Dynamic Data Management for Location Based Services in Mobile Environments

Shiow-yang Wu Kun-Ta Wu
Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien, Taiwan, R. O. C.

Abstract

We characterize the dynamic data management problem for location based services(LBS) in mobile environments and devise a cost model for servicing both location independent and location dependent data. The cost analysis leads to a set of dynamic data management strategies that employs judicious caching, proactive server pushing and neighborhood replication to reduce service cost and improve response time under changing user mobility and access patterns. Simulation results suggest that different strategies are effective for different types of data in response to different patterns of movement and information access.

1 Introduction

The advances in portable devices and wireless communication technologies enables a new form of services named *location based services(LBS)*which deliver location dependent and context sensitive information to mobile users. Typical examples of such services include local maps, local weather, traffic condition, tour guide, and shopping guide, etc. A key characteristic of LBS is that the same service request may need to be answered with complete different results as the user changes his/her location or the targets move. Because of the highly dynamic nature of the problem, traditional information management techniques are not well suited for LBS. Developing proper infrastructure, location management, as well as data management strategies for LBS has been a major challenge to both wireless service providers and application developers. To answer this challenge, we first analyzed the nature of the problem and proposed a classification of potential application domains. Based on the analysis, we characterized the *dynamic data management* problem for LBS and devised a general service architecture flexible enough to provide dynamic access to both location dependent and location independent data. We further devised a cost model to analyze the dynamic behavior of the system as well as the service cost. The cost analysis leads to a set of dynamic data management strategies that employs judicious caching, proactive server pushing and neighborhood replication to improve service response time under changing user mobility and access patterns. Simulation results suggest that different strategies are effective for different types of data in response to different patterns of user movement and information access.

The rest of the paper is organized as follows. Section 2 provides a survey of related issues and research work. Section 3 presents our framework for problem analysis, application characterization and domain classification which leads to the challenging dynamic data management problem. In Section 4, we proposed a system architecture for location-based information services to answer the challenge. Based on the service architecture, we devised a cost model in Section 5 to facilitate the design of a set of dynamic data management strategies as well as the analysis of the system behavior. Simulation results presented in Section 6 demonstrate the feasibility and performance of our framework toward the construction of highly responsive systems for location-based mobile information services. Section 7 concludes the paper.

2 Related Work

Data management in mobile computing environments is especially challenging for the need to process information on the move, to cope with resource limitation, and to deal with heterogeneity. Among the applications of mobile data management, LBS have been identified as one of the most promising area of research and development [1]. The problem has also been studied under various terms such as location-aware, context-aware, or adaptive information systems [8, 9]. Many of the previous work on LBS treated location as an additional attribute of the data tables[4, 10]. LBS queries can be processed like ordinary queries except with additional constraints on the location attribute. Caching techniques specially tailored for LBS or mobile computing environments in general have also been a major research area [2, 4]. Semantic caching techniques employed semantic descriptions of cached items to facilitate better cache admission and replacement decisions that are responsive to the user movement[3, 6]. Research effort on moving objects database are also related to our work in the need to process data in a highly dynamic environment [5, 7, 11]. Our work differs

from the previous works in that we consider the problem from a global point of view. Our strategies take into account both location dependent and location independent queries to information from centralized broadcasting channel such as the stock pro or world news, as well as localized services such as the nearby restaurants or local weather. The dynamic nature of our approach also facilitates timely responses to rapid changes in access and/or mobility patterns.

3 Mobile Information Service Characterization and Classification

Based on the characteristics of data and their typical usage patterns, we classify information service applications in mobile environments along four dimensions.

- Source of data : *central* vs. *local*
  The data can be from a *central* broadcasting channel such as the CNN world news, or from a *local* service station such as area map or tourist attractions.
- Period of validity : *static* vs. *dynamic*
  The period of validity of the data can be relatively *static* such as various kinds of traffic timetables, or *dynamic* such as the current traffic condition.
- Target audience : *public* vs. *personal*
  The intended target audience of the data can be to the general *public* such as a public announcement, or to a particular *person* such as email or personalized news.
- Location dependency : *location independent* vs. *location dependent*
  The nature of the queries and results can be *location independent* such as the stock pro, or *location dependent* such as the nearby restraints within a given range.

In Table 1, we give several examples of the mobile information services along these four dimensions. Note that some services may appear in more than one places. Area news, for example, may be provided by a central news agency or by a local station. The classification helps us in designing a general service architecture that can be easily adapted to incorporate different data management strategies to fit the characteristics of different application domains. For example, information having the properties of central, public, static, and location independent are good candidates for traditional techniques such as keeping cached copies at the client devices or local servers. On the other hand, the data of local, personal, dynamic, and location dependent applications are best handled with dynamic strategies that can quickly respond to the changes in user location or data status. In general, central data sources call for proper dissemination infrastructure while local data need good caching mechanism. Static data can be efficiently served with proper replication schemes while the processing of dynamic data may consider prefetching or semantic caching techniques. The access patterns to public data are often predictable and therefore amenable to proactive dissemination techniques such as pushing hot data closer to the clients. Personalized data, however, may need to be processed with dynamic profiling and data migration techniques. The location dependency of data raises a new issue of answering *range queries* such as the nearby motels within 5 miles of the current location. Supporting such services requires neighboring base stations(local servers) to exchange information on target objects. Some optimization techniques can be adopted such as the prefetching and caching of popular range query targets.

As a summary, the challenge is to design a flexible service architecture and dynamic data management strategies to provide highly responsive and location dependent information access in resource constrained and rapidly changing environments for the application domains discussed above. We call it the *dynamic data management* problem for location based services in mobile environments. The dynamic nature of the problem is strongly emphasized in here since both the clients and targets may change locations at any time.

4 Location-Based Service Architecture

Based on the application classification and service characterization, we devised a general system architecture as the basis for our cost modelling and strategy design. Our system were designed with several criteria in minds. First of all, the architecture must reflect the current and foreseeable future status of the wireless networking technologies. The system must also be flexible enough to allow the installment of different data management strategies on different classes of applications. Finally, since we want to serve both location dependent as well as location independent data, the architecture must take the source characteristics into consideration. Figure 1 depicted our architectural design consisting of the central server, local server, and the client device.

The central server is used to model a service site for centralized data such as the New York Stock Exchange. In addition to a central information database, a push unit is included to facilitate server-initiated pushing strategies that proactively send selected data items toward the clients. Since the downlink bandwidth is usually much larger and cheaper than uplink connection, pushing techniques turn out to be efficient and valuable tools with little extra cost.

The local server is the data manager and wireless information server for a single cell. Each cell is assumed to have a unique local server which provides wireless access for all the clients in its cell and acts as a bridge between the central server and the client devices at the same time. It is the

		Location Independent		Location Dependent	
		Static	Dynamic	Static	Dynamic
Central	Public	global public announcement, traffic timetable, ...	news, stock pro, mobile banking, ...	area transportation info, area map, tourist attractions, ...	area news, area weather, traffic condition, ...
	Personal	email, medical record, personal note, student grade, ...	personalized news, investment guide, chat messages, ...	personalized tour arrangement, route planning, ...	personalized area news, exhibition guide, ...
Local	Public	local public announcement, service directory, ...	area news, area weather, area broadcasting, ...	nearby hotels and restaurants, local transportation info, ...	local news, local traffic, ...
	Personal	personal information (schedule, to do list, shopping list), ...	personalized area news, push services, advertising, ...	nearby specified targets, personalized local attractions, ...	current location, route, nearby moving targets, ...

Table 1: Mobile information service classification

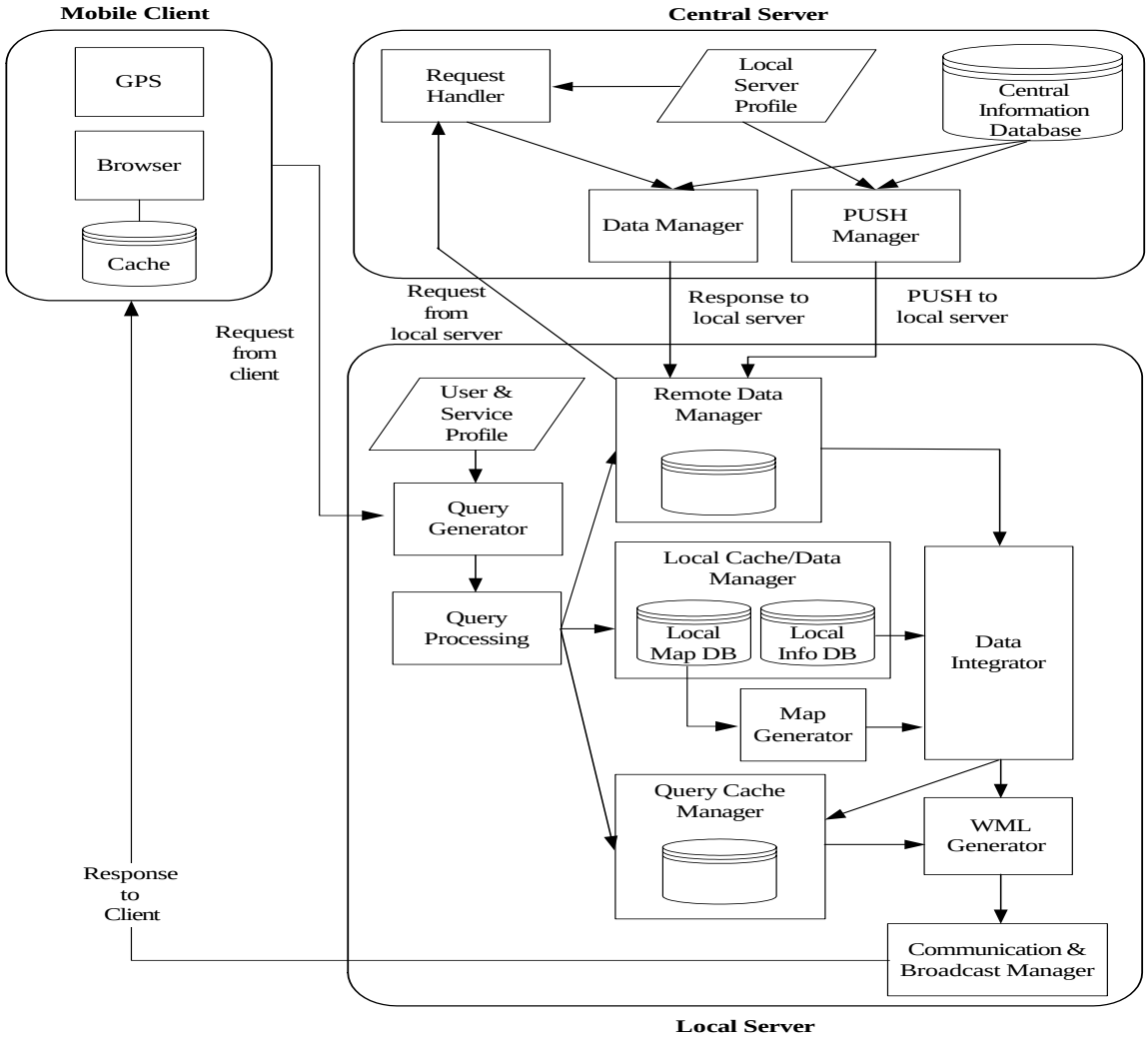

Figure 1: LBS System Architecture.

center for managing local information as well as the key player to provide location based services. All local servers are connected to the Internet via fixed network and therefore can send information to each others with almost negligible delay in comparison with wireless access. This is an important factor since neighboring local servers must work closely together to provide efficient location-based services. The cache and data manager is responsible for maintaining the information received from the central server as well as other local servers. A local server is also equipped with a map database for answering location dependent map queries.

The client is any end user device that is capable of wireless communication as well as user interface services. This is probably the branch of the wireless technologies that evolves with the fastest pace. Therefore we do not presume the computing power and storage capacity of a client device. We only assume that it can send out information requests via a wireless link, can do some local processing if required, and has a client cache for keeping frequently accessed data items. The most distinctive feature of a client is that it moves. A client can change its position at will, in and out of a cell, from one cell to another, without the obligation to notify any server in advance. Since a client can issue a query at any time any where, this is especially challenging for information service providers. In our architectural design, a client always sends requests to the local server of the cell where the client resides. The target objects may be available right at the client cache, at the local server of the same cell, from the local servers of other cells, or from central servers.

To make our design general and flexible, we assume an object-based service environment. The information requests are categorized into three types of queries:

- *Object queries* : that target specified objects.
- *Range queries* : that target objects located within a range restriction and satisfy certain constraints.
- *Map queries* : for area map information.

The service of map queries is out of the scope of this paper. We will discuss only object and range queries. Object queries are like traditional type of queries that are issued by explicitly naming the target objects or by giving the object classes and the constraints that must be satisfied. Range queries are location dependent queries to locate the desired type(s) of object(s) within a specified distance around the client. "List all the restaurants within 5 miles of my current location." for example, is a typical range query. "Give me the hotel reservation phone number of the Pleasant Plaza." on the other hand, is an object query since the target object is directly specified. We note that if the target object(s) is(are) located within the same cell as the client, the local server of the cell can answer the query without consulting other servers. If, however, the target objects reside in other cells or in a central server, the local server must request the desired objects from other servers and pass them to the client when the objects arrive. Since the client can move at will, a target that can be accessed from within the same cell may have to be accessed from other cell for the next request. This is also true for the range queries since any target object can be in and out of range as the client moves. Static query processing strategies have little use in such environments. Dynamic data management strategies that can effectively locate the desired objects as well as quickly respond to the location changes are in order.

5 Cost Model and Dynamic Data Management Strategies

To facilitate the design of dynamic data management strategies for location based services, we devise a simple yet effective cost model that abstract away nonessential details of the wireless environment and characterize only the key players and dominant cost factors. As depicted in Figure 2, a location based service environment is modelled as consisting of four major types of abstract entities:

C The client device.

L The local server of the cell where the client resides.

L' The neighboring servers, i.e. the local servers of the neighboring cells.

S The central information server.

The client access the network through a wireless connection to the local server while all other connections between servers (local and/or central) are through wired links. As the current technology stands, the cost of a wireless link is much higher than that of a wired link. The connection cost between neighboring servers is relatively lower than the cost of accessing from a remote central server. We therefore classify the communication cost into three categories:

d The wireless communication cost between a client and its local server.

l The communication cost between neighboring local servers.

r The cost of accessing information from a remote central server to the local server.

Based on the abstraction and characterization, we proposed three sets of dynamic data management strategies, namely *judicious caching*, *proactive pushing*, as well as *neighborhood replication* to improve service response time and reduce communication cost. The first two sets of strategies are designed for object queries that target information

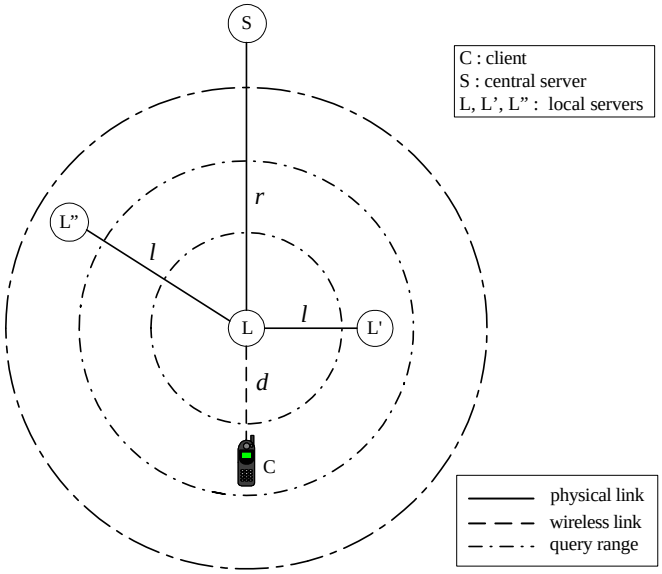

Figure 2: LBS cost model — Central, location dependent, and range services

largely from central data services while the third set is for location based range queries. We then conducted detail cost analysis to understand the exact situations for these strategies to be cost effective. This help us in determining when to apply and adjust the strategies for better services.

Judicious Caching

As discussed in section 4, both local servers and clients have caches to retain downloaded information. If certain types of objects published by a particular central server were accessed frequently by the clients of a cell, a simple idea is to maintain cache copies of all such objects at the local server and keep them always up to date. In this way, the clients in the cell can always access fresh copies of the objects directly from the local server without further delay. We termed this *judicious caching* since the target objects to be cached can range from the entire central server to certain categories or to a particular class of objects. However, any caching strategies bring about maintenance cost as well. To keep the cached copies up to date, a local server must spend the extra cost of getting new copies from the central server whenever updates occur. The question is then how to determine when to apply such a strategy to what types of data objects? Our proposed solution is to first partition the data at the central server according to their publishing characteristics. Data objects that are closely related and updated at the same or similar rate are grouped together. Then for each group of target objects, we perform a detail analysis of the potential benefit of caching those objects in a particular local server based on a set of dynamically maintained access statistics listed as follows.

- R The total number of requests to the target objects issued by the clients in the cell.
- U The total number of updates to the target objects.
- c The average hit rate of the client cache.
- h The average hit rate of the local server cache.

The caching of the target objects is beneficial if the sum of the access cost and the update maintenance cost is lower than conventional on-demand caching. That is,

$$\begin{aligned} Rc + R(1-c)d + Ur \quad &< \quad Rc + R(1-c)hd + \\ & \qquad R(1-c)(1-h)(d+r) \end{aligned}$$

With simple algebraic manipulation, we have

$$U \quad < \quad R(1-c)(1-h)$$

Since $R(1-c)(1-h)$ is the number of requests actually received by the central server to the target objects, the equation above means that as long as the update rate to a particular class of objects is smaller than the request rate experienced on the central server, then it is beneficial to cache that class of objects at the local server and keep them always up to date. This is a reasonable decision to make since if the requests to the same class of objects are much higher than the update rate, the cache maintenance cost can easily be covered by the saving on access cost. Therefore, the judicious caching strategy is to maintain a class of objects at the local server cache and keep them up to date if the update and request rates satisfy the condition above. When the request rate drops at latter time, or the update increases such that the condition

no longer holds, the local server can stop requesting further maintenance update from the central server.

Proactive Pushing

A natural alternative to the judicious caching strategy is to push a certain percentage of "hot" data to the local server in an attempt to minimize the access cost and response time experienced by the clients. We call this *proactive pushing* since it is the central server that proactively pushes selected objects to the local server. Similar to judicious caching, we need to determine exactly when and how to push the data. With two additional parameters, a detail analysis of the access cost with respective to conventional on-demand caching reveals the condition for applying the strategy.

- p The percentage of the "most frequently requested" data to be proactively pushed to the local server.
- q The percentage of update to the proactively pushed data with respective to the total number of updates U.

In other words, among the requests received by the local server, p percentage of which can be responded locally since they have been pushed from the central server to the local server. We also pay the maintenance cost for q percentage of the total update since we must keep them up to date. Similar to judicious caching, we must have a lower cost than conventional on-demand caching for such a strategy to be effective. We therefore have the following cost formula.

$$\begin{aligned} & Rc + R(1-c)pd + R(1-c)(1-p)(d+r) + Uqr \\ < \quad & Rc + R(1-c)hd + R(1-c)(1-h)(d+r) \end{aligned}$$

With simple algebraic manipulation, we obtain the formula

$$Uq \quad < \quad R(1-c)(p-h)$$

which means that if the extra number of maintenance updates due to proactive pushing is less than the additional number of requests that can be satisfied locally, then it is worthy of the cost. We note that whenever p increases, q also increases. When the update cost overwhelms the request saving, further pushing can do more harm than good. Therefore in the proactive pushing strategy, the central server selects an initial p and maintains the statistics discussed above. When the requests to the "hot" data increase, p can be increased accordingly. If the update rate to the pushed data becomes higher, then the central server can reduce the pushing percentage.

So far, we have assumed that all objects at the central server are considered together. There is no reason why we can't partition the data in a similar way as judicious caching and maintain a separate p for each data group. In such case, judicious caching becomes a special case of proactive pushing with p equals 100%.

Neighborhood Replication

For range queries, the target objects are no longer from central servers but from the local servers of the current and neighboring cells within the specified range. Data management strategies must be tailored accordingly. In general, different range queries may overlap with each others. If similar range queries were repeatedly issues by the clients, then it would be beneficial to replicate the data objects of the most frequently accessed neighboring cells at the local server. We name this *neighborhood replication*. The set of replicated cells is called the *replication range* of the strategy. To maintain the cost effectiveness of such a strategy, we need to record a different set of access parameters as follows.

- R The total number of range requests issued by the clients in the cell.
- G The total number of updates to the location dependent data objects.
- c The average hit rate of the client cache.
- m The number of neighboring servers to replicate.
- n The average number of servers within the query ranges but not replicated.
- k The number of neighboring servers to be added to ($k > 0$) or excluded from ($k < 0$) the replication range.
- n' After adjusting the replication range, the average number of servers within the query ranges but not replicated.

The idea is to maintain replica of location dependent data from selected neighboring cells and keep them up to date. On each evaluation period, we try to evaluate the potential benefit of adding k additional neighboring servers into the replication range. Naturally, the inclusion of more replica is beneficial if the expected cost is less than the cost without them. This can be expressed by the following formula.

$$\begin{aligned} & Rc + R(1-c)d + R(1-c)n'l + G(m+k)l \\ < \quad & Rc + R(1-c)d + R(1-c)nl + Gml \end{aligned}$$

The formula can be simplified as

$$\begin{aligned} G \quad & < \quad R(1-c)(n-n')/k \quad \text{if } k > 0 \text{ and } n - n' > 0 \\ G \quad & > \quad R(1-c)(n-n')/k \quad \text{if } k < 0 \text{ and } n - n' < 0 \end{aligned}$$

We note that when adding more servers into the replication range (i.e. $k > 0$ and thus $n - n' > 0$), $R(1-c)(n-n')$ is the number of requests to the neighboring servers that can be saved due to replication. Therefore, the first formula states that if the number of updates to the location dependent data is less than the average number of neighboring requests that can be saved, then we can go ahead replicate those neighboring

servers. On the other hand, if we were to remove some servers from the replication range (i.e. $k < 0$ and thus $n - n' < 0$), $R(1-c)(|n-n'|)$ becomes the additional number of requests that can no longer be satisfied by the cache and must be sent to the neighboring servers. Therefore, the second formula states that if the number of updates is greater than the average increase in requests to the neighboring servers, then we should shrink the replication range.

The neighborhood replication strategy can be summarized as follows. Each local server determines an initial replication range on startup and replicates the location dependent objects from the neighboring servers. On each evaluation period, a local server determines whether to grow or shrink the replication range according to the decision equations. We note that a local server does not need to actually replicate the additional k neighbors in order to obtain n' and G. Since we know where each objects came from. After servicing the client requests, we can always make the assumption of any desirable replication configuration and derive the values of these access statistics whenever we need them.

As a summary, all three types of strategies are designed to dynamically respond to the rapid changes in access contexts and request/update patterns. By carefully analyzing the relative benefits of caching and/or replication, we can determine exactly when to apply and how to adjust the strategies to facilitate highly responsive location based services.

6 Simulation and Evaluation

We have designed and implemented a simulation system (Figure 3) consisting of an environment generator, a request/update generator, as well as a simulation engine. On each experiment, we first select appropriate values for the set of cost model parameters (i.e., d, l, and r) and the set of environment parameters (such as the number of cells, clients, data objects, requests, and updates). The former is for cost accumulation while the latter is for generating a request file. Then a set of server side parameters (such as the cache size, push percentage, and initial replication range) is supplied to initialize the simulation engine. The simulation proceeds through a specified number of rounds. During each round, we simulate the execution of the requests/updates and maintain the statistics needed for each data management strategy accordingly. The changes in access statistics may trigger the dynamic adaptation of the strategy which may in turn affect the access statistics thereafter. In addition to the execution simulation and strategy application, we also keep records of the access cost of each request based on the cost model parameters and write to the result file.

Figure 4 represents the results of performance comparison between conventional on-demand caching and judicious caching with varying cache sizes as well as request/update ratios. The size of the cache is the capacity measured in terms of the percentage of data objects received from the central server. We can see very clearly that the judicious caching strategy is superior in all cache sizes especially when the number of requests are higher than the updates. When the updates are much higher than the request rate, two strategies coincide since caching provides little benefit in such a scenario.

Figure 5 demonstrates the combined effect of request/update ratio and cache size on judicious caching. Due to the judicious nature, our strategy can effectively explore cache resource and take advantages of the situations when the requests are higher than the updates.

In order to understand the responsiveness of judicious caching in reaction to the rapid change of access patterns, we have conducted an experiment of 100 simulation rounds with request/update ratio set to 20/80 for the first 50 rounds and rapidly changed to 80/20 for the rest 50 rounds. The result shown in Figure 6 not only demonstrates the performance advantage of judicious caching over conventional on-demand caching but also the adaptation capability of our approach.

For evaluating the proactive pushing strategy, we fixed the cache size and varied the push percentage to obtain the result presented in Figure 7. Contrary to our expectation, while proactive pushing performs consistently superior than conventional on-demand caching, higher push percentage does not provide significant advantage. After careful examination of the data set, we found that this may be due to the random generation of request patterns which tends to access data evenly without hot spot. In such case, proactive pushing does not provide significant advantage since no data are "really" hot.

A rather interesting result was obtained from the response time comparison between conventional on-demand caching and neighborhood replication on range queries. Figure 8 shows that both strategies are relatively independent of the request/update ratio. This is because the clients are constantly moving which results in the cache being of little use. We can also see that neighborhood replication consistently outperforms on-demand caching by a large gap. This is due to the dynamic nature of our strategy. When the request/update ratio is high, the strategy can take advantage of the situation by increasing the number of neighbors into the replication scope. On the other hand, when the reverse is true, the strategy can automatically decrease the number of replication sites to save the replica maintenance cost.

As a summary, dynamic data management is essential for location based services. Even for location independent data, dynamic strategies are still required since the clients can move from one place to another. By characterizing the data sources and access trends, we have designed effective strategies that successfully balanced between the saving in access cost and the increase in maintenance cost.

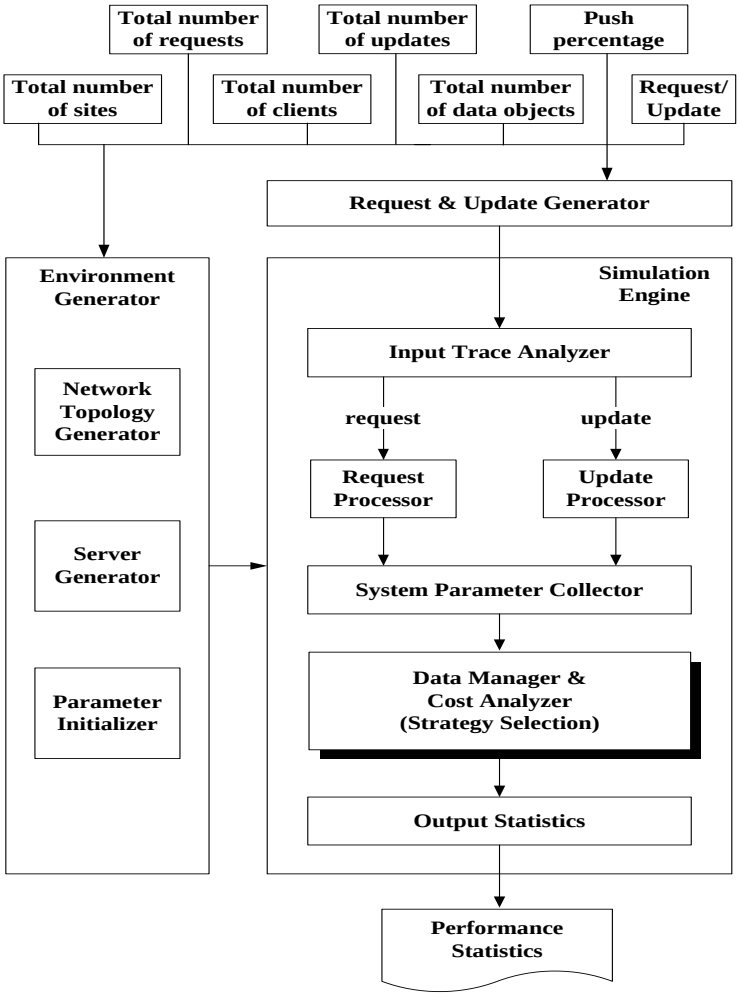

Figure 3: Simulation system architecture.

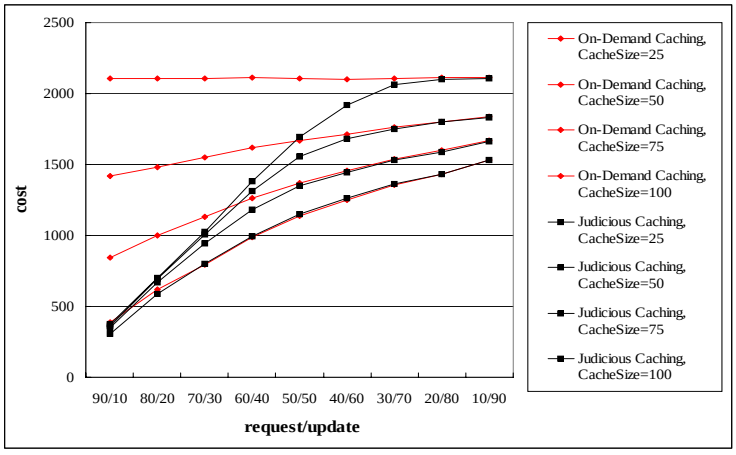

Figure 4: Performance comparison of the judicious caching strategy with on-demand caching.

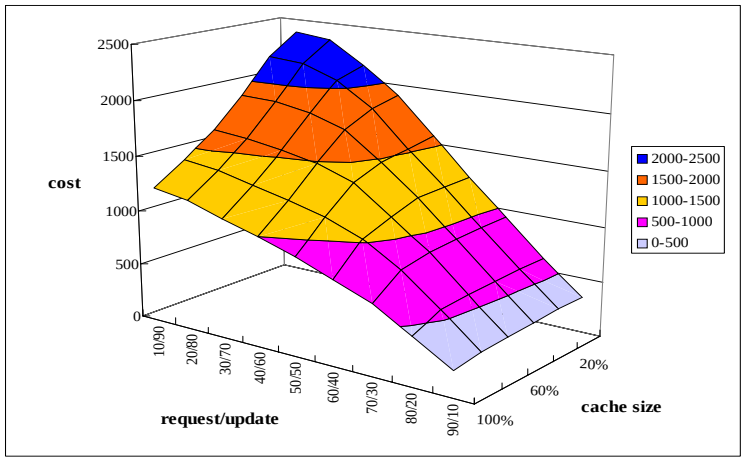

Figure 5: The combined effect of request/update ratio and cache size on the judicious caching strategy.

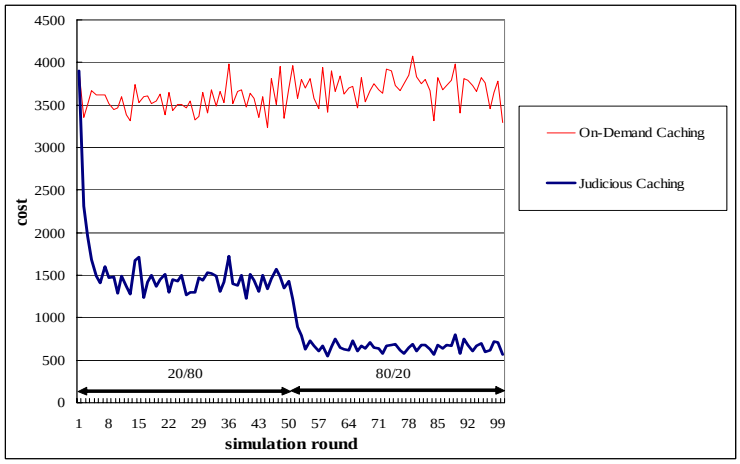

Figure 6: The adaptation capability of the judicious caching strategy in response to the change on request/update pattern.

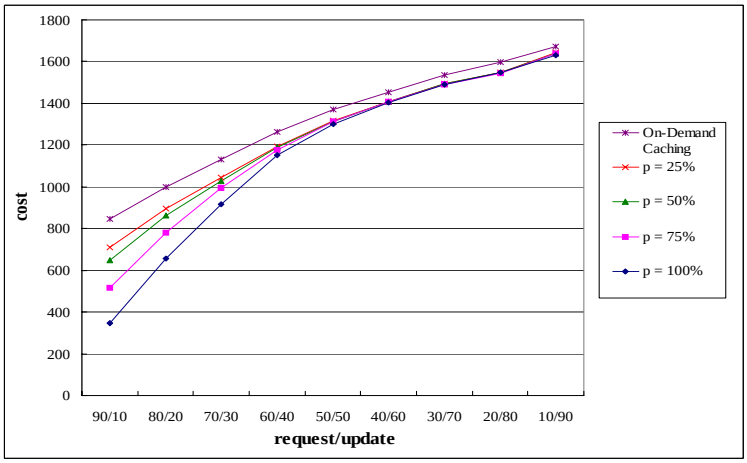

Figure 7: Execution cost of the proactive pushing strategy with varying push rate.

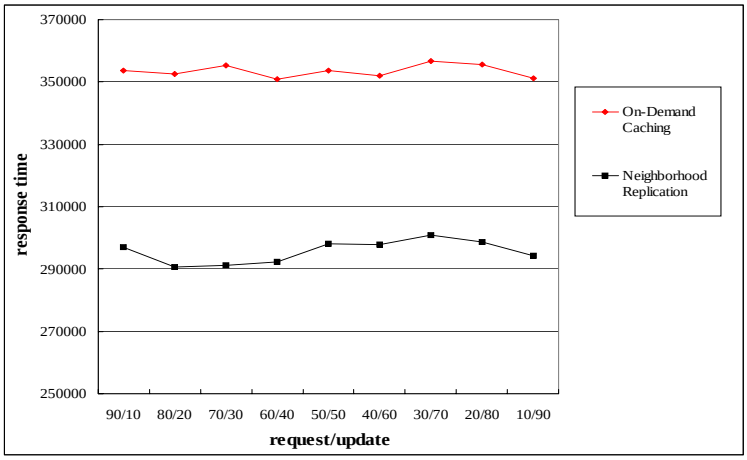

Figure 8: Response time comparison of the neighborhood replication strategy against on-demand caching on range queries.

7 Conclusions and Future Work

We have proposed a service framework, system architecture, simple but effective cost models and dynamic data management strategies for location based services in mobile computing environments. Analytical and simulation results successfully demonstrate not only the feasibility but also the effectiveness of our approach. The most distinctive features of the proposed strategies are their capability to dynamically respond to changes in the mobility and/or access patterns. We plan to further extend our system framework and cost model to handle information services on moving data sources, not just moving clients. We are also evaluating the potential of employing mobile agent technologies to support continuous location based service queries.

References

[1] Daniel Barbara. Mobile computing and databases - a survey. *IEEE Transaction on Knowledge and Data Engineering*, 11(1):108–117, January/February 1999.

[2] B. Y. Chan, A. Si, and H. V. Leong. Cache management for mobile databases: Design and evaluation. In *Proceedings of 14th ICDE*, pages 54–63, 1998.

[3] Shaul Dar, Michael J. Franklin, B. Jonsson, Divesh Srivastava, and Michael Tan. Semantic data caching and replacement. In *Proceedings of VLDB*, pages 330–341, 1996.

[4] M. H. Dunham and V. Kumar. Location dependent data and its management in mobile databases. In *Proceedings of DEXA Workshop*, pages 414–419, August 1998.

[5] R. H. Guting, M. Erwig M. H. Bohlen, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, March 2000.

[6] Qun Ren and Margaret H. Dunham. Using semantic caching to manage location dependent data in mobile computing. In *MobiCom'00: The Sixth Annual International Conference on Mobile Computing and Networking*, pages 210–221, August 2000.

[7] Simonas Saltenis and Christian S. Jensen. Indexing mobing objects for location-based services. In *Proc. 18th International Conference on Data Engineering*, pages 463–472, 2002.

[8] M. Satyanarayanan, B. Noble, P. Kumar, and M. Price. Application-aware adaptation for mobile computing. *Operating System Review*, 29, January 1995.

[9] B. Schilit, N. Adams, and R. Want. Context-aware mobile applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., December 1994.

[10] Ayse Y. Seydim, Margaret H. Dunham, and Vijay Kumar. Location dependent query processing. In *MobiDE'01: 2nd ACM International Workshop on Data Engineering for Mobile and Wireless Access*, pages 47–53, 2001.

[11] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang. Moving objects databases: Issues and solutions. In *Proceedings of the 10th International Conference on Statistical and Scientific Database Management*, pages 111–122, Capri, Italy, 1998.