A Fast Online Clustering Algorithm for Scatter/Gather Browsing

Yong Liu, Javed Mostafa, and Weimao Ke
Laboratory of Applied Informatics Research
School of Information and Library Science
University of North Carolina at Chapel Hill
216 Lenoir Drive, CB#3360, 100 Manning Hall, Chapel Hill, NC 27599-3360
yonliu@indiana.edu, jm@unc.edu, wke@unc.edu

Abstract

We present a fast online clustering algorithm which has linear worst-case time complexity and constant running time average for the well-known online visually oriented browsing modeling called Scatter/Gather browsing (Cutting, Karger, Pedersen, and Tukey 1992). Our experiment shows when running on a single processor, this fast online clustering algorithm is few hundred times faster than the parallel Buckshot algorithm running on multiple processors.

1 Introduction

Efficient browsing methods for large volume data collection have been widely examined in recent years. Among existing implementations of various browsing methods, Scatter/Gather browsing is well known for its ease of use and effectiveness in situations in which it is difficult to specify a query precisely (Cutting, Karger, Pedersen, and Tukey 1992; Hearst and Pedersen 1996). However, the major bottleneck of Scatter/Gather browsing is its clustering routines used in the interaction time. The low efficiency clustering routines used in initial Scatter/Gather browsing implementation prevent this method from being applied to large datasets.

This is an important algorithm in information visualization for supporting effective browsing and retrieval of content. Little work has been done to make scatter/gather work efficiently in a online browsing mode. We present a fast online clustering algorithm which can greatly improve the response time of the Scatter/Gather browsing session. The algorithm is composed of two phases. In the offline phase, a cluster hierarchy is generated using traditional hierarchical clustering algorithms. Later in the online phase, drawing on the previously generated hierarchy, a fast algorithm is used to cluster user selected data items in almost constant time. Our experiments show by using this clustering algorithm, each Scatter/Gather browsing session can achieve a several hundred times faster speed than the original Scatter/Gather browsing algorithm (Cutting, Karger, Pedersen, and Tukey 1992; Hearst and Pedersen 1996).

In section 2, we briefly introduce the clustering problem and the categories of clustering algorithms. We describe existing sequential and parallel clustering algorithms in section 3 and 4. In section 5, we discuss the Scatter/Gather browsing and the bottleneck of

the current approaches. In section 6, we introduce the fast online clustering algorithm and provide detailed analysis of the techniques employed and a time complexity of this algorithm. We show our experimental results in section 7 and conclude in Section 8.

2 Clustering Approaches

Clustering of multidimensional data is an important procedure in many information retrieval applications. In these applications, one or more clustering algorithms are used to group similar items together to form clusters whose center or centroid characterizes the group. In a clustering algorithm, similarity between different items can be measured by well known metrics like Euclidean distance or cosine similarity. Olson (1995) introduced certain additional cluster similarity measuring metrics, including single link, average link, complete link, and median. Among all these metrics, we are especially interested in the single link metric. This metric uses the minimum cost edge between points in the two clusters as the cluster distance. Its power lies in the fact that in clustering algorithms applying the single link metric is very similar to finding the minimum spanning tree in a complete graph (Willett 1988).

There exists a large number of data clustering algorithms. Han and Kamber (2001), Han, Kamber, and Tung (2001) classify these algorithms into two main categories - hierarchical algorithms and partitional algorithms. A hierarchical clustering algorithm generates a cluster hierarchy, which is called a dendrogram. A dendrogram is a tree that records the process of clustering. Similar items are connected by links whose level in the tree is determined by the similarity between the two items. The hierarchical algorithms can be further divided into agglomerative approach and divisive approach. The major difference between the two approaches is agglomerative approach works in a bottom-up manner while divisive approach works in a top-down manner. A partitional clustering algorithm obtains a single partition of the data instead of a cluster hierarchy. In other words, it is flat compared to the hierarchical algorithms. One major advantage of this category of clustering algorithms is their fast speed. The widely used K-means method and its variances belong to this category.

Buckshot algorithm was first proposed by Cutting, Karger, Pedersen, and Tukey (1992). It was a hybrid algorithm designed to meet the needs of scatter/gather method for online interactive clustering. A parallel version of the original buckshot algorithm is discussed in (Jensen, Beitzel, Pilotto, Goharian, and Frieder 2002). Although experimentation on this parallelized buckshot shows a slightly better result than the execution time expected by the original buckshot algorithm's $O(kn)$ growth, this algorithm has not fundamentally improved the time complexity of the original algorithm.

3 Sequential Clustering Algorithms

Given a set of n data points, a clustering algorithm is designed to solve the problem of finding k centers such that the average distance from each data point to its nearest center is minimized. This problem is known to be NP-complete.

Olson (1995) reviewed the time complexity of sequential hierarchical clustering algorithms using different distance metrics. Clustering using the single link metric requires the same computational complexity with finding the minimum spanning tree since they have the identical inherent hierarchical structures. As we already know, $O(n^2)$ algorithm exists to find the minimum spanning tree in k-dimensional space (Yao 1982). For the clustering algorithms using centroid and median metrics, Day and Edelsbrunner (1984) showed that these algorithms still have $O(n^2)$ time complexity. The space complexity required by naive sequential hierarchical clustering algorithms using single link metric is $O(n^2)$. Some efficient algorithms using single link, centroid, or median metrics requires only $O(n)$ space.

Partitional clustering algorithms usually have better time complexity than hierarchical algorithms. The K-means algorithm (MacQueen 1967) is a popular clustering method of this category. This algorithm requires k initial cluster centers as inputs. After these centers are given, the algorithm iterates the following two steps. First, distance from each data point to each cluster center is calculated and a data point is assigned to the nearest cluster. Second, each cluster center is re-calculated (usually the mean of all its constituent documents). This process is repeated until the cluster centers converge. Since the number of iterations which this algorithm needs to converge is a constant, the time complexity of K-means algorithm is $O(kn)$.

K-means algorithm provides an approximate solution to the original clustering problem. It has advantages like ease of implementation and fast clustering speed. Although Bottou and Bengio (1995) showed that K-means will always converge to a local minimum , it is not guaranteed to find the global minimum. The starting cluster centers have a great impact on the final clustering results of K-means. Some research work has been done to combine K-means partitional clustering algorithms with hierarchical algorithms, using the hierarchical algorithm to generate initial cluster centers. Cutting, Karger, Pedersen, and Tukey (1992) discussed two algorithms, Buckshot and Fractionation, which are designed to find those initial cluster centers. Buckshot algorithm is a quick hierarchical clustering algorithm with $O(kn)$. It chooses a random sample of the data points (of size $\sqrt{kn}$) and then applies the common hierarchical agglomeration clustering subroutine. Fractionation algorithm is a little more complicated than Buckshot, but can still achieve $O(kn)$ time complexity. These two algorithms are used in implementing of the original Scatter/Gather method.

4 Parallel Clustering Algorithms

Many authors have previously examined parallel algorithms for both hierarchical clustering and partitional clustering (Heckel and Hamann 2004). (Olson 1995) is a comprehensive review on parallel hierarchical clustering algorithms. Two versions of parallel K-means algorithms are discussed in recent literatures. In (Dhillon and Modha 2000), Dhillon and Modha proposed a parallel K-means algorithm on distributed memory multiprocessors. Xu and Zhang (2003) designed a parallel Kmeans algorithm to cluster high dimensional document datasets, which has low communication overhead. Besides K-means, some other classical clustering algorithms also have their corresponding parallel versions, such as the parallel PDDP algorithm (Xu and Zhang 2003) and the parallel buckshot algorithm (Jensen, Beitzel, Pilotto, Goharian, and Frieder 2002).

Theoretically, in many network topologies (n-node hypercube, n-node butterfly, etc.) a parallel clustering algorithm can achieve a $O(n \log n)$ time complexity by using a specific number of processors (typically $\frac{n}{logn}$). However, since it is impractical to arbitrarily increase the number of processors in a working computing environment, such parallel algorithms still suffer from the amount of computation, especially when the working dataset is huge.

5 The Scatter/Gather Method

Scatter/Gather browsing method was first proposed in (Cutting, Karger, Pedersen, and Tukey 1992). In each iteration of this browsing method, the system scatters the dataset into a small number of data point groups, and presents short summaries of them to the user. User can select one or more of the groups for future study. The selected groups are then gathered together and clustered again using the same clustering algorithm. With each successive iteration the groups become smaller and more detailed. Scatter/Gather is particularly helpful in situations in which it is difficult or impossible to specify a query formally. Iterations in this method can help users refine their queries and find the desired information from a large data collection.

Since Scatter/Gather method requires online clustering on a large data corpus, fast clustering algorithms are essential. Two linear time (considering the number of clusters as constant) clustering algorithms were implemented for the original Scatter/Gather method in (Cutting, Karger, Pedersen, and Tukey 1992), which were Buckshot and Fractionation. Both of the two algorithms were used to generate a specific number of clusters on part of the data collection selected by the user in interaction time. Compared to Buckshot, Fractionation algorithm is slower but with higher accuracy. As we mentioned above, both two algorithms have $O(kn)$ time complexity.

The bottleneck in the original implementation of Scatter/Gather method is the online phase of the clustering algorithm. Although better than a quadratic time complexity, $O(kn)$ is still not fast enough considering the target data collections usually are quite

large. Buckshot algorithm achieves its relatively fast speed by picking a small random sample (of size $\sqrt{kn}$) of the whole data collection to generate the initial k centroids in each iteration, which greatly sacrifices its clustering accuracy. To improve the efficiency of the Scatter/Gather method without changing the hierarchical clustering subroutine used in the Buckshot algorithm, one has to decrease the size of the random sample dataset picked in each iteration, which is undesirable since it will further decrease the algorithm's accuracy.

To get a better efficiency of the Scatter/Gather method, we need to improve the clustering algorithm used to find the initial centroids in each clustering iteration in the online phase. There are two choices here, one is to improve the hierarchical clustering subroutine used in Buckshot algorithm, and the other one is to design a new algorithm to replace the Buckshot in the Scatter/Gather method. Since the known lower bound of hierarchical clustering algorithms is $O(n^2)$, there is no much room to improve the hierarchical clustering subroutine on a fixed size of the sample dataset.

Cutting, Karger, and Pedersen (1993) proposed an algorithm that used a precomputed hierarchy of meta-documents for further expansion of selected items and reclustering of the subset. Only dealing with a subset of M meta-documents in each iteration, the algorithm achieved constant interation-time for scatter-gather browsing. However, the reclustering process is not efficient enough for real time interaction because M cannot be too small ($M >> k$, the number of clusters needed). On the other hand, by summarizing descendant documents, meta-documents might be too large to be reclustered efficiently, or too small to be accurately representative.

Our research focuses on designing a new algorithm to achieve faster online clustering speed for the Scatter/Gather method. The new algorithm also takes advantage of a precomputed hierarchy but does not rely on meta-documents for reclustering. We elaborate on our algorithm below and compare it to the approach proposed by Cutting, Karger, and Pedersen (1993).

6 The Proposed Algorithm

6.1 The Fast Online Clustering Algorithm

The process of the hierarchical clustering is the process of constructing dendrogram. Figure 1 shows an example dendrogram.

A hierarchical agglomeration clustering algorithm first considers each distinct data points in a data collection as a separate cluster. Then at each step, two clusters with the shortest distance or the highest similarity are agglomerated. After all of the data points are agglomerated into one cluster, the final hierarchical cluster structure, which is called dendrogram, is formed.

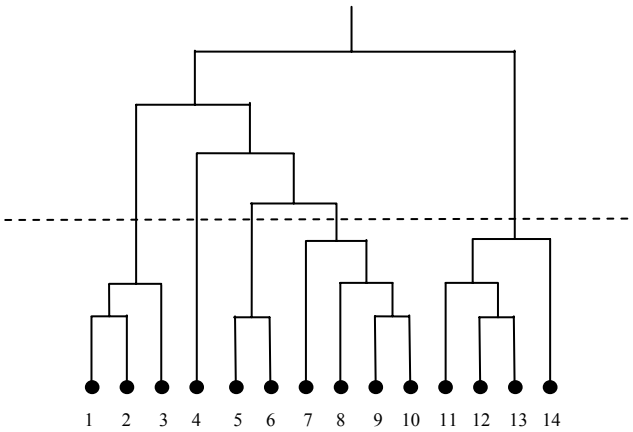

Figure 1: A dendrogram shows how the clusters are agglomerated hierarchically. By cutting the tree at different heights, different number of clusters can be generated. For example, the dashed line in this figure generates five clusters from the dendrogram

When single link metric is used to measure the distance between two clusters, the problem of constructing a dendrogram is very similar to the problem of finding a minimum spanning tree in a complete graph where data points are the nodes and distances between each pair of nodes are the edge weights (see Figure 2).

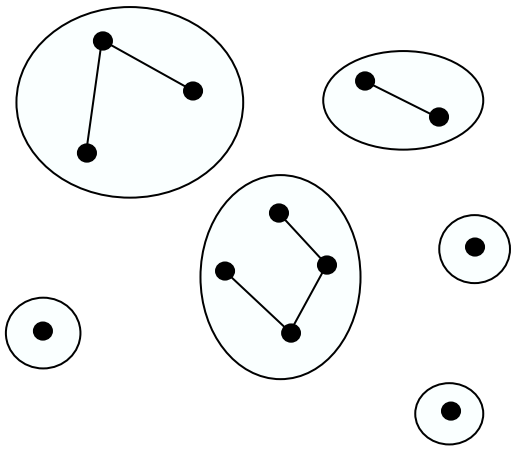

Figure 2: The process of finding a minimum spanning tree using Kruskals algorithm. Each oval corresponds to a cluster generated in the current step.

Since $O(n^2)$ algorithms are known for the minimum spanning tree problem, the hierarchical clustering problem which single link metric used can be also solved in $O(n^2)$ time. Day and Edelsbrunner have shown clustering performed using median metrics is bounded by a function which is $O(n^2)$. In our algorithm, we divide the clustering process into two phases. In the first phase, a hierarchical agglomeration clustering algorithm is used to construct the dendrogram for the whole data corpus, which requires $O(n^2)$ time. In the

second phase, k initial cluster centers are generated drawing on the previous constructed dendrogram in each iteration of online reclustering, which requires only $O(n)$ time.

Although a quadratic time complexity normally means a disaster for the algorithm designers, sometimes it is still acceptable for an offline algorithm which has no strict response time limit, especially when a large scale parallel computing environment is available. The Scatter/Gather method requires a fast online response time. But it can obviously operate with a slower clustering algorithm, since such algorithm may only need to run once per week or even longer. The fast online clustering algorithm shifts the computation load from the online phase to offline phase by making use of this feature of the Scatter/Gather method, so that it achieves better performance in the sense of both speed and accuracy.

In the first phase of the fast online clustering algorithm, an arbitrary hierarchical agglomeration algorithm can be used to construct the dendrogram. The result is represented by a sequence of the agglomerated pairs of data points. Table 1 shows a possible agglomeration sequence of the dendrogram shown in Figure 1.

	Cluster 1	Cluster 2	Distance (Cosine Similarity)	New Cluster No.
1	1	2	0.94	n+1
2	12	13	0.89	n+2
3	5	6	0.70	n+3
4	9	10	0.69	n+4
... ...				
n-1	2*n-5	2*n-2	0.21	2*n-1

Table 1: A possible agglomeration sequence of a dendrogram

Since the size of the data corpus is n, the size of the agglomeration sequence table (or the height of the dendrogram) is $n - 1$. That is to say, the whole agglomeration process takes n-1 steps. To divide the entire data corpora into k groups, only $n - k$ steps of agglomerations are needed when using Kruskal's algorithm. But as our goal in the first phase is to finish as much computation as possible to speed up the second phase online clustering, we still need to construct the entire dendrogram, i.e., fill out the whole agglomeration sequence table.

In the second phase of the fast online algorithm, unlike Buckshot algorithm, we do not choose a random small set as seeds to generate the initial k cluster centers. Instead, in each iteration of online reclustering, we decide upon the initial k centroids by making use of the agglomeration sequence constructed in the first phase.

When the user selects a subset of the scattered clusters for further study, the clustering system needs to recluster all the data points in the subset. In Figure 3, suppose the clusters in the left area to the dashed line are selected, the eight data points in this area then need to be reclustered. Please note the fact that the dashed line will never split a cluster in this case. This is an important prerequisite for the later part of the algorithm.

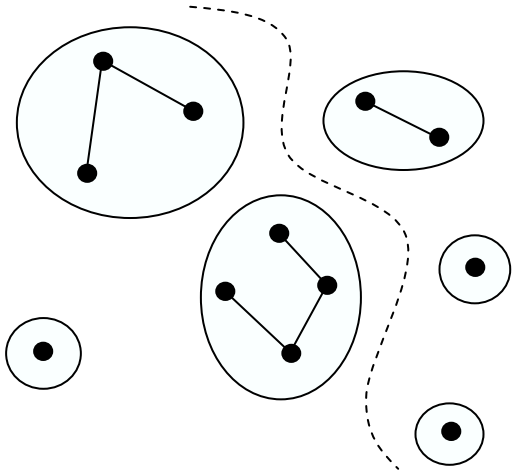

Figure 3: After the user selects a subset of the clusters for further study, the whole cluster set is divided into two parts.

Suppose the desired number of the clusters is k and the number of the clusters in the user selected subset is k', and obviously there exists $k' < k$. Now the problem is transformed to finding k centroids of the data points which are previously clustered into k' groups. Instead of calculating the k centroids from nothing, we make use of the previous knowledge, the agglomeration sequence table. Since we already have k' centroids in current working data collection, we just need to find more centroids to make the total number of centroids equal to k. To achieve this, we split the current k' clusters according to the agglomeration sequence in a bottom-up manner. We scan the table from the bottom, skipping those cluster pairs which has at least one cluster out of the current working data collection. After the first cluster pair whose data points are all in the working collection is found, we split it by removing this entry and adding its two subclusters to appropriate positions in the table. This process is repeated until k. k' clusters have been split, which means k centroids have been identified for the current data collection. The updated agglomeration sequence table is kept for later use in the next Scatter/Gather iteration. This process is shown in Figure 4.

Since the size of the agglomeration sequence table is $n - 1$, the worst case time of the split process is $n + k - k' - 2$, which is $O(n)$. In most cases, the split process will stop in $c(k - k')$ steps, where c is a constant related to k and k' . So this algorithm has almost constant time complexity. Our experimentation demonstrates this conclusion.

6.2 Discussion

The fast online clustering algorithm is much faster than the original Buckshot algorithm, which has $O(kn)$ run time. Another advantage of this algorithm is that it has the same clustering accuracy as the common hierarchical clustering algorithms, which nor-

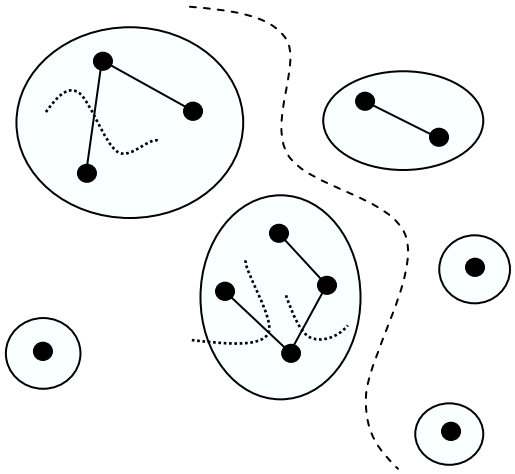

Figure 4: New cluster centers are calculated for the current working subset of the original clusters. The dotted lines denote where split occurs.

mally require quadratic running time. Compared to hierarchical clustering algorithms, the buckshot algorithm generally has much lower accuracy since it only works on a small random subset to calculate the initial centroids. For the offline phase, incremental hierarchical clustering algorithms may be used to do periodic update (Sahoo, Callan, Krishnan, Duncan, and Padman 2006; Can 1993).

Our algorithm looks similar to the one proposed by Cutting, Karger, and Pedersen (1993). They both use a precomputed hierarchy for rescattering and achieve constant interaction-time. One might question the value of the new algorithm here because of the similarities. Nonetheless, there are several essential differences. Firstly, although both are of constant interaction-time, our approach simply traverses the hierarchy and expand the selected clusters without online reclustering. This further improves the online interaction efficiency, essential to a system that provides responsive services.

Secondly, our approach remains flexible for a user to select any clusters in each iteration. Cutting, Karger, and Pedersen (1993) made a mistaken statement that approaches of this kind are too "restrictive" and can serve "only one cluster" at each presentation. Obviously, this is not true as shown in section 6.1. Whereas Cutting, Karger, and Pedersen (1993) focus on coarse-grained patterns of local subsets by reclustering meta-documents, our approach maintains a global view of the local ones and reasonably ignores local reclustering. No evidence has shown that local reclustering produces more relevant results to the users. Previous research have supported the usefulness of traversing a hierarchy without reclustering. Crouch, Crouch, and Andreas (1989) built an interactive browser based on a cluster hierarchy of a hypertext collection, which was revealed to be sufficiently comprehensive and flexible enough to support a variety of user searches.

7 Experimentation

7.1 Setup

The experiments for the sequential and parallel Scatter/Gather clustering algorithms were conducted on research SP cluster at the Research and Academic Computing Center of Indiana University. The research SP cluster is a distributed-memory system consisting of 144 nodes with a total of 646 processors. Each node on the Research SP runs an AIX system. Experiments were conducted on idle nodes of the research SP. All the sequential and parallel clustering algorithms tested in our experiments were written in Java, using JDK 1.4 and mpiJava library, a wrapper for native IBM MPI interface. Experiments were conducted on a cancer dataset and the 2005 Medical Subject Heading (MeSH) list downloadable from the NCBI PubMed (Lipscomb 2000). The cancer dataset consists of 51,783 records containing title and abstracts.

7.2 Experiment Process and Results

In our experiments we examined both a parallel version of the Buckshot algorithm proposed in (Jensen, Beitzel, Pilotto, Goharian, and Frieder 2002) and our fast online clustering algorithms in multiple Scatter/Gather browsing sessions. In each session, both clustering algorithms were used to do online clustering in 10 successive iterations. In each iteration of the Scatter/Gather browsing session, half of the previously generated clusters were randomly selected as the base set on which a further clustering were conducted. For the parallel Buckshot algorithm, 8 processors were used to generate clusters in parallel. For the fast online clustering algorithm, only one processor was used.

	Execution time of Scatter/Gather browsing (seconds)	
	Online parallel Buckshot algorithm (8 processors)	Fast online clustering algorithm (1 processor)
Iteration 1	2.642×10^{1}	3.533×10^{-4}
Iteration 2	1.379×10^{1}	1.813×10^{-4}
Iteration 3	9.177×10^{0}	2.329×10^{-4}
Iteration 4	3.596×10^{0}	6.715×10^{-4}
Iteration 5	2.563×10^{0}	9.348×10^{-4}
Iteration 6	1.367×10^{0}	2.530×10^{-3}
Iteration 7	1.036×10^{0}	1.422×10^{-2}
Iteration 8	6.110×10^{-1}	9.008×10^{-3}
Iteration 9	4.525×10^{-1}	1.984×10^{-2}
Iteration 10	3.406×10^{-1}	3.334×10^{-2}
Average	5.936×10^{0}	8.134×10^{-3}

Table 2: Clustering time of each iteration in a Scatter/Gather browsing session (32 clusters)

Table 2 shows the clustering time of each iteration in a Scatter/Gather browsing session using both two clustering algorithms. In this test, 32 clusters were generated in each iteration. After one clustering iteration was done, 16 out of the 32 clusters were randomly picked to be the base of the next iteration, which simulated user selection. From this table we can see that the average response time in a Scatter/Gather browsing session using parallel Buckshot algorithm is around 6 seconds, which is a little slow for interaction time. When using fast online clustering algorithm, the average response time of a Scatter/Gather browsing session was lowered down to several milliseconds. As it is shown in the table, the clustering speed of the fast online clustering algorithm running on a single processor is several hundreds times faster than the parallel Buckshot algorithm running on 8 processors.

	Execution time of Scatter/Gather browsing (seconds)	
	Online parallel Buckshot algorithm (8 processors)	Fast online clustering algorithm (1 processor)
Iteration 1	4.905×10^{1}	1.012×10^{-3}
Iteration 2	2.585×10^{1}	1.059×10^{-3}
Iteration 3	1.446×10^{1}	1.193×10^{-2}
Iteration 4	9.632×10^{0}	1.942×10^{-3}
Iteration 5	2.343×10^{0}	4.030×10^{-3}
Iteration 6	1.231×10^{0}	1.776×10^{-2}
Iteration 7	1.075×10^{0}	2.261×10^{-2}
Iteration 8	5.107×10^{-1}	3.789×10^{-2}
Iteration 9	3.409×10^{-1}	4.072×10^{-2}
Iteration 10	5.364×10^{-1}	5.668×10^{-2}
Average	1.165×10^{1}	1.957×10^{-2}

Table 3: Clustering time of each iteration in a Scatter/Gather browsing session (64 clusters)

Table 3 shows the clustering time when the number of target clusters were set to 64. When the parallel Buckshot algorithm was used, the average response time in a Scatter/Gather browsing session was around 12 seconds, which is almost unacceptable for online browsing. Considering the parallel Buckshot algorithm achieves a near linear speedup (Jensen, Beitzel, Pilotto, Goharian, and Frieder 2002), the response time will be even worse when using less processors. However, the averaged response time of the fast online clustering algorithm in this case is 20 milliseconds, which is still quite good. To examine the scalability of the fast online clustering algorithm, we tested it using various number of documents and target clusters in a simulated dataset. The results are shown in Figure 5 and 6.

As Figure 5 shows, when the number of target clusters is set to 256, the average response time of the fast online clustering algorithm is around 0.4 seconds on a dataset of 256,000

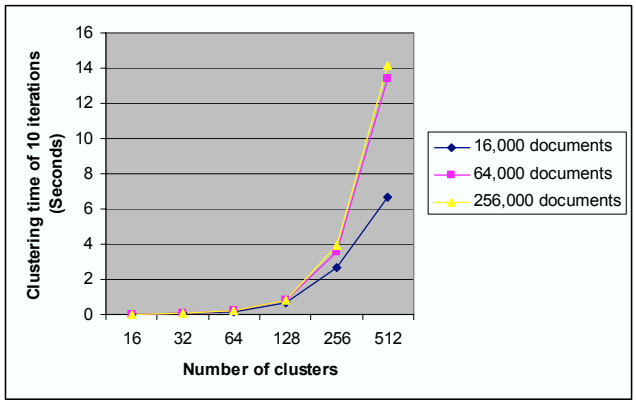

Figure 5: Execution time of fast online clustering algorithm by number of target clusters.

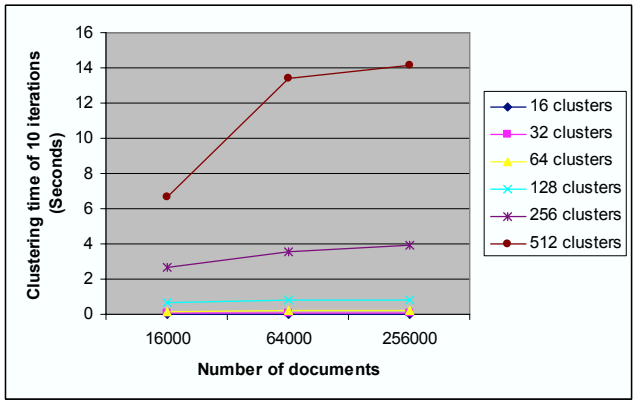

Figure 6: Execution time of fast online clustering algorithm by number of documents.

documents. Only when the number of target clusters increases to 512, the response time of the fast online clustering algorithms starts to sharply increase. Figure 6 shows that the number of target clusters is a dominating factor in the response time. The number of documents to be clustered contributes much less to the time, especially when the number of target clusters is small. This result demonstrates that the fast linear clustering algorithm has an almost constant running time.

8 Conclusion

We have designed a fast online clustering algorithm for Scatter/Gather browsing which achieves near constant response time for each Scatter/Gather iteration. This algorithm requires the construction of a cluster hierarchy using any hierarchical clustering algorithms in the offline phases. A linear table is then used to store the cluster hierarchy for the online phase clustering. On a data collection containing tens of thousands documents, the fast online clustering algorithm runs several hundred times faster than the

parallel Buckshot algorithm. When the size of data collection increases to hundreds of thousands, the clustering time of our algorithm is still satisfactory.

9 Acknowledgments

Authors acknowledge NSF grant #0333623 for support.

References

Bottou, L. and Y. Bengio (1995). *Convergence Properties of the Kmeans Algorithms*, pp. 585–592. Cambridge, MA: The MIT Press.

Can, F. (1993). Incremental clustering for dynamic information processing. *ACM Trans. Inf. Syst. 11*(2), 143–164.

Crouch, D. B., C. J. Crouch, and G. Andreas (1989). The use of cluster hierarchies in hypertext information retrieval. In *HYPERTEXT '89: Proceedings of the second annual ACM conference on Hypertext*, New York, NY, USA, pp. 225–237. ACM Press.

Cutting, D. R., D. Karger, J. O. Pedersen, and J. W. Tukey (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *The 15th Annual ACM-SIGIR*, pp. 318–329.

Cutting, D. R., D. R. Karger, and J. O. Pedersen (1993). Constant interaction-time scatter/gather browsing of very large document collections. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 126–134. ACM Press.

Day, W. and H. Edelsbrunner (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *J. Classification 1*(1), 7–24.

Dhillon, I. S. and D. S. Modha (2000). A data-clustering algorithm on distributed memory multiprocessors. *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence 1759*, 245–260.

Han, J. and M. Kamber (2001). Morgan Kaufmann Publishers.

Han, J., M. Kamber, and A. L. H. Tung (2001). *Spatial Clustering methods in data mining: a survey*. New York.

Hearst, M. A. and J. O. Pedersen (1996). Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 76–84. ACM Press.

Heckel, B. and B. Hamann (2004). Divisive parallel clustering for multiresolution analysis. In *Geometric Modeling for Scientific Visualization*, Heidelberg, Germany, pp. 345–358.

Jensen, E. C., S. M. Beitzel, A. J. Pilotto, N. Goharian, and O. Frieder (2002). Parallelizing the buckshot algorithm for efficient document clustering. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, New York, NY, USA, pp. 684–686. ACM Press.

Lipscomb, C. E. (2000, Jul). Medical subject headings (mesh). *Bull Med Libr Assoc 88*(3), 265–266.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *The 5th Symposium on Math, Statistics, and Probability*, pp. 281–297. Univ. of California Press.

Olson, C. F. (1995). Parallel algorithm for hierarchical clustering. *Parallel Computing 21*.

Sahoo, N., J. Callan, R. Krishnan, G. Duncan, and R. Padman (2006). Incremental hierarchical clustering of text documents. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, New York, NY, USA, pp. 357–366. ACM Press.

Willett, P. (1988). Recent trends in hierarchic document clustering: a critical review. *Inf. Process. Manage. 24*(5), 577–597.

Xu, S. and J. Zhang (2003). A hybrid parallel web document clustering algorithm and its performance study. (366-03).

Yao, A. (1982). On constructing minimum spanning trees in kdimensional space and related problems. *SIAM J. Computing 4*, 21–23.