E–Learning Using Mapreduce

R. RAJU

Assistant Professor, Department of Information Technology
Sri Manakula Vinayagar Engineering College
Puducherry – 605 107, India

V. VIJAYALAKSHMI

Department of Computer Science and Engineering
Sri Manakula Vinayagar Engineering College
Puducherry – 605 107, India

R.T. SHOWMYA

Department of Computer Science and Engineering
Sri Manakula Vinayagar Engineering College
Puducherry – 605 107, India

Abstract - E-Learning is the learning process created by interaction with digitally delivered content, services and support. Learner's profile plays a crucial role in the evaluation process and to improve the e-learning process. The customization of content is necessary to provide better services. Mapreduce is distributed programming model which is developed by Google. The aim of this paper is to increase the speed and to decrease the processing time by using K-MR algorithm instead of K-Means clustering algorithm. K-Means algorithm can be applied to the MapReduce model and can efficiently process large datasets called K-MR algorithm. This system customizes the contents based on learner's performance and effective for both learner and instructor.

Keywords: mapreduce, E-learning, K-Means, K-MR.

I. INTRODUCTION

Rapid innovation in information and communications technology is transforming the way we work, the way we interact, the way we learn, and the way we live. In the education sector, e-learning increases access to education by making it possible for students to fit their education into family and work schedules and by providing a greater programmatic choice of quality courses. E-learning allows multiple students to simultaneously enroll in more than one school in order to achieve their particular learning goals in a timelier manner. At the same time, each school can maintain a different set of general education, prerequisites, academic specialties, and other institutional requirements.

MapReduce is an important technology of Google. It is a kind of programming models, which comes from the traditional functional programming ideas to deal with mass data. The name of MapReduce comes from two core operation in the model: Map and Reduce. *Map*, written by the user, takes an input pair and produces a set of *intermediate* key/value pairs. The *Reduce* function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values.

Zhao, Ma in "Parallel K-Means Clustering Based on MapReduce", [12] demonstrated that the K-Means algorithm can be successfully applied to the MapReduce model and can efficiently process large datasets.

The remaining of this paper is organized as follows. Section 2 describes related works. Section 3 contains explanation of mapreduce and K-means algorithm. Section 4 describes proposed system and about k-MR algorithm. Section 5 contains the Experimental result and Section 6 concludes the paper.

II. RELATED WORKS

In this section we briefly present some of the work related to E-learning. Anane et al, [1] concerned with learning content provision mediated by a learning management system (LMS), which complies with instructional management system (IMS) standards. The requirements for the LMS, as a tool for course management, are specified through different perspectives. Issues related to content creation and delivery are identified and addressed from the perspectives of the instructor, the learner, as well as the requirements of specific learning content.

Wenhua Wang et al, [2] adapted affinity propagation in MapReduce framework to make semantic retrieval applicable to large-scale data, and with this parallel affinity propagation they proposed an approach to retrieve e-learning materials efficiently, which could retrieve semantically relevant materials utilizing conceptual topics produced in advance.

Anna Katrina Dominguez et al, [3] presented a tool where both past and current student data is used live to generate hints for students who are completing programming exercises during a national programming online tutorial and competition. These hints can be links to notes that are relevant to the problem detected and can include pre-emptive hints to prevent future mistakes.

Samia Azough et al, [4] described an adaptive system conceived in order to generate pedagogical paths which are adapted to the learner profile and to the current formation pedagogical objective.

Mofreh A. Hog et al, [5] classified the learners into specific categories based on the learner’s profiles; the learners’ classes named as regular, workers, casual, bad, and absent. The work extracted the statistical usage patterns that give a clear map describing the data and helping in constructing the e-learning system. The work tries to find the answers of the question how to return the bad students who are away back to be regular ones and find a method to evaluate the e-learners as well as to adapt the content and structure of the e-learning system. The work introduces the application of different fuzzy clustering techniques (FCM and KFCM) to find the learners profiles.

III.TECHNOLOGY OVERVIEW

In this section, we firstly give the details about kmeans algorithm then introduce the concept of mapreduce.

A. Kmeans Clustering Algorithm

K-means algorithm [7] is a widely used partition method in clustering. K-Means [18] is one of the algorithms that solve the well known clustering problem. The algorithm classifies objects to a pre-defined number of clusters, which is given by the user (assume *k* clusters). The idea is to choose random cluster centers, one for each cluster. These centers are preferred to be as far as possible from each other. Starting points affect the clustering process and results. After that, each point will be taken into consideration to calculate similarity with all cluster centers through a distance measure, and it will be assigned to the most similar cluster, the nearest cluster center. When this assignment process is over, a new center will be calculated for each cluster using the points in it. For each cluster, the mean value will be calculated for the coordinates of all the points in that cluster and set as the coordinates of the new center. Once we have these *k* new centroids or center points, the assignment process must start over. As a result of this loop we may notice that the *k* centroids change their locations step by step until no more changes are made. When the centroids do not move any more or no more errors exist in the clusters, we call the clustering has reached a minima. Finally, this algorithm aims at minimizing an objective function, which is in this case a squared error function. The algorithm is expressed in Fig. 1. One drawback of KM is that it is sensitive to the initially selected points, and so it does not always produce the same output. Furthermore, this algorithm does not guarantee to find the global optimum, although it will always terminate. To reduce the effect of randomness, the user can run the algorithm many times before taking an average values for all runs, or at least take the median value.

```
1. Start with k cluster centers (chosen randomly or according to some specific procedure).
2. Assign each record in the data to its nearest cluster center.
3. Re-calculate the cluster centers as the "average" of the records in step 2
4. Repeat, until the cluster centers no longer change.
```

Figure 1. Pseudo-code for K-Means algorithm

One popular way to start KM is to randomly choose k points from data. Initial starting points are important in the clustering process; however, the results mainly depend on the initial means. The standard solution is to try a number of different starting points. Moreover, the results also depend on the metric used to measure distance which is not always easy to implement especially in the high-dimensional space. Additionally, the results depend on the value of k, which in the real world are not always known or determined in advance [20]. Unfortunately, there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k clusters and choose the best one according to a given criterion. However, we need to be careful as increasing k results in smaller error-function values by definition, due to the few number of data points each center will represent, and thus it will lose its generalization ability, as well as increasing the risk of over fitting. In other words centroids do not move any more.

B. Mapreduce

MapReduce is a software framework produced by Google to support parallel computations over large (multi-petabyte) datasets on clusters of commodity nodes. Google MapReduce combines two classes of functions: *map* and *reduce*. [6] These functions are defined with respect to data structured in (key, value) pairs. *Map* takes a pair of data with a type and returns a list of key-value pairs: [6]

map(k1, v1) -> list(k2, v2).

The *map* function is applied in parallel to every item in the input dataset. This produces a list of *(k2, v2)* pairs for each call. The framework collects all pairs with the same key from all lists and groups them together, creating one group for each key *k2*.
The *reduce* function is then applied in parallel to each group, which in turn produces a collection of values:

reduce(k2, Union_List(k2,v2)) ->List (v3).

The returns of all *reduce* functions are collected as the desired result set. A simple example of MapReduce is a program that counts the appearances of each unique word in a set of unstructured documents. In this program, the *map* function will emit <*word, countInDocument*> pairs for each word that occurs in each document. The *reduce* function for each word will receive partial counts from each document, sum up the partial counts, and emit the final result.

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value:
EmitIntermediate(w, "1");

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int result = 0;
for each v in values:
result += ParseInt(v);
Emit(AsString(result));
```

The map function emits each word plus an associated count of occurrences (just `1' in this simple example).The reduce function sums together all counts emitted for a particular word.

Parallelism

Map() functions run in parallel, creating different intermediate values from different input data sets. reduce() functions also run in parallel, each working on a different output key, all values are processed *independently*. Bottleneck: reduce phase can’t start until map phase is completely finished.

Fault Tolerance

Master detects worker failures. Re-executes completed & in-progress map() tasks. Re-executes in-progress reduce() tasks. Master notices particular input key/values cause crashes in map(), and skips those values on reexecution. There are many implementations of mapreduce available. The first java open source implementation of mapreduce is Hadoop.

IV. PROPOSED SYSTEM

Our proposed system is based on the customization of the E-Learning application. Learners performance is evaluated using parallel kmeans clustering based on mapreduce algorithm.

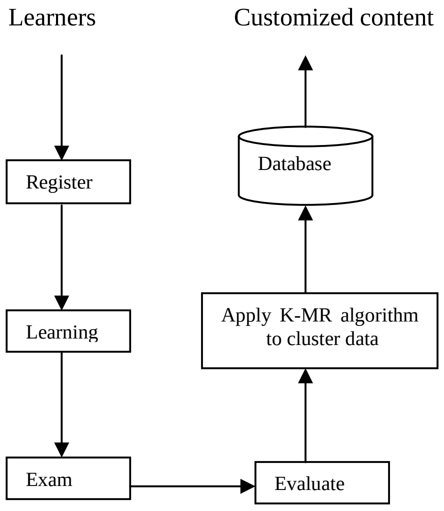

Figure 2. Proposed System

The sample data we have taken to prove that the parallel kmeans clustering algorithm produce the desired clusters with time consuming because of parallel processing using mapreduce.

A. **K-MR algorithm**

As the dataset's scale increases rapidly, it is difficult to use k-means to deal with massive amount of data. A parallel strategy is incorporated into clustering method and a parallel k-means algorithm is proposed. K-MR algorithm partition records among multiple processors [8]. Each processor can read the previous iteration's cluster centers and assign the records on the processor to clusters. Each processor then calculates new centers for its share of records. Each actual cluster center (for the records across all processors) is then the weighted average of the centers on each processor. We need to handle the cluster center information. We create a shared file that has the centroids as calculated for each processor. This file contains the iteration number, cluster id, cluster coordinates, number of records assigned to the cluster. This is the centroid file. An iteration through the algorithm is going to add another set of records to this file. This information is the only information that needs to be communicated globally.

```
1. To simplify the work, K points (K is the number of the clusters we wish to get out of the input dataset) are randomly selected from the input data file as initial cluster centers.
2. Input data set (through a data file) is partitioned into N parts (controlled by the run-time system, not your program). Each part is sent to a mapper.
3. In the map function, the distance between each point and each cluster center is calculated, and each point is labeled with the center index to which the distance is the smallest. Mapper outputs the key-value pairs of label assigned to each point, and the coordinates of the point.
4. All data points of the same current cluster (based on the current cluster centers) are sent to a single reducer. In the reduce function, new cluster center coordinates are easily computed. The output of the reducer is consequently the cluster index and its new coordinates.
5. The new cluster coordinates are compared to the original ones. If the difference is within a preset threshold, then program terminates, and we have found the clusters. If not, use the newly generated cluster centers and repeat step 3 to 5.
```

Figure 3. Pseudo-code for K-MR algorithm

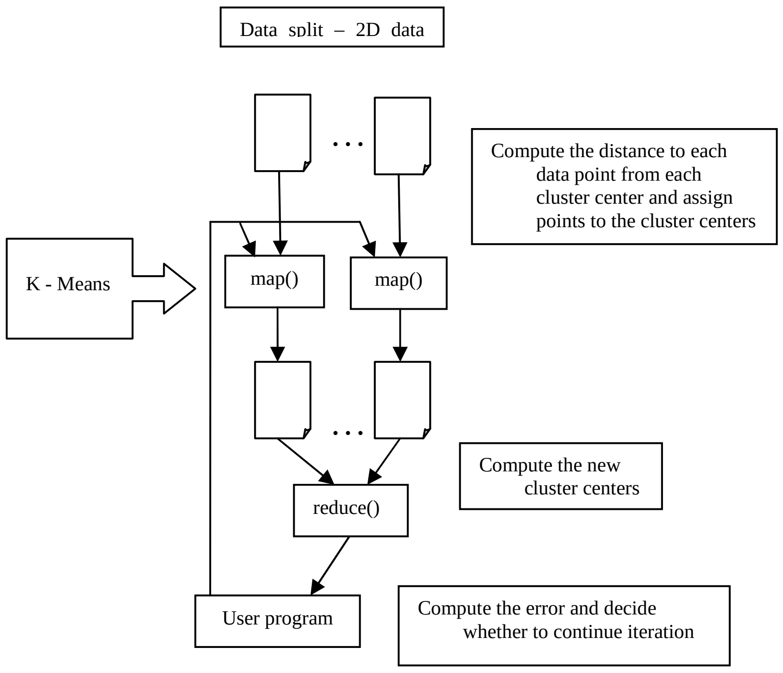
Data split – 2D data
. . .
Compute the distance to each data point from each cluster center and assign points to the cluster centers
K - Means
map()
map()
. . .
Compute the new cluster centers
reduce()
User program
Compute the error and decide whether to continue iteration

Figure 4. K-MR algorithm

K points are randomly selected from the input data file as initial cluster centers. We create a file which contains the information as described in previous slide. This file contains the cluster centers for each iteration. This file is shared among all processors. The Map function reads this file to get the centers from the last finished iteration. It then reads the input records (the data) and calculates the distance (Euclidean Distance) to each center. For each record, it produces an output pair with: Key -- cluster id, Value -- coordinates of records.

Euclidean Distance is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance or simply 'distance' examines the root of square differences between coordinates of a pair of objects.

Formula $$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

In the Combine step the 'map' step produces a lot of data. So we use a Combine function to reduce the size before sending it to Reduce. The Combine function calculates the average of the coordinates for each cluster id, along with the number of records. This is simple, and it produces one record of output for each cluster: – Key is cluster and the Value is number of records and average values of the coordinates. The amount of data now is the number of clusters times the number of processors times the size of the information needed to define each cluster. This is small relative to the data size. The Reduce function calculates the weighted average of its input. Its output is written to the cluster file, and contains: The iteration number; the cluster id; the cluster center coordinates; the size of the cluster. The iteration process can then continue.

V. EXPERIMENTAL RESULT

The data is clustered using 8 processors and the number of cluster is k = 3. We can measure performance of the K-MR algorithm against the performance of the K-Means algorithm by using speedup S(K), where K is the number of processors.

S(K) = Execution time of single processor / Execution time of K processors = O(K/2). The experiment confirms that when N is sufficiently large, speedup gain is O(K/2) as predicted. Let T be the efficiency defined as
T = (speedup gain/maximum speedup)100%
Thus, efficiency of parallel K-MR algorithm T = (K/2)/K x 100% = 50%.

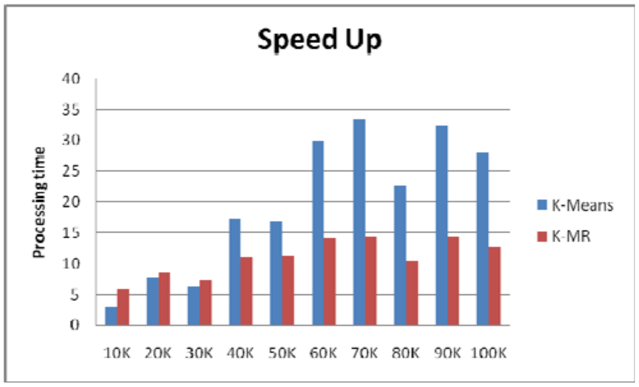

Figure 5. Execution time of K-Means and K-MR algorithm

When the data size is low there is no need of parallelisation whereas when the data is large the parallel processing is necessary such K-MR algorithm to improve the speed and to decrease the execution time. In the above experimental analysis the execution time of k-means is low when the data size is low such 10K, 20K, 30K but the execution time of K-MR is low when the data size is increase such 40K, and so on.

VI. CONCLUSION

In this paper, we have presented the concept of mapreduce and Kmeans algorithm. We have also proposed K-MR algorithm to customize the E-Learning application in such a way that to increase speed and to decrease computation time and to provide customized content to Learners. Since this proposed algorithm uses the parallel processing approach using mapreduce we obtain time efficient and also our experimental result shows the proposed system is efficient comparing to ordinary Kmeans algorithm when the data set is very large. We can conclude that our K-MR algorithm achieves 50% efficiency of time complexity.

REFERENCES

[1] Anane, R. Crowther, S. Beadle, J. Theodoropoulos, G. Sch. (2004): elearning content provision.
[2] Wenhua Wang, Hanwang Zhang, Fei Wu, and Yueting Zhuang. (2008): Large Scale of E-learning Resources Clustering with Parallel Affinity Propagation.
[3] Anna Katrina Dominguez, Kalina Yacef, James R. Curran. (2009): Data Mining for Individualised Hints in eLearning.
[4] Samia Azough ,Mostafa Bellafkih and El Houssine Bouyakhf . (2010): Adaptive E-learning using Genetic Algorithms.
[5] Mofreh A. Hogo. **(**2010**):** Evaluation of e-learning systems based on fuzzy clustering models and statistical tools.
[6] Jeffrey Dean and Sanjay Ghemawat, (OSDI-2004). MapReduce: Simplified Data Processing on Large Clusters,
[7] Kardi Teknomo, (2004): K-Mean Clustering Tutorials, An Online Tutorial.
[8] Gordon S. Linoff. (2008): MapReduce and K-Means Clustering, Blog Entry on Data Miners Blog.
[9] Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet. (2007): MapReduce Theory and implementation, Presentation from Google Code Universit.
[10] Christophe Bisciglia, Aaron Kimball and Sierra Michels-Slettvet. (2007): Clustering Algorithms, Presentation from Google Code University.
[11] Tushar Deshpande,Tejas Vora. (2009): Web Mining – II Parallelizing K-means Clustering with MapReduce.
[12] Zhao, Ma, “Parallel K-Means Clustering Based on MapReduce”.
[13] Geoffrey Fox, 2010 Deterministic Annealing: Oct Trees and High Dimension.
[14] J. MacQueen, “Some Methods For Classification And Analysis Of Multivariate Observations,” In proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1967
[15] J. Pena, J. Lozano, and P. Larranaga, “An Empirical Comparison of Four Initialization Methods For The K-Means Algorithm,” Pattern Recognition Letters, Vol. 20 No. 10, 1999.

Authors

Mr.R.Raju pursuing PhD and received the M.Tech Degree in Computer Science and Engineering from Pondicherry University, Puducherry. His current research involves in Service Oriented Architecture and Software Engineering. Presently working with Sri Manakula Vinayagar Engineering College (Affiliated to Pondicherry University, Puducherry) as Assistant Professor in the Department of Information Technology.

Mrs.V. Vijayalakshmi II year M.tech student in the project phase in the Department of computer Science and Engineering in Sri Manakula Vinayagar Engineering College. Pondicherry University, Puducherry. She holds a MCA can be reached via vivenan09 @gmail.com.

Mrs.R.T. Showmya II year M.tech student in the project phase in the Department of computer Science and Engineering in Sri Manakula Vinayagar Engineering College. Pondicherry University, Puducherry. She holds a B.Tech can be reached via showmya.rt @gmail.com.