2009 International Conference on Parallel and Distributed Computing, Applications and Technologies

An Efficient Hierarchical Clustering Method for Large Datasets with Map-Reduce

Tianyang Sun*, Chengchun Shu†, Feng Li†, Haiyan Yu† and Lili Ma* Yitong Fang‡
*Graduate University of Chinese Academy of Sciences
Beijing, 100390, P.R. China
†Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100190, P.R. China
‡Beijing No. 35 Middle School, Beijing, 100032, P.R. China
{suntianyang, shuchengchun, malili}@software.ict.ac.cn {lifeng, yuhaiyan}@ict.ac.cn
‡ fangyitong@beihua.edu.cn

***Abstract*—Large datasets become common in applications like Internet services, genomic sequence analysis and astronomical telescope. The demanding requirements of memory and computation power force data mining algorithms to be parallelized in order to efficiently deal with the large datasets. This paper introduces our experience of grouping internet users by mining a huge volume of web access log of up to 100 gigabytes. The application is realized using hierarchical clustering algorithms with Map-Reduce, a parallel processing framework over clusters. However, the immediate implementation of the algorithms suffers from efficiency problem for both inadequate memory and higher execution time. This paper present an efficient hierarchical clustering method of mining large datasets with Map-Reduce. The method includes two optimization techniques: Batch Updating to reduce the computational time and communication costs among cluster nodes, and Co-occurrence based feature selection to decrease the dimension of feature vectors and eliminate noise features. The empirical study shows the first technique can significantly reduce the IO and distributed communication overhead, reducing the total execution time to nearly 1/15. Experimentally, the second technique efficiently simplifies the features while obtains improved accuracy of hierarchical clustering.**

***Keywords*-Hierarchical clustering, Batch Updating, feature selection**

I. Introduction

Internet services such as e-commerce sites generate a large volume of web logs every day. The logs from millions of users provide a potential golden mine for understanding user access pattern and promoting new service value. But the large datasets also pose new challenges for data mining algorithms to efficiently process with in given constraints such as memory and execution time. To overcome the constraints, data mining algorithms can be implemented with Map-Reduce[1], which breaks the large datasets into small chunks and process them in parallel on multiple cluster nodes and scales easily to mine hundreds of terabytes data[2] by adding inexpensive commodity computers.

This work is supported by National High Tech. Development 863 Program(Grant No. 2008AA01Z140) and National NSF of China(GN 60873243)

This paper explores the efficiency implementation of hierarchical clustering [3] with Map-Reduce, in the context of grouping Internet users based on web logs. In the application, the Internet users are grouped based on the keywords in title and meta information of web pages that users have accessed to. As a general framework, Map-Reduce provides good scalability but ignores the efficiency optimization of data mining algorithms. As a consequence, it takes nearly 80 hours to finish the mining job for the web logs of 1.2 terabytes involving nearly 1,000,000 users and 100,000 keywords. In contrast, the proposed improved hierarchical clustering method can accomplished the same task in nearly 6 hours.

The low efficiency stems from two major aspects. Hierarchical clustering for large datasets consists of a large number (at the order of 105) of successive iterations of clustering processes, in which feature matrix merging and updating, similarity value modification are common operations. The matrix updating and similarity value modification invoke file operations of distributed file system and constant IO operations, which incur high IO and distributed node communication overhead. In addition, the large dimension of feature vectors demands high memory usages and causes heavy memory paging penalty.

In this paper, we present an efficient hierarchical clustering method for large datasets with Map-Reduce. We propose two optimization techniques to address the efficiency issues, which are believed to be applicable to other data mining algorithms implemented with Map-Reduce. We propose a Co-occurrence based feature selection technique at the preprocessing stage that takes the co-occurrence frequency as a contribution to the element value of feature vectors and makes, for each user, top N keywords with highest value as the selected features. Experiment shows the feature selection reduces the dimension of feature vectors to 49%, but with better accuracy of similarity values for user groups.

We approach Batch Updating to reduce the IO overhead and distributed communication cost during the clustering. The technique batches as many IO and communications operations as possible in one iteration, while assures the

978-0-7695-3914-0/09 $26.00 © 2009 IEEE
DOI 10.1109/PDCAT.2009.46

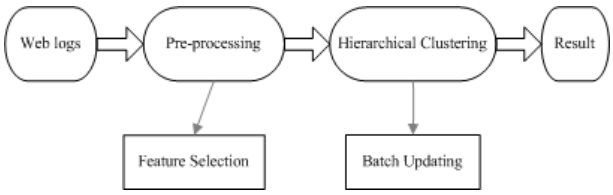

Figure 1. Phases of grouping Internet users based on Web access logs

correctness of hierarchical clustering. We propose the in-memory batch updating principle that treats the top N pairs of user groups with highest similarity values in three different ways. The empirical study shows Batch Updating reduces the number of updating iterations to 6.01%, the clustering is accomplished in 7.45% of the original execution time.

The rest of the paper is organized as follows. Section 2 provides an overview of the mining application as the background of hierarchical clustering with Map-reduce and motivates the optimizations. Section 3 presents the co-occurrence based feature selection in details, followed in Section 4 by Batch Updating and its implementation. Section 5 presents the performance evaluation of the efficiency of clustering with the two optimization techniques. The related work is described in section 6. Section 7 concludes the paper.

II. Overview

In this section, we present an overview of our mining application. We group Internet users based on web access logs, taking the keywords of web pages the users have accessed to as features. The whole application consists of two major phases: pre-processing of the raw data (web access logs) and hierarchical clustering of user groups. Figure 1 illustrates the whole processes from web logs to final clustering results.

Firstly, the pre-processing stage prunes web access logs, extracts web page topics and generates user-keyword matrix. The pruning involves the elimination of less useful access records like css and pictures. Then we use the words in the title and keyword meta information in the header of the web pages as the keywords due to the words from these positions of a web page are generally believed to be representative of the web page topic. The keywords from a Internet user's web pages are collected and are used to build user-keyword matrix. The matrix has users as rows and keywords as columns, where the element value dij is the counts (or normalized counts) of the keyword keyj in user ui's web accesses, which reflects the interest degree of the users for the keywords during the web accesses. Our mining application pre-processes the raw data and outputs a matrix of near 1,000,000 users and 100,000 keywords. The large datasets take a long time to pre-process, and operations on the generated matrix can not be loaded into memory for efficient processing, and results in high IO and distributed communication overhead. In section 3, we propose a co-occurrence based feature selection technique to decrease the dimension of the feature vectors so that reduces the IO cost and memory requirement for hierarchical cluster algorithms.

The second phase groups Internet users based on the user-keyword matrix using hierarchical clustering algorithms. Hierarchical clustering algorithms create a hierarchy of user groups that naturally reflect the containment relationship of the user interested topics. The clustering is parallelized with Map-Reduce and performed in bottom-up fashion that assigns each user as a trivial user group (cluster), and then successfully merges pairs of user groups (clusters) until the defined termination condition is met. One major problem of this phase is heavy IO overhead incurred by the updating of user-keyword matrix and the modification of similarity values in every clustering iteration, because every change of user-keyword matrix or similarity values includes I/O operations and distributed communication among cluster nodes. Section 4 of this paper proposes Batch Updating to combine several iterations of user group merging and updating into one, which empirically decreases the response time of our application a lot.

III. Co-occurrence based Feature Selection

One method to improve efficiency of the hierarchical clustering is to reduce the dimension of user-keyword matrix. In this paper we propose co-occurrence based feature selection, motivated by the observations as follows. Firstly, the list of keywords extracted from the title and the keyword meta information are not equally important in weight and even some of them are noisy keywords. Secondly, the summary of a web page can be described by several keywords that are semantically related. For example, a page with title "Apple-iMac The all-in-one desktop for your photos, movie and music" can be summarized with the subset of keywords "Apple iMac photos movie music". Thirdly, semantically related keywords are more representative for the web page topic than other keywords. For instance in last example, "apple", semantically related to "iMac", is more representative than "desktop", though they both appear in the title.

Co-occurrence based feature selection reduces the dimension of feature vectors by summarizing a user's interested topics with the most representative keywords. For any two keywords, their co-occurrence frequency are calculated to reflect the relationship in semantics between keywords, and higher element value d_{mj} (also called attention degree) of feature vectors is given to more related keywords than the others. Essentially the the element value d_{mj} consists of the contributions of both the number of keyword occurrences and the co-occurrence frequency of the pairs of keywords. The whole procedure of feature selection is performed in 5 phases, and is realized with 17 Map-Reduce tasks. The reduced user-keyword matrix is reused in a large number of

clustering iterations, making the overhead of co-occurrence based feature selection amortized.

A. Calculation of Attention degree for keywords

Phase 1: the calculation of attention degree starts with counting the keywords and the pairs of keywords in title and the keyword meta information of web pages a user has visited. For every page title, this phase outputs Result1 and Result2 in following format:

$$R_1 :< user_m, keyword_i, n_i >$$
$$R_2 :< user_m, keyword_i, keyword_j, n_{ij} >$$

R_1 describes that $keyword_i$ appears n_i times in the page accessed by $user_m$; R_2 describes that the pair ($keyword_i$ and $keyword_j$) co-occurs n_{ij} times visited by $user_m$.

From the keyword meta information, the result set Result3 in format R_3 are yield , describing that pair ($keyword_i$ and $keyword_j$) appeared n_{ij}^m times in meta information of url_s.

$$R_3 :< url_s, keyword_i, keyword_j, n_{ij}^m >$$

Phase 2: With Result1 and Result2, we calculate the attention degree of each keyword using (1) and (2) according to Jarrod frequency. As shown in (1), the attention degree d_{mi}^t of keyword contains semantic information with $cofreq_{ij}^t$. The output of this phase is Result4 in form of R_4:

$$R_4 :< user_m, keyword_i, d_{mi}^t >$$
$$d_{mi}^t = n_i \times (1 + \sum_{j \neq i} cofreq_{ij}^t) \quad (1)$$

$$cofreq_{ij}^t = n_{ij}/(n_j + n_j - n_{ij}) \quad (2)$$

Phase 3: Similarly, the co-occurrence frequency $cofreq_{ij}^m$ for every pair of keywords in the meta information is calculated by the formula (2). The phase outputs the result set Result5 in format R_5:

$$R_4 :< keyword_i, keyword_j, cofreq_{ij}^m >$$

Phase 4: with the contributions of keywords from both the title and the keyword meta information both directly and indirectly (the second term in (3)), the final attention degree $d_m^f i$ of $user_m$ for $keyword_i$ is calculated by (3) and yields the result set Result6 in format R_6:

$$d_{mi}^f = d_{mi}^t + \sum_{j \neq i} (d_{mj}^t \times (cofreq_{ij}^m)) \quad (3)$$

$$R_6 :< user_m, keyword_i, d_{mi}^f >$$

B. Feature selection

Phase 5 performs feature selection and builds the reduced user-keyword matrix of low-dimension feature vector. Given the feature vector for a Internet user, we reduce the dimension by only keeping the top N attention degrees that are most representative for the users' interested topics. The empirical optimal value of N is 100 as shown in Section 5.

The final feature vectors are calculated in the following steps: (1) Select the representative keywords for users. For each user, the accessed keywords are ranked in descending order of the final attention degrees, and the top N keywords are selected as representations of the user's interests. (2) Union the all users' representative keywords to form the final list of the selected features. (3) Obtain the feature vectors for the users based on the final list of selected features. Using result set Result6 as input, the element for keywords not in selected features are removed, and the $user_m$'s attention degree $d_m i$ for $keyword_i$ in the selected features is set to the final attention degree d_{mi}^f.

If the dimension of the feature vector with selected features is m, and the number of users is n, the user-keyword matrix is given in formula (4), where each row is a feature vector for a user and the element value is the attention degree of the user for a keyword. As shown in Section 5, co-occurrence based feature selection reduces the dimension of feature vectors to nearly a half, but with better accuracy of similarity values for user groups.

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nm} \end{bmatrix} \quad (4)$$

IV. Hierarchical Clustering with Batch Updating

Generally, the hierarchical clustering consists of a large number of successive clustering iterations. Every iteration begins with calculating the similarity of every pair of user groups (or users) and finding out the most similar user groups, then merges them and updates the feature vectors in user-keyword matrix file.

We use Pearson correlation coefficient[4] formula to calculate the similarity sim(i,j) of two user groups (or users) u_i and u_j using (5), where d_{im}, d_{jm}, $\bar{d_i}$, $\bar{d_j}$ and σ_i, σ_i are the m-th attention degrees, the expectation values and standard variations of feature vectors u_i and u_j respectively.

$$sim(i,j) = \frac{\sum_{m=1}^{n}(d_{im} - \overline{d_i})(d_{jm} - \overline{d_j})}{n\sigma_i\sigma_j} \quad (5)$$

For the merging process of some user group u_i and u_j in user-keyword matrix, we remove them first; then generate the merged user group $u_i^{'}$=($d_{i1}^{'}$,$d_{i2}^{'}$,$\cdots$,$d_{in}^{'}$) and insert it into user-keyword matrix. Each feature vector $d_{im}^{'}$ of $d_i^{'}$ is calculated as follows:

$$d_{im}^{'} = (d_{im} + d_{jm})/2, m = 1, 2, \cdots, n \quad (6)$$

The direct parallelization of hierarchical clustering has the problem of causing poor performance: a large number of successive clustering iterations will incur high IO and distributed communication overhead due to the read/write

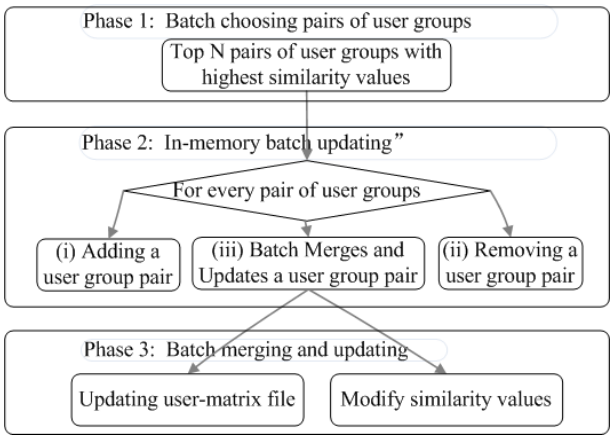

Figure 2. Hierarchical Clustering with Batch Updating

operations both on disks and distributed file system, caused by both matrix updating and similarity value modification.

This section details Batch Updating that is proposed to improve efficiency of hierarchical clustering for large datasets. The idea behind is to combine several iterations of clustering into one so that the updating matrix file and modifying similarity values are processed in batch mode, resulting in a shorter response time for the mining application.

Figure 2 illustrates three major phases of the hierarchical clustering with Batch Updating. Phase 1 batch chooses the top N pairs of user groups for the iteration of clustering; Phase 2 performs in-memory batch updating on the chosen user groups as many as possible; Phase 3 updates the user-keyword matrix file and similarity values, both being stored in distributed file system.

A. Data structures

The improved hierarchical clustering methods use two new data structures for efficient clustering.

(1) C-queue: a queue that stores top N pairs of user groups with highest similarity values. The element in the queue is C_{ij}, which denotes the pair contains user groups ui and uj. All elements in C-queue are sorted in descending order. Without loss of generality, we make the following assumption for the element C_{ij}: (i¡j).

(2) Batch-queue: a queue that stores C_{ij} (i¡j) to be batch processed in one iteration. In fact, the pairs in C-queue are processed in queue order one by one and are inserted into Batch-queue if necessary. Therefore, elements in Batch-queue are also sorted in order.

B. In-memory batch updating Principle

The principle behind in-memory batch updating is to batch process as many iterations of user group clustering as possible, so long as no wrong clustered user groups are generated. To guarantee the correctness of the clustering, an element in C-queue is processed in three different ways: (a) insert the element into Batch-queue; or (b) ignore the element by Batch-queue; or (c) end the iteration of batch updating by merging and updating of the pairs in Batch-queue.

Assume Batch-queue contains m elements, three different ways of processing the element C_{ij} in C-queue and the rationale behind are described as follows.

(a) Insert Cij into Batch-queue. The process takes place when no elements exist in Batch-queue that have user groups equal to the user group u_i or u_j that C_{ij} denotes, as formally defined as follows:

$$\forall C_{pq} \in Batch - queue, p, q \neq i \wedge p, q \neq j \qquad (7)$$

The reason for this is that, the processing including user-keyword matrix and similarity value updating for the elements in Batch-queue has no side-effects on the processing of the element C_{ij}. As a result, the element is directly inserted into Batch-queue.

(b) Ignoring C_{ij}. As long as there exists an element C_{pq} in Batch-queue that u_q is the same as one of the user groups denoting by C_{ij}, C_{ij} can be removed from C-queue and ignored by Batch-queue. The condition is formally given by formula (8).

$$\exists C_{pq} \in Batch - queue, q = i \vee q = j \qquad (8)$$

Hierarchical clustering contains the process of merging user groups, which means, after processing the element C_{pq}(p<q), the user groups u_p and u_q are merged. In our merge strategy, we only keep the user group u_p and replace its feature vector with updated element values using (6), and the user group u_q disappears. As the element C_{ij} meets (8), we just remove it from C-queue and start processing the next element.

(c) Merge and update the pairs in Batch-queue. It is taken when there exists an element C_{pq} in Batch-queue that the user group u_p is same as u_i or u_j. Formally the condition is given as follows:

$$\exists C_{pq} \in Batch - queue, p = i \vee p = j \qquad (9)$$

The reason is that, C_{ij} is at the head of C-queue in advance. If we update the feature vector of user and modify the similarity values, the similarity values of the element C_{ij} and other elements in C-queue may be changed as well as the order of elements in C-queue, which means C_{ij} is probably not at the head of C-queue. Therefore, we need to batch merge the clusters, update user-keyword matrix and modify similarity values by the elements in Batch-queue, then start a new batch updating, based on the up-to-date similarity values.

C. Implementation with Map-Reduce

Our hierarchical clustering using 3 Map-Reduce tasks for batch choosing pairs of user groups (phase 1), updating the user-keyword matrix file (phase3) and modifying the

Table I
THE CHANGES OF KEYWORD (FEATURE) NUMBERS WITH OR WITHOUT FEATURE SELECTION FOR THE GIVEN LARGE DATASETS. EACH USER'S TOP 100 KEYWORDS ARE SELECTED

	# of keywords
With feature selection	53,049
Without feature selection	108,489

similarity value file (phase 3) respectively. The in-memory batch updating (phase 2) involves processing elements in C-queue as well as filling them into Batch-Queue, both of which are directly performed at the master node.

V. EXPERIMENTS

We perform all experiments on 24 cluster nodes, which are Dell PowerEdge 2950 servers with two dual-core processors (Intel Xeon 2.66GHz), 4GB RAM, and installed with Linux RHEL4. The nodes are connected by Gigabit Ethernet cable. The large datasets are the 1.2 TB web logs collected in six months, which contains more than 1,000,900 users, and the accessed keywords number 108,489.

A. Feature selection evaluation

Co-occurrence based feature selection uses a subset of keywords as features to reduce the dimensions of both feature vectors and user-keyword matrix. We build the user-keyword matrix with and without feature selection. Table I lists the changes of keyword numbers, where N=100 top keywords are selected for each user. The dimension of the matrix and the feature vectors is reduced to 49%, from 108,489 to 53,049. As most feature vectors are sparse, memory requirement is decreased and similarity calculation is more efficient.

Feature selection changes the attention degrees of the keywords accessed by users. We measure the changes of top N keywords for the users using metrics given as (10) and (11), which measures the changes of top interested keywords for a single user ui and the global average changes for all the users respectively.

$$change_i = \sum_{j=1}^{N} m_{ij}/N \times 100\% \tag{10}$$

$$aver_change = \sum_{i=1}^{M} change_i/M \times 100\% \tag{11}$$

Where mij is an indicator variable for the changes. m_{ij}=1 if $keyword_j$, one of the user u_i selected features, is not one of the original top N keywords without feature selection, otherwise m_{ij}=0. Figure 3 shows the average change ratio with the different choice of parameter N. The change ratios of top keywords decrease as number of selected keywords for each user grows. We choose N=100 for feature selection as it results in low average change percentage (0.92%).

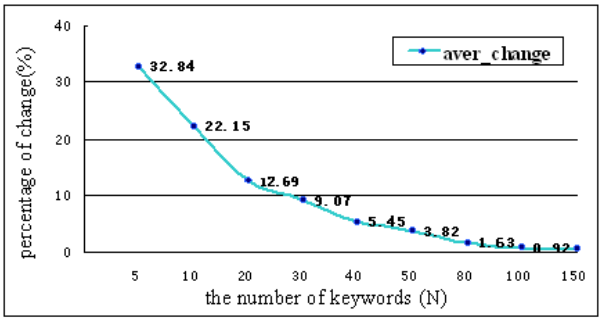

Figure 3. Percentage of Change of Top N Keywords

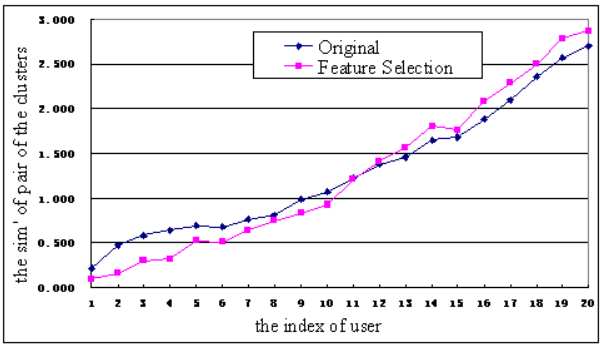

Figure 4. Comparison of similarity values with and without feature selection

Secondly, we evaluate the impact of the feature selection for the hierarchical results. Here we measure the changes indirectly by inspecting the changes of similarity values between user groups. For a specific user group u_i, we calculate twice, with and without feature selection, the similarity values between the user group and other randomly selected 20 user groups. Figure 4 illustrates the comparison results(ordered by the similarity values). It shows feature selection increases the similarity values for the pairs of similar user groups (with log(sim$\times$1000)$>$1.224 of the 11-th pair), but decreases the values for the dissimilar user groups, performing a better accuracy of hierarchical clustering.

B. Batch Updating evaluation

We first evaluate the efficiency of hierarchical clustering with Batch Updating under different size N of C-queue. The size N affects how many pairs of user groups are batch choosed and results in changing the total execution time. Table II shows the numbers of iterations and the total execution times with the different N, which implies the iteration numbers are approximately equal(they are affected by the condition that some critical values are equal) and the bigger N does not necessarily mean more efficient clustering.

Then we evaluate the efficiency improvement of hierarchical clustering with Batch Updating. We choose the size

Table II
NUMBERS OF ITERATIONS OF BATCH UPDATING AND EXECUTION TIMES WITH DIFFERENT SIZE N OF C-QUEUE

N	# of iter.	Execution time(seconds)
500	1678	46962
100	1680	28603
50	1680	21412
10	1684	26615

Table III
NUMBERS OF UPDATES AND EXECUTION TIMES OF HIERARCHICAL CLUSTERING WITH AND WITHOUT BATCH UPDATING

	# of iter.	Execution time(seconds)
With BU	1680	21412
Without BU	27939	287258

of C-queue as N=50, and table III shows the comparison results of hierarchical clustering for the large datasets. The results show that Batch Updating decreases the total execution time of our mining application to 7.45%. The major efficiency contribution is the reduced IO operations and communication costs, incurred by the only 6.02% iterations of hierarchical clustering with Batch Updating.

VI. RELATED WORK

The common methods of feature selection are Mutual Information [5,7], Chi-test[5,6] and frequency based. Co-occurrence based feature selection in this paper selects a subset of most representative keywords for each user according to the frequency. The frequency based feature selection is not applicable for unsupervised learning because it needs the manual labeling, whereas the co-occurrence frequency is calculated using Jarrod formula, which mines the semantic relationship between keywords without human involvement.

BIRCH[8] and CLARAN[9] are two clustering methods proposed for large datasets. BIRCH saves the memory usage and minimizes the IO cost by concentrating on dense regions and using a compact summary CF tree for clusters. CLARAN is a algorithm for a large set of spatial data, based on randomly search. These methods target at clustering large datasets on a single node, while our clustering deals with large datasets in distributed manner using the computation and storage capability of large cluster nodes. Goals of our techniques are efficiency and scalability.

A framework of co-clustering large datasets using technique of matrix decomposition is proposed in [10]. Co-clustering simultaneously clusters both dimensions (row and column) of a contingency table. The technique for efficiently clustering large datasets is the matrix decomposition. Co-clustering gives an illustrative example with a dataset containing 2000 rows and 240 columns, which is in much smaller scale than the real-world large datasets in this paper. Instead, DisCo[11] implements a distributed co-clustering with Map-Reduce both in pre-processing and clustering phases, and demonstrates the performance scalability on mining real-word large datasets, though it don't address the efficiency issues.

VII. CONCLUSION

This paper explores the efficient implementation of hierarchical clustering algorithms with Map-Reduce, in the context of grouping Internet users based on web logs. To reduce the memory requirement for the large size of features, Co-occurrence based feature selection technique is proposed. It adds the co-occurrence frequency to attention degree of features and takes the top N keywords with highest attention degrees as selected features for each user. Experiments shows the dimension of the feature vectors is reduced to 49% with better accuracy of similarity values of highly similar user groups. To lower the IO and distributed communication overhead, we propose Batch Updating to combine as many user groups merging and updating as possible in one update iteration. Experimentally, the technique decreases the iteration of clustering to nearly 1/15, and so does the total clustering execution time.

REFERENCES

[1] J. Dean and S. Ghemawat. *MapReduce: Simpilifed Data Processing on Large Clusters*, In Communications of ACM, 51(1), 107-113,2008.

[2] Yahoo! Launches World's Largest Hadoop Production Application, http://tinyurl.com/2hgzv7.

[3] A.Jain,M.Murty, and P.Flynn. Data Clustering: A Review. In ACM Computing Surveys, 31(3), 1999.

[4] J.L.Rodgers,and W.A.Nicewander.Thirteen ways to look at the correlation coefficient.The American Statistician 42, 59-66, 1988.

[5] D. Lewis, and R. Marc. A comparison of two learning algorithms for text categorization.In SADIR,1994, pp. 81-93.

[6] G. Xiubo, T. Liu, T. Qin, and H. Li. Feature selection for ranking. In Proc. SIGIR. 2007, Pp. 407-414.

[7] Schutze1995] H. Schutze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for routing problem. In proc. SIGIR, 1995, pp. 229-237.

[8] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Effecicent Data Clustering Method for Very Large Databases. In Proc. SIGMOD, 1996, pp. 103-114.

[9] R. T. Ng, and J. Han. Efficient and Effective Clustering Method for spatial Data Mining. In proc. Of VLDB, 1994.

[10] F. Pan, X. Zhang, W. Wang. A General Framework for Fast Co-clustering on Large Datasets Using Matrix Decomposition. In ICDE, 2008.

[11] S.Papadimitriou, J.Sun. DisCo: Distributed Co-clustering with Map-Reduce A Case Study Towards Petabyte-Scale End-to-End Mining. In ICDM,08, 512-521, 2008.