An Adaptive Query Processing Method according to System Environments in Database Broadcasting Systems

M. KASHITA
Graduate School of Engineering,
Osaka University
2-1 Yamadaoka, Suita,
Osaka 565-0871, Japan
kashita@ise.eng.osaka-u.ac.jp

T. TERADA
Cybermedia Center,
Osaka University
5-1 Mihogaoka, Ibaraki,
Osaka 567-0047, Japan
tsutomu@cmc.osaka-u.ac.jp

T. HARA
Graduate School of Information
System and Tech., Osaka University
2-1 Yamadaoka, Suita,
Osaka 565-0871, Japan
hara@ist.osaka-u.ac.jp

M. TSUKAMOTO
Graduate School of Information System
and Tech., Osaka University
2-1 Yamadaoka, Suita,
Osaka 565-0871, Japan
tuka@ist.osaka-u.ac.jp

S. NISHIO
Graduate School of Information System
and Tech., Osaka University
2-1 Yamadaoka, Suita,
Osaka 565-0871, Japan
nishio@ist.osaka-u.ac.jp

Abstract

In the recent evolution of wireless communication technology, there has been an increasing interest in the database broadcasting system where a server periodically broadcasts contents of a database to mobile clients such as portable computers and PDAs. There are three query processing methods in the database broadcasting system: the on-demand method, the client method, and the collaborative method which we have proposed. The performance of these methods is strongly affected by environmental conditions such as query generation intervals and size of query results. Since the system condition keeps changing, it is difficult to choose the optimal method statically. In this paper, we propose an adaptive query processing method which dynamically chooses a method which gives the best performance at the moment. Our method chooses a method according to query generation intervals in order to reduce the response time and to increase the query success rate. We also show simulation results regarding performance evaluation of our method.

1 Introduction

The recent evolution of wireless communication technologies has led to an increasing interest in information systems in which data is disseminated via broadcast channels. In such systems, a server broadcasts various data periodically via a broadband channel, and a client picks out and stores necessary data.

Many studies have been done so far to improve the performance of data broadcasting systems. They include data scheduling techniques at the server side [1, 9], caching techniques at the client side [1], update propagation techniques[2], integration of push-based and pull-based techniques[3], and data pre-fetching techniques[4]. Most of them deal with broadcast data only as *data items*, and do not address performance improvement by considering contents, characteristics, and types of broadcast data. In this paper, we assume a data broadcasting system in which a server broadcasts contents of a relational database and clients issue database queries to retrieve data from the database. We call such a system a *database broadcasting system*.

Currently, there are three query processing methods in the database broadcasting system: the on-demand method, the client method and the collaborative method[6] we proposed. The performance of these methods is strongly affected by environmental conditions such as query generation intervals and size of query results. Since the system condition always keeps changing, it is difficult to choose the optimal method statically. In this paper, we propose an adaptive query processing method which chooses adaptively a method which gives the best performance at every moment. Our method chooses a method according to query generation intervals in order to reduce the response time and to increase the success rate of queries.

The remainder of this paper is organized as follows. In section 2, we explain the outline of a database broadcasting system and introduce three basic query processing methods in the database broadcasting system. In section 3, we explain the adaptive query pro-

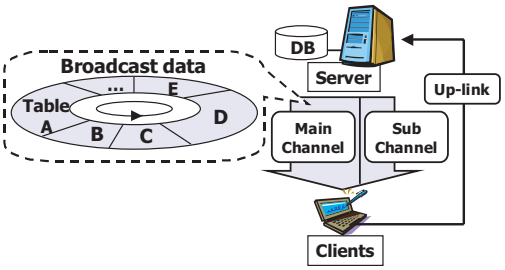

Figure 1: Database broadcasting system.

cessing method we propose in this paper. In section 4, we evaluate the performance of our method. Finally, we conclude this paper in section 5.

2 Database broadcasting system and query processing methods

Figure 1 illustrates the concept of the database broadcasting system. The server broadcasts contents of a relational database via a broadcast channel and clients issue queries to retrieve necessary data from the database. We also put the following assumptions:

Clients: Clients have a small storage, low battery, and low CPU power.

Dual downlink channels: The broadcast channel from the server to clients is divided into two channels: broadband *main channel* for sending the contents of databases, and narrowband *sub channel* for sending the other data.

Uplink channel: There is a narrowband communication channel from a client to the server.

In the database broadcasting system, there are three query processing methods, on-demand, client and collaborative methods. In the following, we explain each of three methods.

2.1 On-demand method

The processing procedure in the on-demand method is as follows:

1. A client sends a query described in SQL to the server via the uplink.

2. The server processes the query and broadcasts the query result via the sub channel.

In this method, since the query processing is completely done within the server, no workspace is required for query processing at the client. Moreover, in case where there is no waiting data to be broadcast for the sub channel, the client can receive the query result immediately. However, since the sub channel is exhausted when queries are issued frequently or when the size of query results is very large, it takes long time for the client to receive the query result.

2.2 Client method

The processing procedure in the client method is as follows:

1. A client monitors the main channel and stores broadcast tables which are necessary for processing the query on its storage.

2. The client processes the query when all the necessary tables are stored.

In this method, even if the number of clients intensively increases, each client can receive all of necessary data within one broadcast cycle. Moreover, since this method uses no uplink, it can work even in environments without the uplink. However, since a client has to store all necessary tables for query processing, a large storage is required on the client. Accordingly, there is the possibility that the client cannot process the query in case that the storage size of the client is small.

2.3 Collaborative method

The processing procedure in the collaborative method is as follows:

1. A client sends a query to the server via the uplink.

2. The server allocates each tuple in the broadcast database the *query identifier* and the *combination identifier*. The query identifier is a unique identifier given by the server for each query. The combination identifier is an identifier to match tuples when a query result is constructed from multiple tables such as the case of a join operation. The server investigates which tuples appear in the query result. Then, the server writes the query identifier and the combination identifier in those tuples. By referring to these two attributes, the client can store only tuples appearing in the query result.

3. The server creates rules which specify how the client receives only the necessary tuples for the query result and reconstructs the query result. Then the server broadcasts them via the sub channel.

Table 1: Characteristics of three methods.

Method	Response time		Required storage size
	Query generation interval: short	long	
On-demand	Bad	Good	Good
Collaborative	Medium	Medium	Medium
Client	Good	Medium	Bad

4. Based on the received rules, the client receives necessary tuples and reconstructs the query result automatically by combining these tuples.

Since the size of rules is generally much smaller than that of a query result, the sub channel is rarely exhausted in the collaborative method as compared with the on-demand method. Moreover, since the client can store only the necessary data by referring to the identifiers in the collaborative method, the storage size required on the client is reduced as compared with the client method.

2.4 Performances of each methods

Table 1 shows the characteristics of three methods. The response time, which is the elapsed time from generation of a query to receipt of the query result of the on-demand method, is strongly affected by the frequency for query generations. The response time of the collaborative method is little affected by the frequency of query generations and that of the client method is never affected by it. When queries are issued with the low frequency, the server in the on-demand method can broadcast the query results immediately, and thus, the response time is short.

The response time of the collaborative method is slightly longer than that of the client method since the broadcast cycle becomes longer due to the added identifiers.

The on-demand method requires no workspace to process the query since the client receives only the query result. On the other hand, the client method requires a large free storage since clients have to store all tables for query processing. In the collaborative method, the necessary storage size is smaller than that in the client method since clients store only tuples which appear in the query results.

In the collaborative method, the response time is slightly long compared with that of the client method since the broadcast cycle becomes long by identifiers.

3 Adaptive method

As mentioned above, the system performance, when each method is used individually, changes according to environmental conditions. Thus, in this section, we propose the adaptive method which dynamically chooses the best method from three methods, on-demand, the client, and the collaborative methods, according to the change of environmental condition. The proposed method improves the system performance by choosing on the appropriate method according to query generation intervals.

3.1 Processing procedure

The processing procedure in the adaptive method is as follows:

1. A client sends a query to the server via the uplink.
2. The server chooses an appropriate method from three methods by comparing the total size of pending data to be broadcast the data size in the *broadcast queue*.
3. The server inserts to the broadcast queue the data produced by each method for query processing:
   (a) If the server has chosen the on-demand method, the server inserts the query result.
   (b) If the server has chosen the collaborative method, the server inserts the created rules.
   (c) If the server has chosen the client method, the server inserts a message, which is an instruction for the query issue client to process the query by the client method.

3.2 Criteria of method choice

As mentioned in section 2.4, the system performance of each method depends on the frequency of query generations. In the adaptive method, we use the size of pending data to be broadcast in the broadcast queue as the parameter which represents the frequency of query generations. The system chooses the on-demand method when the data size in the broadcast queue is small, the collaborative method when the size is moderate, and the client method when the size is large. By this means, the system can keep the average total size of pending data in the broadcast queue almost a constant and can reduce the response time.

The adaptive method uses two thresholds, OC_Th and CC_Th where $OC\_Th \leq CC\_Th$. OC_Th is a threshold to make a choice between the on-demand and collaborative methods. CC_Th is that to make a choice between the collaborative and client method. The system chooses an appropriate method by comparing to these thresholds with the total size of pending data in the broadcast queue, S. The on-demand method is chosen when S is smaller than OC_Th, the collaborative method is chosen when S is between

OC_Th and CC_Th, and the client method is chosen when S is larger than CC_Th.

4 Evaluation

In this section, we evaluate our method by comparing with the on-demand, collaborative and client methods from the following two view points:

Response time: The elapsed time from the query initiation to the receipt of the query result. It should be noted that the response time does not include the time for processing the query locally at the client side.

Success rate: The ratio of the number of queries that the client could get the results to the number of queries issued in the entire system.

4.1 Simulation model

We assume the following simulation environment:

- The database schema and the query model represents on an information service in a shopping center. In this service, the server broadcasts the information on shops, goods and so forth. Clients walk around with PDAs and receive the broadcast data.

  There are two tables for each of genres such as *apparel*, *interior*, and *restaurant*; one of the tables is a *shop table* {shopID, shop name, image} and the other is a *goods table* {goodsID, shopID, goods name, image}. For the sake of simplicity, we suppose all tables are of the same size. Moreover, a client issues only natural join queries with a shop table and a goods table in the same genre.

- The query generation intervals are given by the exponential distribution, where the mean is changed in the simulation experiments.

- The tuple usage rate, which represents the ration of the number of tuples appearing in the query result to the total number of tuples in a table, is given by the normal distribution, and the mean and the dispersion are changed in the simulation experiments.

- The query fails in case that the client cannot get the query result by the time limit and in case that the necessary storage size for the query processing is larger than the storage size of the client.

- Even if the query fails, the client does not issue the same query again.

Table 2 shows parameters and their values used for this evaluation.

Table 2: Parameters.

Name	Value
Number of genres	10
Number of shops per a genre	40
Number of goods per a shop	100
Size of a tuple	5[KByte]
Size of a rule	140[B]
Bandwidth of the main channel	10[Mbps]
Bandwidth of the sub channel	1[Mbps]
Mean of the tuple usage rate	$3, 5^*, 7$[%]
Dispersion of the tuple usage rate	1
Storage size of a client	20[MByte]
Number of identifiers per a tuple	100
OC_Th	$0.5, 1^*, 3$[MByte]
CC_Th	$2, 4^*, 8$[MByte]
Time limit	200[s]
Query number	10000

$*$: default value

4.2 Simulation results

Figure 3 and Figure 6 show the simulation results when using the default values. Figure 3 shows the average response time in changing the mean of query generation intervals. When queries are issued with low frequency, the response time of the on-demand method is smaller than that of the other methods. However, the longer the mean of query generation intervals becomes, the longer the response time becomes exponentially. This is because the sub channel is exhausted for sending query results. In the client method, even if query generation intervals change, the response time does not change. In the collaborative method, the response time does not change so much because the size of rules is small and the sub channel is hardly exhausted for sending them. Since the broadcast cycle in the collaborative method, becomes slightly longer due to added identifiers and extra waiting time for receiving rules, the response time is a little longer than that in the client method.

The adaptive method always achieves the good response time. When the mean of query generation intervals is less than 3, the response time is slightly longer than that in the client method. This is because some queries are processed by the collaborative or on-demand method, while all queries should be processed by the client method. Furthermore, when the mean of query generation intervals is more than 20, the response time is slightly longer than that in the on-demand method. This is because some queries are processed by the collaborative or client method due to queries happened incidentally with short intervals.

Figure 6 shows the success rate in changing the mean of query generation intervals. In the on-demand method, as the frequency of query generation gets higher, the size of pending data in the broadcast queue gets larger rapidly, and thus, the response time gets

much longer. As a result, many queries fail since the response time becomes longer than the time limit. In the client method, since queries fail only when the total size of data the client must store is larger than the storage size of the client, the success rate is not affected by query generation intervals. In the collaborative method, when queries are issued with extremely high frequency, the space for added identifiers runs short, and thus, the response time becomes long for waiting until the space is released.

In the adaptive method, when the mean of query generation intervals is small, most queries are processed by the on-demand method. On the other hand, when the mean of query generation intervals is large, most queries are processed by the client method. As a result, both timeout and lack of storage space are rarely happened, and thus, the success rate is always high.

4.2.1 Evaluation in changing the tuple usage rate

Figures 2, 4, 5, and 7 show the evaluation results in changing the tuple usage rate, r. In the client method, since a client stores the whole associated tables with the query, the tuple usage rate does not affect both the response time and the success rate. In the collaborative method, since the tuple usage rate does not affect the time necessary for receiving rules and storing the data, it also does not affect both the response time and the success rate. In the on-demand method, when the tuple usage rate is large, the size of query results is also large, and thus, the response time is long. Consequently, many queries fail and the success rate is low. In the adaptive method, the response time is affected by the tuple usage rate only in case where the on-demand method is frequently chosen. Therefore, the larger the tuple usage rate becomes, the response time becomes longer slowly.

4.2.2 Evaluation in changing OC_Th

Figure 8 and Figure 9 show the evaluation results of our proposed method for three different values of OC_Th. In Figure 8, lines for three OC_Th values cross on the point where the mean of query generation intervals is about 10. When OC_Th is small and the mean of query generation intervals is large, most queries are not processed by the on-demand method. When OC_Th is large and the mean of query generation interval is small, most queries are processed by the on-demand method. In Figure 9, the larger OC_Th becomes, the smaller the success rate becomes. This is because more queries are processed by the on-demand method as OC_Th gets larger.

The above results show that, when the mean of query generation intervals is large, the response time is improved by setting OC_Th small.

4.2.3 Evaluation in changing CC_Th

Figure 10 and 11 show the evaluation results of our proposed method for three different values of CC_Th. In figure 10, the larger CC_Th becomes, the longer the response time is. This is because queries are rarely processed by the client method when CC_Th is large. In Figure 11, the larger CC_Th becomes, the larger the success rate becomes. This is because more queries are processed by the client method as CC_Th gets larger, and thus, fewer queries fail due to lack of the storage size.

These results show that the response time is reduced by setting CC_Th small, and the success rate is increased by setting CC_Th large.

5 Conclusion

In this paper, we proposed the adaptive method for efficient query processing in a database broadcasting system. In the proposed method, the system chooses adaptively a query processing method which gives the best performance according to environmental conditions such as query generation intervals.

Moreover, we evaluate the proposed method by simulation studies. The simulation results showed that the adaptive method improves the response time and the success rate compared with those in the case where each of the conventional methods is used individually.

As part of our future work, we plan to extend the proposed method by exploiting the queuing theory and by considering the clients' processing capacity.

Acknowledgements

This research was supported in part by "The 21st Century Center of Excellence Program" of the Ministry of Education, Culture, Sports, Science and Technology of Japan, Special Coordination Funds for promoting Science and Technology of the Ministry of Education, Culture, Sports, Science and Technology of Japan, and Grant-in-Aids for Scientific Research on Priority Areas numbered 13780331 and 14019063 from Japan Society for the Promotion of Science.

References

[1] S. Acharya, M. Franklin, and S. Zdonik: "Broadcast Disks: Data Management for Asymmetric Communication Environments," *Proc. ACM SIGMOD*, pp. 199–210 (1995).

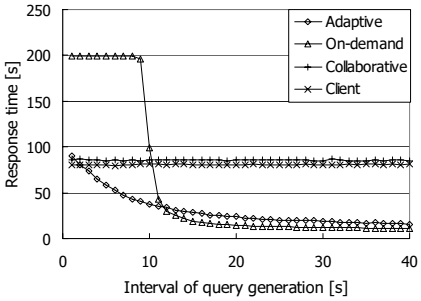

Figure 2: Response time(r=3).

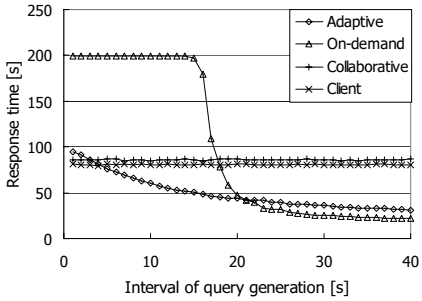

Figure 3: Response time(r=5).

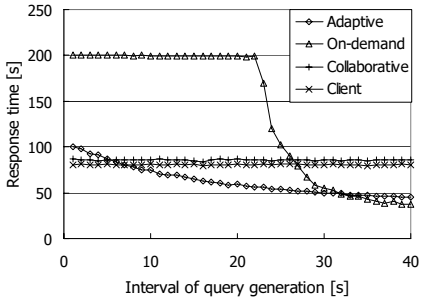

Figure 4: Response time(r=7).

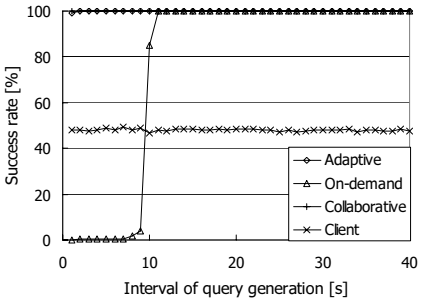

Figure 5: Success rate(r=3).

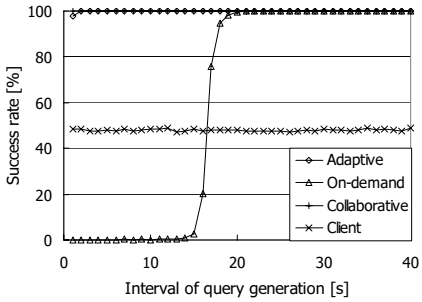

Figure 6: Success rate(r=5).

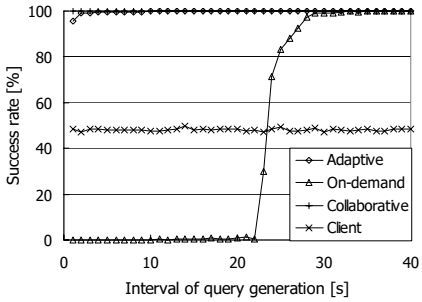

Figure 7: Success rate(r=7).

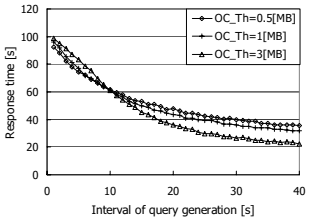

Figure 8: Response time in changing OC_Th.

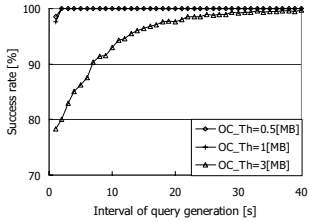

Figure 9: Success rate in changing OC_Th.

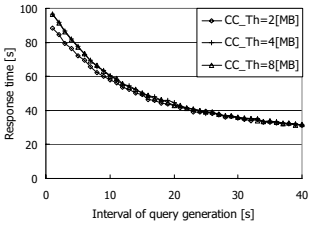

Figure 10: Response time in changing CC_Th.

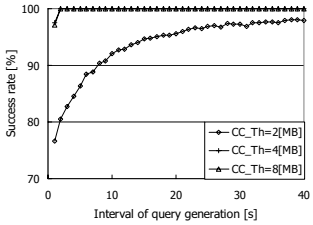

Figure 11: Success rate in changing CC_Th.

[2] S. Acharya, M. Franklin, and S. Zdonik: "Disseminating Updates on Broadcast Disks," *Proc. VLDB Conference*, pp. 354–365 (1996).

[3] S. Acharya, M. Franklin, and S. Zdonik: "Balancing Push and Pull for Data Broadcast," *Proc. ACM SIGMOD*, pp. 183–194 (1997).

[4] D. Aksoy, M. Franklin and S. Zdonik: "Data Staging for On-Demand Broadcast," *Proc. VLDB Conference*, pp. 571–580 (2001).

[5] Q. Hu, D. Lee, and W. Lee: "Performance Evaluation of a Wireless Hierarchical Data Dissemination System," *Proc. The Fifth Annual International Conference on Mobile Computing and Networking*, pp. 163–173 (1999).

[6] M. Kashita, T. Terada, T. Hara, M. Tukamoto, S. Nishio : "A Collaborative Query Processing Method for a Database Broadcasting System," *Proc. of IASTED Int'l Conf. on Communications, Internet and Information Technology (CIIT 2002)*, (2002 to appear).

[7] G. Lohman, L. Bruce, P. Hamin, and K. Bernhard, "Extentions to Starburst: Object, Types, Functions, and Rules," *Communications of the ACM*, vol. 34, no. 10, pp. 94–109 (1991).

[8] W. Peng, and M. Chen: "Dynamic Generation of Data Broadcasting Programs for a Broadcast Disk Array in a Mobile Computing Environment," *Proc. Int'l Conf. on Information Knowledge Management (ACM CIKM 2000)*, pp. 38–45 (2000).

[9] E. Yajima, T. Hara, M. Tsukamoto, and S. Nishio: "Scheduling Strategies of Correlated Data in Push-Based Systems," *Information Systems and Operational Research (INFOR)*, pp. 152–173 (2001).