International Journal of Database Theory and Application
Vol.9, No.12 (2016), pp.55-66
http://dx.doi.org/10.14257/ijdta.2016.9.12.06

Research on Apriori Algorithm Based on Mapreduce Model

Haili Xu[1] and Feng Qi[2,*]

[1,2]*Jiamusi College, Heilongjiang University of Chinese Medicine, Jiamusi 154007, China,*
[1]*xuhaili31@126.com,* [2]*qifeng0012@hrbeu.edu.cn*
[2,]**Corresponding author*

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 modes make people unable to make effective use of growing computational resources. Hence cloud computing appears. With the utilization of Map Reduce parallelized model, existing computing and storage capabilities are effectively integrated and powerful distributed computing ability is provided. Firstly, transform Apriori algorithm to Map Reduce model; realize Apriori parallel transformation; then use the way of compressing original transaction sets to improve the performance of Apriori algorithm in Hadoop framework; lastly, Map Reduce-Apriori algorithm is realized which is highly scalable for running in cloud computing environment.

Keywords*: Cloud computing, MapReduce, Apriori, Hadoop*

1. Introduction

Due to emergence of cloud computing, it's possible to get quickly and dynamically big cheap computing and storage ability, solving the most fundamental problem of data mining about how to acquire inexpensively powerful data computing ability [1-7]. Related researchers directed attentions to cloud computing platform, in the hope of implementing data mining algorithm with high scability, applying cheap computing of cloud computing to data mining based on storage ability, thus overcoming the shortcomings in traditional data mining, reducing calculation cost and enhancing data mining efficiency [8-14].

With a view to the broad and promising future of cloud computing, the integration of studying and applying cloud computing and existing data mining algorithm has become hot concern in various industries [15-17]. Here we transform classical data mining correlative algorithm Apriori into implementing in cloud computing environment based on Map Reduce model; meanwhile, according to characteristics of Map Reduce model, we improve Apriori algorithm to make Map Reduce-Apriori algoritm with strong scalability, fit for tremendous data analysis and processing. Finally, utilize Map Reduce-Apriori parallel algorithm to test the proposed method with running time from the perspective of data volume and computing node quantity to get practically meaningful data mining results [18].

2. Algorithm Analysis and Parallelization Transformation

Apriori algorithm generally includes two questions:

(1) Find itemsets whose support degree is smaller than the minimum value, which are named frequent itemsets.

ISSN: 2005-4270 IJDTA

(2) Used itemsets found in last step to produce association rules which suffice confidence degree.

To complete the first step, Apriori employs recursive method to search iteratively layer by layer to generate gradually frequent itemsets of all layers. The process algorithm pseudo code as follows:

```
1 L1 =find_frequent_1-itemsets(D)
2 for (k=2; Lk−1 ≠ Φ ;k++){
3 Ck =Apriori_gen( Lk−1 ,min_sup)
4 for each transaction t∈ D{
5 Ct =subset ( Ck ,t)
6 for each candidate c∈ Ct
7 c.count ++;}
8 Lk ={c∈ Ck |c.count>min_sup}}
9 return L= Uk Lk ;
```

Function Apriori_gen during the iteration is responsible for producing candidate set at the kth layer from the (k-1)th layer set. The function gives rise to superset item at the kth layer by linking to frequent itemset at the (k-1)th layer. If all subitems at the (k-1)th layer in such superset are of frequent itemset at the (k-1)th layer, it's candidate frequent itemset at the kth layer; otherwise, it's deleted. The process makes full use of features of Apriori. Any non-empty subset in frequent itemset is certainly that frequent itemset deminishes the number of candidate itemsets. The pseudo code of the algorithm is described as follows:

```
1 Apriori_gen( Lk−1 ,minsup)
2 for each itemsets1 in Lk−1
3 for each itemsets2 in Lk−1
4 if (itemsets1[1]=itemsets2[1]&&( itemsets1[2]=itemsets2[2])&&..
      ( itemsets1[k-1]=itemsets2[k-1])
5 c=itemsets*itemsets2
6 if has infrequent_subset(c, Lk−1 ) then
            Delete c
        Else add c to Ck
7 return Ck ;
```

3. Data Initialization

In order to express convenience, the meaning of each symbol is defined as Table1:

Table 1. Symbolic Meaning Table

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

Calculating frequent itemset at data initialization stage is actually numerical statistics of the entire dataset; then screen out frequent set. When the 1st itemset is being produced, without judging too much, arrange and combine directly items in frequent set. That is algorithmically much simpler than iteration; however it has more huge computational data amount. So this process is regarded as data initializing stage.

During data initializing stage, data processing includes three stages:

(1) Produce frequent itemsets;

(2) Ranking of frequent itemsets;

(3)Generate 2nd layer candidate itemsets. Hereunder we presents the parallelized implementation at data initialization stage based on Map Reduce model. The results are 2nd candidate itemsets after order ranking, which is called C_2. The pseudo code flow of this phase is shown below:

```
//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])
    }}
```

In the second stages, we sort results got in the first stages according to support degree; then map results after ranking into id expressed by number, which is named GMap. GMap transforms frequent set data from character strings into digit group, reducing greatly data

transmission in Hadoop, increasing frequent set comparing speed and optimizing system performance.

Results at data initializing stage are 2nd layer candidate itemsets grouped by host nodes. The data flow expression form of entire data at initializing stage is shown in Table 2.

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	1, a, 4	(1,12)	(1,[12,12,14]
abcde	c ,4	2, c ,4	(1,13)	(2,[23,24])
bdgac	d ,3	3 ,d ,3	(1,14)	(3,[34])
adc	g, 3	4 ,g, 3	(2,23)	
fhg			(2,24)	
			(3,34)	

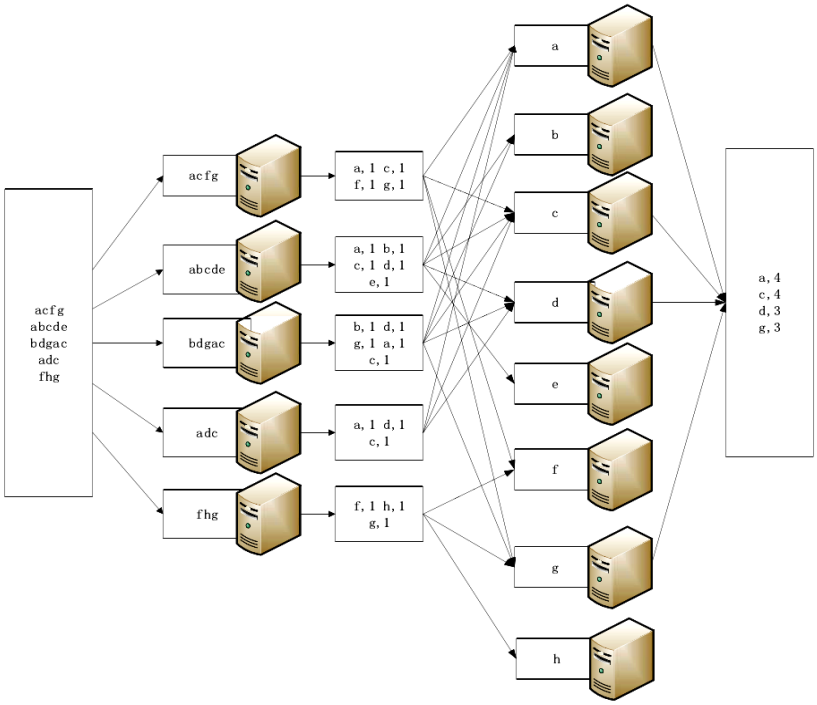

Figure 1. Distribution of Data Nodes in Support of 50%

The distribution graph of ideal data nodes for calculating 1st layer frequent itemsets after Map Reduce is put in Figure 1. In the picture1, each machine each Map process at Map stage represents and Reduce process at Reduce stage. At Map stage, the box at the left of machine stands for input data into the Map; box at the right stands for data output at Map stage. At Reduce stage, box at the left of machine means clustered key, *e.g.* box with mark "a" meaning all data outputing "a" as key at Map stage are clustering in Reduce; box at the right of Reduce referring to final output data at Reduce stage.

Figure 2 presents data node distribution when 2nd layer candidate itemsets are generated. Since at stage 1-2, 1st layer frequent itemsets are sorted sequentially and digitally mapped, input data at this stage are converted to pure digits. Similarly the picture shows implementation in ideal condition when complete distribution resources are always

sufficient. From data distribution graph, we can see during the stage, serious load imbalance exists: in the 1st Map computation, it needs to produce three candidate itemsets, without operations at the last Map step. So making load balancing at this stage is effective measure to improve efficiency at the stage.

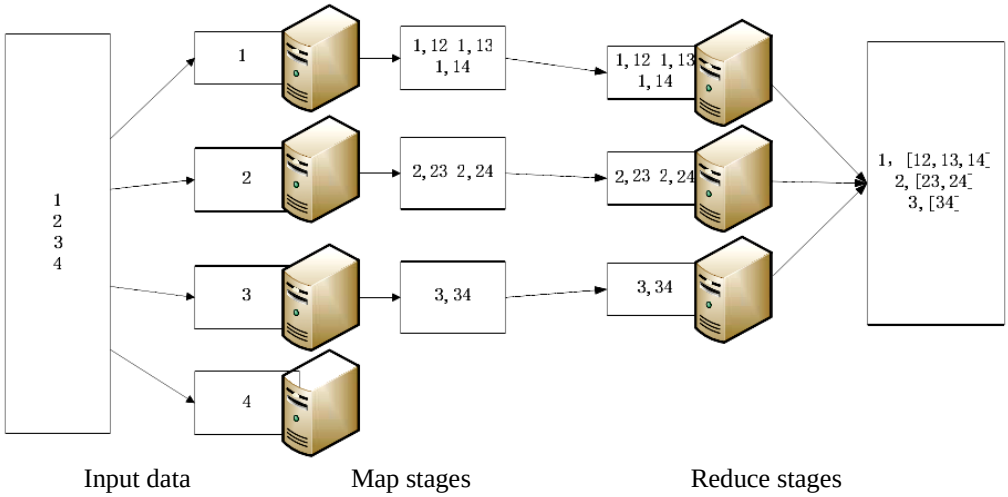

Figure 2. Generating Second Layer Candidate Set Data Node Distribution Graph

4. Iterative implementation

After data initializing stage is over, 1st frequent itemsets and 2nd layer candidate itemsets are produced. During iterative stage, all frequent itemsets are generated on that basis. This period has two steps:

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

4.1 Calculate Frequent Itemsets at the kth layer

Map input is file's row data; output key is value of each column, which is 1; to speed up calculation time, candidate frequent itemsets and GMap at the kth layer are read into internal memory at this stage. The pseudo-codes are implemented as follows:

```
//MapReduce Stage2-1
Mapper{
Setup(C_k)
Map() {
value=MapToid(value,GMap)
value.sort()
For each itemsets in C_k
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 is the data stream in the calculation of the second layer frequent set.

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			

Figure3 is data node distribution graph of frequent itemsets generated at the 2nd layer. From the picture, we find steps and method for producing frequent itemsets at the 2nd layer are basically identical to the method for generating itemsets at the 1st layer. The only difference is input data at 2nd layer is no longer transaction set and candidate itemsets at the 2nd layer are delivered, because at Map stage, it needs to compare candidate sets as to get ultimate statistical counting. We note that since candidate itemsets at the 2nd level are bigger dataset, there will have big bottleneck at this stage. To enhance effciency at the stage, we can selectively read in candidate itemsets according to characteristics of transaction sets.

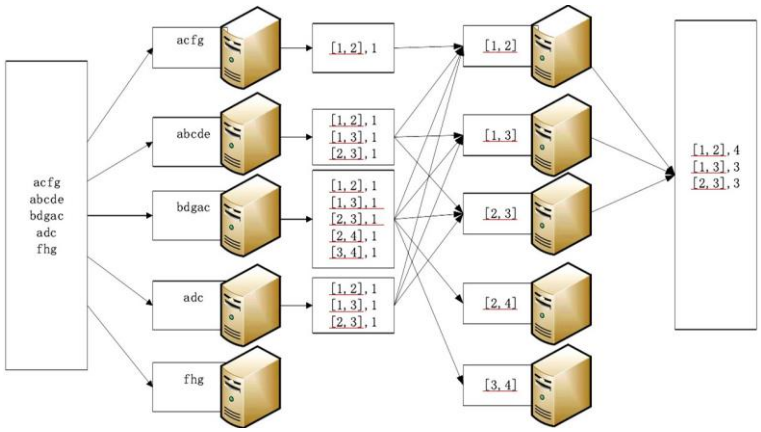

Figure 3. Generating Second Layer Frequent Set Data Node Distribution Graph

4.2. Calculate Candidate Itemsets at the (K+1) Th Layer

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 carry out, this stage can be divided into two stages: Map and Reduce stage. Figure 4 shows the data iteration phase.

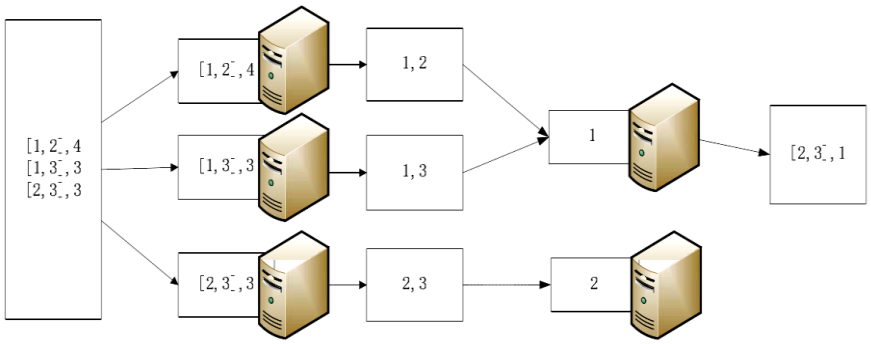

Figure 4. The Third Layer Data Generated Superset Node Distribution

5. Generation of Association Rules

Association rules are generated when supporting rate is bigger than the minimum 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 subset a to get relative rule $a \Rightarrow (l-a)$ 1-a and its confidence degree is support(l) ÷ support(a); when confidence degree is bigger than the minimum confidence, the association rule is produced.

The generation of association rules has these features: when the association rule produced by the maximum subset of frequent itemset can't meet with the minimum confidence, then it's believed smaller subset in that subset can't meet with the minimum confidence as well. Take for instance frequent itemset [1-4]. If the confidence degree of 1,2,3 ⇒4 can't suffice the minimum value, it's inevitablely 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 the process of producing association rules, we can assign each in frequent itemsets to different Map for generating simultaneously. So the parallel generation of association 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 ⇒ (l − a) , confidence(l,a)
a=l-1 }}}
Reducer{
Reduce(){
  Emit(key,value);
}};
```

6. Experimental Analysis and Results

6.1. Experimental Data Set

The experimental data this time were chosen from all call data in August 2014 in some place, of which daily data is about 10 billons in the format of text of size 40G around. During the mining, in order to protect user privary, we took samples of original data, choosing 10G, 20G, 30G for testing.

6.2. Experimental Test Analysis

To validate the scalable performance of Map Reduce-Apriori algorithm, here we do experiments on testing time from the angle of data volume and computing node number and analyze results of data mining, showing significance of data mining results.

To verify the computational efficiency and expansion capability of Map Reduce-Apriori, in the experiment, we use 10G, 20G, 30G to run on respectively 5,10,15,20 operating nodes with support degree of 5%, 10% and 15%.

The first group of experiment is about 10G data performing on 5,10,15,20 operating nodes. 10G data includes approximately 100 million pieces of transactions. It is shown in Figure5.

From experimental results, we note that with increasing data volume, Map Reduce-Apriori algorithm's calculation time is basically growing linearly, which manifests the feature of Hadoop parallelizing data as per size of data block. Figure 9 presents the scalable performance of Map Reduce-Apriori algorithm when data size is 10GB and support degree is 5%. It's obvious that with more and more nodes, Map Reduce-Apriori algorithm keeps a better scalability, proving that the algorithm based on Hadoop platform is quite suitable for cloud computing application, able to expand effectively computation to calculate resources, improving the overall performance.

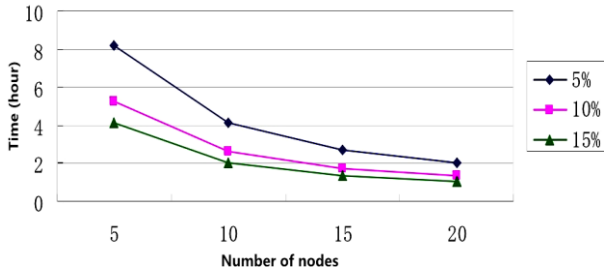

Figure 5. Test Performance in 10G Data

The second group of experiment is about 20G data performing on 5,10,15,20 operating nodes. 20G data includes approximately 200 million pieces of transactions. It is shown in Figure6.

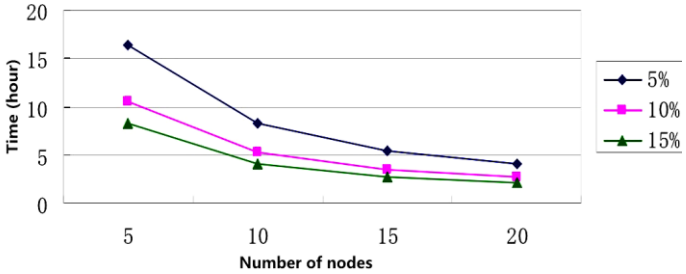

Figure 6. Test Performance in 20G Data

The third group of experiment is about 30G data performing on 5,10,15,20 operating nodes. 30G data includes approximately 300 million pieces of transactions. It is shown in Figure7-Figure8.

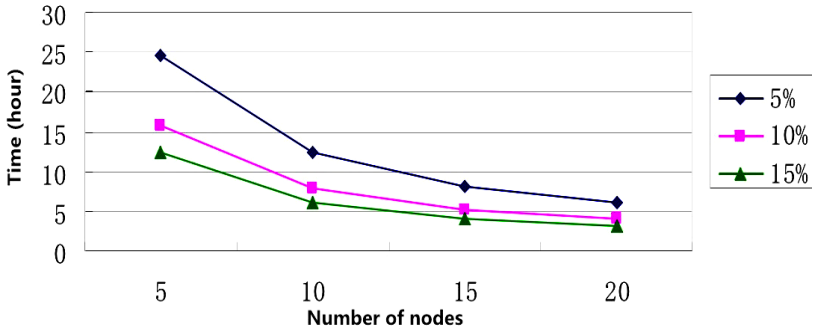

Figure 7. Test Performance in 30G Data

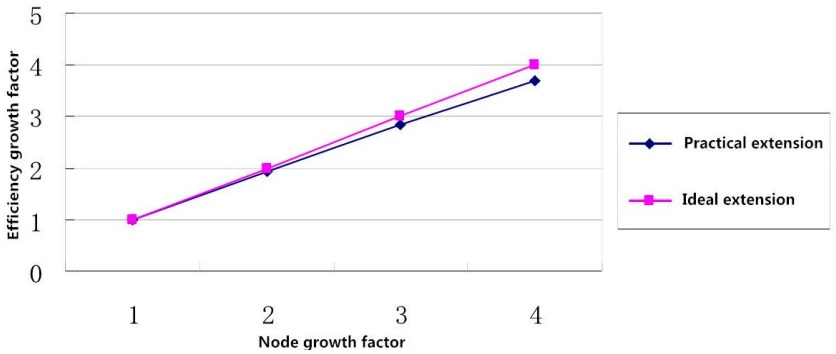

Figure 8. Linear Expansion Capability

6.3. Analysis of Data Mining Results

After testing on scalability is over, we make data mining of call data. Relative association rules are generated about the whole call record and personal call record. Hereunder we analyze those rules.

To discover the whole call record, we set support degree 5% and confidence degree 10%; meanwhile to make conversation time more significant, during mining, we convert conversation time to day of the week, daytime or nighttime. Daytime refers to 9AM to 9PM; the rest refers to nighttime. If conversation time is 05, which is transformed to Monday daytime for associated data mining. It is shown in Figure9.

7. Conclusion

In this paper, we find that the use of Apriori association data mining algorithm to analyze call data record is helpful to dig out personal behavior features of user. So association mining results are practically meaningful to something like public security department which focuses on individuals. Limited by the data dimension of conversation records, the mining results we got is commercially less valuable. To make mining results good for business decision, it requires data containing business information, such as user's consumption records bound with conversation record. Then with the proposed Map Reduce-Apriori algorithm for mining, it's likely to acquire more practical and meaningful association information.

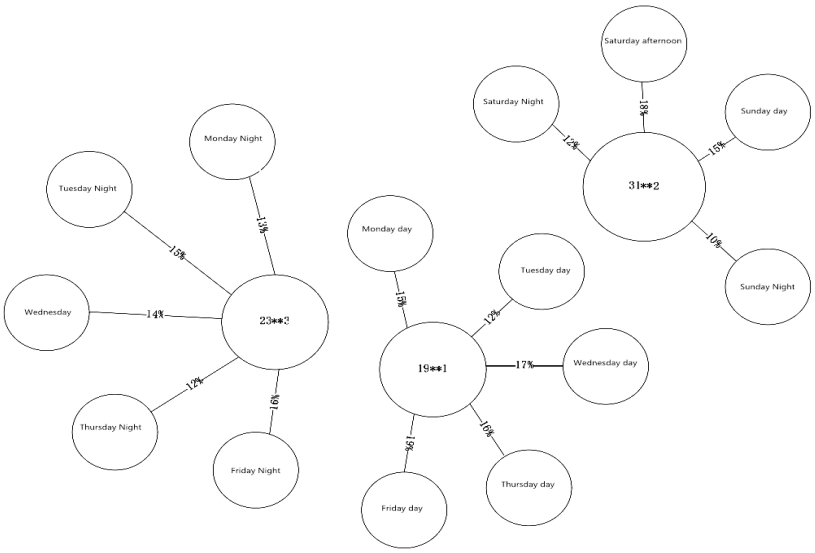

Figure 9. Overall Call Record Mining Results

References

[1] H. Liqin and L. Yanhuang, "Research on the improvement of Apriori algorithm based on MapReduce parallel algorithm", Journal of Fuzhou University (Natural Science Edition), vol. 39, no.18305, **(2011)**, pp. 680-685.
[2] Z. Danping and T. Guoqiang, "Research on MapReduce algorithm based on Apriori in the cloud computing environment", Jiangxi communication science and technology, no.11802, **(2012)**, pp. 16-19.
[3] Z. Zhigang and J. Genlin, "Based on iterative MapReduce Apriori algorithm design and implementation", Journal of Huazhong University of science and Technology (Natural Science Edition), vol. 40, no. 355, **(2012)**, pp. 9-12.
[4] 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.
[5] 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.
[6] L. Li, "Optimization Research of Apriori algorithm based on MapReduce in the cloud computing environment", Automation and instrumentation, no. 17707, **(2014)**, pp. 1-4.
[7] 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.
[8] W. Ling, W. Yongjiang and G. Changyuan, "Improvement of Apriori algorithm based on BigTable and MapReduce", Computer science, vol. 4210, **(2015)**, pp. 208-210.
[9] 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)**.
[10] 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.
[11] 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.
[12] F. Yanyan, "Research on Distributed Association Rule Mining Algorithm Based on MapReduce", Harbin Engineering University, **(2013)**.
[13] S. Fenfen, "Research on massive data parallel mining technology", Beijing Jiaotong University, **(2014)**.
[14] L. Shijia, "Research on frequent itemsets mining algorithm based on MapReduce framework", Harbin University of Science and Technology, **(2015)**.
[15] W. Daming, "Optimization Research of Apriori algorithm based on cloud computing and medical big data", Beijing University of Posts and Telecommunications, **(2015)**.
[16] Z. Xiaofeng, "Research and application of data mining methods for road transportation information system", South China University of Technology, **(2014)**.
[17] Z. Anzhu, "Research on improvement and transplantation of Apriori algorithm based on Hadoop", Huazhong University of Science and Technology, **(2012)**.
[18] L. Hailong, "Research and application of data mining algorithm for power cloud data analysis platform", North China Electric Power University, **(2014)**.

Author

Haili Xu, She received her B.S degree from Qiqihar University and received her M.S degree from Harbin Engineering University. She is a lecturer from Jiamusi College, Heilongjiang University of Chinese Medicine. She is in the research of Network security, Software engineering.