

# DISTRIBUTED HIERARCHICAL DOCUMENT CLUSTERING

Debzani Deb, M. Muztaba Fuad and Rafal A. Angryk<sup>†</sup>  
Department of Computer Science  
Montana State University, Bozeman, MT 59717, USA  
{debzani,fuad,angryk}@cs.montana.edu

## ABSTRACT

This paper investigates the applicability of distributed clustering technique, called RACHET [1], to organize large sets of distributed text data. Although the authors of RACHET claim that the algorithm generates quality clusters for massive and high dimensional data set, the algorithm was not yet evaluated on a well known academic data set. This paper presents performance analysis of the algorithm and tests its suitability for distributed document clustering. This work uses three widely known hierarchical algorithms to generate local clusters at each of distributed repositories and then the RACHET is applied to merge distributed hierarchies of clusters. We perform our own tests of the algorithm on standard document corpora [2], using popular cluster evaluation measures [3, 4] and discuss important implementation details.

## KEYWORDS

Distributed hierarchical clustering, document clustering.

## 1. Introduction

As the amount of text information available online in the form of electronic publications, digital libraries, web pages etc. is increasing rapidly, the need to automatically organizing this information is gaining more and more importance. Document clustering is an automatic grouping of text documents into clusters where documents of the same cluster are more similar than the documents in different clusters. Recently, hierarchical document clustering is being used in document browsing to organize results returned by search engines. There are numerous clustering algorithms available in the literature, however, most of them face a major challenge in case of clustering text documents mainly because of 1) very large dataset size and 2) high dimensionality of documents. Moreover, nowadays, the massive sets of documents are inherently distributed among various sites connected by the Internet. When a user wants to build a hierarchical organization of documents spread out on multiple servers of her company, current centralized solutions require bringing all the documents (full texts or their vector representations) to a central warehouse before running a clustering algorithm. It generates additional costs due to 1) the communication cost and latency for sending large number of documents to

the central site; 2) the extra storage space and computing resources that are needed at the central site. For a very large number of documents, these costs are usually extreme and therefore, a centralized approach to cluster massive sets of distributed documents becomes not feasible.

By applying the distributed version of the clustering algorithms, it is possible to cluster documents in the same location where they reside and then aggregate those locally generated clusters into a global clustering of all distributed documents with additional communication cost which is far lower than the cost associated with sending all documents in case of a centralized approach. Since distributed document clustering that uses the *k means* [3, 5, 6] approach to produce flat clustering of documents has a disadvantage of supplying the number of clusters  $k$  as input, the hierarchical decomposition of distributed text documents is of interest in most cases.

There are several important aspects of RACHET that makes it a feasible choice for our experimentation in case of clustering distributed documents. The most important reason is its claimed  $O(n)$  time, space and communication complexity [1], where  $n$  is the number of text documents. The above  $O(n)$  time complexity excludes the time to generate local clusters in each site. As we are dealing with massive high dimensional data set, this linear complexity makes the algorithm very attractive for our future research purposes. The other important reason is its suitability in case of naturally and inherently distributed data set. RACHET assumes horizontal data distribution i.e. each site has the same set of features but different set of data points, which is very useful for our basic interests concentrated on finding hierarchical organization of distributed documents returned by a search result as in that case it is highly likely that these documents share many of the same words over all sites. RACHET does not make any assumption about the number of clusters generated in each site which is crucial in our research as the number of local clusters may vary depending on the existing documents on a site.

While the original approach [1] seems reasonable, the RACHET algorithm was evaluated only on small data sets, what does not provide sufficient examination for scalability issue. Moreover, instead of evaluating the algorithm using widely used cluster quality evaluation metrics such as *entropy* [4] or *F score* [4, 7], the authors applied rarely

<sup>†</sup> Partially supported by NASA Grant Consortium, Award No. M166-05-Z3184.

used evaluation standards and did not provide careful analysis of performance issues. To generate the local clusters, they use a centroid-based hierarchical clustering algorithm, where there are numerous hierarchical approaches available and the performance is reported to vary significantly among them. Furthermore, none of the techniques used in RACHET was ever applied to document clustering. This paper resolves the vagueness about the performance of the algorithm and examines its suitability in case of document clustering. The authors [1] also did not provide a detailed description of the implementation in RACHET. Our paper provides a detailed discussion of the implementation issues related to distribution and coordination along with some enhancements we decided to implement.

The rest of the paper is organized as follows: section 2 discusses literature related to distributed clustering, section 3 describes various aspects of RACHET, section 4 discusses the distribution and implementation issues, section 5 presents and discusses the results and section 6 concludes the paper.

## 2. Related literature

Dhillon and Modha [5] developed a parallel implementation of the K-means clustering algorithm on distributed memory multiprocessors. A data set of size  $n$  is divided into  $P$  blocks of roughly equal size. During a K-means iteration, each distributed site updates the current  $K$  centroids based on the local data and then broadcast their centroids. After receiving all the centroids from other sites, a site can form the global centroids by averaging. The approach used in this study is easy to implement and comprehend. However, the implementation only generates flat clustering of data set and requires supplying the value of  $K$  as parameter.

Forman and Zhang [6] took an approach similar to above [5], but extended it to K-harmonic means. Eisenhardt et al. [3] extended K-means with a ‘probe and echo’ mechanism for updating cluster centroids. Along with the problems associated with K-means clustering mentioned above, this approach generates lots of message traffic in the network because of its peer to peer communication model.

In their study, Lazarevic *et al.* [8] combined local clusterings in order to generate a global regression-based classifier. However, the assumption made in this work is that each site should generate same number of clusters.

Johnson and Kargupta [9] also merged local clusters generated by hierarchical clustering algorithm into a global one. Similarly to RACHET, they also used statistical bounds to represent clusters efficiently across sites. However, this algorithm is limited to the case of vertically partitioned data (features are different in different distributed sites) what conflicts with our interests.

## 3. Discussion of RACHET approach

RACHET [1] is a distributed clustering technique that is claimed to be suitable for very large, high dimensional and horizontally distributed data. In RACHET, a local dendrogram is generated from each of the distributed data sites using a hierarchical algorithm. RACHET also generates descriptive statistics that approximately characterize each of the generated clusters in the dendrogram. To reduce communication cost, this approximate representation, which is significantly smaller than the original representation of the clusters, is then transmitted to a merger site. Local dendrograms are then merged to a global dendrogram that gives us the required hierarchical organization of the distributed documents.

To build a distributed clustering algorithm, RACHET [1] assumes  $S$  distributed sites available in the system and  $n$  data objects with  $d$  dimensions are horizontally distributed among those sites. The merger site has a set of locally generated dendrograms  $D = \{D_1, D_2, \dots, D_{|S|}\}$  where each node of those dendrograms represents a cluster. The objective is to find a global dendrogram using the local dendrograms from  $|S|$  sites such that the quality of the distributed approach is comparable to that of a centralized approach.

In order to produce efficient clustering, there is a need to represent local clusters in a compact form than the naive representation (sending all data points of the cluster), that also provides valuable information about the documents in the cluster. RACHET uses *covering hypersphere* ( $\bar{c}, R_c$ ) scheme to represent a cluster. It is defined as a hypersphere centered at  $\bar{c}$  and with radius  $R_c$ , such that the cluster centroid  $\bar{c} = (f_{c1}, f_{c2}, \dots, f_{cd})$  is the mean vector of all the document vectors in cluster  $C$ . The radius of a cluster  $R_c$  is defined as the average squared Euclidean distance of document vector from the centroid of the cluster.

As centroid itself is a high dimensional vector (specially when  $d$  is very large), sending the centroids of every node (cluster) of all dendrograms generated from all sites can increase the communication cost to  $O(nd)$ , which is not desirable in order to maintain the efficiency and scalability of the algorithm. Therefore, RACHET computes Descriptive Statistics (DS) of the cluster centroids that minimizes the representation further into a 6-tuple  $DS(\bar{c}) = (N_c, R_c, SUM_c, MIN_c, MAX_c, SQSUM_c)$ .

After receiving locally generated dendrograms along with the descriptive statistics from  $S$  sites, the merger site computes the global dendrogram by applying the following algorithm, where  $D[]$  is an array containing local dendrograms .

### Dendrogram global\_dendrogram(Dendrogram D[])

1. Calculate a distance matrix that contains the distance between all pairs of dendrograms and whose  $ij^{th}$  entry

gives the Euclidean distance between the centroids of the root node of dendrograms  $D_i$  and  $D_j$ , where  $i, j \leq |S|$ .

2. Repeat the following steps for  $|S| - 1$  times
  - a. Find the closest dendrograms  $D_1$  and  $D_2$  from the distance matrix.
  - b.  $D_3 = \text{merge\_dendrogram}(D_1, D_2)$
  - c. Update distance matrix so that it now include  $D_3$  and calculates the distance between  $D_3$  and other dendrograms (except those used to merge  $D_3$ )
3. Return Global dendrogram.

Step 1 in *global\_dendrogram()* algorithm involves computing the Euclidean distance between centroids of the root of two dendrograms such as  $d(\vec{c}_1, \vec{c}_2)$ . However, since only the descriptive statistics about the centroids are available at the merger site, the approximate distance  $d_{app}(\vec{c}_1, \vec{c}_2)$  using  $DS(\vec{c}_1)$  and  $DS(\vec{c}_2)$  is computed instead.

The most critical component of RACHET is *merge\_dendrogram()* method that merges two dendrograms  $D_1$  and  $D_2$  into a new dendrogram.

### Merge\_dendrogram( $D_1, D_2$ )

1. If clusters  $C_1$  and  $C_2$  represented by the root node of dendrograms  $D_1$  and  $D_2$  are well separated, then create a new node  $D$  and assigns  $D_1$  and  $D_2$  as its leaves (Figure 1(b)).
2. If  $C_1$  is contained within  $C_2$ , then following checks need to be performed accordingly.
  - a. If  $C_1$  does not intersect with any other child clusters  $C_{21}, C_{22}, \dots$  of  $C_2$ , then make  $D_1$  a child of  $D_2$  (Figure 1(c)).
  - b. Else, find the best child of  $D_2$  such that the amount of intersection with  $C_1$  is the maximum considering all children.
    - i.  $\text{new\_child} = \text{merge\_dendrogram}(D_1, \text{best\_child})$
    - ii. Add this  $\text{new\_child}$  in place of the  $\text{best\_child}$  in  $D_2$ .
    - iii. Generate descriptive statistics of  $\text{new\_child}$  from statistics of  $\text{best\_child}$  in  $D_2$  and  $D_1$  itself. (Figure 1(d))
  - c. Set  $D = D_2$
3. If  $C_2$  is contained within  $C_1$ , same steps are followed as in the above case while altering  $D_1$  and  $D_2$  and later setting  $D = D_1$ .
4. If  $C_1$  and  $C_2$  intersects, then following steps need to be performed
  - a. Create a new node  $D$  and assign  $D_1$  and  $D_2$  as its children.
  - b. Find the intersecting children of  $D_1$  and  $D_2$ .
  - c. Delete the intersecting children from  $D_1$  and update the descriptive statistics of  $D_1$  while excluding the intersecting children.
  - d. Perform the above step for  $D_2$ .

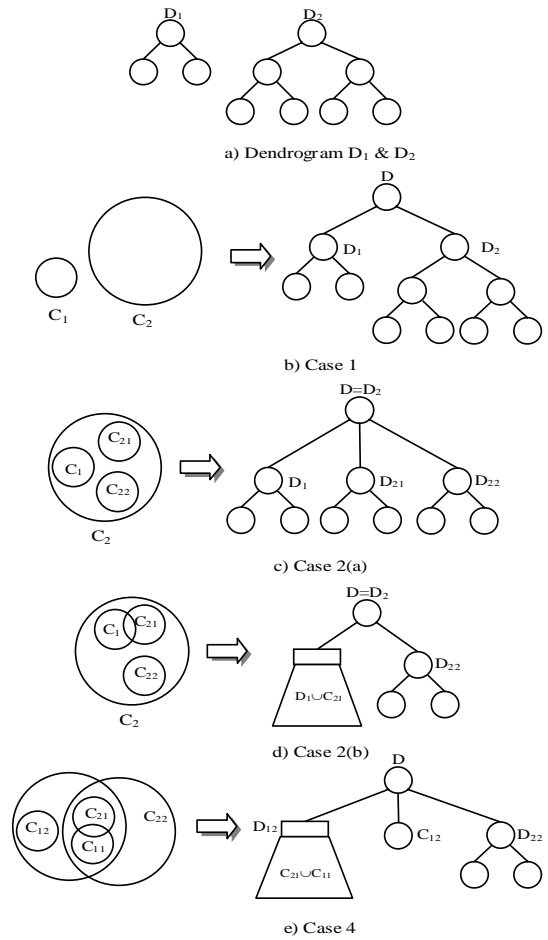


Figure 1. *merge\_dendrogram()* algorithm

- e.  $\text{new\_child} = \text{global\_dendrogram}(\text{Children}[])$ , where  $\text{Children}[]$  is an array containing the intersecting children.
- f. Add  $\text{new\_child}$  as a child of  $D$  (Figure 1(e)).
5. Compute centroid and radius of the root node of  $D$  using descriptive statistics of the children and return  $D$  as the merged dendrogram.

It should be noted that the function *merge\_dendrogram()* employs recursion in its implementation. The recursion stops when step 1, 2(a) or 3(a) occurs or a leaf node is reached. For every non-leaf node, *merge\_dendrogram()* descends a level in input dendrogram and calls either itself or *global\_dendrogram()* and proceeds recursively.

## 4. Implementation details

To make our evaluation results comparable to other published results in text mining, we choose 20 newsgroups collection [2] as our evaluation corpora. This corpus is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups corresponds to a different topic.

In RACHET, the authors claim that the algorithm performs best when all data points of the same cluster resides in the same site and the worst case scenario

happens when each data site contains a subset of data points from each cluster. When justifying the applicability of RACHET, we choose their worst case scenario as our test case and we randomly and evenly distribute documents of each class among  $S$  sites. All messages in 20 newsgroups collection are given a unique identification number so that they can be identified unambiguously across the distributed environment.

In our study, the local dendrograms are generated at each site by using three most commonly used agglomerative hierarchical clustering schemes [10, 11, 12] such as single-link, complete-link and average-link. We then prune the dendrograms at a particular level that maximizes the inter-similarities among the clusters produced at that level. For each leaf node of the pruned dendrogram, descriptive statistics are then generated. The descriptive statistics of the non-leaf nodes are generated from the statistics generated at leaf nodes. After that, we obtain a list of dendrograms along with descriptive statistics that capture the major information about the clusters in the dendrogram. We then transmit those dendrograms to a single merger site, where the agglomeration is performed and a global dendrogram is built.

In our distributed model, built with Java RMI, each node is designed to act as a remote server. Once each remote server builds local dendrograms based on its local corpus, it will send that to the central client. Initially, the client is given a list of machines where the remote corpuses are and where the remote servers are waiting for client invocation. Client then initiates a series of remote calls to those servers and starts the distributed clustering process. This way of initiating is well suited for evaluating our algorithm with different corpuses with varying properties. For example, to execute the clustering process on a particular corpus with certain algorithm, only the client need to be instructed about the information and it can then convey that information to the servers, causing the servers to work accordingly.

Once the local dendrograms are created and pruned at distributed sites, those dendrograms need to be transmitted to a merger site. The way Java RMI manages the communication is that, it recursively serializes all objects that are being passed between client and server. As the dendrograms are represented as a tree and contains pointer to left and right nodes, it creates stack overflow problem when Java RMI tries to serialize it, because each node is serializable and the tree itself is serializable. As a result, during serialization, the Java Virtual Machine (JVM) runs out of stack space. To overcome this problem, we need to encode the tree along with the descriptive statistics attached to each node in some way so that we can send it without using the pointers but still preserving the left node, right node information. On the merger site, we need to decode those encoded information again to get back the original tree structure. An encoding scheme that encodes

the tree to a string was devised for this purpose. This algorithm not only saves time (as serialization is slow compared to our encoding/decoding) but also uses less communication bandwidth as now we have full control on what we are sending and the information is represented as sequence of characters, rather than an object hierarchy, as done by serialization. After receiving the encoded string, the client then decodes the string into a dendrogram and reassigns the corresponding statistical information to the nodes of the dendrogram.

All remote clustering method calls are made multi-threaded. Otherwise, one node needs to wait for others to finish as all RMI calls are blocking. We use asynchronous method calls by changing the remote object to a Java thread. This greatly improves the execution time. However, as calls are now asynchronous, we need to implement a synchronous barrier on the client side to make sure that the merging of dendrograms at client node begins only after each of the remote sites finish transmitting its local dendrogram.

One of the important step in *merge\_dendrogram()* is to find the cluster that matches best with a particular cluster C when multiple clusters exist that have intersection with C. However, this significant step is not elaborated in RACHET, and thus we adopted an algorithm to find the best intersection cluster for C. We first find the intersection of the hyperspheres described by  $(\vec{c}_1, R_{c_1})$  and  $(\vec{c}_2, R_{c_2})$  using the following formula [13]:

$$\begin{aligned} \text{Circle Intersection} = & R_{c_1}^2 \cos^{-1} \left( \frac{d^2 + R_{c_1}^2 - R_{c_2}^2}{2dR_{c_1}} \right) + R_{c_2}^2 \cos^{-1} \left( \frac{d^2 + R_{c_2}^2 - R_{c_1}^2}{2dR_{c_2}} \right) \\ & - \frac{1}{2} \sqrt{(-d + R_{c_1} + R_{c_2})(d + R_{c_1} - R_{c_2})(d - R_{c_1} + R_{c_2})(d + R_{c_1} + R_{c_2})} \end{aligned}$$

where  $d$  is the Euclidean distance between centroids of two clusters. Since, we can not calculate  $d(\vec{c}_1, \vec{c}_2)$ , we use  $d_{app}(\vec{c}_1, \vec{c}_2)$  instead. After finding the amount of intersections between C and other clusters, we then compute what percentage of the area of C is occupied by the intersection. Finally, we find the cluster that has the largest intersection with C found by the previous step. The reason we convert the intersection volume to a percentage of area of C is that, it normalizes intersection volumes with other clusters and then chose the largest one.

## 5. Results of tests and discussion

### 5.1 Testing environment

We execute the distributed clustering program on four machines connected to the campus LAN. Two of the machines are Pentium IV, 2.0+ GHz with 512 MB RAM and two other are older Pentium III, 900 MHz with 128 MB RAM. Unlike RACHET, our execution time measurement includes the time spent on different sites

along with the time spent to merge the remote clusters into a global cluster. We use Java’s standard time measurement function to measure all the time and we average them over different runs.

### 5.2 Cluster quality

Figure 2 shows the *F score* measures of the distributed version in case of three clustering algorithms. If we compare the performance with the centralized version (Figure 3), then we observe at most 30% performance degradation in case of distributed clustering. Depending on the purpose of distributed clustering, such behavior may be treated as acceptable taking under consideration the fact that only approximate information (i.e. Descriptive Statistics and hierarchies themselves) is used to represent clusters in merger site. This “sacrifice” must affect the quality of global clusters.

From figure 2 and 3, we can observe that the quality of cluster decreases with the increase in number of classes of documents being considered. We believe that, this particular behavior is solely caused by the characteristics of the 20 Newsgroup data set. In case of the particular data set involving messages from 5 classes, the classes are well separated and contain messages that discuss very different topics and therefore results in better clusters. On the other hand, in case of 20 class data set, the classes share many of the similar words because of the Newsgroup’s common subject matter and this is the reason behind the poor performance.

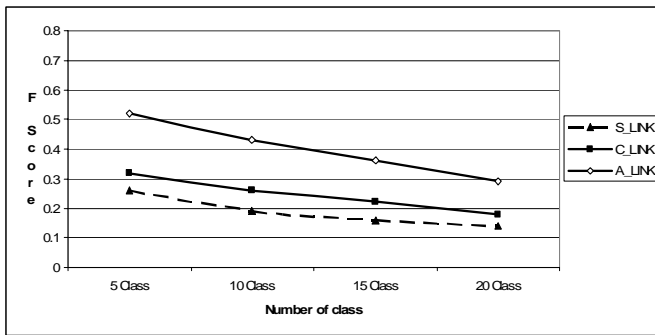


Figure 2. *F Score* measurement for distributed version.

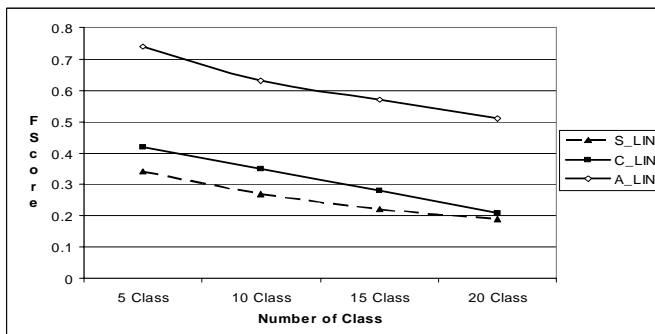


Figure 3. *F Score* measurement for centralized version.

RACHET [1] evaluated the percentage of misclassifications of their distributed algorithm relative to

the results produced by the centralized algorithm. Depending on the separation of classes in the test data set, dendrogram pruning level and number of distributed sites, the misclassification rate varies from 8% to 49% with an average of 25%. However, the authors of RACHET also admit that their evaluation extremely depends on the algorithm used to cluster data in a centralized fashion and therefore it should not be treated as an accurate measurement. On the other hand, our study shows that in case of well separated classes, more than 50% of time the distributed algorithm correctly classifies the documents while in case of centralized algorithm the percentage of correct classification is approximately 75%.

### 5.3 Execution time

Figure 4 and 5 shows the execution time of centralized and distributed approach in logarithmic scale. From the figures, it can be observed that, the distributed version runs significantly faster (30% - 45%) than the centralized approach in case of all three algorithms. Figure 6 shows a closer comparison of centralized and distributed approach in case of average-link algorithm using linear timing scale. It should be noted that the timing measurement for centralized approach does not include the time to bring the documents in a central site. Therefore, a distributed document clustering algorithm certainly runs faster than the centralized approach.

## 6. Conclusions and future works

The results of our experimental study show that distributed version certainly executes faster than the centralized approach, while the quality of the generated clusters decreases because of the distribution. Our results show that, application of RACHET approach to text data may lead to 30% decrease in quality of generated clusters, but it speeds up the whole process about 40%. Such trade-off might be worth of consideration, in the cases when we have limited resources on the merger site and very large datasets in remote repositories. Additional factor which needs to be taken under consideration, when thinking about data centralization, is possible waste of resources for removal of redundant text files, typically duplicated on remote servers of the same company.

Unlike RACHET, we evaluated our implementations on a standard document corpus using widely accepted cluster quality measure. By doing this, we got a clear idea about how much performance degradation occurs when distributed approach is adopted. We believe these tests are crucial to justify applying clustering algorithms in environments of large, distributed repositories of data with high dimensionality, like massive repositories of text documents for instance.

As the main reason of the performance degradation is the approximation of cluster representation used in the merger site, in future, we would like to carry on our research on an

improved approximation that produce more accurate clusterings of documents. For instance, we may replace centroid measurement with measurements of median or mode. Also, we may consider representations that contain the centroid and few other perimeter points of the cluster and observe the effect of those representations on overall performance. At the current moment, due to technical difficulties, we are unable to extend our study beyond four distributed sites. In future we hope to include more machines in our system and conduct exhaustive studies on scalability. We also want to broaden our study so that it incorporates few more well separated and overlapped text corpuses and hope to draw more accurate and general conclusions after analyzing them.

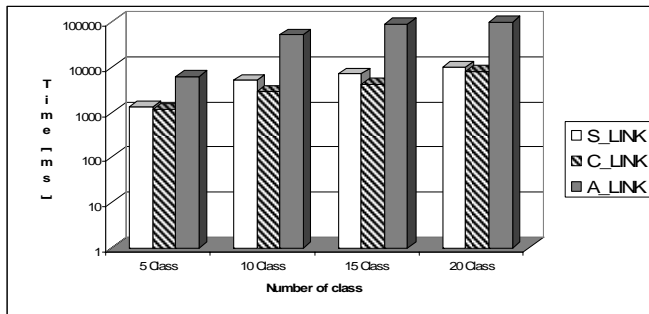


Figure 4. Execution time in log scale for centralized setting

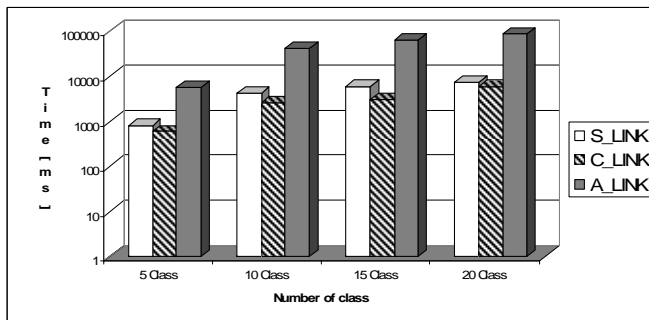


Figure 5. Execution time in log scale for distributed setting

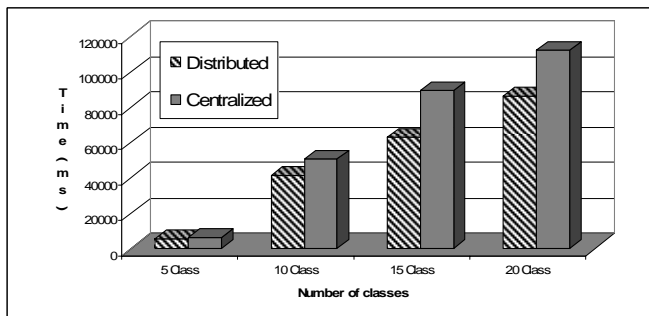


Figure 6. Execution time for average link clustering.

Another important aspect we like to investigate in future is how the algorithm performs in case of different data distributions, such as uneven distribution of classes in each site, partial distribution of a class over few sites, whole distribution of a class in a single site etc. We also hope to extend our study to cluster vertically distributed data. If we consider clustering unrelated documents from different

sites, those documents may not have a common set of useful words, which is one of the primary assumption made by authors of RACHET and us. In that case clustering based on horizontal data distribution will incur a significant amount of errors. Therefore, specifically for document clustering, a distributed clustering approach that can take into account both the horizontal and vertical distribution of data seems to be preferred alternative.

## 7. Acknowledgments

Rafal Angryk would like to thank the Montana NASA EPSCoR Grant Consortium for partially supporting this research.

## References

- [1] N. F. Samatova, G. Ostrouchov, A. Geist, & A. Melechko, RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets, *Distributed and Parallel Databases*, 11(2), 2002, 157-180.
- [2] S. Hettich & S. D. Bay, 20 Newsgroup data set, The UCI KDD Archive, University of California, Dept. of Information and Comp. Science, Irvine, CA, 1999, <http://kdd.ics.uci.edu>.
- [3] M. Eisenhardt, W. Muller, & A. Henrich, Classifying Documents by Distributed P2P Clustering, *In Proceedings of Informatik 2003, GI Lecture Notes in Informatics*, Frankfurt, Germany, September 2003.
- [4] M. Steinbach, G. Karypis, & V. Kumar, A Comparison of Document Clustering Techniques, *In KDD Workshop on Text Mining*, 1999.
- [5] I. S. Dhillon, & D. S. Modha, A Data-Clustering Algorithm on Distributed Memory Multiprocessors, *In the Proc. of Int. Conf. of Large-Scale Parallel Data Mining*, 1999, 245-260.
- [6] G. Forman, & B. Zhang, Distributed Data Clustering Can Be Efficient and Exact, *SIGKDD Explorations*, 2(2), 2000, 34-38.
- [7] C. J. Rijsbergen, *Information retrieval* (Buttersworth, London, second edition, 1989).
- [8] A. Lazarevic, D. Pokrajac, & Z. Obradovic, Distributed Clustering and Local Regression for Knowledge Discovery in Multiple Spatial Databases, *In Proc. of the 8<sup>th</sup> European Symp. on Artificial Neural Networks*, 2000, 129-134.
- [9] E. Johnson & H. Kargupta, Collective, Hierarchical Clustering From Distributed, Heterogeneous Data, *In Lecture Notes in Computer Science*, 1759, Springer-Verlag, 1999, 221-244.
- [10] A. Jain & R. Dubes, *Algorithms for clustering data* (Prentice Hall, 1988).
- [11] R. D'andrade, U-Statistic Hierarchical Clustering, *Psychometrika*, 4, 1978, 58-67.
- [12] A. K. Jain, M. N. Murty & P. J. Flynn, Data clustering: A Review, *ACM Computing Surveys*, 31(3), September, 1999, 264-323.
- [13] <http://mathworld.wolfram.com/Circle-CircleIntersection.html>.