Research Journal of Applied Sciences, Engineering and Technology 6(9): 1669-1673, 2013
DOI:10.19026/rjaset.6.3888
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 19, 2013 Accepted: March 02, 2013 Published: July 15, 2013

Research Article

Design and Realization of user Behaviors Recommendation System Based on Association rules under Cloud Environment

[1]Wei Dai and [2]Peng Hu
[1]School of Economics and Management,
[2]School of Mathematics and Physics, Hubei Polytechnic University, Huangshi 435003, China

Abstract: This study introduces the basal principles of association rules, properties and advantages of Map Reduce model and Hbase in Hadoop ecosystem. And giving design steps of the user's actions recommend system in detail, many time experiences proves that the exploration combined association rules theory with cloud computing is successful and effective.

Keywords: Association rule, cloud computing, mapreduce, hbase

INTRODUCTION

Association rules (Zheng *et al*., 2001), are classical and effective data mining method, it is used in many circumstances, such as market basket analysis, library transactions records. But association rule method meet velocity performance bottleneck in face of mass data sets. Through reforming association rule method with Map Reduce, we can rapidly gain association rules results by introducing cloud computing compute capacity. This project is supported by imbursement of science and technology dissertations online of China (EB/OL) for increasing the users' loyalty; we dispose a mass of user's actions historical records and giving design steps of the user's actions recommend system in detail, many time experiences proves that the exploration combined association rules theory with cloud computing is successful and effective, which have valuable recommend information to improving user experience.

COMPUTATIONAL METHODS

Association analysis: I = $\{i_1, i_2, \ldots, i_d\}$ is all item sets in data analysis, T = $\{t_1, t_2, \ldots, t_N\}$ is all transactions sets. Set which involves 0 or many items are named as itemset. A itemset which involves k items is called k-itemset. Transaction width means item number of a Transaction.

Definition 1: Support count: transaction number which certain itemset in all transaction sets. In mathematic, itemset X's support count σ(X) is expressed:

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}|$$

Symbol $|.|$ express element number of set.

Definition 2: Association rule: is contained express like X→Y, and X∩Y= Ø. The strength of association rule can be measured by support and confidence. Support shows frequent degree to certain data set and confidence shows Y's frequent degree in transactions which contains X. Support(s) and confidence(c) can be defined formalized as following (Cheung *et al*., 1996):

$$s(X \to Y) = \frac{\sigma(X \cup Y)}{N} \quad (1)$$

$$c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (2)$$

Association rule mining task can be decomposed into 2 steps:

Step 1: Generating frequent item set: The object is to find out all itemset that satisfy minimal support threshold.

Apriori principle all its sub sets are also frequent while a itemset is frequent.

At the beginning, every item is regarded as candidate 1-itemset. Some item are cut after pruning based on support count. The other become formal 2-itemset. And then formal 2-itemset are used to generate candidate 2-itemset by special function.

Apriori (D, minsup) {K=1
Repeat
K = k+1
C_k = apriori-gen(F_{k-1}) //generate candidate itemset

Corresponding Author: Wei Dai, School of Economics and Management, Hubei Polytechnic University, Huangshi 435003, China

for every t∈T do
C_t = subset($C_{k,t}$) //distinguish all candidate itemset of t
For every candidate itemset c∈ C_t do
σ(c) = σ(c) +1 //support increase
End for
End for
F_k = {c|$c \in C_k \wedge \sigma(c) \geq$ N×minsup} //refine frequent k-itemset
Until F_k = Ø
Result = ∪ F_k
}
Apriori algorithm (Han and Kambr, 2001)
Proceduce apriori_gen (L_{k-1}: frequent (k-1)-itemsets; minsup: min support threshold)
For each itemset $I_1 \in L_{k-1}$
For each itemset $I_1 \in L_{k-2}$
If($(I_1[1] = I_2[1]) \wedge ... \wedge (I_1[k-1] = I_2[k-1])$)
Then
{c = $I_1 \cup I_2$;
If has_infrequent_subset (c, L_{k-1})
Then delete c;
else add c to C_k;
}
Return C_k;
Proceduce has_infrequent_subset (c:candidate k-itemset; L_{k-1}: frequent (k-1)- itemset)
for each (k-1) –subset s of c
If (!(s∈ L_{k-1}))
Then return TRUE;
Else return FALSE;

Step 2: Generating rules: The object is to extract all rules with high confidence from all frequent itemsets.

Association rules can be extracted like this: Itemset Y is divided into non-empty two sub sets X and Y-X, simultaneously X→Y-X must satisfy with confidence threshold. Rule's confidence can be calculated by formula σ({X∪{Y-X}})/σ({X}).

We can generate 2^k-2 association rules from every frequent k-itemset because rules which like Ø→Y or Y→Ø is ignored.

Theorem 1: If rule X→Y-X can not satisfy with confidence threshold, rules like X'→Y-X'must can not satisfy with confidence threshold, that X' is sub set of X.

For every frequent k-itemset f_k, k ≥ 2 do
H_1 {i|$i \in f_k$}//rule's 1-itemset consequent
Call ap-genrules (f_k, H_1)
End for
Proceduce ap-genrules (f_k, H_m)
K = |f_k|//frequent itemset size
m = |H_m|// rule consequent size
If k>m+1 then
H_{m+1} = apriori – gen(H_m)
For every $h_{m+1} \in H_{m+1}$do
Conf = σ(f_k) / σ(f_k- h_{m+1})
If conf≥ min conf then
Output: rule (f_k- h_{m+1})→ h_{m+1}
Else
Delete h_{m+1} from H_{m+1}
End if
End for
Call for
Call ap-genrules (f_k, H_{m+1})
End if

Mapreduce model: MapReduce (Dean and Ghemawat, 2004) which it is invented by google company is a simplified distributed model, it is often used in parallel computing of mass data set. Its stick programming model makes program simple under cloud environment. MapReduce decomposes the problem that needs to be processed into two steps-map stage and reduce stage. Data sets are divided into unrelated blocks, which are respectively deposed by every compute in whole distributed cluster and then reduce stage output ultimate result by collecting all mid results. MapReduce framework uses master-slaves architecture. Master runs a JobTracker, which manages work sub tasks allocation of a job and monitors their run circumstances; master will demand to rerun them when many tasks fail, while every slave runs a Task Tracker, which carries out computing task to small data block of data sets. Computing task allocation observes the rules that data block location. It adequately embody 'moving computing is easier than moving data' in distributed system design. Figure 1 shows detail dispose process of MapReduce model.

Hbase architecture: Hbase (http://hbase.apache.org/) follows a construction of master-slave server, every Hbase cluster always involve a master server and multiple regionservers. Every region comprises of successive record rows in a table, from start key to end key. And then all rows of a table are saved in a series of regions. Different regions are made a distinction by table name and start key or end key. Every table can be divided into multiple sub tables, which are managed by regionserver and master assign them to regionserver. Hbase contains the following conceptions. Rowkey, the only identifier of a row, can be any character string; it is saved as byte array. When storage, data record sorted by byte order of rowkeys. Column Family, is a basal unit of access control, disk and memory' use count, is a table scheme design. Qualifier, further partition under Column Family, qualifier name is used with Column Family prefix. Cell, fixed a crossed storage

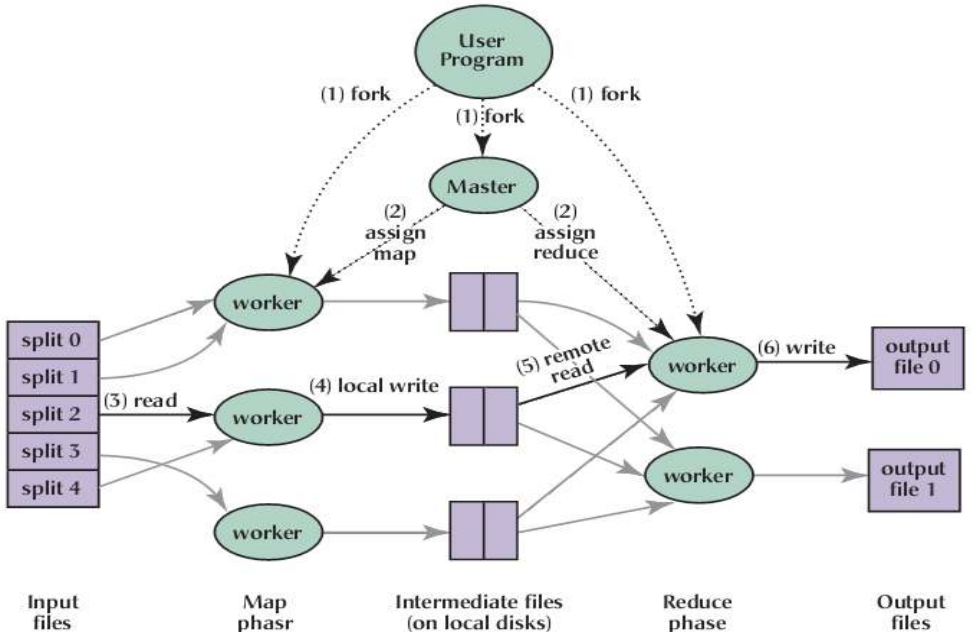

Fig. 1: MapReduce model depose procedure

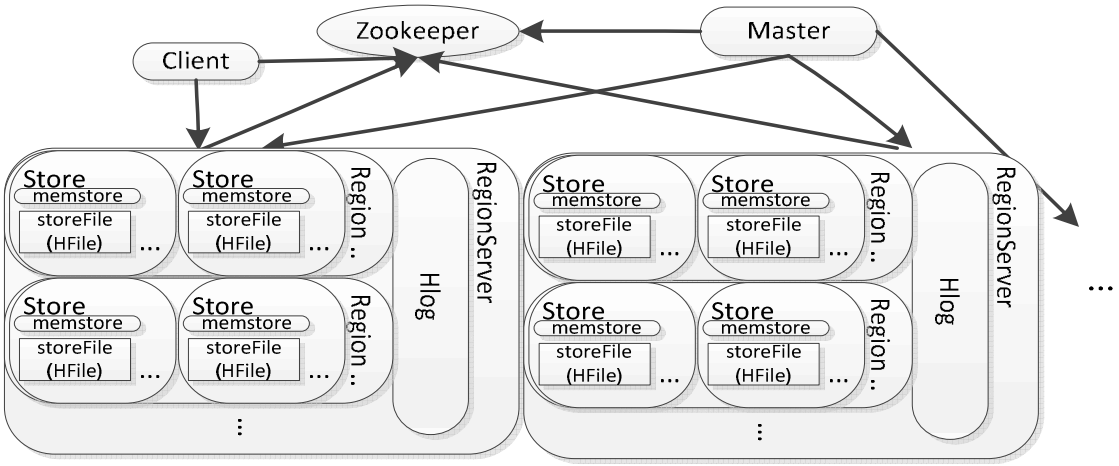

Fig. 2: Hbase architecture

unit with row and column Qualifier, every cell stores different vision of data, which distinguish by timestamp in Fig. 2 (http://hbase.apache.org/).

Column-Oriented property and Hbase's advantages: Column-oriented databases save their data grouped by columns. Subsequent column values are stored contiguously on disk. This differs from the usual row-oriented approach of traditional databases, The reason to store values on a per column basis instead is based on the assumption that for specific queries not all of them are needed. Especially in analytical databases this is often the case and therefore they are good candidates for this different storage schema. Reduced IO is one of the primary reasons for this new layout but it offers additional advantages playing into the same category: since the values of one column are often very similar in nature or even vary only slightly between logical rows they are often much better suited for compression than the heterogeneous values of a row-oriented record structure: most compression algorithms only look at a finite window. Specialized algorithms, for example delta and/or prefix compression, selected based on the type of the column (i.e., on the data stored) can

yield huge improvements in compression ratios. Better ratios result in more efficient bandwidth usage in return.

Hbase is a sub-project of Hadoop (http://hadoop.apache.org/), is a data manage software built in HDFS (http://hadoop.apache.org/hdfs/) (Chen *et al*., 2011) distributed file system. HBase stores data on disk in a column-oriented format, but it is not a Column-oriented database through and through. It is distinctly different from traditional columnar databases: whereas columnar database excel at providing real-time analytical access to data, HBase excels at providing key-based access to a specific cell of data, or a sequential range of cells.

RESULTS AND DISCUSSION

Hbase data tables structure of this system: We full use the advantages of Hbase, design the following Hbase tables to find out association rules. ArticlesDetail table stores every article's detail information. ArticleID is its rowkey, string 'f' is Column Family, 'ArticleID_Author1, Author2, Author3_Author1Dep, Author2Dep, Author3Dep' is Qualifier, null is corresponding value. OriginalTransactions table stores every transaction's detail information. TransactionID is its rowkey, string 'f' is Column Family, 'ArticleID1, ArticleID2, ArticleID3' which ArticleIDs in every download is Qualifier, null is corresponding value in Table 1.

We orderly generate all download articles' sub sets of every row record according to lattice structure (Zhao *et al*., 2000), simultaneity insert every sub set item into transactionsSupport table witharchitecture that sub set item is its rowkey, string 'f' is Column Family, TransactionIDs like 't1_t2_t3_...' is Qualifier, TransactionID number is corresponding value. If certain item appear in a new transaction, Qualifier will be appended by '_new TransactionID' and value(TransactionID number) should be increase by Table 2.

Table 1: Hbase original transactions

Rowkey1 = t1	f ArticleDI.ArticleD2.ArticleD3.. Null
Rowkey2 = t2	f ArticleDI.ArticleD2.ArticleD3.. Null

Table 2: Hbase transactions support

Rowkey1 = sub set ilem 1	f t1_t2_t3_.. 3
Rowkey2 = sub set ilem2	f t1_t2_t3_t7.. 4

Lattice structure is used to orderly enumerate all potential itemset. Figure 3 shows the lattice structure of I = {a, b, c, d}.

Every item that generate by apriori algorithm will be inserted into frequentItems table with architecture that sub set item is its rowkey, string 'f' is Column Family, Item's support is Qualifier, null is corresponding value in Table 3.

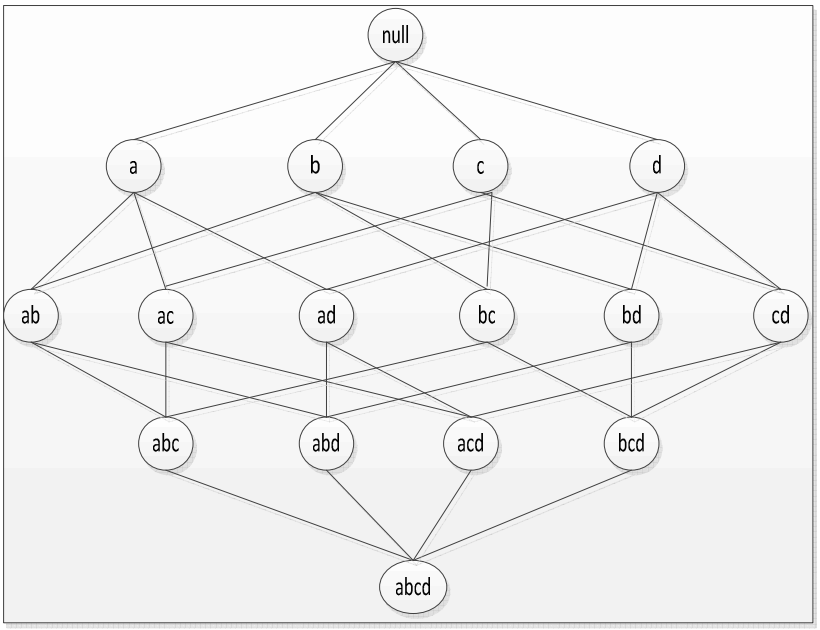

Fig. 3: Lattice structure

Table 3: Hbase frequentItems

Rowkey1 = sub set ilem 1	f Item1's support null
Rowkey2 = sub set ilem2	f Item2's support null

Table 4: Computing time contrast

Record number	UnMap reduce time (ms)	Map reduce time (ms)
500	3367	355
1000	5904	612
1500	11140	887
2000	15658	1249

Parallel dispose frequentItems table's items with MapReduce mode: After generating frequentItems according to apriori algorithm, all frequent Items are stored in frequentItems table; we can parallel generate association rules with adequate confidence by MapReduce program. Because every k-Item can generate many association rules, MapReduce mode can improve dispose process by full using compute cluster. Table 4 describes the contrast of two ways in 5000 records.

CONCLUSION

The study describes user behaviors recommendation system design approach based on association rules and cloud computing in detail, make full use of the computing ability of cloud computing, design Hbase tables smartly, improve the computing course and can generate association rules rapidly. The system improves user experience to some extent, improves recommend response time largely, it is proved to be a successful exploration. Base on the established system, providing more and complicated models will be the future work.

AKNOWLEDGMENT

The research has been financially supported by School-level innovative talents project (Grant No.12xjz20C).

REFERENCES

Chen, Z., Y. Xu, X.J. Wang and Y.L. Jin, 2011. A new fault tolerance system for cloud storage. J. Convergence Inf. Technol., 6(4): 34-41.

Cheung, D.W., J.W. Han, V.T. Ng, A.W. Fu and Y.J. Fu, 1996. A fast distributed algorithm for mining association rules. IEEE 4th International Conference on Parallel and Distributed Information Systems, Miami Beach, Florida.

Dean, J. and S. Ghemawat, 2004. Map reduce: Simplified data processing on large clusters [C]. Proceedings of OSDI '04: 6th Symposium on Operating System Design and Implementation, San Francisco, CA, pp: 137-150. Science and Technology Dissertations Online of China [EB/OL]. Retrieved from: http://www. paper. edu.cn/.

Han, J. and M. Kambr, 2001. Data Mining: Concepts and Techniques [M]. Higher Education Press, Beijing.

Zhao, Y., J.L. Shi and P.F. Shi, 2000. A limited lattice structure for incremental association mining. Springer: PRICAI 2000 Topics in Artificial Intelligence, pp: 102-103.

Zheng, Z., R. Kohavi and L. Mason, 2001. Real world performance of association rule algorithms. Proceeding of 2001 ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, ACM, pp: 401-406.