Parallel Implementation of Information Retrieval Clustering Models

Daniel Jiménez and Vicente Vidal

Department of Computer Systems and Computation,
Polytechnic University of Valencia. Spain
{djimenez, vvidal}@dsic.upv.es

Abstract. Information Retrieval (IR) is fundamental nowadays, and more since the appearance of the Internet and huge amount of information in electronic format. All this information is not useful unless its search is efficient and effective. With large collections parallelization is important because the data volume is enormous. Hence, usually, only one computer is not sufficient to manage all data, and more in a reasonable time. The parallelization also is important because in many situations the document collection is already distributed and its centralization is not a good idea.

This is the reason why we present parallel algorithms in information retrieval systems. We propose two parallel clustering algorithms: α-Bisecting K-Means and α-Bisecting Spherical K-Means. Moreover, we have prepared a set of experiments to compare the computation performance of the algorithms. These studies have been accomplished in a cluster of PCs with 20 bi-processor nodes and two different collections.

1 Introduction

Briefly, the development of a total IR system is constituted by three phases:

1. Preprocessing: This part considers all processes once the collection is given and the weighted matrix of terms by documents is created. For example: lexical analysis, stemming, etc.
2. System Modeling: This part considers all necessary processes to prepare the collection for the queries. Modeling with LSI techniques, clustering methods, statistical techniques, etc.
3. System use: This part considers the set of queries given by the user.

This paper is centered in the second part. We study the parallelization of several models of IR systems, more specifically clustering, included in the algebraic paradigm. We assume a weighted matrix of terms by documents already distributed. Finally we stop when the system model is parallelized.

In this paper we present the parallelization of the α-Bisecting K-Means and the α-Bisecting Spherical K-Means algorithms. We also compare the performance between the sequential and parallel versions. This paper is organized as follows:

M. Daydé et al. (Eds.): VECPAR 2004, LNCS 3402, pp. 129–141, 2005.

In section 2 we present the parallelized IR models including all parallel algorithms developed. Next, in section 3, we explain the experiments executed and the collections used. Furthermore, we describe the cluster of PC's where the experiments have been executed. And finally we present the results of the experiments. In the last section, section 4, we present all the conclusions reached.

2 Parallelized IR Models

Our algorithm is based upon IR clustering models, concretely the K-Means algorithm family [1, 2, 3, 4, 5]. The main reasons are their good performance in IR, and their natural tendency to parallelization.

In all algorithms, we have used the following syntax: M represents the weighted matrix of terms by documents; M_i represents the i-th sub-matrix of M; m_j represents the j-th column of a matrix (j-th document); m represents the number of terms; and n represents the number of documents.

2.1 Bisecting K-Means Algorithm

The α-Bisecting K-Means [3, 6] is a variant of the well known algorithm K-Means [7, 3], which has been parallelized many times [8, 9, 10]. The α-Bisecting K-Means uses the α-Bisection algorithm to perform all bisections required. In the α-Bisection, the α parameter depends of the document collection and number of clusters searched. This parameter is used to guarantee that all documents in a cluster have affinity, and all clusters have the same level of affinity.

Algorithm 1 Sequential α-Bisection

Input: $M \in \Re^{m \times n}$, α, *tol, maxiter.*

Output: $\pi_l \in \Re^{m \times nl}$ and $\pi_r \in \Re^{m \times nr}$ where π_l is the left cluster, π_r is the right cluster and $nl{+}nr{=}n$.

Step-1: Select a normalized concept vector, $c \in \Re^m$, of a set of documents.

Step-2: Divide M into two sub-clusters π_l and π_r according to:

$$m \in \pi_l \text{ if } m^{\mathrm{T}} \bullet c \geq \alpha$$
$$m \in \pi_r \text{ if } m^{\mathrm{T}} \bullet c < \alpha$$

Step-3: Calculate the new normalized concept vector of π_l, nc, defined as:

$$t = \frac{1}{nl} \sum_{m \in \pi_l} m \; ; \; nc = \frac{t}{\|t\|}$$

Step-4: If the stopping criterion ($\|nc - c\| \leq tol$) has been fulfilled or *maxiter*=0 then finalize, else decrement *maxiter*, $c{=}nc$ and go to step-2.

A collection could be seen as a sphere . The sphere's volume is determined by its radius raised to the document's dimension (number of terms). But we attempt

to create k clusters, therefore we need to divide the collection in k sub-spheres with the same approximate volume. The α's value represents the radio of each k sub-sphere. Although we have considered the theoretical aspects indicated, also we have refined the formula that represents the α parameter experimentally. Finally we define α as follows:

$$\alpha = 1 - \frac{\overline{x} - \sigma}{\sqrt[m]{k}} \tag{1}$$

where $\overline{x}$ is the mean cosine distance, σ the standard deviation among each document of the collection and the centroid of the collection, and m is the number of terms.

In our implementation, step-1 of algorithm 1 is omitted because it is received as a parameter. With algorithm 1, we obtain two clusters, however we want to build k clusters $\{\pi_j\}_{j=1}^{k}$. Here is where appears algorithm 2, the α-Bisecting K-Means.

Algorithm 2 Sequential α-Bisecting K-Means

Input: $M \in \Re^{m \times n}$, $k \in \aleph$.
Output: k disjoint clusters$\{\pi_j\}_{j=1}^{k}$.
Step-1: Select M as the cluster to split, set t=1 and set α using expression (1).
Step-2: Find two clusters π_t and π_{t+1} with the α-Bisection, algorithm 1.
Step-3: If t=k-1 then finalize else select π_{t+1} as the cluster to split, increment t and go to step-2.

In the implementation, when algorithm 2 calls the α-Bisection algorithm (step-2) it also sends the required concept vector. Actually, the concept vector selected is a document of the collection chosen randomly.

In our parallel version of the α-Bisecting K-Means (algorithm 2), developed in this paper, we have assumed a distribution by documents as show in figure 1.

In others words, each processor has a little number of documents from the collection, achieving the necessary communications to all processors so they have the same normalized concept vector. Hence, we have developed a parallel algorithm to create a normalized concept vector (algorithm 3), a parallel version of

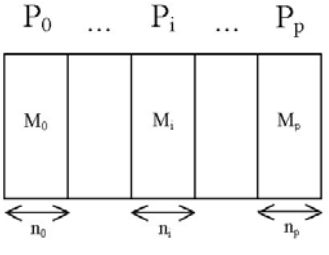


Fig. 1. Distribution of the collection

the α-Bisection (algorithm 4) and finally the parallel version of the α-Bisecting K-Means algorithm (algorithm 5).

Algorithm 3 Parallel Normalized Concept Vector

Note: This code is run by the i-th processor.
Input: $M_i \in \Re^{m \times ni}$, $idx_i \in \Re^{nidx_i}$.
Output: $ncv \in \Re^m$.
Step-1: $ncv_i = \sum_{j=1}^{nidx_i} m_{idx_i(j)} \ : \ m_{idx_i(j)} \in M_i$.
Step-2: Execute the global reduction adding all $nidx_i$ in N. Now N, in all processors, is equal to the sum of all $nidx_i$.
Step-3: Execute the global reduction adding all ncv_i in ncv. Now ncv, in all processors, is equal to the sum of all processed documents.
Step-4: $ncv = \frac{ncv}{N}$
Step-5: $ncv = \frac{ncv}{\|ncv\|}$

We should remark three things. First, that algorithm 3 calculates the normalized concept vector of several columns of the input matrix, actually the columns included in the index vector, *idx*. Second, step-2 and step-3 are optimized in the implementation. And third, in the implementation of the *ncv* 2-norm, it is also achieved in parallel, distributing the *ncv* vector and using a global reduction.

When we look for a cluster, we do not build explicitly the clusters. Instead, we build a vector *idx*. Which includes the cluster index of each document. The vector *idx* is used in the parallel Normalized Concept Vector algorithm and the parallel version of the α-Bisection.

Algorithm 4 Parallel α-Bisection

Note: This code is run by the i-th processor.
Input: $M_i \in \Re^{m \times ni}$, $idx_i \in \Re^{nidx_i}$, α, *tol*, *maxiter*.
Input/Output: $MyClusters_i \in \Re^{ni}$.
Step-1: Select a set of local documents and calculate in parallel the normalized concept vector $c \in \Re^m$ (algorithm 3).
Step-2: Divide M_i into two sub-clusters according to:

$$\left.\begin{array}{l} MyClusters_i[j] = MyClusters_i[j] + 1 \text{ if } m_j^{\mathrm{T}} \bullet c \geq \alpha \\ MyClusters_i[j] = MyClusters_i[j] \text{ if } m_j^{\mathrm{T}} \bullet c < \alpha \end{array}\right\} \begin{array}{l} j = idx_i(l) \\ l = 1 \ldots nidx_i \end{array}$$

Step-3: Calculate (algorithm 3) the new normalized concept vector, *nc*.
Step-4: Evaluate the stopping criterion ($\|nc - c\| \leq tol$). If it has been fulfilled or *maxiter*=0 then return $MyClusters_i$, else decrement *maxiter*, *c*=*nc* and go to step-2.

We should remark that, in our implementation, the 2-norm of$\|nc - c\|$ (step-4, algorithm 4) is achieved in parallel, distributing the vector and using a global

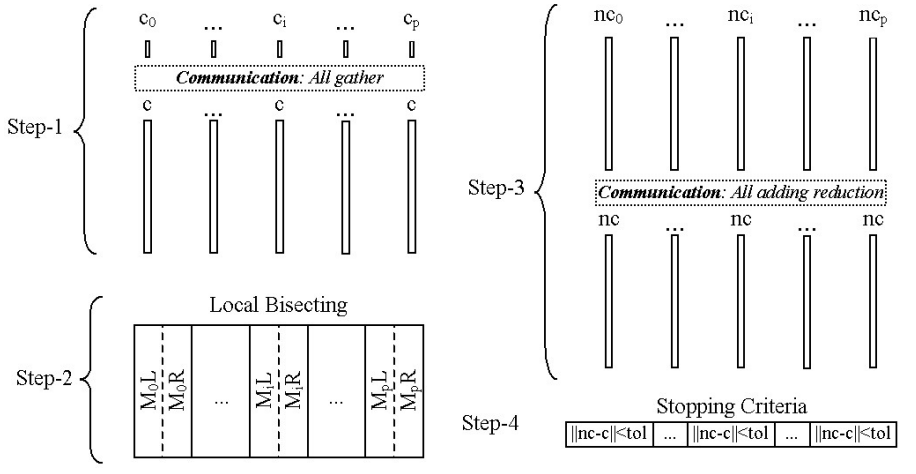


Fig. 2. Parallel α-Bisection scheme

reduction, analogous to the calculus of the *ncv* 2-norm in algorithm 3. Additionally, as done in the sequential version, the implementation of step-1 of algorithm 4 is omitted because it is received as a parameter.

We show in figure 2 a diagram depicting the parallel version of the α-Bisection.

And finally we present the parallel Bisecting K-Means algorithm and its diagram (algorithm 5 and figure 3):

Algorithm 5 Parallel α-Bisecting K-Means

Note: This code is run by the i-th processor.
Input: $M_i \in \Re^{m \times ni}$, $k \in \aleph$.
Output: k disjoint clusters$\{\pi_j\}_{j=1}^{k}$.
Step-1: Select M_i as the cluster to split, set t=1 and set α using expression (1).
Step-2: Find clusters π_t and π_{t+1} with the parallel α-Bisection, algorithm 4.
Step-3: If t is equal to k-1 then return all clusters else select π_{t+1} as the cluster to split, increment t and go to step-2.

The calculation of α is performed in parallel (step 1 algorithm 5). Concretely the parallelization is achieved to obtain the mean and the standard deviation. Each processor works with its own part of the collection, then all processors perform a global reduction adding the partial local sums. Just to remark, that each processor selects randomly a document of its sub-collection, then all processors calculate the normalized concept vector of all the documents chosen.

An schematic diagram depicting the parallel α-Bisecting K-Means algorithm is showed in figure 3.

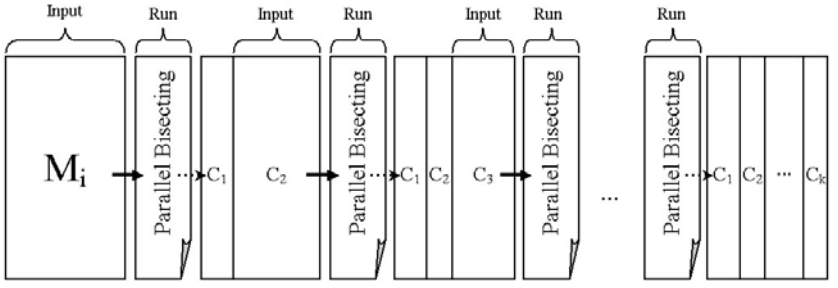


Fig. 3. Parallel Bisecting K-Means scheme

2.2 Bisecting Spherical K-Means Algorithm

An important variant of the K-Means algorithm is the Spherical K-Means [11, 4], which uses the cosine similarity. Hence, the clusters obtained constitute spheres, accordingly to its name. This algorithm seeks k disjoint clusters $\{\pi_j\}_{j=1}^{k}$ maximizing the quality of the obtained partitioning. This quality is given by the following objective function:

$$f\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k} \sum_{x \in \pi_j} x^{\mathrm{T}} \bullet c_j \tag{2}$$

Consequently, the stopping criterion normally used is:

$$\left| f\left(\{\pi_j^t\}_{j=1}^{k}\right) - f\left(\{\pi_j^{t+1}\}_{j=1}^{k}\right) \right| \leq \epsilon \bullet f\left(\{\pi_j^{t+1}\}_{j=1}^{k}\right) \tag{3}$$

Algorithm 6 Sequential α-Bisecting Spherical K-Means

Input: $M \in \Re^{m \times n}$, *tol*, *maxiter*, $k \in \aleph$.
Output: k disjoint clusters$\{\pi_j\}_{j=1}^{k}$.

Step-1: Calculate k initial clusters $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$ applying the α-Bisecting K-Means, algorithm 2 and its normalized concept vectors $\left\{c_j^{(0)}\right\}_{j=1}^{k}$. Initialize t=0.

Step-2: Build a new partition $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ induced by $\left\{c_j^{(t)}\right\}_{j=1}^{k}$ according to:

$$\pi_j^{(t+1)} = \left\{\forall x \in M : \; x^{\mathrm{T}} \bullet c_j^{(t)} > x^{\mathrm{T}} \bullet c_l^{(t)} \; , \; 1 \leq l \leq k \; , \; j \neq l\right\}, \; 1 \leq j \leq k$$

Step-3: Calculate the new concept vector $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$ of the new partition.

Step-4: Finalize if the stopping criterion, expression 3, is fulfilled ($\epsilon = tol$) or t is equal to *maxiter*, else increment t and go to step-2.

The modification of the Spherical K-Means includes the α-Bisecting K-Means as the first step. The α-Bisecting Spherical K-Means, was developed in [5]. Furthermore, in this paper, we improve it because we have optimized the α-Bisecting K-Means changing the initial selection of the concept vectors and defining the α parameter in terms of the collection and number of clusters. The idea of the α-Bisecting Spherical K-Means is to use the α-Bisecting K-Means algorithm to obtain a first approximated solution and later refine it with the Spherical K-Means, taking advantage of both.

In this paper, we develop a parallel version of the α-Bisecting Spherical K-Means. We use a distribution by documents (see figure 1), the same distribution used in the parallel α-Bisecting K-Means (algorithm 5).

Algorithm 7 Parallel α-Bisecting Spherical K-Means

Note: This code is run by the i-th processor.
Input: $M_i \in \Re^{m \times ni}$, *tol*, *maxiter*, $k \in \aleph$.
Output: k disjoint clusters$\{\pi_j\}_{j=1}^{k}$.

Step-1: Calculate k clusters $\left\{\pi_j^{(0)}\right\}_{j=1}^{k}$ applying the parallel α-Bisecting K-Means, algorithm 5 and in parallel its concept vectors $\left\{c_j^{(0)}\right\}_{j=1}^{k}$, algorithm 3. Initialize t=0.

Step-2: Build a new partition $\left\{\pi_j^{(t+1)}\right\}_{j=1}^{k}$ induced by $\left\{c_j^{(t)}\right\}_{j=1}^{k}$ according to:

$$\pi_j^{(t+1)} = \left\{\forall x \in M_i : \; x^{\mathrm{T}} \bullet c_j^{(t)} > x^{\mathrm{T}} \bullet c_l^{(t)} \; , \; 1 \leq l \leq k \; , \; j \neq l \right\}, \; 1 \leq j \leq k$$

Step-3: Calculate in parallel the new normalized concept vector $\left\{c_j^{(t+1)}\right\}_{j=1}^{k}$ associated to the new partition, with algorithm 3.

Step-4: Finalize if the stopping criterion, expression 3, is fulfilled ($\epsilon = tol$) or t is equal to *maxiter*, else increment t and go to step-2.

The evaluation of the stopping criterion also occurs in parallel. In fact, the objective function evaluation is done in parallel. Basically, each processor calculates the partial sum with the documents that it contains; then all processors perform a global reduction adding the partial sums, obtaining the final result.

3 Experiments

3.1 Document Collections

In the study we use two test collections with different characteristics, what follows next is a concise description of both collections.

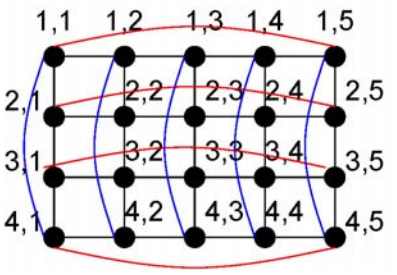


Fig. 4. Topology of the Cluster

Times Magazine of 1963: This collection contains articles from a 1963 Times Magazine and it has been obtained from ftp://ftp.cs.cornell.edu/pub/smart/time/ site. The collection language is English. A total number of 425 documents are included in the collection. Each document have an average of 546 words and 53 lines. The contents referred to world news, especially politics frequently mentioning words which in fact, reminded us of the typical news contents available in the cold war era.

DOE-TREC: This collection has been obtained from TREC (Text REtrieval Conference) site [12]. It contains a lot of abstracts of the U.S. Department of Energy (DOE). In this case, the collection language is English, too. The DOE-TREC collection is larger than the Times Magazine collection, presenting over 225000 documents.

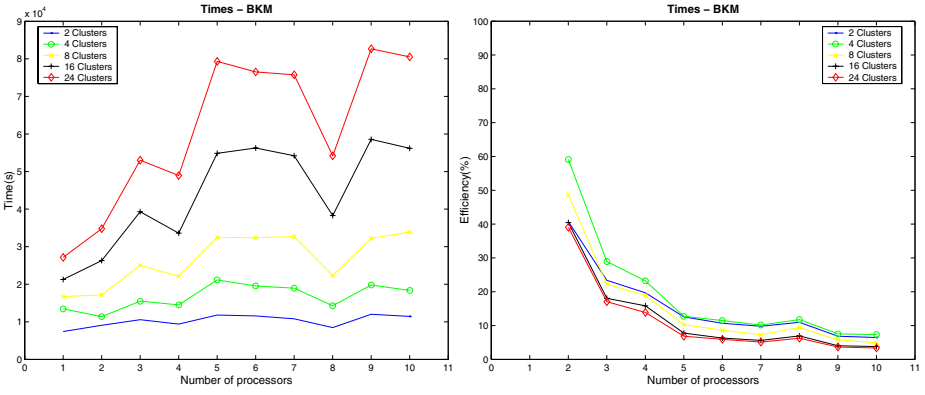
3.2 Cluster of PCs

All performance studies of the parallel algorithms developed have been performed in a cluster of PCs with the following characteristics: the cluster has 20 bi-processor nodes (Pentium Xeon at 2 Ghz). These nodes have a toroidal 2D topology with SCI interconnection, as shown in figure 4. The main node is constituted by 4 Ultra SCSI hard disk of 36 Gbytes at 10000 rpm. The others nodes have an IDE hard disk of 40 Gbytes at 7200 rpm. All nodes have the same amount of RAM, concretely 1 GB.

Actually, we only include the experiments performed into 10 processors. We have always selected processors located into different nodes to optimize the performance. Two processors of the same node have to share an unique net card. Besides, the results obtained with 10 processors show clearly the performance evolution, and therefore we avoid presenting complex performance evolution graphs.

3.3 Description of Experiments

We compare the final parallel versions with their respective sequential versions. We study mainly the total time, efficiency and scalability. In all experiments, we use the two document collections described above, and tests were performed in the cluster of PCs commented above, too.



(a) Total time (b) Efficiency

Fig. 5. The α-Bisecting K-Means with the Times collection

In summary, the experiments to be carried out are indicated below:

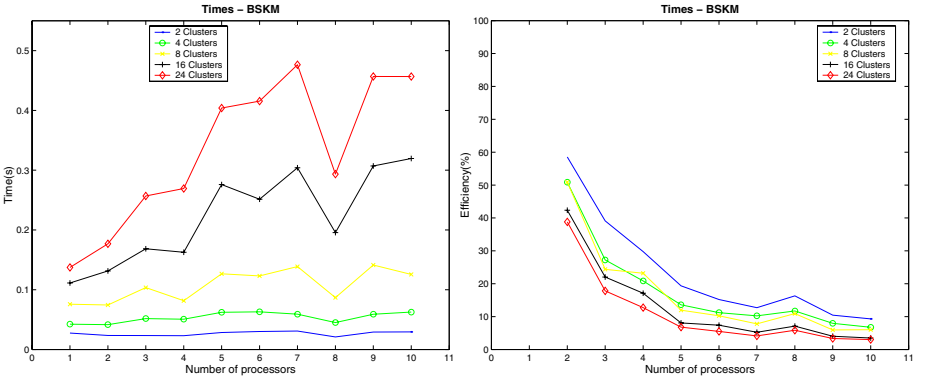
- Parallel α-Bisecting K-Means vs. Sequential α-Bisecting K-Means: total time, efficiency and scalability.
- Parallel α-Bisecting Spherical K-Means vs. Sequential α-Bisecting Spherical K-Means: total time, efficiency and scalability.

All studies have been achieved with mean times, because the random selection of first centroids cause different number of required iterations and, hence, different times. So, we measure 11 times each sample (each method with a number of clusters and a number of processors).

3.4 Results of Experiments

Next, we present a series of graphs the summarizing results of our experiments. In figure 5, we show the study of the α-Bisecting K-Means method with the Times collection, we present the total time consumed by the algorithm and its efficiency. Analogously, in figure 6, we present the study of the α-Bisecting Spherical K-Means, also with the Times collection. In the same way, in figures 7 and 8, we depict the study of both methods, the α-Bisecting K-Means and the α-Bisecting Spherical K-Means algorithms, using as data the DOE collection.

In figure 5.a we observe that, in general, when we increase the number of processors, independently of the number of clusters, the total time increases, too. The reason of this behavior is the collection size. Due to the fact that the Times collection has so few documents, the communication times obtained are much bigger than the time reduction obtained when using more processors. But, we can highlight that when we work with four or eight processors performance improves. This occurs in all cases, independently of the number of clusters that we have used as a parameter. We suppose this behavior is caused by cache memory or a



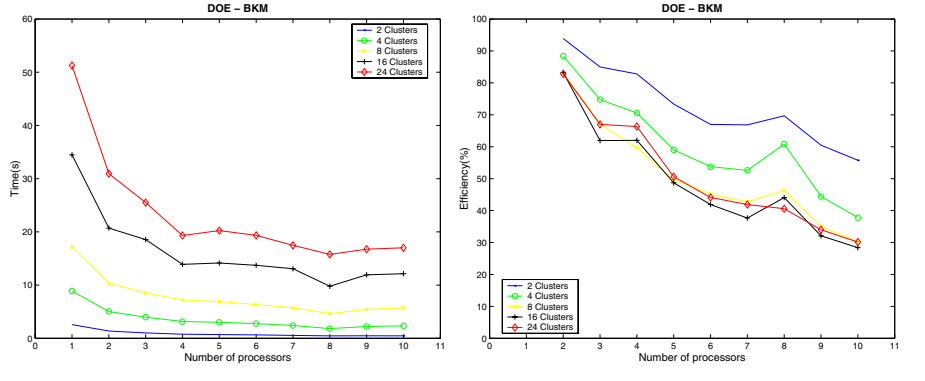
(a) Total time (b) Efficiency

Fig. 6. The α-Bisecting Spherical K-Means with the Times collection

favorable selection of the Cluster's nodes that improve the communications, as figure 4 shows depending on the nodes selected, communication costs can vary.

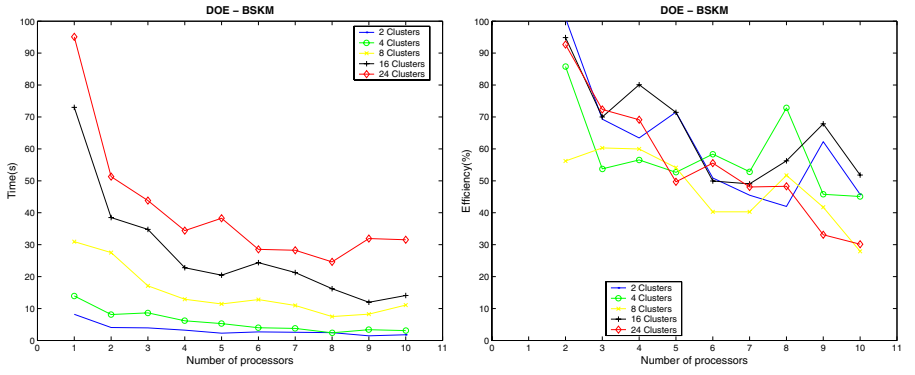
On the other hand, working with four and eight clusters and two processors we improve a little the total time. But the difference is insignificant. We can see this in figure 5.b, the efficiency study. All curves have a poor efficiency, and when we increase the number of processors efficiency decreases. Only with four and eight clusters and two processors we obtain a relatively acceptable efficiency (approximately 60% and 50% respectively).

The behavior of the α-Bisecting Spherical K-Means method (figure 6) is analogous to the α-Bisecting K-Means method (figure 5), both with the Times col-



(a) Total time (b) Efficiency

Fig. 7. The α-Bisecting K-Means with the DOE-TREC collection



(a) Total time (b) Efficiency

Fig. 8. The α-Bisecting Spherical K-Means with the DOE-TREC collection

lection. When the number of processors increase the total time increases (figure 6.a), except with four and eight processors that decreases a little. But in this case, the curves obtained in small cluster groups (i.e. two, four and eight) are softer. And, the efficiency found in two clusters has improved much (figure 6.b), although the others are maintained approximately equal.

When the study is performed with a bigger collection, the parallelism benefits appear. In figure 7.a we show the total time needed to cluster the DOE collection with the α-Bisecting K-Means method. We observe, independently of the number of document clusters built, the total time improves when we use more processors. Logically, this improvement is more distinguishable when more document clusters are formed. We highlight the times obtained with four and eight processors. In both cases the performance relatively improves more than in the rest of cases. This situation is the same that in the study with the Times collection. Furthermore we assume that the reason is also the same, either the cache memory or the document distribution of the collection into the nodes. Seeing figure 7.a, we deduce that for the DOE collection size performance improves acceptably using up-to four processors. If we use more than four processors performance falls drastically.

This can be observed better in the efficiency study (figure 7.b). We observe upper efficiency, approximately, 60% when we work with up-to four processors. And with five or more processors efficiency falls down to 50% or lower. Though, it can be seen that the graph curve for four clusters delays its descent whereas the graph curve for two clusters maintains a level above 60% almost all the time, only dropping down to 55% with ten processor. Figure 7.b also presents small improvements in efficiency corresponding with four and eight processors.

Finally, we have studied the α-Bisecting Spherical K-Means method with the DOE collection. It can be seen in figure 8. In general, we observe a similar behavior to the α-Bisecting K-Means method with respect to the total time study (figure 8.a), but the curves, in this case, are more irregular. The relative

improvement with four and eight processors is softer in this case, although it can also be observed.

Figure 8.b shows the efficiency of this method. The curves are more chaotic, these increase and decrease without apparent criterion. Though, the level of efficiency in almost all cases are over 50% and when we use two processors we achieve about 90% of efficiency.

The chaotic behavior could be caused by the aleatory initialization of the α-Bisecting K-Means method, which is achieved as an initialization of the α-Bisecting Spherical K-Means method. In this case, when the collection size is large it appears to be insufficient three iterations only of the α-Bisecting K-Means method to make stable the initialization of the α-Bisecting Spherical K-Means method.

As a finally conclusion, we would like to say two things. First, we can see in the experiments that when more clusters are built, better is the time improvement, but efficiency is deteriorated. This occurs because when a cluster is built the methods work with a subcollection of the original collection, and so on. Then as these subcollections are smaller than the original collection the performance is deteriorated. And second, these methods are scalable because as the collection size increases, it is possible to use more processors to improve performance.

4 Conclusions

The principal conclusion is with the parallelization of these algorithms we reduce the necessary time to build the required clusters. Of course, the collection size has to be enough large to obtain an improvement of performance, but nowadays the actual collections are usually very large and these will grow in the future. Besides, the parallel algorithms showed have a good efficiency, and good scalability, too. On the other hand we did not achieve great efficiency because the collection matrix is very sparse (0.0270 in the Times collection and 0.00053 in the DOE collection), and hence the number of operations to execute is small. We use functions optimized for sparse data (for example, the SPARSKIT library [13]).

Another conclusion is that the distribution of the collection used in the parallel algorithms presented in this paper cause good performance, a conclusion which we might indicate as simple and natural. As a matter of fact when a real collection is classified as distributed, what we would normally assume is that its documents are distributed as well.

We can also conclude that the most important operation in the algorithms is the calculation of the normalized centroid. Therefore, its optimization is fundamental to obtain good performance.

Finally, we want to indicate three future works in this research. First, we should improve the selection of the initial centroids in the α-Bisecting K-Means method, because the solution of the algorithms is more dependent and varies according to the chosen centroids. We should eliminate the randomness or modify the algorithm to grant independence respect to the initial selection.

Second, we should look for some mechanism to improve the calculation of the normalized centroid. Alternatively, we should develop a new version of both parallel algorithms to replace in some iterations the global parallel calculation of the concept vector by a local calculation.

And third, we should study what is the optimum number of iterations to use in the α-Bisecting K-Means method to make stable the initialization of the α-Bisecting Spherical K-Means method in the most of cases.

Acknowledgments

This paper has been partially supported by the Ministerio de Ciencia y Tecnología and project FEDER DPI2001-2766-C02-02

References

1. Savaresi, S., Boley, D.L., Bittanti, S., Gazzaniga, G.: (Choosing the cluster to split in bisecting divisive clustering algorithms)
2. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17** (2001) 107–145
3. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. KDD-2000 Workshop on Text Mining (2000)
4. Dhillon, I.S., Fan, J., Guan, Y.: Efficient clustering of very large document collections. In R. Grossman, C. Kamath, V.K., Namburu, R., eds.: Data Mining for Scientifec and Engineering Applications. Kluwer Academic Publishers (2001) Invited Book Chapter.
5. Jiménez, D., Vidal, V., Enguix, C.F.: A comparison of experiments with the bisecting-spherical k-means clustering and svd algorithms. Actas congreso JOTRI (2002)
6. Savaresi, S.M., Boley, D.L.: On the performance of bisecting k-means and pddp. First Siam International Conference on Data Mining (2001)
7. Hartigan, J.: Clustering Algorithms. Wiley (1975)
8. Dhillon, I.S., Modha, D.S.: A data-clustering algorithm on distributed memory multiprocessors. In: Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence. (2000) 245–260
9. Kantabutra, S., Couch, A.L.: Parallel k-means clustering algorithm on nows. NECTEC Technical Journal **1** (2000) 243–248
10. Xu, S., Zhang, J.: A hibrid parallel web document clustering algorithm and its performance study. (2003)
11. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Technical report, IBM (2000)
12. http://trec.nist.gov/: (Text retrieval conference (trec))
13. Saad, Y.: SPARSKIT: A basic tool kit for sparse matrix computations. Technical Report 90-20, NASA Ames Research Center, Moffett Field, CA (1990)