An Efficient Market Basket Analysis Technique with Improved MapReduce Framework on Hadoop: An E-commerce Perspective

Md. Rezaul Karim[1], Azam Hossain[1], Md. Mamunur Rashid[1], Byeong-Soo Jeong[1], and Ho-Jin Choi[2]

[1] *Department of Computer Engineering, Kyung Hee University, Republic of Korea*
[2]*Department of Computer Science, KAIST, Republic of Korea*

E-mail: {asif_karim, azam, mamun, jeong}@khu.ac.kr; hojinc@kaist.ac.kr

Abstract. Market basket analysis techniques are substantially important to everyday's business decision, because of its capability of extracting customer's purchase rules by discovering what items they are buying frequently and together. But, the traditional single processor and main memory based computing is not capable of handling ever increasing large transactional data. Moreover, due to the presence of null transactions mining performance degrades drastically in existing approaches. Therefore, in this paper an attempt has been taken to remove these limitations. First we remove null transactions and infrequent items from the segmented dataset prior applying our proposed HMBA algorithm using the 'ComMapReduce' framework on Hadoop to generate the complete set of maximal frequent itemsets; since, the maximal frequent itemsets for a particular customer reveals his/her purchase rules. Later on, modified version of the 'sequence close level' is used for counting the distance between a pair of items for mining customer's purchase rules. Experimental results show that our market basket analysis technique is not only increases the performance, but also highly scalable in terms of increasing loads, and can analyze the customer's purchase rules in a reasonable time.

Keywords: ComMapReduce, Market basket analysis, Hadoop, Sequence close level, Maximal frequent itemsets, Customer's purchase rules.

1 Introduction

Market basket analysis is one of data mining approaches to analyze the association of items for the daily buying/selling. The basic idea is to find the associated pairs of items in a store from the transaction dataset. Therefore, to raise the probability of purchasing, to control the stocks more intelligently, and to promoting certain items together the corporate manager of a shop can place the associated items at the neighboring shelf. Thus, he/she can have much better chance to make profits by controlling the order of goods and marketing. Online shopping is another form of market basket whereby, consumers directly buy goods from a seller in real-time over the Internet. Similarly, an online shop, e-shop, web-shop, or virtual stores are the physical analogy of purchasing products in a shopping centre. For extracting and identifying useful information from these large amounts of data, data mining technique is a crucial [11, 12]. This purchase sequence provides a description of the changes in a shopper's preferences over time [11-13]. But, traditional Apriori or FP-tree like methods been proposed for single processor and main memory based computing and is not capable of handling peta/tera bytes of transactions data. To remove this limitation MapReduce and Hadoop based techniques have been introduced [1-11]. Hadoop is the parallel programming platform built on Hadoop Distributed File Systems

for MapReduce computation that processes data as <key, value> pairs. Map and Reduce are two important primitives in the MapReduce framework. In a Hadoop cluster, a master node controls a group of slave nodes on which the Map and Reduce functions run in parallel. The master node assigns a task to slave nodes. An input file passed to Map functions resides on the HDFS on a cluster. Hadoop's HDFS splits the input file into fragments, which are distributed to a pool of slaves for Map/Reduce processing. Despite the scalability and flexibility of existing MapReduce framework it has no communication mechanism among the Master node and Slave nodes. Besides the improved version of existing MapReduce framework the '*ComMapReduce*' [10], gives more flexibility for global communication to reduce the map and reduce time. In this paper, to facilitate the existing Map/Reduce operations, i) first we remove null transactions and infrequent items from the segmented dataset, ii) sort the items in support ascending order then apply the parallel FP-growth to generate the complete set of maximal frequent itemsets using the improved ComMapReduce framework. Later on, for mining customers purchase rules from the dataset we calculate the sequence close level of each maximal frequent itemset found. In brief contributions of this paper can be summarized as follows:

- An improved 'ComMapReduce' framework and Hadoop based market basket analysis technique has been proposed for the first time ever. Our framework not only mines frequent itemsets by means of what items the customers buy most, frequently and together in baskets at a store but also customer's purchase rules. Therefore, our approach is more efficient for analyzing large market baskets. We developed an effective 'HMBA algorithm' which not only mines our desired frequent itemsets but also highly scalable in terms of increasing large dataset.

The rest of this paper is organized as follows: Section II analyzes previous works. Section III describes some basic concepts of sequence close level *CL* (T^k) and problem statements. Section IV describes our proposed framework to mine customer's purchase rules. Section IV illustrates an example to demonstrate our approach. We devised some experimental results at Section V, and finally we conclude at section VI.

2 Related Works and Motivations

2.1 Related Works

Google's MapReduce [5] was first proposed in 2004 for massive parallel data analysis in shared-nothing clusters. Literature [8] evaluates the performance in Hadoop and HBase for Electroencephalogram (EEG) data and saw promising performance regarding latency and throughput. Karim et al. [3] proposed a Hadoop based MapReduce framework for mining DNA sequence dataset and found outstanding throughput and scalability. Woo et al. [1, 2], proposed market basket analysis algorithm that runs on Hadoop based traditional MapReduce framework with transactional dataset stored on HDFS. This work presents a Hadoop and HBase schema to process transaction data for market basket analysis technique. First it sorts and converts the transaction dataset to <key, value> pairs, and stores the data back to the HBase or HDFS. But sorting and grouping of items then storing back it to the original nodes does not take trivial time. Hence, it is not capable to find the result in a faster way; besides this work also not so useful to analyze the complete customer's preference of purchase behavior or rules. Karim et al. [4] proposed a parallel and distributed approach for mining correlated, associated-correlated, and independent patterns synchronously using improved parallel FP-Growth on Hadoop based MapReduce framework. Contrary to these

researches, ComMapReduce [10] adopts efficient communication mechanisms to improve the performance of query processing of massive data in the cloud. This literature proposes an improved version of MapReduce called '*ComMapReduce*'. It maintains the main features of MapReduce, while filtering the unpromising data objects by communicating with the Coordinator node. Three communication strategies, Lazy, Eager and Hybrid, are presented to effectively implement query processing applications of massive data in MapReduce framework. Experimental results demonstrate that the performance of ComMapReduce is beyond of MapReduce dramatically in all metrics. More details about ComMapReduce have been given at section 3.2. Literature [13], addressed the problem of mining e-shopper's purchase rules based on k-trees sequential patterns by finding maximal frequent itemsets [13, 14]. First it constructs k-tree sequential itemsets using Apriori algorithm then mine purchase rules based on the calculation of sequence close level for each maximal frequent itemset found. But this works has three limitations: firstly, it has no proper bound due the incorrectness of the 'sequence close level' formula; secondly it leads a tedious problem due to multiple times original dataset scanning. Thirdly it cannot handle large dataset like peta or tera bytes of transactional data.

2.2 Motivations and the Screening of the Null Transactions

Relational DBMS has been a core to store data for business, and research but there are some critical issues to handle huge volumes of data. For Tera or Peta-bytes of data, RDBMS has scalability problems in partitioning and availability issues in replication. At RDBMS, it is almost impossible to read or write concurrently for transactional and analytical data. Besides, it is slower in reading the data from physical storages than from the latest fast network. Most Apriori or FP-tree like approaches have been proposed for the single processor and main memory based system on the traditional RDBMS. Hence, this limited hardware resources are not capable of handling such a large business oriented dataset and obviously are not efficient. Despite some merits over RDBMS, distributed data mining approaches also need lots of message passing and I/O operations since the distributed nodes has to share/pass the needed data. This poses impractical to use distributed system for large dataset mining. Advantageous thing of MapReduce over he distributed system is that it only needs to share/pass the support of individual candidate itemset rather passing itself and communication cost is lower than that of distributed environments [3, 4]. To avoid many sequential access to the dataset, and to make Map/Reduce operation faster; first we remove *'null transactions'* and infrequent items from the segmented dataset then we use balanced parallel FP-growth to mine maximal frequent itemsets using the *'ComMapReduce'* framework. Finally, we use the concept of modified *'sequence close level'* proposed at [16] for counting distance between a pair of items to analyze complete customer's purchases rules for market basket analysis. A null transaction is a transaction that does not contain any itemsets being examined. Typically, the number of null transactions can outweigh number of individual purchases because, for example, many people may buy neither milk nor coffee, if these itemsets are assumed to be two frequent itemsets. Therefore, performance degrades drastically especially when the transactional datasets has many null transactions. Since, large datasets typically have many null transactions, it is important to consider the null-invariance property [16] for pattern evaluation for market basket analysis. Therefore, finding null transactions and later eliminating them from future schemes is the initial part of this proposed MapReduce framework. This saves a lot of precious computation time while

performing Map/Reduce phases. It is quite possible to find null transactions by finding those transactions that don't appear against any frequent 1-itemset.

3 Backgrounds and the Problem Statements

3.1 Overview of the ComMapReduce

In order to optimize the intermediate data objects of the original MapReduce framework, ***'ComMapReduce'*** is an efficient lightweight communication framework [10] extending MapReduce by pruning unpromising data objects of query processing dramatically. ComMapReduce inherits the basics of MapReduce and takes HDFS to store the original data objects and the final results of each application. In ComMapReduce, the Master node and a Slave node play the same roles as in the original MapReduce framework; in addition the Coordinator node is deployed to share global communication information to enhance the performance of the original MapReduce without sacrificing the availability and scalability. The Coordinator node communicates with the Mappers and Reducers with simple lightweight communication mechanisms to filter the unpromising data objects. The Coordinator node receives and stores some temporary variables, and then generates ***filter value***s of the query processing applications. Each Mapper obtains a ***filter value*** from the Coordinator node to prune its data objects inferior to the other ones. After filtering, the output of Mappers and the input of Reducers both decrease dramatically. Therefore, performance of massive data on ComMapReduce framework is enhanced.

3.2 Mining Maximal Frequent Itemsets

Let a set of distinct items $\boldsymbol{I} = \{i_1, i_2 \ldots i_n\}$ and $\boldsymbol{n}$ is the number of distinct items. A transactional database $\boldsymbol{T} = \{t_1, t_2 \ldots t_N\}$ is a set of $\boldsymbol{N}$ transactions and $|\boldsymbol{N}|$ is the number of total transactions. A set $\boldsymbol{X} \subseteq \boldsymbol{I}$ is called a pattern or itemset. If $\boldsymbol{X} \subseteq \boldsymbol{t}$, it is said that $\boldsymbol{X}$ occurs in $\boldsymbol{t}$ or $\boldsymbol{t}$ contains $\boldsymbol{X}$. If $\boldsymbol{support\ (X)} \geq \boldsymbol{min\_sup}$ we say that $\boldsymbol{X}$ is a frequent itemset. If $\boldsymbol{X}$ is a frequent itemset and no superset of $\boldsymbol{X}$ is frequent, we say that $\boldsymbol{X}$ is a maximal frequent itemset.

Table 1. A Transactional Database

TID	Itemset (Sequence of items)
10	A, B, C, F
20	C, D, E
30	A, C, E, D
40	A
50	D, E, G
60	B, D
70	B
80	A, E, C
90	A, C, D
100	B, E, D

So, given a transactional database DB and a user defined minimum support threshold '***min_sup'***, now the problem of mining maximal frequent itemsets is to find the complete set of maximal frequent itemsets whose ***support*** is no less than the ***min_sup*** threshold. For example, in Table 3, the occurrences of itemsets "***CD***", "***DE***" and "***CDE***" are 3, 3 and 2 respectively. If the ***min_sup*** is 2, all of these are frequent itemsets. But, "***CD***" is not a maximal frequent itemset because its super itemset "***CDE***" is also a frequent itemset.

3.5 Sequence close level to analyze the customer's purchase rules

In this subsection we present the idea of modified '*sequence close level*' proposed at [19]. According to e-shopper's purchase items, a record for transactional data usually is depicted as sequence pattern. In addition, most of the counting support approaches count only the frequency of occurring itemsets and do not address the distance between a pair of items in a transactional database [14]. Hence, for the complete market basket analysis we also consider this issue. The length between two items in a transaction is called distance and distance of the same pair of items in different sequences varies. For example, consider the filtered database presented in Table 3; in transaction 10; item C and D are neighbors, however C and D are far from each other in transaction *20*.The distance between two items in a sequence assists in determining the closeness of their relationships. If two items are far from each other, their relationship is loose. On the contrary, their relationship is close if two items are neighbors. If two items are neighbors, the distance between them is set to be 1.To estimate the significance of the distance for sequence, sequence close level is defined in [28] as follows:

$$CL(T^k) = \sum_{i=1}^{k-1}\left(\frac{1}{d_i}\right) \Big/ k-1$$

Here k is the length of k –maximal frequent itemset T^k; d_i is the distance between two items, and

$$d_i = p(t_i+1) - p(t_i), \qquad (2)$$

Where, $p(t_i)$ is the position of ith item in transaction t_i. The value of $CL(T^k)$ is between 0 and 1; but when $CL(T^k)$ is equal to 1, all items in maximal frequent itemsets in a transaction are close to each other and the sequence is very important. If the value is close to 0, the sequence has lower significance. When $CL(T^k)$ is equal to 0, the sequence does not contain any useful information about the customer's purchase rules or purchase history.

Example 1: For the filtered database presented in Table 3, pattern {A, C, D} is a maximal frequent pattern occurs in TID 30 and 90 and the sequence close level $CL(T^k)$ can be calculated by formula (1) as follows: The distances from A to C in TID 30 is d_1=1 and from C to D is 2; so we have,

$$CL(T^2) = (1/1)+(1/2) / (3-1)=0.75$$

In TID 90, distances from A to C is d_1=1 and C to D d_2=1, and, $CL(T^2)$ =(1/1)+ (1/1) / (2-1) = 1 ; and the maximal frequent itemset {A, C, D} does not contain equal importance in terms of sequence close level, that is a customer is more likely to purchase items A, C and D in TID 90 but an customer first buy item A, C then another item E and item D in transaction 30. But this measurement has one limitation. For example in Table 3, {A, C, E} is a maximal frequent itemset that occurred at TID 30 and 80 and using the above formula of distance calculation we cannot determine the sequence close level. Therefore to remove this limitation Karim at el. [16] proposed the following modification of distance calculation for the 'sequence close level' as follows: If the occurrence of items in a maximal frequent pattern is consecutive then the distance can be calculated as follows:

$$d_i = |p(t_i) - p(t_i+1)| \qquad (3)$$

Now, in both case we can use formula (1) and (3) for the calculation of sequence close level even if the occurrence of items in a maximal frequent pattern is consecutive. Therefore, from example 1 it is clear that the bigger value of $CL(T^k)$ is the more desirably online customers purchase the items in transaction T^k.

4 Proposed Approach

4.1 Proposed Programming Model and the HMBA Algorithm

The transactional dataset is splited into segments automatically after stored on the HDFS. The Hadoop components perform job execution, file staging and workflow information storage on the segmented dataset [1-9]. We used Hadoop and improved *'ComMapReduce'* based technique and follow the data placement strategy indicated at [11]. Figure 1 shows the HMBA algorithm for market basket analysis, Figure 2 shows workflow of the proposed framework, and Table 2 shows the I/O schemes to the proposed MapReduce framework. Operations go as follows: Master node assigns tasks to idle Slave nodes for map operation. The Coordinator node repeatedly communicates using 'filter *values*' with Mappers/Reducers. A assigned worker node i) scans the transactions in its assigned segment(s) as <ID, itemset> pairs, ii) removes null transactions and infrequent items, iii) sorts the items of a transaction in support ascending order to avoid the duplicated keys and iv) the parallel FP-growth is applied to produce the output as <CPB, CT_i> pairs after finding the prefixes. These values are stored on the local disk of Slaves as intermediate results. HDFS performs shuffle and sorting operations on the conditional transactions using the key value (i.e. CPB) and stores the results as <conditional pattern base, conditional database> pairs. Now, the Master node assigns tasks to idle worker nodes for the reduce operation; then a reducer takes input as <CPB, CDB_i> pairs and reduces the output as <CPB, conditional FP-tree> pairs, this is the end of reduce phase 1. In reduce phase 2 a Slave node takes the output of reduce phase 1 as <CPB, $CFPT_i$> pairs and reduces the result set as < frequent itemset, support> pairs in step 1 and <maximal frequent itemset, support> pairs in step 2.

4.2 Analysis of the HMBA Algorithm

To read each transaction in assigned segments(s) on Hadoop, time complexity is *O (n)* where *n* is the number of items in a transaction. Then, prefix creation on the sorted items has the complexity of O (mn), where m is the number of prefixes that occurs together for n items in a transaction using at most (n-1) keys. Excluding the scanning time the complexity of merging on the intermediate result is *O (nlog n)* on merge sort and the complexity of message passing with Coordinator among Mappers/Reducers is almost constant. Thus, the time complexity of each Mapper is *O (n+mn)*. So, much better than the previous approaches [1,2] also now it has to scan less items/transaction due to the removal of null transactions and infrequent items in each segment(s).

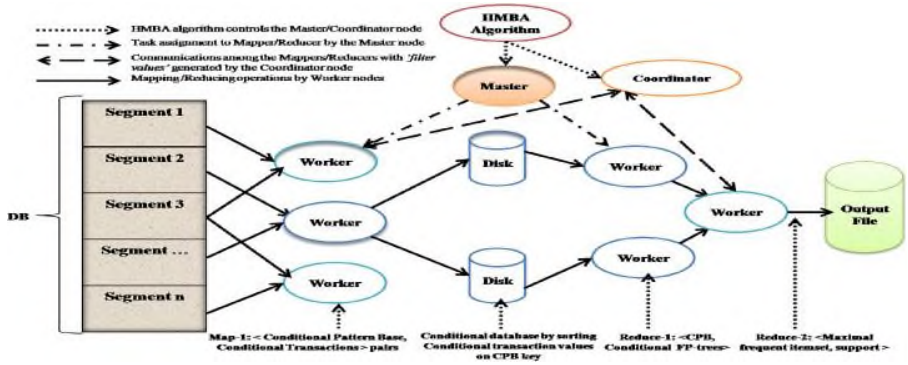

Figure 1. Workflow of the proposed Framework

5 An Illustrative Example

Suppose, 'min_sup' and 'min_seq_close_lev' are two 'filter values' generated by the Coordinator node. And let, the dataset at Table 1 is splited into two segments 01 and 02, where, TIDs 10, 20, 30, 40, 50 are in segment 01; and TIDs 60, 70, 80, 90, 100 are on segment 02. Suppose, the Master node assigned segment 01 and 02 to worker nodes 1 and 2 respectively. Now, worker 1 and 2 scans the transactions in segments 01 and 02 as <*itemset, ID* > pairs using the balanced FP-growth and produces the output as <CPB, CT_i> pairs. In segment 01, transactions 10, 40 and 50; and transaction 70 at segment 02 are the null transactions and are not considered for mining (more details can be found at literature [19]. In segment 01, transactions 10, 40 and 50; and transaction 70 at segment 02 are the null transactions and are not considered for mining (more details can be found at literature [19]. In segment 01, transactions 10, 40 and 50; and transaction 70 at segment 02 are the null transactions and are not considered for mining (more details can be found at literature [19]. Since, a null transaction with just 1 item has no contribution for correlation/association rule mining in market basket analysis; hence removed prior mining. On the other hand, items F & G in segment 01 are infrequent, consequently are pruned. Table 3 shows the segmented dataset. The HDFS automatically performs the merge sort on the conditional transactions using the key values E, A, B, C, and D. In the reduce phase 1 worker nodes take input as <CPB, CT_i> pairs and reduce output as <CPB, conditional FP-tree> pairs as shown in Figure 3 and we use to optimize the map operation using 'filter value' 2.

Table 3: The Segmented Dataset

Segments	TID	Itemset
Segment 01	10	A, B, C, F
	20	C, D, E
	30	A, C, E, D
	40	A
	50	D, E, G
Segment 02	60	B, D
	70	B
	80	A, E, C
	90	A, C, D
	100	B, E, D

In reduce phase 2 slave nodes take the output of reduce phase 1 as input and reduce the result set as (i) < FI, support> pairs in step 1 and (ii) <MFI, support> pairs as shown by Figure 3. After that the Coordinator node send another filter value 0.60 as minimum sequence close level to the reducer nodes to calculate CL($T^{k)}$ for each maximal frequent itemset. . It is now quite possible for one Slave or worker node to do this since; the number of maximal frequent itemset is much smaller than the frequent patterns set. In Figure 3 we have found 4 maximal frequent itemsets to analyze the sequence close level. To understand the importance of maximal frequent itemsets in a transaction, calculation of *CL* (T^k) can made in any one of the Slave nodes which has the empty task slot for each maximal frequent itemset found. Results of *CL* (T^k) in Table 4 indicate that for a maximal frequent itemset, the bigger value of *CL* (T^k), the closer all items in the transactions, and more desirably customers purchase the items according to k-trees sequential pattern [13]. For example, purchase sequence like {A, C, E}, {A, C, D} in terms of TID 30; {B, D} in terms of TID 60 are very important. Moreover, {A, C, D} in terms of 90; {A, C, E} in terms of TID 80; and {C, D, E} in terms of TID 30 are less important rules compared to others.

Table 4: CL (T^k) calculation for maximal frequent itemsets

Maximal Frequent Itemsets	CL(T^k)	TID	Transactions
{B, D}	1	60	B, D
	0.5	100	B, E, D
{A,C, D}	1	30	A, C, E, D
	0.75	90	A, C, D
{A, C, E}	1	30	A, C, E, D
	0.75	80	A, E, C
{C, D, E}	1	20	C, D, E
	0.75	30	A, C, E, D

Input: i) A transactional database on HDFS ii) '*min_sup*' threshold (iii) '*min_seq_close_lev'* threshold.
Output: i) Complete set of maximal frequent itemsets that satisfy the '*min_sup'* ii) The complete set of purchase rules that satisfy the '*min_seq_close_lev'* threshold.
Preprocessing: The Master node splits the dataset into smaller segment $D_{\alpha i}$ when stored on the HDFS. The Coordinator node repeatedly sends light weight communication value using the *'filter value'* during the Map/Reduce operations. A mapper scans its assigned segments(s) and i) remove null transactions & infrequent 1-itemsets ii) sorts the items of a transaction in support ascending order before applying the map operation.
Mapper ($D_{\alpha i}$) {
//Map: Slave nodes scan the assigned segment(s) and map the output as $<CPB_i, CT_i>$ pairs; CPB are considered as keys.

1. *for each Transaction T_i, i=1 to N*
2. **Call** Balanced _Parallel_ FP (TID, I) to generate conditional transactions based on conditional pattern bases.
3. **Write** the output on the local disks as $<CPB, CT_i>$ pairs.

}
Sort ($<CPB, CT_i>$, key) // Sorts the intermediate output and write the result set on
the local disks.
{

4. **Combine** the conditional transactions as list according to the keys CPB as the conditional databases.
5. **Sort** the list using merge sort and store the result on the local disk as $<CPB, <CT_1, CT_2....CT_i> >$ pairs.
   Return the sorted list of $<CPB_i, <CT_1, CT_2... CT_i> >$ pairs. // these are used as input to the reduce phase 1.

}
Reducer ($<CPB, <CT_1, CT_2...CT_i> >$ pairs**)**
{
//Reduce 1: Assigned nodes take input as $<CPB, <CT_1, CT_2...CT_i> >$ pairs & find the complete set of frequent itemsets.

8. **Find** frequent itemset from the conditional transactions list. Suppose
9. Y is an item in the conditional database.
10. *If sup(Y)* $\geq$ *min_sup,*
    CPB.Y is a frequent itemset.
11. *else*
12. **Prune** Y from the list. //infrequent item in the conditional transactional list
13. **Write** result as <FI, support > pairs on the local disks.

//Reduce 2: Worker nodes find the complete set of maximal frequent itemsets from the set of frequent itemsets**.**

```
//Step-1:   Finding the complete set of maximal frequent itemset
14.       If CPB.Y is a frequent itemset
15.           If no superset of CPB.Y in the frequent itemset list is
              frequent
16.               CPB.Y is a maximal frequent itemset
17.       Write the complete set of the maximal frequent itemsets on the
          local disks as <MFI, Support> pairs.
//Step-2:   Finding the complete set of purchase rules. Suppose CPB.Y is a
          maximal frequent itemset
18.       for each CPB.Y i=1 to N
19.           If (CL(T_i^k)> min_seq_close_lev
20.               CPB.Y is an important purchase rule.
21.       else
22.               CPB.Y is not an important purchase rule.
23.       }
```

Figure 3. The HMBA algorithm using ComMapReduce on Hadoop

I/O to the Map Phase for Segment 01 and 02

Segment 01 ID	Itemset	Map Input Key: TID, Value: Itemset	Map Output Key: CPB, Value: CT_i
10	C D E	< 20, DCE >	E: DC, C:D, D: Φ
20	AC E D	< 30, DCEA >	A:DCE, E:DC, C: D, D: Φ
60	B D	< 60, DB >	B: D, D: Φ
Segment 02 ID	Itemset	Map Input	Map Output
		Key: TID, Value: Itemset	Key: CPB, Value: CT_i
80	C E A	< 80, CEA >	A: CE, E:C, C: Φ
90	D C A	< 90, DCA >	A:DC, C:D, D: Φ
100	D E B	< 100, DEB >	B: DE, E: D, D: Φ

Sorting of intermediate values on HDFS on key values-CPB, using Merge Sort/ Input to Reduce Phase-1

Key (Conditional Pattern Base)	Value (Conditional Databases)
E	{ DC, DC, C, D }
A	{ DCE, CE, DC }
B	{ D, DE }
C	{ D, D, D }

I/O to the Reduce Phase 1 & 2

Reduce-1: Input	Reduce-1: Output /Reduce -2 Input	Reduce-2 Output : Step-1	Reduce-2 Output : Step-2
Key: CPB, Value: CDB,	Key: CPB, Value: CFPT	Key: Frequent Itemset Value: Support	Key: Maximal Frequent Itemset Value: Support
< E, (DC, DC, C, D) >	E: { (DC:2, D: 3, C: 3) }	< EDC, 2 >, < ED, 3>, < EC, 3 >	CDE:2,ADE:3, ACE:2, BD:2
<A, (DCE, CE, DC) >	A: { (DC:2, DE: 3, CE: 2, C:3, D:2, E:2) }	< ADC, 2>, < ADE, 3>, < ACE, 2>, < AC, 3>, <AD, 2>, < AE, 2>	
<B, (D, DE) >	B: {(D:2)}	< BD,2>	
< C, { D, D, D } >	C: { (D:3) }	< CD, 3 >	

Figure 3. The Map/Reduce operations and results using ComMapReduce

6 Experimental Results

We used Hadoop 0.20.0, running on a cluster of 10 commodity PCs in a high speed Gigabit LAN network with one PC as the Master node and the Coordinator node, and other 8 are as Slave nodes. Master and the Coordinator node have Intel Quad Core 2.66GHZ CPU with 4GB of RAM and each Slave has Intel Core 2 duo 2.60GHz processor with 2GB of RAM. The balanced parallel FP-growth

[9] was modified in Java using MapReduce library functions for the HMBA algorithm and configured HDFS on Ubuntu-11.04. We apply our HMBA algorithm on practical and randomly generated datasets. The practical 'Connect-4', Mushroom and Agaricas datasets have been downloaded from http://rchive.ics.uci.edu/. Connect-4 contains 135,115 transactions, lengths 8~35, unique items are 76 and size of 200MB. On the other hand Mushroom and Agaricas has 78,265 and 92,653 transactions with size of 120MB and 170MB respectively. For the first experiment, we used three different transaction files 800MB (13M transactions), 1200MB (25M transactions) and 1500MB (30M transactions) with 42 unique items in each and executed on 3, 5, and 8 computing nodes, respectively. For the second experiment, datasets were splited into segments after stored on the HDFS across 2, 3 and 4 nodes; for this we follow the load distribution indicated at [6]. Since, the previous works [1, 2] did not observe any significant difference in execution time when the data size is smaller than 800MB; hence, we used transaction files larger than or equal to 800MB and for this purpose we directly copied these data files on the HDFS without segmentation. We set min_seq_close_lev to 0.65 and vary min_sup 1 to 10. In the first experiment we compared the running time of our framework (HMBA algorithm) and the previous approach (MBA algorithm) for Connect-4, Mushroom and Agaricas datasets. Figure 5 shows the measured execution time of map and reduce phase as the performance metric with two filter values as min_sup and min_seq_close_lev. We can observe that when the data size is larger like 1200 or 1500MB, the execution became much faster to compute the frequent and maximal frequent itemset than the previous approaches for 5 and 8 computing nodes. Although, it seems to have linear speedup of the MBA algorithm [1, 2] but since, first it sorts and group the dataset for the <key, value> pairs explicitly; for this consequence a lots of candidates are generated for making groups, and then copy dataset to the original locations again, so the execution time is not satisfactory. Fig. 5 compares the execution time for our approach and the previous approach for the randomly generated transactional datasets. Also for the previous works the execution time linearly increased but, it reached the maximum parallelization at the instances around 10 and 15 nodes. But, for our framework since we are applying the load distribution as described by the literature [6], therefore we can scale our framework for more nodes for the parallelization. In the second experiment we measured the execution time of our HMBA algorithm for the map/reduce operations on practical Connect-4, Mushroom and Agaricas datasets as shown by Table 6.

Table 5 Execution time (Map/Reduce) on randomly generated data files

Type / #Nodes	Time(ms) 800MB	Time(ms) 1200MB	Time(ms) 1500MB
3	285,218	411,236	516,918
5	180,657	235,897	337,231
8	150,123	172,963	231,847

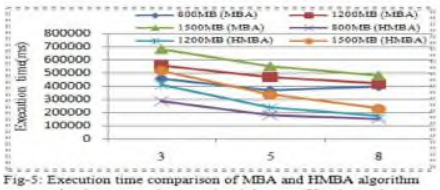

Fig-5: Execution time comparison of MBA and HMBA algorithm on randomly generated transactional datasets (X-axis #nodes).

Table 6 Execution time (Map/Reduce) for the Connect-4, Mushroom and Agaricas dataset

Type / #Nodes	Time(ms) Connect-4	Time(ms) Mushroom	Time(ms) Agaricas
2	5,500	2,589	3,574
3	3,872	1,978	3,085
4	2,182	1,269	1,642

7 Conclusions

Advanced implementations of market basket analysis leverage near-instant results to encourage interactive analysis, enabling retailers to drill down into customer's buying patterns over time. Our

proposed market basket analysis technique using the '*ComMapReduce*' framework extends and improves the limitations of previous approaches for massive market basket data.

Acknowledgements

This work was supported by a grant from the NIPA (National IT Industry Promotion Agency, Korea) in 2012 (Global IT Talents Program).

References

1 J. Woo, S. Basopia, and S.H. Kim, "Market Basket Analysis Algorithm with NoSQL DB HBase and Hadoop", In proc. ofthe 3rd International Conference on Emerging Databases (EDB2011), pp: 56-62, Incheon, Korea, Aug 25~27, 2011.

2 J.Woo, S. Basopia, and S. Ho Kim, "Market Basket Analysis Algorithm with Map/Reduce of Cloud Computing", In proc. of the Intl. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA 2011), Las Vegas, USA, July 18-21, 2011.

3 M.R. Karim, M.A. Hossain, M.M. Rashid, B.S. Jeong and H.-J. Choi, "A MapReduce Framework for Mining Maximal Contiguous Frequent Patterns in Large DNA Sequence Datasets", IETE Technical Review, Vol .29, No. 2, March-April, 2012.

4 M. R. Karim, M.M. Rashid, M.A. Hossain and B.S. Jeong, "A Parallel and Distributed Programming Model for Mining Correlated, Associated-Correlated and Independent Patterns in Transactional Databases Using MapReduce on Hadoop", In Proc. ofthe 6th Intl. Conf. on CUTE, 2011, pp: 271-276, Seoul, Korea.

5 J.Dean and S.Ghemawa,"MapReduce: Simplified Data Processing on Large Clusters", pp. 137–150 OSDI, 2004.

6 J. Xie, S. Yin, X. Ruan, Z. Ding, Y.Tian, J. Majors, A. Manzanares, and X. Qin:"Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters" IEEE Intl. Symp. IPDPSW, 2010.

7 Apache-Hadoop: http://hadoop.apache.org/

8 H.Dutta, A.Kamil, and J. Demme, "Distributed Storage of Large Scale Multidimensional Electroencephalogram Data using Hadoop and HBase", Book Chapter in Grid and Cloud Database Management, Springer, 2011.

9 L.Zhou, Z.ong, J.Z.ng, and S.Feng; "Balanced Parallel FP-Growth with MapReduce" IEEE Youth Conf. on Info. Computing and Telecom, 2010.

10 L.Ding, G.Wang,&S.Huang,"ComMapReduce: An Improvement of MapReduce with Lightweight Communication Mechanisms", LNCS, V.7238/2012, 303-319.

11 H.Song, and J.Kim: "Mining change of customer behavior in an Internet shopping mall", Expert System with Applications, 157–168, 2001.

12 L.Jian, and W.Chong, "Prediction of E-shopper's Behavior Changes Based on Purchase Sequences" Intl. Conf. on Artificial Intelligence and Computational Intelligence (AICI), 2010.

13 C.Wang, J.Liu, and Y.Wang: "Mining E-shopper's Purchase Rules Based on K-trees Sequential Patterns", 7th Intl. Conf. on Fuzzy Systems Knowledge Discovery (FSKD), 2010.

14 H. Wang, C. Hu, and Y. Chen, "Mining Maximal Itemsets Based on Improved FP-lattice structure and Array Technique" Proc. ofthe 2nd Intl Conf. on Future Computer and Communication, 2010.

15 K. Gouda, M. Zaki, "A New Method for Mining Maximal Frequent Itemsets," In Proc. ofthe World Congress on Engineering London, U.K, 2008.

16 M. R. Karim, J.-H. Jo and B.-S. Jeong, "Mining E-shopper's Purchase Behavior Based on Maximal Frequent Itemsets: An E-commerce Perspective", Proc. of the 3rd Intl. Conf. on Information Science and Applications (ICISA, 2012), May: 23-25, 2012, Suwon.