5

Caching Data Over a Broadcast Channel

H. V. Leong A. Si B. Y. L. Chan
Department of Computing,
The Hong Kong Polytechnic University,
Hung Hom, Hong Kong.
Telephone: (852)-2766-7243.
Fax: (852)-2774-0842.
Email: {cshleong,csasi,csylchan}@comp.polyu.edu.hk

Abstract

We consider an environment in which a collection of mobile clients interacts with a stationary database server. Due to the low bandwidth of wireless communication channels, it is necessary to broadcast highly popular data items and deliver other data items on a demand basis to the mobile clients. This broadcast wireless media can be considered as an extra layer of cache storage, which we term air-storage. We investigate several mechanisms in selecting the data items to be placed over this new layer of air-storage, and illustrate their effectiveness by means of simulation.

Keywords

Mobile database, data broadcasting, data caching

1 INTRODUCTION

In a mobile computing environment, users carrying portable computers, referred to as mobile clients, will be able to access information via wireless channels to a stationary database server (Imielinski and Badrinath, 1994). In such a mobile environment, the broadcast paradigm is effective for disseminating database items from the server to multiple clients since the downstream communication bandwidth from the server to the clients is usually much larger than the upstream communication bandwidth from the clients back to the server (Acharya *et al.*, 1995). Each client then, picks its own items of interest selectively from the broadcast database (Imielinski *et al.*, 1994).

In general, the performance of a query over the broadcast database will largely depend on the length of a broadcast cycle (Imielinski *et al.*, 1994; Leong and Si, 1995). which indicates the amount of time required to broadcast one copy of all database items. Figure 1 illustrates this idea on a database with five relations.

In Figure 1, the average response time for 100 selection queries against a broadcast

J. L. Encarnação et al. (eds.), *Mobile Communications*

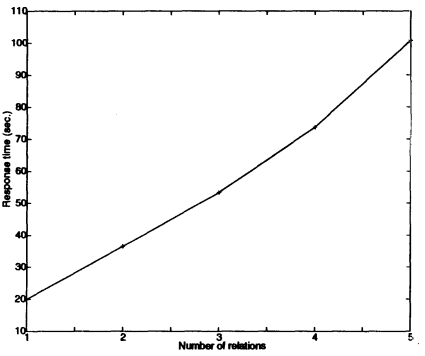

Figure 1 Response time *versus* number of relations.

database over a wireless channel at a bandwidth of 19.2 Kbps is measured. A broadcast cycle is composed of 1 to 5 relations. Each relation contains 640 tuples with each tuple occupying 128 bytes. The access probabilities of the five relations are 0.5, 0.2, 0.15, 0.1, and 0.05 respectively. As shown in Figure 1, the response time of a query increases if all database items are broadcast uniformly regardless of their access frequencies.

To optimize database queries against broadcast database, the server should only broadcast database items that would be frequently accessed by a lot of mobile clients. These database items are usually called "hot items". "Cold items" which are accessed much less frequently should be delivered on an "as needed" basis. In this paper, we propose schemes to identify the database items suitable to be broadcast over the wireless channel. We conduct simulated experiments to study the effect of various proposed schemes on the availability of a requested database item over the broadcast channel. This paper is organized as follows. In Section 2, we offer a brief description on our model and propose several schemes in identifying those hot candidates to be broadcast. Section 3 describes our experiments and illustrates some preliminary results, along with explanations to the results. We conclude this paper with a brief discussion on our current research directions.

2 ENERGY EFFICIENT CACHING SCHEMES

When a database is periodically broadcast over a wireless channel, the channel in effect acts as a storage layer which we term **air-storage**. The database server can thus be regarded as a tertiary storage and the broadcast database items can be considered to be cached into this air-storage. The size of this air-storage is characterized by the length of a broadcast cycle as it indicates the amount of database items that could be accommodated in the air-storage. This forms a hierarchy of storage systems with increasing bandwidths: database items frequently requested by most clients are cached over the air-storage with a low bandwidth; database items frequently requested by a particular mobile client could be cached into its local disk storage with a much higher bandwidth (Barbara and Imielinski, 1994); and finally, currently queried database items will be cached into a client's main memory buffer with the highest bandwidth.

By treating the broadcast channel as an air-storage, its management becomes similar in spirit to the management of cache memory. In particular, strategies to determine which database items should be broadcast over the channel or refrain from being broadcast will be similar to the cache replacement strategies. However, due to the different natures of the air-storage and conventional cache memory, four issues need to be readdressed.

First, the size of the air-storage, i.e., the length of a broadcast cycle, could be dynamically changed. This is in contrast to conventional cache memory whose size is static. Obviously, increasing the size of the air-storage might penalize the performance of certain data retrievals, but it might benefit the overall performance for the whole collection of mobile clients globally. Critical here is the mechanism to determine the size of the air-storage for each broadcast while striking a balance between local and global optimizations. Owing to space limitation, we are not able to discuss this issue further here.

Second, database items stored in conventional cache memory are accessed in a random manner; by contrast, database items stored over the air-storage could only be accessed sequentially. Techniques tailored to the sequential nature of the air-storage for retrieving the useful items are in demand and have been addressed to some extent in Acharya *et al.* (1995), Imielinski *et al.* (1994), and Leong and Si (1995).

Third, conventional caching mechanisms are usually performed on a per-page basis due to the principle of locality that the page containing the database items currently being accessed will have a high probability of being accessed in the near future. Since the bandwidth of the air-storage is much lower than that of conventional cache memory, caching the whole page of data from the server into the air-storage will be too expensive. A smaller granularity for caching is needed. In our research, we are experimenting in using an entity as a caching unit for the air-storage, i.e., the database is broadcast on an entity basis. Each entity will roughly correspond to an object in an object-oriented database and will correspond to a tuple in a relational database.

Finally, with the size of the air-storage being fixed for a particular broadcast, we need to identify the hot items (entities) suitable to be cached into the air-storage, i.e., being broadcast. During subsequent broadcast cycles, additional entities might be qualified as hot. Since hot items are those accessed by many clients, the access pattern of each client on the entities must be propagated from the clients where the accesses are originated, back to the server where the statistics will be compiled and caching strategy is being implemented. If the air-storage is exhausted, replacement strategy is needed to replace aged cold database items being broadcast currently with new hot items. While caching is performed on an entity basis, the spatial locality information among the different cached items in database applications is lost. This is amplified by the time lag until the server is aware of the access patterns on database items issued by the clients. This in turn implies that conventional replacement strategies that perform good (such as least recently used or LRU) (Korth and Silberschatz, 1991) are no longer suitable in the new context since they are based on the principle of locality. New cache replacement strategies need to be introduced.

Without the locality information among the cached entities, it is natural to adopt the access probabilities of entities as an indicator for the necessity of being cached into or being replaced from the air-storage. The access probabilities could be collected at each mobile client and forwarded to the server at regular intervals. They may also be piggybacked in a request to the server, when there is an air-storage miss, i.e., the requested database entity

is not being broadcast in the current broadcast cycle. We have experimented with several ways of estimating the access probabilities and their effectiveness in cache replacement.

The simplest way to keep track of the access probability for each entity is by measuring the cumulative access frequency. We call this the **mean** scheme. If the air-storage is not exhausted, an entity is cached if its cumulative access frequency is higher than a certain threshold τ. However, if the air-storage is exhausted, an entity is cached only if its accumulated access frequency is higher than the lowest access frequency of a cached entity. In other words, with an air-storage of size c, only those c items with the highest access frequencies are broadcast. Alternatively, a window for the access frequencies spanning several broadcast cycles could be maintained for each entity. The cached entity with the lowest average access frequency within the window is replaced. We call this scheme the **window** scheme. The effectiveness of the window scheme depends on the window size W. A problem for the window scheme is the amount of storage needed in maintaining the windows. To avoid the use of a moving window and to adapt quickly to changes in access patterns, our third scheme measures the exponentially weighted moving average of access frequencies. Access frequencies of current broadcast cycle have higher weights and the weights tail off as they become aged. This is called the **exponentially weighted moving average (EWMA)** scheme. A parameter to EWMA is the weight α.

The changing metrics in the schemes above can be computed incrementally. Assume that a new access frequency $f_{x,n+1}$ from time interval n to $n+1$ is collected for object x and the metric for the previous n access frequencies is $\overline{f}_{x,n}$. Here a typical time interval for the access frequencies is the broadcast cycle. At the end of a broadcast cycle, the server accumulates the required information used to compute the metrics at the beginning of a new broadcast cycle which in turn acts as a guideline in selecting the database items to be broadcast. In the mean scheme, the new metric $\overline{f}_{x,n+1}$ can be computed as $(n\overline{f}_{x,n} + f_{x,n+1})/(n+1)$. For the window scheme with a window size W_x for object x, when the window is not full (i.e., $n < W_x$), computing the new metric when adding a new access frequency is the same as that for the mean scheme. Otherwise, the new metric $\overline{f}^{(W_x)}_{x,n+1}$ is computed as $(W_x\overline{f}^{(W_x)}_{x,n} + f_{x,n+1} - f_{x,n-W_x+1})/W_x$. A total of W_x intermediate values need to be stored for object x. Finally, in EWMA with weight α_x, the weighted moving average $\overline{f}^{(\alpha_x)}_{x,n+1}$ is computed as $\alpha_x\overline{f}^{(\alpha_x)}_{x,n} + (1-\alpha_x)f_{x,n+1}$. No intermediate value needs to be maintained.

It can be observed that the window scheme reduces to the mean scheme if an infinite window size W is used. When $\alpha = 0$, the EWMA scheme becomes identical to the window scheme with $W = 1$; if α tends towards 1, older values have larger weights and the scheme becomes similar to the mean scheme.

3 SIMULATION RESULTS

To illustrate the relative effectiveness of the three cache replacement schemes, namely, mean, window, and EWMA, we have conducted a simulation study. In this simulation, we assume that there are 1000 entities in the database server and the air-storage has a capacity to broadcast 300 entities. The size of each entity is set to 128 bytes. The experiments are also repeated with an entity size of 16 bytes. Two different access patterns on data entities are experimented with. The first experiment assumes that the "temperature"

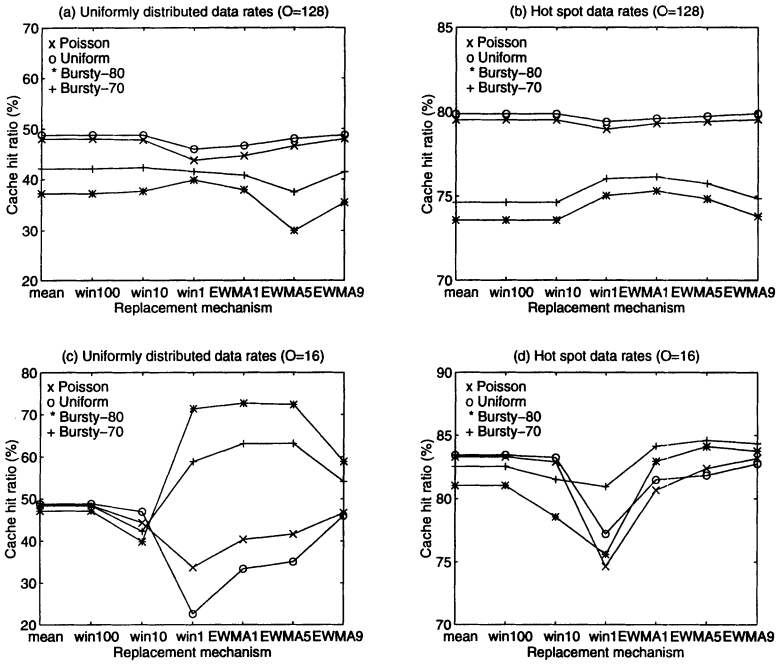

Figure 2 Effectiveness of cache replacement strategies.

of data entities are uniformly distributed, in that the access probabilities of the entities follow a uniform distribution. The second experiment is more realistic in reflecting the data affinity of data accesses. About 20% of the data entities have a very high access rate, constituting 80% of the operations; the remaining 80% of the entities have a much lower 20% access rate. This models a situation in which the access patterns exhibit some locality or skewed behavior (Franklin *et al.*, 1992). We investigate different arrival patterns of the operations for accessing the data entities including the standard Poisson arrival following the exponential distribution, the simple uniformly distributed arrival events, and the bursty arrival showing some form of temporal locality. We experiment with two bursty arrival patterns: Bursty-80 and Bursty-70, meaning that 80% (and respectively 70%) of the operations arrive uniformly in 20% (and respectively 30%) of the time span and the remaining 20% (and respectively 30%) arrive uniformly in another 80% (and respectively 70%) of time span. A total of 100 000 read accesses are generated in each experiment. Each experiment is repeated 30 times, and the average over the 30 experiments is taken to be a data point.

As an indicator for the effectiveness of a replacement strategy, we measure the cache (air-storage) hit ratio which is the number of hits by the client on the air-storage divided by 100 000 operations, i.e., the number of times that a requested entity could be found on the broadcast channel. The average cache hit ratios of mean scheme, window schemes with $W = 100$ (**win100**), $W = 10$ (**win10**), $W = 1$ (**win1**), and EWMA schemes with

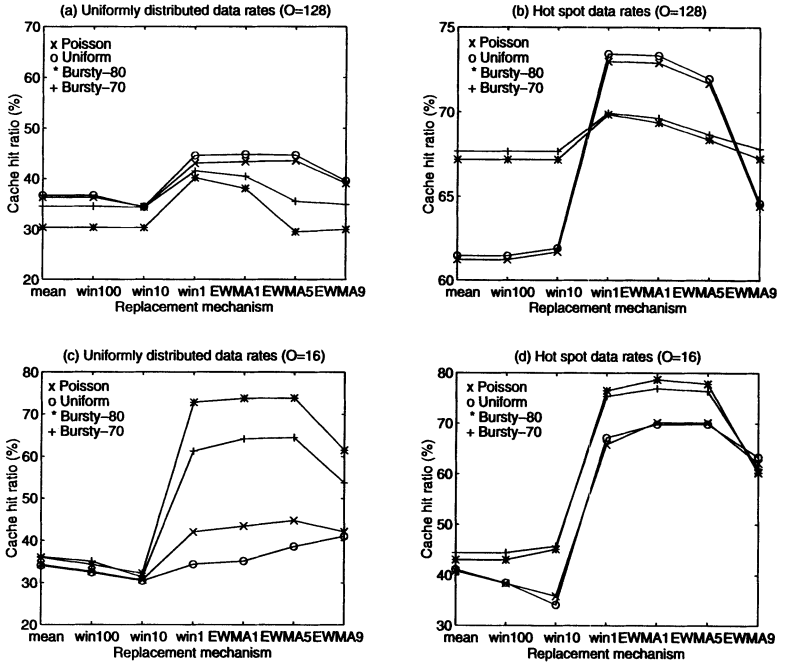

Figure 3 Impact of changes in access patterns.

$\alpha = 0.1$ (EWMA1), $\alpha = 0.5$ (EWMA5), and $\alpha = 0.9$ (EWMA9) over 30 repeated runs are measured. The cache hit ratios of the various replacement schemes for the uniform and skewed access patterns when each entity has a size of 128 bytes (O=128) are depicted in Figures 2*a* and 2*b* respectively. The corresponding hit ratios when the entity size is 16 bytes (O=16) are shown in Figures 2*c* and 2*d*.

It is obvious in Figure 2 that the performances of all schemes are better when there are hot spots in the database items. All the proposed schemes are capable of using access information to identify the hot entities and to broadcast them over the air-storage. Furthermore, we can observe that the performances of the window-based schemes are not stable and that the mean scheme performs relatively well. EWMA also seems to perform good, especially when the entity size becomes smaller (16 bytes). A smaller object size results in a shorter broadcast cycle such that changes in access patterns can be incorporated into the scheme in a more timely fashion and be reflected in the next broadcast cycle. When compared with using EWMA for local caching which exhibits a stable performance (Leong and Si, 1996), EWMA in the management of air-storage does show some fluctuations in performance. This is mainly due to the lag-behind of the changes on broadcast database items to cope with the changes on access patterns.

EWMA schemes usually perform better than the mean scheme in reality, when we consider that access patterns may change over time. To demonstrate this feature, we conducted another set of experiments. This time, however, the data access rates to each

database entity are randomly permuted after every five broadcast cycles; this has an effect of moving the hot spot randomly every few cycles. The result is depicted in Figure 3, in one-to-one correspondence with those in Figure 2. We observe that all hit ratios drop since the server now is not able to cope with the changes in access patterns very well. The mean scheme, in particular, suffers seriously from this change and performs very poorly, since the changes in patterns cannot be reflected rapidly in a change of the cumulative access frequencies. The window scheme with large window sizes also suffers from the same phenomenon. The EWMA scheme with a small or moderate value of α reacts very well to this change. This is due to the much stronger adaptive nature of EWMA than the mean scheme. One might argue that window scheme with $W = 1$ (**win1**) has the best performance and should always be used. It is not surprising that **win1** performs best in this case since it is able to completely forget about all past history, except the information of the previous cycle. Any change in the access patterns will instantaneously be reflected in the next broadcast cycle. However, its performance is highly unstable in most cases, by forgetting virtually everything. EWMA therefore appears to be a valuable compromise, with a good and yet relatively stable performance in all cases. Other statistical operators may also be used, to exploit the temporal and spatial information about the database entities accessed.

4 DISCUSSION

In this paper, we show that the problem of determining if a database item needs to be broadcast is similar to the cache management problem and propose a solution based on cache replacement strategy to the problem. Several new replacement strategies have been proposed and preliminary experimental results have also demonstrated that our proposed strategies are promising. We are currently conducting experiments to measure the effectiveness of the different replacement strategies under different conditions such as in an environment containing a group of mobile clients, each showing a different access pattern.

Another issue that we are currently working on is to characterize the factors that determine the length of a broadcast cycle, i.e., the size of the air-storage. This issue is very important as that affects how many database items could be broadcast over a cycle and hence the response time. We believe that in a typical mobile environment, it is best to employ a hybrid broadcast and on-demand communication paradigms so that the hot items will be broadcast while the cold items will be disseminated on-demand. The size of the air-storage will thus depend on the relative ratio of the response time between on-demand and broadcast queries. For instance, if the average response time for an on-demand query is t, the response time for queries against broadcast database should not exceed t or clients might as well access the data on demand. This will in turn create additional traffics over the on-demand channel and drive up the average response time t.

ACKNOWLEDGEMENT

This research is supported in part by the Hong Kong Polytechnic University Central Grant numbers 353/029 and 350/570.

REFERENCES

S. Acharya, R. Alonso, M. Franklin, and S. Zdonik (1995). Broadcast Disks: Data Management for Asymmetric Communication Environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 199–210.

D. Barbara and T. Imielinski (1994). Sleepers and Workaholics: Caching Strategies in Mobile Environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1–12.

M. Franklin, M. Carey, and M. Livny (1992). Global Memory Management in Client-Server DBMS Architectures. In *Proceedings of International Conference on Very Large Databases*, pages 596–609.

T. Imielinski and B. Badrinath (1994). Mobile Wireless Computing: Challenges in Data Management. *Communications of the ACM*, 37(10):18–28.

T. Imielinski, S. Viswanathan, and B. Badrinath (1994). Power Efficient Filtering of Data on Air. In *Proceedings of EDBT*, pages 245–58.

H. F. Korth and A. Silberschatz (1991). *Database System Concepts.* McGraw-Hill.

H. V. Leong and A. Si (1995). Data Broadcasting Strategies over Multiple Unreliable Wireless Channels. In *Proceedings of ACM International Conference on Information and Knowledge Management*, pages 96–104.

H. V. Leong and A. Si (1996). Semantic Caching in a Mobile Environment: Model and Evaluation. Submitted for publication.

BIOGRAPHICAL NOTES

H. V. Leong was graduated from the University of California at Santa Barbara, and is currently an assistant professor at the Hong Kong Polytechnic University. He received his undergraduate and first graduate degree from the Chinese University of Hong Kong in Computer Science. Since then, he had been working as teaching and research assistants throughout his graduate career and held several scholarships and fellowships, and has published over a dozen of research papers. His research interests are in distributed systems, distributed databases and mobile computing.

A. Si received his PhD. degree in Computer Science from University of Southern California. He is currently an assistant professor of the Department of Computing at the Hong Kong Polytechnic University. He has served as external reviewer for a number of international conferences and journals, and as program committee member for a number of international conferences. His research interests include heterogeneous and federated databases, mobile computing, distributed and parallel databases. He is a member of ACM, IEEE Computer Society, and Usenix Association.

B. Y. L. Chan is a graduate student at the Hong Kong Polytechnic University. He received his undergraduate degree from the University, before committing himself to the research degree. His current research focuses on distributed systems and mobile computing.