

An Optimization of Apriori Algorithm through the Usage of Parallel I/O and Hints

María S. Pérez¹, Ramón A. Pons¹, Félix García², Jesús Carretero² and María L. Córdoba¹

¹ DATSI. FI. Universidad Politécnica de Madrid. Spain

² Departamento de Informática. Universidad Carlos III de Madrid. Spain

Abstract. Association rules are very useful and interesting patterns in many data mining scenarios. Apriori algorithm is the best-known association rule algorithm. This algorithm interacts with a storage system in order to access input data and output the results. This paper shows how to optimize this algorithm adapting the underlying storage system to this problem through the usage of hints and parallel features.

1 Introduction

Association rules are very useful and interesting patterns in many data mining scenarios. These patterns represent features that “happen together” in event-recording logs. Their very best known application is what is known as “basket analysis”, a generic problem that deals with the identification of the products a client gets in the same purchase. The “basket analysis” defines a clear pattern-recognition task, which is applicable to many other problems such as network failure prediction, genome analysis or medical treatments, to name a few.

An association rule is an implication of the form $A \rightarrow B$, where A and B are sets of data features called “items”. These itemsets have no common elements ($A \cap B = \emptyset$) and they are non-empty sets ($A \neq \emptyset \wedge B \neq \emptyset$). The rule $A \rightarrow B$ is read as “if an instance has the feature A then it also has the feature B ”.

Apriori algorithm [4] is the best-known association rule algorithm. It has two different phases: (i) frequent itemsets calculation and (ii) rule extraction using these frequent itemsets. The first of these two phases becomes a performance “bottleneck” in all the Apriori algorithms. In the first phase it is necessary to access to the storage system for getting a huge amount of data which must be processed. I/O system access is slow and traditional systems do not increase I/O operations performance. Parallel I/O was developed in an attempt to face up to this problem, aggregating logically multiple storage devices into a high-performance storage system. Parallel I/O allows applications to access in parallel to the data, providing better performance in the operations of the file system.

This work presents an optimization of Apriori algorithm, making use of MAPFS (Multi-Agent Parallel File System) [2], a high-performance parallel file system, which distributes the data and provides parallel access to files data, what reduces the bottleneck that constitutes the accesses to conventional servers.

Apriori algorithm is lightly modified in order to use the MAPFS interface. The storage system provides parallel access and makes decisions about the frequent itemsets calculation, reducing the candidates in the first phase of the algorithm.

This paper is organized into four sections. Section 2 describes our proposed solution. Section 3 shows the solution evaluation. Finally, section 4 summarizes our conclusions and outlines the future work.

2 Association Rules Calculation using Parallel I/O

In [3] the relation between Data Mining applications and the underlying storage system is shown, proposing MAPFS system as a suitable framework. This paper shows how to use MAPFS in order to increase Apriori algorithm efficiency.

2.1 Problem Scenario

Data mining processes require multiple scans over databases. An important characteristic is that these scans are not always complete. Induction tree construction or itemset calculation (in an association extraction algorithm) eliminates records on early stages of the algorithm that are not required in following iterations.

If the behaviour of a data mining algorithm is known, I/O requirements could be achieved in advance before the algorithm actually needs the data. An optimal usage of computational resources, like I/O operations or disk caches is also a key factor for high-performance algorithms.

Other kind of algorithms start with the complete set of attributes and during the multiple iterations this set is reduced by several elements. If these attributes are marked as removed, an interesting strategy could be skip this positions on further readings. This is the technique used by our proposal. The storage system is responsible for discarding data blocks whose items are not suitable candidates for the following steps in the algorithm.

2.2 I/O Model

This section describes the I/O model we propose. This I/O model is based on the MAPFS (Multi-Agent Parallel File System) I/O model [2], but adapted to the problem described previously.

Although MAPFS is a general purpose file system, it is very flexible and it can be configured in order to adjust to different I/O access patterns. The main configuration parameters of the MAPFS file system are the following: (i) **I/O Caching and Prefetching**, data items from secondary storage are “cached” in memory for faster access time. This approach allows to increase the performance of the I/O operations. (ii) **Hints**, the previous tasks can be made in a more efficient way using hints on future access patterns. Storage systems using hints provide greater performance because they use this information in order to decrease the cache faults and prefetch the most probable used data in the next executions. Analogously, the hints can be used for increasing the performance

of file processing because additional information related to data is stored, what can help in making decisions about them. This is the way in which we have to use the hints for increasing Apriori algorithm performance.

2.3 Apriori Algorithm: A Case of Hints Optimization

Apriori algorithm is used in order to build significant association rules between items in databases.

For optimizing Apriori algorithm, we have used the Christian Borgelt's implementation that uses a prefix tree to organize the counters for the item sets [1]. This implementation uses flat-file as input data and output files containing rules together with the confidence and support of the association rule.

Our optimization is made at file system level. The storage system discards for the iteration i those data blocks that are not members of the candidates in the level $i - 1$.

Christian Borgelt's implementation reads all the input file lines in every iteration. In order to increase this implementation performance, the chosen block is one line per input file. The storage system uses hints that define if a block (a line) must be evaluated in the next iteration or not. The blocks are marked as removed if they are not selected in the previous iteration. If a block is marked as removed, the file system skip this block and the input data is reduced.

We have modified Christian Borgelt's implementation in two aspects: (i) the new implementation uses MAPFS I/O operations; (ii) also, it has to interact with MAPFS in order to communicate the storage system that an itemset belongs to candidates. If the storage system is not said that any itemset belongs to candidates during a line processing, this block is marked as removed.

3 Apriori Algorithm Evaluation

This section shows the evaluation of our proposal, which consists of the optimization of two aspects: (i) parallel access to the data; (ii) usage of hints for discarding data blocks.

In order to evaluate our optimization, two parameters have been analysed: (i) time for the algorithm conclusion with the usage of hints and (iii) comparison between the parallel version and sequential version. We have calculated these parameters varying the data support.

For making the evaluation, we have used an input file with size 1000 MB. and the results have been calculated as the average value of five executions, in order to avoid noise. The figures 1 and 2 represent these evaluations.

As it is shown in the graphics, the usage of hints and parallel features increases the algorithm performance.

4 Conclusions and Future Work

Apriori algorithm interacts with a storage system in order to access input data and output the results. This paper evaluates our Apriori algorithm optimization,

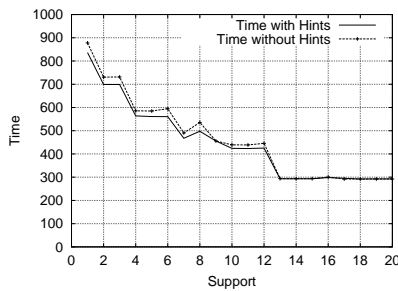


Fig. 1. Performance Comparison with the usage of hints in the parallel version

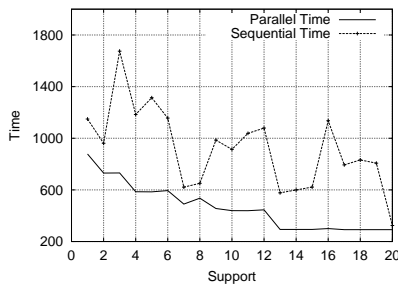


Fig. 2. Performance Comparison of the Parallel Version versus Sequential Version

in which the underlying storage system makes decisions about the input data. This proposal improves the Apriori algorithm global time by means of both parallel access and hints usage, taking into account that the selected implementation of such algorithm is very optimized.

As future work, this technique could be applied to other algorithms with the same structure, that is, which need to filter some of their input data. Furthermore, it would be desirable to build a generic framework for adapting this I/O model to different algorithms.

References

1. Christian Borgelt's Homepage <http://fuzzy.cs.uni-magdeburg.de/borgelt>.
2. Maria S. Perez et al. A New MultiAgent Based Architecture for High Performance I/O in clusters. In *Proceedings of the 2nd International Workshop on MSA'01*, 2001.
3. Maria S. Perez et al. A Proposal for I/O Access Profiles in Parallel Data Mining Algorithms. In *3rd ACIS International Conference on SNPD'02*, June 2002.
4. Rakesh Agrawal et al. Mining Association Rules between Sets of Items in Large Databases. In *The ACM SIGMOD International Conference on Management of Data*, 1993.