Journal of Computer Science 1 (3): 351-354, 2005
ISSN 1549-3636
© Science Publications, 2005

A New Semantic Cache Management Method in Mobile Databases

Shengfei Shi, Jianzhong Li and Chaokun Wang
School of Computer Science and Technology, Harbin Institute of Technology
Harbin 150001,People's Republic of China

Abstract: In mobile database systems, caching is a crucial way to improve the performance because a new query can be partially answered locally when the client is continuously moving around. Semantic caching has been paid much attention in traditional database systems. However, the methods of semantic cache management proposed in those systems are not suitable for mobile computing environment. Moreover, although many improved semantic caching strategies for mobile computing application have been proposed in recent years, few papers discussed semantic cache coherence control, which is very important in mobile computing environment. In this paper, we propose some new semantic cache model and management algorithms, which can significantly reduce the unnecessary uplink requests and downlink broadcasts compared to previous schemes. Simulation experiments are carried out to evaluate the proposed strategy.

Key words: Semantic cache, coherence management, mobile databases

INTRODUCTION

Semantic caching has been paid much attention in traditional database systems. In [3], a semantic model for client-side caching and replacement is proposed. Compared with the traditional caching strategies, such as page caching and tuple caching, semantic caching maintains a semantic description of the data in cache. And a distance function called Manhattan distance is proposed to determine which cache item to be discarded.
In [4], two issues are discussed. One is called cache completeness, which determines whether a new query can be satisfied in its local cache or not, and another one is called cache currency, which deals with the effect on client caches of updates committed at the database server. In [11], heterogeneous systems are discussed for semantic cache management.
In [1], a semantic caching scheme is used to access location dependent data in mobile computing. The emphases of the paper are put on modeling the mobility of the mobile client and the LDQ, and the method of cache replacement is proposed. The contribution of the paper is its FAR algorithm of cache replacement, which takes the mobility of the client into consideration.
In [5], a cluster-based approach is proposed to manage semantic cache. Queries are divided semantically into related or adjacent groups, which are called clusters. The cache replacement method, which deals with the semantic information of clusters, is more efficient than traditional cache replacement methods such as LRU etc. And many works such as in [12] also proposed approach to deal with the problem.
But up to now, few papers have discussed semantic cache coherence control, which is very important in mobile computing environment. Although many cache invalidation strategies for mobile environments have been proposed such as in [6,7,8,9,13,14,15], they have nothing to do with semantic caching. In [10], a novel cache invalidation report method is proposed and it turns out to be a semantic-based cache management strategy.
In this paper, we will address the problems associated with the IR-based semantic cache invalidation strategies. Firstly, we propose some new semantic cache model, PR-tree and TS-Cache. Secondly, some cache management algorithms which are based on these model are proposed. Finally, simulation experiments are carried out to evaluate the proposed strategy.

System Model
Hybrid Predicate Model: Suppose a relation which consists of n attributes: A_1, A_2, … A_n. We divide these attributes into two classes [1]. One is called LDA (Location Dependent Attribute), which can be used to identify the rectangle scope of the result of a LDQ (Location Dependent Query). Another class is called NLDA (Non Location Dependent Attribute), which is often used to specify the filter conditions of a query in a rectangle area restricted by LDAs.
In our proposed method we use the hybrid predicate representation. We use multi-attribute hashing function on the LDAs to represent the predicate scope. Suppose the LDAs are AX and AY representing the latitude and longitude coordinates respectively. Other NLDAs are denoted by A_3, A_4, … , A_n. Each LDA is hashed into string a_x and a_y by hash function H_x and H_y separately. Different from previously proposed method [10] we use additional string a′ to indicate the exact range value of a LDA in a predicate representation. For NLDA, we don't hash them into a string but keep their original value a_i instead. Then, the predicate for a record or a LDQ is denoted by "$a_x| a_y, a_x', a_y', a_3, \dots, a_n$", where '|' is the string concatenation operator. In a predicate, if an attribute is not included, we use character '*' to denote

"don't care" condition just like in [10].

PR-tree Based Global Predicate Caching Management Model at the Server

Definition 1: PR-tree (Predicate R-tree): PR-tree is a R-tree based index structure used by the server to index the predicate of LDQ committed by the client.
Leaf Node of PR-tree is:

LN={(R_LDA, μ_f, q_f, t_{last_update}, t_{last_query}),

where R_LDA is the rectangle section specified by the scope values of LDA in a LDQ, μ_f and q_f respectively refer to the update frequency and query frequency of area which R_LDA covers, and t_{last_update} and t_{last_query} are the latest update and query time of hashing area covered by R_LDA. Suppose the number of hashing sections covered by R_LDA is k, then

$$\mu_f = \left(\sum_{i=1}^{k}(n_i \cdot u_i)\right) \Big/ \sum_{i=1}^{k} n_i \ , \quad q_f = \left(\sum_{i=1}^{k}(n_i \cdot q_i)\right) \Big/ \sum_{i=1}^{k} n_i$$

where n_i is the number of data items in the hashing section i, and u_i and q_i are the update and query frequency of hashing section i respectively.

Definition 2: PU (Point Update Operation); In the PU operation, the server only updates one record.

Definition 3: RU (Range Update Operation); The server updates a set of records according to a range predicate.

Definition 4: HP-IR (Hybrid Predicate Invalid Report) is defined as a tuple **(HPR,**

Update Operation, TS, [Rid]), where HPR is the hybrid predicate representation. Update_Operation is PU or RU, TS is the timestamp of update operation, and Rid is the record id which is updated by PU and it is only used when Update_Operation is PU.

Predicate Caching Management Model at the Client

Definition 5: HPCD (Hybrid Predicate Cache Description) is defined as a set of tuple (id, HPR, p-content, UQ, Ts, Count_Update), where id is the identifier of the PCD item, HPR is hybrid predicate representation of the data records received from the server before, p-content is the pointer point to the real data buffer, UQ is a queue of some records to be retrieved from the server later; **Ts** is the timestamp of this PCD; Count_Update is the updating times of the cache content.

TS-Cache Model

Definition 6: The probability model of data is defined as PM_{odel} ::= (A_{ttr}, $L_{ocation}$, TS-M, CR, T_{stamp}). Where A_{ttr} is the attribute of data, $L_{ocation}$ identifies the location of data, TS-M is time-series model of data such as AR or ARMA; CR is confidence region of error bound between actual value of data and predicted value from TS-M with specific probability, and T_{stamp} is time stamp in which the TS-M is established.

Definition 7: The TS-cache model is defined as TS-Cache::= (Q_{uery}, PM_{odel}). Where Q_{uery} is the query which gets results from servers. PM_{odel} has been defined above and is used to answer the Q_{uery}.

Definition 8: Time-Series based Cache Invalid Report Model (TS-IR) is defined as TS-IR::= (A_{ttr}, $L_{ocation}$, TS-M|RawData, Δ_{error}, H_{ost}| $B_{roadcast}$, T_{stamp}). TS-IR is used to inform the clients to change the parameters of PM_{odel} when the error of predicted value exceeds the error bound predefined. A_{ttr} , $L_{ocation}$ and TS-M are the same as defined in PM_{odel}. RawData is raw datao of data source. Δ_{error} shows the difference between the data values which are calculated by old parameters and new parameters separately.

Algorithm of Coherence of Semantic Caching Management

Insertion Algorithm of PR-tree: After executing a LDQ, the server inserts the predicate of the LDQ into the PR-tree, in order to reduce the workload of the server, we only insert the LDA predicate, which limits the rectangle scope *R* in two dimensions geographic plane.

(I1) Compute the set of hashing sections covered by rectangle **R** and let it be S;
(I2) For each hashing section $s_i \in S$, modify its query frequency and the last time of being queried;
(I3) For each rectangle O_i of leaf node, which is overlapped by ***R***, modify the query frequency and its last time of being queried of the leaf node;
(I4) If R is covered by union of O_i, R will not be inserted into the PR-tree, and the server will return the result of the query and will attach the code "n_Covered" to indicate that the predicate of the LDQ is not inserted into the PR-tree but can be cached by the client;
(I5) Compute the μ_f and q_f of R. If μ_f >1/L or q_f <min_q, R will not be inserted into the PR-tree, and the server will return the result of the query and will attach the code "n_discarded" to indicate that the result of the query can not be cached by the client; The parameter L is the length of broadcasting window of the server, and min_q is the threshold defining the query frequency of a rectangle section composed of some hashing sections;
(I6) Insert R into the PR-tree, and return the code "n_cached" to indicate that the result of the query can be cached in the client.

Algorithm of Construction of HPIR Queue:
After an update operation is executed, the server will

judge which data items have been effected and construct the predicate invalid report to notify the clients who have cached these items to update its cache. Let the rectangle of the update operation be R.

(C1) Update the μ_f values of hashing sections and leaf nodes covered by R;
(C2) If update operation is a PU, find the leaf node which covers the point of PU. If it is found, then construct a new HPIR and insert it into the HPIR queue;
(C3) If the operation is a RU, find the leaf node which is overlapped by R. If find, then construct a new HPIR and insert it into the HPIR queue.

Algorithm of Coherence Management of Semantic Caching at the Client: When the client receives the result from the server, it checks the return code attached to the result to decide whether to insert the predicate of the query into the HPCD. If the return code is not "n_discard" the client will insert the predicate into the HPCD and cache the result of the query.
The most important issue of semantic caching management is the coherence of semantic cache.

Algorithm of Update of PCD

```
(1)  Listen to the HPIR broadcasting and download the
     HPIRs, which corresponding to the local HPCD;
(2)  Let $S_i$ be a HPCD item, and $UO_i$ be an update
     operation whose scope of LDAs is overlapped by
     that of $S_i$;
     If $UO_i$ is a PU
  BEGIN
  If the id of the record PU updated is already in the
     cache, then update the record locally;
  Else if all values of attributes updated by this PU
     satisfy the predicate of the PCD, then insert an
     item into the update queue UQ;
  END;
  Else BEGIN
```

Let the rectangle intersection of S_i and UO_i specified by the values of LDAs be *A*, and *A* is not empty. Let the intersection of predicate specified by the values of NLDAs of S_i and UO_i be *B*.
If *B* is not empty
Begin

Update all data records locally, which satisfy the predicate *A and B;*

Let all the records whose values of NLDAs satisfy the predicate: **(NLDA (RU)\B) and A,** be denoted by set ***C***, in which the records are not in the S_i.

If UO_i updates all the attributes in NLDAs of S_i and the modified values of NLDAs of a record in ***C*** satisfy the predicate of S_i, then

```
  Begin
     Insert a query into the UQ, the predicate of which
     is:  ((NLDA (RU)\B ) and A) and the record id is
     not in the set E. E is the set of record id of $S_i$;
  End
End
End
```

TS-Cache Coherence Management Algorithm Notation:

Client-Side:
Listen to the wireless channel to receive IR in time during connection time;

```
While(new data received)
{
  $\delta'_t = x_t - x'_t$
  if ($\delta'_t > \delta_t$)
    $N_{total}$ ++;
  if ($N_{total} > N_0$)
  {
     Compute the new parameters of data model;
     for (all neighbour hosts of MS)
       compute $W_i$;
     $W_s = W_i / \sum_{1}^{|neighbours\ of\ MS|} W_i$;
     Assign each neighbour host the task to broadcast the TS-IR $W_s * W_{IR}$ times;
  }
}
```

If (reconnected from disconnection mode)
Get IRs from server which overlap it now;
If (IRs received are related with its TS-Cache)
Choose the newest IR and update the parameters of the model;
Else
If (timestamp of local TS-Cache is older than the threshold predefined)
Iinvalidate local TS-Cache;

Performance Analysis and Simulation Results: We evaluate the effectiveness of our method in comparison with the approach proposed in [10] which is called PIR by us. At first, we introduce the main parameters of simulation model.

Number of clients	200
Database size	2000 records
Broadcast interval	10 minutes
Hot records	600 records
Hot data access prob	0.8

1. **The Mean Number of PIR Broadcasted:** From Fig. 1, we can see that the number of PIR grows as the update number increases. However, the growing trend is different between the PIR algorithm and our algorithm. The PIR method will construct a PIR whenever a record is updated, no matter whether the updated record is cached by clients or not. However, in our method, the server can determine whether to construct a new IR according to the PR-tree. On the other hand, if the updated records are not hot data, the probability of being cached by clients is low. In our algorithm, the server does not manage the predicate of cold data, so many updated cold data will not be broadcasted. As a result, the growing trend of the PIR number of our algorithm develops more

slowly than that of PIR algorithm.

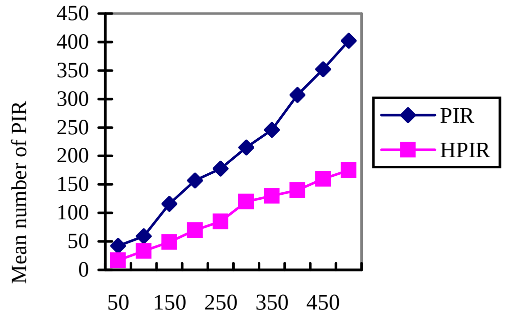

Fig. 1: Mean Update Number Per Interval

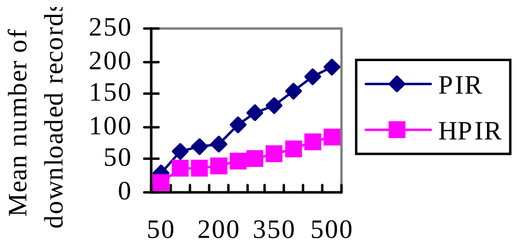

Fig. 2: Mean Update Number Per Interval

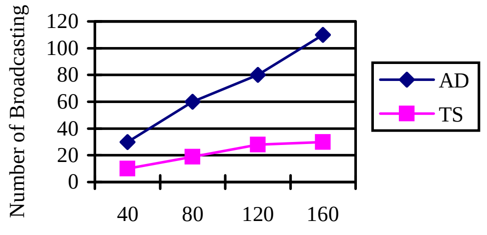

Fig. 3: Updated Hot Data

2. **Cost of Bandwidth for Downloading the New data:** The client will download some new data via requests to the server in order to make coherence of the cache, so some communication bandwidth will be used to do this. The Figure 2 shows the comparison between algorithm PIR and our algorithm as for the number of records which have been downloaded.

3. **Mean Communication Cost of Server:** From Fig. 3 above we can find that in our TS method the number of broadcast data is much smaller than that of AD. The reason is that the server in AD method must broadcast hot data which are updated. But in TS method, only the data whose error bound exceeds the threshold will be broadcast to clients.

CONCLUSION

In this paper, we propose some semantic cache mode and management algorithms which are based on PR-tree and Ts-Cache method. Simulation results show that our algorithm could save considerable bandwidth compared to the previously proposed algorithms. In future research, we will extend our algorithm to deal with the problem of transaction processing with semantic cache and we will extend our algorithm by using TS-Cache to deal with the problem of query optimize.

REFERENCES

1. Qun Ren, Margaret H. Dunham: Using semantic caching to manage location dependent data in mobile computing. MOBICOM 2000: 210-221
2. Seydim, MH.Dunham,V Kumar:Location Dependent Query Processing.MobeDe2001 USA 47-53
3. S. Dar, et al, "Semantic Data Caching and Replacement", *Proc. VLDB Conf. 1996.*
4. AM.Keller,Julie Basu :A predicate-based caching scheme for client-server database architectures.VLDB Journal(1996)5:35-47
5. Q Ren,MH.Dunham:Using Clustering for Effective Management of a Semantic Cache in Mobile Computing.MobiDe1999 WA USA,94-101
6. Guohong Cao: A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments.ACM MOBICOM2000 MA USA 200-209
7. D. Barbara and T.Imielinkai, Sleepers and workaholics: Caching strategies for mobile environments," ACM SIGMOD,pages 1-12,1994
8. T.Imielinksi, S.Viswanathan, and B. Badrinath, "Energy Effcient Indexing on Air", IEEE Transactions on Knowledge and Data Engineering, 9(3):353-372, May/June 1997
9. G. Forman and J. Zahorjan, "The Challenges of Mobile Computing," IEEE Computer, 27(6), April 1994
10. YD Chung et al: Predicate-based Cache Management for Continuous Partial Match Queries in Mobile Databases. KAIST CS Technical Report CS/TR-2001-163
11. P. Godfrey and J. Grys. Semantic query caching in heterogeneous databases. In Proceedings of KRDB at VLDB, pages 6.1-6.6, Athens, Greece, August 1997.
12. K.C.K.Lee, H.V.Leong, and A. Si. "Semantic query caching in a mobile environment". Mobile Computing and Communications Review, 3(2):28-36, April 1999
13. Distributed Caching and Broadcast in a Wireless Mobile Computing Environment
14. J. Jing, A. Elmagarmid, A. Helal, and R. Alonso, "Bit-Sequences: An adaptive Cache Invalidation Method in Mobile Client/Server Environments", Mobile Networks and applications, pages 115-127, 1997
15. Yin-Huei Loh et al: A Hybrid Method for Concurrent Updates on Disconnected Databases in Mobile Computing Environments. SAC (2) 2000:

563-565