A SpatioTemporal Placement Model for Caching Location Dependent Queries

Anand Murugappan and Ling Liu
College of Computing, Georgia Institute of Technology, Atlanta, USA
{anandm, lingliu}@cc.gatech.edu

***Abstract*—Client side caching of location dependent queries is an important technique for improving performance of location-based services. Most of the existing research in this area has focused on cache replacement and invalidation through incorporating some aspects of the spatial and temporal semantics embedded in the location queries, while assuming an ad hoc cache placement. Very few have studied the impact of spatial and temporal validity semantics and the motion behavior of mobile clients on the effectiveness of cache placement and ultimately the performance of the client cache. This paper proposes an adaptive spatio-temporal placement scheme for caching location dependent queries. The cache placement decision is made according to the potential cache benefit of the query results based on the spatio-temporal properties of query results and the movement patterns of the mobile client, aiming at increasing the cache hit ratio. We introduce the concept of 'Overlapping Cache Benefit' as a measure of the hit rate of a cached item, and present three spatio-temporal cache placement schemes, which provide a step-by-step in-depth analysis of various factors that may affect the performance of a client cache in mobile environments. We implemented the spatio-temporal placement model in the first prototype of the MOBICACHE system. Our experimental evaluation shows that the spatial locality and the movement patterns of mobile clients are critical factors that impact the effectiveness of cache placement and the performance of client cache, and the proposed adaptive spatio-temporal cache placement approach yields higher hit ratio and better response time compared to existing mobile cache solutions.**

I. INTRODUCTION

One of the critical issues in mobile environments is the growing demand of scalable solutions for efficient delivery of content and information services. *Location Dependent Queries* (LDQ) form a special class of location-based information services in the sense that location dependent queries are issued by mobile clients on the move, thus the query results are dependent on the movement patterns and the current location of the mobile client, no matter whether target objects of the location queries are still objects such as gas stations, restaurants, or moving objects such as taxi on the road.

Location-dependent information delivery faces many new difficulties inherent to mobile computing environments in addition to the scalability and performance challenges confronted by Web content delivery in wired networks. Caching of frequently accessed data items on the client side is an important general technique that helps address some of these challenges. By storing copies of data objects locally, data accessibility can be enhanced and access cost can be reduced. However, frequent network disconnections, mobility of the clients, and limited local resources on the mobile clients, complicate the provision of information delivery services to mobile users, making location-aware client caching a challenging problem.

We argue that in a mobile computing environment, the cache placement decision should take into account both the temporal validity and the spatial locality of the data items to be placed in the client cache and the motion behavior of the mobile client. Consider a simple example in which a mobile user is driving on the I85 North highway at 60 mph speed, and wants to find the nearest gas stations within certain range. If the query is asking for gas stations within 5 miles, then the spatial validity of the gas stations returned is 5 miles. Obviously, these gas stations will no longer be relevant after 5 minutes of driving. If the query interval of this mobile client is typically longer than 5 minutes, then placing these gas stations in her cache is a waste in terms of both placement cost and replacement/invalidation cost.

Bearing these challenges in mind, in this paper we propose an adaptive spatio-temporal placement scheme for caching location dependent queries, which predicts the potential cache benefit of the query results based on multiple spatio-temporal properties of mobile clients. The main design idea of this new placement model is based on two observations. First, in a mobile environment, some query results may have an extremely low likelihood of being used in the near future due to mobility and spatio-temporal constraints. Thus the ability of identifying such items and dropping them earlier instead of placing them in the client cache and replacing them later can greatly increase the cache hit ratio and improve the overall cache performance. Second, due to the motion dynamics of mobile clients, such as changing speeds and changing query intervals, an adaptive cache placement scheme can better respond to the dynamic motion behavior and changing query requirements of mobile clients.

Our contributions in this paper can be summarized in two folds. First, we present three spatio-temporal cache placement schemes, which provide step-by-step in-depth understanding of various factors that may affect the performance of a client cache in mobile environments. The first one is called threshold based STP scheme, which emphasizes on the importance of combining temporal locality and spatial locality on cache efficiency. The second approach is called Bound-based STP, which highlights the importance of adequate control of the coordination between the spatio-temporal placement and the ad hoc placement. The third one is the speed and query interval

adaptive STP scheme (SQI STP), which stresses the importance of incorporating both motion pattern and query pattern into the cache placement decision. Second, we implement and evaluate the proposed schemes in a prototype system, called MOBICACHE.

The remaining of the paper proceeds as follows. We give an overview of the MobiCache system, including the reference system model and the problem statement in Section II. We describe the three spatio-temporal cache placement schemes in Section III. The experimental evaluation of the proposed solutions are provided in Section IV. We conclude the paper with a discussion on related work and a summary in Section V and Section VI respectively.

II. MOBICACHE SYSTEM OVERVIEW

We briefly describe the reference system model, including the basic notions and assumptions used in the paper, and the problem statement, including two concrete scenarios and the set of critical factors to be considered in making the cache placement decision and how they would impact cache performance.

A. Reference System Model

In this paper we assume a cellular network as the underlying infrastructure. The system consists of mobile clients, stationary servers and mobile support stations. We also assume a geometrical location model where locations are represented in terms of two dimensional coordinates. The clients can move from one location to another and communicate (submit queries and obtain results) with the stationary servers through the mobile support stations.

A Location Dependent Query (LDQ) denoted by Q_i ($1 \leq i \leq n$), is defined as a tuple $< O_i, L_x, L_y, R_i, F_i >$ where O_i is the object type being queried,(L_x, L_y) the location of the mobile client making the query, R_i denotes the spatial range of the query, and F_i denotes the filter condition of the query. For simplicity, all queries are assumed to be associated with a circular range. For example, *<restaurants,-77,38,2 miles, restaurant type = French >* represents the query "Show me all French restaurants within 2 miles radius from my current location (-77,38)".

Queries are issued by the mobile client to the respective servers through its mobile support station. For each moving query over moving objects, the MOBICACHE system on the mobile client will assign a query dependent time-to-live (TTL) timestamp to each query result set returned by the server based on the query range and location, speeds of the mobile target objects. For instance, a moving location query "find the nearest taxi within 5 miles of my current location" will be processed by the taxi service provider in the area and return a list of taxi cars ordered by the distance to the current location of the mobile client. Assuming the typical speed of taxi on the city street is 30mph, the MobiCache system will associate 10 minutes as the maximum TTL for the query results – the taxi objects returned by the server. By default, all query result objects are assigned a system-supplied large value as the maximum TTL. For all moving queries over static objects, such as "find the nearest gas station within 5 miles of my current location", the result objects will have their TTL set to the system default.

B. Problem Statement and Important Parameters

In mobile information delivery systems, ad-hoc cache placement suffers from a number of problems. In this section we first describe these problems and then discuss the set of parameters that are critical for improving the cache performance of a mobile client cache.

The fundamental objective of cache placement is to keep all items that will potentially be requested by subsequent queries. Due to the limited bandwidth, intermittent connectivity, and restricted computing capacity at mobile clients, the cache placement strategy for a mobile client should make effort to avoid placing data items that will unlikely be used by subsequent queries in the near future. There are certain situations where we can predict the potential cache benefit of the query results with reasonable accuracy.

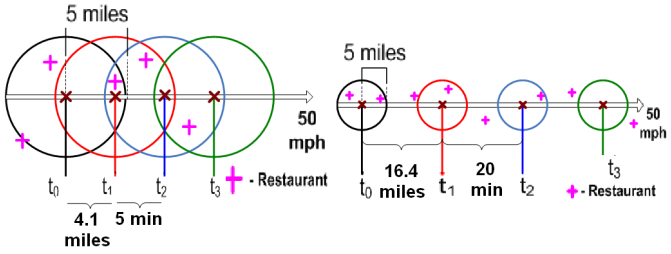

Fig. 1. Importance of spatial locality in cache placement

Figure 1 provides two example scenarios. The scenario on the left side of Figure 1 illustrates an example where the mobile user driving across a city at the speed of 50 mph issues a continuous location query: "Show me all restaurants within 5 miles (of the user's current location) every *5 minutes* in the next hour". This continuous query will be evaluated at the interval of every *5 minutes* (equivalent to every 4.1 miles) from its installation for twenty consecutive times (since the stop condition of this query is 1 hour). Each execution is shown as a circle (with the mobile user at the center) in Figure 1. In such a scenario, caching the query results is beneficial due to the overlap of query results between two consecutive query evaluations. However, if the mobile user issued the following query instead: "Show me all restaurants within 5 miles radius (of the user's current location) *every 20 minutes* for the next hour", then the placement of this query results in the mobile client cache will not generate a cache hit for the next query (and other subsequent queries), because the query ranges of the two consecutive evaluations do not overlap at all, as shown in Figure 1(right). In fact, storing the results of this query in the cache not only generates zero cache benefit but incurs both the cost of placement and the cost of replacement, because placement of such results might trigger replacement of some other entries from the cache that might be more useful. This example scenario amounts to say that when the mobile client

moves far away from the spatial scope of the current query results long before posing the next query, even if the next query is posed on the same object type at the future location, there will unlikely be any spatial overlapping between the current query results and the next query results due to the relatively small spatial validity scope of the query results with respect to the query interval and the movement speed of the mobile client. Hence it is cost-effective if we can detect such scenarios and avoid unnecessary placement.

Another situation where early detection of unnecessary placement is obviously beneficial is when the query results are valid only for a short period of time. A typical example of such scenarios includes those in response to LDQs made over moving objects. Thus, even if the very same query is made from the very same location at a later time the cached results may no longer be valid due to the query rate and the movement patterns of the query result objects. For instance, consider a LDQ issued by a mobile client: "Find all AAA vehicles moving on the road and within 10 miles of my current location". Given that the AAA vehicles in question are on the move, the query results might not be valid for a long time. If the AAA vehicles in question are moving at 50 mph on average, the results will be invalid after 12 minutes. If the predicted query interval for the next query is 20 minutes, then the query results will have expired quite some time ago. Hence, it will be prudent not to place such results into the cache and thus avoid both the cost of placement and the potential problem of replacement of a potentially more useful entry in the client cache. Typically when the temporal validity period of the query results is short with respect to the query interval, the placement decision should be made with care.

By closely examining scenarios like those in above examples, we made two observations. First, ad-hoc placement is not always beneficial in mobile environments. Second, the spatial relationship between the two consecutive query ranges and the distance between the current and the next query locations are critical measures for determining the effectiveness of the cache placement decision. More interestingly, the distance between the two consecutive query locations depends greatly on the query interval and the movement speed of the mobile client.

III. Adaptive Spatio-Temporal Cache Placement

In this section we first describe the concept of spatial area overlapping and the general metric for computing the overlapping cache benefit. We then present an in-depth analysis of the adaptive placement model through the step-wise development of three spatio-temporal placement (STP) schemes: Threshold-based STP, Bound-based STP, and Speed-Interval adaptive STP. Each of these three schemes shows one specific way of measuring the overlapping cache benefit. We describe our experimental evaluation of the effectiveness of these schemes in Section IV.

A. Basic Concepts and Threshold Based STP Scheme

In MobiCache we introduce two basic concepts: spatial area overlapping and overlapping cache benefit. The overlapping cache benefit is defined in the spirit of spatial area overlapping, and is used as the core technique for developing adaptive spatio-temporal placement schemes.

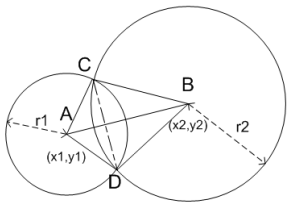

Fig. 2. Accurately measuring the Area of Overlap

Area Overlapping Function: Recall example scenarios in Section II, we show that if there is a possibility that the next query to be made might overlap with the current query range, then the results should be placed in the cache since there is a potential for cache hit. On the other hand, if there is an indication that there is no chance of overlapping in the near future it is better not to place the current query results into the cache. Thus, the expected overlapping serves as a measure of the likelihood that the query result will generate a cache hit. To perform this test we first predict where and when the next query is likely to be made based on the client's movement pattern and query interval history. Then we perform the area overlapping test by considering a scenario that maximizes the possibility of an overlapping. One such scenario would be when the immediate next query is for the same object type. Query results that fail to pass the test will not be placed in the cache.

Formally, let two range queries be Q_1 and Q_2, let the query range of Q_1 centered at $A(x_1, y_1)$ with radius r_1 and the range of Q_2 centered at $B(x_2, y_2)$ with radius r_2. Let the two points of intersection between Q_1 and Q_2 denote by C and D as shown in Figure 2. The area overlap function of two location queries Q_1 and Q_2, denoted by $AOF(Q_1, Q_2)$, can be computed as follows:

$AOF(Q_1, Q_2) = \frac{{r_2}^2(\angle CBD - \sin(\angle CBD)) + {r_1}^2(\angle CAD - \sin(\angle CAD))}{2}$
Where, $\angle CBD = 2\cos^{-1}(\frac{{r_2}^2 + AB^2 - {r_1}^2}{2 \times r_2 \times AB})$ and $\angle CAD = 2\cos^{-1}(\frac{{r_1}^2 + AB^2 - {r_2}^2}{2 \times r_1 \times AB})$

The computation of the area overlapping between Q_1 and Q_2 shows that the bigger the overlapping area, the higher the likelihood that Q_1s results will be reused to answer Q_2, thus the more benefit we will gain by placement of the results of Q_1 into the client cache.

Given that the location where the next query (i.e., Q_2) will be issued is typically unknown at the time when we determine whether to place the results of the current query (say Q_1) into the client cache, one approach is to use the distance between the location where the current query (Q_1) is issued and the expected location where the next query (Q_2) will be issued, denoted by d, to approximate the overlapping effect. This leads us to introduce the concept of *Overlapping Cache Benefit*.

Overlapping cache benefit: Let r_i denote the radius of the

current query Q_i and d_i denote the distance between the location where the current query Q_i is issued and the expected location where the next query Q_{i+1} will be issued. Let α_c denote a system-defined constant and w_i be the weight function that balances the measurement of overlapping benefit through query-dependent adaptation. We define the *Overlapping Cache Benefit* for a given query Q_i as follows:

$$OCB(Q_i) = \frac{r_i}{d_i} - w_i\alpha_c \quad (1)$$

The intuition behind the *Overlapping Cache Benefit* formula can be illustrated through the discussion of the three variables d_i, r_i, and w_i. We first describe how to compute d_i, the distance between the location where the current query (Q_i) is issued and the expected location where the next query Q_{i+1} will be issued. The distance d_i can be computed as the Euclidean distance between the two center points of the two query ranges, using the formula $d = \sqrt{(c_x - l_x)^2 + (c_y - l_y)^2}$, where ($c_x$,$c_y$) denotes the location of the current query Q_i and (l_x,l_y) denotes the expected future location when the next query is likely to be made. The future location (l_x, l_y) can be predicted using the current location (c_x, c_y), the current velocity (v_x, v_y), and the expected query interval q_{i+1} of the mobile client, i.e., $l_x = c_x + v_x q_{i+1}$ and $l_y = c_y + v_y q_{i+1}$. The estimated query interval (q_{i+1}) is determined based on the query history of the mobile client. By replacing (l_x, l_y) in the above formula for computing d, we have $d = q_{i+1}\sqrt{v_x^2 + v_y^2}$. Let t_1 denote the time when the initial query issued by the mobile client and t_2 be the next query issued by the same client. The expected query interval after the current query Q_i, denoted by q_{i+1}, can be estimated using an exponential averaging scheme: $q_2 = t_2 - t_1$ and $q_{i+1} = \beta(t_i - t_{i-1}) + (1-\beta)q_i$ where i > 2 and t_1, t_2, t_i, t_{i-1} represent the timestamps when the mobile client issued the first query, the second query, the current query, and the previous query respectively. q_i and q_{i-1} represent the query interval for the current query Q_i and the query interval for the previous query Q_{i-1} respectively. β is a constant between 0 and 1.

Now we discuss the role of variable r_i in measuring the overlapping cache benefit $OCB(Q_i)$. For any two queries Q_i with radius r_i and Q_{i+1} with radius r_{i+1} to overlap, the sum of r_i and r_{i+1} should be larger than the distance d_i between the two query centers. Namely $r_i + r_{i+1} > d_i$ holds. Given that at time t_i when the current query Q_i is processed, it is not always obvious when and where the next query Q_{i+1} will be posed. One way to approximate the next query for the same query object type is by assuming that it is similar to the current query (i.e., $r_i = r_{i+1}$). Thus, the following condition holds: $r_i/d_i < 1/2$. This implies that the results of query Q_i will be placed in the cache only if r_i/d_i is higher than the specified threshold constant α_c and α_c should be set greater than $1/2$.

Finally, we discuss the role of $w_i\alpha_c$ in measuring the overlapping cache benefit. α_c is a system-defined parameter and is typically set by a constant. The weight w_i can be set as a query independent constant or a query dependent variable. The product $w_i\alpha_c$ serves as the *'Control Knob'* to enable the incorporation of spatio-temporal placement to the cache placement decision whenever the overlapping cache benefit $OCB(Q_i)$ is greater than zero. For example, by setting $w_i = 1$ and $\alpha_c = 10$, the above *Overlapping Cache Benefit* computation will be instantiated to the following threshold-based formula with $\alpha_c = 10$.

$OCB_{threshold}(Q_i) = \frac{r_i}{d_i} - \alpha_c$

Clearly, keeping the *'Control Knob'* value very high can be risky because it could result in eliminating most of the data items, while keeping a very low value of $w_i\alpha_c$ (close to zero) would make the ad-hoc cache placement dominate the placement decision. Thus, an important challenge is to determine the setting of the weight w_i and thus the *'Control Knob'* to meet the objective of the chosen cache placement scheme.

Temporal Cache Benefit Measure: The Overlapping Cache Benefit measure can be seen as a technique used in MOBICACHE to capture the spatial locality of cached data items. Recall Section II-B, in mobile environments, location queries may be posed over moving objects, when the expected query interval for the next query is much longer than the time-to-live (TTL) timestamp of the moving objects returned by the current query, such query results should not be placed in the client cache. One straightforward and yet effective way to model the temporal locality of cached data items is to compare the TTL of the query results with the expected query interval of the next query. Only when the TTL of a data item is greater than the expected query interval, the placement of such an item in the client cache can be granted. Thus, if the TTL of the query result items will expire much sooner than the expected query interval for the next query, then such items should not be placed into the client cache.

Threshold-based STP Scheme: Threshold-based STP scheme implements the most basic spatio-temporal placement strategy. In this scheme, the placement decision is made by analyzing two orthogonal and yet complimentary factors: (1) the overlapping cache benefit measure to determine if the following spatial placement condition $\frac{r}{d} \leq \alpha_c$ holds, and (2) the temporal cache benefit measure which uses the TTL of the target objects from the current query against the expected query interval of the next query and test if $TTL(Q_i) > q_{i+1}$ holds. Only the data items that pass the test of both overlapping cache benefit measure and the temporal cache benefit measure are placed in the client cache.

Setting α_c in MobiCache: In MOBICACHE, the spatial control knob constant α_c is typically set to a positive value greater than 0.5, favoring placement of those query results that are likely to overlap with the next expected query. Naturally, the larger the α_c value is, the higher degree of spatial overlapping is preferred for cache placement. Thus choosing a slightly higher value for α_c would ensure that only the query results that are highly likely to overlap with the next expected query would be placed in the cache. However, setting the α_c value too high may cause too few query results to enter the cache, affecting hit ratio unfavorably.

The *Threshold based Spatio-Temporal Placement* scheme is non-adaptive because the *'Control Knob'* in the Overlapping Cache Benefit calculation remains fixed throughout the life time of the cache. This motivates us to develop the Bound based STP and the Speed and Query Interval (SQI) adaptive STP scheme.

B. Bound-based Spatio-Temporal Placement Scheme (Bound STP)

One of the reasons that the threshold-based STP scheme improves ad-hoc placement is the fact that the threshold-based STP selectively discards certain query results based on their overlapping cache benefit measure and their temporal cache benefit measure. It is also observed that when the mobile client moves at higher speeds the threshold-based STP drops much more items than when the mobile client travels at lower speeds. In order to maintain a reasonable hit ratio for the client cache, we need to avoid extremely high drop rate or extremely low drop rate during cache placement when responding to changing motion behavior and changing query patterns. A simple and straightforward way to address this problem is to configure the system with a *lower discard bound*, denoted by T_{Lower}, and an *upper discard bound*, denoted by T_{Upper}. Instead of setting the *STP Control Knob* as a fixed system-wide parameter, we devise an adaptive approach. Whenever the percentage of discarded items (denoted by $\lambda_{discard}$) drops below the specified lower bound T_{Lower}, the Bound-based STP scheme will increase the *Control Knob weight* by a pre-configured constant γ_c. Similarly, when the percentage of discarded items becomes greater than the specified upper discard bound T_{Upper}, the Bound STP scheme will be relaxed by reducing the *Control Knob weight* by the pre-defined constant γ_c. The testing of $\lambda_{Discard}$ against T_{Upper} and T_{Lower} is performed at the end of each epoch, which is when the adaptation happens. The goal of this placement scheme is to keep the amount of discarded items during the cache placement within the upper and lower bound in each epoch. This is achieved by comparing the percentage of the discarded items ($\lambda_{Discard}$) during the cache placement decision in the current epoch against the system-supplied lower and upper bound T_{Lower} and T_{Upper}, and adapting the setting of the control knob weight w_i (one γ_c step at a time) for the next epoch. If the percentage of discarded items ($\lambda_{Discard}$ in one epoch continues to be greater than T_{Upper} (system-supplied parameter), then the weight for the next epoch will be further reduced by one γ_c at the end of the current epoch. On the other hand if the percentage of discarded items is lesser than T_{Lower} the weight for the next epoch will be increased by one γ_c at the end of the current epoch. Eventually, the percentage of discarded items during the cache placement will fall into the range specified by the lower and upper bound and stabilize. As expected, this Bound STP scheme offers better performance than the Threshold-based STP scheme when the movement speed of the mobile client changes more frequently.

Now we formally describe the Bound-based STP scheme. Let w_i and w_{i-1} denote the weight to be set and the previous weight respectively and γ_c denotes the constant increment or decrement step, such as 10 (a lower value of γ_c would result in slower but more stable adaptation while a higher value would result in faster but potentially unstable adaptation). Let $\lambda_{Discard}$ denote the percentage of discarded entries for the current epoch, T_{Upper} and T_{Lower} denote the upper discard bound and the lower discard bound respectively. We can define the control knob weight w_i as follows:

$$w_i = \begin{cases} w_{i-1} - \gamma_c & \text{if } \lambda_{discard} > T_{Upper} \\ w_{i-1} + \gamma_c & \text{if } \lambda_{discard} < T_{Lower} \\ w_{i-1} & \text{otherwise} \end{cases}$$

Thus the *Overlapping Cache Benefit* for a query with radius r_i and the expected distance of d_i from the center of the current query (Q_i) to the center of the next query (Q_{i+1}) is defined as follows:

$$OCB_{bound}(Q_i) = \frac{r_i}{d_i} - w_i \alpha_c. \quad (2)$$

In the Bound-based STP, the control knob weight w_i is determined at the end of each epoch based on the weight w_{i-1}, the percentage of discarded items $\lambda_{Discard}$, and the pre-defined system-wide parameters γ_c, T_{Upper}, and T_{Lower}. In MOBICACHE the lifetime of the client cache is divided into epochs of equal duration. Adaptations happen only at the end of each epoch. By introducing epochs, the system adapts only when there is a consistent drop or increase in the percentage of items discarded.

C. Speed and Query Interval Adaptive STP Scheme (SQI STP)

The *Bound STP* offers better performance in comparison to *threshold-based STP* in most cases. On one hand, bound STP is able to adapt the STP control knob weight periodically through increasing or decreasing the weight by a constant γ_c at the end of each epoch. By maintaining the percentage of discarded items ($\lambda_{Discard}$) during the cache placement within the specified upper and lower bound, one can avoid making over pessimistic or over-optimistic placement decision. On the other hand, the threshold STP uses a fixed threshold throughout the lifetime of a client cache, which severely limits the flexibility of the placement scheme to adapt to changing movement pattern and query interval of the mobile client. However, the flexibility of adaptation supported by the Bound STP scheme is limited by the pre-defined lower and upper bound on the percentage of discarded items, T_{Lower} and T_{Upper}.

In order to support dynamic adaptation of cache placement to the changing movement speed and changing query interval of the mobile client, we develop the third spatio-temporal placement scheme, called the *Speed-Query Interval Adaptive STP* scheme (SQI-Adaptive STP or simply SQI STP for short). In this scheme, the *'STP Control Knob weight'* is determined based on both the query interval and the movement speed of the mobile client. Intuitively, we observe that the higher the movement speed and the longer the query interval a mobile client has, the lower the overlap cache benefit will be. This is because at higher speeds or higher query intervals the query

ranges of consecutive queries are further apart and the chances of spatial overlapping are greatly reduced. Thus, instead of using the fixed upper and lower bound to tune the control knot weight and thus the cache placement quality, the *'STP Control Knob weight'* should be set to be inversely proportional to the speed and the query interval for better cache performance. Based on this rationale we define the *'STP Control Knob weight'* based on the movement speed of the mobile client at the time when the ith query Q_i was issued, denoted by $speed_i$, and the expected movement speed when the next query Q_{i+1} is expected to be posed, denoted by $ExpSpeed$.

$$w_i = \frac{1}{ExpSpeed_{i+1} \times q_{i+1}} \tag{3}$$

Where $i > 1$, q_{i+1} is the expected query interval when the next query will be posed, and

$$ExpSpeed_2 = speed_1 \tag{4}$$

$$ExpSpeed_{i+1} = \zeta(speed_i - speed_{i-1}) + (1-\zeta)ExpSpeed_i \tag{5}$$

$$q_2 = t_2 - t_1 \tag{6}$$

$$q_{i+1} = \beta(t_i - t_{i-1}) + (1-\beta)q_i \tag{7}$$

Where $i \leq 2$, t_i, t_{i-1} represent timestamps for the mobile client to ask the current query and the previous query respectively, q_i and q_{i-1} denote the current query interval and the previous query interval respectively, and β is a constant between 0 and 1, and usually set to 0.5 in our experiments.

In the Speed-Query Interval Adaptive STP scheme, the *Overlapping Cache Benefit* measure for a query Q_i with radius r_i and expected distance of d_i from the center of the current query to the next query is defined as follows:

$$OCB_{SQIadapt}(Q_i) = \frac{r_i}{d_i} - \left(\frac{1}{ExpSpeed_{i+1} \times q_{i+1}}\right)\alpha_c \tag{8}$$

where α_c is set by the system default (recall the discussion in Section III-A).

IV. Experimental Evaluation

In this section we conduct the experimental evaluation on the proposed three spatio-temporal placement schemes and compare them with the popular ad-hoc placement scheme used in most of existing mobile cache systems [10], [16], [5], [4], [13], [11], [6], [12]. The experiments are divided into two sets. One set of experiments is designed to study the impact of different characteristics of mobile clients on cache performance in terms of hit ratio. Another set of experiments is designed to study the impact of different characteristics of location-dependent queries on cache performance. We show that spatio-temporal placement schemes can significantly improve the cache hit ratio (between 5% to 10% improvement). Among the three STP schemes, the SQI adaptive placement offers the highest hit ratio and at the same time is highly robust in responding to changes in speed and query interval of the mobile client. We also show that the proposed STP placement schemes work well with both temporal replacement policy like LRU and spatial replacement policy like FAR [10]. The combination of our SQI-adaptive STP with FAR offers higher hit ratio than any other combination of placement and replacement strategies, be it ad-hoc and FAR, ad-hoc and LRU, Bound and FAR, threshold-based STP and FAR, or SQI-adaptive placement and LRU replacement. In the rest of this section we first describe the mobility simulation model used for our experiments, including the input generator and the play module. Then we report our experimental results in detail.

A. Simulation Model

We use a variant of the popular *Random Walk Model* [2] for simulation of client mobility and motion behavior of moving objects. The client starts at location (0,0) with an initial velocity (v_x, v_y). v_x and v_y represent the x component and y component of the velocity vector. A positive value indicates that the movement of the client is along the positive axis while a negative value indicates the movement along the negative axis. Thus, the velocity values represent both speed and direction. The client is modeled to move with a given velocity for some time (movement interval), then wait for a duration at the new destination (pause interval) and then change its velocity randomly (speed and direction). This process repeats over time. The client makes a large number of queries based on the given query intervals while on the move. By varying the input parameters for the model, such as the movement interval, pause interval, query interval, several different work loads can be generated. For instance, having a very large movement interval would force the client to travel in a straight line for longer durations (e.g., simulating a vehicle such as truck traveling from city to city on the highway), while having a small movement interval would allow a mobile client moving along a straight line only for a short duration (e.g., simulating tourists traveling within a city or a tourist attraction). Our simulation model consists of two components - (a) Input Generator and (b) Play module. **The input generator** takes parameters, such as client velocity, static or moving objects, per object query radius distribution, query interval, movement interval, and generates a sequence of location dependent queries, each of such queries is associated with the time when the query was made, the size of the result, and the TTL of the result. The TTL for queries on moving objects is calculated using the radius of the query range and the speed of the moving target object, which is randomly set between 30 to 60 mph. Null values are used for query results that are still objects such as restaurants, gas stations, and so forth. Queries were generated and assigned random values of range (R) using a normal distribution on R. For example, if the object type (O) is 'restaurant' and the range R is 10 miles then the queries with restaurant as the object type would set the radius following a normal distribution around 10 miles. The object types used in the queries are generated based on a zipf distribution, namely a few object types are queried very often while most of other object types are queried once in a while. This module is representative of the real-world query scenarios in mobile environments. The **Play module**

simply 'plays' the location-dependent queries generated by the input module under different cache placement and replacement policies and different cache sizes. The play module consists of implementations of various placement, invalidation and replacement policies. This module simulates the mobile client. It estimates the query interval, the movement speed and other parameter of the mobile client at run time and uses them in the spatio-temporal placement algorithms described in Section III. The advantage of separating the input generator module from the play module is to enable the exact same set of queries at the exact same set of locations to be replayed under different placement and replacement policies and different cache sizes, allowing us to conduct an in-depth study of these parameters and their impacts on cache performance.

The key parameters used in the experiments reported in this paper are summarized in Table 1. The number of location queries used in the experiments is 30,000. For the constants, we set α_c to 10, β to 0.5, γ_c to 10, ζ to 0.8, T_{Upper} to 60%, T_{Lower} to 50% and epoch size to 100 . All the simulations were run on a Linux server with 4 3GHz processors and a total memory of 4GB. The results are reproducible on any other machine, since the mobile client specific information is implemented as a part of the play module.

Parameter	Range	Default
Replacement Policy	LRU/FAR	LRU
Invalidation Scheme	Lazy	Lazy
Number of Objects	50 - 10000	5000
Cache Size	4 MB to 20 MB	10 MB
Query Interval	0.1 to 11 hrs	0.2 hrs
Movement Interval	0.1 to 11 hrs	2 hr
Pause Interval	0.1 to 11 hrs	1 hr
Minimum Speed	10 to 100 mph	20 mph
Maximum Speed	25 to 115 mph	30 mph
Percentage of moving object queries	0 to 100%	10%

Table 1: Simulation Parameters

B. Experimental results

We first report the experimental evaluation of the effectiveness of our spatio-temporal placement schemes by studying the impact of the characteristics of mobile client, such as changes in cache size, query interval, movement interval and movement speed on hit ratio. Then we report the experimental results on the impact of different query characteristics on cache hit ratio, including the percentage of queries over moving objects and the number of objects returned by queries. Finally we report the study on the effect of using different replacement policies on the performance of our spatio-temporal cache placement schemes in terms of hit ratio.

1) Mobile Client Characteristics on Cache Hit Ratio:

Impact of Cache Size on Hit Ratio: Figure 3(leftmost) shows the impact of cache size on hit ratio on cache sizes of the mobile devices. The spatial placement refers to using only the Overlapping cache benefit measure to make the placement decision, whereas the temporal placement refers to using only the temporal cache benefit measure (TTL against query interval) to make the placement decision. This experiment shows that both spatial and temporal placement strategies perform better compared to the ad hoc placement. By combining spatial and temporal cache benefit measures together as the Threshold-based STP, the cache performance is even better. It is important to note that, in the *Threshold-based STP* scheme, the *STP Control Knob* is fixed with $w_i = 1$ and $\alpha_c = 10$. However, in the *Bound STP*, the overlap benefit factor varies depending on the percentage of discarded items in each epoch, the system-supplied bound parameters, T_{Lower} and T_{Upper}, and the system-defined increment/decrement constant γ_c. Similarly, in the *Speed-Query Interval adaptive STP* scheme, the overlapping cache benefit measure rises and falls depending on the movement speeds and query intervals of the mobile clients. Hence, the adaptive STP schemes offer better overall cache performance than threshold-based or ad-hoc placement approaches.

Impact of Query Interval on Hit Ratio: Figure 3(center-left) shows the variation of hit ratio with changes in query interval. When the client frequently poses queries, the hit ratios for all the placement strategies (including the Ad hoc placement) are high. This is intuitive because the client is not likely to have traveled far away between the queries. With an increase in query interval, the hit ratio drops for all the placement schemes (except for Spatial and Threshold STP initially). The deviation for spatial and threshold STP schemes for smaller query intervals is because they have fixed overlapping benefit factors and hence cannot adapt to the changes in query interval.

Impact of Client Movement Interval on Hit Ratio: Figure 3(center-right) shows the variation of hit ratio with changes in movement interval. Movement interval is defined as the average time interval between stops when the client is on the move. At the end of every movement interval the client pauses for a fixed amount of time (pause interval). As the movement interval increases, the probability of the mobile client changing its velocity is small, and thus the chance of overlapping with previous queries is smaller (since the client would turn around less often). In all the three STP schemes the *STP Control Knob weight* is chosen to maximize the overlapping cache benefit. As a result, with increase in movement interval (taking a longer time for velocity changes), the hit ratio for spatial placement and hence all STP schemes but bound STP increases. However, this improvement is limited to the beginning stage of the adaptation to the *STP Control Knob weight*.

Impact of Client Movement Speed on Hit Ratio: Figure 3(rightmost) shows the variation of the hit ratio as the movement speed of the mobile client changes. In the experiments the radius of spatial range of the queries is set between 1 to 100 miles with 90% of the objects having the radius range less than 10 miles. The gain in hit ratio due to spatio-temporal placement is higher at lower speeds. At higher speeds like 50 mph to 100 mph, the hit ratio reduces for all placement schemes. Not surprisingly, in all cases the Bound scheme or speed and query interval adaptive STP scheme offer the best

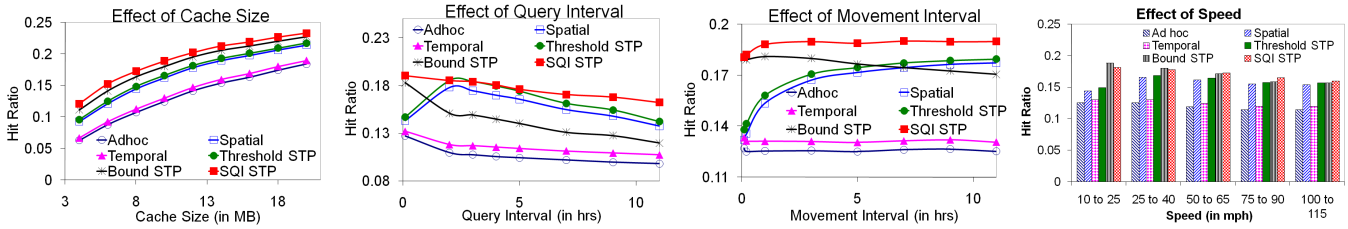

Fig. 3. Effect of Characteristics of Mobile Client on Cache Hit Ratio

performance.

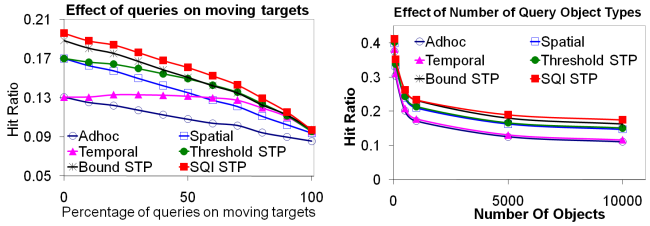

Fig. 4. Effect of Query Characteristic on Hit Ratio

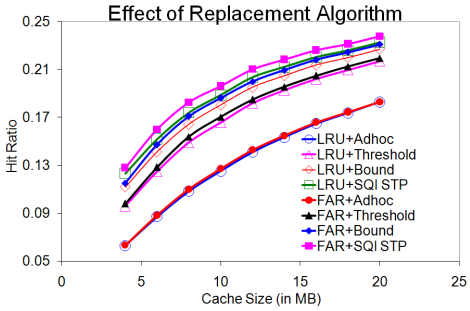

Fig. 5. Effect of Replacement Algorithm

2) Query Characteristics on Mobile Cache Performance: **Percentage of Queries for Moving Objects on Hit Ratio:** In the MOBICACHE system, clients can make queries for both static objects and other moving objects. Queries over moving objects are important to study because they represent results which have very low TTLs since the target objects may soon move out of the area of interest and hence the query results may quickly become invalid. Figure 4(left) shows that when the percentage of queries on moving objects is extremely low (close to 0%), the gain because of temporal placement is close to zero. This is simply because all queries are over static objects such as gas stations, restaurants, and the query results will only expire when the mobile client moves out of the current query range.

Number of Query Object Types on Hit Ratio: Figure 4(right) shows how the varying number of query object types impacts the hit ratio. Clearly, with the increase in the number of object types, the hit ratio drops irrespective of which cache placement scheme one uses. This is because, with increase in object types, the probability of overlapping will drop and so does the hit ratio.

C. *Effect of replacement algorithm*

From the graph 5 it can be seen that irrespective of which cache replacement algorithm we use, either purely temporal like LRU or spatial like FAR [10], the spatio-temporal placement schemes always outperform the corresponding ad hoc placement-replacement algorithm combination. Another interesting observation is that the difference in the cache hit ratio between *SQI STP+FAR* combination and *Ad hoc+FAR* combination is higher than that between *SQI STP+LRU* and *Ad hoc+LRU*. This demonstrates that the cache performance can be improved to an even higher extent by incorporating the spatio-temporal techniques in both cache placement and cache replacement decision.

V. RELATED WORK

Client side caching of location dependent queries is an important technique for improving performance of location-based services. Most of the existing research in this area has focused on cache replacement and invalidation through incorporating some aspects of the spatial and temporal semantics embedded in the query result objects or the location queries, while assuming an ad hoc cache placement. Examples include cache invalidation work [1], [7], [16], [3], [14] and cache replacement proposals [10], [16], [5], [4], [13], [11], [6], [9], [8], [15], [12]. FAR [10] is one of the pioneer work on spatial cache replacement algorithm using distance as a parameter in addition to temporal parameters like those in LRU. PA and PAID [16] offered improved cache replacement algorithms for caching nearest neighbor queries. MARS [5] continued the research by adding frequency of access, query rate, velocity to further improve the cache replacement efficiency. Surprisingly, all existing research efforts have focused on employing spatiotemporal strategies for improving cache invalidation and cache replacement.

To our best knowledge, none have studied the potential of incorporating spatio-temporal aspects and the motion behavior of mobile clients into the cache placement decision and the impact of spatio-temporal placement on the performance (hit ratio) of the client cache. Furthermore, the development of spatio-temporal strategies for cache replacement and invalidation are different in both design principle and engineering algorithms, and cannot be directly applied to cache placement for obvious reasons. The factors that affect the decision of cache placement are quite different than those that are critical to cache replacement simply because placement happens much earlier in the life cycle of cached items, thus the parameters

that are significant for cache replacement are not available for making cache placement decision.

VI. Conclusion

We have described a spatio-temporal placement model for caching location-dependent queries. This paper makes three unqiue contributions. First, we introduce the concept of *'Overlapping Cache Benefit'* as a measure of the potential hit rate of cached data items and predicts the potential cache benefit of the query results based on multiple spatio-temporal properties of mobile clients. Second, we develop three spatio-temporal cache placement schemes, which provide step-by-step in-depth understanding of various factors that may affect the performance of a client cache in mobile environments. In our spatio-temporal placement model, the decision of whether to place an item into the cache of a mobile client is made by combining both the spatial validity and temporal validity of query results and the motion behavior and query patterns of the mobile client.

References

[1] D. Barbara and T. Imieliski. Sleepers and workaholics: caching strategies in mobile environments. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 1994. ACM Press.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[3] H. Chung and H. Cho. Data caching with incremental update propagation in mobile computing environments. volume 30, pages 77–86, The Australian Computer Journal, 1998.

[4] I.-D. Jung, Y.-H. You, J.-H. Lee, and K. Kim. Broadcasting and caching policies for location-dependent queries in urban areas. In *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*, pages 54–60, New York, NY, USA, 2002. ACM Press.

[5] K. Y. Lai, Z. Tari, and P. Bertok. Location-Aware Cache Replacement for Mobile Environments. In *IEEE Global Telecommunications Conference (Globecom 2004)*, pages 3441–3447, December 2004.

[6] K. C. K. Lee, H. V. Leong, and A. Si. Semantic query caching in a mobile environment. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(2):28–36, 1999.

[7] H. V. Leong and A. Si. On adaptive caching in mobile databases. In *SAC '97: Proceedings of the 1997 ACM symposium on Applied computing*, pages 302–309, New York, NY, USA, 1997. ACM Press.

[8] M. H. K. V. Ren, Q. Dunham. Semantic caching and query processing. volume 15, pages 192–210. IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, 2003.

[9] Q. Ren and M. Dunham. Semantic caching in mobile computing. 98-CSE-04.

[10] Q. Ren and M. H. Dunham. Using semantic caching to manage location dependent data in mobile computing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 210–221, New York, NY, USA, 2000. ACM Press.

[11] G. Santhanakrishnan, A. Amer, and P. K. Chrysanthis. Towards universal mobile caching. In *MobiDE '05: Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access*, pages 73–80, New York, NY, USA, 2005. ACM Press.

[12] B. J. D. S. M. T. Shaul Dar, Michael J. Franklin. Semantic data caching and replacement. pages 330–341. In Proc. of VLDB, 1996.

[13] K.-L. Wu, P. S. Yu, and M.-S. Chen. Energy-efficient caching for wireless mobile computing. In *ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering*, pages 336–343, Washington, DC, USA, 1996. IEEE Computer Society.

[14] J. C.-H. Yuen, E. Chan, K.-Y. Lam, and H. W. Leung. Cache invalidation scheme for mobile computing systems with real-time data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):34–39, 2000.

[15] B. Zheng and D. L. Lee. Semantic caching in location-dependent query processing. pages 97–116. Springer-Verlag, 2001.

[16] B. Zheng, J. Xu, and D. L. Lee. Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments. *IEEE Transactions on Computers*, 51(10):1141–1153, 2002.