Advanced Science and Technology Letters
Vol.66 (Networking and Communication 2014), pp.38-42
http://dx.doi.org/10.14257/astl.2014.66.10

A Study on the Improvement of Concurrency Control via SCL Techniques

Hui-Jeong Ju, Sung-Je Cho[2]

[1] Dept of Education at Dongbang Graduate University, dbaguru@daum.net
[2] Dept of Education at Dongbang Graduate University, chosj715@daum.net

Abstract. The purpose of this paper is to study a new concurrency control model for the improvement of performance via T-SCL techniques. The recent increase in the internet use via mobile devices has resulted in many studies on how to improve transaction-processing speeds. The traditional technique called T-SCL applies the T-tree indices, resulting in a problem with an increase of overhead in the verification of the entire coverage of concurrency control. Therefore, this study proposes a new concurrency control technique applying an index that enables faster processing of inquiries with an aim to increase overhead in the verification of the entire coverage.

Key words: T-SCL, CST*-tree, database, mobile, concurrency control, transaction model

1 Introduction

The recent universal dissemination and use of mobile smart devices has resulted in a rapid increase of the internet use including information search, electronic transaction and internet banking. As a result, real-time processing to ensure efficient transaction processing is more than required in the area of mobile communication application [1-3]. Therefore, the study attempted to develop a model to reduce transaction overhead in the verification of the entire coverage for concurrency control. The traditional technique, T-SCL model suggested a concurrency technique in the mobile environment. This technique applies the T-tree indices and produces higher overhead imposed on concurrency control. Therefore, the study proposes a new technique improved from the previous T-SCL model.

2 Related Studies

2.1 SCL Techniques

This technique reveals various problems arising from broadcasting servers using wireless communication networks of limited bandwidth and in the mobile client environment and suggests a positive concurrency control technique that has improved

ISSN: 2287-1233 ASTL

the bottleneck phenomenon [4]. However, the sequential files are composed of directories and this produces a problem with increasing overhead in the verification of concurrency control.

2.2 T-SCL Applied Mobile Concurrency Control Technique

With an aim to solve the problem of an increased overhead pertaining to the verification of concurrency control in the SCL technique, the T-SCL technique was suggested. The T-SCL technique pertains to positive concurrency control in the mobile computing environment and will be implemented without an interruption in the process of reading. If transactions do not collide later, Commit shall be ordered. If there is a collision, the issue shall be resolved for the maintenance of data consistency [5]. However, there still remains this problem of overhead that occurs in the real-time transaction verification.

2.3 Index Techniques

2.3.1 T-tree

T-tree is a tree structure combining AVL-tree consisting of binary search and height balance and B-tree consisting of various data within one node. T-tree is of a structure converted in a way to enable one node to contain n data with an aim to improve the problems such as frequent spatial waste pertaining to AVL-tree as well as rotational computation. All the data placed in the tree on the left of T-tree node shall be lower than the minimum values of nodes while those on the right of the node shall be greater than the maximum values of nodes [6].

2.3.2 T*-tree

T*-tree is an index structure suggested with an aim to supplement the processing of the scope query of T-tree. As for T-tree, a rear pointer that sequentially connects each of the nodes instead of one index entry will be added and via the rear pointer, the scope query will be efficiently processed. Just like T-tree, it contains the index entry in its internal node and there is no need to search up to terminal nodes. However, while T*-tree is more well-performing in terms of scope query by reducing unnecessary node rounds, T*-tree will not be so efficient in terms of the cache memory use due to the size of node, which is the same case applied to T-tree [7].

2.3.3 CST-tree

CST-tree is an index structure devised to reduce the frequency of cache miss arising from the sizes of T-tree nodes. CST-tree will have each node of the existing T-tree applied as a data node and use the maximum key value of each data node to newly

establish a binary tree. While T-tree will entangle cache miss each time one node is brought to the cache memory, CST-tree will apply the binary tree-based node blocks consisting of maximum values to the cache memory, which reduces the frequency of cache miss [7, 8].

However, CST-tree will derive searches via a binary tree consisting only of maximum key values and if you need to search a key value existent in any internal node, you will have to search every terminal node, which is inefficient. Also, as for a scope query such as in T-tree, it is inefficient as you will need to search the nodes of key values within the search range from route nodes each time [7, 8].

2.3.4 CST*-tree

CST*-tree is an index structure which has supplemented the problem with less efficient use of cache memory by T*-tree in consideration of scope of query and solved the inconvenience of T+-tree which necessitates a search up to terminal nodes to derive a key value existing in the tree as well as the higher tree height. CST*-tree will contract internal nodes by maintaining the minimum and maximum key values of internal nodes of T-tree while establishing internal index entries via internal index nodes so that the relevant pointers can be maintained. The internal nodes contracted like this will consist of node blocks of a size identical to that of cache blocks. They will be applied to cache memory to decrease the frequency of cache miss [7].

2.4. Comparison of Performance of Indices

The following is the formula-based calculation of the frequency of cache miss in the event of single-key searches: T-tree: $\sum_{i=1}^{n} \lfloor \log_2(t+1) \rfloor / \frac{n}{s}$, T*-tree: $\sum_{i=1}^{\frac{n}{s-1}} \lfloor \log_2(t+1) \rfloor / \frac{n}{s-1}$, CST-tree: $\left\lfloor \log_s\left(\frac{n}{s}(k-1)+1\right) \right\rfloor + 1$, and CST*-tree: $\left(\sum_{i=1}^{\frac{n}{s-1}} \left\lfloor \log_s\left(\frac{t}{2}(k-1)+1\right) \right\rfloor + 1\right) / \frac{n}{s-1}$. The frequency of cache miss was lower from higher in the order of CST*-tree, CST-tree, T-tree, and T*-tree. As for CST*-tree, the frequency of cache miss fell by 60-70% and 70-80% in the single-key search and the scope query, respectively, in comparison with T-tree. CST*-tree produced the lowest level of cache miss, resulting in the highest level of usage of cache memory [7].

3 CST*-tree Applied CST*-PCL Technique

With an aim to solve problems with the T-SCL technique, the study proposes the transaction model using CST*-tree (Cache Sensitive T*-tree: hereinafter referred to as CST*-tree). The suggested model is called CST*-Page Commit List (hereinafter referred to as CST*-PCL) and based on this technique, concurrency control will be applied.

3.1 Organization of CST*-PCL Technique

The CST*-tree applied CST*-PCL model will carry out a transaction limitlessly based on the hand-off characteristics in the smart phone network system and then, decide whether there has been a collision of Lock Mode via CST*-PCL. Also, there will be no interruption in the phase of reading based on positive concurrency control. By the point of completion, if there is no contradiction to collision between transactions based on the transaction phase out of CST*-PCL, execute Commit. If there is a violation based on this mode, solve the collision-related issues to maintain data consistency.

4 Concurrency Control Model

The concurrency control technique suggested in this paper was based on the positive concurrency control using the CST*-tree. The CST*-tree applied concurrency control model has been established as an LTE-A communication model between smart phone stations and smart phones. The CST*-tree applied positive concurrency control technique will result in less collisions between transactions, which makes it more efficient than any other locking-based techniques. Also, with an aim to transmit variously-sized multimedia data, page-based units, rather than the existing data item or segment-based units, were applied.

5 Conclusion

The recent rapid development of smart devices has resulted in active studies to explore transaction models pertaining to faster and more accurate data processing with smart devices. However, the traditional transaction model produces a problem that generates a lot of overhead. With an aim to solve this problem, the study suggested the CST*-PCL technique with overhead improved from the existing concurrency control technique. Based on the results of performance assessment, the CST*-PCL techniques improved significantly more than the existing techniques. The study aims further at recovery techniques using CST*-PCL in the future.

References

1. Kam-Yiu Lam, Tei-Wei Kuo, Wai-Hung Tsang and Gray c.k. Law, "Concurrency Control in Mobile Distributed Real-Time Database Systems", Information Systems, Vol. 25, No. 4, pp. 261- 322, (2000).
2. Michael J. Franklin , Michael J. Carey , Miron Livny, "Local Disk Caching for Client-Server Database Systems", Proc. VLDB, pp. 641-655, (1993).
3. Carey, M., Frankin, M., Liviny, M., Schekita, E., "Data Caching Tradeoffs in Client-Server DBMS Architecture", Proc. ACM SIGMOD Conf., Denver, (1991).

4. SungJe-Cho, "A Concurrency Control Method using Optimistic Control in Mobile Computing DB Environment", Journal of the Korea society of computer and information, vol.11, No.2, pp. 131-143, (2006).
5. Chang-Ho Suk, Sung-Je Cho, “T-tree : Mobile Concurrency Control Improvement using T-SCL”, Journal of the Korea Society of IT Services, Vol.10, No,8, pp.135-144, (2012).
6. Ig-hoon Lee, Snag-goo Lee, “Cache Sensitive T-tree Index Structure”, Journal of KIISE, Vol.32, No.1, pp.12-23, (2005).
7. Sang-Jun Choi, Jong-Hak Lee, "Cache Sensitive T-tree Main Memory Index for Range Query Search", Journal of Korea Multimedia Society, Vol.12, No.10, pp.1374-1385, (2009).
8. Dae-hee Kang, Jae-won Lee, Sang-Goo Lee, "Efficient range search of CST-tree", Journal of Computing Science and Engineering, Vol.33, No.1, pp.67-69, (2006).

