Технічні науки

УДК 004.75

Yaremenko V.
student
National technical university of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

Яременко Вадим Сергійович
студент
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Яременко Вадим Сергеевич
студент
Национальный технический университет Украины
«Киевский политехнический институт им. Игоря Сикорского»

DISTRIBUTED DATA CLUSTERING CURE ALGORITHM APPROBATION USING HADOOP MAPREDUCE

АПРОБАЦІЯ РОЗПОДІЛЕНОГО АЛГОРИТМУ КЛАСТЕРИЗАЦІЇ CURE З ВИКОРИСТАННЯМ HADOOP MAPREDUCE

АППРОБАЦИЯ РАСПРЕДЕЛЕННОГО АЛГОРИТМА КЛАСТЕРИЗАЦИИ CURE С ИСПОЛЬЗОВАНИЕМ HADOOP MAPREDUCE

Summary: In this article the modification of clustering CURE algorithm for distributed calculations and results of its work in the Hadoop MapReduce system are described.

Keywords: Data mining, Distributed clustering, CURE, Hadoop, MapReduce.

Анотація: У даній статті описано модифікацію алгоритму кластеризації CURE для роботи у розподіленій системі, а також результати роботи цього алгоритму, отримані з використанням засобів Hadoop MapReduce.

Ключові слова: Інтелектуальний аналіз даних, Розподілена кластеризація, CURE, Hadoop, MapReduce.

Аннотация: В данной статье описано модификацию алгоритма кластеризации CURE для работы в распределенной системе, а также результаты работы этого алгоритма, полученные с использованием средств Hadoop MapReduce.

Ключевые слова: Интеллектуальный анализ данных, Распределенная кластеризация, CURE, Hadoop, MapReduce.

1. Introduction

With the increase in the amount of information, it is necessary to develop algorithms for its fast and efficient processing. Parallel clustering algorithms and implementation techniques are the key to meeting the scalability and performance requirements entailed in such scientific data analyses. So far, there are some parallel clustering algorithms, but all of them have following drawbacks: a) They assume that all objects can reside in main memory at the same time; b) Their parallel systems have provided restricted programming models and used the restrictions to parallelize the computation utomatically. Both assumptions areprohibitive for very large datasets with millions of objects. Therefore, dataset oriented parallel clustering algorithms should be developed [1, p.675]. One of such approaches is described in this article.

2. Parallel CURE algorithm based on MapReduce

In this section described the information about general CURE algorithm design and parallel MapReduce CURE algorithm design. First part is needed to present parallel parts of CURE algorithm, that could be implemented as map and reduce operations.

2.1. CURE algorithm

CURE (Clustering Using REpresentatives) algorithm begins from taking a small sample of the data and cluster it in main memory. In principle, any clustering method could be used, but as CURE is designed to handle oddly shaped clusters, it is often advisable to use a hierarchical method in which clusters are merged when they have a close pair of points.

During the second step it is necessary to Select a small set of points from each cluster to be representative points. These points should be chosen to be as far from one another as possible.

Then each of the representative points should be moved a fixed fraction of the distance between its location and the centroid of its cluster. Perhaps 20% is a good fraction to choose. Note that this step requires a Euclidean space, since otherwise, there might not be any notion of a line between two points [2, p.263].

The next phase of CURE is to merge two clusters if they have a pair of representative points, one from each cluster, that are sufficiently close. The user may pick the distance that defines "close." This merging step can repeat, until there are no more sufficiently close clusters [2, p.264].

So, after the analysis of above steps I understood that such operations could be performed separately in the distributed system, no need to have the whole dataset on one PC. Even more at the one moment of time it is enough to have only the description of already known clusters (representative points and centroid) and new point from the dataset. In the next part the details of such approach are described.

2.2. CURE algorithm based on MapReduce

The general model consists of *N* nodes that run Map program and 1 node that runs a Reduce program. Chunks from datasets are input to the map program.

One chunk is a description of one point (e.g. in Cartesian coordinate system it is an array with coordinates). Map program finishes its work after the full dataset on this node has being processed.

The output of map program is a set of key-value pairs, where the key is equals to 1 and the value is a set of representative points from *cluster(i,j)*. In this case *i* – it is a number of the node and *j* – a cluster number from the ith node. For example, if on the node *2* there are 3 clusters of data, the output should be following: key *1* value *cluster(2,1)*; key *1 value cluster(2,2)*; key *1* value *cluster(2,3)*.

The reduce program gets the output of the map program. The main task of reduce program is to merge clusters produced from all datasets. After merging is done, the reduce program produces the resulting dataset which contains a description of each cluster: centroid and representative points [3, p.204].

3. Experimental results

For the experiment two programs were developed using the Java language. The first program has one thread and could be launched on each PC with pre-installed Java virtual machine and the second program is written for Hadoop MapReduce distributed system. Algorithms of both programs are described in the part 2 of this article.

As I don't have the direct access to a real distributed system, I launched above programs on my local PC with such characteristics: Intel® Core^{TM} i7-2630QM 2GHz, 4 GB DDR3 RAM, Ubuntu OS. Hadoop MapReduce system was configured as a single-node cluster according to the official tutorial [4].

For the dataset generating one more program was developed. Input of this program is size of future dataset, dimensions, number of clusters, size of field (x-axis, y-axis borders). Output of this program is a file with floating-points values – Cartesian coordinates of each point. For this experiment such parameters were used: 2-dimensional coordinates, 100 MB, 250 MB, 500 MB, 1000 MB, 2000 MB, 4000 MB, 6000 MB and various number of clusters for each dataset.

Plot with received results could be found on the figure 1 below. With increasing of dataset size the efficiency of distributed CURE algorithm implementation is clearly visible. But in the future it is necessary to get results from the real environment, not from the local PC. Other parameters are following: field borders are [0; 500] for x-axis and [0; 500] for y-axis, number of clusters is 80, number of representatives points in cluster is 20, critical distance (to merge clusters) is 20, move fraction is 20%.

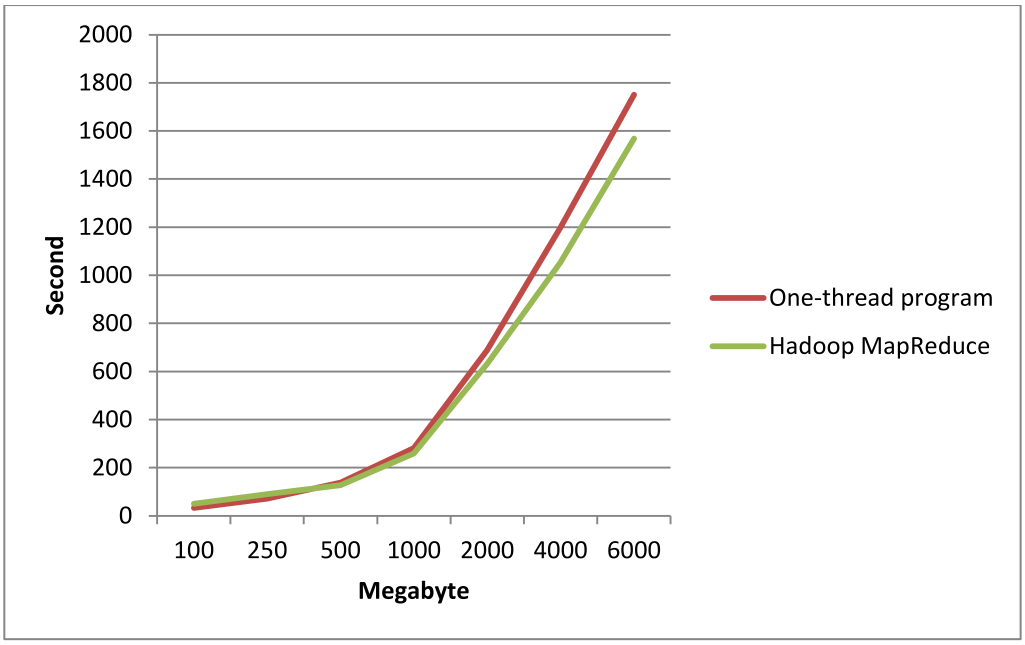

Fig. 1. Running time in comparison with dataset size [author work]

Also, it is necessary to mention that such results differ for different datasets. For example, in case of just one cluster programs run faster, because of small amount of points to process, otherwise a big amount of comparison operations will be performed between representative points of clusters. That's why a lot of experiment's details are described above.

4. Conclusion

In this article was described an approach of the distributed CURE algorithm implementation. The advantage of the described approach is in distributed calculations on each node separately. Also, it is not necessary to load a full dataset into the RAM because map task processes a chunk of data in a correct way.

After the description of distributed CURE algorithm's there are presented details of efficiency of the proposed approach. As seen on the figure 1, the efficiency of the algorithm is especially noticeable with a growth of dataset size. The efficiency of the distributed implementation is seen on the dataset larger than 400 MB. Unfortunately, there is one disadvantage of received results: there were no ability to test Hadoop MapReduce program in the real distributed environment, so presented values have been got from the personal computer.

Reference:

1. W. Zhao, H. Ma, Q. He (2009). Parallel K-Means Clustering Based on MapReduce. CloudCom 2009, LNCS 5931, pp. 674-679.
2. J. Leskovec, A. Rajaraman, J. D. Ullman (2014). Mining Of Massive Datasets, Second Edition. Cambridge University Press. ISBN-13: 978-1107077232. Print.
3. V. Yaremenko (2017). An approach for data clustering CURE algorithm implementation using the MapReduce technology. System Analysis and Information Technologies. 19-th International Conference SAIT 2017 Kyiv, Ukraine. ISBN 978-966-2748-94-2. Print.
4. Hadoop: Setting up a Single Node Cluster. Access date: 06 May 2017. Access link: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html.