# Location Based Access to Moving Data Sources

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

Wei-chung Ko

7-10F, No. 135, Jian-Yi Road
Chung-Ho City, Taipei Hsien
Taiwan, Republic of China

**Abstract**

Location based services in mobile environments have been recognized as a challenging problem that calls for new service models and dynamic strategies. The problem becomes even harder when not only the clients but also the data sources can move. We tackle the problem by providing a general service architecture that facilitates the employment of mobile codes. We then device a concise but effective cost model for information services to instantaneous and continuous queries on both designated objects and location based range data. The cost analysis leads to a set of data management strategies that employs mobile codes with intelligent caching and proactive pushing to reduce service cost and improve response time. For each strategy, we present analytical characterization of the execution cost to pinpoint the exact situation for applying the strategy. Simulation results suggest that the use of mobile codes can be very effective in location based services to moving data sources.

## 1 Introduction

In a recent work[13], we characterized *location based services* (LBS) as the delivery of location dependent and context sensitive information services to mobile users. A key characteristic of LBS is that the same request may need to be answered with completely different results as the user location changes. The problem becomes even harder when the target data sources can also move. Traditional data management techniques are not well suited for LBS, let alone moving data sources. Developing proper infrastructure and strategies has been a major challenge to both service providers and application developers. To answer this challenge, we devise a general architecture that facilitates conventional data management and mobile code techniques. We provide service strategies for *instantaneous* as well as *continuous* access to both location dependent/independent data. In particular, mobile codes can be launched to travel alone with the moving data sources and continuously report the status changes. They can also proactively push "hot" data toward the base stations and clients. While the dynamic behavior of systems with mobile codes has been known to be hard to analyze, we devise a concise but effective cost model that captures the essential elements of the service process. The model facilitates analytical characterization of the data management overhead and service cost. By doing so, we were able to pinpoint the exact situations for the strategies to be superior than conventional approach. Simulation results suggest that proper data management strategies with mobile codes can play valuable roles in location based services to moving data sources.

## 2 Related Work

Data management in mobile environments is challenging for the need to process information on the move. Among the mobile applications, location based services have been identified as one of the most promising area[1]. The problem has also been studied under terms such as location-aware computing[8], adaptive information systems [10], and context-aware computing[4, 7]. In general, an application is context-aware if it is capable of discovering and taking advantage of contextual information. Location is among the most important context for mobile applications. Many previous works treated location as an additional attribute of the data tables[5, 11]. In this way, LBS queries can be processed like ordinary queries except with additional constraints on the location attribute. Caching techniques specially tailored for mobile environments have also been a major research area [3, 5]. Semantic caching techniques employed semantic descriptions to facilitate better cache admission and replacement decisions that are responsive to the user movement[9]. The issues related to dynamic data management in mobile environments have also been discussed in the context of replicated databases[2, 12]. The use of mobile codes or mobile agents in mobile environments are along the similar idea of exploiting code mobility for user/data mobility[6]. Our work differs from the previous works in that we provide data management strategies for both instantaneous and continuous queries, with and without mobile codes. Furthermore, we conduct detail analysis to precisely capture the execution cost and management overhead by giving each strategy proper analytical characterization and cost formula. This can help both the system administrators and users in determining when and how to apply proper strategies for different situations.

## 3 Service and Query Classification

In [13], mobile information services were classified along four dimensions based on service properties. In this paper, we characterize the services from the clients' point of view and explore the query space along two additional dimensions:

- **Target**: *objects* vs. *ranges*
  The target can be either designated objects or objects within designated location ranges.
- **Duration**: *instantaneous* vs. *continuous*
  An instantaneous query is a one-time only query while a continuous query must be processed continuously as the client or targets move.

The two dimensions are orthogonal and constitute a natural classification of the requests, namely, *instantaneous object queries*, *instantaneous range queries*, *continuous object queries*, and *continuous range queries.* In Table 1, we give several examples of the queries on each categories. More specifically, *instantaneous object queries* ask for specific objects in disregard of their locations while *instantaneous range queries* target objects within designated location ranges. Both are intended to be processed only once and answered with current status. On the other hand, *continuous object/range queries* are used to continuously monitor the status of designated objects/ranges. The queries must be continuously processed whenever the status change affect the query results. All four types of queries are frequently used in mobile environments, especially for LBS. Our goal is to provide highly flexible framework and data management strategies to facilitate effective services to all four types of requests.

## 4 System Architecture

Our system architecture is depicted in Figure 1. Both of the client and base station have caches. Re-

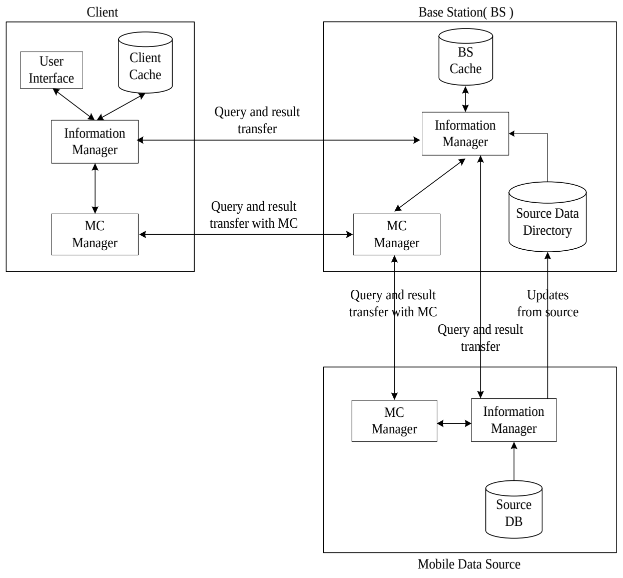

Figure 1: System architecture.

quests are first answered using the client cache or forwarded through the wireless connection to the base station on cache miss. A request can be sent by the *information manager* or delegated to mobile codes through the *MC manager*. On base station cache hit, the results are sent directly back to the client. Otherwise, the query is forwarded further to the moving data sources. Likewise, the request forwarding can be handled by the information manager or through the launching of mobile codes. Each data source has information manager to manage the *source DB* as well as handling the incoming requests. If the requests were carried in by mobile codes, the MC manager is responsible for hosting the mobile codes and passing the requests to the information manager. If the target sources are in different cells, the base station must contact other base stations for requests forwarding (or mobile codes routing). The base station keeps information about all the data sources currently reside in its cell with the *source data directory* which is also responsible for accepting updates or invalidation from the data sources. The mobile codes can be directed to travel alone with designated data sources for update monitoring and proactive pushing. The architecture is for cost modelling and strategy design and therefore not intended to be comprehensive.

## 5 Model and Strategies

The architectural design leads to an abstract cost model as depicted in Figure 2. In particular, we iden-

| | | Target of service | |
|---|---|---|---|
| | | Objects | Ranges |
| Duration of service | Instantaneous | "List the address and phone number of the NDHU University.", ... | "List the names of all restraints within 5 miles of my current location.", ... |
| | Continuous | "Continuously report the location and altitude of the Flight PF911.", ... | "Continuously report the location of any vehicles within 10 miles range.", ... |

Table 1: Query classification for mobile information services

tify three major types of entities and their relationships, namely, the clients($C$), base stations($B_i$), and moving data sources($S_i$). All base stations are connected to the fixed network and communicate with stable and high speed wired connection. Both the clients and the moving data sources communicate through the wireless connection. As the current technology stands, wireless access still costs much higher than wired connection. However, there is no essential differences between a client and a moving data source. Therefore we model the wireless connection cost to be the same. Wireless communication can be further classified into data messages and the dispatching of mobile codes. Since the later incur management overhead and system resources, sending a data message should cost no more than having a mobile code to carry the same message across the net. We therefore classify the communication cost into three categories as follows.

- $k$ The message cost between base stations over fixed network.
- $d$ The message cost between a client (or data source) and its corresponding base station.
- $M$ The cost of launching a mobile code.

In general, $k$ is much smaller than $d$ and $M$.

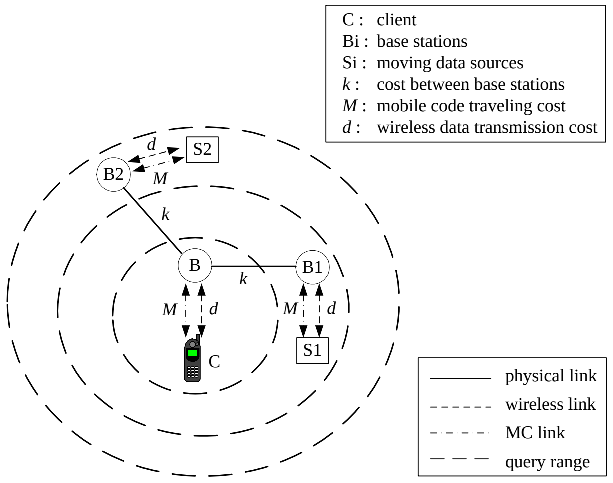

Figure 2: Cost model for location based services.

## 5.1 Instantaneous Object Queries

Instantaneous object queries are one-time only queries that target specific objects and must be answered immediately.

**Basic On-demand Strategy**

The basic on-demand strategy is the one that do not perform any optimization beyond that of on-demand request services and conventional caching. More specifically, a request is first issued to the information manager on the client device. If the request can be satisfied by the client cache, no further processing is needed. Otherwise the request must be sent through the wireless link to the base station. Similarly, if the client request can be found in the base station cache, the target objects are sent back to the client directly. When both attempts to the client and base station caches fail, the request must be forwarded to the data sources for the latest versions.

For detail analysis of the execution cost of the data management strategies, we need some additional parameters as follows to model the execution behavior.

- $R$ The total number of requests issued by the clients.
- $h$ The average hit rate of the client cache with on-demand services and conventional caching.
- $h1$ The average hit rate of the base station cache with on-demand services and conventional caching.

With the model above, we can derive the cost formula of the basic on-demand strategy as follows.

$$R \cdot h + R \cdot (1-h) \cdot h1 \cdot 2d + R \cdot (1-h) \cdot (1-h1) \cdot 4d$$

The basic on-demand strategy is used as the baseline strategy for performance comparison purpose. Note that, as we explained earlier, the communication cost over fixed network (i.e. $k$) can be safely ignored since $k$ is much smaller than $d$ and the services of instantaneous queries do not accumulate cost alone fixed links.

### On-demand with Mobile Codes

Instead of forwarding the request, a client can choose to launch a mobile code on cache miss. The mobile code starts its trip by visiting the base station (cache) first to see it contains the desired objects. If not, the trip goes on toward the data sources by going through other base stations. Whenever the target objects are located, the mobile code is also responsible for sending them back to the client. After finishing the current job, the mobile code can either terminate itself or stay around for status monitoring and update notification. While the strategy seems to be a natural alternative, it must be used with care. In general, the strategy is effective only when the cost of dispatching a mobile code is smaller than a data message cost(i.e. $M < d$). This can be derived by comparing the execution cost of on-demand mobile code and the basic on-demand strategy. The former is similar to the later except that the request forwarding is replaced by mobile code. The strategy is beneficial if the cost is smaller than the basic on-demand cost. That is,

$$\begin{aligned} & R \cdot h + R \cdot (1-h) \cdot h1 \cdot (M+d) + \\ & R \cdot (1-h) \cdot (1-h1) \cdot (2M+2d) \\ < \quad & R \cdot h + R \cdot (1-h) \cdot h1 \cdot 2d + \\ & R \cdot (1-h) \cdot (1-h1) \cdot 4d \end{aligned}$$

$$\Rightarrow M < d$$

### Proactive Pushing with Mobile Codes

For a moving data source, a natural idea is to send a mobile code to travel alone with it. We explore this idea further by having the mobile codes perform proactive pushing of hot data or updates toward the base stations. More specifically, we let the base station launch a mobile code to each data source that owns at least one cached object. The mobile code is to monitor the target objects and proactively send back certain percentage of the updates for maintaining the cache consistency. To evaluate the cost effectiveness and the exact situations for applying the strategy, we need to carefully model and analyze the update and pushing behavior as follows.

| | |
|---|---|
| $U$ | The total number of updates. |
| $w$ | The percentage of updates that are proactively pushed back to the base station. |
| $q$ | The percentage of the pushed data that are actually accessed. |

With the model above, we can derive the situation for the proactive pushing strategy to be beneficial. More specifically, for the strategy to be cost effective, the cost ratio of a mobile code w.r.t. a data request (i.e. $M/d$) can not be larger than the ratio of the original base station cache miss w.r.t the proactive pushing cache miss. To see why this is the case, we observe that, due to proactive pushing, we will have additional $U \cdot w \cdot q$ base station cache hits that save us the cost of going to the data sources. However, we also need to spend additional cost of $U \cdot w \cdot d$ in maintaining cache consistency. For the proactive pushing strategy to be effective, the combination of cost saving and consistency maintenance must be smaller than the on-demand cost. That is,

$$\begin{aligned} & R \cdot h + [R \cdot (1-h) \cdot h1 + U \cdot w \cdot q] \cdot 2d + \\ & [R \cdot (1-h) \cdot (1-h1) - U \cdot w \cdot q] \cdot (M+3d) + \\ & U \cdot w \cdot d \\ < \quad & R \cdot h + R \cdot (1-h) \cdot h1 \cdot 2d + \\ & R \cdot (1-h) \cdot (1-h1) \cdot 4d \end{aligned}$$

$$\Rightarrow \frac{M}{d} < \frac{R \cdot (1-h) \cdot (1-h1) - U \cdot w \cdot (1-q)}{R \cdot (1-h) \cdot (1-h1) - U \cdot w \cdot q}$$

For the largest possible $M$, i.e. when $q = 1$, the equation above can be simplified to

$$\frac{M}{d} < \frac{R \cdot (1-h) \cdot (1-h1)}{R \cdot (1-h) \cdot (1-h1) - U \cdot w}$$

which is exactly the case stated earlier.

## 5.2 Continuous Object Queries

Mobile codes show their real strength when it comes to the services of continuous queries. The biggest challenge here is that both the client and the data sources can move. They can be in the same or different cells during the course of continuous query execution. When residing in the same cell, the base station can continuously retrieve the states of the sources and send them to the requesting client. Otherwise, the job must be delegated to the base stations hosting the target sources. We note that since the object states change continuously, caching offers little help.

### Basic Processing Strategy

For comparison purpose, we present a basic processing strategy where a continuous query is processed as repeated sending of instantaneous queries. Since a client does not know when and how often will the data sources be updated, it can not determine the best timing and frequency of requests. To compensate the lack of optimization, we assume that the client only needs to send the same number of requests as the number of updates. Under this assumption, we can derive the cost formula for the repeated processing strategy with some additional statistics.

- $p$ The number of updates which is, as described above, also the number of requests.
- $\alpha$ The percentage of time when the client and the desired data sources reside at the same cell.

The basic repeated processing cost can be characterized as the cost of accessing from within the same cell and the out of cell access. Note that the cost on the fixed links (i.e. $k$) can no longer be ignored here since it can accumulate alone with repeated requests.

$$C_b = p \cdot \alpha \cdot 4d + p \cdot (1-\alpha) \cdot (2k + 4d)$$

### Continuous Object Queries with Mobile Codes

In stead of repeatedly sending the same request, the client can send a mobile code to travel alone with the data source and continuously send back the new values of the data objects. This approach is especially attractive for continuous queries since almost all the repeated requests can be saved. In the following analysis, we demonstrate the superiority of the mobile code strategy over the repeated processing strategy.

The execution cost of the mobile code strategy includes the initial cost of launching the mobile code and the cost of continuously sending back the newly updated object. The strategy is beneficial if the cost is smaller than $C_b$. That is,

$$\begin{array}{rl} & 2M + p \cdot \alpha \cdot 2d + p \cdot (1-\alpha) \cdot (k + 2d) \\ < & p \cdot \alpha \cdot 4d + p \cdot (1-\alpha) \cdot (2k + 4d) \end{array}$$

$$\Rightarrow M < \frac{1}{2}(p \cdot \alpha \cdot 2d + p \cdot (1-\alpha) \cdot (k + 2d)) = \frac{1}{4}C_b$$

In other words, the mobile code strategy is effective if the cost of launching the mobile code is less than 1/4 of the basic repeated processing cost. As long as the query is issued long enough, $C_b$ is much larger than $M$. We therefore conclude that the mobile code strategy is almost always superior than the repeated processing strategy for continuous object queries.

## 5.3 Instantaneous Range Queries

The processing of location dependent range queries differs significantly from that of object queries. In particular, all objects reside within a user-specified range and satisfy the query constraints must be returned to the client. Some of the target objects may happen to be at the same cell as the client. Others may be in different cells. For moving targets residing at the same cell, we can access them by going through the base station. Otherwise we need to ask for help from the neighboring base stations. Both for supporting data management and behavior analysis, we need to maintain some access statistics.

- $n$ The number of data sources that satisfy the location dependent range queries.
- $\alpha$ The percentage of the desired data sources residing at the same cell with the client.

### Basic Processing Strategy

The simplest way to process an instantaneous range query is to first determine the data sources satisfying the query and then send a request to each of them. Both the client cache and the base station cache are still useful for instantaneous range queries. We only need to send request to the data sources not in the caches. Therefore the cost formula of the basic strategy can be derived as follows.

$$\begin{array}{rcl} C_r & = & R \cdot h + R \cdot (1-h) \cdot h1 \cdot n \cdot 2d + \\ & & R \cdot (1-h) \cdot (1-h1) \cdot n \cdot \alpha \cdot 4d + \\ & & R \cdot (1-h) \cdot (1-h1) \cdot n \cdot (1-\alpha) \cdot (4d + 2k) \end{array}$$

### Range Queries with Mobile Codes

Mobile codes are useful for range query processing as well. The basic idea is actually quite natural. Instead of sending a location dependent range query to the base station, the client launches a mobile code to do the job. The mobile code is responsible for coordinating with the base station in launching additional codes to visit the data sources and to collect all the result messages sent back.

The execution cost analysis of the mobile code strategy can be divided into four parts: on client cache hit, on base station cache hit, on cache miss but co-located data sources, and on cache miss and out of the cell data sources. For the mobile code strategy to be effective, the sum of the four parts must be smaller than $C_r$. That is,

$$\begin{array}{rl} & R \cdot h + R \cdot (1-h) \cdot (M + n \cdot d) + \\ & R \cdot (1-h) \cdot (1-h1) \cdot n \cdot \alpha \cdot (M + d) + \\ & R \cdot (1-h) \cdot (1-h1) \cdot n \cdot (1-\alpha) \cdot (M + d + 2k) \\ < & C_r \end{array}$$

$$\Rightarrow M < \frac{(2-h1)n}{(1-h1)n+1} \cdot d$$

In the worst case, the base station hit rate $h1$ drops down to 0, the equation simplifies to

$$M < \frac{2n}{n+1} \cdot d \approx 2d$$

We conclude that, the mobile code strategy is effective as long as the cost of launching a mobile code is smaller than twice the cost of data message. In the best case (i.e. $h1 = 1$), the equation simplifies to

$$M < n \cdot d$$

which gives more room for the mobile code to spare.

**Range Queries with Proactive Mobile Codes**

A possible improvement over the previous mobile code strategy is to delegate the mobile codes not only the data retrieval job but also the job of proactively sending new updates toward the base station. This can help improve the hit rate of the base station cache. Similar to the case for object query processing, we also need to determine the push rate and maintain other parameters for cost analysis.

| | |
|---|---|
| $U$ | The average number of updates of a single source. |
| $w$ | The percentage of updates that are proactively pushed back to the base station. |
| $q$ | The percentage of the pushed data that are actually accessed. |

We can then identify the situation for applying the strategy by comparing the execution cost with $C_r$. The cost analysis needs to consider both the saving due to increased base station hit rate as well as the pushing overhead. The increased hit rate can also be modelled as the reduction in miss rate due to proactive pushing, that is

$$B_m = R \cdot (1-h) \cdot (1-h1) - U \cdot w \cdot q$$

This leads to the cost comparison as follows.

$$\begin{array}{ll} & R \cdot h + R \cdot (1-h) \cdot (M + n \cdot d) + \\ & B_m \cdot n \cdot \alpha \cdot (M + d) + \\ & B_m \cdot n \cdot (1-\alpha) \cdot (M + d + 2k) + \\ & n \cdot \alpha \cdot U \cdot w \cdot d + n \cdot (1-\alpha) \cdot U \cdot w \cdot (d+k) \\ < & C_r \end{array}$$

In the worst case, i.e. when all $h1$, $q$, and $\alpha$ are 0, the equation above can be simplified into

$$M < (2 - \frac{U \cdot w}{R(1-h)})d - \frac{U \cdot w}{R(1-h)}k$$

which means if the proactively pushed data failed to increase the base station hit rate, the proactive mobile code strategy may not provide performance advantage unless $M$ is very small. We note that since $\frac{U \cdot w}{R(1-h)}$ is the actual update/request rate seen by the base station, we can easily determine when to apply the strategy.

On the other hand, if every thing goes well, i.e. when all $h1$, $q$, and $\alpha$ are 1, we obtain the simplified equation as

$$M < \frac{n}{1 - \frac{U \cdot w}{R(1-h)}n}d$$

which leaves more room for the mobile code strategy.

## 5.4 Continuous Range Queries

The processing of continuous range queries is similar to that of continuous object queries except that there may exist many target sources that satisfy the location based range constraints. Mobile codes are particularly useful in this case again since they can be instructed to say with the sources and continuously sending back new values of the data objects.

**Basic Strategy**

Similar to the processing of continuous object queries, we assume a basic strategy that repeatedly sending instantaneous requests in a rate as often as the update rate. Note that since each request results from a source update, caching offers little help in here. Assume that $p$ represents the number of updates which is, as described above, also the number of requests, then the execution cost of the basic strategy can be modelled as the sum of the cost of accessing sources located at the same cell and that of accessing data sources not in the same cell.

$$p \cdot n \cdot \alpha \cdot 4d + p \cdot n \cdot (1-\alpha) \cdot (4d + 2k)$$

**Continuous Range Queries with Mobile Codes**

Mobile codes can be dispatched by the base station in advanced or on the first request to all related data sources and stay with them. The newly updated object values can then be sent back continuously by the mobile codes. We note that once the mobile codes are in places, no more repeated requests need to be issued again since the agents will keep sending back the latest object values. The primary cost is therefore on the cost of sending mobile codes and the cost of continuously sending back update values.

$$\begin{array}{l} (M + d) + n \cdot \alpha \cdot M + n \cdot (1-\alpha) \cdot (M + k) + \\ p \cdot n \cdot \alpha \cdot d + p \cdot n \cdot (1-\alpha) \cdot (k + d) \end{array}$$

Similar to the analysis in previous sections, we compare the two cost formula and simplify the result to obtain the criteria for applying the strategy.

$$\begin{array}{lll} M & < & \frac{(3pn-1)d + n(1-\alpha)(p-1)k}{1+n} \\ & \approx & 3pd + (1-\alpha)pk \end{array}$$

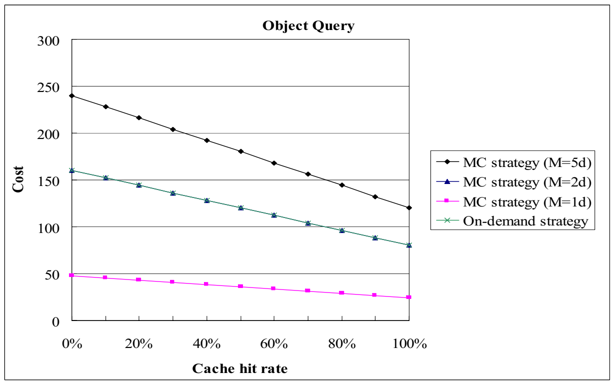

Figure 3: Instantaneous object querying cost.

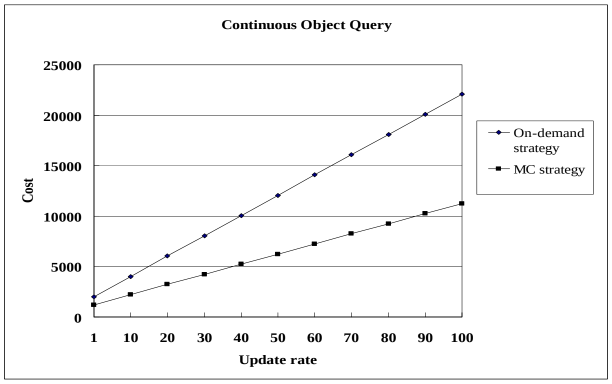

Figure 5: Continuous object querying cost.

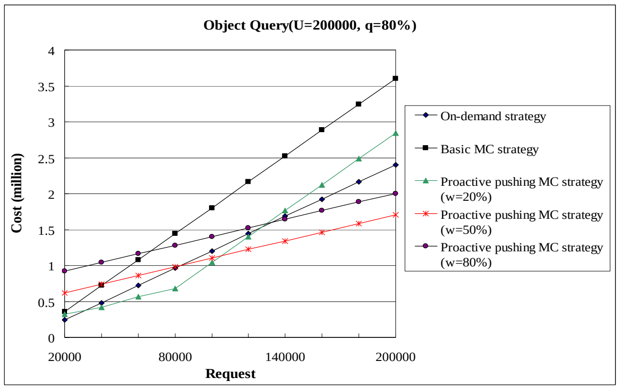

Figure 4: Instantaneous object querying with proactive pushing.

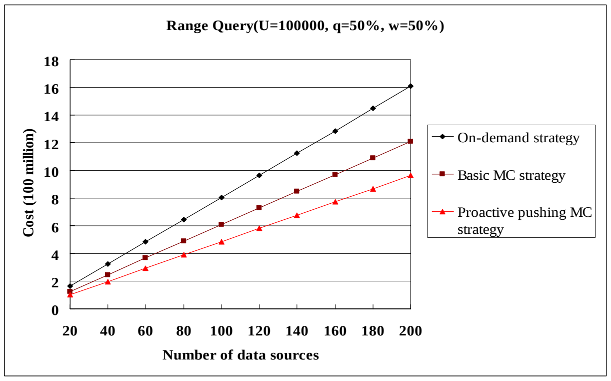

Figure 6: Instantaneous range querying cost.

From the analysis above we can conclude that the mobile code strategy is almost always preferable than basic strategy on continuous range queries processing.

## 6 Simulation and Evaluation

To evaluate the effectiveness of our strategies against on-demand strategies, we have conducted several sets of simulation with the Powersim Studio.

For instantaneous object queries, we can observe from Figure 3 that unless the cost of a mobile code can be kept lower than the cost of a dada message, simply sending mobile codes does not offer any performance advantages. On the other hand, Figure 4 demonstrates the advantage of proactive pushing over conventional on-demand and basic mobile code strategies especially when the request volume is high. We can also observe that higher push rate does not necessarily implies higher performance. This can happen when the pushing overhead outweighs the cost saving.

For continuous object queries, Figure 5 shows a large performance gap between the mobile code and on-demand strategies, especially when the update rate gets higher. We note that the performance advantage is independent of the cost of a mobile code since it needs to be sent only once.

For instantaneous range queries, even the basic mobile code strategy shows its performance benefit as demonstrated in Figure 6. The cost advantage is even greater with proactive pushing under just moderate push rate and utilization(say 50%). The larger the number of target sources, the better results we get.

Finally, Figure 7 demonstrates the superiority of mobile code strategy over on-demand strategy on continuous range queries. The performance advantage improves with higher update rate as well as larger number of data sources satisfying the range constraints. This shows the flexibility of mobile codes in travelling with the data sources and continuously sending back new values of data objects.

As a summary, except for instantaneous object queries, mobile codes with proper data management are powerful strategies for location based services in

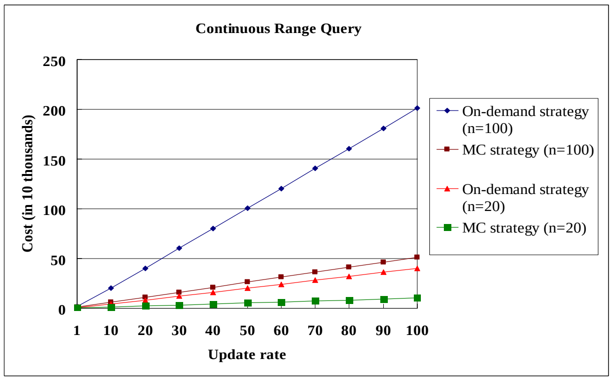

Figure 7: Continuous range querying cost.

mobile environments. Even for location independent data, carefully designed access methods are still required since the clients can move from one place to another. By characterizing the data sources and access patterns, we have designed effective strategies that successfully balanced between the saving in access cost and the increase in maintenance cost.

## 7 Conclusions

We have developed concise but effective cost models and data management strategies with mobile codes for location based services. Analytical and simulation results demonstrate that mobile code can be quite valuable in mobile environments. We have implemented a prototype using the Concordia mobile agent platform for dispatching the mobile codes. We plan to further extend our system framework and cost model to handle information services on streaming data. We are also evaluating the potential of employing distributed cooperation and P2P techniques to support continuous location based service queries.

## References

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

[2] D. Barbara and H. Garcia-Molina. Replicated data management in mobile environments: Anything new under the sun? In *Applications in Parallel and Distributed Computing*, pages 237–246, 1994.

[3] 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.

[4] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[5] 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.

[6] R. S. Gray, D. Kotz, S. Nog, D. Rus, and G. Cybenko. Mobile agents for mobile computing. In *Proceedings of the Second Aizu International Symposium on Parallel Algorithms/Architectures Synthesis*, Fukushima, Japan, March 1997.

[7] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2-3):187–197, 2002.

[8] C.A. Patterson, R.R. Muntz, and C.M. Pancake. Challenges in location-aware computing. *IEEE Pervasive Computing*, 2(2):80–89, April-June 2003.

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

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

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

[12] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM Trans on Database Systems*, 22(2):255–314, June 1997.

[13] Shiow-yang Wu and Kun-Ta Wu. Dynamic data management for location based services in mobile environments. In *IDEAS2003: The 7th International Database Engineering and Application Symposium, Hong Kong, July 16-18*, 2003.