Broadcast Disks: Scalable solution for an asymmetric environment

Byungho Lee
Computer Science & Engineering, University of Texas at Arlington
blee@cse.ute.edu

Abstract

As mobile computing gains more popularity, the challenge of providing mobile clients with ubiquitous access to data must be met. One of the main challenges is overcoming the asymmetric environment that exists between servers and mobile clients. Broadcast disks use a push-based technology to overcome such challenge in a scalable way by creation of an arbitrarily fine-grained memory hierarchy on the broadcast medium. We will take a look at basic broadcast disks concept and go over different research issues that needs to be addressed in improving broadcast disk technology.

1. Introduction

In the mobile computer environment, the communication channel between the client and the server is asymmetric [1][2]. It is asymmetric in several ways. The downstream bandwidth allocated to from server to client is much greater than upstream bandwidth from client to server. The data flow from server to client is also much greater than from client to server. And the number of clients is much greater than the number of servers. All these factors make the mobile communication channel asymmetric.

There are also other things to consider in the mobile environment. Mobile users may be disconnected from the server due to large cost or unavailability of uplink connection. They also have limited battery power contributing to voluntary disconnections. Mobile devices may have limited memory to cache or store all the data users may need.

Broadcast disks use data broadcast in order to exploit the server's advantage in communication capacity in asymmetric environments. Key advantage of broadcast delivery is its scalability; it is independent of the number of clients the system is serving [3]. It also saves battery consumption by avoiding sending data from client to server, which is costly compared to receiving data. In contrast to a pull-based protocol where all the users require dedicated bandwidth for communication, broadcast frees up more bandwidth by serving multiple users with a single broadcast channel.

This paper is organized as follows. We'll take a look at a general broadcast disks concept in section 2. In section 3, we'll touch on the different issues is research that have been done related to broadcast disks. To go in depth in section 4, multi channel broadcast scheduling which has been proposed recently is studied. Discussion and Conclusion is followed in section 5.

2. Basic Broadcast Disks Concept

The broadcast disk is created by assigning data items to different "disks" of varying sizes and speeds, and then multiplexing the disks on the broadcast channel. Items stored on faster disks are broadcast more often than items on slower disks. This approach creates a memory hierarchy in which data on the fast disks are closer to the clients than data on slower disks. The number of disks, their sizes, and relative speeds can be adjusted, in order to more closely match the broadcast with the desired access probabilities at the clients.

The original work on broadcast disks assumes a restricted broadcast environment. The restrictions include; 1) no change in client population and their access patterns, 2) data is read only, 3) no prefetching on client side, and 4) no feedback channel from the client to the server.

It is easy to think of two simple broadcast scheduling schemes. Let's suppose there is a database of pages A, B and C. One is a flat broadcast which simply takes the union of the requests and does a cyclic broadcast of the resulting set of data items. Such a broadcast is would be in the order of A-B-C, depicted in Figure 1.(a). Another scheme is a skewed broadcast where subsequent broadcast of more important data is clustered together, depicted in Figure 1.(b). With the flat broadcast, expected wait for an item on the broadcast is the same for all items regardless of their relative importance to the clients. For a skewed broadcast, expected waiting time for data with less importance is much higher than that of an important data. Figure 1.(c) depicts a multi-disk broadcast where page A was stored on a single-page "disk" that is spinning twice as fast as the second two-page "disk" containing pages B and C.

Flat disk has the best expected performance for uniform page access probabilities. However, as the access probabilities become increasingly skewed, the non-flat program performs increasingly better [2]. For a skewed access probability, multi-disk program always out performs skewed program. Multi-disk broadcast has no difference in variance for inter-arrival time for each page. If data requests are exponentially distributed, then lower inter-arrival variance results in lower expected waiting times.

Fundamental constraint of the broadcast disk paradigm would be that increasing the broadcast rate of one item must necessarily decrease the broadcast rate of one or more other items. Because broadcast channel is shared, some clients receive unwanted data before accessing desired data.

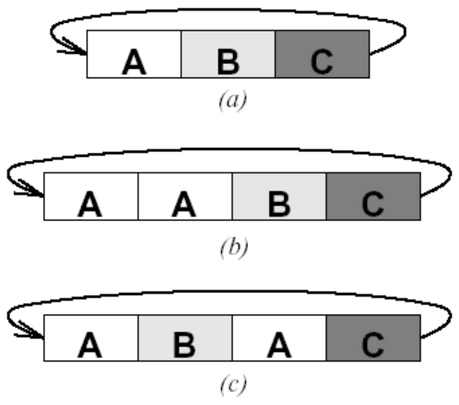

Figure 1: Three Example Broadcast Programs

3. Research Issues and Overview

3.1 A General View

Mainly broadcast on a mobile system has some restrictions compared to broadcast on a wired system. However there are lots more common ground that general broadcasting algorithms without mobile environmental restrictions also need to be looked at closely.

In a more general sense, broadcast disks scheme is just some efficient way of delivering data using air as a broadcast medium. So the problem broadcast disks address is really finding an optimal broadcast scheduling policy. With server-side scheduling, caching at the client-side is also needed to make more optimal use of broadcast disks.

There are two main research groups involved in broadcast disks. The networking research community mainly focuses on minimizing average expected delay time and tuning time for broadcast data through dynamic scheduling. Tuning time is the amount of time spent in actively listening to the broadcast channel; it has a direct implication on the energy (battery) consumption. Average expected delay time is the amount of time client must wait for an average request. Typically, at each slotted time unit, each item is given a priority based on its rate of explicit requests and last transmission time. The item with the highest priority is transmitted.

Research in the database community on broadcast tends to focus on data integrity, client-side cache management, and indexing, rather than average expected delay performance per se. For instance, version information can be periodically transmitted to guarantee mutual consistency among items. Such works rely on predictable cyclic data transmission.

Much of the work done in bandwidth allocation therefore precludes that done in the database community and vice versa. For example, the dynamic broadcast scheduling makes pre-broadcast of indexing information impossible. On the other hand, cyclic data scheduling typically uses bandwidth inefficiently.

Since there are many factors involved it is hard to determine which broadcast scheduling is best. Each scheme focus on some different aspect from others and it is best to apply it in the context of the network environment.

3.2 Index Broadcasting

Original broadcast disks scheme do not use indexing. So the broadcast data has to be self-identifying and broadcast has to be done periodically for concurrent access to the data. Even with fixed cyclic broadcast, clients have to spend lots of time actively listening to the broadcast to acquire the information they need.

Index broadcasting is necessary to improve data access performance, especially tuning time. By using indexes for broadcast data tuning time is reduced to save battery consumption. The index provides fast item lookup by providing the time a required item will be broadcast. A client can then doze until the item is available to save energy. The indexed structure may be in the form of a B+ tree or simply organized as a sorted list of data items. Since the index is an important data structure, several fault-tolerant indexing schemes various are proposed [9].

3.3 Client-Side Caching

Due to the shared nature of the broadcast disk, while in principle allowing for nearly unlimited scalability, in fact gives rise to a fundamental tradeoff. Fundamental constraint of the broadcast disk paradigm would be that increasing the broadcast rate of one item must necessarily decrease the broadcast rate of one or more other items. A way out of this dilemma is to exploit the local memory and/or disk of the client machines to cache pages obtained from the broadcast. With the introduction to broadcast, role of client caching needs to be changed from a traditional client server information system. Traditional pull-based client server system caches their hottest data. In the push-based environment, this use of cache can lead to poor performance if the server's broadcast is poorly matched to the client's page access distribution. This difference arises because of the serial nature of the broadcast disk – broadcast pages are not all equidistant from the client.

In a push-based system client must use their cache not to store simply their hottest pages, but rather, to store those pages for which the local probability of access is significantly greater than the page's frequency of broadcast. Above argument leads to the need for cost-based page replacement. That is, the cost of obtaining a page on a cache miss must be accounted for during page replacement decisions.

Most algorithms on caching are done by original authors who proposed the broadcast disks. Following are couple of examples of client caching algorithms.

PIX

It can be shown that under certain assumptions, an optimal replacement strategy is one that replaces the cache-resident page having the lowest ration between its probability of access (P) and its frequency of broadcast (X). This ratio (P/X) is referred to as PIX (P Inverse X).

While PIX can be shown to be an optimal policy under certain conditions, it is not a practical policy to implement because it requires: 1) perfect knowledge of access probabilities and 2) comparison of PIX values for all cache-resident pages at page replacement time.

LIX

LIX is an implementable cost-based algorithm that is intended to approximate the performance of PIX. LIX is a modification of LRU (Least Recently Used) that takes into account the broadcast frequency. LRU maintains the cache as a single linked-list of pages. When a page in the cache is accessed, it is moved to the top of the list. On a cache miss, the page at the end of the chain is chosen for replacement.
In contrast, LIX maintains a number of smaller chains: one corresponding to each disk of the broadcast. When a new page enters a cache, LIX evaluates a *lix* value only for the page at the bottom of each chain. The page with the smallest *lix* value is ejected, and the new page is inserted in the appropriate queue. LIX performs constant number of operations per page replacement which is same order as that of the LRU.

3.4 Hybrid Approach

Originally broadcast disk with no feedback from the user was assumed. Broadcast scheduling is based on a predetermined statistical data of the clients. The problem with this is that broadcasting scheme does not always match client requests. There might be a scenario where client requests or group of clients change rapidly over time. Another problem is broadcasting unpopular items are waste of bandwidth for most clients and the waste of bandwidth increases with more users joining the network. Some hybrid approach, using push-based broadcast and pull-based query, have been developed to address such issues.

Paper by Guo, Das and Pinotti [6] suggest dividing data item into two disjoint sets: on consisting of more popular items and the other consisting of less-popular items. They propose an algorithm to decide on a cut-off point where more popular and less popular data are divided. Two channels are used for data dissemination: the index channel to transmit the index numbers of the broadcast data items, and the data channel to transmit the actual data. Server uses any push-based scheduling algorithms to broadcast. Whenever the server receives a request from a client, it will complete the current broadcasting, then send the requested item. Upon completion of the sending, the server will resume broadcasting.

On the client side, a client first listens to the index channel to check whether the needed item is currently broadcast or not. If the data item that the client needs is in the index, then the client just waits until the data is broadcast. If the item is not in the index, then the client asks the server to send the item for it.

There are also other hybrid approaches that have been proposed [7][8]. They are mainly similar in concept, using broadcast for sending more popular data and using pull-based query for less popular ones.

4. Efficient Data Allocation over Multiple Channels

Earlier works have assumed on using a single channel for broadcast. Recent papers argue that instead of having a single high bandwidth broadcast channel it is more likely that server has access to multiple lower bandwidth channels for broadcast [11]. This section focuses on a recent paper [5] proposing an algorithm to efficiently solve the multiple disjoint broadcast channel allocation problem.

4.1 Algorithms Used for Comparison

The most obvious algorithm, *FLAT* [2] simply allocates an equal number of items to each channel. *FLAT* is simple and easy to design. It can be reconfigured easily in case of change in system configuration. It also simplifies consistency control and tuning time optimization. However, it has performance problems since it allocated the same amount of bandwidth to popular and unpopular data items. Other algorithms include those that skew the amount of data allocated to each channel.

VF^K [10] allocates data hierarchically. It greedily partitions a set of data items, ordered by popularity, where the decrease in average expected delay is greatest. It then heuristically picks the partition with the greatest delay, and repeats the partitioning step until *K* partitions are generated. Assume that data items are arranged in a row, where the data items to the left have higher popularities than those to the right. In this case, splitting the row at some point creates new left and right partitions. VF^K restricts its search space by requiring that, during each split, the new right partition contains more data items than the new left one. Data items from each partition are then assigned to a broadcast disk.

Another skewed-allocation algorithm, which we call *BP*, is based on bin-packing and equalizes the access probability of each channel [11]. Among these three algorithms, there is a tradeoff between complexity and performance. *FLAT* is trivial to compute (O(*K*), where *K* is the number of channels), but offers the worst performance. VF^K gives good performance for a wide range of parameters, but is expensive to compute ($O(KN^2 logK)$, where *N* is the number of items). *BP* lies somewhere in between in terms of both complexity ($O(NlogN)$) and performance.

4.2 Architectural Assumptions

We assume that there are K channels in a broadcast area, each denoted C_i; $1 \le i \le K$. A database is made up of *N* unit-sized items, denoted d_j; $1 \le j \le N$. Each item is broadcast on one of these channels, so channel *i* broadcasts *Ni* items, $1 \le i \le K$, $\sum_{i=1}^{K} Ni = N$. Each channel cyclically broadcasts its items.

Time is slotted into units called ticks, which are defined as the amount of time necessary to transmit a unit of data. Each item d_i is assigned an access probability, p_i. Requests are for single items and assumed to be exponentially distributed, so p_i does not vary from tick to tick.

We also assume that clients know a priori the contents of the channels. In practice, however, some index information must be made available to clients. We make this simplifying assumption because the indexing problem is orthogonal to the allocation problem.

4.3 Problem Statement

Expected delay, w_i, is the expected number of ticks a client must wait for the broadcast of item i. Average expected delay is the number of ticks a client must wait for an average request and is computed as the sum of all expected delays, weighted by their access probabilities:

$$\text{Average Expected Delay } (AED) = \sum_{i=1}^{N} w_i p_i \ . \qquad (1)$$

When N_i items are cyclically broadcast on channel i, the expected delay in receiving an item j on this channel is With K channels, AED can be rewritten as:

$$\text{Multi-channel Average Expected Delay } (MCAED) = \frac{1}{2}\sum_{i=1}^{K}\left(N_i \sum_{d_j \in C_i} p_j \right). \quad (2)$$

The goal is to allocate database items to K channels in a way that minimizes *MCAED*. The canonical allocation technique, called *flat* design, allocates an equal number of items to each channel. With flat design, Ni=$\frac{N}{K}, \forall i$, and, from (2), *MCAED*=$\frac{N}{2K}$. Skewed design is discussed here, where items are placed on varying sized channels, depending on their popularities. Skewed design has been shown to be better than flat design at reducing *MCAED*.

4.4 Optimal Allocation with the DP Algorithm

The goal is to allocate n items to k disjoint subsets minimizing *MCAED*. Allocation problem is a partitioning problem, which can be optimally solved using dynamic programming. Dynamic programming solves problem by combining the solutions to sub programming, which is typically applied to optimization problems. It requires following two properties; optimal substructure and overlapping subproblems.

Theorem 4.1. In an optimal solution, for any two partitions, C_i, C_j, if $|C_i| < |C_j|$, then $\forall d_k \in C_i, d_l \in C_j, p_k \geq p_l$. Similarly, if $p_k > p_l$, then $|C_i| \leq |C_j|$.

Theorem 4.1 states that, in an optimal solution, more popular items are in smaller partitions. We can therefore order the items by popularity, then find an optimal solution by optimally "splitting" this set K - 1 times. This also implies that it has overlapping subproblems since temporary results to split each partition can be reused.

Dynamic programming from the two properties given can be used to construct an optimal solution. Define C_{ij} as the *AED* of a channel containing items *i* through *j*.

$$\text{Single Channel AED } (SCAED) = \text{Cik} = \left(\frac{j-i+1}{2}\right)\sum_{q=i}^{j} p_q \,, \qquad j \geq i\,,\ 1 \leq i, j \leq N \quad (3)$$

Let $opt\_sol_{i,K}$ be the optimal solution (i.e., minimum *MCAED*) for allocating items from *i* through *N* on *K* channels. The optimal solution for items *i* through *N* given one channel is then $opt\_sol_{i,1} = C_{i,N}$. We can now express the optimal MCAED with the following recurrence:

$$opt\_sol_{i,K} = \min_{l \in \{i,i+1,\ldots,N-1\}} (C_{il} + opt\_sol_{l+1,K-1})$$

Using dynamic programming to minimize the recurrence above requires O(KN^2) time and O(KN) space to keep track of partial solutions. We refer to this algorithm as *DP*.

4.5 A Cheaper Approximation - *GREEDY*

Although *DP* yields an optimal solution, its time and space complexity may preclude it from practical use. Therefore an alternative, significantly cheaper algorithm, called *GREEDY* is offered. *GREEDY* finds the split point among all the partitions that decrease cost the most. This process is repeated until there are *K* partitions.

Algorithm 4.1

```
GREEDY
input: set of N unit sized items ordered by popularity, K partitions
begin
        numPartitions := 1;
        while numPartitions < K
                do Let $point_k$ be the split point that most reduces
```

```
            SCAED for each partition k.
        Let point' be the point_k that most reduces MCAED.
        Create a split at point'.
        numPartitions := numPartitions + 1;
    od
end
```

Example. Consider the problem of allocating the set of N = 6 items from table 2. Using the GREEDY algorithm, the first split occurs between items 2 and 3 and the second occurs between items 1 and 2. These two splits reduce the average expected delay from 3 ticks to 0.95 ticks. This is illustrated in Figure 2.

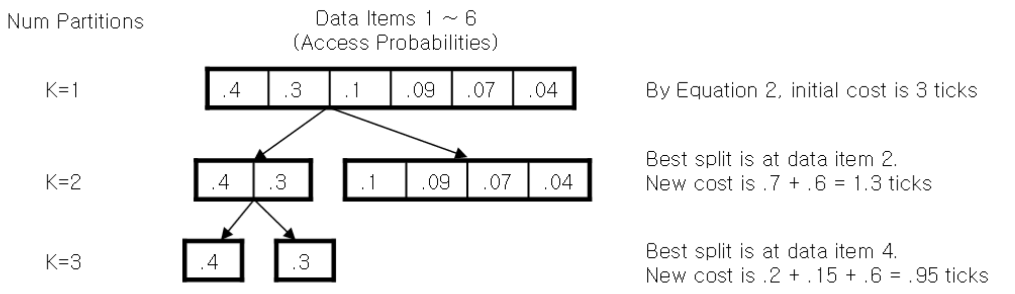
Num Partitions
Data Items 1 ~ 6
(Access Probabilities)
K=1
.4 .3 .1 .09 .07 .04
By Equation 2, initial cost is 3 ticks
K=2
.4 .3
.1 .09 .07 .04
Best split is at data item 2.
New cost is .7 + .6 = 1.3 ticks
K=3
.4
.3
Best split is at data item 4.
New cost is .2 + .15 + .6 = .95 ticks

Fig.2. Example of the *GREEDY* Algorithm. A Data set containing *N*=6 items is split into *K*=3 partitions.

4.6 Performance Results

Performance results were measured based on compute time and average expected delay among *FLAT*, *BP*, VF^K, *GREEDY*, and *DP*. Except for *FLAT*, the compute times of all algorithms increase with database size. *DP* requires too much memory (O(*KN*)) and VF^K has a high complexity($O(KN^2 log K)$). Average expected delay was measured with varying number of channels (*K*), the size of the database (*N*), and popularity skew (Θ).

GREEDY algorithm has a good performance/simplicity tradeoff compared to other algorithms. *FLAT* is simple to compute, but results in an *MCAED* up to 40 percent higher than that of *GREEDY*. VF^K has near-optimal *MCAED* for most parameters, but it takes orders of magnitude more time to compute than *GREEDY*, making it a poor choice for dynamic scheduling and single-channel schedule generation.

5. Discussion and Conclusion

Broadcast disks have been proposed to overcome asymmetric environment in the mobile network. There have been extensive researches done in different aspects of the push-based technology. There seems to be no winning scheme due to different approaches that are taken from different assumptions. It is very difficult to compare results since each paper may focus on different goals on different environments. Mobile network can be diverse and it should be taken into consideration the context where the broadcast based technology is being applied to. So there seems to be no absolutely correct way of doing a broadcast. What seems more important is applying appropriate technology in a given context.

However I find that there needs to be some comprehensive study of existing researches that have been done. Many researches are focused on just single aspect of the broadcast disk problem and it is difficult to put together different researches done in varying issues. It was also mentioned that some area of research preclude different area of research. Some comprehensive framework or model needs to be built to focus on combining existing research to form an effective solution.

References

[1] S. Acharya, M. Franklin, S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Personal Communications*, vol. 2, no. 6, pp. 50 -60, Dec. 1995.

[2] Swarup Acharya, Rafael Alonso, Michael Franklin, Stanley Zdonik, "Broadcast disks: data management for asymmetric communication environments," *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, vol. 36, no. 7, May 1995.

[3] Peter Triantafillou, R. Harpantidou, M. Paterakis, "High performance data broadcasting systems," *Mobile Networks and Applications*, vol. 7, no. 4, August 2002.

[4] Swarup Acharya, Michael Franklin, Stanley Zdonik, "Balancing push and pull for data broadcast," *ACM SIGMOD Record*, vol. 26, no. 2, pp. 183-194, June 1997.

[5] Wai Gen Yee, S.B. Navathe, E. Omiecinski, C. Jermaine, "Efficient data allocation over multiple channels at broadcast servers," *Computers, IEEE Transactions on* , vol. 51, no. 10, pp. 1231-1236, Oct. 2002.

[6] Yufei Guo, Sajar K. Das, Cristina M. Pinotti, "A new hybrid broadcast scheduling algorithm for asymmetric communication systems: push and pull data based on optimal cut-off point," *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pp.123-130, July 2001.

[7] K. Stathatos, N. Roussopoulos, and J.S. Baras, "Adaptive Data Broadcast in Hybrid Networks," *Proc. VLDB*, 1997.

[8] W.C. Lee, Q. Hi, and D.L. Lee, "A Study on Channel Allocation for Data Dissemination in Mobile Computing Environments," *J. Mobile Networks and Applications*, vol. 4, pp. 117-129, 1999.

[9] K.C.K. Lee, Hong Va Leong, A. SiA, "Semantic data broadcast for a mobile environment based on dynamic and adaptive chunking," *Computers, IEEE Transactions on*, vol. 51, no. 10, pp. 1253 -1268, Oct. 2002.

[10] W.C. Peng and M.S. Chen, "Dynamic Generation of Data Broadcasting Programs for Broadcast Disk Arrays in

a Mobile Computing Environment," *Proc. ACM Conf. Information and Knowledge Management (CIKM)*, pp. 35-45, Nov. 2000.

[11] K. Prabhakara, K.A. Hua, and J. Oh, "Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment," *Proc. Int'l Conf. Data Eng. (ICDE)*, 2000.

[12] Anindya Datta , Debra E. VanderMeer, Aslihan Celik, Vijay Kumar, "Broadcast protocols to support efficient retrieval from databases by mobile users," *ACM Transactions on Database Systems (TODS)*, vol. 24, no. 1, March 1999.

[13] K. Foltz, J. Bruck, "Splitting schedules for Internet broadcast communication," *Information Theory, IEEE Transactions on*, vol. 48, no. 2, pp. 345-358 Feb. 2002.