Mining Blocks' Association Rules for Disk Data Perfecting

College of Computer and Information Science
Southwest University Chongqing
China

*Corresponding author

Abstract—as the processor-I/O gap continues widening, the modern I/O-bound applications such as Online Transaction Processing (OLTP) often suffer from great latency of disk access. In this paper, we argue that the block association rule can be recognized as common semantic information embedded in disk I/O traces and it can be utilized to direct disk prefetching. We propose a novel mining approach to extract such association rules to benefit prefetching. The experimental results show that this proposed approach is effective in increasing prefetching accuracy, and the disk latency can be reduced by this association rule based prefetching.

Keywords-association rule; prefetching; storage system

I. INTRODUCTION

As the speed gap between processor and disk continues to grow, the disk-based storage system is becoming the bottleneck of data-intensive application system. We are now allowed to apply much more delicate methods to the storage system to attack the problem. Prefetching is a common solution to mask the I/O latency. This approach hides the latency with early I/O initiation by utilizing the relatively redundant computational resource, as well as efficient and accurate prediction algorithm.

In general, there are mainly two fundamental perfecting techniques [1] [2]: heuristic and speculative schemes. The later approach 'pre-executes' fragments of code in the targeted process via an extra helper thread [3] [4] [5]. The speculative execution thread only keeps track of I/O related operations and identifies future I/O references. This kind of approach generally achieves high accuracy and can be more suitable for applications whose access patterns are complex or random. However, it usually involves source code transformation, prefetching injection and hardware support. On the other hand, the heuristic scheme predicts future access based on patterns obtained by analyzing past access history (or so-called trace) [5] [6] [7]. This approach can be implemented without hardware support, and has no need to touch the application.

In response to all these observations, we propose a novel association rule mining approach used for heuristic prefetching scheme. It is designed for improving the I/O performance for complex workloads. Our mining approach solely targeting at conducting prefetcher, outputs the association rules that can be directly used to predict future data access. The mining approach is optimized in order to extract a small number of necessary rules.

II. EXPLOITING BLOCK CORRELATIONS AS SEMANTICS

A. Block Correlations

The essential idea of block correlations is based on the fact that the data processed and stored are somehow related with each other by the semantics they contain. Two blocks can be considered 'linked' if the data stored by these blocks is correlated. However, data semantics can be much more complex and rich, thus making the block correlations hard to detect. Consider the correlation involves blocks that store indexes and data the indexes point to in database systems. It is both depended on high-level semantics of the data itself and the data structures the application defines. Therefore it is difficult to derive a general yet effective mathematical description of these kinds of correlations.

Association rule [10] is a straightforward expression of block correlation and can be directly used to predict future block reference. These rules describe a phenomenon that certain items occur with a high frequency after an occurrence of other specific bunch of items. If we discover that block 5 is always requested following block 1, 2 and 3, then we obtain one association rule represented as {1,2,3}→{5}, where the left hand {1,2,3} is called antecedent and right hand is called consequent. We only need to mine one-item-consequent rules here.

B. Time Constrains

In order to constrain the search space, time constrains are applied to the association rules. It is obvious that those rules are of less use even when they are indeed frequent and highly confident, if whose antecedents have a long time span or there is a huge time lag between the two antecedents and consequents.

Formally, a time window w on a block sequences is represent as w[Ts, Te], where Ts and Te are start time and end time of window w. Usually we constrain the width of the searching window so that only closely occurring sequences are searched. Besides, the consequent is required to occur in a limit period of time right after antecedent stops, which means if the antecedent stops at time t1, and the maximum time lag is defined as tl, then the consequent should occur within the window of [t1, t1+tl].
III. IMPLEMENTATION OF MINING AND PREDICTION

A. Preprocessing

There are three tasks for this stage. First, it is required that block references are indexed in a single unified-space, which means we can only access only one disk. However, many common publicly available disk traces consist of an array of multiple disks. Each block reference in these traces is indexed by two variables, disk number and block number in that disk. Thus, a transformation is needed to map the two dimensional indexing space to the unified one dimensional space.

Second, there may be two or more blocks occurring at the same timestamp in some traces. At this stage, the timestamp of one of the blocks with identical timestamp is added with a tiny value to differ from another one.

Third, the original long trace is cut up into smaller pieces in fixed length. And each small piece of trace is one by one mined in the first mining stage.

B. Mining all One-Item-Antecedent Rules

The first mining procedure is to extracting one-item antecedent rules. The algorithm repeatedly scans each piece, records all qualified rules for each scan, and then merge the rules with those recorded in the previous scans. The following pseudo-code shows the algorithm.

```
Algorithm 2 expand(r)
Input: rule r to be expanded
Output: rule set rs, return 0, if rule r or one of more of its descendants saved. Return 0 otherwise.
1 rule set tmp_rs ←{} 
2 for each instance i of r 
3 ts, te refer to timestamp of antecedent's start/stop, and tc is timestamp of consequent's item; searching window w[te-ts, min(te+tw, k)] 
4 for each block b in w, such that b>max_Lmr(r) and rule(b>cons(r)) is included in rule set produced by last stage 
5 Call update(tmp_rs, rule( [ante(r),b] ) > cons(r)) 
6 end for 
7 end for 
8 flag ←0 
9 for each candidate rule rc in tmp_rs such that supp(rc) > min_support 
10 if conf(rc) > cutoff_conf 
11 put rc into rs and flag ←1 
12 else if conf(rc) > conf(r) and expand(rc) == 1 
13 flag ←1 
14 end if 
15 end for 
16 if flag=0 
17 if conf(rc) > min_conf 
18 put rc into rs and flag ←1 
19 end if 
20 end if 
21 return flag 
```

Function update () does the same task as the one in previous line and conf(rc) denotes the confidence of rule r. The variable tw is pre-defined maximum searching time windows described in section 2.3. This algorithm consists of two parts. The first part collects all possible candidate blocks in a limited search space. Notice line 9, It only picks up blocks with greater numbers than those already included in rule r’s antecedent so that it is guaranteed that no duplicate blocks are brought in. The second constrain is based on following lemma: if rule [a] > [b] is infrequent, none of the rules whose antecedents consists of item a and share the same consequent are frequent.

In the second part, the algorithm decides whether continue searching and whether save current rule r. Only the candidates rules that have higher confidence than their ascendant but not higher than cutoff_conf appears in line 10 should be worth further mining, so that all rules left in rule set rs are at their highest confidence and searching space is pruned as another benefit.

IV. EVALUATION

A. Methodology

To evaluate the performance of our approach, a trace-driven simulator is implemented. It consists of a cache, interfaces of replacement policy, prefetcher and DiskSim. When the simulator reading the disk trace, it maintains the cache and provokes replacement policy and prefetcher. The actual disk access generated by the simulator is forwarded to DiskSim [11], which simulates accesses being handled by real disk system.
We select the following traces for our experiments:

1) TPC-C trace is collected on a storage system backing a Microsoft SQL Server. The trace is generated during the clients running TPC-C benchmark for 2 hours. 2) OLTP trace is offered by Storage Performance Council. The trace samples 1.5 hours of I/O activity of an OLTP application running at a financial institution. The storage system is composed of 19 disks. 3) Kernel trace is collected by the authors during compiling Linux kernel. The compiling computer runs a modified Linux kernel, which intercepts VFS’s routines and records every disk block references. The total time is about 48 minutes.

We only use the first half part of trace to mine association rules. And evaluate the performance of prefetcher directed by the rules using the second part. We also implement sequential prefetching algorithm based on the same idea adapted by readahead [8] in Linux kernel. All experiments use Least Recent Used (LRU) cache replacement policy.

B. Constrains’ Impact on Performance

Figure 5-1 and 5-2 show the effects of hit ratio by two constrains, minimum confidence min_conf and maximum time searching window tw. To evaluate the performance of our approach, we measure the response time produced by the simulators without prefetching and with sequential prefetching, and figure 5-3 compares the results.

When min_conf decreases, more association rules satisfy the criteria. We can see the ratio increases slowly when confidence is small and finally stop rising at some point. This is because when the criterion is low, too many low quality rules are kept. And as it goes higher than certain points, although the prefetched blocks are highly possible being hit, the chance of triggering prefetching becomes rare and stabilized.

Constrain tw denotes the maximum time window allowed, which limits the searching space and the number of items of antecedents. As figure 1 and 2 show, although all traces share the same tendency that wider the windows lead to higher the hit ratio, the hit ratios themselves respond to tw differently. There are steep rise and stabilizing two stages for TPC-C and OLTP hit ratio, while in kernel trace it increases steadily. With this prior knowledge of the targeted workload, we can tune the mining algorithm faster.

Figure 3 shows response times produced by the three traces. In each bar group, the left bar is simulated with none prefetcher, middle bar is with sequential prefetcher, and right one is with our association rule guided prefetcher.

Our prefetcher can effectively decrease the response time over these three traces, especially on TPC-C and OLTP, considering the measurements decrease by 8%-10% comparing with the experiments with none prefetching. We observe that when simulating using sequential policy on TPC-C trace, the response time even goes higher than the one with none prefetcher. Although its performance can be promoted by tuning the policy, the response time cannot decrease as much as our approach achieves.

V. Conclusion

In this paper, we have proposed an association rule mining approach designed for the prediction purpose for storage systems. We have discussed the idea of using the correlation information of I/O access events in the block-level to direct the prefetcher operating for complex workloads. Therefore, an algorithm for extracting association rule is proposed to contribute to prefetching. The evaluation experiments on real workloads have shown that our approach can efficiently discover a set of accurate prediction rules at an acceptable cost, as a consequence, the performance of the storage system can be promoted by the association rule based prefetching.

ACKNOWLEDGEMENT

In this paper, the research was sponsored by National Key Technologies Research and Development Program of China (Project No. 2013BAD15B06).
REFERENCE


[9] Yan, Xifeng, Jiawei Han, and Ramin Afshar. "CloSpan: Mining closed sequential patterns in large datasets." Proceedings of SIAM International Conference on Data Mining. 2003.
