Location-aware cache replacement for mobile environments

Lai, Kwong; Tari, Zahir; Bertok, Peter

Lai, Tari, Z., & Bertok, P. (2004). Location-aware cache replacement for mobile environments. Globecom '04 IEEE Global Telecommunications Conference, 3441–3447.
https://doi.org/10.1109/GLOCOM.2004.1379006

Published Version: https://doi.org/10.1109/GLOCOM.2004.1379006

Location-Aware Cache Replacement for Mobile Environments

Kwong Yuen Lai Zahir Tari Peter Bertok
School of Computer Science and Information Technology
RMIT University, Melbourne, Australia
Email: {kwonlai, zahirt, pbertok}@cs.rmit.edu.au

Abstract— **Traditional cache replacement policies rely on the temporal locality of users' access pattern to improve cache performance. These policies, however, are not ideal in supporting mobile clients. As mobile clients can move freely from one location to another, their access pattern not only exhibits temporal locality, but also exhibits spatial locality. In order to ensure efficient cache utilisation, it is important to take into consideration the location and movement direction of mobile clients when performing cache replacement. In this paper, we propose a mobility-aware cache replacement policy, called MARS, suitable for wireless environments. MARS takes into account important factors (e.g. client access rate, access probability, update probability and client location) in order to improve the effectiveness of on-board caching for mobile clients. Test results show that MARS consistently outperforms existing cache replacement policies and significantly improves mobile clients' cache hit ratio.**

I. Introduction

Advances in mobile communication infrastructures and global positioning technologies have resulted in a new class of services, referred to as Location Dependent Information Services (LDIS) [1], becoming available to users. Location dependent information services provide users with the ability to access information related to their current location. Examples of LDIS include providing users with local weather information, access to news and information about nearby businesses and facilities, etc.

There are many benefits in providing support for LDIS. Firstly, with the help of location information, data objects can be stored at servers nearest to where they are accessed most frequently. This improves access time and reduce network traffic. The use of location dependent data also provide convenience for mobile users, as mobile device can automatically fetch data based on location, ensuring the most relevant information is provided to the users.

Despite the benefits of LDIS, a number of challenges (e.g. limited cache space, limited bandwidth, limited client transmission power) [2], [1], [3] must be overcome before these benefits can be realised. Existing research have shown that caching is an important technique in combating the limitation of mobile environments. By caching on mobile devices, data availability is increased, access speed improved and the need of transmission over the wireless channel reduced. In this paper, we focus our attention on the issue of cache replacement for mobile clients and how client-side caching can improve the performance of mobile devices when utilising LDIS.

Tradition caching replacement policies rely on the temporal locality of clients' access pattern to determine which data objects to replace when a client's cache becomes full. However, in mobile networks where clients utilise location dependent services, the access pattern of mobile clients do not depend only on the temporal properties of data access, but is also dependent on the location of data, the location of the clients and the movement direction of clients [4]. Relying solely on the temporal properties of data access when making cache replacement decisions will result in poor cache hit ratio. In order to improve the performance of mobile clients' caches, it is important to consider both temporal and spatial[1] locality when making cache replacement decisions.

Many existing cache replacement policies (e.g. [5], [6]) use cost functions to incorporate different factors including access frequency, update rate, size of objects..etc, however very few of these policies account for the location and movement of mobile clients. Cache replacement policies such as LRU, LFU and LRU-k [7] only take into account the temporal characteristics of data access, while policies such as FAR [2] only deal with the location dependent aspect of cache replacement but neglect the temporal properties. One policy which does consider both spatial and temporal properties of data objects is the PAID proposed in [6]. However, PAID does not take into account updates to data objects. It also deals with client movement in a very simplistic way (i.e. only considers client's current movement direction).

In this paper, we propose a new gain-based cache replacement policy that takes into account both the temporal and spatial locality of clients' access pattern. The proposed strategy, called Mobility-Aware Replacement Scheme (MARS), ties together various factors that are important when making cache replacement decisions through a cost function. This cost function comprises a temporal score and a spatial score. The temporal score is calculated from access probabilities, update and query rates, while the spatial score is calculated based on client location, the location of data objects and client movement direction. When the cache of a mobile client becomes full and a new object needs to be cached, the cost function is used to generate a cost value for each cached object. The object with the lowest value is evicted from the client's cache and replaced by the new object. We show through

[1]Spatial in terms of the geographical location associated with a data object

simulation that mobile clients using MARS are able to achieve an significant improvement in cache hit ratio compared to clients using other existing cache replacement policies.

The rest of this paper is organised as follows. Section II provides a survey of existing mobility-aware cache replacement policies. Section III describes the location model used in our work. In Section IV, we propose a new cache replacement strategy called MARS, suitable for mobile clients using LDIS. Simulation results are presented in Section V together with an analysis of the results. This is followed by our conclusion and a discussion of possible future work in Section VI.

II. Related Work

A. Modelling Location Dependent Data

The issue of modelling location dependent data for mobile environments was first addressed in [4]. Location dependent objects can have both temporal replicas and spatial replicas. By binding queries and spatial replicas to data regions, it becomes possible for clients to perform location dependent queries. While the issue of location dependent caching is discuss in this work, it was not investigated in depth. A more advanced semantic based location model was proposed in [8], where attributes of data objects are either location related, or non-location related. For example, a location related attribute, $city$, may take values from a domain containing city names (e.g. Melbourne, Sydney). While it is possible to define domains of different granularity, this comes at a higher overhead. It is also unclear from this work how a user's location is determined and mapped to the location-related domains.

B. Cache Replacement Policies

1) Temporal Locality Based Cache Replacement Policies: There are many existing temporal-based cache replacement strategies, including LRU, LFU and LRU-K [7]. These replacement polices are based on the assumption that clients' access pattern exhibits temporal locality (i.e. recently accessed objects are likely to be accessed again in the near future, objects that were queried frequently in the past will continue to be queried frequently in the future). While these policies are suitable for a network with stationary clients, they are unsuitable for supporting location dependent services. They do not take into account the location and the movement of mobile clients and data objects when making cache replacement decisions. Location dependent data is cached in the same way as non-location dependent data, resulting in inefficient cache utilisation for clients who move frequently and access information based on their location.

FAR (Furthest Away Replacement) [2] is one of the earliest mobility-aware cache replacement policies. It uses the current location of mobile clients and the movement direction of clients to make cache replacement decisions. In FAR, cached objects are grouped into two sets. Those in the moving direction of the client are put into the in-direction set, while those that are not are put into the out-direction set. Data objects in the out-direction set are always evicted first before those in the in-direction set. FAR improves the cache hit ratio of existing temporal based policies as it considers the location and the future movement of mobile clients. However it only deals with spatial locality and neglects the temporal properties of clients' access pattern. It is also ineffective when mobile clients change direction frequently as objects will often be move between the in-direction and out-direction sets.

PA (Probability Area) [6] is a cost-based replacement policy, where each cached object is associated with a cost calculated with a predefined cost function. When cache replacement takes place, the data object with the lowest cost is replaced. The cost function used in PA takes into account the access probabilities (P_i) of data objects and their valid scope area $A(v_i')$. The cost of an object d_i is defined as $c_i = P_i.A(v_i')$. Although the PA schemes takes into account both the valid scope area of data objects, and the temporal property of objects through their access probabilities, it does not consider the location and future movement of clients. This leads to poor cache performance, because objects that are close to the client are often replaced before objects that are far away, simply because their valid scope area is smaller.

PAID (Probability Area Inverse Distance) [6] is an extended version of PA that deals with the problem of distance. In PAID, the distance between mobile clients and data objects included as part of the cost function used for making cache replacement decisions. The cost function is defined as $c_i = \frac{P_i.A(v_i')}{D(v_i')}$, where $D(v_i')$ is the distance from the client to the valid scope v_i'. Based on this function, objects that are further away will have their cost reduced by a larger factor compared to those objects nearer to the client. PAID also attempts to cater for client movement direction, by multiplying the cost of objects which are not in the direction of the client's movement with a large factor. Results from [6] shows that both PA and PAID performs better than other existing schemes such as LRU and FAR. However, these two schemes suffer a similar problem in that the temporal property of data objects is only represented by one parameter, the access probability. Other factors such as the cost of retrieving a missing object, the size of data objects, the combined cost of objects evicted and the update probabilities have not been accounted for.

III. The Location Model

We assumed that mobile devices are equipped with positioning capabilities, such as GPS or power-based positioning. We define a location L as a pair of coordinates, (x, y) where x and y are the latitude and longitude respectively. There are two main benefits in representing locations this way. Firstly, since most current positioning systems are already capable of providing latitude and longitude coordinates, this location model can be supported easily without having to upgrade existing hardware. Secondly, the latitude-longitude model eliminates ambiguities as any location can be uniquely represented by a coordinate pair. Our location model only considers two-dimensions space, however it can be easily extended to cater for 3 dimensional space by including the third dimension z. The distance between any two locations is the length of a direct line connecting the two points (i.e. the Euclidean distance). For

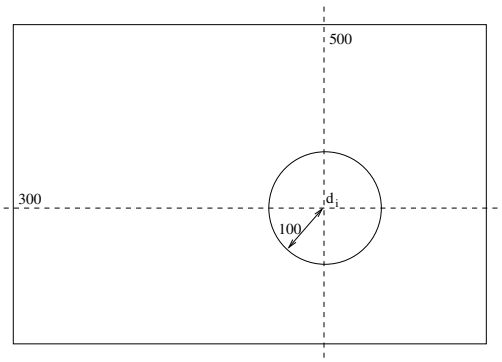

Fig. 1. Valid Scope Example

example, given a client located at $(x1, y1)$ and a data object located at $(x2, y2)$, the distance between the client and the object is equal to $\sqrt{|x1 - x2|^2 + |y1 - y2|^2}$.

In a traditional database system, each data object only has one value, even when multiple replica of an object exist. The value of each object may change over time, however, the value of the replicas are the same at any one time. However, in a wireless environment, where location dependent information services are provided, data objects are also bounded to their location by their valid scope. The valid scope of an object is the area in which the value of the object is meaningful and correct. For example, a query for the data object $local\_time$, will return different results depending on where the query is issued. Another example might be the data object $nearest\_hotel$. We refer to this type of data objects as Location Dependent Data (LDD) objects. The value of a LDD object depends on the location from which it is queried.

We will now provide a definition of the valid scope of data objects. The valid scope $vs(d_i)$ of an object d_i is defined as a tuple $(Lx_i, Ly_i, range_i)$, where Lx_i and Ly_i is the center reference point of $vs(d_i)$ and $range_i$ is the radius defining the maximum euclidean distance from (Lx_i, Ly_i) in which d_i is valid. This representation of valid scopes is suitable for LDIS in wireless environments because it is small in size, thus does not introduce a high storage/communication overhead.

As an example, given a data object d_i with a valid scope 500, 300, 100, the value of d_i is valid within a 100 distance units radius from the point (500, 300). The valid scope of d_i is illustrated in Figure 1.

Lastly, a representation of velocity is needed to model the movement of mobile clients. A velocity can be represented by a tuple (Vx, Vy), where Vx is the speed along the x direction, Vy is the speed along the y direction. In our model, we define the x direction as running from west to east, and the y direction runs from south to north.

IV. A New Location-Dependent Cache Replacement Policy - MARS

As mobile clients move from location to location utilising LDIS, the locality of their access pattern change dynamically based on their movement pattern. It is important for cache replacement policies to dynamically adapt to this change in access locality to ensure high cache hit ratio is achieved. In this section, we propose a new cache replacement policy called MARS (Mobility-Aware Replacement Scheme), to address the issue of mobility and caching of location-dependent data.

MARS is a cost based scheme. Associated with each cached data object is a replacement cost. When a new data object needs to be cached, and there is insufficient cache space, the object with the lowest replacement cost is removed until there is enough space to cache the new object. The calculation of the replacement cost takes into account both temporal and spatial properties of information access to cater for the mobile nature of users in wireless environments. Temporal locality is accounted for by considering access probability, access frequency, update probability and update frequency of data objects. On the other hand, clients' current location, velocity and location of data objects are used to account for spatial locality. Cache size, object size, and retrieval cost are also included in the cost function to capture the condition of the client's cache.

A. System Model

In order to present the MARS policy in details, we must first describe the wireless network model used. We consider a network which consists of both wired and wireless entities. Wired entities include stationary clients and information servers. Information servers host copies of data objects that are of interest to the clients. Also connected to the wired network are base stations and access points that provide wireless communication channels needed for mobile clients to access information on the information servers. The area covered by each base station or access point is called a cell. Mobile clients can connect to the network wirelessly while inside the coverage area of a cell. Although both stationary and mobile clients exist in our network model, we focus on mobile users as the goal of our work is on improving cache performance for these users. Stationary clients can employ any one of the many existing cache replacement policies designed for stationary clients.

Mobile clients access data objects at the information servers by issuing queries at a mean rate of λ queries per time unit. When answering a query, a client first looks in its own onboard cache to see if the objects needed are available. If the objects are cached, the query is resolved immediately without the need of communicating with the information servers. On the other hand, if the objects needed are not cached, an uplink request is sent to the nearest information server to request the objects.

We also consider the update pattern of data objects. In our model, we assume objects are only updated on the server side. Updates are generated at a rate of μ updates per time unit. Updates and queries are distributed among data objects following two distributions $Pr(d_i \in update)$ and $Pr(d_i \in query)$. The actual functions used for these distributions are application dependent. For example, in applications where client queries exhibit high locality, $Pr(d_i \in query)$ may take the form of a Zipf distribution [9]. In applications where

TABLE I

Cost function parameters

Parameter	Value
d_i	Data object with ID i
$vs(d_i)$	The valid scope of d_i
λ	Mean query rate
μ	Mean update rate
$L_i = (Lx_i, Ly_i)$	Location of d_i
$L_m = (Lx_m, Ly_m)$	Location of client m
$Pr(d_i \in query)$	Probability that d_i is chosen for a query
$Pr(d_i \in update)$	Probability that d_i is chosen for an update
$cacheSize_m$	Size of client m's cache
$objSize_i$	Size of d_i
$t_{current}$	Current time
$t_{q,i}$	Time d_i was last queried
$t_{u,i}$	Time d_i was last updated
c_i	Cost of retrieving d_i from remote server
$V_m = (Vx_m, Vy_m)$	Velocity of client m

updates to data objects are randomly generated, $Pr(d_i \in update)$ can be represented by a uniform distribution (i.e. $Pr(d_i \in update) = \frac{1}{N}$ where N is the number of data objects on the server.)

In our model, objects are only evicted from a client's cache under two conditions. Firstly, invalidation reports are sent to clients by the server periodically to inform them of objects that have been updated. Clients use these invalidation reports to identify out-dated objects and remove them from their cache to maintain cache consistency. Secondly, when a client's cache becomes full and a new object needs to be cached, objects are removed to make room for the new object.

Table I summarises the various parameters used in our cost function.

B. The Replacement Cost Function

With the basic parameters defined, we can now described the replacement cost function used in MARS. The cost of replacing an object d_i in client m's cache is calculated with the following equation:

$$\text{cost(i)} = \text{score}_{temp}(i) \times \text{score}_{spat}(i) \times c_i \tag{1}$$

where $score_{temp}(i)$ is the temporal score of the object, $score_{spat}(i)$ is the spatial score of the object and c_i is the cost of retrieving the object from the remote server. The definition of $score_{temp}(i)$ and $score_{spat}(i)$ are given in Section IV-C and Section IV-D respectively.

The goal of the MARS replacement policy is to determine the set of objects S to evict from the client's cache, such that the cost of the objects evicted are minimised. The problem can be formally defined as follows. Find S, such that

$$\min\left(\sum_{d_i \in S} cost(i)\right) \text{ and } \sum_{d_i \in S} objSize_i \geq d_{new} \qquad \text{where } S \subseteq D \tag{2}$$

This problem can be mapped to the classical 0/1 knapsack optimisation problem in the following form:

$$\max\left(\sum_{d_i \in S} cost(i)\right) \text{ and } \sum_{d_i \in S} objSize_i \leq cacheSize_m \tag{3}$$

The objective of the knapsack problem is to find the set of objects that will maximise $\sum_{d_i \in S} cost(i)$ (i.e. finding the maximum benefit that can be gained) while satisfying the cache size restriction. It is well known that the knapsack problem is NP complete, however heuristics and dynamic programming techniques can be used to find a sub-optimal solution in polynomial time. In order to reduce the complexity of MARS and ensure timely cache replacement decisions are made, we have chosen a heuristic algorithm to solve the knapsack problem. When a client needs to insert a new object into its cached and the cache is full, the cached object with the lowest cost is removed until there is enough available space in the cache to store the new object.

C. Temporal Score

The temporal score, $score_{temp}(i)$, is used in the MARS replacement cost function to capture the temporal locality of data access, we define the temporal score of a data object d_i as:

$$score_{temp}(i) = \frac{t_{current} - t_{u,i}}{t_{current} - t_{q,i}} \times \frac{\lambda_i}{\mu_i} \tag{4}$$

where $\lambda_i = \lambda.Pr(d_i \in query)$ is query rate of d_i, and $\mu_i = \mu.Pr(d_i \in update)$ is the update rate of d_i. To avoid division by zero in Equation 4, the first term is set to 1 if the value of $(t_{current} - t_{q,i})$ = 0. Similarly, if μ is 0, then $\frac{\lambda_i}{\mu_i}$ is set to 1.

Two aspects of temporal locality is catered for in the above definition, recency of use and frequency of use. The first term in Equation 4 is a ratio between the time d_i was last updated and the time d_i was last queried. This term deals with the recency of use. It is assumed that objects which were queried recently are likely to be queried again soon, thus $(t_{current} - t_{q,i})$ is used as the denominator to ensure recently queried objects will have a high temporal score. Similarly, $(t_{current} - t_{u,i})$ is placed in the numerator to reduce the temporal score of objects that have been recently update, as it is likely these objects will be updated again soon and evicted from the client's cache. The second term in the equation caters for the frequency of use. It is a ratio of the query rate and update rate of d_i. Objects that are queried frequently but hardly updated will have a high ratio, while objects that are update frequently but rarely queried will have a low ratio. This is based on the assumption that clients are likely to continue querying objects that were frequently queried in the past. Update rate has a role in the calculation as it can be viewed as a risk factor. Objects with a high update probability will have a reduced temporal score because they carry more risk of being evicted from the client's cache due to updates from the server than objects with a low update probability.

D. Spatial Score

As mobile clients move within the wireless network and utilise location dependent information services, their access pattern exhibits spatial locality. A mobile client is more likely to access data objects related to its immediate surrounding and along its path than objects that are far way from the client's current location. In order to capture this characteristic,

we assign a spatial score to each data object to help mobile clients determine the relevance of each object.

The spatial score is calculated based on the following assumptions. An object d_i is more likely to be accessed by client m than data object d_j if:

- $|L_m - L_i| < |L_m - L_j|$, i.e. if d_i is closer to m than d_j
- $area(vs(d_i)) > area(vs(d_j))$, that is if d_i has a larger valid scope area than d_j
- d_i is in the direction of m's movement while d_j is not.

Based on these assumptions, we define the spatial score of an object d_i as:

$$score_{spat}(i) = \left(\frac{1}{|L_m - L_i|} \times \frac{1}{|(\frac{V_m}{\lambda_i} + L_m) - L_i|} \right) \times area(vs(d_i)) \quad (5)$$

where $\frac{1}{|L_m - L_i|}$ is the inverse of the distance between the client and the object d_i's valid scope reference point with $|L_m - L_i| = \sqrt{(Lx_m - Lx_i)^2 + (Ly_m - Ly_i)^2}$. Similar to the calculation of temporal score, if either of the denominator in Equation 5 is equal to 0, the associated term is set to 1 to avoid division by zero.

The second term in Equation 5 accounts for the future movement of the mobile client. $|(\frac{V_m}{\lambda_i} + L_m) - L_i|$ is the distance between the client's anticipated location when it next query object d_i, and the valid scope reference point of d_i. If $|(\frac{V_m}{\lambda_i} + L_m) - L_i| < |L_m - L_i|$ then the client is moving closer to d_i. Otherwise the client is moving away from d_i. The last term in Equation 5 incorporates the size of the object's valid scope into its spatial score.

V. Simulation and Results

Simulation was carried out to evaluate the performance of the proposed scheme. We compare MARS with a number of existing cache replacement policies which includes both non-mobility aware policies (LRU) and mobility-aware policies (FAR, PA and PAID).

The simulation model consists of a network covering a geographical area $map\_width \times map\_height$ in size. Mobile clients can move inside the region covered by the network without any restrictions. Clients change direction at exponentially distributed intervals with a mean time between change of direction equal to $time_{change}$. Different values for $time_{change}$ are used to simulate clients with random movements and clients with regular movements. Small values for $time_{change}$ models mobile clients moving in random paths, while large value for $time_{change}$ will see mobile clients following longer regular paths, and only changing direction occasionally. At the time of a change of direction, a client randomly selects a new direction with a value between 0 to 2π (i.e. 360 degrees). The speed of the client also changes by selecting a new speed uniformly distributed between $speed_{min}$ and $speed_{max}$.

We assume there are N data objects stored at a master server. The size of the objects are uniformly distributed between $objSize_{max}$ and $objSize_{min}$. The size of an object is increased by $overhead$ bytes if valid scope information is attached. Clients query data objects following a Poisson process with a mean rate of λ queries per time unit. Clients communicate with the server via a uplink channel with bandwidth BW_{uplink}, while the server reply the clients via a downlink channel with bandwidth $BW_{downlink}$. Clients have a cache of $sizeCache$, which is used to store data objects, valid scope information and any other parameters needed to maintain the cache. In order to model the utilisation of location dependent services, $percentLDQ\%$ of queries performed by clients are location dependent and $1 - percentLDQ\%$ are non-location dependent. Non-location dependent queries are distributed among the N data objects following a Zipf distribution with a skewness parameter α. Location dependent queries are generated by considering the inverse distance of each data object's valid scope from the client's current location. Given a client currently at location L_m, the probability of it querying an object with a valid scope center reference point at L_i is equal to :

$$Pr(i) = \frac{1}{|L_m - L_i|} \times \frac{1}{\sum_{j \in N} \frac{1}{|L_m - L_j|}} \quad (6)$$

Based on the definition in Equation 6, queries will be distributed among data objects based on their distance from the client current location. Objects that are nearer to the client are more likely to be queried than those further away.

We assume data objects are updated only at the server. A timestamp-based broadcast based cache invalidation method [10] is used to help the clients maintain cache consistency. Table II summarises the parameters used in our simulation and their default values:

The cache hit ratio of MARS and the other cache replacement policies are plotted against cache size in Figure 2. MARS performs up to 20% better than the next best scheme $PAID$ for a small cache, and around 5% better for a large cache. An interesting result is that the temporal based LRU replacement policy achieved cache hit ratio that is just slightly lower compared to other mobility-aware policies. This can be explained by realising spatial locality and temporal locality

TABLE II

Simulation Parameters

Parameter	Description	Value
map_width	Map Width	4000m
map_height	Map Height	4000m
$simTime$	Simulation Duration	10000s
N	Number of data objects	5000
C	Number of mobile clients	30
λ	Mean query rate	1 query/s
μ	Mean update rate	0.1 update/s
$cacheSize_m$	Size of client m's cache	10% of N
$objSize_{max}$	Maximum size of objects	200 bytes
$objSize_{min}$	Minimum size of objects	100 bytes
$overhead$	Valid scope information overhead	9 bytes
$percentLDQ$	Percentage of LDD queries	80%
BW_{uplink}	Uplink bandwidth	200kbps
$BW_{downlink}$	Downlink bandwidth	20kbps
α	Zipf parameter	0.7
$time_{change}$	Time between change of direction	100s
$speed_{min}$	Minimum client movement seed	1m/s
$speed_{max}$	Maximum client movement speed	25m/s

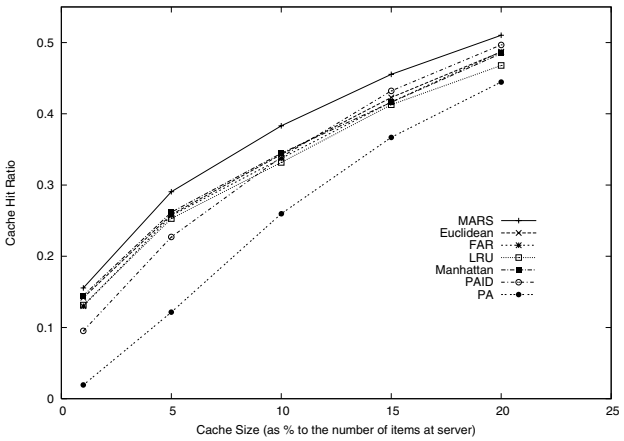

Fig. 2. Cache Hit Ratio vs. Cache Size

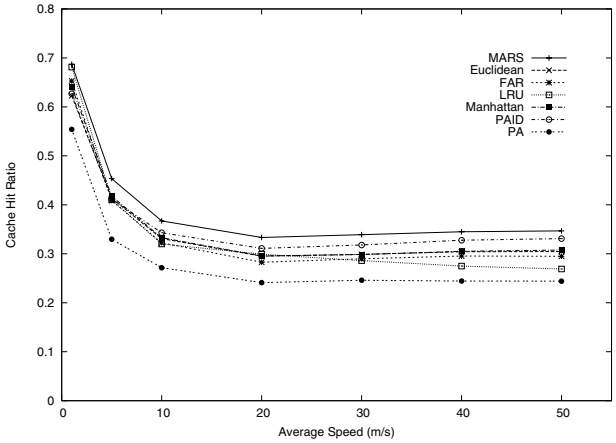

Fig. 4. Cache Hit Ratio vs. Average Client Speed

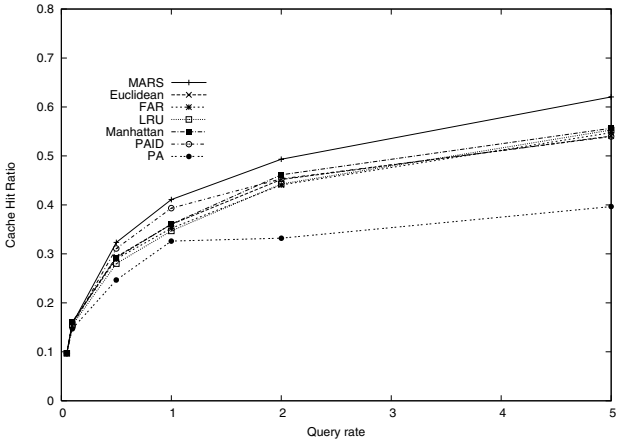

Fig. 3. Cache Hit Ratio vs. Query rate

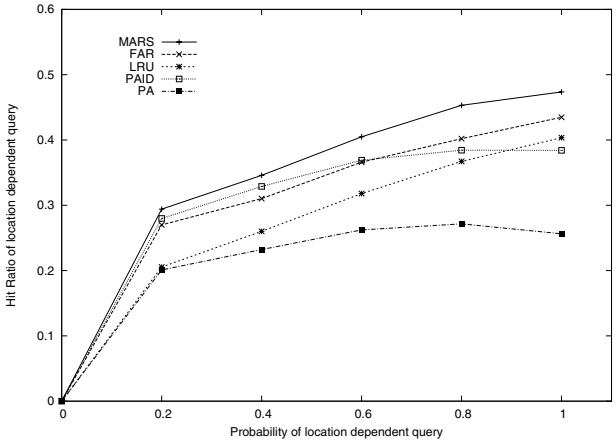

Fig. 5. Hit Ratio of LDD query vs. Probability of LDD query

are not independent properties. The access pattern of a client who frequently accessing local information will also exhibit a high level of temporal locality governed by the location of the objects. We can imagine as a client moves from one location to another, the ”hot spot” of its temporal access pattern is shifting from the group of data objects at its current location to another group of data objects at the new location.

Figure 3 shows the cache hit ratio against query rate. We found that MARS consistently outperforms the other cache replacement schemes. The PA scheme performs significantly worst than the other policies as it only consider the area of object's valid scope. The cache hit ratio increases as query rate increase because when query rate is high, more number of queries is executed at each location. This allows the client to fill its cache with local information more quickly, resulting in more cache hits.

Client cache hit ratio is shown against average client movement speed in Figure 4. It can be seen that very high cache hit ratio can be achieved for clients with low movement speed. Similar to an increase in query rate, when the movement speed is low, clients spend more time at each geographic location. This allows the clients' cache to be filled with more information about local areas, which result in higher cache hit ratio. As the movement speed of clients increase, the cache hit ratio drops quickly. The mobility-aware approaches (MARS, FAR, PAID and PA) however are able to maintain a steady cache hit ratio after the average client speed reach around 10m/s, while the cache hit ratio of LRU continues to drop. This can be explained by realising that as the client's speed increases, the temporal locality of its access pattern decreases (due to frequent change in location), resulting in a drop in performance from LRU. By anticipating clients' future location when making cache replacement decisions, MARS is able to maintain good performance even for clients travelling at high speed.

The graph in Figure 5 shows that the mobility-aware cache replacement policies perform significantly better than the temporal base policy (LRU) when it comes to location dependent queries. When the percentage of location dependent queries are low (20%), MARS, FAR and PAID achieves a hit ratio of 30% on location dependent queries compared to 20% achieved by LRU and PA. As the probability of location dependent queries increase, the performance of LRU improves because in this situation, most of the cache space is used by LRU to

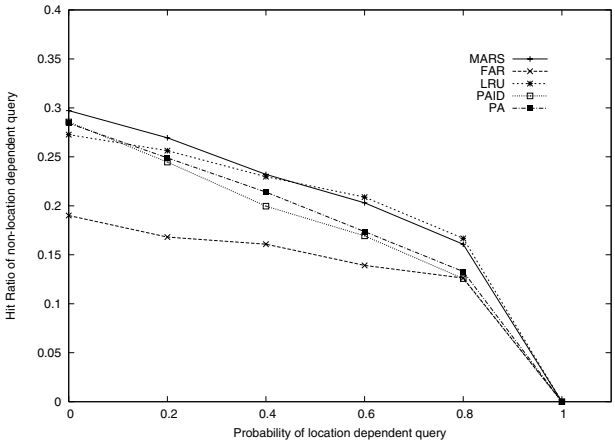

Fig. 6. Hit Ratio of non-LDD query vs. Probability of LDD query

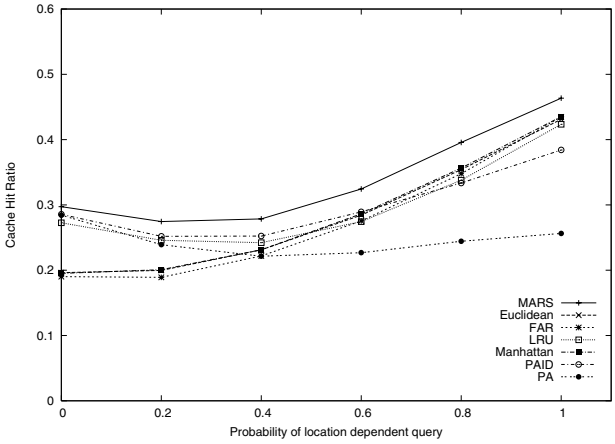

Fig. 7. Cache Hit Ratio vs. Probability of LDD query

store objects obtained from location dependent queries.

Figure 6 shows the hit ratio of non-location dependent queries against the probability of location dependent queries. Again, the hit ratio of MARS is significantly higher compared to other mobility-aware replacement policies. At high location dependent query probability, LRU performs slightly better than MARS because more cache space is used by MARS to store objects obtained from location-dependent, thus reducing the number of objects cached from non-location-dependent queries. Overall however, when considering both location dependent query hit ratio and non-location dependent query hit ratio, MARS is able achieve better performance compared to all the other schemes as we will show in the next graph.

Lastly, Figure 7 shows the overall cache hit ratio against the probability of location dependent queries. The best performance is achieved by each approach when there is a high number of location dependent queries. When the probability of location dependent queries is high, client caches are filled with information relevant to the clients' current location, resulting in more queries being satisfied by the cache. As illustrated in the graphs, MARS performs best compared to other existing policies across the whole range of location-dependent query probabilities.

VI. Conclusion

In this paper, we have presented a mobility-aware cache replacement policy that is efficient in supporting mobile clients using location dependent information services. The proposed policy, MARS, takes into account both the temporal properties and spatial properties of clients' access pattern to improve caching performance. A range of important factors including access probability, update probability, client location, client velocity are incorporated into the cost function in MARS to ensure each client utilises its limited cache space efficiently. Test results show that MARS provide efficient cache replacement for mobile clients and is able to achieve a 20% improvement in cache hit ratio compared to existing replacement policies.

As part of our future work, we intend to develop detailed analytical models of MARS and the other mobility-aware cache replacement policies investigated in this study. We are also interested in the possibility using path prediction algorithms to detect regular client movement patterns in order to improve cache replacement decisions.

Acknowledgment

This project is supported by the ARC (Australian Research Council - under the Linkage-Project scheme, no. LP0218853) and SUN Microsystems grant no.7832-030217-AUS.

References

[1] J. Xu, X. Tang, D. L. Lee, and Q. Hu, "Cache coherency in location-dependent information services for mobile environments," in *Proceedings of the International Conference on Mobile Data Access*, 1999, pp. 182 –193.

[2] Q. Ren and M. H. Dunham, "Using semantic caching to manage location dependent data in mobile computing," in *Proceedings of the International Conference on Mobile Computing and Networking*, 2000, pp. 210–221.

[3] I. Jung, Y. You, J. Lee, and K. Kim, "Broadcasting and caching policies for location-dependent queries in urban areas," in *Proceedings of the International Workshop on Mobile Commerce*, 2002, pp. 54–60.

[4] M. H. Dunham and V. Kumar, "Location dependent data and its management in mobile databases," in *Proceedings of the 9th International Workshop on Database and Expert Systems*, 1998, pp. 414–419.

[5] L. Yin, G. Cao, and Y. Cai, "A generalized target-driven cache replacement policy for mobile environments," in *Proceedings of the IEEE Symposium on Applications an the Internet*, January 2003, pp. 14–21.

[6] B. Zheng, J. Xu, and D. . Lee, "Cache invalidation and replacement strategies for location-dependent data in mobile environments," *IEEE Transactions on Computers*, vol. 51(10), pp. 1141–1153, October 2002.

[7] E. O'Neil and P. O'Neil, "The LRU-k page replacement algorithm for database disk buffering," in *Proceedings of the ACM SIGMOD conference*, May 1993, pp. 296–306.

[8] A. Sedim, M. Dunham, and V. Kumar, "Location dependent query processing," in *Proceedings of the MobiDE workshop*, 2001, pp. 47–53.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the INFOCOM*, 1999, pp. 126–134.

[10] K. Lai, Z. Tari, and P. Bertok, "An analytical study of broadcast based cache invalidation in mobile computing networks," in *Proceedings of the 11th International Conference on Cooperative Information Systems*, November 2003, pp. 554–572.