E. Workshop on Databases and Software Engineering

Performance Improvement and Evaluation for Indexing on Nonuniform Data Broadcast

Chun-Lin Han* Jing Chen
Department of Electrical Engineering
National Cheng Kung University
No. 1, Dashiue Rd., Tainan, Taiwan 701, R.O.C
E-Mail：{n2689151, jchen}@ccmail.ncku.edu.tw
TEL: (886) 916-082998, (886) 6 275-7575 ext. 62376
FAX: (886) 6 234-5482

Abstract

In this paper, we consider wireless data broadcasting as a way of disseminating information to a massive number of clients equipped with battery powered palmtops. Using indexing technique, the clients can selectively tune in at desirable portion of the broadcast and conserve the usage of energy. We review the pervious works for indexing on nonuniform data broadcast in this paper and propose an new indexing scheme, namely, balanced hybrid indexing to improve the performance. To measure the performance of different indexing schemes, we conducted three experiments for performance evaluation. The results show our scheme has better performance in both the average access time and the average tuning time than pervious works.

Keywords: indexing, data broadcast, nonuniform broadcast, selective tuning.

***the contact author**

1. Introduction

In wireless environment, there are two fundamental modes to provide clients with information [IVB94b]:

(1) *Interactive/On-Demand*: The client sends requests to the server through uplink and the server responds by sending the desired data to the clients.

(2) *Data Broadcasting*: The server broadcasts data on a communication channel periodically. One or more clients just listen to the channel and wait for data coming.

Because data broadcast allows users to retrieve data simultaneously with a cost independent of the number of users, it becomes an attractive solution to compensate for the limited resources in the mobile environment.

Mobile devices usually use battery as the source of power. It is expected that the lifetime of a battery will increase only 20% over the next 10 years [SCB92]. Conserving energy of batteries for longer working time thus becomes an important issue. As a solution, the concept of selective tuning was introduced [AAFZ94][IVB94a]. The server broadcasts index information before data, so the mobile device can operate in the less power consuming doze mode and operate in active mode while listening to channel to receive the desired data. For example, the ratio of power consumption in active mode to doze mode is 5000 for the Hobbit chip from AT&T, so using indexing technique properly can conserve the batteries.

Conserving energy of batteries has two advantages [IVB94a]:

(1) *Downsize the mobile devices*: Battery size has been a limitation in reducing the size of mobile devices. By using indexing technique we can use smaller and less powerful batteries to run the same job.

(2) *Environmental consideration*: Every battery that is disposed is an environmental hazard.

Therefore, indexing technique becomes an important technique for wireless environment and real world.

In section 2, we describe our computational model and the concept of broadcast with indexing. Section 3 reviews related works. Our improvement for indexing on nonuniform broadcast is presented in section 4. Section 5 reports a performance evaluation to compare the various approaches in terms of average access and average tuning time. We conclude our discussion in section 6.

2. Computational Model

We consider the environment that a server broadcasts records periodically to a number of clients on a communication channel. Each period of the broadcast is called a broadcast cycle or bcycle. We assume that every record is broadcast with a primary key which can be used by the clients to filter unwanted records. The records are not necessarily static and can be updated frequently but updates to the records are reflected only between successive broadcasts. Hence the content of a broadcast cycle is completely determined before the start of that broadcast cycle.

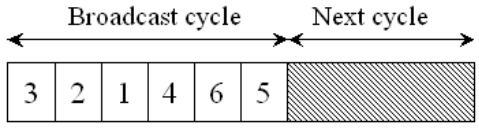
2.1 Broadcast and Indexing

In data broadcasting, we concern about two parameters:

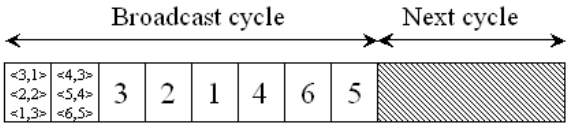
(1) *Access time*: The time elapsed from the moment a client wants a record to the time when required record is downloaded.

(2) *Tuning time*: The amount of time spent by a client listening to the channel. This will determine the power consumed by the client to retrieve the require record.

In a broadcast program with no indexes, both the access time and the tuning time will be N/2 in average where N is the cycle time of this broadcast program.



(a) A broadcast program with no indexes



(b) A broadcast program with indexes

Figure 2.1 An example of broadcast program

The concept of selective tuning is to add some index entries into a broadcast program. Every index entry is a <key, offset> pair. When client reads a <8, 6> pair, it means the record whose primary key is 8 will be broadcasted after 6 time units. Figure 2.1(a) is a broadcast program without indexing and Figure 2.1(b) is a broadcast program with indexes. If a client wants to the record with key 6, the client needs listen to the channel persistently before download record with 6. The access time and tuning

time will be 5 in Figure 2.1(a). In Figure 2.1(b), the client listens to the channel and gets index information for record 6, then tunes into doze mode and tunes into active mode when record 6 is broadcast. The access time is 7 and tuning time is 3.

We observed that indexes can decrease the tuning time but will increase the access time at the same time because of the increasing in the broadcast cycle. Hence, we need to add index information properly to get a good balance between access time and tuning time.

2.2 Nonuniform Broadcast

In uniform broadcasting, all data will be broadcast once in each broadcast cycle. Under this approach, all data will have the same average access time of a half of the length of a broadcast cycle. In the real world, however, some records are more frequently accessed than others; for example, hot stock information to cold one. In this case, it makes sense to broadcast the frequently accessed records more often than those that are less popular. This will reduce the average access time for records that are frequently accessed, while increasing the average access time for records that are less in demand. We call a broadcast is nonuniform if some data will be broadcast more than once in a broadcast cycle. Such a nonuniform broadcast was shown to be superior in terms of average access time than uniform broadcast [AAFZ94].

In nonuniform broadcast, we concern about two parameters:

(1) *Average access time*: The average of the access time of all clients listening the broadcast

channel.

(2) *Average tuning time*: The average of the tuning time of all clients listening the broadcast channel.

3. Related Works

Selective tuning on uniform data broadcast was studied in [IVB94a] and [IVB94b]. In [IVB94b], two indexing schemes were proposed to minimize the tuning time. The (1,m) indexing technique broadcasts an index m times during the broadcast of one version of a file. While this method reduces the tuning time, it leads to long access time because of the additional m copies of indexes. To cut down on the access time, the distributed indexing scheme exploits only partial replication at the index level so that several distinct indexes (with some common internal nodes) are used to index different portion of the data broadcast. Lo and Chen extended distributed indexing and proposed an adaptive access methods which tolerates the access failures [LC00]. Agrawal and Chrysanthis modified distributed indexing and proposed constant-size I-node distributed indexing [AC01], which offered lower access time and tuning time when records were fewer than 12000. Imielinski, Viswanathan and Badrinath proposed hashing scheme and flexible indexing scheme [IVB94a]. Hashing scheme uses hashing function and a shift pointer in the data buckets to support selective tuning. Flexible indexing scheme provides users using parameters to control the balance between access time and tuning time.

W.C. Lee and D.L. Lee used signature techniques to support selective tuning [LL96]. This method

is suitable for real-time information filtering on mobile clients and when the ratio of the size of the search key to data is large. Hybrid indexing proposed in [HLL01] combined strengths of the signature and the index tree techniques. This method has the advantages of both the index tree method and the signature method and has a better performance than the index tree method [HLL01].

Selective tuning on nonuniform data broadcast was studied in [TY96] and [TY97]. Tan and Yu extended the flexible indexing scheme to adapt to nonuniform broadcast [TY96]. In [TY97], Tan and Yu also gave a mathematical analysis to some indexing schemes for nonuniform broadcast. Note that the signature techniques and hybrid indexing can also adapt to nonuniform broadcast after some modifications.

Acharya, Alonso, Franklin and Zdonik proposed Broadcast Disk algorithm [AAFZ94], which was shown to be superior to uniform broadcast in average access time when access frequency of records is not uniform. Their algorithm was extended by Baruah and Bestacros to support fault-tolerant and real-time properties [BB97].

3.1 Broadcast Disk Algorithm

Since this paper focuses on indexing schemes on nonuniform broadcast, we introduce the most famous nonuniform broadcast algorithm called Broadcast Disk algorithm first. The broadcast disk algorithm has the following steps (for simplicity assume that data items are “pages”, that is, they are of a uniform, fixed length):

(1) Order the pages from hottest to coldest.

(2) Partition the list of pages into multiple ranges, where each range contains pages with similar access probabilities.

(3) Choose the relative frequency of broadcast for each of the partitions. The only restriction on the relative frequencies is that they must be integers. For example, given two partitions, partition 1 could be broadcast three times for every two times that partition 2 is broadcast, thus rel_freq(1) = 3 and rel_freq(2) = 2.

(4) Split each partition into a number of smaller units. These units are called chunks (C_{ij} refers to the *j*th chunk in partition i). First, calculate max_chunks as the Least Common Multiple (LCM) of the relative frequencies. Then, split each partition i into num_chunks(i) chunks, num_chunks(i) = max_chunks / rel_freq(i). In the previous example, num_chunks(1) would be 2, while num_chunks(2) would be 3.

(5) Create the broadcast program by interleaving the chunks of each disk in the following manner:

```
1 for i := 0 to max_chunks – 1
2    for j := 1 to num_disks
3     Broadcast chunk C_{j,(i mod num_chunks(j))}
4    endfor
5 endfor
```

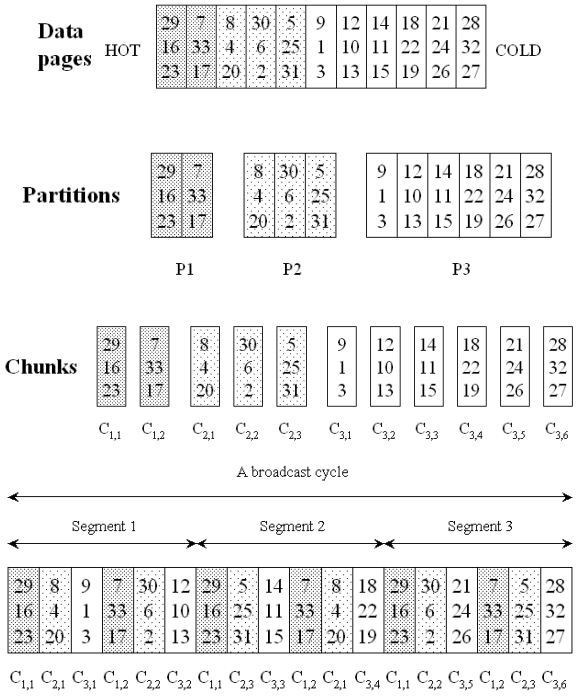


Figure 3.1 Deriving a server broadcast program

The broadcast can also be viewed as a sequence of equal sized segments such that partition 1 appears in all segments. Figure 3.1 shows an example of generating broadcast program. Let a data file comprises 33 records be split into 3 partitions P1, P2 and P3, where partitions P1, P2 and P3 comprise 6 records (chunks $C_{1,1}$ and $C_{1,2}$), 9 records (chunks $C_{2,1}$, $C_{2,2}$and $C_{2,3}$) and 18 records (chunks $C_{3,1}$ to $C_{3,6}$) respectively. Suppose partitions P1, P2 and P3 are to be broadcast 3, 2 and 1 times respectively in a broadcast cycle, i.e., they will be split into 2, 3 and 6 chunks respectively.

3.2 Distributed Indexing

Tan and Yu proposed distributed indexing scheme [TY96], which adapted and extended the flexible index scheme in [IVB94a] for nonuniform broadcast. At the beginning of each section, there is a control index, which comprises two components: a global index and a local index. The global index is

used to determine the section, which a record may be found, while the local index provides the offset to the portion within the section where the record may be found. The scheme works as follows:

(1) Organize the nonuniform broadcast into segments.

(2) Sort the records in each segment.

(3) Build a flexible index for the records in each segment. For section i of a k-section segment, its global index contains k – i + 1 entries that provide offset to all sections that come after it. Its local index will have m pointers that split the data in the section into m + 1 portions and provides the offsets to the m + 1 portions.

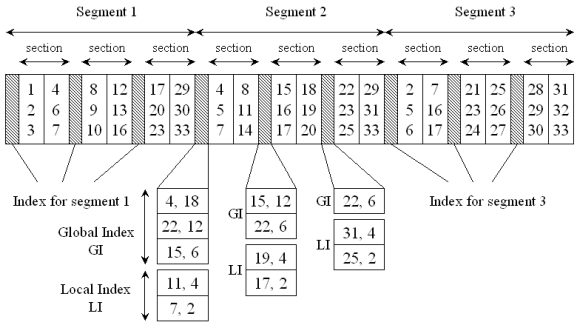


Figure 3.2 Distributed indexing scheme

Figure 3.2 is a Distributed indexing scheme example. Here, each segment is divided into 3 sections of 6 records each. Consider segment 2. For its first index, there are 3 entries in the global index. The first entry, (4, 18), implies that any record that is less than 4 can be found after the 18th record, i.e., in the next segment. The second entry (22, 12) means that a record larger than or equal to 22 can be found after the 12th record, i.e., in the last section of the segment. Similarly the third entry (15, 6) indicates

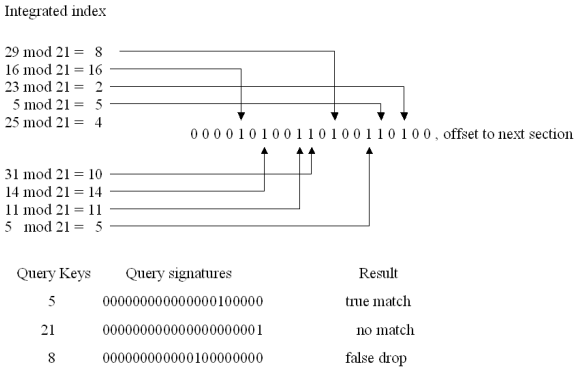
that a record between 15 and 22 may be found 6 entries away, i.e., in the second section of the segment. Implicitly, record that may be found in the first section of the segment follows the local index. The local index has 2 entries. Consider the local index for the first index of segment 2. The entry (11, 4) means that a record with value 8 and above may be found after 4 records away. The entry (7, 2) refers to a record with value above 3 but less than 8 can possibly be found 2 records away. Implicitly, any record less than value of 3 can be found at the end of the local index.

3.3 Signature Technique

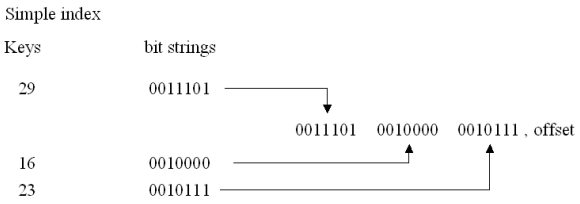
Signature methods have been widely used for information retrieval. A signature of a data frame is a bit vector generated by first hashing the values in the data frame into bit strings and then superimposing them together. The signature technique interleaves signatures with their associated data frames in data broadcasting [LL96]. Although the signature techniques does not design for nonuniform broadcast, we can still apply it to nonuniform broadcast after some modifications. In our model, we make signatures form the primary key of a record. At the beginning of each section, there is a control index, which comprises two components: an integrated index and a simple index. The integrated index is used to determine the section in which a record may be found.

Figure 3.3(a) shows the format of integrated index. Each integrated index has two parts. The first part is a signature formed by primary keys of 9 records and the second part is the offset to next section. If we want to check if the record with key 5 is in this section or not, we generate a query signature and

do an AND operation with first portion of integrated index. Because the result is not zero, the record with key 5 may be in this section. Note that even if the result is not zero, we still cannot be sure if the record is in this section, for example, the record with key 8. We call this situation as a false drop, and we need to check the every simple index further.



(a) The format of integrated index



(b) The format of simple index

Figure 3.3 Signature-based index entries

The simple index provides the offset to the portion within the section where the record may be found. Figure 3.3(b) shows the format of simple index. The simple index can divide into 4 parts. The first three parts are the signatures of three corresponding records in this section. The last part is a pointer to the start address of the first record of three corresponding records in this section. If we want to filter the record with key 8 is in this section or not, we cut the lower 7 bits of key 8 and compare it with each part of the first three parts of the simple index. Then we can find out that the record with

key 8 is not in this section.

This scheme works as follows:

(1) Build an integrated index for every 9 records in each segment.

(2) Build 3 simple indexes to point 9 records.

Figure 3.4 shows an example of the signature-based indexing scheme.

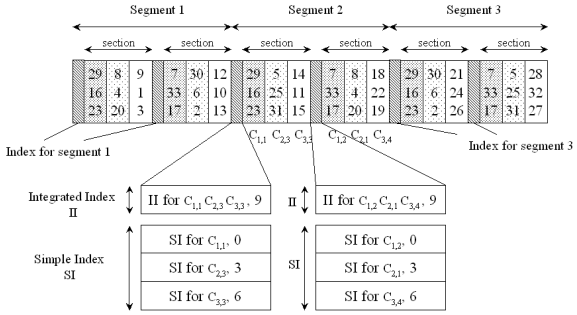


Figure 3.4 Signature-based indexing scheme

4. Our Scheme

Both the distributed indexing and the signature have advantages and disadvantages in one aspect or the other. For example, the distributed indexing provides a more accurate and complete global view of the data frames based on its indexed value. Since the clients can quickly find out the arrival time of the desired data, the tune-in time is normally very short. However, sorting the records in each segment may lead the hottest record to be broadcast later than the coldest one and it will increase the average access time. A signature does not contain global information about the data frames, so it can only help the clients to make a quick decision on whether the current frame (or a group of frames) is relevant to the query or not. The filtering efficiency heavily depends on the false drop probability of the signature. As a result, the tune-in time is normally high and is proportional to the length of the broadcast cycle. Intuitively, to address the above issues, a hybrid indexing approach such as out proposed in [HLL01] can be applied to nonuniform broadcast. However, it still has the same problems as distributed

indexing. Moreover, although the sparse tree which hybrid indexing used can provide the more precise global information, it also has larger overhead in average access time [AC01]. Therefore, a more sophisticated scheme appears necessary if a hybrid indexing approach is adapted.

4.1 **Balanced Hybrid Indexing**

Based on the above, we propose an indexing scheme, called balanced hybrid indexing scheme, which use the concept of the hybrid indexing and integrate the advantages of the distributed indexing scheme and the signature. This scheme works as follows:

(1) Organize the nonuniform broadcast into segments.

(2) Sort the records in each chunk. If the number of records in the chunk is smaller than ServerDBsize/10, we combine adjacent chunks and sort them (ServerDBsize is the number of the records to be broadcast).

(3) Build a flexible index for the records in each chunk. For section i of a k-section chunk, its global index contains k – i + 1 entries that provide offset to all sections that come after it. We replace the local index with simple index and use enough simple indexes to make sure that simple indexes can cover all records in the section.

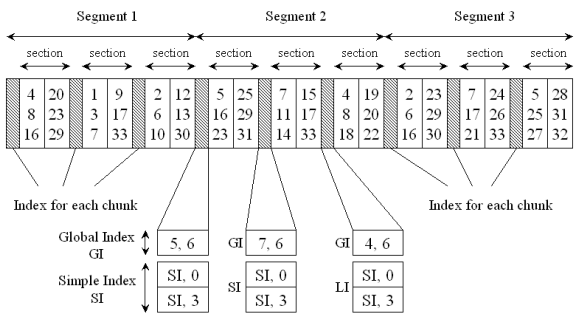


Figure 4.1 Balanced hybrid indexing scheme

Figure 4.1 shows an example of the balanced hybrid indexing scheme.

Because the balanced hybrid indexing scheme sort records and build flexible index in each chunk, the situation that hottest record is broadcast later than coldest one will not happen. The records in the same chunk have similar access frequency, so we sort the records in the each chunk will not influence the average access time too much. In addition, build index in each chunk needs fewer global indexes than distributed indexing scheme to complete the same thing, and using flexible index has lower overhead in the average access time than hybrid indexing. Hence, balanced hybrid indexing scheme can always has less average access time than distributed indexing scheme and hybrid indexing scheme.

Balanced hybrid indexing scheme has larger average tuning time by intuition, because we need to tune into each chunk to search records, and distributed indexing scheme and hybrid indexing scheme only tune into each segment to search records. But when the number of records in the segment is large, distributed indexing scheme needs more local indexes to keep low average tuning time and too many local indexes will increase the average access time. Balanced hybrid indexing scheme uses simple indexes to replace local indexes and the number of the records in chunk is less than segment, so we can use fewer simple indexes to keep low average tuning time and do not influence average access time too much. To compare balanced hybrid indexing scheme with hybrid indexing scheme, hybrid indexing scheme seems to have better average tuning time. However, in the situation that two schemes build index information for the same number of the record, hybrid indexing will generate

more index information than balanced hybrid indexing. It will increase the average tuning time, because there is more index information to be filter.

5. Performance Evaluation

To measure the performance different indexing schemes, we conducted three experiments for performance evaluation. In each experiment, 1,000 queries were posed. The average access time and average tuning time under the various schemes – distributed indexing scheme (DI), signature-based indexing scheme (SI), hybrid indexing scheme (HI) and balanced hybrid indexing scheme (BHI) – were compared.

5.1 Simulation Model

The model comprises a set of clients and a server. The server broadcasts the data file periodically. The file has *ServerDBSize* records and is organized into *NumPart* partitions. The default values for *ServerDBSize* and *NumPart* are 10,000 and 3 respectively. To determine the number of times each partition is broadcast, we follow the work in [AAFZ94] by introducing the relative frequency parameter δ. Partition *NumPart* will be broadcast once. Partition i, $1 \leqq i \leqq NumPart$, will be broadcast $((NumPart - i)\,\delta + 1)$ times. Setting $\delta = 0$ leads to a uniform broadcast, otherwise the broadcast will become nonuniform.

To determine the size of each partition, we assume that the number of records in each partition

follows a Zipf-like distribution [Knut81] [Gray94]. The Zipf distribution is typically used to model non-uniform access patterns. It produces access patterns that become increasingly skewed as Zipf-factor θ increases. For a file with *ServerDBSize* records, the *i*th partition, for 1 ≦ i ≦ *NumPart*, has such number of records as given by the following expression:

$$PartSize_i = \frac{ServerDBSize}{(NumPart - i + 1)^{\theta} \times \sum_{j=1}^{NumPart} \frac{1}{j^{\theta}}} \qquad (5.1)$$

When $\theta = 0$, all partitions have the same number of records, and setting $\theta = 1$, we have the nonuniform pure Zipf distribution. A Zipf-factor of 1 is used as default. We assume an index element can be put into a packet and the size of a record is 32 packets default.

Each client issues a query, listens to the broadcast, and downloads the desired record. In our study, we have also varied the frequency of access (in percentage) of partition i according to the Zipf distribution as follows:

$$FrequencyAccessed_i = \frac{100}{i^{\theta} \times \sum_{j=1}^{NumPart} \frac{1}{j^{\theta}}} \qquad (5.2)$$

5.2 Experiment 1: Effect of ServerDBSize

In this experiment, we study the effect of different *ServerDBSize*. We set the value of *ServerDBSize* from 1,000 to 10,000 and compare the performance of each scheme.

Figure 5.1 shows the average access times for different numbers of records on a single channel. As we expected before, the average access times of SI and BHI are superior to DI and HI. This is because

SI and BHI use less index entries than DI and HI, and keep the hotter records to broadcast first.

Figure 5.2 shows the average tuning times for different numbers of records on a single channel. We can see that BHI is still superior to DI in average tuning time. This is because when the number of records is getting large, the portions of section divided by local index will increase too. It will take more tuning time to check whether the desired record is in the portion or not. However, if we use simple index, we can filter more precisely before we check the records in the portion. The average tuning time of HI will grow faster than BHI when the number of data items increases, because the more records to be broadcast the more index overhead in HI.

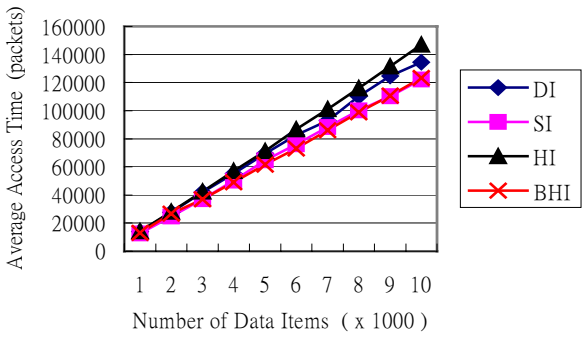


Figure 5.1 Average access time for 1000-10000 data items

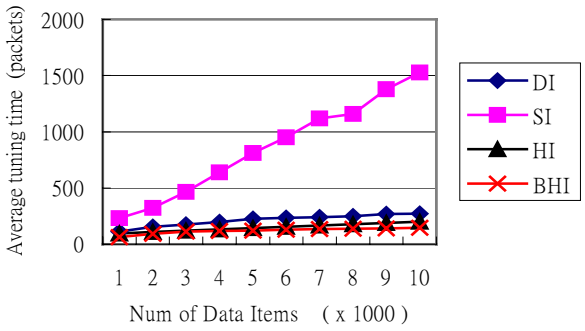


Figure 5.2 Average tuning time for 1000-10000 data items

The SI has the worst average tuning time, because SI needs to tune into channel every 9 records before it finds its desired record.

5.3 Experiment 2: Effect of partition size and frequency of access

The partition size is determined by equation 5.1, and the frequency of access is determined by

equation 5.2. The result as the value of θ from 0.2 to 1.0 is shown in Figure 5.3 and Figure 5.4.

First, The relative performance of the various schemes remains unchanged: BHI still has better performance than DI and HI. SI has worst average tuning time. Second, as θ increase, both the access time and tuning time decrease. This is expected since a higher θ value implies that the size of the most frequently accessed partition is smaller, and the number of clients accessing this partition increases.

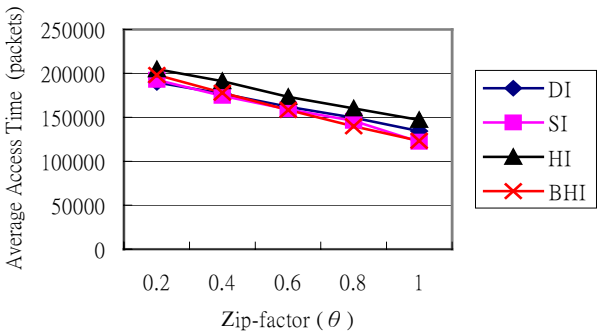


Figure 5.3 Average access time for different θ

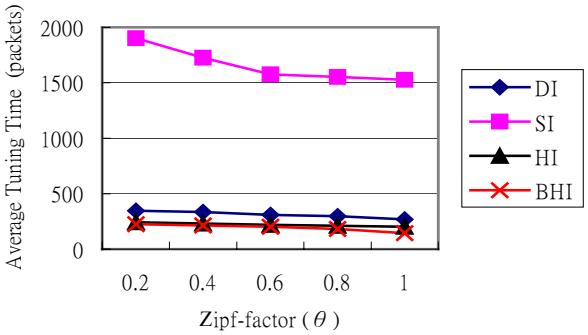


Figure 5.4 Average tuning time for different θ

5.4 Experiment 3: Effect of δ

Varying δ leads to different frequency of broadcast for each partition. The result as the value of δ from 1 to 7 is shown in Figure 5.5 and Figure 5.6.

The BHI will be getting worse when δ increase. This is because the number of chunk in each partition will increase when δ increase, BHI need to use more index items. This will increase the average access time and average tuning time. It does not matter since DI and HI cannot get any advantage in larger δ, so we can choose smaller δ to avoid this problem.

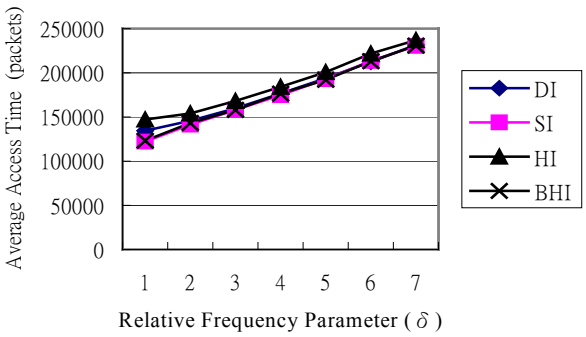


Figure 5.5 Average access time for different δ

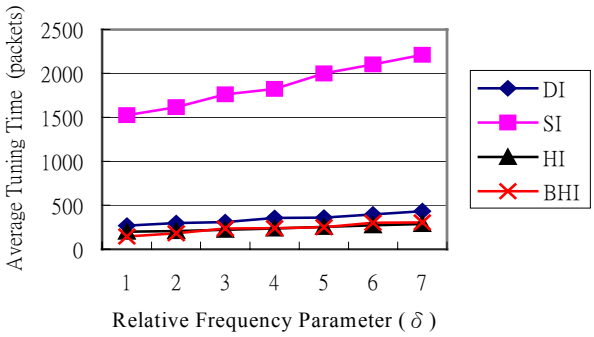


Figure 5.6 Average tuning time for different δ

6. Conclusion

Recent advances in technology have provided portable computers with wireless interfaces that allow networked communication even while a user is moving. In the near future, we can expect that millions of mobile users equipped with portable computers. They can retrieve information through the wireless networks anytime and anywhere. Conserving energy of batteries is very important for mobile devices. Using indexing technique, the clients can selectively tune in at desirable portion of the broadcast and conserve the usage of energy.

In this paper, we review the pervious works for indexing on nonuniform data broadcast and propose an new indexing scheme, namely, balanced hybrid indexing to improve the performance. We conducted three experiments for performance evaluation and the results show our scheme has better performance in both the average access time and the average tuning time than pervious works.

7. References

[Kunt81] D. Knuth, “The Art of Computer Programming, Vol II”, Addison Wesley, 1981.

[SCB92] Samuel Sheng, A. Chandrasekharan, R. W. Broderson, “A portable multimedia terminal for personal communications,” *IEEE Communicaton Magazine*, pp. 64-75, December 1992.

[AAFZ94] Swarup Acharya, Rafael Alonso, Michael Franklin and Stanley Zdonik, “Broadcast Disk: Data Management for Asymmetric Communication Environments,” *Communications of the ACM*, Vol. 37, No. 10, October 1994.

[Gray94] J. Gray, et al., “Quickly Generating Billion-Record Synthetic Databases”, *Proc. ACM SIGMOD Conf.*, May 1994.

[IVB94a] T. Imielinski, S. Viswanathan and B.Badrinath, “Power Efficient Filtering if Data on Air,” *4th International Conference on Extending Database Technology (EDBT)*, pp. 245-258, Cambridge, England, March 1994.

[IVB94b] T. Imielinski, S. Viswanathan and B.Badrinath, “Energy Efficient Indexing on Air,” *Proc. ACM SIGMOD Conf.*, pp. 25-36, May 1994.

[LL96] W. C. Lee and D. L. Lee, ”Using Signature Techniques for Information Filtering in Wireless and Mobile Environments,” *Distributed and Parallel Databases*, Vol. 4, No. 3, pp. 205-227, July 1996.

[TY96] K.L. Tan and J.X. Yu, “Energy Efficient Filtering of Nonuniform Broadcast,” *16th IEEE International Conference on Distributed Computing Systems*, pp. 520-527, Hong Kong, May 1996.

[BB97] Sanjoy Baruah and Azer Bestacros, “Timely and fault-tolerant data access from broadcast disks: A pinwheel-based approach,” *ACM DART ‘96*, pp. 45-49, 1997.

[TY97] K.L. Tan and J.X. Yu, “An Analysis of Selective Tuning Schemes for Nouniform Broadcast,” *IEEE Trans. on Knowledge and Data Engineering*, Vol. 22, No. 3, pp. 319-344, 1997.

[LC00] Shou-Chih Lo and Arbee L.P. Chen, “An Adaptive Access Method for Broadcast Data under an Error-Prone Mobile Environment,” *IEEE Trans. on Knowledge and Data Engineering*, Vol. 12, No. 4, pp. 609 –620, July 2000.

[AC01] Ruchi Agrawal and Panos K. Chrysanthis, “Efficient Data Dissemination to Mobile Clients in E-Commerce Applications,” *3rd WECWIS International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pp. 58-65, June 2001.

[HLL01] Qinglong Hu, Wang-Chien Lee, Dik Lun Lee. “A Hybrid Index Technique for Power Efficient Data Broadcast,” *Distributed and Parallel Databases*, Vol. 9, No. 2, pp. 151-177, 2001.