Implementation of Apriori Algorithm Based on Hadoop Clusters

Kim TongGuk*, Pak CholRyong, Ryang KwangJin
Faculty of Information Science, **Kim Il Sung** University, Pyongyang,
Democratic People's Republic of Korea
Corresponding Author Email Address: kdg198931@star-co.net.kp

Abstract

With manufacturing technology developing persistently, hardware manufacturing cost becomes lower and lower. More and more computers equipped with multiple CPUs and enormous data disk emerge. Existing programming modesmake people unable to make effective use of growing computational resources. Hence cloud computing appears. With theutilizationofMapReduceparallelizedmodel,existingcomputingandstorage capabilitiesareeffectivelyintegratedandpowerfuldistributedcomputingabilityis provided.

Associationrulescanforcefullygetahorizontal relationinthebigdata,theApriorialgorithmisoneofthe most significant association rules. Traditional mining based on parallel Apriori algorithms needs much more time in data IO withtheincreasingsizeoflargetransactiondatabase.This paperimprovestheApriorialgorithmfromcompressing transactions,reducingthenumberofscansandsimplifying candidate set generation. And then the improved algorithm is parallelized on the Hadoop framework. The experiments show thatthisimprovedalgorithmissuitableforlarge-scaledata mining and has good scalability and effectiveness.

Keywords*:Apriori algorithm, Association rules, Data Mining, MapReduce,Hadoop*

1. Introduction

In the context of the development of big data "spraying wells", there isfrequently a close relationship between vast amountsofdata[1].Analysisanddecisionmakingthrough datamininghavebecomethemainstreamofsocial development.Inordertobetterfindtherelevanceof transactiondatasets,someresearchershavediscoveredthe conceptofassociationruleminingtechnology[2].Withthe attention of many researchers at home and abroad caused by theconceptionoftheassociation rule mining,theyhavedonealotof analysisinthisfieldandputforwarda lot ofdatamining algorithms.

One of the most famous association rule algorithms is the Apriorialgorithm,whichisaclassicassociationrule algorithmdesignedbyAgrawal[3-4]in1994.Itisa level-by-levelsearchiterationmethodthatconstructsa k-item set to constitute a k+1-item set. The main ideas of this algorithm are: Firstly, all frequency sets are counted from the transactiondatabase,andthesupportofthisfrequentset must not be less than the minimum support degree; Secondly it enters into the process of strong association rule generation, andtherulesneedtosatisfythesupportandconfidence thresholdsatthesametime;Thirdly,onlyallrulesthat containcollectionitemsareretained.Oncetheserulesare retained and generated, that are greater than or equal to the MinConfidence.

Due to emergence of cloud computing, it's possible to get big cheapcomputingandstorageability quickly and dynamically,solvingthemostfundamentalproblemfordata miningabouthowtoacquireinexpensivelypowerfuldatacomputingability[5-8]. Relatedresearchersdirectedattentions to cloudcomputingplatform,inthehopeof implementingdatamining algorithm withhighscalability,applyingcheapcomputingof cloudcomputingtodataminingbasedonstorageability, thus overcoming the shortcomingsin traditionaldatamining,reducingcalculationcostandenhancingdata mining efficiency [9-14].

With a view on the broad and promising future of cloud computing, the integration of studying and applying cloud computing and existing data mining algorithm has become hotconcerninvariousindustries[15-18].

The design of the Hadoop[19] framework originatedwas fromanopensourceprojectdevelopedbytheApache organizationFoundation.Becauseofitsinter-temporal significance, the Hadoop framework has been widely used in theinformationfieldathomeandabroad.Therearetwo importantmodulesintheHadoopframe-DistributedFile SystemHDFSandDistributedComputingFrame MapReduce[20].Asadistributedfilesystem,HDFS functionsaimstoimplementdatastorage.Itwillworkin conjunction with the computational framework. MapReduce works to provide the underlying support for data calculations; AndtheideaofMapReduce[21]isbasedonapaperby Google.Inshort,itscoremethodis"thedecompositionof tasks and the statute of results."

Herewetransformclassicaldatamining correlativealgorithmAprioriintoimplementingincloudcomputingenvironmentbased onMapReducemodel;meanwhile,accordingtocharacteristicsof MapReducemodel, weimproveAprioriálgorithmtomakeMapReduce-Apriorialgorithmwithstrong scalability, fit for tremendous data analysis and processing. Finally, utilize Map Reduce-Aprioriparallelalgorithmtotesttheproposedmethodwithrunningtimefromthe perspectiveofdatavolumeandcomputingnodequantitytogetpracticallymeaningful data mining results [18].

2. Brief and Research Status of Apriori Alogorithm

2.1 Overview of Apriori algorithm

The Apriori algorithm is a iterative level-by-level search method that consists of a k-item set to construct a (k+1)-item set.First,obtainafrequent1-itemset.

L1cangeneratea frequent2-itemsetL2,andL2cangenerateafrequent 3-item set L3. According to this rule, when a frequent k-itemsetcannotbefound,thealgorithmends[22].Thespecific operation is as follows:

1) Iteratethroughtheinitialtransactiondatabaseand count the frequency of occurrence of the candidate set. The resultisthesupportoftheproject.Allprojectswhoseall supports level no lower than the preset threshold generate a frequent 1-item set L1.

2) ThealgorithmusesL1JOINL1toformacandidate C2-item set C2.

3) Using the items in C2, traverse the database again to obtain the support degree of each candidate set. All projects with support levels not lower than the support level generate frequent 2-item set L2.

4) ThealgorithmusesL2JOINL2toformasetC3of candidate 3-item sets.

5) Using the items in C3 to traverse the database again, thesupportdegreeofeachcandidatesetcanbeobtained.

All items with support levels not lower than the support level generate frequent 3-item set L3.

Theaboveprocessisperformediterativelyuntilthe candidatesetCkisempty.TheApriorialgorithmdoes multiple IO operations on the database. Each stage consists of two parts, namely connection and pruning.

2.2The shortcomings of Apriori algorithm

1) WhentheAprioríalgorithmgeneratesthecandidate item set, it needs to perform the self-connection operation on thefrequentitemsetsobtainedinthepreviousstep.Then scan the transaction data set again and compare the candidate set formed by the self-connection with min_sup. During the self-connectionoperation,alargeamountofcomparison work will be performed.

2) Apriorialgorithmneedtorescantransactiondatasets before pruning, and then compare with min_sup. Therefore, when the size of the transaction dataset is getting larger and larger,eachscanwillconsumealotoftime,resultingin inefficient mining.

3) In the current situation where the data information has a high dimension and the type is complex, the classical Apriori algorithm can't satisfy users.

4) Because the classic Apriori algorithm is only applicable toasinglemachine,whenthesizeoftransactiondatasets gradually becomes larger and larger, it will lead to inefficient mining, insufficient storage space, and even system crashes.

3.Implimentation of Apriori Algorithm Based on Hadoop Clusters

3.1 Reduce frequent item sets self-connection comparison times and pruning steps

In this paper, a method to reduce data when scan for candidate set has been introduced.

If n-dimensional data set is not considered as frequent set, then its n-1dimensional data set is also not the frequent data set so that by comparing and deleting not-frequent data set is finally resulted in smaller size of data to scan leading to high efficiency of scanning algorithm.

3.2 Reduce the Number of Scanned Databases

It is kind of problematic to scan database for scanning frequent item sets because of frequent I/O but making the database as vertical data table effectively reduce the number of scanned databases because finally it is resulted in scanning one transaction database.

3.3Combining Apriori Algorithm and Hadoop Platform

3.3.1 Data Initialization

The symbols used in this paper are described in table 1.

Table 1. List of Symbols

Symbol	Meaning
L_k	Frequent sets of the kth layer
GMap	First layer frequently set the item set to id mapping table
LM_k	Ordered and mapped to the kth layer of id in GMap frequent sets
C_k	Candidate sets for the kth layer
S_k	A superset of the kth layer
MP	Master mode
MPS_k	Master mode (k-1) sub mode
GP	Generation model
GPS	Generation mode base

T	Original transaction set
T_k	kth layer transaction set after compression

After producing first frequent itemset, then arrange it and combine it in frequent set so that simpler calculation is implemented without complex comparisons , still, it is needed to do huge amount of calculation so that it can be considered as independent stage called data initializing stage.

Three steps have been done through data initializing stage.

1) Producing frequent itemsets

2) Ordering frequent itemsets

3) Generating second candidate itemsets.

The bellow pseudo code is parallelized process which is for data initializing stage to generate second frequent itemsets C_2 basing on Map Reduce model. Here C_2 is odered.

```
//MapReduce Stage1-1
    Mapper{
    Map() {
    For each itemsets  in value
    Key=itemsets
    Value=1
    Emit(key,value)
    }}
//Stage1-2
    LM1=SortOutputAndMap(L1)
//Stage1-3
    Mapper{
    Map() {
    For i=0; i< LM1.size()-1; i++
    For j=i+1; j< LM1.size()-1; j++
    Emit(LM1[i], LM1[j])
    }}
```

Through the second stage, results from the first stages are sorted by support degree and then ranked into digital id which is called GMap. GMap makes the data easier to treat because it transforms the character expressed data into numerical group attributing to high efficiency of data calculation such as comparison.

The results of second stage is second candidate itemsets which is grouped by host nodes. Table 2 shows the dataflow thru the entire data initializing stage.

Table 2. Initialization Phase Data Stream

Original input	After sorting the frequent sets (minimum support degree is 50%)	Calculate the 1st itemset

	sort	GMap	MapOutPut	ReduceOutPut(C_2)
acfg	a, 4 c ,4 d ,3 g, 3	1, a, 4 2, c ,4 3 ,d ,3 4 ,g, 3	(1,12) (1,13) (1,14) (2,23) (2,24) (3,34)	(1,[12,12,14] (2,[23,24]) (3,[34])
abcde				
bdgac				
adc				
fhg				

3.3.2 Iterative implementation

Finding frequent itemsets could be carried out by iterating bellow two oerations. These operations are basing on dataset from data initializing stage.

1) Compute the kth frequent itemset

In this stage all data is loaded to internal memory so that increase the efficiency of computation. Input data to Map is files' row data and the output key is value of each column. The pseudo-codes are implemented as follows:

```
//MapReduce Stage2-1
Mapper{
Setup()
Map() {
Value=MapToid(value, GMap)
value.sort()
For each itemsets in
If(value.contains(itemsets))
Emit(itemsets, 1)
}}
Reducer {
Reduce() {
sum=0
For each value in values
sum=sum + 1
If sum > minsup
Emit(key, sum)
}};
```

Table 3. Calculating the Data of 2nd Layer Frequent Sets

Original input	Map stages		Reduce stages	
	Mapping and scheduling	output	input data	output
acfg	1,2	[1,2],1	[1,2],1 [1,2],1 [1,2],1 [1,2],1	[1,2],4 [1,3],3 [2,3],3
Abcde	1,2,3	[1,2],1 [1,3],1 [2,3],1	[1,3],1 [1,3],1 [1,3],1	

bdgac	1,2,3,4	[1,2],1 [1,3],1 [2,3],1 [2,4],1 [3,4],1	[2,3],1 [2,3],1 [2,3],1	
adc	1,2,3	[1,2],1 [1,3],1 [2,3],1	[2,4],1 [3,4],1	
fhg	4			

2) Compute the (k+1)th candidate itemset

This stage includes two steps of implementation:

(1) produce superset at (k+1)th layer from the kth frequent sets;

(2)trim superset at (k+1)th layer to generate candidate itemsets at (k+1)th layer. So to carryout,thisstagecanbedividedintotwostages:MapandReducestage.Figure1 shows the data iteration phase.

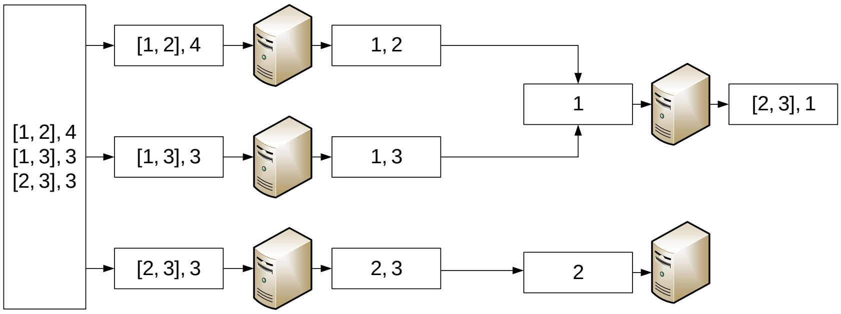

Figure 1.The Third Layer Data Generated Superset Node Distribution

4. Generation of Association Rules

Associationrulesaregeneratedwhensupportingrateisbiggerthantheminimum confidence degree after calculation of the rate in frequent itemsets. It’s done in following steps: for given frequent itemsets 1 producing association rules, check 1 each non-empty subsettogetrelativerule $a \Rightarrow (1-a)$1-aanditsconfidencedegreeis support(l) ÷ support(a); when confidence degree is bigger than the minimum confidence, the association rule is produced.

In this process, one thing to consider is that if the association rule produced by the maximum subset of frequent itemset do not meet the requirement of minimum confidence, then it is obvious that any subset frequent itemsets also cannot meet the minimum confidence requirement as well.

Take for instance frequent itemset [1-4]. If the confidence degree of 1,2,3 ⇒4 can’t suffice the minimumvalue, it’s inevitably the confidence of 1,2 ⇒ 3,4 can’t suffice the minimum degree, without consideration of subset. Hence by that feature, we can improve efficiency of overall operation during actual computation. To parallelize theprocessofproducingassociationrules,wecanassigneachinfrequentitemsetsto differentMapforgeneratingsimultaneously.Sotheparallelgenerationofassociation rules based on the MapReduce model pseudo code as follows:

```
//MapReduce Stage3
Mapper{
Map() {
a=l-1
```

```
i=1
While(confidence(l,a)>minisupport&&>i+1{
i=i++
Emit(a ⇒ (1 − a) , confidence(l,a)
a=l-1 }}}
Reducer{
Reduce(){
Emit(key,value);
}};
```

5. Experimental Analysis and Results

This section evaluates the performance of the proposed algorithm by comparing its execution time performance against proposed algorithm.

To analyze the proposed algorithm, we set up a cluster of 3 nodes. They have Intel(R) Core (TM) i3-4500U CPU @ 1.80GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s). All nodes have 8GB RAM, and a 500GB hard disk. The worker nodes are installed on Ubuntu 18.04, Hadoop 2.7.1.One is for namenode ant two of others are working as datanode. Replication is set to 2 and block size is 128MB.

Experiment 1: Performance Comparison between Apriori Algorithm and Proposed Apriori Algorithm The transaction data set for this experiment is stored as a file,Performanceanalysisofminingtimebeforeandafter improved with 3 nodes Hadoop cluster test algorithm. First, onthepremisethatthenumberofnodesintheHadoop clusterisunchanged,continuouslyincreasethenumberof itemsetsintheexperimentaldataitemset,andsetthe minimumsupporttothesame,thatis,min_sup=0.3.The experimental results are shown in Table 4.

Table 4. Comparing Apriori Algorithm with Proposed Algorithm

Trasaction itemsets	Apriori Alogorithm(s)	Proposed Algorithm(s)
1500	13.6	7.3
3000	19.1	10.6
4500	28.6	16.2
6000	40.2	28.9
7500	60.4	42.8

From the table 4, proposed algorithm is often better than classical Apriori algorithm in temporal performance, and with the increasing number of transaction item sets, apriori algorithm running on a computer can significantly improve the time of mining analysis. However, with the proposed algorithm, as the number of transaction item sets increases, the time performance is getting better and better. Because with the increase in the number of transaction items, the nodes of the distributed cluster will gradually increase. In summary, the improved proposed algorithm is superior to the apriori algorithm in temporal performance.

Experiment 2: Performance Comparison between Apriori Algorithm and Proposed Algorithm under Different Supporting Degrees.

First ,this paper tests the data set RETAIL, selects the minimum support threshold range [0.02, 0.20]. And within this range, evenly increase the step: 0.02, so there will be a threshold of 10. Then, this paper use the data set retail to run the Apriori algorithm and the proposed algorithm respectively, and record the running time (Note that the running time is second). Figure 2 shows the experimental data obtained by executing the above three algorithms. Horizontal axis: support; vertical axis: time/s.

Experiments show that the proposed algorithm runs much less time than the Apriori algorithm under different support levels. The higher the support, the longer the Apriori algorithm will run than the proposed algorithm. In summary, the temporal performance of the proposed algorithm under different support levels is always superior to the traditional Apriori algorithm.

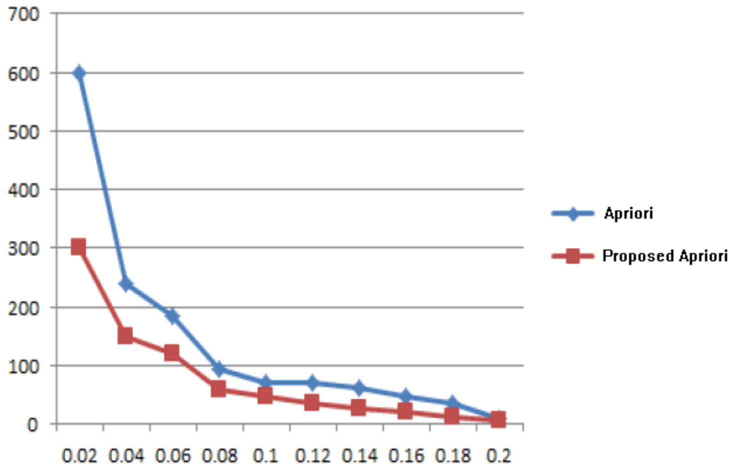

Figure 2. Performance Comparison under different support levels

6. Conclusion

Aiming at the traditional Apriori algorithm, when mining frequent itemsets, you need to continuously scan transaction data sets , So that the system I / O overhead and other shortcomings. In this paper, we improved Apriori algorithm in three aspects: compression in the transaction, reducing the number of scanning areas, and simplifying the candidate set generation. At the same time, the improved algorithm is parallelized in the Hadoop framework. The simulation results show that compared with the traditional Apriori algorithm, the proposed algorithm has good performance and security in temporal performance, mining frequent candidate itemsets and different support levels. However, it needs to be continuously improved in the future work.

References

[1] Gao Pengfei, “Research and Improvement of Apriori Algorithm Based on Hadoop”, International Journal of Advanced Network, Monitoring and Controls, Volume 03, No.03, 2018

[2] Yan Xiaofei. “Research on Association Rule Mining Algorithm”. Chongqing: Chongqing University, 2009:15-21.

[3] AGRAWAL R.SRIKANT, “Fast algorithm for mining association rules”, /Proceedings of 20th Int. Conf. Very Large Data Bases(VLDB). Morgan Kaufman Press,1994:487-499.

[4] REN W J, YU B W. “Improved Apriori Algorithm Based on Matrix Reducation”. Computer and Modern,2015,10. 2-3. (in Chinese)

[5] C. Fangjian, Z. Mingxin and Y. Kun, “MapReduce parallel implementation of the Apriori algorithm of Boolean matrix”, Journal of Changshu Institute of Technology, vol. 28, no. 17402, (2014), pp. 98-101.

[6] L. Changfang, Y. Y. Wu, H. Zhongkai and H. Shaojun, “Apriori algorithm based on MapReduce parallel”, Journal of Jiangnan University (Natural Science Edition), vol. 13, no. 7404, (2014), pp. 411-415.

[7] L. Li, “Optimization Research of Apriori algorithm based on MapReduce in the cloud computing environment”, Automation and instrumentation, no. 17707, (2014), pp. 1-4.

[8] Z. Yixue and H. Yijie, “An improved algorithm of Apriori based on MapReduce”, Journal of Lanzhou Institute of technology, vol. 21, no. 8406, (2014), pp. 13-16.

[9] W. Ling, W. Yongjiang and G. Changyuan, “Improvement of Apriori algorithm based on BigTable and MapReduce”, Computer science, vol. 4210, (2015), pp. 208-210.

[10] G. Minjie, “Analysis and processing of massive network traffic data based on cloud computing and key algorithms research”, Beijing University of Posts and Telecommunications, (2014).

[11] L. Zhiliang and L. Fang, “An improved algorithm based on Apriori Mapreduce”, Journal of Henan Institute of Education (Natural Science Edition), vol. 22, no. 8104, (2013), pp. 34-36.

[12] L. Xiaofei, “MapReduce parallelization of Apriori algorithm in cloud computing environment”, Journal of Changchun University of Technology (Natural Science Edition), vol. 34, no. 12906, (2013), pp. 736-740.

[13] F. Yanyan, “Research on Distributed Association Rule Mining Algorithm Based on MapReduce”, Harbin Engineering University, (2013).

[14] S. Fenfen, “Research on massive data parallel mining technology”, Beijing Jiaotong University, (2014).

[15] L. Shijia, “Research on frequent itemsets mining algorithm based on MapReduce framework”, Harbin University of Science and Technology, (2015).

[16] W. Daming, “Optimization Research of Apriori algorithm based on cloud computing and medical big data”, Beijing University of Posts and Telecommunications, (2015).

[17] Z. Xiaofeng, “Research and application of data mining methods for road transportation information system”, South China University of Technology, (2014).

[18] L. Hailong, “Research and application of data mining algorithm for power cloud data analysis platform”, North China Electric Power University, (2014).

[19] GUNARATHNE T , WU TL, QIU J ,et al. “MapReduce in the Clouds for Science”, IEEE Second International Conference on Cloud Computing Technology and Science (Cloudcom). IEEE,2010;565-572

[20] DEAN J,GHEMAWAT S.“ MapReduce: simplified data processing on large clusters”. Communications of the ACM, 2008, 51(1):107-113.

[21] HE B S, TAO M, YUAN X M. “Alternating direction me-thod with Gaussian back substitution for Separable convex programming”. SIAM J. Optimization, 2012, 22(2): 313-340.

[22] CHEN Z M, WAN L, YANG Q Z. “An Inexact Direction Methodfor Structured Variational Inequalities”. Journal of Optimization Theory & Applications, 2014, 163(2): 439-459.