複数入力を持つ拡張 R-tree 検索アルゴリズムを用いた PUB/SUBシステムの平均応答時間の改善

Adaptively Improving Average Response Time of Pub/Sub System Based on Extended R-Tree Search Algorithm with Multiple Inputs

1. Introduction

Efficient event filtering with a faster response time is very important for event processing with multiple steps like event join, and is one of important factors to provide good service for subscribers. Generally the rate of event arriving is time varying and unpredictable. For example, traffic monitoring, ticket reservation, internet access, stock price, etc. In contrast to stable rate, it's very possible that a batch of events arrive in one unit time and no event arrives during another unit time.¹

In the context of publish/subscribe system, even many index techniques such as multiple one-dimensional indexes based [1], [2], [3], [4], [5] and multidimensional index based [6], [7] have been proposed for event filtering, all these techniques are designed for events filtering one by one. They can not deal directly with batch events in a fast average response time if those events arrive at same time with different workloads. Meanwhile, we found that event filtering based on multidimensional index [6], [7] is more efficient and flexible than that based on multiple one-dimensional indexes. In order to improve average response time of event filtering in the case that multiple events arriving at the same time, in this paper, we first propose a R-tree based event filtering algorithm to improve average response time of filtering multiple events arriving at same time. There a cost model to estimate relative workloads of these events is built to arrange the filter order of these events with Short-Job First (SJF) policy. Further, because the number of events arriving at the same time and index size change dynamically, an adaptive model is proposed to filter events with average response time always same as or close to the possible best time.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation. Section 3 introduces the algorithm to improve average response time. Section 4 proposes the adaptive model. In Section 5, the event filtering algorithm and the adaptive model are evaluated and analyzed in a simulated environment. Section 6 discusses the related work. Finally, conclusion and future work are given out in Section 7.

2. Background and Motivation

As introduced in [6], [7], multidimensional index (R-tree or UB-tree) based event filtering is feasible² and is much efficient and flexible than that based on the popular multiple one-dimensional indexes based technique - count algorithm [3], [4]. Meanwhile, SJF is one well-known policy used to improve average response time while scheduling multiple jobs. The critical thing is to estimate workloads correctly.

¹ Even logically for most of the events, there exist absolutely different arriving times, in this paper, we regards the events arriving in the same unit time as the events arriving at same time.

² For details, please refer to [6], [7].
Because UB-tree partitions space with space filling curve, original UB-tree's search algorithm is depth-first and no Minimum Bounding Rectangle (MBR) information is required and kept inside its nodes. It's not easy to use MBR of event to calculate the number of multiple search paths at specified middle levels of original UB-tree without accessing leaf nodes. The structure of R-tree doesn't have this problem. For this reason, we choose R-tree as the basis of our proposal in this paper.

3. Improve Average Response Time

3.1 Basic Idea

The basic idea is that for the multiple events arriving at same time, the relative workloads are estimated respectively. The workload is defined as the number of search at a specified level. There are two steps for the multiple events filtering algorithm (called BatchSearch later) 1) first based on their different numbers of search paths on R-tree by search R-tree to a specified level 2) and then do filtering (R-tree search from the specified level) event by event with SJF policy with assumption that the more the number is, the higher the workload is. Because the algorithm used at step 2 is similar to original R-tree search algorithm, we introduce only the data structure and algorithm to estimate the workload in step 1. We assume the reader has enough background about R-tree.

3.2 Data Structures to Estimate Workload

WorkloadTable is an array of items with structure shown in Fig.1. Each item corresponds to one event. The Workload in Fig.1 is the number of nodes located at the ending of search paths stopped at the specified level. WorkloadTable is filled and sorted by function EstimateWorkload as introduced as Fig.2.

A data structure named IntersectBuffer Fig.2 is used to record events whose MBRs intersect with those of items of one R-tree node which is located in the paths from root to the specified level.

In function BatchIntersect, line 1-2, read one item from IntersectBuffer of last level (the level nearer to root) and gets all event IDs kept in the item. That item corresponds to the item of R-tree node at last level which includes the pointer pointing to CurrentNode. Line 3 checks the ending condition of recursive search and line 4 adds the WorkloadTable with the event IDs gotten at line 1-2 and CurrentNode. Line 6-15 fill intersect buffer of CurrentLevel. Line 16-18 search next level by accessing subnodes of CurrentNode.

4. Model of Adaptive Search

For same batch events, the performance changes with different possible values of R-tree Level. At the same time, the number of events arriving at same time is not fixed, the size of index changes dynamically also. In this section, we will propose a self-adaptive model in order to filter kinds of events with average response time same as or close to the possible best time.

4.1 Performance Analysis

While filtering multiple events arriving at same time, time cost to estimate workloads is overhead compared to the processing with original R-tree search algorithm event by event without the workload estimation. The overhead becomes larger with the value increment of Level. At the same time, because the higher the Level is, the more accurate of workload estimation is, the efficiency of SJF become more and more better with the value increment of Level also. For the same batch events, the average response time based on the BatchSearch is a function of Level L. Their relationships can be described in the left part of Fig.3 because it has shape of concave as shown
論文

このはあくまでサンプルですね ##

on the leaf of Fig.3 with mark "Ideal and logical". The best level exits for the batch events with same event number and it should be located between level 0 to level TreeHeight-1. The best level changes for different number of input events and size of index.

The evaluation results are shown in Fig.4. Fig.4-a shows that the best level changes slowly with increment of index size. Fig.4-b compares the average response time of BatchSearch algorithm to that which just inputs events to original R*-tree in a random sequence. Further, it shows that the cost to estimate workload (algorithms shown in Fig.1 can be neglected compared to average response time. It also shows that the larger the size of input is, the more the average response time can be improved. The maximum is nearly up to 50% in our evaluation.

Fig.4-c and Fig.4-d show the changing of response time which are calculated with same and different events gotten at different levels. There the index size is 1.5 million and the height of tree is 7. It shows that with increment of loop counter, the trembling of response time marked "Unideal and practical" in Fig.3, is captured.
clearly in Fig.4-c (loop=4 or loop=16). Different events, becomes more and more stable with shape of ideal concave and merge into the line gotten with the same events at each level.

The Fig.4-d shows the performance of unstable status compared to the performance without using the adaptive model (same as the "No BatchSearch" performances shown in Fig.4-b) and the possible best performance. There, the size of index changes from 0.5 million to 2.6 millions, the Threshold is 300,000, and the loop counter is 64. When the system becomes stable, 300,000 objects (subscriptions) are inserted into the index.

So the left part of Fig.5 shows the performance of unstable status.

We can find that performance with the adaptive model is much better than the performance without the adaptive model, the performance differences are almost at same level as those shown in Fig.4-b which are gotten at stable status. The performance of the adaptive model even in unstable status is very close to the possible best performance as shown in Fig.5-d. We can say that with the adaptive model, the arrays of events can be filtered with response time close to the possible best time. The difference can be neglected compared to the performance without adaptive model.

6. Related Work

A lot of algorithms related to event filtering have been proposed. They are proposed for publish/subscribe systems [2], [4], [5], [6], for continuous queries [1] [8] [9] and for active database [3].

Predicate indexing techniques have been widely applied. There, a set of one-dimensional index structures are used to index the predicates in the subscriptions, the representative algorithm is called counting algorithm [4] and Hanson algorithm [4]. They differ from each other by whether or not all predicates in subscriptions are placed in the index structures.

In [6] and [7], multidimensional index based event filtering is proved to be feasible and efficient. It's the basis of this paper.

Event filtering is one critical step of continuous queries. In [1], predicate index is built based on Red-Black tree, there algorithm is similar to bruteforce that scans the Red-Black tree for event filtering each time. Count algorithm was used in [8], [9].

[9] implemented routing policies to let faster operators filter out some tuples before they reach the slower operators. In [10], queries are optimized based on rate of input.

The problem of multiple events arriving at same time with different workloads is not considered in above techniques.

7. Conclusion

In this paper, for pub/sub system, we first proposed an event filtering algorithm with multiple events as input based on R-tree. Further an adaptive model is designed to filter multiple events for different event numbers and changing index size. According to the evaluation results, the average response time can be improved maximally up to nearly 50% with our algorithm and the adaptive model can work well with average response time same as or close to the possible best time in both stable and unstable statuses.

Research Fellow at Institute of Industrial Science, the University of Tokyo. He received the Ph.D degree in computer science in 2000 from Kyushu University. His research interests include spatial-temporal database, parallel processing and data stream. He is a member of DBSJ.

张 旺 研究生
Ph.D student of Graduate School of Information Science and Technology, the University of Tokyo. He received the Master degree in information engineering in the above graduate school. His research interests include data clustering and data stream. He is a student member of DBSJ.

喜连川 優 教授
Professor and the director of center for information at Institute of Industrial Science, the University of Tokyo. He received the Ph.D degree in information engineering in 1983 from the University of Tokyo. His research interests include parallel processing and database engineering. He is a member of steering committee of IEEE ICDE, PAKDD and WAIM, and has been a trustee of the VLDB Endowment. Now he servers as general chair of ICDE2005. He was the chair of data engineering special interest group of Institute of Electronic, Information, Communication, Engineering, Japan, the chair of ACM SIGMOD Japan, Chapter. He is currently a trustee of DBSJ.