Distributed Incremental Approximate Frequent Itemset Mining Using MapReduce

Mohsin Shaikh[†1],Irfan Ali Tunio[†2], Syed Muhammad Shehram Shah[††3], Fareesa Khan Sohu[†4] Abdul Aziz [†††5], Ahmad Ali[†††4]

engr_mohsin@quest.edu.pk, engr.irfan.ali@quest.edu.pk , shehram.shah@faculty.muet.edu.pk,, engr.fareesa.khan@quest.edu.pk, aziz.memon@iba-suk.edu.pk , alee@yonsei.ac.kr

[†]Quaid-e-Awam University of Engineering, Science & Technology Campus Larkana, 77150,Pakistan

[††]Department of software Engineering,MUET Jamshoro, Pakistan

[†††]IBA University, Sukkur, Pakistan

Abstract

Traditional methods for datamining typically assume that the data is small, centralized, memory resident and static. But this assumption is no longer acceptable, because datasets are growing very fast hence becoming huge from time to time. There is fast growing need to manage data with efficient mining algorithms. In such a scenario it is inevitable to carry out data mining in a distributed environment and Frequent Itemset Mining (FIM) is no exception. Thus, the need of an efficient incremental mining algorithm arises. We propose the Distributed Incremental Approximate Frequent Itemset Mining (DIAFIM) which is an incremental FIM algorithm and works on the distributed parallel MapReduce environment. The key contribution of this research is devising an incremental mining algorithm that works on the distributed parallel MapReduce environment.

Keywords:

Frequency Itemset minings, distributed Incremental Approximation, MapReduce.

1. Introduction

The knowledge discovery and data mining (KDD), encouraged by the advancements in the technology of data collection. KDD is concerned with the retrieval of useful and interesting information from large datasets, Frequent Itemset Mining (FIM) was proposed in the early 90s which aims to find frequently occurring subsets of items in a given set of transactions. Finding useful frequent item sets plays a valuable role in numerous real-life applications such as customer relationship management, e-commerce, DNA analysis and network Intrusion detection. FIM is also employed in the discovery of diverse patterns such as sequential patterns [1], trees, graphs and so on. The task of finding frequent item sets requires a lot of CPU and I/O resources. Apriori[2] is the first FIM algorithm and it needs to scan a dataset multiple times. To generate level-wise item sets of different length and to overcome the drawback of Apriori, several algorithms such as FP-Growth [3] have been proposed. FP-Growth overcomes the drawback of Apriori by introducing a new and compact data structure called a frequent pattern tree(FP-tree).

In a typical application domain, a number of new transactions are generated daily, so that its dataset is incrementally extended. When the most updated set of all the frequent itemsets for such an incremental dataset needs to be found, then anaïve method would be re-executing a FIM algorithm on the entire set of transactions generated so far. This method not only ignores its previously discovered knowledge but also require a huge amount of computation and I/O resources. To cope with such a problem, many incremental FIM algorithms have been proposed. In order to find the most up-to-date set of frequent itemsets, the common idea of incremental FIM algorithms [4][5][6] over a dynamic dataset is to combine the mining result of new transactions with the mining result of the old transactions previously processed. Many problems arise in merging the newly obtained results with the previously obtained knowledge. When a set of new transactions is added, some previously identified frequent item sets in the previous transactions may no longer be frequent or vice versa. When one of the above two cases arises, these algorithms try to minimize the evaluation of old transactions [7][8].

Over the past decade, researchers endeavor to explore significance and importance of data in software systems extensively [9][10][11]. However, most of these parallel algorithms suffer from the issues of network communication and I/O overheads. Implementing a parallel system raises several other problems like load-balancing, fault-intolerance and scalability. To overcome these problems, a new parallelization framework called MapReducewas introduced. To utilize the benefits of MapReduce framework, Hadoop is used, which uses hadoop distributed file system (HDFS) that can support, transform and analyze a very huge dataset effectively. A Hadoop clustercan contain a large number of nodes which can scale the computation, storage and IO capacity by simply using commodity computers. The run-time system of the MapReduce framework takes care of the details of partitioning the input dataset., scheduling the parallel execution of tasks, handling node failures, and

Manuscript received May 5, 2023
Manuscript revised May 20, 2023
https://doi.org/10.22937/IJCSNS.2023.23.5.22

managing inter-node communication. The MapReduce framework partitions the workload of a given task into a number of independent smaller sub-tasks each of which is assigned to a node in parallel environment. It can be scaled out to hundreds or even thousands of nodes effectively, offers a simplistic programming model, handles parallel job scheduling automatically, manages network communication with guaranteed robustness and fault tolerance and it also benefits the application programmer to easily perform large scale data processing in a distributed environment.

To handle a large, potentially infinite amount of an incremental dataset, a distributed and incremental mining algorithm must be used. For this purpose, this paper proposes an efficient parallel and distributed incremental approach namely Distributed Incremental Approximate Frequent Itemset Mining (DIAFIM), which works in a distributed parallel MapReduce environment. Compared with other MapReduce-based FIM algorithms, the performance of the proposed scheme is significantly enhanced by introducing an efficient approximation method while sacrificing its accuracy. Given an error bound parameter called significant support, our key idea is to approximate the support of frequent item sets within the error bound parameter of significant support, avoid multiple scans over a dataset and to speed up the mining process using a distributed and parallel environment of MapReduce using Hadoop.

The rest of the paper is organized as follows: Section 2 highlights related work in parallelization of frequent itemsetmining algorithms. Section 3 proposes DIAFIMand its details using Mapreduce over Hadoop. Experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2. Ease of Use

Diverse parallelization schemes for association rule mining are discussed [12]. Based on various trade-offs among computation, communication, synchronization and memory consumption, Agrawal and Shafer classified three different parallelizing strategies, namely Count Distribution, Data Distribution, and Candidate Distribution [13]. The count distribution is a simple distributed implementation of Apriori[14]. All distributed sites produce the entire set of candidate item sets, and each site can thus; independently get local support counts from its partition. In each iteration, this algorithm performs a sum-reduction operation, to obtain the global support counts by exchanging local support counts with all other remaining sites. Since only the support counts are exchanged among the sites, the communication overhead is reduced. However, it performs one round of communication per iteration. The data distribution algorithm generates a set of disjoint candidate item sets on each distributed site. However, to generate global support counts, each site has to scan the entire database (its local as well as all remote partitions) in each iteration of the algorithm. Hence, this approach suffers from high I/O overhead. During each iteration, the candidate distribution algorithm partitions its candidates disjointly, so that each site can generate disjoint candidates independently of the other sites, but it still requires one round of communication per iteration [15].

3. Proposed Algorithm

In this section an incremental algorithm for finding frequent item sets in a fast growing dynamic dataset is proposed. The proposed algorithm avoids multiple dataset scans and significantly enhances its performance by introducing an efficient approximation method while sacrificing the output accuracy. The proposed algorithm consists of three major phases: i*ncremental Data (**ΔD^t**), sharing (partitioningΔD^t)*, *distributed FIM* and *profile integration*. The distributed FIM phase is achieved by chaining two consecutive MapReducejobs: *Local Itemset-Mining* and *Approximate Global-Itemset Mining*.

Table 1: Pseudo Code- Itemset Mining

class MAPPER
Method MAP (key, value = a set of transactions)
Input: a dataset shard
for all shard *sh_i*∈(ΔD^t) **do**
EMIT<random_key,FIM(sh_i,s_sig)>
class REDUCER
method REDUCE(key=random_key, value=concatenated_output)
for all key *y***do**
for all value *v***in***ysvaluelist***do**

EMIT(null,value.set(output.append(v)))

In given Table 1, pseudo code can be explained through following steps:

1. Method mapper maps the keys an values of transaction and further distributing it into shared architecture. Mapping of data continues until particular threshold. Here, EMIT represents the point of data processing, mapping of data continues.
2. Shared architecture of local and global itemset mining takes place for randomly selected sigmoid function.
3. Class Reducer further reduces the data to transform and complete the mining process. REDUCE function is based on concatenation of

all the output being generated from MAPPER function.

4. REDUCER function processes the transaction for all the values of until EMIT point is satisfied.

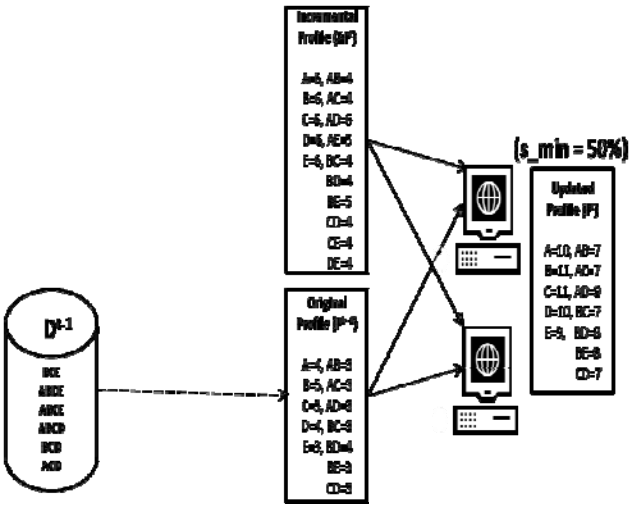

Fig.1 Profile Integration– Example

Figure 1 simple real world example experimental scene where incremental approach is applied to data in order to mine relevant set. Additionally, it reflects how machines in cloud based architecture mines the data and perform profiling of integrating relevant transactions.

4. Paragraphs and Itemizations

If you would like to itemize some parts of your manuscript, please make use of the specified style "itemize" from the drop-down menu of style categories

In the case that you would like to paragraph your manuscript, please make use of the specified style "paragraph" from the drop-down menu of style categories

5. Experimental Setup

To evaluate the overall performance of the proposed DIAFIM algorithm, several experiments are conducted. Most of the experiments related to the speedup and scalability of DIAFIM were performed using FP-Growth as FIM algorithm for local itemset mining phase on Hadoop 1.0.3 cluster of four nodes. Each node contains an Intel Quad core processor with4GB memory and Ubuntu 12.04. The algorithm was implemented in Java with the JDK 1.7. Amazon Elastic Compute Cloud (Amazon EC2) Hadoop cluster comprised of 20 M1 Medium Instances [16] is used to measure the speedup of the proposed algorithm. Design of algorithms has been always subject to multiple dimension of data sciences [17][18][19].

We used several synthetic datasets for the evaluation process. The IBM Quest data generator was used to generate several datasets with diverse characteristics. Table 1shows the average transaction length, total number of transactions and size of each dataset in Mega Bytes (MBs).Broad range of experiments is conducted to explore as many aspects of DIAFIM as possible.

A. Data Cleaning

First experiment we conducted is the comparison between the processing time of DIAFIM and a traditional distributed non-Incremental FIM algorithm using a 4 node cluster of Hadoop. In this experiment four different datasets are used with the incremental size of 1Million transactions, as expected the processing time is directly proportional to the size of the increment. As shown in Figure 2 and Table 2, the processing time of a traditional distributed non-Incremental FIM algorithm scales up almost linearly with the increasing size of a dataset. On the other hand Figure 2 shows that the processing time of DIAFIM remains almost the same with the increasing size of the dataset[20][21][22]. This is because of the incremental mining method used in DIAFIM, which does not require the entire dataset to be processed again.

Table. 2 DATA SETS

Dataset	Data configuration		
	Table column subhead	*Subhead*	*Subhead*
T10I4D10M	10	10Million	576.5
T15I4D10M	15	10Million	768.5
T20I4D10M	20	10Million	962.5
T20I5D1000K	20	1Million	130
T30I5D1000K	30	1Million	184

B. Map Reduce Scaling

In a distributed environment, one of the most important criterion to exhibit the performance of an algorithm is to test the efficiency and speedup with the increase in number of distributed computational nodes i.e. increase in the size of the MapReduce cluster.

For this experimental setup we used two different datasets with fixed size of 1Million transactions but with different average transaction length. Both datasets were processed using Hadoop cluster with increasing number of Amazon EC2 M1 Medium Instances. Number of nodes in Hadoop cluster was increased from 1 node up to 20 nodes. Figure

2 shows the relation between processing time and the number of nodes, i.e. processing time decreases with the increase in the number of nodes.
While Figure 2 shows the speedup percentage with respect to the number of nodes and it can be observed that the proposed algorithm speeds-up almost linearly with the increased number of nodes in a cluster.

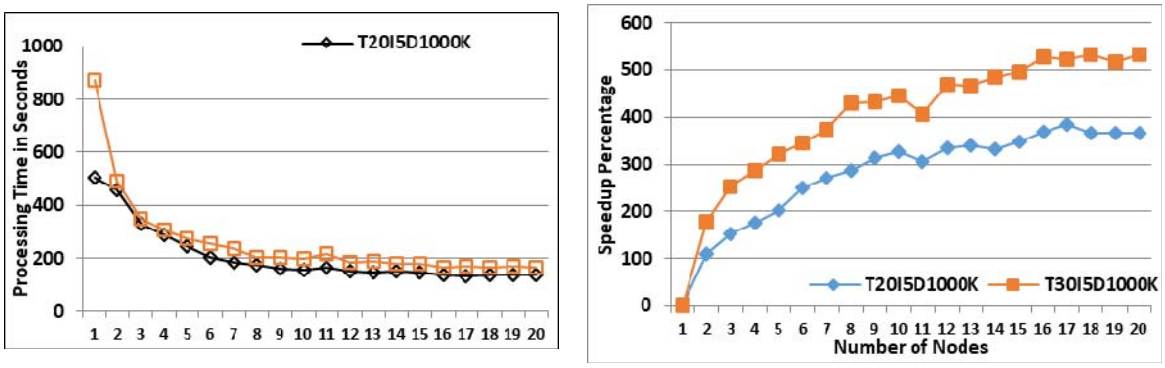

Fig. 2 Speed and processing analysis

6. Conclusion

In this paper we have considered the problem of a traditional non-incremental FIM algorithm. We proposed the distributed and incremental FIM algorithm using MapReduce framework. MapReduce and its open-source Hadoop implementation can handle many difficult problems that are inherent to the parallel process, such as concurrency control, fault control, communication over a network.
In this paper the notion of approximation method is introduced that doesn't require any additional step for checking the exact support of the item sets, which in turn helps the user to decide the trade-off between accuracy and performance.

References

[1] CláudiaAntunesandArlindoL.Oliveira, Sequential Pattern Mining Algorithms: Trade-offs between Speed and Memory, InstitutoSuperiorTécnico/INESC-
[2] Ming-Yen Lin , Pei-Yu Lee , Sue-Chen Hsueh, Apriori-based frequent itemset mining algorithms on MapReduce, Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, February 20-22, 2012, Kuala Lumpur, Malaysia
[3] Ming-Yen Lin , Pei-Yu Lee , Sue-Chen Hsueh, Apriori-based frequent itemset mining algorithms on MapReduce, Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, February 20-22, 2012, Kuala Lumpur, Malaysia
[4] D. Cheoung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered associated rules in large databases: An incremental updating technique," in Proc. 12th Int. Conf. Data Engineering, Feb. 1996, pp. 106-114
[5] D. Cheoung, S. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," in Proc. 5th Int. Conf. Database System Advanced Application, Apr. 1997, pp. 1-4
[6] V. Ganti, J. Gehrke, and R. Ramakrishnan, "Demon: Mining and monitoring evolving data," in Proc. 16th Int. Conf. Data Engineering, San Diego, CA, 2000, p. 439-448
[7] Otey, M.E., Parthasarathy, S., Wang, C., Veloso, A., Meira, W., Jr., Parallel and distributed methods for incremental frequent itemset mining. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(6), 2439-2450.
[8] ID,DepartmentofInformationSystemsandComputerScience, Av.RoviscoPais1,1049-001Lisboa,Portugal.

[9] Shaikh, Mohsin, Ki-Seong Lee, and Chan-Gun Lee. "Assessing the Bug-Prediction with Re-Usability Based Package Organization for Object Oriented Software Systems." IEICE TRANSACTIONS on Information and Systems 100.1 (2017): 107-117.
[10] Shaikh, Mohsin, and Chan-Gun Lee. "Aspect Oriented Re-engineering of Legacy Software Using Cross-Cutting Concern Characterization and Significant Code Smells Detection." International Journal of Software Engineering and Knowledge Engineering 26.03 (2016): 513-536.
[11] Shaikh, Mohsin, et al. "Open-source electronic health record systems: A systematic review of most recent advances." Health Informatics Journal 28.2 (2022): 1460458222109982
[12] M.J.Zaki, "Parallel and distributed association mining: A survey," IEEE concurrency, vol. 7(4), pp. 14-25, 1999
[13] R. Agrawal and J. Shafer. Parallel mining of association rules. Transactions of Knowledge and Data Engineering, 8(6):962-969, 1996
[14] AgrawalRakesh and RamakrishnanSrikant. Fast algorithms for mining association rules.In Proc. 20th Int. Conf. Very Large Data Bases, VLDB, 1994.
[15] Otey, M.E., Parthasarathy, S., Wang, C., Veloso, A., Meira, W., Jr., Parallel and distributed methods for incremental frequent itemset mining. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(6), 2439-2450.
[16] Amazon Elastic Compute Cloud (Amazon EC2)http://aws.amazon.com/ec2/
[17] Mohsin, Shaikh, and Zeeshan Kaleem. "Program slicing based software metrics towards code restructuring." In *2010 Second International Conference on Computer Research and Development*, pp. 738-741. IEEE, 2010.
[18] Shaikh M, Jalbani AH, Ansari A, Ahmed A, Memon K. Evaluating Dependency based Package-level Metrics for Multi-objective Maintenance Tasks. International Journal of Advanced Computer Science and Applications. 2017;8(10).
[19] Shaikh M, Ibarhimov D, Zardari B. Assessing Architectural Sustainability during Software Evolution using Package-Modularization Metrics. International Journal of Advanced Computer Science and Applications. 2019;10(12).
[20] Mujeeb-ur-Rehman Jamali, Abdul Ghafoor Memon, Nadeem A. Kanasro, Mujeeb-u-Rehman Maree. "Data integrity issues and challenges in next generation non-relational document-oriented database outsourced in public cloud", International Journal of Emerging Trends in Engineering Research, Volume 9. No. 4, April 2021- ISSN 2347 – 3983.
[21] Mashooque Ahmed Memon , Mujeeb-ur-Rehman Maree Baloch , Muniba Memon , Syed Hyder Abbas Musavi," A Regression Analysis Based Model for Defect Learning and Prediction in Software Development", Mehran University Research Journal of Engineering and Technology, Vol.40, No. 3, 617- 629, July 2021, p-ISSN: 0254-7821, e-ISSN: 2413-7219.
[22] Memon Abdul Ghafoor, Jianwei Yin, Jinxiang Dong, Maree Mujeeb-u-Rehman, "Service-oriented Mobile Calculus Technology in M-Business interoperability between Customer and e-Shop", Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05).