An Efficient Distributed Programming Model for Mining Useful Patterns in Big Datasets

Md. Rezaul Karim, Chowdhury Farhan Ahmed[1], Byeong-Soo Jeong and Ho-Jin Choi[2]

Departments of Computer Engineering, Kyung Hee University, Korea, [1]Computer Science and Eng., University of Dhaka, Bangladesh, [2]Computer Science, KAIST, Korea

Abstract

Mining combined association rules with correlation and market basket analysis can discover customer's buying purchase rules along with frequently correlated, associated-correlated, and independent patterns synchronously which are extraordinarily useful for making everyday's business decisions. However, due to the main memory bottleneck in single computing system, existing approaches fail to handle big datasets. Moreover, most of them cannot overcome the screenings and overhead of null transactions; hence, performance degrades drastically. In this paper, considering these limitations, we propose a distributed programming model for mining business-oriented transactional datasets by using an improved MapReduce framework on Hadoop, which overcomes not only the single processor and main memory-based computing, but also highly scalable in terms of increasing database size. Experimental results show that the technique proposed and developed in this paper are feasible for mining big transactional datasets in terms of time and scalability.

Keywords

Associated pattern, Business-oriented data, ComMapReduce, Correlated pattern, Independent pattern, Null transactions, Transactional database.

1. Introduction

Data mining is defined as the process of discovering significant and potentially useful patterns in large volume of data [1]. In data mining, a pattern is a particular data behavior, arrangement, or form that might be of a business interest, even though we are not sure about that yet. The well-known algorithm for finding association rules [2] in large transaction databases is Apriori [3]. One objective of association rule mining is to discover correlation relationships among a set of items. One difficulty is how to select a proper interestingness measure that can effectively evaluate the association degree of patterns, as there is still no universally accepted best measure for judging interesting patterns [4].

On the other hand, correlation mining is much effective because of the large number of correlation relationships among various kinds of items in the market baskets. However, an independent pattern might have much more probability than a correlated pattern to be a novel paired or grouped items even if they have same support for the sake of the downward closure property of independence [5,6]. Moreover, combined mining of association rules with correlation can discover frequently correlated, associated-correlated, and independent patterns synchronously, which is useful for making everyday's business decisions. Therefore, to raise the probability of purchasing, to control the stocks more intelligently, and to promote 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. It also allows the retailers to quickly and easily look the size, contents, and value of their customer's market baskets [7].

Based on the concept of strong rules [8,9], R. Agrawal *et al*. [3] introduced the association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale systems in supermarkets and in recent market concept of super-shop is very popular among the people since these shops keep almost everything according to customers' preferences and very often these super shops have lots of branch across the country, and the number of transactions and purchases are also huge. A set of e-shoppers known as neighbors, who have exhibited similar behavior in the past and in web access sequences [10], can be found through calculating the correlations among e-shoppers [11,12].

Therefore, to predict e-shoppers or customer's purchase behavior changes, an organization's management first identifies target e-shoppers who share similar preferences for products and looks for those products that target e-shoppers are most likely to purchase. Besides, in a business-oriented data, customer's purchase rules

can be examined by the maximal frequent patterns or items since the maximal frequent patterns reveal his/her purchaser rules [2,12].

In this paper, to remove the drawbacks of RDBMS and main memory–processor-based computing and to facilitate the existing Map/Reduce framework [3,13-22], (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 [16] to generate the complete set of frequent itemsets with the improved ComMapReduce framework [23] on Hadoop and based on some constraints/thresholds, we generate the complete set of customer's purchase rules by means of maximal frequent itemsets along with the useful and interesting patterns. In brief, contributions of this paper can be summarized as follows:

- An improved framework for analyzing business-oriented transactional datasets by using the ComMapReduce/Hadoop-based technique has been proposed for the first time
- Our proposed framework mines not only frequent itemsets by means of what items the customers buy most frequently and together in baskets, but also customer's purchase rules by using the correlation relationship of association/independence. Therefore, our approach is more efficient for analyzing large market baskets and business-oriented datasets
- Along with customer's purchase rules, we mined some important and interesting patterns simultaneously which are as follows: Frequent patterns, correlated patterns, associated patterns, associated-correlated patterns, and independent patterns
- The proposed framework can handle massive dataset which is a burning issue of recent times
- To make the framework effective, we developed an effective algorithm namely " Associated-Correlated-Independent (ACI) algorithm" which is highly scalable in terms of increasing data loads.

This paper is organized as follows: Section 2 describes some related works on correlated pattern mining and handling big datasets by using the MapReduce framework. Section 3 describes the background study and the motivation behind this research work. Section 4 represents the problem formulation and some related preliminary definitions. Section 5 represents our proposed MapReduce framework and the ACI algorithm. Section 6 represents experimental results. Conclusions are presents in section 7. In this paper, we use the terms: "itemsets" and "patterns"; "database" and "datasets" interchangeably.

2. Related Works

2.1 Related Works on Correlated Pattern Mining

Many research works have been done in the field of correlated frequent pattern mining. Among them, Miccinski [24] introduced three alternative interestingness measures, called any-confidence, all-confidence, and bond for mining associations.

Y.K. Lee *et al.* [24] used the concept of all-confidence to discover interesting patterns, although both of them defined a pattern which satisfies the given minimum all-confidence as a correlated pattern. Later on, B. Liu *et al.* [25] used contingency tables for pruning and summarizing the discovered correlations. In this paper, a new interestingness measure corr-confidence is proposed for correlation mining.

Z. Zhou *et al.* [5] mines all independent patterns and correlated patterns synchronously in order to get more information from the results by comparing independent patterns with correlated patterns. Moreover, an effective algorithm is developed for discovering both independent patterns and correlated patterns synchronously, especially for finding long independent patterns by using the downward closure property of independence. They also combine association with correlation in the mining process to discover both associated and correlated patterns at literature [15]. A new interesting measure corr-confidence was proposed for rationally evaluating the correlation relationships. This measure not only has proper bounds for effectively evaluating the correlation degree of patterns, but also is suitable for mining long patterns. Actually, mining long patterns is more important because a practical transactional database may contain a lot of unique items. However, none of the above mentioned works dealt with the problems/overhead of null transactions and big dataset mining.

2.2 The MapReduce Framework for Handling Big Datasets

Google's MapReduce [17] was first proposed in 2004 for massive parallel data analysis in shared-nothing clusters. Literature [18] evaluates the performance in Hadoop/HBase for Electroencephalogram (EEG) data and saw promising performance regarding latency and throughput. Karim *et al.* [19] proposed a Hadoop/MapReduce framework for mining maximal contiguous frequent patterns (which was first introduced at literature [26] in RDBMS/single processor-main memory based computing) from the large DNA sequence dataset and showed outstanding performance in terms of throughput and scalability.

Literature [20] proposes a MapReduce framework for mining-correlated, associated-correlated and independent patterns synchronously by using the improved parallel FP-growth on Hadoop from transactional databases for the first times ever. Although it shows better performance, however, it also did not consider the overhead

of null transactions. Woo *et al*. [21,22], 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. However, 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.

3. Background and Motivations

3.1 Motivations

Most data mining algorithms are based on object oriented programming and these algorithms have been proposed for single processor and main memory based machine. For many years, the 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 storage than from the latest fast network. For example, to read 1 TB of data with 10 Mbps network speed, it takes about 28 hours; however, if there are one hundred 10 GB datasets with 10 Mbps network bandwidth each, it takes about 17 minutes. It shows that distributed clustered DBs are much faster [21,22]. 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 for mining and analyzing and so, obviously are not efficient.

On the other hand, with the rising of parallel processing, parallel data mining have been well investigated in the past decade. Especially, much attention has been directed to parallel association rule mining. Traditional parallel and distributed data mining work assumes data is partitioned and transmitted to the computing nodes in advance. However, it is usually the case in which a large database is generated and stored in some station. Therefore, it is important to efficiently partition and distributes the data to other nodes for parallel computation. In this application, the workload is partitioned database segments of transactions. 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. Moreover, in this environment there needed lots of message passing and I/O operation since the distributed nodes have to share and pass the data needed [19,27]. This poses very impractical to use distributed system for large datasets mining. Contrary, of these and the RDBMS, Hadoop and the MapReduce [13,14,16] is very suitable platform to mine these sorts of datasets. In addition, using MapReduce on Hadoop only needs to share and pass the support of individual candidate itemsets rather passing whole dataset itself. Therefore, communication cost is low compared to the distributed environments.

3.2 Screenings of the Null Transactions

One of the critical problems is the presence of null transactions in the business oriented transactional datasets. A null transaction is a transaction that does not contain any itemsets being examined [4,12]. 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 [12] for pattern evaluation for market basket analysis and accurate analyzing of business oriented data.

Therefore, finding null transactions and later eliminating them from future schemes is the initial part of this proposed MapReduce framework. Consider an instance that, an electronic shop has 100 transactions where 20% are null transactions. FP-tree or any other related methods would scan all the 100 transactions while, our proposed approach attempts to reduce it to 80% by considering just the valid 80 transactions after screening the 20 null transactions. 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.

3.3 MapReduce, Hadoop, and Hadoop's Distributed File System

The MapReduce programming model was proposed by Google to support data-intensive applications running on parallel computers like commodity clusters [17]. Two important functional programming primitives in MapReduce are Map and Reduce. The Map function is applied on application specific input data to generate a list of intermediate < key, value > pairs. Then, the Reduce function is applied to the set of intermediate pairs with the same key.

Typically, the Reduce function produces zero or more output pairs by performing a merging operation.

All the output pairs are finally, sorted based on their key values. Programmers only need to implement the Map and Reduce functions, because a MapReduce programming framework can facilitate some operations like grouping and sorting on a set of < key, value > pairs. In the line with, the Hadoop is a successful implementation of the MapReduce model [13,14,16-22]. The Hadoop framework consists of two main components: The MapReduce language and the Hadoop's Distributed File System (or HDFS for short). The Hadoop runtime system coupled with HDFS manages the details of parallelism and concurrency to provide ease of parallel programming with reinforced reliability.

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 a slave node that has any empty task slot. Typically, computing nodes and storage nodes in a Hadoop cluster are identical from the hardware's perspective. Such a homogeneous configuration of Hadoop allows the MapReduce framework to effectively schedule computing tasks on an array of storage nodes where data file are residing, leading to a high aggregate bandwidth across the entire Hadoop cluster. An input file passed to Map functions resides on the Hadoop distributed file system on a cluster. Hadoop's HDFS splits the input file into even-sized fragments, which are distributed to a pool of slaves for further MapReduce processing. But some cases the distribution depends on hardware configuration [28].

3.4 ComMapReduce Overview

In contrary to the original MapReduce framework, *'ComMapReduce'* [23] adopts efficient communication mechanisms to improve the performance of query processing of massive data in the cloud. 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. In order to optimize the intermediate data objects of the original MapReduce framework, it is an efficient lightweight communication framework extending MapReduce by pruning unpromising data objects of query processing dramatically. Actually, the ComMapReduce inherits the basics of MapReduce and takes HDFS to store the original data objects and the final results of each application.

In ComMapReduce framework, the Master node and a Slave node play 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 values* of the query processing applications. Each Mapper and Reducer 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.

4. Problem Statement

This section first formalizes some related problem of mining-correlated, associated-correlated, and independent patterns and then presents some preliminary knowledge that will be used in our algorithm and these have been adopted from literature [5,15,29].

Suppose, we have a transactional database DB in Table 1, and user given some threshold values, now the problem is to mine the complete set of correlated, associated, correlated-associated, and independent pattern efficiently and synchronously by using the ComMapReduce framework on Hadoop. In statistical theory, $A_1, A_2 \ldots A_n$ are independent if $\forall\ k$ and $\forall\ 1 \le i_1 \le i_2 \le \ldots \le i_k \le n$,

$$P(A_{i1}\ A_{i2} \ldots A_{ik}) = P(A_{i1})\ P(A_{i2}) \ldots P(A_{ik}) \quad (1)$$

1. If a pattern has two items, such as pattern AB, then

$$\rho(AB) = \frac{P(AB) - P(A)P(B)}{P(AB) + P(A)P(B)} \quad (2)$$

2. If a pattern has more than two items, such as pattern $X = \{i_1, i_2 \ldots i_n\}$, then

Table 1: A transactional database

TID	Transactions
10	A, B, C
20	C, D, E
30	D
40	A, C, D, E
50	D, E
60	B, D
70	G, H
80	A, C, E
90	A
100	A, C, D
110	B, D, E
120	C

$$\rho(X) = \frac{P(i1i2\ldots in) - P(i1)P(i2)\ldots P(in)}{P(i1i2\ldots in) + P(i1)P(i2)\ldots P(in)}, \text{where } n \geq 1 \quad (3)$$

From (2) and (3), we can see that ρ has two bounds, i.e., $-1 \leq \rho \leq 1$. Let ∂ be a given minimum corr-confidence, if pattern X has two items A, B and if $|p(AB)| > \partial|$, then X is called a correlated pattern or A and B are called correlated, else A and B are called independent. If pattern X has more than two items, we define a correlated pattern and an independent pattern as follows:

Definition 1: Correlated pattern - Pattern X is called a correlated pattern, if and only if there exists a pattern Y which satisfies $Y \subseteq X$ and $|\rho(AB)| > \partial$.

Definition 2: Independent pattern - If pattern X is not a correlated pattern, then it is called an independent pattern. Now, the associated patterns can be defined as follows: Let $T = \{i_1, i_2, i_3 \ldots i_m\}$ be a set of m distinct literals called items and D is the set of variable length transaction over T. Each transaction contains a set of items, $\{i_{j1}, i_{j2}, i_{j3} \ldots i_{jk}\} \subset T$, where pattern X is a subset of T. Let P (X) be a power set of pattern X. The interestingness measure all-confidence denoted by α of pattern X is defined as follows [15]:

$$\alpha(X) = \frac{Sup(X)}{Max\_item\_sup(X)}$$

Definition 3: Associated pattern - A pattern is called an associated pattern if its all-confidence is greater than or equal to the given minimum all-confidence.

Definition 4: Associated-correlated pattern - A pattern is called an associated-correlated pattern if it is not only an associated pattern but also a correlated pattern. Let pattern X be an associated-correlated pattern, then it must have two subsets A and B which satisfy the condition that the sale of A can increase the likelihood of the sale of B.

Example: For the transaction database DB in Table 1, we have $\alpha(AC) = 4/5$ and $\alpha(CE) = 3/5$. We also have

$\rho(AC) = P(AC) - P(A)\,P(C)/P(AC) + P(A)\,P(C) = 3/13$ and

$\rho(CE) = P(CE) - P(C)\,P(E)/P(CE) + P(C)\,P(E) = -1/49$

Let the given minimum all-confidence is 0.35 and the given minimum corr-confidence is 0.10, then both AC and CE are associated patterns. However, pattern AC is a correlated pattern and pattern CE is an independent pattern. Therefore, pattern AC is an associated-correlated pattern and pattern CE is an associated but not correlated pattern.

5. Proposed Approach

We mine the complete set of purchase rules and frequently correlated, associated, associated-correlated, and independent patterns in two steps. First, we discover all the frequent patterns, and then test whether they are correlated, associated, associated-correlated, and independent patterns or not. For the maximal frequent itemsets mining, we later on combined the set of frequent itemsets.

5.1 Proposed Programming Model

The transactional database is split into smaller segments automatically after being stored on the HDFS. The Hadoop components perform job execution, file staging, and workflow information storage and use the files replace the database to store datasets automatically. We used Hadoop-based parallel FP-Growth proposed at literature [16] and we follow the data placement strategy indicated at [28]. Figure 1 shows the input/output schemes of our proposed framework. Figure 2 indicates the workflow of the proposed MapReduce-framework; on the other hand, the pseudo code of the proposed framework has shown in Figure 3.

Operations go as follows: After splitting the transactional database into smaller segments, the master node assigns task to each idle worker. Each worker scans the transactions in the smaller data segments as < *ID*, *itemset* > and the balanced FP-growth [16] was applied to produce output as <CPB, CT> or <conditional patterns base, conditional transaction > pairs.

These values are to be stored in the local disk as intermediate results and the HDFS will perform the sorting or merging operations to produce results as <CPB, CDB> or <conditional pattern base, conditional data-

I/O Schemes for our Proposed Approach with ComMapReduce Framework

I/O	Map	Merge	Reduce-1	Reduce-2	Reduce-3
Input: key/value pairs	Key: *TID* Value: *Itemset*	Key: *CPB* Value: *CDB*	Key: *CPB* Value: *CDB*	Key: *CPB* Value: *CFPT*	Key: *FI* Value: *Support*
Output: key/value pairs	Key: *CPB* Value: *CT*		Key: CPB, Value: CFPT	Key: *FI* Value: *Support*	Key: *MFI* Value: *Support*

CPB - Conditional pattern base; CT - Conditional transaction; CDB - Conditional database; FI - Frequent itemset; MFI - Maximal frequent itemset and CFPT - Conditional FP-Trees

Figure 1: Input/output schemes for the proposed approach with ComMapReduce framework.

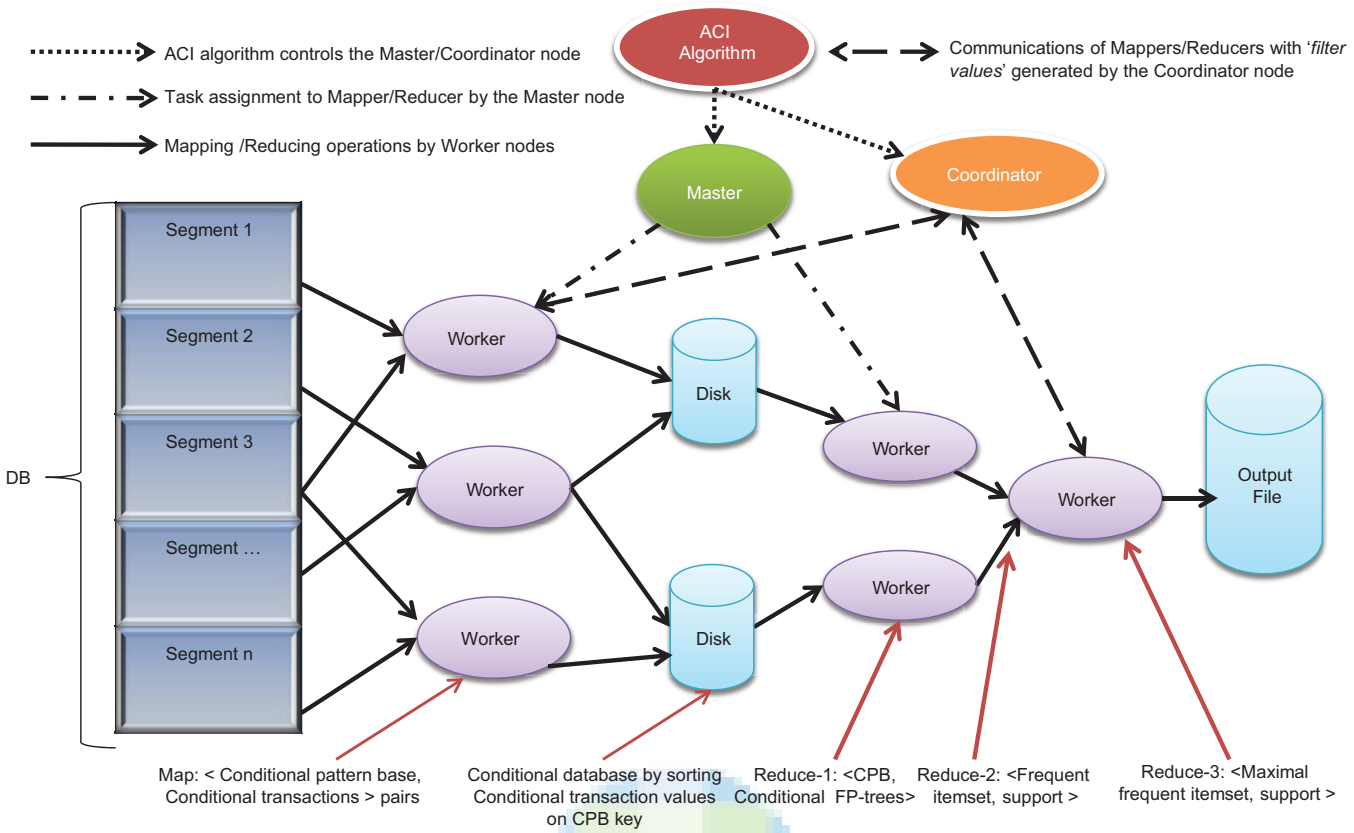

Figure 2: Proposed framework for mining-correlated, associated-correlated and independent patterns along with customer's purchase rules.

base> pairs. The Coordinator node receives and stores some temporary variables, and then generates *filter values* of the query processing applications [23]. For this, we used two levels of pruning techniques, local pruning, and the global pruning, using two minimum support thresholds; *local_min_sup* and *global_min_sup*. Each Mapper obtains two *filter values by means of global_min_sup and local_min_sup* from the Coordinator node to prune its unwanted data objects or infrequent items or candidate itemsets. The Local pruning is applied on map phase in each segment and the global pruning will be applied in the reduce phase. While Reducer obtains two *filter values by means of min_all_conf and min_corr_conf* from the Coordinator node to prune its unwanted data objects inferior to the other ones in map and reduce phases.

In reduce phase 1, idle Reducers take input as <CPB, CDB> pairs and reduce the data objects as <CPB, CFPT> or <conditional pattern base, conditional FP-Trees> pairs. In reduce phase 2, input is taken as the <CPB, CFPT> pairs and reduce the data objects as <FI, Support> or <Frequent Itemset, Support> pairs. In reduce phase 3, idle Reducers take input as <FI, Support> pairs and reduce the data objects as <MFI, Support> or <Maximal Frequent Itemset, Support >pairs. Another extra task is also done during this reducing phase that is applying constrains mentioned by the definition 1, 2, 3, 4 to generate ACI that is the complete set of correlated, associated, associated-correlated and independent patterns.

Finally, the output is stored in the local disks as indicated in Table 2. Actually, in this phase, incorporation of three phases is possible, since the amount of produced dataset is less compared to the map phase.

5.2 Complexity Analysis of the ACI Algorithm

To read each transaction in assigned segment(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 occur 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 $O(n + mn)$. It has to scan less items/transaction due to the removal of null transactions and infrequent items in each segment(s).

5.3 An Example

Suppose the min_global_sup is 2, minimum correlated

The ACI algorithm with ComMapReduce framework on Hadoop for mining Big Datasets

Input: i) A transactional database on HDFS, ii) '*min_global_sup'*, iii) '*min_local_sup',* iv) *'min_corr_conf',* and v) *'min_all_conf'*.

Output: i) The complete set of purchase rules as the maximal frequent itemsets that satisfy the *min_global_sup* and ii) the complete set of frequently correlated, associated, associated-correlated, and independent patterns which satisfies the *min_corr_conf* and *min_all_conf*.

Preprocessing: Master node does three preprocessing task before Map/Reduce operations: i) Removes '*infrequent items*' from the dataset, ii) Removes *'null transactions',* iii) Splits the database into segment $D_{\alpha i}$ when stored on HDFS. The Coordinator node sends the lightweight values using the *min_corr_conf* and *min_all_conf* during the Map/Reduce operations.

Mapper ($D_{\alpha i}$) {
//Map: Slave nodes scan the assigned segment(s) and map the output as $<CPB_i, CT_i>$ pairs.

1. ***for*** *each Transaction* T_i , *i=1 to N*
2. ***Call*** *Balanced_Parallel_FP (TID, I)* and generate conditional transactions based on conditional pattern base.
3. ***Write*** the output on the local disks as $<CPB_i, CT_i>$ pairs.
4. }

Sort $<CPB_i, CT_i>$, key) // Sorts the intermediate output and write results on the local disks.
{
5. ***Combine*** the conditional transactions as list according to the key CPB_i.
6. ***Sort*** the list using merge sort and store the result on the local disk as $<CPB_i, <CT_1, CT_2 \ldots CT_i> >$ pairs.
7. ***Return*** sorted list of $<CPB_i, <CT_1, CT_2 \ldots CT_i> >$ pairs.
}

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

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

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

//Step-1: Finding the complete set of purchase rules as the maximal frequent itemset

14. ***If*** CPB_i.Y is a frequent itemset:
15. ***If*** *no superset of* CPB_i*.Y in the frequent itemset list is frequent*
16. CPB_i.Y is a maximal frequent itemset
17. ***Write*** these rules on the local disks

//Step-2: Finding the complete set of frequently correlated, associated, associated-correlated and independent patterns.
Suppose CPB_i.X is a frequent itemset but not a maximal frequent itemset:

18. ***for*** each CPB_i.X i=1 to N
19. ***If*** (∂(CPB_i.X> min_corr_conf) // ∂ is the calculated threshold value for min_corr_conf
20. CPB_i.X is a correlated pattern.
21. ***If*** (|(α(CPB_i.X)) | > min_all_conf) // α is the calculated threshold value for min_all_conf
22. CPB_i.X is an associated pattern
23. ***If*** (∂(CPB_i.X) > min_corr_conf && | (α(CPB_i.X)) | > min_all_conf)
24. CPB_i.X is an associated-correlated pattern.
25. ***Else***
26. CPB_i.X is an independent pattern
27. }

Figure 3: The ACI algorithm by using ComMapReduce framework on Hadoop.

Table 2: Final output from the worker 03

Frequent pattern	$\|\rho(X)\|$	$\alpha(X)$	Correlated?	Associated?	Associated-correlated?	Independent?	Purchase rules?
A, C	0.23	0.8	√	√	√		
A, D	0.20	0.33	√				
A, E	0.11	0.40	√	√	√		
B, D	0.059	0.33				√	
C, D	0.11	0.50	√	√	√		
C, E	0.021	0.60		√		√	
D, E	0.033	0.67		√		√	
A, C, D	0.20	0.33	√				√
A, C, E	0.11	0.40		√	√		√
C, D, E	0.033	0.40		√		√	√

confidence is 0.10 and minimum all-confidence is set to be 0.35. Now, according to our proposed framework, the transactional database in Table 1 has been split into two segments, each having four transactions; TID 10 to 60 in first segment and transaction TID 70 to 120 in second segment. In segment 01, transactions 30; where transactions 90 and 120 at segment 02 are null transactions and are not considered for mining (more details can be found at literature [12]). Since a null transaction with just 1 item has no contribution for correlation/association rule mining in market basket analysis, it was removed prior mining. On the other hand, items G and H in segment 02 are infrequent, consequently are pruned. Let the master node has assigned segment 01 to worker node 01 and segment 02 to worker node 02. The Figure 4 shows the Map/Reduce phases and the result after the necessary calculations. Table 2 shows the necessary patterns and purchase rules based on the calculated values defined by definition 1, 2, 3, and 4.

6. Experimental Results

6.1 Programming Environment

We used Hadoop version 0.23.0, running on a cluster with nine machines (1 master, 1 coordinator, 7 worker). Master node has 3.7 GHz Intel core 2 duo processor with 4 GB of RAM and each worker machine has Pentium D 2.60 GHz processor with 2 GB RAM. The balanced FP-growth [16] was modified using Java on MapReduce library functions and we configured HDFS on Ubuntu-11.04.

6.2 Description of the Datasets

We follow the load distribution indicated at literature [28]. We apply the ACI algorithm on practical and randomly generated datasets. The practical "Connect-4", Mushroom, and Agaricas datasets have been downloaded from http://rchive.ics.uci.edu/. The dataset Connect-4 contains 135,115 transactions, lengths 8~35, unique items are 76, and size of 200 MB. On the other hand, Mushroom and Agaricas has 78,265 and 92,653 transactions with size of 120 MB and 170 MB, respectively.

For the first experiment, we used three different transaction files; 800 MB (13 M transactions), 1200 MB (25 M transactions), and 1500 MB (30 M transactions) with 42 unique items in each and executed on 3, 5, and 8 computing nodes, respectively. For the second experiment, datasets were split into segments after stored on the HDFS across 2, 3, and 4 nodes. Since there is no significant difference of MBA algorithm in execution time when the data size is smaller than 800 MB in some works [21,22], we used transaction files larger than or equal to 800 MB and for this purpose, we directly copied these data files on the HDFS without segmentation.

6.3 Performance and Result Analysis

We did not compare our result with any existing algorithm since all of the previous works were implemented on main memory-based single processor machine. Another issue is that we mine not only the frequent patterns or itemsets, but also customer's complete purchase rules by means of maximal frequent patterns, correlated, associated-correlated, and independent patterns why our framework takes more time to finish the Map and Reduce phases.

In the first experiment, we measured the execution time of map and reduce phase as the performance metric with three filter values as *min_global_sup* = 12, *min_corr_conf* = 10~90%, and *min_all_conf* = 40~95% *on the randomly generated datasets*. It can be observed that when the data size is larger like 1200 or 1500 MB, the execution became much faster to compute the frequent itemsets, maximal frequent itemset, or other related patterns for 5 and 8 computing nodes as shown by Table 3.

In addition, it also has the linear speedup of the ACI algorithm, since we avoided the sorting and grouping on the original dataset for the <key, value> pairs explicitly; for this consequence, less amount of candidates are generated which also saves the huge time to copy the dataset to the original locations again, so the execution time is satisfactory. The execution time linearly increased, but it reached the maximum parallelization at the instances

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

Segment 01		**Map Input**	**Map Output**
ID	Itemset	**Key:** TID, **Value:** Itemset	**Key:** CPB, **Value:** CT_i
10	A B C	< 10, ABC >	C: AB, B:A, A: Φ
20	C D E	< 20, CDE >	E: CD, D:C, C: Φ
40	A C D E	< 40, ACDE >	E: ACD, D:AC, C: A, A: Φ
50	D E	<50, DE>	E:D, D: Φ
60	B D	< 60, BD >	D: B, B: Φ
Segment 02		**Map Input**	**Map Output**
ID	Itemset	**Key:** TID, **Value:** Itemset	**Key:** CPB, **Value:** CT_i
80	A C E	< 80, ACE >	E:AC, C:A, A: Φ
100	A C D	< 100, ACD >	D:AC, C:A, A: Φ
110	B D E	< 110, BDE >	E: BD, D: B, B: Φ

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

Key (Conditional Pattern Base)	Value (Conditional Databases)
A	{ Φ }
B	{ A }
C	{ AB, A, A, A }
D	{ C, AC, B, AC, B }
E	{ CD, ACD, D, AC, BD }

I/O to the Reduce Phase 1 & 2

Reduce-1: Input	**Reduce-1:** Output /Reduce -2 Input	**Reduce-2:** Output of Step-1	**Reduce-2:** Output of Step-2
Key: CPB, **Value:** CDB_i	**Key:** CPB, **Value:** CFPT	**Key:** Frequent Itemset **Value:** Support	**Key:** Maximal Frequent Itemset **Value:** Support
< B, (A) >	B: { (A:1) }		ACD:2 CDE:2 CDE:2
< C, (AB, A, A, A) >	C: { (AB:1,B:1,A:4) }	< AC, 4>	
<D, (C, AC, B, AC, B) >	D: {(A:2, B: 2, AC:2, C:3)}	< AD, 2>, <BD, 2>, <ACD, 2>, <CD, 3>	
< E, (CD, ACD, D, AC, BD)>	E: { (D:3, A: 2, B: 1, C: 3, CD: 2, AC:2, BD:1, AC:1 }	< DE, 3>, <AE, 2>, <CE, 3>, <CDE, 2>, <ACE, 2>	

Figure 4: The Map/Reduce phase: Result calculations.

around 10 and 15 nodes in map and reduce phases in some works [18,20-22], but for our framework since we are applying the load distribution as described by the literature [28], we can scale our framework for more nodes for the parallelization.

In the second experiment, we measured the execution time of our ACI algorithm for the map/reduce operations on practical Connect-4, Mushroom, and Agaricas datasets as shown by Table 4 with *min_global_sup* = 10~90%, *min_corr_conf* = 10~60%, and *min_all_conf* = 5~50%.

In the third experiment, speed-up process has been shown by our proposed framework. Table 4 and Figure 5

Table 3: Execution time (Map/Reduce) on random datasets

Data type #Nodes	Time (ms) 800 MB	Time (ms) 1200 MB	Time (ms) 1500 MB
3	298,218	418,236	520,918
5	187,657	246,897	350,231
8	155,123	184,963	245,847

Table 4: Execution time (Map/Reduce) on practical datasets

Data 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

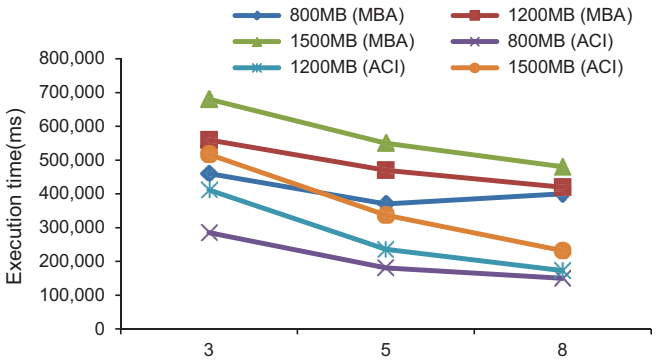

Figure 5: Execution time with change of three filter values on Randomly generated dataset (X-axis #nodes).

show the running time on Connect-4 datasets across 3, 5, and 8 data nodes. We can observe almost 50% improvement on running time on synthetic and real dataset by increasing number of nodes from 3 to 8. This also speeds up the overall proposed MapReduce framework.

7. Conclusion

In this paper, we proposed a Hadoop and improved MapReduce framework and an efficient "ACI algorithm" that effectively mines the complete set of customer's purchase rules along with the correlated, associated, associated-correlated, and independent patterns synchronously. Experimental results show correctness and scalability in terms of increasing load. We also showed how correlated pattern mining can be performed on the top of an implementation of the traditional Apriori and FP-growth frequent itemset mining algorithm. Besides, our proposed approach by using the ComMapReduce framework also extends and improves the limitations of previous approaches for massive market basket data.

8. Acknowledgment

This work was supported by the National Research Foundation (NRF) grant (No. 2012-0001001) of Ministry of Education, Science and Technology (MEST) of Korea. We are also grateful and would like to thanks the anonymous reviewers for their suggestions and comments which really helped to improve the readability and quality of the manuscript.

References

1. S. Berchtold, D.A. Keim, and H.P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," Proceedings of 22th International Conference on Very Large Data Bases(VLDB'96), September 3-6, 1996, Mumbai (Bombay), India, pp. 28-39, 1996.
2. S.K. Tanbeer, C.F. Ahmed, and B.S. Jeong, "An Efficient Single-Pass Algorithm for Mining Association Rules from Wireless Sensor Networks," IETE Technical Review, IETE Technical Review, Vol. 26, no. 4, pp. 280-9, Jul-Aug. 2009.
3. R. Agrawal, and R. Srikant, "Fast algorithms for Mining Association Rules," Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94), San Francisco, USA, pp. 487-99, 1994.
4. J. Han, and M. Kamber, Data Mining: Concepts and Techniques, 3rd Ed, Burlington, Massachusetts: Morgan Kaufmann; 2011.
5. Z. Zhou "Mining Frequent Independent Patterns and Frequent Correlated Patterns Synchronously", Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'2008), Vol. 5, pp.552-6, Oct. 2008.
6. Z. Zhou, Z. Wu, and W. Fengyi, "Mining both associated and correlated patterns," Lecture Notes in Computer Science, Vol. 3994, pp. 468-75, 2006.
7. M. Khalil, and Y. Taha, "Beyond Market Baskets: Generalizing Association Rules to Correlations", Proceeding of the ACM SIGMOD international conference on Management of data, Vol. 26, no. 2, pp. 265-76, Jun. 1997.
8. H.T. Reynolds, "The Analysis of Cross-Classifications", University of Minnesota Free Press, 1977.
9. G. Piatetsky-Shapiro, "Discovery, Analysis and Presentation of Strong Rules", Knowledge Discovery in Databases, Cambridge, MA: AAAI/MIT Press; pp. 229-48, 1991.
10. C.F. Ahmed, S.K. Tanbeer, and B.S. Jeong, "A Framework for Mining High Utility Web Access Sequences," Vol. 28, no.1, pp. 3-78, Jan-Feb. 2012.
11. L. Jian, and W. Chong, "Prediction of E-shopper's Behavior Changes based on Purchase Sequences," International Conference on Artificial Intelligence and Computational Intelligence (AICI), 2010.
12. 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), Vol. 1, pp. 1-6, May. 2012.
13. T. Elsayed, J. Lin, and W. Oard, "Pairwise Document Similarity in Large Collections with MapReduce," the 32nd Intl. ACM Conf. on Research & Development, 2009.
14. J. Ekane, and G. Fox, "MapReduce for data intensive scientific analyses", IEEE Fourth International Conference on Science, E-Science '08, pp. 277-84, 2008.
15. Z. Zhou, Z. Wu, C. Wang, and Y. Feng, "Mining Both Associated and Correlated Patterns", Lecture Notes in Computer Science, Vol. 3994, pp. 468-75, 2006.
16. L. Zhou, Z. Jong, and S. Feng, "Balanced Parallel FP-Growth with MapReduce," IEEE Youth Conf. on Information Computing and Telecom (YC-ICT), 2010.
17. J. Dean, and S. Ghemawa, "MapReduce: Simplified Data Processing on Large Clusters," OSDI, 2004.
18. H. Dutta, and J. Demme, "Distributed Storage of Large Scale Multidimensional EEG Data using Hadoop/HBase," Grid and Cloud Database Management, New York City: Springer; 2011.
19. M.R. Karim, 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, pp. 162-8, Mar-Apr, 2012.
20. M.R. Karim, and B.S. Jeong, "A Parallel and Distributed Programming Model for Mining Correlated, Associated-Correlated & Independent Patterns in Transactional Databases Using MapReduce on Hadoop," Proc. of the 6th Intl. Conf. on CUTE, pp. 271-6, 2011.
21. J. Woo, S. Basopia, and S.H. Kim, "Market Basket Analysis Algorithm with NoSQL DB HBase and Hadoop," In proc. of the 3rd International Conference on Emerging Databases (EDB2011), Korea, pp. 56-62, Aug. 2011.
22. J. Woo, 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, Las Vegas, USA, Jul. 2011.
23. L. Ding, and S. Huang, "ComMapReduce: An Improvement of the MapReduce with Lightweight Communication Mechanisms," LNCS, Vol. 7238, pp. 303-19, 2012.
24. E. Omiecinski, "Alternative interesting measures for mining associations," The IEEE Transaction on Knowledge and Data Engineering, Vol. 15, no. 1, pp. 57-69, 2003.

25. B. Liu, W. Hsu, and Y. Ma, "Pruning and Summarizing the Discovered Association". Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, USA, pp. 125-34, 1999.
26. S.F. Zerin, and B.S. Jeong, "A Fast Contiguous Sequential Pattern Mining Technique in DNA Data, Sequences Using Position Information," IETE Technical Review, Vol. 28, no. 6, pp. 451-520, Nov-Dec. 2011.
27. S.K. Tanbeer, C.F. Ahmed, and B.S. Jeong, "Parallel and Distributed Algorithms for Frequent Pattern Mining in Large Databases," IETE Technical Review, Vol. 26, no.1, pp. 55-65, Jan-Feb. 2009.
28. J. Xie, J. Majors, and X. Qin, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters", IEEE Intl. Symposium IPDPSW, 2010.
29. Y.K. Lee, W.Y. Kim, and J. Han, "CoMine: Efficient Mining of Correlated Patterns," in proceedings of the IEEE intl. conf. of data mining (ICDM'03), Vol. 1, pp. 581-4.

AUTHORS

Md. Rezaul Karim received the BS degree from the Dept. of Computer Science & Engineering, University of Dhaka, Bangladesh, in 2009 and the MS degree from the Dept. of Computer Engineering, Kyung Hee University, Korea, in 2012. Currently, he is working as a Senior Software Engineer at Samsung Bangladesh R&D Center (SBRC). His research interest includes data mining, ubiquitous data management, mobile computing, and bioinformatics.

E-mail: asif_karim@khu.ac.kr

Chowdhury Farhan Ahmed is an Associate Professor of Computer Science & Engineering department at the University of Dhaka. He received the BS and MS degrees in Computer Science from the University of Dhaka, Bangladesh, in 2000 and 2002, respectively, and the Ph.D. degree from the Kyung Hee University, Korea, in 2010. From 2003-2004, he worked as the faculty member in the Institute of Information Technology, University of Dhaka. Since 2004, he has been working as the faculty member of Computer Science & Engineering department at the University of Dhaka. He has published more than 50 scientific research papers in top ranked international journals and conferences. His research interests include database systems, data mining, and knowledge discovery.

E-mail: farhan@cse.univdhaka.edu

Byeong-Soo Jeong received the BS degree in Computer Engineering from Seoul National University, Korea, in 1983, MS in Computer Science from the Korea Advanced Institute of Science and Technology, Korea, in 1985, and the PhD in Computer Science from the Georgia Institute of Technology, Atlanta, in 1995. From 1996, he is a professor at the Dept. of Computer Engineering, Kyung Hee University, Korea. From 2003 to 2004, he was a visiting scholar at the Georgia Institute of Technology. His research interests include database systems, data mining, and mobile computing.

E-mail: jeong@khu.ac.kr

Ho-Jin Choi received the BS degree in Computer Engineering from Seoul National University, Korea, in 1985, MS in Computing Software and Systems Design from Newcastle University, UK, and the PhD in Artificial Intelligence from Imperial College, London, in 1995. Between 1995 and 1996, he was a post-doctoral researcher at IC-PARC, Imperial College, London. From 1997 to 2002, he worked as an assistant professor of Computer Engineering at Korea Aerospace University. Currently, he is an associate professor at the Dept. of Computer Science at KAIST, Korea. His research interests include artificial intelligence, data mining, software engineering, and biomedical informatics.

E-mail: hojinc@kaist.ac.kr

DOI: 10.4103/0256-4602.107340; Paper No. TR 377_11; Copyright © 2013 by the IETE