Predicate-based Caching in Mobile Clients for Continuous Partial Match Queries

Yon Dohn Chung, Ji Yeon Lee, Yoon Joon Lee and Myoung Ho Kim
Division of Computer Science, Department of EECS,
Korea Advanced Institute of Science and Technology
373-1, Kusung-dong, Yusung-gu, Taejon, 305-701, Korea

Abstract

This paper proposes a cache management scheme for continuous partial match queries in mobile computing systems. Conventional cache management methods for mobile clients are record ID-based ones. However, since the partial match query is a content-based retrieval, the conventional record ID-based approach cannot properly manage the cache consistency. We show the predicate-based approach is an effective cache management in mobile environments.

1 Introduction and Motivation

The continuous partial match query is a continuous query whose type is a partial match query which retrieves data records by specifying some attributes, not all attributes [4]. The continuous query is a query whose result continues to exist in the client's memory consistently, not discarded after use [5].

For processing continuous queries, we must have a cache management method. The cache management methods used in distributed systems [6] (i.e., wireline systems) are too expensive to be used in mobile systems due to energy and bandwidth restrictions [1]. Therefore, the broadcasting-based cache management methods are widely used in mobile systems. In the broadcast-based cache management methods [1, 3], the server periodically broadcasts cache invalidation report (CIR) which has the information such as "which record is changed or not," "which record is changed when," and so on. Then, the mobile client receives the CIR and investigates the consistency of its cache. And, if some of the cached data are found inconsistent, then the client sends a request for updated data to the server.

However, since the partial match query is a content-based retrieval, conventional record ID-based CIR methods [1] are not efficient in cache management of mobile clients. Suppose that a mobile client caches the data records which are the results of Q. The data records in the server can change and their change is delivered to the client via CIR. But the checking of data records which the client currently caches is not sufficient for consistency maintenance because some data records which were not in the cache can be changed, and now become the result of Q.

In this paper, we propose a predicate-based CIR scheme for continuous partial match queries. In the proposed scheme we represent a partial match query as a partial match predicate using multiattribute hashing techniques [4], and construct the CIR with partial match predicates.

2 Proposed Cache Management Scheme

In the paper, we use the predicate representation using multiattribute hashing. The predicate is a bit stream of '0', '1', and '*', where specified attributes are hashed into binary bit streams and unspecified attributes are represented by bit streams of '*'.

We give an running example for explaining predicate representation. This is a wireless information system where mobile clients submit queries for stock price information. The stock price record consists of four attributes: 'A_1 (company ID)', 'A_2 (amount of sale)', 'A_3 (amount of purchase)' and 'A_4 (current price)'. The hash functions are as follows. Here, x is the attribute value of the given data record or query.

$$H_{A_1}(x) = \begin{cases} 00 & \text{if } x < 100 \\ 01 & \text{if } 100 \leq x < 200 \\ 10 & \text{if } 200 \leq x < 300 \\ 11 & \text{if } 300 \leq x \end{cases}$$

$$H_{A_2 \text{ and } A_3}(x) = \begin{cases} 00 & \text{if } x < 10000 \\ 01 & \text{if } 10000 \leq x < 20000 \\ 10 & \text{if } 20000 \leq x < 30000 \\ 11 & \text{if } 30000 \leq x \end{cases}$$

Proceedings of the Seventh International Conference on Database Systems for Advanced Applications (DASFAA '01)
0-7695-0996-7/01 $10.00 © 2001 IEEE

$$H_{A_4}(x) = \begin{cases} 000 & \text{if } x < 5 \\ 001 & \text{if } 5 \leq x < 10 \\ 010 & \text{if } 10 \leq x < 20 \\ 011 & \text{if } 20 \leq x < 30 \\ 100 & \text{if } 30 \leq x < 40 \\ 101 & \text{if } 40 \leq x < 50 \\ 110 & \text{if } 50 \leq x < 60 \\ 111 & \text{if } 60 \leq x \end{cases}$$

Suppose there is a data record R_k whose attribute values are "$A_1 = 150$, $A_2 = 13000$, $A_3 = 2000$, $A_4 = 22$." By concatenating the hashed bit streams of attributes, the predicate for R_k is "010100011." The predicate for query Q_1, which is to retrieve stock records such that " 20 < price < 30 and amount of purchase < 10,000", is "$****$00011."

In our caching scheme, we represent the cache state as a predicate (called *cache predicate*) that is the query submitted by the client. The mobile client submits a query Q_k i.e., a continuous partial match query, and the query is processed at the server and its result is returned to the client. Then, the result is cached at the client's cache memory.

We propose the predicate-based CIR method, where the CIR is organized with predicates and their timestamps. We call the new CIR structure P-CIR. Mobile clients receive the P-CIR and find predicates which are relevant to the client's cache predicate. Then, the clients investigate the consistency of their cache, and update some parts of the cache (if necessary) [2].

3 Comparison

As described in Section 1, the conventional record ID-based caching methods cannot support continuous partial match queries, since partial match queries are content-based. Thus, in this paper, we do not consider the conventional caching methods for comparison. Also, we do not consider the stateful server approach because it does not scale to a large number of clients that is our assumed environment. Thus, we compare the effectiveness of our proposed caching scheme with that of the whole data broadcasting.

The data set used for this comparison is the *Customer* table in the TPC-C benchmark. The *Customer* table consists of 30,000 records, and a record consists of 16 attributes. The size of a record is 686 in bytes. We set the communication rate as 14.4 Kbps, which is the data transmission rate of *IS*-95. We set the size of the predicate as 16 bits, which is based on the load factor 0.5 ($\sim \frac{30{,}000\ records}{2^{16}\ addresses}$) in the hashing scheme design.

Figure 1 shows the temporal delays for broadcasting all data and predicate-based cache invalidation information. As shown in the figure, broadcasting all data is not appropriate due to the long period of cache inconsistency.

Whole Data Broadcasting	Predicate-based Cache Invalidation
3 hours per broadcast	1 second per CIR (about 550 predicates)

Figure 1: Comparison of CIR Size

4 Conclusion

In this paper, we have proposed a cache management scheme for continuous partial match queries in mobile environments. To the best of our knowledge, there is no previous work on cache management for continuous partial match queries in mobile environments. Also, general caching methods in mobile environments are based on record identifiers, so they are not effective for partial match queries.

In the proposed scheme we represent the cache state of mobile client a predicate, and constructs the CIR with predicates and their timestamps. By comparison of the *cache predicate* of a mobile client with predicates in the P-CIR, the mobile client can investigate the consistency of its cache.

Acknowledgement

This work was supported in part by the Hacking and Virus Research Center.

References

[1] D. Barbara and T. Imielinski. "Sleepers and Workaholics : Caching Strategies in Mobile Environments". In *Proceedings of ACM SIGMOD Conference*, pages 1–12, 1994.

[2] Y. D. Chung, J. Y. Lee, Y. J. Lee, and M. H. Kim. "Predicate-based Cache Management for Continuous Partial Match Queries in Mobile Databases". Technical Report CS-TR-01-000, KAIST, Department of Computer Science, 2001.

[3] A. H. J. Jing and A. Elmagarmid. "Client-Server Computing in Mobile Environments". *"ACM Computing Surveys"*, 31(2):117–157, 1999.

[4] M. H. Kim and S. Pramanik. "Optimal File Distribution For Partial Match Retrieval". In *Proceedings of ACM SIGMOD Conference*, pages 173–182, 1988.

[5] D. Terry, D. Goldberg, D. Nichols, and B. Oki. "Continuous Queries over Append-Only Databases". In *Proceedings of ACM SIGMOD Conference*, pages 321–330, 1992.

[6] Y. Wang and L. A. Rowe. "Cache Consistency and Concurrency Control in Client/Server DBMS Architecture". In *Proceedings of ACM SIGMOD Conference*, 1991.