

A MAPREDUCE BASED FREQUENT ITEMSETS MINING USING HASH BASED APRIORI ALGORITHM

¹Ravishankar Sahu, ²Abhishek Badholia

¹P.G. Student, ²Assistant Professor

¹Department of Computer Engineering, Central College of Engineering and Management, Raipur, India

²Department of Computer Engineering, Central College of Engineering and Management, Raipur, India

Abstract: Keeping in mind the end goal to enhance the effectiveness of Apriori algorithm for mining frequent sets. Hashed based-Apriori calculation was intended for huge information to address the poor performance issue of existing algorithm. Hashed based-Apriori takes favorable circumstances of MapReduce and hashed tree together to advance Apriori calculation. In this paper, we propose a novel method in combination with Hadoop MapReduce and hashed based tree for mining of frequent pattern. Proposed method outperforms the existing method in terms of time.

Keywords - Big Data, Apriori Algorithm, Frequent Pattern Mining, Hadoop, MapReduce.

I. INTRODUCTION

The Apriori algorithm [2] is one of most well-known techniques for mining frequent itemsets in a value based database. The algorithm works inside a numerous finish generation-and-test system, including the joining and the pruning stages to reduce the quantity of candidates before examining the database for support counting. Numerous algorithms, for example, the FP-growth algorithm [7] and we have been proposed to overcome the shortcoming of the Apriori algorithm: level-wise applicant generations and various pass database checks. Specialists too try to parallelize these frequent itemset mining algorithms to accelerate the mining of the regularly expanding measured databases.

Parallelized mining endeavors to separate the mining issue into littler ones and take care of the sub-issues utilizing homogeneous nodes with the end goal that every node may work autonomously and at the same time. In spite of the fact that the parallelization may enhance the mining execution, it likewise raises a few issues including the parceling of the input data, the balancing of the workloads, the arrangement of worldwide data from local nodes, and the minimization of the correspondence costs. Conveying the mining strategies into a grid figuring condition by breaking down the task into smaller pieces and dispatching the sub-undertakings to grid nodes may likewise enhance the execution. The assumption of all grid nodes are safeguard and all assignments can be accurately finished may be excessively solid in real grid environments.

As a rule, the potential errors because of failure nodes increment as the quantity of grid nodes increments. The failure of nodes can't be disregarded since mistaken or deficient data are presented. Surprisingly more terrible, the failure may cause a perpetual re-execution of the considerable number of occupations. Furthermore, the adaptability of the grid is restricted in light of the fact that running on several nodes is inclined to error.

To defeat the above issue, the MapReduce structure [5] has been presented. MapReduce empowers the distributed processing of enormous data on expansive clusters, with great versatility what's more, vigorous adaptation to non-critical failure. A scale-up of hundreds or even a huge number of nodes can be easily settled. Algorithms running in the structure are depicted by two noteworthy functions, map and reduce. The input data are partitioned (and conceivably duplicated) and put away in various nodes. A master node starts and calendars the two functions for executions in the nodes. The map function takes (from its node) the input data as a <key, value> combine and outputs a rundown of <key, value> pairs in a distinctive domain. The reduce function takes the arranged output of the map function as <key, rundown of-values> and outputs a gathering of qualities. The two functions (i.e. a map errand and a reduce assignment) can be performed in parallel. In this manner, MapReduce can be a productive stage for frequent itemset mining in tremendous datasets of terabyte or then again bigger scale. Utilizing the MapReduce structure to parallelize the mining undertaking on a cluster of cheap servers may outflank a few serial mining algorithms on an intense server.

Changing over a serial Apriori-like mining algorithm into a distributed algorithm on the MapReduce system won't not be troublesome, however the mining execution may be inadmissible. The different pass feature of the Apriori algorithm requires different map-reduce stages. The master must timetable employments to instate each map-reduce stage. The map function of a mapreduce stage can't begin until all the reduce functions of its past stage have wrapped up. Nodes that completes their reduce functions need to sit tight for all the incomplete nodes to finish.

The scheduling and the waiting are unadulterated overheads profoundly mining task. Furthermore, each map function gets just a segment of the input data since the input data are partitioned into numerous nodes. The partitioned input represses viable local data trades among map errands. Worldwide data can be acquired just by the reduce function. Along these lines, the map function is unconscious of the consequences of other map errands and must output all the local data without pruning in a stage. To proficiently executing the level-wise mining algorithms utilizing the MapReduce system, we have to limit the quantity of

scheduling invocations, augment the use of every node amid each map-reduce stage, and adjust the workloads of all nodes in the MapReduce cluster.

II. LITERATURE SURVEY

- Agrawal et al. [1], in this paper author propose algorithms for age of frequent item sets by progressive development of the nodes of a lexicographic tree of item sets. Author talks about various systems in age and traversal of the lexicographic tree, for example, breadth-first search, depth-first search, or a mix of the two. These procedures give diverse exchange offs as far as the I O, memory, and computational time necessities. author utilize the various leveled structure of the lexicographic tree to progressively extend exchanges at every node of the lexicographic tree and utilize matrix relying on this decreased arrangement of exchanges for finding frequent item sets. Author tried our algorithm on both genuine and engineered information.
- Zaki et al. [2], various vertical mining algorithms have been proposed as of late for association mining, which have appeared to be extremely successful and typically outflank flat methodologies. The fundamental preferred standpoint of the vertical arrangement is bolstering for quick frequency tallying by means of crossing point activities on exchange ids (tids) and programmed pruning of unimportant information. The primary issue with these methodologies is when middle of the road aftereffects of vertical tid records turn out to be too vast for memory, hence influencing the algorithm versatility.
- Han et al. [3], Association rule mining is an information mining procedure. It is utilized for finding the items from an exchange list which happen together frequently. A portion of the algorithms which are utilized most prevalently for association rule mining are i) Apriori algorithm ii) FP-tree algorithm. This paper researches on utilization of present day algorithm Apriori for book search for prescribing a book to a client who needs to purchase a book in light of the data that is kept up in the exchange database. The aftereffect of this contrasted and other algorithm accessible for association rule mining.
- Lin et al. [4], numerous parallelization procedures have been proposed to improve the execution of the Apriori-like frequent item set mining algorithms. Portrayed by both guide and diminish capacities, MapReduce has risen and exceeds expectations in the mining of datasets of terabyte scale or bigger in either homogeneous or heterogeneous clusters. Limiting the booking overhead of each guide decrease stage and amplifying the use of nodes in each stage are keys to fruitful MapReduce usage. In this paper, Author propose three algorithms, named SPC, FPC, and DPC, to explore compelling usage of the Apriori algorithm in the MapReduce structure.
- Li et al. [5], Searching frequent examples in value-based databases is considered as a standout amongst the most vital information mining issues and Apriori is one of the commonplace algorithms for this errand. Growing quick and productive algorithms that can deal with vast volumes of information turns into a testing errand because of the huge databases.
- Hammoud. et al. [6], over the most recent couple of years, various cooperative grouping algorithms have been proposed, i.e. CPAR, CMAR, MCAR, MMAC and others. This theory likewise presents another MapReduce classifier that based MapReduce affiliated rule mining. This algorithm utilizes diverse methodologies in rule disclosure, rule positioning, rule pruning, rule forecast and rule assessment techniques. The new classifier chips away at multi-class datasets and can deliver multi-mark predications with probabilities for each anticipated name.
- Li et al. [7], frequent itemset mining (FIM) is a valuable device for finding frequently co-occurrent items. Since its beginning, various significant FIM algorithms have been produced to accelerate mining execution. Shockingly, when the dataset estimate is immense, both the memory utilizes and computational cost can at present be restrictively costly. In this work, we propose to parallelize the FP-Growth algorithm (we call our parallel algorithm PFP) on disseminated machines.
- Zhou et al. [8], As a critical piece of finding association rules, frequent item sets mining assumes a key part in mining associations, relationships, causality and other imperative information mining undertakings. Since some conventional frequent item sets mining algorithms can't deal with gigantic little records datasets viably, for example, high memory cost, high I/O overhead, and low figuring execution, an enhanced Parallel FP-Growth (IPFP) algorithm and talk about its applications in this

paper. Specifically, a little documents handling system for gigantic little records datasets to remunerate imperfections of low read/compose speed and low preparing effectiveness in Hadoop.

- Riondato et al. [9], In this paper, author have portrayed PARMA, a parallel algorithm for mining semi ideal accumulations of frequent item sets and association rules in MapReduce. Author appeared through hypothetical examination that PARMA offers provable assurances on the nature of the yield accumulations. Through experimentation on an extensive variety of datasets going in measure from 5 million to 50 million exchanges, we have exhibited a 30-55% runtime change over PFP.

III. METHODOLOGY

In this section we discuss the proposed methodology in detail. Fig. 1. Shows the work flow of proposed method

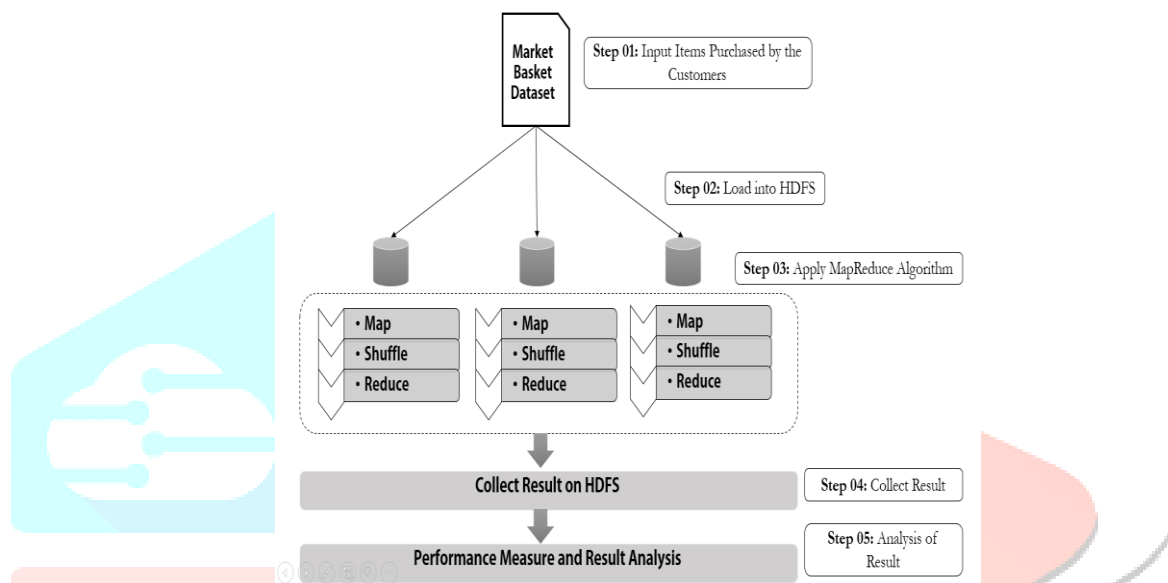


Fig. 1. Proposed Work Flow

As shown in fig. 1, the framework consists of different modules.

- Input → Market Basket Dataset
- Load Data into HDFS
- Map, Shuffle and Reduce Algorithm
- Result Aggregation in HDFS
- Result Analysis

Market Basket Dataset

The dataset of different customers are collected from the shopping mart. The dataset has various parameters which can be used for analysing the customer purchase behaviour.

Load into HDFS

Load the dataset into HDFS for analysis into the input directory. The HDFS is Simple Storage used for storing the object files.

MapReduce Algorithm

After loading dataset into HDFS for analysis, the cluster needs to be started and configured for analysis. This phase includes the execution of MapReduce algorithm on the clusters. The MapReduce algorithm consists of 3 different phases:

- Map Phase
- Shuffle Phase
- Reduce Phase

Algorithm: Mapper

Input: Market Basket dataset

Parameters: HDFS Input Directory

HDFS Output Directory

Maximum Number of Passes (MaxP)

Minimum Support Percentage

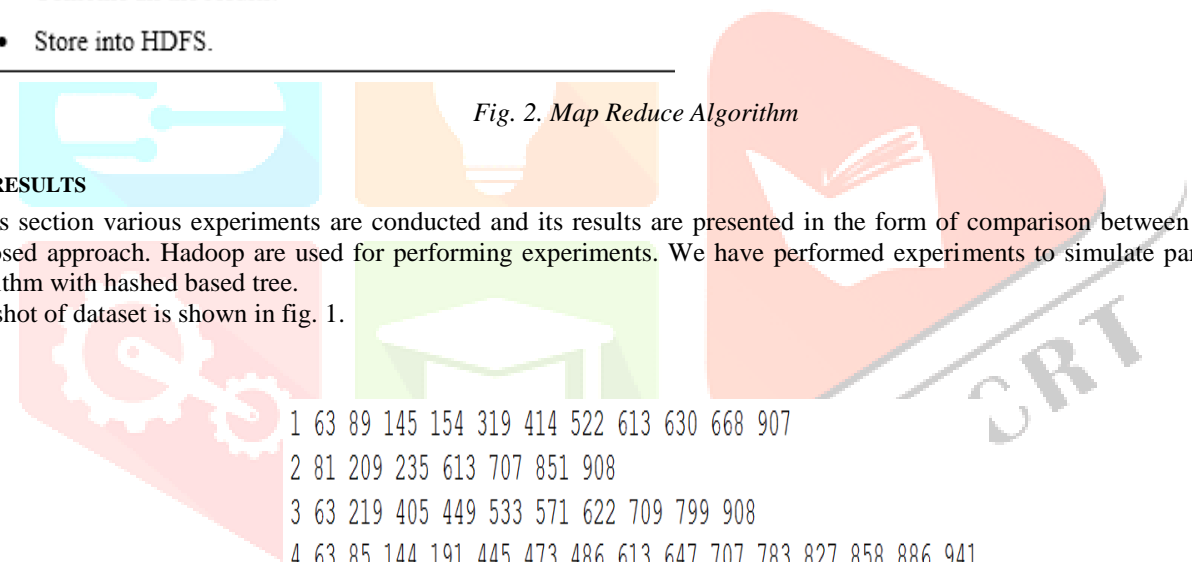
Maximum Number of Transaction

-
- Start
 - For Loop pass = 1 && pass <= MaxP
 - Call Mapper for each pass
 - Hash Based Tree Construction from item sets.
 - End Loop
 - Output Execution time for each Map task
 - Start Reducer
 - Iterate through each item set in the dataset.
 - Combine all the results.
 - Store into HDFS.
-

Fig. 2. Map Reduce Algorithm

IV. RESULTS

In this section various experiments are conducted and its results are presented in the form of comparison between existing and proposed approach. Hadoop are used for performing experiments. We have performed experiments to simulate parallel Apriori algorithm with hashed based tree. Snapshot of dataset is shown in fig. 1.



```

1 63 89 145 154 319 414 522 613 630 668 907
2 81 209 235 613 707 851 908
3 63 219 405 449 533 571 622 709 799 908
4 63 85 144 191 445 473 486 613 647 707 783 827 858 886 941
5 281 418 432 478 561 568 613 644 966
6 34 67 255 259 333 500 613 815 829
7 0 3 7 289 300 496 575 592 820 917
8 344 372 443 517 690 827 855 870 912 981
9 11 432 850 974
10 13 71 109 143 432 461 517 518 739 790 795 852 945 995
11 17 85 104 123 446 545 592 710 797 808
12 3 229 377 379 451 558 704 824 829 946
  
```

Fig. 1. Shows the market basket

FOR PASS – 01

0	1310
10	788
100	1811
101	1430
104	1594
106	1601
108	898
109	1305
11	3881
112	1539
116	4061
117	916

In the above pass, first column is item code and second column is transaction id or customer id.
for pass – 02

116,132	774
116,439	1141
116,727	840
116,993	1202
125,222	780
125,744	761
132,853	995
143,432	849

In the above pass, first and second column is item code and third column is transaction id or customer id.

FOR PASS – 03

143,432,517	779
143,432,796	783
143,432,995	789
143,517,796	773
143,517,995	779
143,796,995	786
158,778,973	833
16,363,612	822

In the above pass, first, second and third column is item code and fourth column is transaction id or customer id.

FOR PASS – 04

143,432,517,796	738
143,432,517,995	742
143,432,796,995	748
143,517,796,995	739
432,517,796,995	743
471,592,594,807	746

In the above pass, first, second, third and fourth column is item code and fifth column is transaction id or customer id.

V. CONCLUSION

Hashed based-Apriori calculation was intended for huge information to address the poor performance issue of existing algorithm. Hashed based-Apriori takes favorable circumstances of MapReduce and hashed tree together to advance Apriori calculation. The existing approach follows simple MapReduce paradigm with Apriori implementation. The proposed approach is more advanced in terms of execution time. The hashed based implementation is useful for searching of the items from the dataset easily and effectively.

REFERENCES

- [1] R. Agrawal, C. Aggarwal, and V. Prasad, "A Tree Projection Algorithm for Generation of Frequent Item Sets," *Parallel and Distributed Computing*, pp. 350-371, 2000.
- [2] Mohammed J. Zaki, Karam Gouda, "Fast Vertical Mining Using Diffsets", 2003 ACM.
- [3] Han Jiawei, KamberMiclina. Fan Ming, MengXiaofeng translation, "Data mining concepts and technologies". Beijing: Machinery Industry Press. 2001.
- [4] Lin, M. Y., Lee, P. Y., & Hsueh, S. C. (2012, February). Apriori-based frequent itemset mining algorithms on MapReduce. In *Proceedings of the 6th international conference on ubiquitous information management and communication* (p. 76). ACM.
- [5] Li, N., Zeng, L., He, Q., & Shi, Z. (2012, August). Parallel implementation of apriori algorithm based on MapReduce. In *Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on* (pp. 236-241). IEEE.
- [6] S. Hammoud. *MapReduce Network Enabled Algorithms for Classification Based on Association Rules*. Thesis, 2011.
- [7] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang and Edward Chang, " PFP: Parallel FP-Growth for Query Recommendation", ACM 2008.
- [8] Zhou, L., Zhong, Z., Chang, J., Li, J., Huang, J. Z., & Feng, S. (2010, November). Balanced parallel FP-growth with mapreduce. In *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on* (pp. 243-246). IEEE.
- [9] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal. *PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce*. In *Proc. CIKM*, pages 85-94. ACM, 2012.
- [10] Moens, S., Aksehirli, E., & Goethals, B. (2013, October). Frequent itemset mining for big data. In *Big Data, 2013 IEEE International Conference on* (pp. 111-118). IEEE.

