TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.2, February 2014, pp. 1480 ~ 1486
DOI: http://dx.doi.org/10.11591/telkomnika.v12i2.3443

Research on Personalized Behaviors Recommendation System Based on Cloud Computing

Wei Dai*[1], Peng Hu[2]
[1]School of Economics and Management, Hubei Polytechnic University, Huangshi 435003, China
[2]School of Mathematics and Physics, Hubei Polytechnic University, Huangshi 435003, China
*Corresponding author, e-mail:dweisky@163.com

Abstract

Data scale becomes a bottleneck in user behaviors analysis, many data mining algorithms become Inefficient slow in this circumstances. This paper explores an effective approach to mine latent knowledge in large scale data, which combines the basal principles of association rules, MapReduce model and Hbase database. First, general principles and algorithm of association rules are given. Second, the work mechanism and traits of MapReduce model and HBase are introduced. Finally, it gives detailed design methods that how to combine the basal principles of association rules, MapReduce model and Hbase database. Sufficient experiments prove that the processing velocity of parallel approach nearly decuple unparallel approach's. Therefore, the approach combined association-rule and cloud computing is a successful and valuable exploration.

Keywords: *cloud computing, data mining, association rules, MapReduce, Hbase*

1. Introduction

Many data mining algorithms are used in user behaviors analysis field. Association-rule [1], is classical and effective data mining method, which is used in many circumstances, such as market basket analysis, library transactions records. It is also used widely to analyze users' browse and download behaviors. But association rule method meet velocity performance bottleneck to mass data sets of user behav*iors*. In this circumstances, algorithm's execution velocity becomes slow so much so that beyond the gauge of our tolerance. Through reforming association rule method with MapReduce, we can rapidly gain association rules results by introducing cloud computing compute capacity. This project aims at analyzes users' browse activities to improve users' loyalty to target website, which we choose Chinese National Knowledge Infrastructure [http://www.cnki.net/]. We dispose historical records and web log about user behaviors in half a year, and give detailed design steps of the user behaviors recommend system. Sufficient experiments prove that the exploration combined association rules theory with cloud computing [2, 3] is successful and effective, which have valuable recommend information to improve user experience.

2. The Proposed Algorithm

2.1. Association Analysis

$I=\{i_1,i_2,\cdots,i_d\}$ is all item sets in data analysis, $T=\{t_1,t_2,\cdots,t_N\}$ 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 $\sigma(X)$ is expressed:

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

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

Received June 10, 2013; Revised September 6, 2013; Accepted September 28, 2013

Definition 2: Association rule: is contained express like $X \to Y$, and $X \bigcap Y = \varnothing$. 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 [4]:

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

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

Association rule mining task can be decomposed into 2 steps:

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

Apriori principle: all its sub sets are also frequent while an 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 becomes formal 2-itemset. And then formal 2-itemset is 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
for every $t \in T$ do
$C_t = subset(C_k, t)$ //distinguish all candidate itemset of t
For every candidate itemset $c \in C_t$ do
$\sigma(c) = \sigma(c) + 1$ //support increase
End for
End for
$F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq N \times \min\sup\}$ //refine frequent k-itemset
Until $F_k = \varnothing$
Result = $\cup F_k$
}

Apriori algorithm [5]
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 $I_2 \in L_{k-1}$
$If((I_1[1] = I_2[1]) \wedge ... \wedge (I_1[k-1] = I_2[k-1]))$
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(!($s \in L_{k-1}$))

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 $X \to Y - X$ must satisfy with confidence threshold. Rule's confidence can be calculated by formula $\sigma(\{X \bigcup \{Y - X\}\}) / \sigma(\{X\})$.

We can generate $2^k - 2$ association rules from every frequent k-itemset because rules which like $\varnothing \to Y$ or $Y \to \varnothing$ is ignored.

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

For every frequent k-itemset f_k, k $\geq$2 do

$H_1 = \{i \mid i \in f_k\}$ //rule's 1-itemset consequent

Call ap-genrules (f_k, H_1)

End for

Proceduce ap-genrules (f_k, H_m)

K= $\mid f_k \mid$ //frequent itemset size

$m = \mid H_m \mid$ // 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

$conf = \sigma(f_k) / \sigma(f_k - h_{m+1})$

If $conf \geq \min conf$ then

Output: rule $(f_k - h_{m+1}) \to h_{m+1}$

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

2.2. MapReduce Model

MapReduce [6] 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 TaskTracker, which carries out computing task to small data block of data sets. Computing task allocation observes the rules that data block location. It adequately embodies 'moving computing is easier than moving data' in distributed system design. Figure 1 shows detail dispose process of MapReduce model.

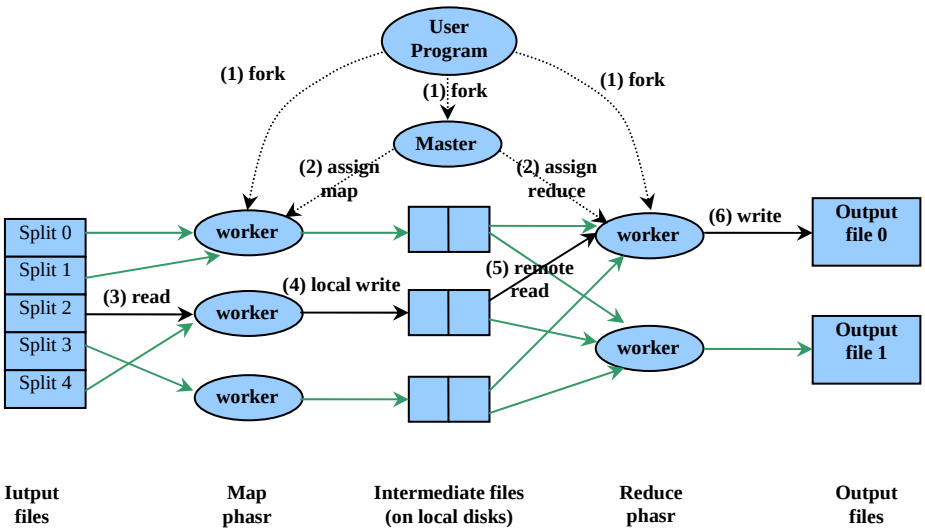

Figure 1. MapReduce Model Dispose Procedure

2.3. Hbase architecture

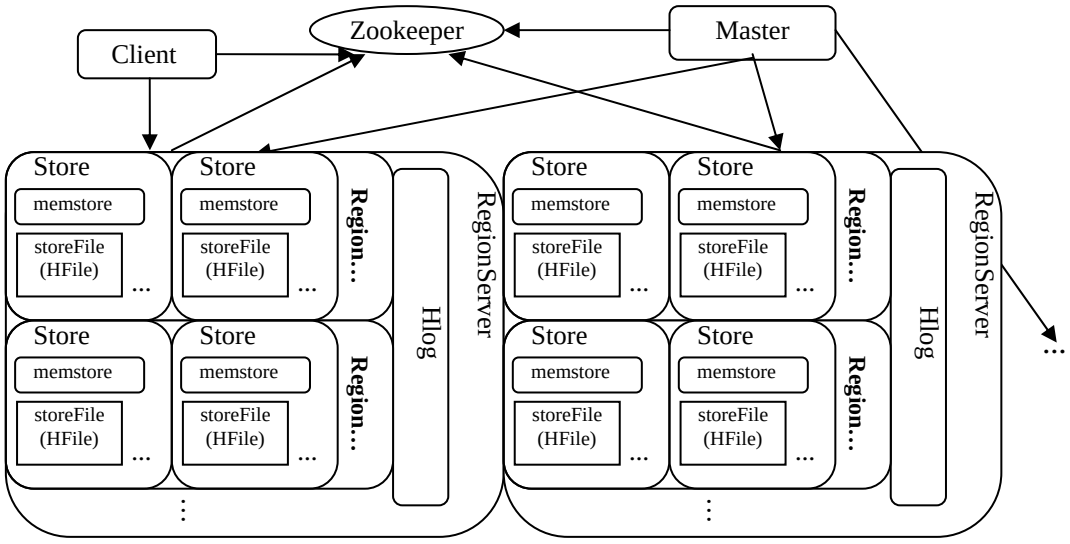

Figure 2. The Structure Diagram of Hbase architecture

.

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 unit with row and column Qualifier, every cell stores different vision of data, which distinguish by timestamp in Figure 2.

2.4. 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, is a data manage software built in HDFS [7] 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.

3. Research Method

3.1. Design and Method of Hbase Tables

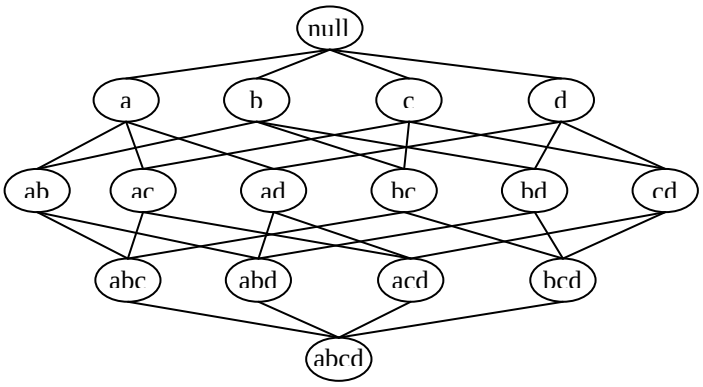

Figure 3. An Example of Lattice Structure

With the advantages of Hbase's alphabet order [8], the following Hbase tables are designed to find out association rules in large scale Original Transactions sets. 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. Original Transactions 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.

We orderly generate all download articles' sub sets of every row record according to lattice structure[9, 10], simultaneity insert every sub-set's item into transactionsSupport table with architecture 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 appears in a new transaction, Qualifier will be appended by '_new TransactionID', and value (TransactionID number) should be increased by 1.

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 generated 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.

4. Results and Discussion

After generating frequent items according to apriori algorithm, all frequent Items are stored in frequent items 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. Figure 4 shows clearly that processing velocity rate improved significantly with the growth of data scale.

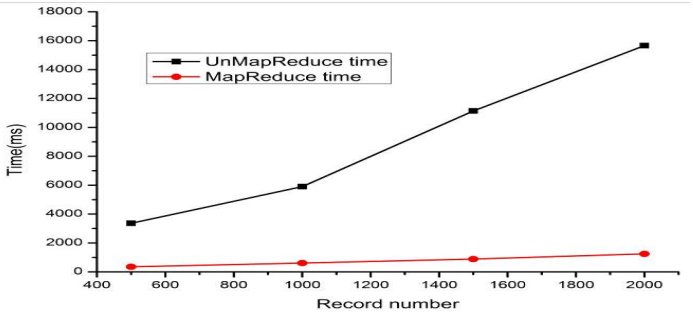

Figure 4. Time-consuming Comparison

5. Conclusion

A design approach of user behaviors recommendation system combined association rules and cloud computing is expounded. It takes full advantage of the powerful computing ability of cloud computing, improves the computing course by smartly designing Hbase tables according to its alphabet order, and can generate association rules rapidly. This system improves user experience to some extent, advances recommend response time largely, and it is proved to be a successful exploration. Therefore, more complicated models and efficient methods will be provided in the future work.

Acknowledgements

This study has been financially supported by Humanities and Social Science Youth Foundation of Ministry of Education Project (Grant No.13YJCZH028) and School-Level Innovative Talents Project (Grant No.12xjz20C).

References

[1] Zheng Z, Kohavi R, Mason L. *Real orld performance of association rule algorithms*. Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York. 2001; 1: 401-406.

[2] Zhou LJ, Wang H, Wang WB. Parallel implementation of classification algorithms based on cloud computing environment. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2012; 10(5): 1087-1092.

[3] Cheng, FC, Lai, WH. Creating the environment for the prosperity of cloud computing technology. *Telkomnika.* 2012; 10(4): 864-875.

[4] Cheung DW, Han J, Ng V, Fu A, Fu Y. *A fast distributed algorithm for mining association rules*. Proceedings of the 4th International Conference on Parallel and Distributed Information Systems. Miami Beach, FL. 1996; 1: 31-44.

[5] Han J, Kambr M. Data mining: concepts and techniques, Morgan Kaufmann, 2001.

[6] Dean J, Ghemawat S. *MapReduce: Simplified data processing on large clusters.* Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation. San Francisco, CA. 2004; 1: 10.

[7] Chen Z, Xu Y, Wang XJ, Jin YL. A new fault tolerance system for cloud storage. *Journal of Convergence Information Technology*. 2011; 6(4): 34-4.

[8] Gu HY. Large-alphabet-oriented scheme for Chinese and English text compression.*Software- Practice and Experience (SPE)*. 2005; 35: 1027-1039.

[9] Glen A. Characterization of fine words over a finite alphabet. *Theoretical Computer Science.* 2008; 391: 51-60.

[10] Zhao Y, Shi JL, Shi PF. A Limited Lattice Structure for Incremental Association Mining. 6th Pacific Rim International Conference on Artificial Intelligence Melbourne. Australia. 2000; 1: 102-103.