A Novel Parallel Algorithm for Clustering Documents Based on the Hierarchical Agglomerative Approach

Amal Elsayed Aboutabl[1] and Mohamed Nour Elsayed[2]

[1]Computer Science Department, Faculty of Computers and Information,
Helwan University, Cairo, Egypt
aaboutabl@helwan.edu.eg
[2]Faculty of Computers and Information Sciences, Princess Noura University (Former Riyadh University), Riyadh, Kingdom of Saudi Arabia
mnour@pnu.edu.sa

ABSTRACT

As the amount of internet documents has been growing, document clustering has become practically important. This has led the interest in developing document clustering algorithms. Exploiting parallelism plays an important role in achieving fast and high quality clustering. In this paper, we propose a parallel algorithm that adopts a hierarchical document clustering approach. Our focus is to exploit the sources of parallelism to improve performance and decrease clustering time. The proposed parallel algorithm is tested using a test-bed collection of 749 documents from CACM. A multiprocessor system based on message-passing is used. Various parameters are considered for evaluating performance including average inter-cluster similarity, speedup and processors' utilization. Simulation results show that the proposed algorithm improves performance, decreases the clustering time, and increases the overall speedup while still keeping a high clustering quality. By increasing the number of processors, the clustering time decreases till a certain point where any more processors will no longer be effective. Moreover, the algorithm is applicable for different domains for other document collections.

KEYWORDS

Hierarchical Clustering, Parallel Algorithms, Simulation, Document Collection, Performance Evaluation

1. INTRODUCTION

Document clustering is the grouping of documents into clusters where each cluster contains documents which are more similar to each other than documents in other clusters. The need for document clustering emerges from the existence of vast amount of information on the internet nowadays. This huge amount of documents needs to be organized in such a way to facilitate faster and easier access. There are mainly three challenges with document clustering: high dimensionality of data, high volume of data and the high clustering quality needed [1]. Various document clustering algorithms attempt to find a tradeoff between accuracy and speed.

Clustering techniques are, in general, classified into *partitioning* and *hierarchical* methods [1,2]. Partitioning methods, which are based on the K-means method and its variants, assign every document, iteratively, to a single cluster until k clusters are formed. An initial clustering is chosen (often at random). Every document is compared to the set of cluster centroids and the document is assigned to the cluster with the most similar centroid. Hierarchical clustering produces a set of nested clusters organized as a hierarchical tree, normally visualized as a dendrogram (Figure 1) showing the sequence of merges and splits. The desired number of clusters can be obtained by cutting the dendrogram at the proper level. There are two main types of hierarchical clustering: *splitting* (divisive) and *agglomerative* methods. Splitting methods

DOI : 10.5121/ijcsit.2011.3211

work in a top-down manner starting with one cluster including all documents. Splitting occurs, iteratively, until the desired number of clusters is reached. Agglomerative clustering algorithms are more popular where clusters are merged in a bottom-up manner to form a hierarchy of clusters. Initially, each document is assumed to be a single cluster. Iteratively, the closest pair of clusters are merged until k clusters are formed. A similarity matrix is used to store cluster-to-cluster similarities [3]. The basic hierarchical agglomerative algorithm is as follows:

Compute the similarity matrix
Let each data point be a cluster
Repeat
    Merge the two closest clusters
    Update the proximity matrix
Until the desired number of clusters is reached

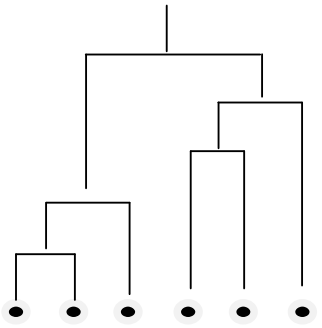


Figure 1. An example dendogram showing how clusters are merged/split at each step

This paper is organized as follows: previous related work is first presented. Second, the proposed parallel algorithm is introduced. Then, simulation work is presented followed by a discussion of the experimental results. Finally, some concluding remarks are recommended.

2. RELATED WORK

A number of document clustering algorithms have been proposed in literature [1]. Hierarchical clustering algorithms are typically viewed as more accurate than other types of algorithms [6]. The reason is that it is a bottom-up algorithm which, initially, assumes that each document is a cluster and then merges the most similar clusters in an iterative manner. The number of iterations depends on k, the number of needed clusters. Because of its quadratic computational complexity, hierarchical clustering algorithms are unpractical for large document collections. A number of algorithms that exploit parallelism in hierarchical clustering algorithms have been introduced in literature [3,4,5,7,8,9,10]. A parallel hierarchical clustering algorithm is introduced in [3] which is used as the clustering subroutine for a parallel buckshot algorithm. A distributed clustering technique (RACHET) is developed [9] in which hierarchical clustering algorithms are used to generate local dendrograms. Descriptive statistics are used to minimize the representation of cluster centroids and hence decrease communication cost.

Various methods are classically known for defining inter-cluster similarity [6,12,13,14]. The *Single-link* method calculates the similarity of two clusters based on the two most similar points (documents) in different clusters. This single-link merge criterion is local. In the *complete-linkage* method, similarity of two clusters is based on the two least similar (most distant) points in different clusters. This complete-link merge criterion is non-local; the entire structure of the clustering can influence merge decisions. Similarity, in the *group average* method, is the average of the pairwise distance between points in the two clusters. This method can avoid the pitfalls of the single-link and complete-link criteria both of which equate cluster similarity with the similarity of a single pair of documents.

3. PROPOSED ALGORITHM

In this work, parallelism is exploited to perform document clustering based on the hierarchical agglomerative clustering method. Hierarchical agglomorative clustering starts by considering each document as a separate cluster. The two closest clusters are merged (agglomerated) into a single cluster. The agglomeration step is repeated forming a hierarchical structure (tree). This tree structure can be broken at certain edges to obtain the final clustering based on some criterion, such as the number of needed clusters or some cluster quality criterion. In our algorithm, a threshold value is used to stop merging. This threshold value is decremented with each iteration of the loop till a minimum threshold value is reached.

3.1. Data Model

The data considered for clustering in this work are documents. The vector space model, a common representation for data to be clustered [6,11], is used to represent documents for clustering. Each document is represented as a vector whose dimensionality is the number of terms (features) considered for clustering.

$D = \{d_i\}$ $i=1,\ldots,N_D$

$T= \{t_j\}$ $j=1,\ldots,N_T$

$V = \{v_i\}$, $i=1,\ldots, N_D$

$v_i= [v_{ij}]$

$$v_{ij} = \begin{cases} 1 \; iff \;\; t_j \in d_i \\ 0 \; otherwise \end{cases}$$

The vector size is equal to the number of terms considered.

D is the set of documents

T is the set of document terms considered

N_T is the number of terms considered

V is the set of document vectors

N_D is the total number of documents

v_i is the vector space representation of document d_i

v_{ij} is the j^{th} element in vector v_i and has the value of 1 if term t_j exists in document d_i , 0 otherwise.

Processors can be represented as a connected graph G(P,E) where $P=\{p_i\}$, $i=1,\ldots N_P$ is the set of processors, N_P is the number of processors. $E=\{e_{ij}\}$ is the set of edges (communication links) connecting the nodes where the edge e_{ij} connects the nodes p_i and p_j , $i \neq j$. Processors are assumed to be fully connected, so the number of edges is $|E|= N_P (N_P -1)/2$

Initially, documents are distributed equally among processors. Each processor p_i is assigned N_D/N_P documents. The set of documents assigned to p_i is denoted by D_i, $D_i \subset D$ and $|D_i|=N_D/N_P$. The proposed clustering method is based on hierarchical agglomerative clustering which is performed independently on each processor in parallel. As an initial step, each document is assumed to be a cluster on its own. The set of clusters on p_i is denoted by $C_i = \{c_k\}$ $k=1,\ldots, N_{C_i}$ the number of clusters on processor p_i. Each cluster c_k contains a set of documents $D_k \subset D_i$. Initially, $|C_i|=|D_i|$ and $|c_k|=1$. Each processor p_i performs clustering based on data about clusters existing in its local memory only. At each merging step, the cluster-to-cluster similarity matrix is computed. Each processor has its own similarity matrix. The two most similar clusters are merged into one cluster. If c_l and c_m are the two clusters to be merged on p_i then a new cluster $c_k=D_l \cup D_m$ and $C_i= (C_i- \{c_l, c_m\}) \cup c_k$ and the number of clusters $|C_i|$ is decremented by one. Merging continues on each processor until the distance between the two most similar clusters is below *threshold* (an input threshold value). A new merging phase starts after clusters are exchanged between processors. Every even-numbered processor exchanges half of its clusters with the next processor. In the following exchange step, every odd-numbered processor exchanges half of its clusters with the next processor. Exchange of clusters continues to occur

interchangeably between even and odd-numbered processors. It is assumed that the number of processors N_p is even. The value of *threshold* is decremented before each merging step by a predefined value $thresh_{dec}$. Exchange of clusters between processors allows each processor to take the clustering decision i.e. which clusters to merge, based on a larger pool of clusters than the one assigned at the beginning. Decrementing threshold before each merging step aims at improving clustering quality with each merging step. The following sequence of steps occurs on each processor:

- merge the two most similar clusters until threshold is reached.
- p_i and p_{i+1} (i is even) exchange half of their clusters
- decrement threshold by $thresh_{dec}$
- merge the two most similar clusters until threshold is reached.
- p_i and p_{i+1} (i is odd) exchange half of their clusters
- decrement threshold by $thresh_{dec}$

This sequence is repeated till threshold reaches a predefined input value $thresh_{min}$.

3.2. Parallel Threshold-based Clustering Algorithm

Input : P, D, threshold, $thresh_{dec}$, $thresh_{min}$
Output : C_i $\forall p_i$ (the set of clusters C_i formed on processor p_i)

1. Assign N_D/N_P documents to each processor p_i.
2. Each cluster contains only one document.
3. odd=true
4. while $threshold \geq thresh_{min}$
5. for each processor p_i do in parallel
6. Compute similarity matrix for clusters C_i on p_i
7. Find the two most similar clusters c_l and c_m
8. if distance(c_l,c_m)$\leq$ *threshold*
9. Merge c_l and c_m into one cluster c_k , $c_k=D_l \cup D_m$ and $C_i=(C_i-\{c_l, c_m\})\cup c_k$
10. Repeat from step 6
11. endif
12. endfor
13. if odd=true
14. for each processor p_i i=1,3 to N_P-1 do in parallel
15. p_i and p_{i+1} exchange half of their clusters
16. endfor
17. odd = false
18. else // odd is false
19. for each processor p_i i=2,4 to N_P do in parallel
20. p_i and $p_{(i \bmod Np)+1}$ exchange half of their clusters
21. endfor
22. odd = true
23. endif
24. Decrement threshold by $thresh_{dec}$
25. endwhile.

4. SIMULATION

The described models are simulated by a developed C++ program. The input parameters to the simulator are the number of processors, the number of documents, document vectors, $thresh_{old}$, $thresh_{min}$ and $thresh_{dec}$. The simulator outputs include the resulting clusters on each processor and some quality measures such as the average cluster similarity, number of merges performed

on each processor, maximum number of merges, number of sends by each processor and maximum number of sends.

4.1. Data Structures

Double linked lists are used to represent processors, clusters and documents. These linked lists, which are created dynamically, are nested so as to simulate the parallel clustering system (Figure 2).

Processors double linked list: Each node in the linked list represents a processor. The attributes associated with each processor are:
Lclus : the number of local clusters
fdoc and ldoc : the first and last document id's assigned initially to the processor
nmerges : number of cluster merges
nsends : number of clusters sent
chead,ctail : point to the head and tail of the clusters' linked list of the processor
SM : similarity matrix
next and prev : point to the next and prev processor's node in the linked list

Clusters double linked list : Each processor's node in the processors linked list points to its own local double linked list of clusters. Each node in the clusters' double linked list represents a cluster on the related processor. The attributes associated with each cluster are :
cid : cluster id
centroid : a vector representing the computed cluster centroid
ndocs : the number of documents in the cluster
dhead,dtail : point to the head and tail of the documents' linked list of the cluster
next and prev : point to the next and previous cluster in the linked list

Documents double linked list: Each cluster node in the clusters linked list points to its double linked list of documents. Each node in the documents double linked list represents a document having a document id docid, next and prev pointers.

Merging of clusters is implemented by connecting the documents double linked lists of one of the two clusters to be merged to the end of the documents double linked list of the other cluster. The first cluster is then freed from memory and the attributes of the other cluster node are updated. Merging is performed easily and quickly using the dhead and dtail attributes in the clusters double linked list as well as the next and prev attributes in the documents double linked list.

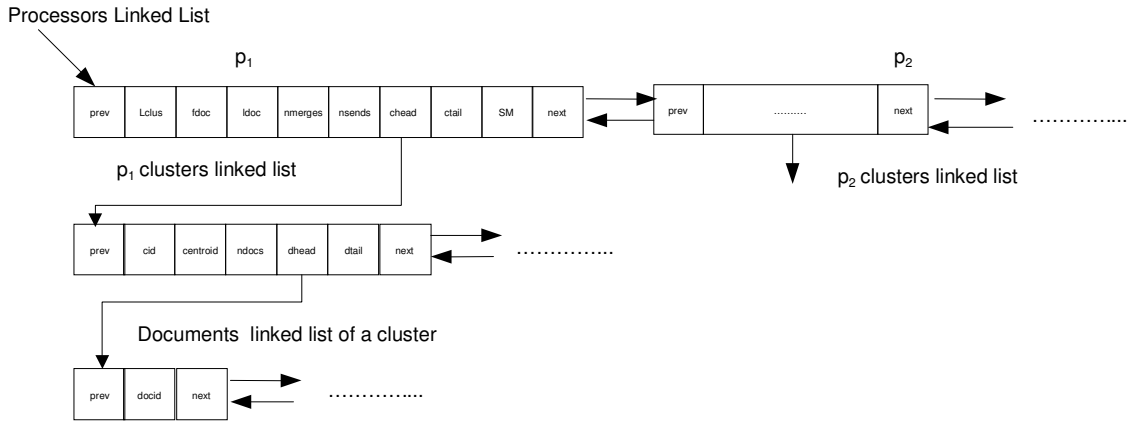


Figure 2. Nested double linked lists used for simulating processors, clusters and documents

4.2. Architectural Model

The architectural model adopted in this work is the multiprocessor system based on message passing (Figure 3). The parallel architectural model consists of a set of processing nodes; each has its own memory. The clustering operation is done in parallel by the computer processors. During the document exchange among processors, the processors can communicate together via the interconnection network which is fully-connected. Such network facilitates sending and receiving messages among processors besides any other data transfer.

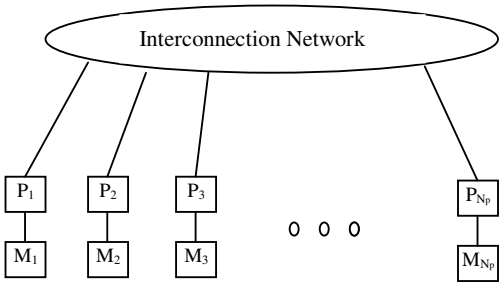


Figure 3. The Multiprocessor System based on Message Passing

4.3. Domain Model

The proposed parallel algorithm is run on a document collection with 749 documents obtained from CACM. The collection of the CACM is one of the most popular datasets and/or benchmarks for testing the document clustering operation. Each document in the collection is described using the vector space model. Each document is represented also to be a binary vector in the term-space. The document vectors are extracted based on 18 terms. For more details about the chosen document collection, the reader can refer to http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/.

4.4. Cost Model

The cost model in this research work involves several aspects. It includes the merging operations of the most similar clusters, exchanging clusters among processors, updating the similarity matrix on each processor and monitoring of threshold values as well as the sizes of the cluster pools. The number of merges is computed as the sum of the maximum number of merges performed by all processors during all iterations of the while loop of the algorithm. The maximum number of merges during an iteration is considered to be the processing makespan for this iteration. Hence, the number of merges are computed as $\sum_i \max m_j \quad \forall P_j$, where i is the number of iteration of the outer loop of the algorithm and m_j is the number of merges on processor P_j. Computation of the cluster centroids and the similarity values are considered as follows.

Computing Cluster Centroids : A cluster centroid is a vector representing the central point of the cluster. It is the average of all document vectors in the cluster. Its coordinates are the arithmetic mean of each dimension separately over all document vectors in the cluster. We denote the centroid vector for cluster c_k as $\overrightarrow{w_k}$. The i^{th} element of the centroid vector $\overrightarrow{w_k}$ is computed as :

$$w_{k_i} = \frac{\left[\sum_{i=1}^{|D_k|} x_i\right]}{|D_k|} \quad \forall \vec{x} \epsilon V_k$$

where x_i is the i^{th} element of document vector $\vec{x}$, $|D_k|$ is the number of documents in cluster c_k and V_k is the set of document vectors in cluster c_k.

Computing similarity : The average cluster similarity is computed based on the cosine similarity measure; the most commonly used similarity measure in document clustering [3,11]. Cosine similarity is a normalized metric because its values fall in [0,1]. Using cosine similarity, the similarity between the two vectors $\vec{x}_i$ and $\vec{x}_j$ is computed as :

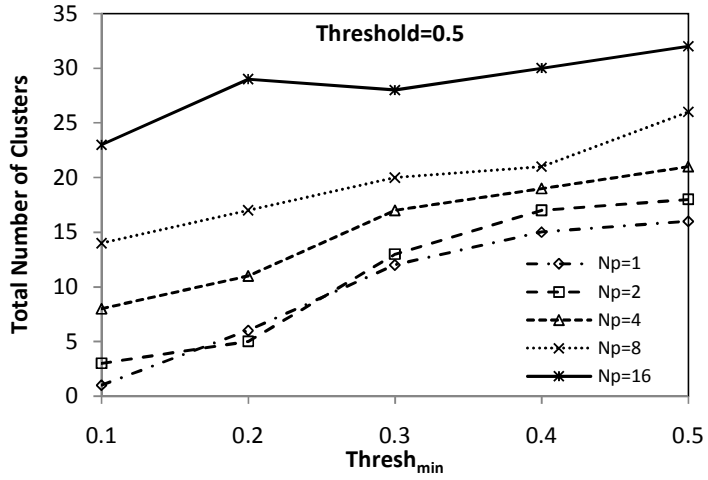
$$S(\vec{x_i}, \vec{x_j}) = \frac{\vec{x_i} . \vec{x_j}}{\|\vec{x_i}\| \|\vec{x_j}\|}$$

where the Euclidean norm $\|\vec{v}\| = \sqrt{v_1^2 + \cdots + v_d^2}$, d is the dimensionality of the vector .

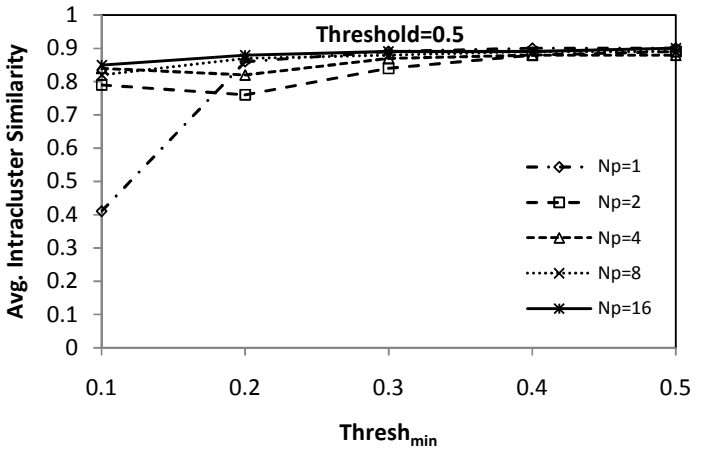
5. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, simulation results are presented to measure the performance of the proposed parallel algorithm. It is important to mention that speed and capacity of various components in high performance computers have increased dramatically. This involves processors, memories, links, and others. The interconnection network topologies have become more flexible and efficient. As the adopted interconnection network is fully-connected, the waiting time to exchange data among processors is negligible. In our case there are exactly 120 links when the number of processing nodes considered is 16. On the other hand, the number of sends among processors and the time consumed for merging the most similar documents are considered. Simulation experiments are performed by varying the number of processors, threshold and $thresh_{min}$. Figures 4(a-c) and Figures 5(a-c) show the simulation results obtained when threshold values of 0.8 and 0.5 are used. As the resulting number of clusters is not predefined, each experiment results in a different number of clusters. The number of processors used is varied from 1 to 16 as shown and a fully-connected architecture is assumed. The resulting number of clusters increases as the number of processors and $thresh_{min}$ are increased. Higher number of processors results into more documents partitioning and hence a larger number of clusters. Moreover, higher values for $thresh_{min}$, which is meant to increase the cluster quality by restricting the merging process, also leads to an increase in the number of clusters. It is noticeable that the number of resulting clusters has a maximum of 38 clusters for threshold values from 0.1 to 0.6 for all values of N_p while the number of clusters increases at a higher rate for $thresh_{min}$ values from 0.7 to 0.9 with a maximum of 168 clusters. The average intracluster similarity, based on cosine similarity, is used to measure the clustering quality. Higher $thresh_{min}$ values produce higher quality of clusters. The effect of using more processors in increasing the cluster quality is more apparent for smaller than for higher $thresh_{min}$ values. Merges and sends are computed (Table 4) as the sum of the maximum number of merges/sends performed by all processors during all iterations of the while loop of the algorithm.

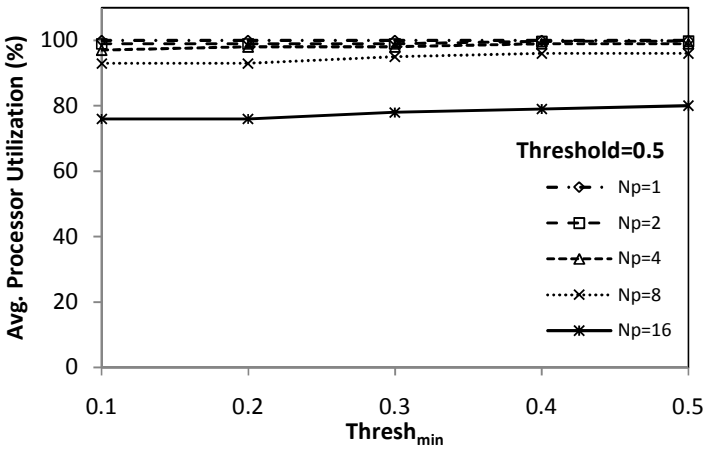
A sample of the simulation results recorded per iteration is shown in Tables 1-3. These three experiments produce the same number of clusters (20 clusters); the reason for which these particular experiments are presented here to evaluate the benefits of parallelism. Our algorithm does not require a predefined number of clusters. Based upon these three experiments, speedup, utilization and efficiency are calculated. Regardless of whether the number of clusters is known or unknown beforehand, the achieved speedup has the same meaning here since the number of input documents and the number of resulting clusters are identical in the chosen experiments (the same problem size). Speedup is the ratio of the execution time for clustering documents on one processor to that time for clustering the same documents into the same number of clusters on N_p processors. Utilization is a measure of how much, of the total time, processors are busy. Efficiency is the ratio between the speedup achieved and the number of processors used.



(a)

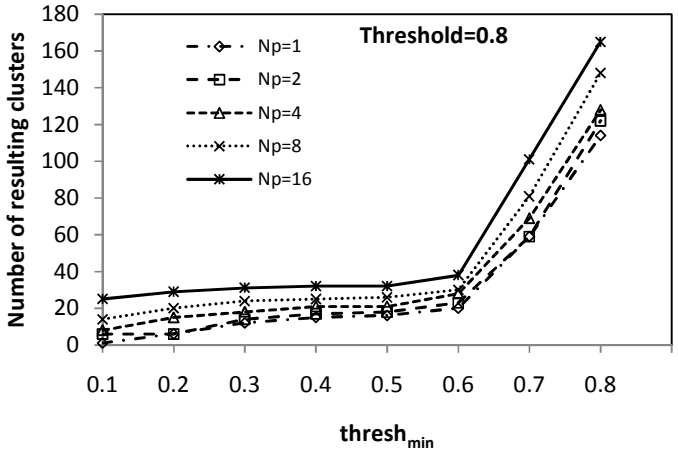


(b)

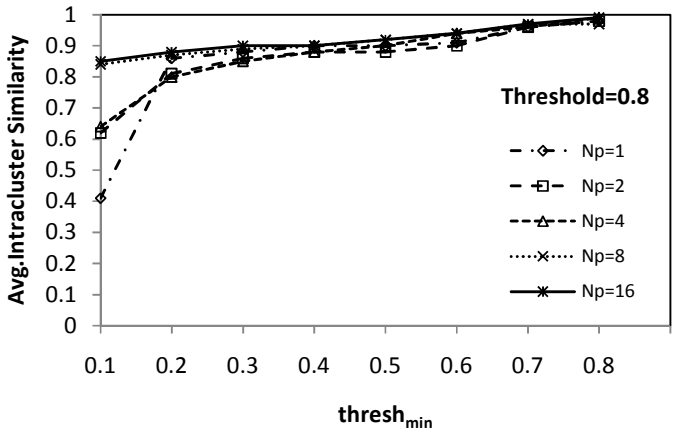


(c)

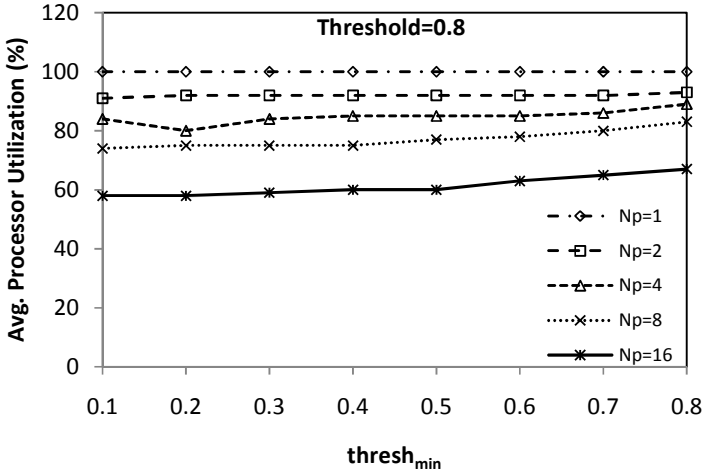
Figure 4 (a-c). Simulation results with thresh=0.5, $thresh_{min}$=0.1,0.2,…,0.5 and $thresh_{dec}$=0.1. Each point in the charts represents an independent experiment.



(a)



(b)



(c)

Figure 5 (a-c). Simulation results with thresh=0.8, thresh$_{min}$=0.1,0.2,…,0.8 and thresh$_{dec}$=0.1. Each point in the charts represents an independent experiment.

Table 1. Experiment results for three consecutive iterations with N_p=1 , $thresh_{dec}$=0.1 .

Measure		i=1 t=0.8	i=2 t=0.7	i=3 t=0.6
Per iteration	No. of Clusters	114	59	20
	No. of Merges	635	55	39
	Max. Merges	635	55	39
	Avg. intra-cluster similarity	0.98	0.96	0.93

Table 2. Experiment results for five consecutive iterations with N_p=4 , $thresh_{dec}$= 0.1 (i denotes the iteration number and t is the threshold).

Measure		i=1 t=0.8	i=2 t=0.7	i=3 t=0.6	i=4 t=0.5	i=5 t=0.4
Per iteration	No. of Clusters	128	69	36	24	20
	No. of Merges	621	59	33	12	4
	No. of Sends	0	65	36	19	14
	Max. Merges	175	22	12	4	2
	Max. Sends	0	24	14	6	4
	Avg. intra-cluster similarity	0.99	0.96	0.94	0.90	0.88
End of exp.	Speedup	3.05				
	Efficiency	75%				
	Utilization	0.85				

Table 3. Experiment results for 7 consecutive iterations with N_p=8 , $thresh_{dec}$ = 0.1 .

Measure		i=1 t=0.8	i=2 t=0.7	i=3 t=0.6	i=4 t=0.5	i=5 t=0.4	i=6 t=0.3	i=7 t=0.2
Per iteration	No. of Clusters	148	83	48	35	29	26	20
	No. of Merges	601	65	35	13	6	3	6
	No. of Sends	0	76	44	26	20	17	15
	Max. merges	90	14	8	4	3	1	2
	Max. sends	0	15	10	5	3	3	2
	Avg. intra-cluster similarity	0.99	0.97	0.94	0.92	0.90	0.89	0.87
End of exp.	Speedup	5.17						
	Efficiency	65%						
	Utilization	0.75						

6. CONCLUSION

In this research work, we have analyzed and discussed a method for document clustering based on the agglomerative hierarchical approach. We proposed a high performance method for parallelizing the agglomerative document clustering algorithm. The experimental results in this research work presented the effectiveness of the proposed parallel algorithm. The effectiveness of the proposed parallel algorithm was very clear in decreasing the clustering time, increasing the overall speedup, and achieving high quality clustering. This was achieved for all the resulting clusters for all the experiments. Parallel processing in this concern is a very successful solution for the main problems associated with the document clustering. It is concluded that parallel processing plays an important role in dealing with the complexities of high

dimensionality of data used during the clustering operation. Parallelism has also a vital role in handling the high volume of data of any document collection. Apart from the document clustering, it is expected that the proposed parallel algorithm is scalable and effective for other object clustering in different applications.

Table 4. Max. number of merges and max. number of sends summed over all iterations of the algorithm . The number of processors N_p is varied. Threshold=0.5 and 0.8 and $thresh_{min}$=0.1,..,0.8

Threshold t		**t=0.5**					**t=0.8**							
$Thresh_{min}$		0.1	0.2	0.3	0.4	0.5	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
N_p=1	Merges	748	743	737	734	733	748	743	737	734	733	729	690	635
	Sends	0	0	0	0	0	0	0	0	0	0	0	0	0
N_p=2	Merges	376	374	370	367	366	408	404	400	398	396	392	370	337
	Sends	16	14	10	5	0	86	83	79	74	68	61	42	0
N_p=4	Merges	191	189	187	185	184	221	219	217	215	213	209	197	175
	Sends	12	10	7	4	0	56	54	51	48	44	38	24	0
N_p=8	Merges	99	98	96	95	94	124	122	120	119	116	112	104	90
	Sends	10	8	6	4	0	40	38	36	33	30	25	5	0
N_p=16	Merges	60	59	58	57	56	78	77	75	74	72	68	62	54
	Sends	7	5	4	2	0	28	26	24	22	19	15	8	0

REFERENCES

[1] M. Steinbach , G. Karypis & V. Kumar (2000), "A comparison of document clustering techniques", *In 6th ACM SIGKDD World Text Mining Conference*, Boston, MA, USA.

[2] S. Xu & J. Zhang (2004), "A Parallel Hybrid Web Document Clustering Algorithm and its Performance Study", *The Journal of Supercomputing (Special issue: Parallel and distributed processing and applications)*, Vol. 30 , No. 2, pp. 117-131.

[3] R. Cathey, E. Jensen, S. Betzel, O. Frieder & D. Grossman (2007), " Exploiting parallelism to support scalable hierarchical clustering", *Journal of the American Society for Information Science and Technology*, Vol. 58, No. 8, pp. 1207-1221.

[4] E. Dahlhaus (2000), "Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition", *Journal of Algorithms* , Vol. 36, No. 2, pp. 205–240.

[5] S. Ranka & S. Sahni (1991), "Clustering on a Hypercube Multicomputer*", IEEE Transactions on Parallel and Distributed Computing*, Vol. 2, No. 2, pp.129-137.

[6] C. Olson (1995), "Parallel algorithms for hierarchical clustering", *Parallel Computing* , Vol. 21, pp. 1313–1325.

[7] K. Hammouda & M. Kamel (2006), "Collaborative Document Clustering" , *In proceedings of 2006 SIAM Conference on Data Mining (SDM06)*, pp. 453-463, Bethesda, Maryland.

[8] S. Rajesekanan (2005), "Efficient Parallel Hierarchical Clustering Algorithms", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 6, pp. 497-502.

[9] D. Deb, M. Muztaba Fuad & R. Angryk (2006), "Distributed Hierarchical Document Clustering", *In Proceedings of the 2nd IASTED International Conference on Advances in computer science and technology*, pp. 328-333, Puerto Vallarta, Mexico.

[10] R. Alfred, E. Paskaleva, D. Kazakov & M. Bartlett (2007), "Hierarchical Agglomerative Clustering of English-Bulgarian Parallel Corpora", *In Proceedings of International Conference on Recent Advances in Natural Languages Processing (RANLP 2007)*, Borovets, Bulgaria.

[11] M. Berry, Z. Drmac & E. Jessup (1999), "Matrices, Vector Spaces and Information Retreival", *Siam Review*, Vol. 41 No. 2, pp. 335-362.

[12] M. Khalilian & N. Mustapha (2010), "Data Stream Clustering: Challenges and Issues", *In Proceedings of the International Multiconference of Engineers and Computer Scientists IMECS 2010*, Hong Kong, pp. 978-988.

[13] G. J. Torres, R. B. Basnet, A. H. Sung, S. Mukkamala, & B. M. Ribeiro (2009), "A Similarity Measure for Clustering and its Applications", *International Journal of Electrical, Computer, and Systems Engineering*, Vol. 3, No. 3, pp. 164-170.

[14] R. Martinez-Morais, F.J. Alfaro-Cortes, & J. L. Sanchez (2010), "Providing QoS with the Deficit Table Scheduler", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 3, pp. 327-341.

Authors

Amal Elsayed Aboutabl is currently an Assistant Professor at the Computer Science Department, Faculty of Computers and Information, Helwan University, Cairo, Egypt. She received her B.Sc. in Computer Science from the American University in Cairo and both of her M.Sc. and Ph.D. in Computer Science from Cairo University. She worked for IBM and ICL in Egypt for seven years. She was also a Fulbright Scholar at the Department of Computer Science, University of Virginia, USA. Her current research interests include parallel computing and performance evaluation.

Mohamed Nour Elsayed received his M.Sc. and Ph.D. degrees in computer engineering from the Faculty of Engineering Ain Shams University, Cairo in 1987 and 1993 respectively. He taught about twenty years at the American University in Cairo. Also, he was the head of the Informatics Dept. at the Electronics Institute, Cairo. Currently, he is a Professor of Computers at the Faculty of Computers and Information Sciences at Princess Nora University (former Riyadh University), Riyadh-KSA. He is a member of the University Scientific Council. He served as a program committee member of several computer conferences. His research interests include high performance computing, performance analysis, and artificial intelligence applications. He is a member of the IEEE, IEEE Computer Society, Computational Intelligence Society, and Language Engineering Society.