IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-03, Issue-09 E-ISSN: 2347-2693

# A Streamlined Frequent Item set excavating using FP growth from Map Reduce

Ahilandeeswari. G[1] and R.Manicka Chezian[2*]
[1,2*] Department of Computer Science, NGM College, Pollachi, India

www.ijcseonline.org

Received: Aug/12/2015 Revised:Aug/28/2015 Accepted:Sep/20/2015 Published: Sep/30/2015

***Abstract—*** As a significant part of discovering association rules, frequent item sets excavating plays a key role in removal associations, correlations, bass and other imperative data mining tasks. Since a little customary frequent item sets mining algorithms are incapable to knob gargantuan small file datasets effectively, such as high recall cost, high I/O operating price, and squat computing recitals, a better Parallel FP-Growth (EPFP) algorithm and converse its applications in this paper. In particular, a small file processing strategy for huge small file datasets to reimburse defects of squat read/write speed and low processing efficiency in Hadoop. Moreover, utilize of Map Reduce to execute the parallelization of FP-Growth algorithm, thereby improving the general performance of frequent item set mining. The investigation results demonstrate that the EPFP algorithm is practicable and suitable with a excellent speedup and a higher mining efficiency, and can convene the rapidly growing needs of frequent item sets mining for enormous petite file data sets.

***Keywords***: Frequent item set mining, Hadoop Map Reduce, Parallel FP-Growth, Small files problem

## I. INTRODUCTION

Association rules are one of the most active research methods in data mining. Association rule mining is to find strong association rules. That can be alienated into two sub-problems, discovering a frequent item-sets and generating association rules. The main features of data mining technology are that it discovers implicit and useful knowledge from huge, multifaceted and high-latitude data sets. This puts special challenges on association rules techniques. In 1994, Agrawal proposed the famous Apriori algorithm, but there are two drawbacks in it. First, because it repeatedly scans the contract database, it needs a lot of I/O load; Second, it will cause vast candidate set. FP-Growth is a good solution to the above two problems. The biggest advantage of the FP-Growth algorithm is that it only scans database twice. It directly compresses the database into a frequent pattern tree instead of using a candidate set and finally generates association rules through the FP-tree. As one of the important investigate directions of data mining, frequent item sets mining plays an essential role in mining associations, correlations, causality and other vital data mining tasks [1] which is a brawny impetus to the applications of association rules in markets selection, decision analysis and business management [5]., massive data are growing rapidly. Apache Hadoop is an open source distributed software platform for storing and processing data. Written in Java, it runs on a cluster of industry-standard servers configured with direct-attached storage. Association rules show attributes worth conditions that occur frequently jointly in a given dataset.

Corresponding Author: Corresponding Author: *Ahilandeeswari.G, ahilaplatinum@gmail.com, Department of Computer science, NGM College, Pollachi India*

Distributed File System (HDFS) and Map Reduce parallel programming replica provide a novel idea for handling big data. In the frequent item sets mining for large-scale data, a Map Reduce looms of parallel FP-Growth (PFP) algorithm is proposed in [8], and the performance of PFP algorithm is improved by adding load balancing features in [11], but these methods ignore frequent Item sets mining for massive small file data sets in Hadoop. Diminutive files usually refer to those file sizes, which are less than 64 MB. According to a study in 2007 at the National Energy Research Scientific Computing Center, 43% of the over 13 million files on a shared parallel file system are under 64 KB and 99% are under 64 MB (Petascale Data Storage Institute (2007)), and more scientific applications consist of a large number of small files are interpreted in [4]. Nevertheless, in the face of immense small file data sets, the constructed FP-tree in Parallel FP-Growth (PFP) algorithm cannot fit into the memory, which frequently causes problems such as memory overflow and mammoth communication overhead. Meanwhile, the computing efficiency of the Hadoop platform largely depends on the recital of HDFS and Map Reduce [10], and Hadoop was, at first, designed specifically to handle streaming large files, so when dealing with colossal small files, there are significant limitations. Huge small files will reduce the performance of Hadoop, which is mainly shown in the following two aspects: [7].

*(1) The entrée competence of HDFS is decreased.*
*(2) The added overhead of Map Reduce is increased.*

Hadoop skeleton is trendy for HDFS as well as Map Reduce. HDFS is the Hadoop file system and comprises two main components: namespaces and blocks storage service. The namespace service manages operations on files and

 

directories, such as creating and modifying files and directories.

## II. RELATED CONCEPTS AND DESCRIPTION

### A. Item Sets Space approach

Agrawal et al. R ecognized the item-sets space theory for transactional database excavating. The hub attitude of this theory is that the subsets of recurrent item-sets are frequent item-sets; the superset of non-frequent item-sets are non-frequent [6]. This attitude, anti-monotone property, has been applied as a typical data mining theory. In 1994, they proposed Apriori and the Apriori algorithm has been still widely discussed as the standard association rule mining algorithm. But with further research, its shortcomings bare. For every k cycle, the algorithm

Has to scrutinize the database once to confirm it, whether or not to join Lk for every element of the candidate set. That is exponential expansion. When the number of frequent item sets is large, it will fabricate a huge candidate set. This is a confront for time and memory space. In order to recover the effectiveness of Apriori algorithm; it appears a series of improved algorithms, such as the data partitioning technique, hash-based method, transaction compression method and so on. Although they still follow the above hypothesis, due to the preface of the relevant technology, these algorithms improve the adaptability and efficiency of the Apriori algorithm to some amount.[1]

### B. FP-Growth Algorithm

Apriori is a common algorithm for mining frequent item-sets, which has an awfully important nature: all non-empty subsets of frequent item-sets must are also frequent. But it has to scrutinize record multiply prior to it produces frequent patterns and at the same time produces a large number of candidate frequent sets. That makes the Apriori algorithm have larger time and space complexity. Besides, the recital of the Apriori algorithm in mining long frequent patterns is often low. In 2000, Han proposed the FP-Growth algorithm.The primary idea is that first sweep the Transactional database to locate frequent 1-item sets, and then construct the FP-tree. At last it discovers conditional pattern base to pit regular pattern based on the FP-tree [3].Use transaction database information to construct FP-tree

*1) Scrutinize database for the first occasion, get frequent 1-item sets L.*

*2) Create the origin of the tree with the "root" tag. Scan DB for the second time and make a branch for each transaction.*

Mining frequent patterns of the FP-tree

*1)For apiece item, generate its provisional pattern base and then its conditional FP-tree;*

*2) For apiece new generated provisional FP-tree, repeat This step until the FP-tree is null or it only has unique branch;*

The algorithm mining frequent patterns in the FP-tree is as follows:

*Input:* the tectonic fine FP-tree; transaction database

DB; minimum support threshold Minsup.

*Output:* the complete set of frequent patterns. Method: Call FP-growth (FP-tree, null).The hub for mining FP-Tree algorithm is the FP-growth path. It achieves frequent patterns in the form of recursive calls.

## III. SYSTEM MODEL

Hadoop is able to take packed advantage of the control of clusters to figure and store tasks in high speed. Hadoop is a software outline which can process large amounts of facts distributed and is reliable, efficient, and scalable. Hadoop assumes to computing elements and storage tin fail and so it maintains many working copies of data to ensure the redistribution process for the failed lump. So it is reliable. Hadoop works in similar and speeds up processing through the mode of parallel computing. So it is able. Hadoop is scalable and capable of handling the PB rank data [11]. Therefore, Hadoop is suitable for the algorithm. The structure of Hadoop components is exposed in Fig.1. In the architecture, Hadoop general provides a generic function block to support the Hadoop subprojects. Map Reduce components supply Map and Reduce processing. HDFS compiles distributed sleeve storage mechanism. ZooKeeper provides vital services like to distribute lock for building distributed applications [5]. The most core designs of Hadoop are HDFS and Map Reduce computation model [4], HDFS is an implementation of Hadoop Distributed File scheme and provides the underlying support for distributed computing storage. The design of Map Reduce was initially raised by one of Google's papers. The Easiest explanation for Map Reduce is that task decomposition and a summary of the results

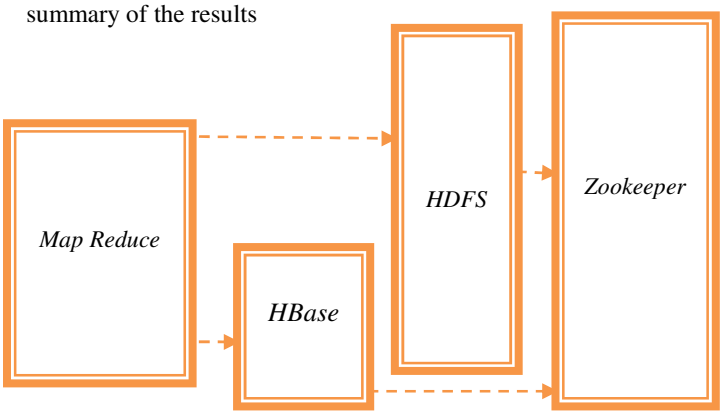


Figure 1: Hadoop module arrangement

HDFS is a simple but really powerful distributed file system. It is able to store data reliably at main scale. HDFS deployments live with thousands of nodes storing hundreds of petabytes of client data. The distributed file system component, the major example of which is the Hadoop Distributed File System,

though other file systems, such as IBM GPFS-FPO, are supported.[5]
HDFS is a highly fault-tolerant distributed system. Distributed File System has the following basic characteristics:
*1) A solitary namespace for the entire cluster*

*2) Data consistency, appropriate for write-once many read model. The client cannot see the survival of the file before it is not successfully created.*

*3) The sleeve will be divided into multiple folders.*

Each file block is allocated to amass the data node. It will have to replica the file block to guarantee the security of the data according to the configuration. HDFS is suitable for deployment in a cheap machine. HDFS provides high throughput facts access and ideal for applications on large-scale data sets. HDFS is a master-slave structure system. HDFS clusters are made from a Name Node and many Data Nodes. Each is a node frequent PC HDFS has three important roles, Name Node, Data Node, and Secondary Name Node, job tracker, task tracker and Client.[7][3]

Name Node can be seen as a manager for the distributed file system. It is primarily responsible for managing the file system's namespace, cluster configuration and storage block replication. Info concerning each file block in Data Node. Data Node is the basic unit of the file storage. It stores Block in the local file scheme and saves only the Metadata of Block. At the same time it sends reports of all existing Blocks periodically to Name Node. The Client is the application procedure that needs to obtain the distributed file system files.[8][9]

Secondary Name node periodically merges the namespace image with the audit log and maintains a copy of this namespace image. It usually runs on a separate machine. However the Secondary Name node lags in state with the primary Name node, thus in case of failure of a primary Name node some data loss occurs for sure. A Job tracker coordinates all the jobs that are run on the system by scheduling each task to run on task trackers. It is the responsibility of Job tracker to reschedule a failed task on a different task tracker.[1][2]

Map Reduce is a equivalent programming framework that integrates with HDFS. It allows users to express data analysis algorithms in terms of a little number of functions and operators, chiefly, a map function and a reduce function. The Map Reduce component, which is a framework for performing calculations on the data in the distributed file system. Pre-Hadoop 2.2 Map Reduce is referred to as Map Reduce V1 and has its own built-in resource manager and scheduler[5]. MapReduce is an important advance because it allows ordinary developers, not just those skilled in high-performance computing, to use parallel programming constructs without worrying about the complex details of intra-cluster communication, task monitoring, and failure handling. MapReduce simplifies all that.[4]Map Reduce is a programming model for large-scale data set (more than 1TB). System's namespace, cluster configuration and storage block replication. Moreover Map reduce saves the programmers from writing code for node failure and handling dataflow as these are handled implicitly by Map Reduce. Whereas Grid Computing provides greater control to handle data flow and node failures.

Name Node will store Metadata of the file system in memory. These include the file information, the file block information corresponding to each file and the information of each file block in Data Node. Data Node is the basic unit of the file storage. It stores Block in the local file system and saves only the Metadata of Block. At the same time it sends reports of all existing Blocks periodically to Name Node. The Client is the application procedure that needs to obtain the distributed file system files. Map Reduce is a programming model for large-scale data set (more than 1TB). Its main idea is borrowed from the functional programming language as well as vector Programming language. It greatly facilitates that programmers make their own procedures run in the distributed system without knowing the parallel programming.
Fig. 2 shows the estimated data flow diagrams of Map Reduce. It is a highly efficient.[10]

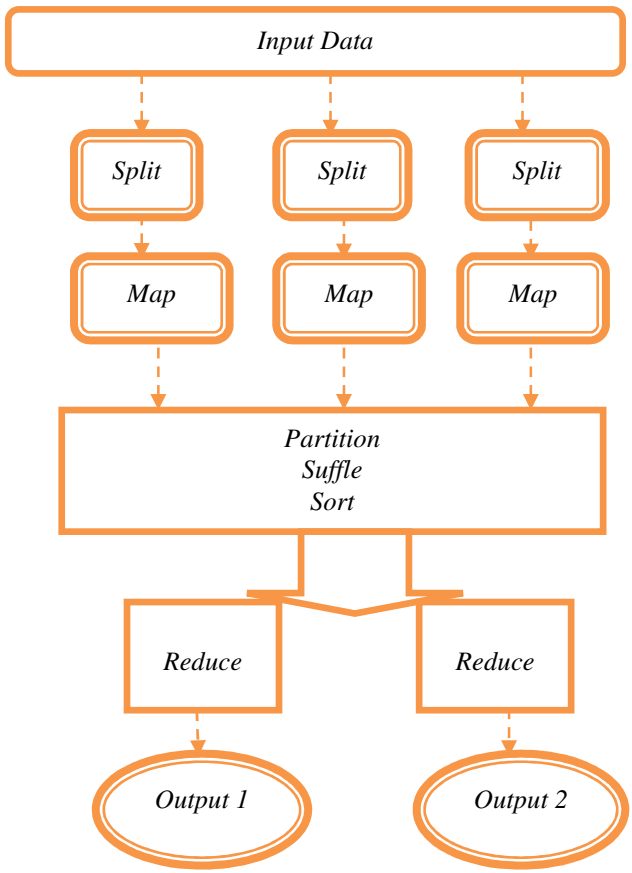


Figure. 2 Map Reduce estimated dataflow diagram

## IV. FP GROWTH ALGORITHM USING MAP REDUCE

Map Reduce distributes the operations on big data sets to a master node and some sub-node in sort to complete it together. Then during integration of the intermediate results of each node it gets the final result.[10]

TABLE 1
TRANSACTION DATABASE

| TID | Transaction | TID | Transaction |
|---|---|---|---|
| 1 | J1 J3 J5 | 6 | J1 J2 J5 |
| 2 | J1 J2 J3 | 7 | J1 J2 J3 |
| 3 | J3 J5 | 8 | J1 J3 |
| 4 | J2 J3 J4 | 9 | J1 J2 J3 J4 J5 |
| 5 | J2 J4 | | |

The task of each Mapper is responsible for adding up the number of individual items in the Mapper data wedge. Combiner midway function merges the intermediate results outputted by each Mapper in sort to diminish the transmission of data between map tasks and reduce tasks.[11] Then after the Map Reduce framework handles it, finally the harvest data is sent to reduce the function to get the final result. Reckon the number of each item in the database and store items whose support is greater than or equal to the minimum support in F-List (the support of thesis instance is 3), F-List = {I3: 7, I1: 6, I2: 6, I5: 4, I4:3}, F-List memories the utmost frequent item sets. A Combiner intermediate function is mentioned here. The available bandwidth of clusters limits the number of Map Reduce jobs, so the most important thing is to try to Avoid the transmission of data between map tasks and reduce tasks. Hadoop allows users to denote a merge function Combiner for output of plan tasks. Its production is the contribution of reduce. Merge utility is an optimization program. No stuff how many times Combiner utility is called when the schedule is running, the final output is reliable with each other

***The pseudo-code for Mapper***

```
Modus operandi:Mapper(key, value=Ji)
foreach item mi in Ji do
Call Output(<mi,'1'>);
End
```

Figure 3:Pseudo-code for Mapper

***The pseudo-code for Combiner and Reducer***

```
modus operandi: Combiner(key,value=Output(<mi,'1'>) of
each mapper)
C ←0;
foreach item '1' in mi do
C← C+1;
Call Output(<mi,C>);
End

modus operandi: Reducer(key=mi , value=S(mi)) D← 0;
foreach item 'C' in Ji do
D ← D+C;
end
Call Output(<mi,D>);
```

Figure 4 : Pseudo-code for Combiner and Reducer

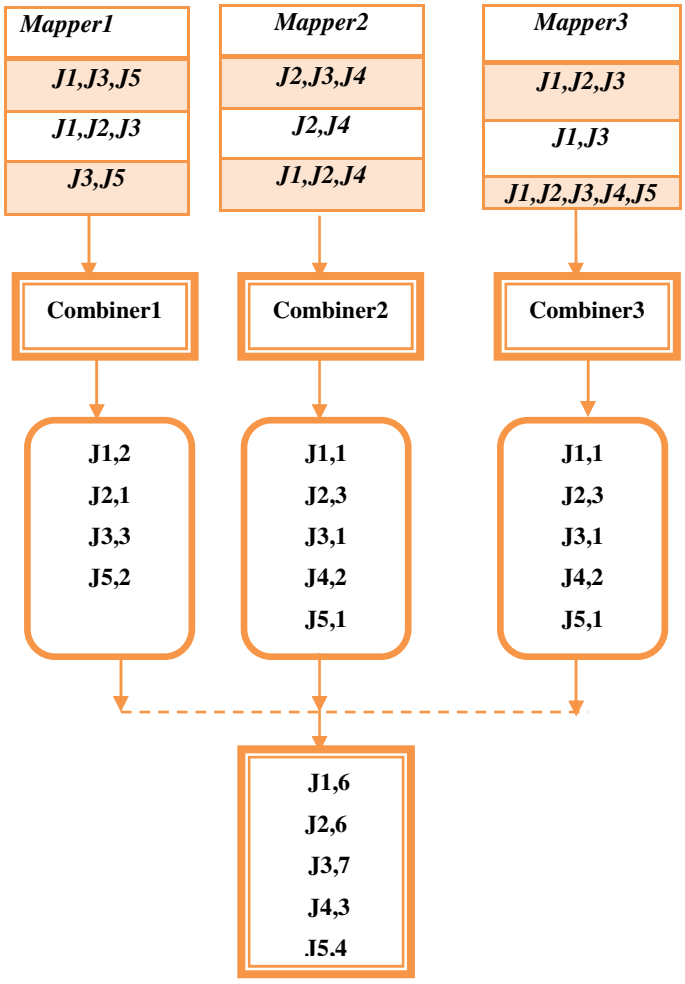


Figure 5: Disseminated statistic vision of objects

The transaction database is sorted in accordance with the frequency size of items. The results are shown in table II

TABLE 2
SORT TRANSACTION DATABASE

| TID | Transaction | TID | Transaction |
|---|---|---|---|
| 1 | J3 J1 J5 | 6 | J1 J2 J5 |
| 2 | J3 J1 J2 | 7 | J3 J1 J2 |
| 3 | J3 J5 | 8 | J3 J1 |
| 4 | J3 J2 J4 | 9 | J3 J1 J2 J5 J4 |
| 5 | J2 J4 | | |

The FP-growth algorithm based on a linked list mining in each computer node through Map Reduce computation model

## V. ENHANCED FP GROWTH ALGORITHM DESCRIPTION

EPFP algorithms for mining frequent itemsets in massive small file datasets in detail.

(1) Write a small file processing program—Sequence File. The Sequence File is used to combine all massive small files, which are composed of a huge amount of transaction datasets stored in HDFS, into a hefty transaction data file (transaction database).
(2) Equally split the transaction database into several sub operation databases and then consign them to different nodes in Hadoop huddle. This step is automatically operated by HDFS, when necessary, we canister use the poise command enabling its file system to attain load balancing. [7]
(3) Divide J_list into M groups, denoted as cluster_list (abbreviated as C_list), and disperse group_id for each group sequentially and each C_list contains a set of items.[7]
(4) calculate support count of each item in the transaction database by Map Reduce, and then attain the set of J_list from support count in descending order.[7] [3]
(5) Complete the parallel computing of FP-Growth Algorithm by Map Reduce. The Map function compares the item of each transaction in the sub-transaction database with the item in C_list. If they are same, next distribute the corresponding transaction into the machine associated with C_list. Otherwise, compares to the next item in C_list. Eventually, the independent sub-transaction databases corresponded to C list will be produced. The Reduce function recursively computes the sovereign sub-transaction databases generated in step and then constructs the FP-tree. This stride is similar to the process of customary FP-tree generation, but the difference is a size K maxheap HP which stores frequent pattern of each item.[7]

(6) Amassed the local recurrent item sets generated from each node in the cluster by Map Reduce, and finally get the global frequent item sets

## VI. CONCLUSION

In this paper, it is described that the smaller file processing approach, the EPFP algorithm can diminish memory cost greatly and recover the efficiency of data access, thus avoids recall overflow and reduces I/O overhead. Meanwhile, the EPFP algorithm is migrated to the Map Reduce environment, which can absolute frequent item sets mining efficiently and thus augment the overall recital of FP-Growth algorithm. The experimental results explain that EPFP algorithm can make a breakthrough where PFP algorithm has its defects in handling huge small file datasets, and has a superior speedup and a higher mining efficiency.

## REFERENCES

[1] Khurana K and Sharma S, ―A comparative analysis of association rule mining algorithms, International Journal of Scientific and Research Publications, Volume 3, Issue 5, pp 38-45, May **2013**.
[2] Peng Zhao, "Research Mining Frequent Items Algorithm in Massive High-dimensional Data Sets", Computer Applications and Software, **2012**.
[3] Ahilandeeswari.G, DR.R Manicka Chezian, "A Comparative analysis of Association rule excavating in Big Data Mining Algorithms ", International Journal Of Computer Science and Engineering, Volume 3, Issue 6, pp 82-88,June **2015**
[4] Ms. Dhamdhere Jyoti L., Prof. Deshpande Kiran B. "An Effective Algorithm for Frequent Itemset Mining on Hadoop.", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 8, August **2014**.
[5] Guojun Mao, Lijuan Duan, Shi Wang, Yun Shi, "data mining principles and algorithms (the second edition)", Tsinghua University Press, Beijing, **2007**.
[6] Tom White, "Hadoop: The Definitive Guide, Second Editon", Tsinghua University Press, **2011**.
[7] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, Edward Chang,"Pfp: Parallel Fp-Growth for Query Recommendation", RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems, Pages 107-114 ACM New York, NY, USA ©**2008**.
[8] Ferenc Kovacs and Janos Illes "Frequent Itemset Mining on Hadoop.",IEEE 9th International conference on Computational Cybernetics, Volume 2 Issue 4, June **2013**.
[9] A. Swami, T. Imielienski, R. Agrawal," Mining Association Rules between Sets of Items in Large databases.", ACM Press, pp 207–216, July **1993**
[10] Yang Liu, Maozhen Li, Alham, N.K., Hammoud, S.,Ponraj, M. "Load balancing in MapReduce environments for data intensive applications", Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on,Page(s): 2675 - 2678 ,**2011**.

[11] Ferenc Kovacs and Janos Illes "Frequent Itemset Mining on Hadoop.",IEEE 9th International conference on Computational Cybernetics, Volume 2 Issue 4, June **2013**.

**AUTHORS BIOGRAPHY:**

MS Ahilandeeswari.G received an MCA degree from Anna University,Chennai. In 2006 and 2009 respectively,currently a research scholar at Department of Computer Science,NGM College, Pollachi. Her research interest lies in the area of Data Mining and Big Data Mining

DR. R.Manicka chezian received his M.Sc., degree in Applied Science from P.S.G College of Technology, Coimbatore, India in 1987. He completed his M.S. degree in Software Systems from Birla Institute of Technology and Science, Pilani, Rajasthan, India and Ph D degree in Computer Science from School of Computer Science and Engineering, Bharathiar University,Coimbatore, India. He served as a Faculty of Maths and Computer Applications at P.S.G College of Technology, Coimbatore from 1987 to 1989. Presently, he has been working as an Associate Professor in N G M College (Autonomous), Pollachi under Bharathiar University, Coimbatore, India since 1989. He has published more than 120 papers in various International Journals / Conferences. His research focuses on Network Databases, Data Mining, Distributed Computing, Mobile Computing, Real Time Systems and Bio-Informatics.

 