM_x + 20)$ ^ $(UM_y - 20 \leq Hy \leq UM_y + 20)$	[(20,80), (25,85)]	2	T1	1
S_2	Restaurant	(Rtipo = "chinese") ^ $(UM_x - 5 \leq Rx \leq UM_x + 5)$ ^ $(UM_y - 5 \leq Ry \leq UM_y + 5)$	[(15,25), (35,45)]	5	T2	1
S_3	Drugstore	$(UM_x - 10 \leq Dx \leq UM_x + 10)$ ^ $(UM_y - 10 \leq Dy \leq UM_y + 10)$	[(20,40), (15,35)]	8	T3	1

TABLE 1: Example of Semantic Cache Index

G_{ID}	G_{Seg}
A	$S_1 \rightarrow S_2 \rightarrow S_3$

TABLE 2: Example of Group Structure

5.2 Query Processing

The previous section defined our semantic cache model organization. This section describes important practical issues such as: query processing steps and the formation of semantic segments. As discussed before, the results of previous LDQ are cached in MU memory to be reused later. To reuse this information, not only the selecting conditions (predicate) but also the location must be satisfied. When a LDQ is issued, the first step is attaching to the predicate the user current location. After that, an LDD query can be processed normally using different strategies.

To efficiently verify the cache content and locate the candidate segments to answer queries, we propose first to verify which groups are candidates to answer the query. This is done through the analyses of intersected geographic area of the query (rectangle defined by MU current position and the dimension defined on the query predicate) with the geographic area of segments.

The same way, after selecting the candidate groups, the algorithm verifies witch semantic segments are candidate to answer the query. With the selected segments set, then is verified which of them have the same relation and predicate of the query. Finally, the query is executed in the selected segments. To locate candidate segments spatial index can be used.

This procedure is illustrated in example 2 (see Figure 2). Consider that different queries were issued and cached. Suppose that in T8 the client Mx,y issues the query Q= "Give me all hotels within 30 km". First the groups A and B will be selected. Secondly, the segments S3 and S4 are candidates. After verifying the relations between predicates, only S4 is candidate. Note that this case, only a small part of the total result is in cache. The intersected area contains the probe query results. The reminder query is submitted to the server for processing. The advantage of this method is that it avoids searches in segments that cannot answer the query. The spatial analysis eliminates all no-candidate segments.

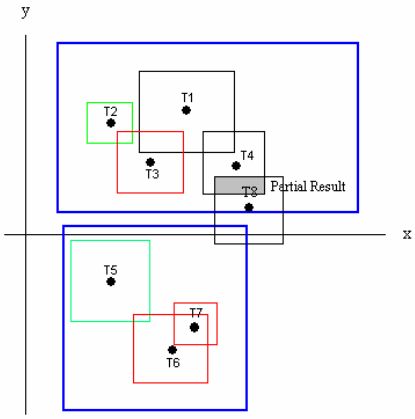

Figure 2: Example of Queries and Moving Path

5.2.1 Semantic Segments Formation

Another important issue related to query processing is the semantic segment formation and management. To avoid redundant data, we cannot keep S and Q in the cache. The way semantic regions are formed determines the granularity of the segments in cache and thus has a significant effect on the performance of the semantic cache. When a query intersects semantic segments in cache, different approaches can be used.

The no coalescing approach generates three disjoint parts: (Q ∩ S), (S ^ ¬Q) e (¬ S ^ Q). The disadvantage is that it may result in a large number of small segments. Besides, for our semantic cache model it would be too much difficult to represent the geographic area of a segment after it was trimmed for queries.

Complete coalescing approach always coalesces the three parts into one larger segment. This approach with small queries (relative to cache size) leads to good performance. However, when the answer takes a large portion of the cache, this strategy can result in semantic regions excessively large. This is not good for replacement due to the fact that a large segment can empty a significant portion of the cache resulting in poor cache utilization [1].

According to [8] the "partial coalescence in query" approach is the best one. The segment is decomposed in two parts: the overlapped part and the no overlapped part. Then, on must coalesce every part of Q together and cache the result of Q.

In this research, we use the "partial coalescence in query" approach and the idea proposed in [1]. An adaptive heuristic where regions formed at the same time were coalesced if either of them was smaller than 1% of the cache size. This way, the client can adjust the coalescing strategy dynamically.

The pre-defined groups represent urban areas. Our cache admission politics consider that when a MU is out of the pre-defined groups, this query should not be cached because those data are about a rural area and probably those data will not be used again.

5.3 Replacement Policy

In a mobile computing environment the users are equipped with mobile equipments which can have different mobility models. An optimal replacement policy discards the segments which will not be used in the future. For this, user-provided information is necessary, such as his movement, queries he is going to ask, future location and so on. Since this information is not always unavailable, the replacement policy must analyze the MU profile and define a better replacement.

One of the main motivations of our work is the desire to use geographic information to the cache management at client. This section describes our new replacement policy ASCR (Adaptive Semantic Cache Replacement Approach), a hybrid method for location dependent cache replacement based on temporal and semantic locality.

Before describing ASCR algorithm, let's consider the following examples 3 and 4 to illustrate different cache replacement scenarios. For both examples consider that the user begins issue location dependent queries in a yellow page database, and the client's semantic cache stores the results of the queries submitted. Consider also that the client's cache space is limited and when there is not sufficient space, a victim must be chosen to be replaced.

Example 3 - Low mobility: The user arrives in a city A, becomes a guest in a hotel and begins to issue queries for tourist information. During some time (maybe hours, days or weeks, etc.) while he visits the scenery he continues issuing LDD queries. When the cache replacement comes up, to eliminating data that are more distant from the user location may not be the best policy, since probably most of the cache content is about the local city. The LRU (least recent used) policy in this case is more suitable, simpler and avoids unnecessary and costly calculus to compute distance, future location and so on.

Example 4 - High mobility: The user is a traveler that visits many places. He begins his route in city A following to the cities B, C, D, an so on. He can issue some queries in all visited cities or not. Also it is possible that he comes back to any visited city. When the cache replacement comes up, for this case it is interesting first to eliminate data that are too distant from the user location. If the most distant data have already been eliminated and free space still is necessary, the next replacement can be done by the LRU policy.

For both exemplified cases, the user has no deterministic path and it is difficult to determine his real future location. ASCR defines the best policy according to the analyses of the previous client's movement. This can be done through the analyses of the group structure (table 2). Trough the MU current position, it's group can be easily identified (MUg). If there is more than one group in the structure, the older segments (ols) from the most distant group (mdsg) are replaced first. Otherwise, when there are segments from a single group, only the LRU is performed. Thus, distance computation is used only when there is more than one group.

To compute the distance between groups and the MU current position, different distance measure can be used. We assume the distance is defined as Euclidean calculus. This procedure is described in the algorithm ASCR (C, M), where C represent the client cache and M the mobile unit.

Algorithm: ASCR (C,M)
{ dsg ← NULL
For each *group* in C {
  If UMg = *group*
   then go to next *group*
   else
    dsg ← *dsg* + *group*}
if *dsg* = NULL
then LRU policy is performed over the semantic segments in cache
else GMD (*dsg*) }

GMD (*dsg*)
While (*dsg* != empty) {
Mdsg ← the most distant group from the *UMg*
For each segment S in *mdsg* {
  ols ← the older segment from the group
  discard *ols* from C
  add data in free space
  if (free space is enough) return (Success); }
if (free space still is enough) then LRU policy is performed over the single remaining group }

6. FINAL REMARKS AND CONCLUSIONS

In this paper we have explored cache management issues for location-dependent applications. We first presented different strategies proposed in the literature for the page and semantic cache model.

Compared to page caching, a semantic caching scheme is more efficient in location dependent applications since it uses semantic information to organize data. Table 4 [9] summarizes the advantages.

Issue	Semantic Cache (SC)	Page Cache	SC Advantage
Communication cost	low	high	Only required data are transferred.
Cache space	low	high	Only data satisfying previous queries are stored in SC
Parallelism in query processing	easy	difficult	Client and server work in parallel
Disconnection handling	efficient	inefficient	More autonomy of client since partial results can be obtained locally.
LDD data management	Efficient	inefficient	The management is done in semantic granularity.

TABLE 3: SC versus Page Caching

We have argued the need of new strategies that consider the spatial information on cache management issues. Based on this need, we present a new cache organization approach based on pre-defined regions, an improved cache replacement policy called ASCR and a strategy for build new semantic segments for LDIS. One of the main motivations of our work is the use of geographic information on client cache management to improve cache hit.

This model differs from other semantic cache schemes in the following aspects. First, we further propose to maintain spatial information, the rectangle that represents the geographic area of the data in the segment. This spatial information will be very useful in cache admission, replacement and query processing. Second, we propose an extra structure named “groups” that represents a geographic region previously defined in the system.

One of the main motivations of our work is the use of geographic information to the cache management at the client. We extend previous work in the following aspects. FAR policy is based on future locations of moving objects. Our model is based on previous locations. We don’t use any mobility model, due to the fact that in a mobile environment

users have different path models. In order to verify the previous movement we use extra data structures. In FAR policy, each time the MU changes his direction, the sets in-direction and out-direction must be recalculated. To increase systems performance, our model avoids distance calculus. When cache replacement is necessary, our model utilizes its profile to decide the best replacement policy. Our politics considers that, besides the client is an equipment that can move, sometimes it can act as a stationary clients. In this case a policy based on temporal locality (e.g. LRU) is expected to perform well. Otherwise, if the client moves to different locations, a policy based on distance function is used.

FAR assumes that a MU moves at a speed that keeps constant during a period of time and may change from time to time. In some cases such as car moving in highways this model is suitable. Otherwise, in general an object can move anywhere with varied speed and direction. For instance, a car moving in an urban area, the velocity and mainly direction changes are very frequent. For future research, we will refine these ideas and use them in an evaluation study.

References

[1] Dar, S., Franklin, Michael J., Jónsson, Björn T. Srivastava, D., Tan, M. Semantic Data Caching and Replacement. Proceedings of the 22nd VLDB Conference Mumbai (Bombay), India, 1996.

[2] Dunham, Margaret H., Kumar, V. Location Dependent Data and its Management in Mobile Databases. Proceedings of the 9th International Workshop on Database and Expert Systems Applications, p.414, August 26-28, 1998.

[3] Franklin, M. J. Client Data Caching: A Foundation for High Performance Object Database Systems. Kluwer Academic Publishers, Norwell, Massachusetts, 1996.

[4] Lee, K.C.K., Leong, H.V. and Si, A. Semantic query caching in a mobile environment. In ACM Mobile Computing and Communications Review, Volume 3, number 2, pages 28-36, April 1999.

[5] Manica, H., Camargo, M. S. Caching Estrategies for Mobile Computing Systems. Proceedings of the 6th International Conference on Enterprise Information Systems. Universidade Portucalense, Porto – Portugal. April, 2004.

[6] Ren, Q.; Dunham M. H. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. Proceedings of Mobicom 2000: 210-221, May 2000.

[7] Ren, Q.; Dunham M. H.. Using Clustering for Effective Management of a Semantic Cache in Mobile Computing. MobiDE – Seattle – USA. ACM 1999.

[8] Ren, Q.; Dunham, M. H.; Kumar, V.. Semantic Caching and Query Processing. IEEE Trans. on Knowledge and Data Engineering, vol 15, n. 1, jan/feb 2003.

[9] Ren, Q. Semantic Caching in Mobile Computing. PhD Thesis presented to Souther Methodist University. May, 2000.

[10] Xu, J., Tang, X., Lee, D. L. and Hu, Q. Cache Coherency in Location-Dependent Information Services for Mobile Environment, Proc. the 1st Int. Conf. on Mobile Data Access (MDA'99), Hong Kong, Dec. 1999, Springer-Verlag LNCS, vol. 1748, pp. 182-193.

[11] Xu, J., Tang, X., Lee, D. L. Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments. TKDE 15 (2) 474-488, 2003.

[12] Zheng, B.; Xu, J.; Lee, D. L. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. IEEE Trans. on Computers, Special Issue on Database Management and Mobile Computing, 51(10): 1141-1153, October 2002.

[13] Jung, Y. Y.; Lee, J. and Kim, K., 2002. Broadcasting and Caching Policies for Location-Dependent Queries in Urban Areas. WMC'2002. Georgia, USA.