PARALLEL MINING OF FREQUENT ITEMSETS USING MAP REDUCE AND FIDOOP

[1]J. Sree Subhashini, [2]V. Bakyalakshmi
[1,2]Sri Jayendra Saraswathy Maha Vidyalaya, College of Arts and Science, Coimbatore, India
[1]subhachandran91@gmail.com, [2]vbakyalakshmi@yahoo.co.in

Abstract: FI Hadoop uses parallel mining algorithms on the basis of I/O overhead, data distribution, storage, scalability, load balancing, automatic parallelization and fault tolerance. On the basis of comparisons done, we get the most efficient parallel frequent itemsets mining algorithm i.e. FiDoop using FIUT and Map Reduce programming model. Compared to related work, FIUT has four main metrics. First, it minimizes I/O overhead by scanning the database twice. The second metrics is FIU-tree an improved way to partition a database, which results from clustering transactions, and significantly reduces the search space. Third metrics is only frequent items in each transaction are inserted as nodes into the FIU-tree for compressed storage. The last metrics is all frequent itemsets are generated by checking the leaves of each FIU-tree, without traversing the tree recursively, which significantly reduces computing time.

Keywords: *Load mining, Automatic parallelization, Scalability, Clustering, Frequent itemsets, Tree recursively, computing time.*

1. INTRODUCTION

1.1 data mining

Data as an abstract concept can be viewed as the minimum level of abstraction, from that information and then knowledge are derived. Raw data are unprocessed data, refers to a collection of numbers, characters and is a relative term; data processing usually happens by stages, and the "processed data" from one stage may be considered the "raw data" of the next. Field data refers to raw data that is collected in an uncontrolled in suitable environment.

1.2 Frequent itemsets frequent

This proposed algorithm is used to dynamically create an optimal schedule to finish the submitted jobs in a High Performance Computing environment showing promising results which achieves significantly lower resource usage costs for the jobs. The resource management techniques include cost-aware resource provisioning, VM aware scheduling and online virtual machine reconfiguration.

2. DATA ANONYMIZATION

Privacy is one of the most concerned issues in data publishing. Personal data like electronic health records and financial transaction records are extremely sensitive although that can be analyzed and mined by organization. Data privacy issues need to be addressed urgently before data sets are shared. Data anonymization refers to as hiding sensitive data for owners of data records. Large scale data sets are generalized using two phase top-down specialization for data anonymization.

1 k-Anonymity

The k-anonymity is to shield a dataset against re-identification by simplifying the attributes that might be utilized in a linkage attack (quasi identifiers). An information set is taken into account k-anonymous if each information item not differentiated from a minimum of k-1 alternative data things.

l-Diversity

l-diversity could be a variety of cluster based generally anonymization that's wont to preserve privacy in knowledge sets by minimizing the coarseness of a knowledge representation. This reduction may be a tradeoff that ends up in few loss of efficiency of knowledge management or mining algorithms so as to achieve some privacy. The l-diversity model is associate degree extension of the k-anonymity model that minimizes the roughness of information illustration

victimization techniques with generalization and suppression specified any given record maps onto a minimum of k alternative records within the data.

t-closeness

t-closeness could be an additional refinement of l-diversity cluster based mostly anonymization that's accustomed preserve privacy in knowledge sets by reducing the coarseness of an information representation. t-closeness could be an extra refinement of l-diversity cluster primarily based anonymization that's wont to preserve privacy in knowledge sets by reducing the coarseness of an information illustration

3. TREE PARTITION BASED PARALLEL FREQUENT ITEMSETS MINING ON SHARED MEMORY SYSTEMS

The main idea is to build only one FP-Tree in the memory, partition it into many independent parts and transfer them to various threads. A heuristic algorithm is devised to balance the workload. Our algorithm can not only alleviate the impact of locks during the tree-building stage, but o avoid the overhead that do great harm to the mining stage. We present the experiments on different kinds of datasets and compare the results with other parallel approaches. This approach results is great advantage in efficiency is achieved, especially on certain kinds of datasets. As the number of processors increases, our parallel algorithm shows good scalability. Association rule mining searches for interesting relationships among items in a given data set. One of the most famous examples of association rule mining is the market basket problem.

However, as for extremely large datasets, the currently proposed frequent pattern mining algorithms still consume too much time.

- One solution is to design more efficient mining algorithms to reduce the repeated I/O scans as well as to minimize the memory requirement and calculating time. So algorithms like kDCI , FPGrowth , etc, are proposed
- Another alternative solution is to parallelize the algorithm. The present frequent pattern mining algorithms can be divided into two categories: apriori-like algorithms, which are the implementations of the classical apriori algorithm, and the other ones, which are completely different in structure with the classical apriori algorithm. Among both kinds of algorithms, FP-Growth can achieve good efficiency. It's faster than any of the apriori-like algorithms and it only has to scan the whole database twice.

FP-Growth algorithm is based on tree structures.

The algorithm can be divided into two steps.

- Building FP-Tree Algorithm

FP-tree construction

Input: A transaction database DB and a minimum support threshold ξ.

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is derived steps are given below.

One time transaction database DB is scan. Collect F which is the set of all frequent itemsets, and the support of each frequent item. Sort F in support-descending order as F-List, the list of frequent items

Create the root of an FP-tree, T , and label it as ―null‖ for each transaction Trans in DB, do the following

FiDoop

In light of the MapReduce programming model, we design a parallel frequent itemsets mining algorithm called FiDoop. The design goal of FiDoop is to build a mechanism that enables automatic parallelization, data distribution and load balancing for parallel mining of frequent itemsets on large clusters. To facilitate the presentation of FiDoop, we summarize the notation used throughout this paper in Table I. Aiming to improve data storage efficiency and to avert building conditional pattern bases, FiDoop incorporates the concept of FIU-tree .

4. METHODOLOGY

- First MapReduce Job

The first MapReduce job is responsible for creating all frequent one-itemsets. A transaction database is partitioned into multiple input files stored by the HDFS over data nodes of a Hadoop cluster. Each mapper sequentially reads each transaction from its local input split, where each transaction is stored in the format of pair. Then, mappers compute the frequencies of items and generate local one-itemsets.

- Second MapReduce Job: Given frequent one-itemsets generated by the first MapReduce job, the second subsequent MapReduce job applies a second round of scanning on the database to prune infrequent items from each transaction record. The second job marks an itemset as a

k-itemset if it contains k frequent items ($2 \leq k \leq M$, where M is the maximal value of k in the pruned transactions).

- Third MapReduce Job: The third MapReduce job—a computationally expensive phase—is dedicated to: 1) decomposing itemsets; 2) constructing k-FIU trees; 3) mining frequent itemsets. The main goal of each mapper is twofold: 1) To decompose each k-itemset obtained by the second MapReduce job into a list of small-sized sets, where the number of each set is anywhere between 2 to k − 1 and 2) 2) to construct an FIU-tree by merging local decomposition results with the same length.

Algorithm 3: Mining k-itemsets: Mine All Frequent Itemsets

Input: Pair(k, k-itemset+support);//This is the output of the second MapReduce.

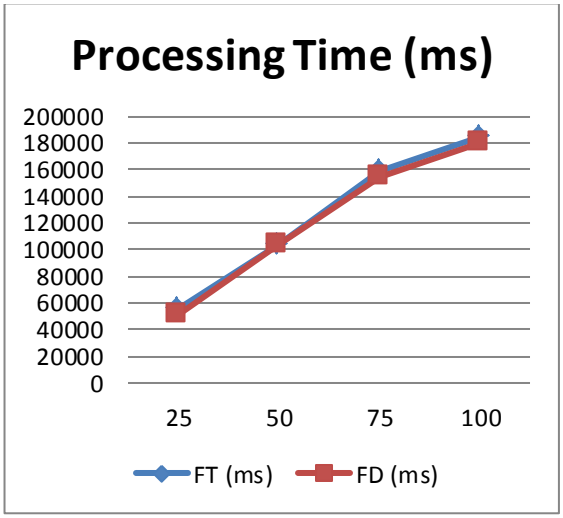
Output: frequent k-itemsets;

- function MAP(key k, values k-itemset+support)
- De-itemset ← values.k-itemset;
- decompose(De-itemset,2,mapresult); /* To decompose each De-itemset into t-itemsets (t is f from 2 to De-itemset.length), and store the results to mapresult. */
- for all (mapresult with different item length) do
- //t-itemset is the results decomposed by k-I t e m itemset(i.e. $t \leq k$);
- for all (t-itemset) do
- t − FIU − tree ← t-FIU-tree generation(local-F I FIU-tree, t-itemset);
- output(t, t-FIU-tree);
- end for
- end for
- end function
- function REDUCE(key t, values t-FIU-tree)
- for all (t-FIU-tree) do
- t − FIU − tree ← combining all t-FIU-tree from each mapper;
- for all (each leaf with item name v in t-FIU-tree) do
- if (count(v)/| DB |≥ minsupport) then
- frequent h − itemset ← pathitem(v);
- end if
- end for
- end for
- output(h, frequent h-itemset);
- end function

5. ADVANTAGES

5.1 Processing time

Processing time is defined as the time it takes to complete a prescribed procedure.
This graph shows the processing time. When compared to the existing method, there is less processing time in the proposed method.



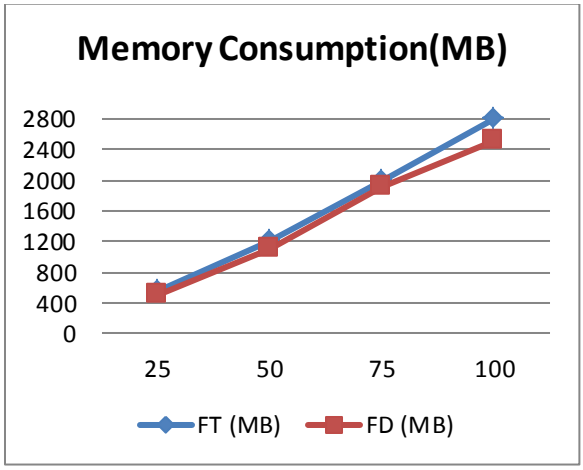
TIME COMPARE CHART

No. of Items	FT (ms)	FD (ms)
25	55200	51379
50	104083	102758
75	159122	154107
100	184526	179826

5.2 Memory consumption

Memory consumption is defined as the amount of memory consumed to execute the dataset.
Memory consumption

This graph shows the memory allocation. Memory consumption is defined as the amount of memory consumed. When compared to the proposed method, more memory is allocated for the existing method.



MEMORY COMPARE CHART

No. of Items	FT (MB)	FD (MB)
25	550	498
50	1200	1115
75	2000	1900
100	2800	2500

6. CONCLUSION

To solve the scalability and load balancing challenges in the existing parallel mining algorithms for frequent itemsets, we applied the MapReduce programming model to develop a parallel frequent itemsets mining algorithm called FiDoop. FiDoop incorporates the frequent items ultrametric tree or FIU-tree rather than conventional FP trees, thereby achieving compressed storage and avoiding the necessity to build conditional pattern bases. FiDoop seamlessly integrates three MapReduce jobs to accomplish parallel mining of frequent itemsets. The third MapReduce job plays an important role in parallel mining; its mappers independently decompose itemsets whereas its reducers construct small ultrametric trees to be separately mined. We improve the performance of FiDoop by balancing I/O load across data nodes of a cluster.The FIU tree achieves compressed storage. FiDoop runs three MapReduce jobs. The third MapReduce job is important. In third job the mapper independently decomposes itemsets and reducer built the ultrametric trees.

REFERENCES

[1] Dean J. and Ghemawat S. (2008), ―Mapreduce: Simplified Data Processing on Large Clusters,‖ Comm. ACM, vol. 51, no. 1, pp. 107-113, 2008.

[2] Fung B.C.M., Wang K., Chen R., and Yu P.S. (2010), ―Privacy-Preserving Data Publishing: A Survey of Recent Developments,‖ ACM Computing Surveys, vol. 42, no. 4, pp. 1-53.

[3] Fung B.C.M., Wang K., Chen R., and Yu P.S. (2007), ―Anonymizing Classification Data for Privacy Preservation,‖ IEEE Trans. Knowledge and Data Eng., vol. 19, no. 5, pp. 711-725, May.

[4] Machanavajjhala, Gehrke J., and Kifer D. (2006), ―LDiversity: Privacy beyond K- Anonymity‖, in Proc. of the IEEE ICDE, pp. 24.

[5] Sweeney L. (2002), ―k-Anonymity: A Model for Protecting Privacy,‖ Int'l J. Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 5, pp. 557-570.

[6] Xiao X. and Tao Y. (2006), ―Anatomy: Simple and Effective Privacy Preservation,‖ Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06), pp. 139-150.

[7] Xiao X and Tao Y. (2006), ―Personalized Privacy Preservation,‖ Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 229-240.

[8] Apache (2013), ―Hadoop, ‖ http://hadoop.apache.org.

[9] Microsoft Health Vault (2013), http://www.microsoft.com/health/ww/products/Pages/healthvault.aspx