2011 Workshops of International Conference on Advanced Information Networking and Applications

K-means clustering in the cloud - a Mahout test

Rui Máximo Esteves *
rui.esteves@uis.no

Chunming Rong*
chunming.rong@uis.no

Rui Pais *
rma.pais@stud.uis.no

* Department of Electrical and Computer Engineering, University of Stavanger, Norway

Abstract*—. The K-Means is a well known clustering algorithm that has been successfully applied to a wide variety of problems. However, its application has usually been restricted to small datasets. Mahout is a cloud computing approach to K-Means that runs on a Hadoop system. Both Mahout and Hadoop are free and open source. Due to their inexpensive and scalable characteristics, these platforms can be a promising technology to solve data intensive problems which were not trivial in the past. In this work we studied the performance of Mahout using a large data set. The tests were running on Amazon EC2 instances and allowed to compare the gain in runtime when running on a multi node cluster. This paper presents some results of ongoing research.

Keywords- K-means, cloud computing, mahout, map reduce

I. INTRODUCTION

The K-means is one of the most frequently used clustering algorithms. It is simple and straightforward and has been successfully applied during the last few years. Under the assumption that datasets tend to be small, research on clustering algorithms has traditionally focused on improving the quality of clustering. However, many datasets now are large and cannot fit into main memory [1].

Mahout/Hadoop can be a promising and inexpensive solution to solve problems with large data sets. However, there is a lack of studies regarding its performance. With this work we wanted to study the gain in runtime when using this solution and also to test if there was any loss in the clusters quality. To meet those aims we used Amazon EC2 instances and set up a small cluster. We chose to study K-means since it is one of the most popular and easy algorithms.

The rest of this paper is organized in the following way: section II presents some theoretical foundations. The experiments are described in section III and section IV the results, interpretation and discussion are presented. Future work is in section V and the final conclusions are in section VI.

II. THEORY

Clustering

Clustering refers to the process of grouping samples into different classes based on their similarities. Samples within a class have high similarity in comparison to one another but are very dissimilar to samples in other classes. These groups or classes are called clusters. Clustering algorithms can divide the samples into clusters automatically without any preconceptions about what kind of groupings should be found. Within the context of machine learning, clustering is considered to be a form of unsupervised learning since there is no target variable to guide the learning process. Unlike classification, clustering and unsupervised learning do not rely on predefined classes and class-labeled training examples. For this reason, clustering is a form of learning by observation, rather than learning by examples [2]. The number of clusters, size, and shape are not in general known in anticipation, and each of these parameters must be determined by either the user or the clustering algorithm [3].

Clustering has been used in many areas, including data mining, statistics, biology, and machine learning. Clustering can also be used for outlier detection, where outliers (values that are “far away” from any cluster) may be more interesting than common cases [2]. One useful application example is fault detection [4, 5]. Other examples of applications can be found in [2, 6]. The concept of clustering is not particularly new, but it is still an important topic of research. A useful survey about recent advances can be found in [7].

Cluster algorithms types

The two major types of cluster algorithms are hierarchical and partitional.

The first type produces a hierarchical decomposition of the data set into partitions. It merges or splits the partitions into clusters using a similarity criterion.

In the second type, an algorithm creates an initial division of the data set, according to a given number of partitions. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion for partitioning is a combination of high similarity of the samples inside of clusters with high dissimilarity between distinct clusters. This means that samples in the same cluster are “close” or

978-0-7695-4338-3/11 $26.00 © 2011 IEEE
DOI 10.1109/WAINA.2011.136

related to each other, whereas samples of different clusters are "far apart" or very different. [2, 6]

Each clustering type is better suited than to a particular type of problem. Abbas [8] concluded that partitional algorithms are suited for large data set while hierarchical are for small data sets. Singla, Yadav and Sing [9] presented some arguments in favor of partitional algorithm. According to them, there is no need to determine the full dendrogram with a partitional algorithm. They also found that the time complexity of partitonal algorithm is little compared with hierarchical. Zhao and Kaprypis [10] had similar results. On the other end, hierarchical algorithms have the advantages of an easily interpreted graphical representation and ignorance of influence from initialization and local minima [11].

K-means

K-means clustering is a widely used partition algorithm. It partitions the samples into clusters by minimizing a measure between the samples and the centroid of the clusters. Euclidean distance is a similarity criterion extensively used for samples in Euclidean space. Manhattan distance is a simpler measure also for Euclidean space. Cosine distance and Jaccard is often employed for documents clustering. [12]

The algorithm for the standard K-means is given as follows [13]:

1. Choose a number of clusters k
2. Initialize centroid[1] c1,… ck
3. For each data point, compute the centroid it is closest to (using some distance measure) and assign the data point to this cluster.
4. Re-compute centroids (mean of data points in cluster)
5. Stop when there are no new re-assignments.

The K-means clustering is simple but it has high time complexity when the data sets are large. In these circumstances the memory of a single machine can be a restriction. As a consequence it has not been used in the past with large data sets [13].

Hadoop/Mahout solution

Hadoop[2] is a software framework which allows the effortless development of cloud computing systems. It supports data intensive distributed applications on large clusters built of commodity hardware. This framework is designed to be scalable, which allows the user to add more nodes as the application requires. Hadoop uses a distributed computing paradigm named MapReduce.

The MapReduce programming model consists of two user defined functions: map and reduce, specified on a job. The job usually splits the input dataset into independent blocks which are processed by the map tasks in a parallel way. Hadoop sorts the outputs of the maps, which are then the input to be processed by the reduce tasks [14]. Hadoop uses a distributed file system HDFS, which creates multiple replicas of data blocks and distributes them on computer nodes throughout a cluster to enable reliable, extremely rapid computations [15].

Mahout[3] is a machine learning library that runs over a Hadoop system. It has a collection of algorithms to solve clustering, classification and prediction problems. It uses MapReduce paradigm which in combination with Hadoop can be used as an inexpensive solution to solve machine learning problems.

Mahout is a project still in development which has been used mainly for recommendation engines, document classifiers and to solve other web typical problems. At the moment, its usage for clustering is not sufficiently explored.

Hadoop and Mahout are free and open source projects. Due to their inexpensive and scalable characteristics, these platforms can be a promising technology to solve data intensive problems which were not trivial in the past.

However, it is important to study the tradeoff between the overhead of using Map/Reduce and the gain of performance by distributing the computation. It is also essential to check if the clusters maintain their quality.

Mahout contains various implementations of clustering, like K-means, fuzzy K-means, meanshift and Dirichlet among others. To input the data for Mahout clustering it is necessary to do some procedures first. If the data is not numerical it has to be first preprocessed. It is required then to create vectors. If the data set is sparse it allows the user to create sparse vectors that are much more compact. The vectors are finally converted to a specific Hadoop file format that is *SequenceFile*.

The K-means clustering algorithm takes the following input parameters:

- A *SequenceFile* containing the input vectors.
- A *SequenceFile* containing the initial cluster centers. If not present it will attribute them randomly.
- A similarity measure to be used.
- The convergence Threshold that will be the stopping condition of the K-means. If in a particular iteration, any centers of the clusters do not change beyond that threshold, then no further iterations are done.
- The maximum number of iterations to be processed if it does not converge first.
- The number of reducers to be used. This value determines the parallelism of the execution.
- The vector implementation used for the input files.

As output the user gets the centroids coordinates and the samples attributed to each cluster. The output files are in *SequenceFile* format. Mahout provides the necessary tools for file conversion and creating the vectors [16].

There are three stages in a K-means job (figure 1):

- Initial stage: the segmentation of the dataset into HDFS blocks; their replication and transfer to other machines; according to the number of blocks and cluster configuration it will be assigned and distributed the necessary tasks.

[1] Centroid refers to the center of the cluster.

[2] http://hadoop.apache.org/

[3] http://mahout.apache.org/

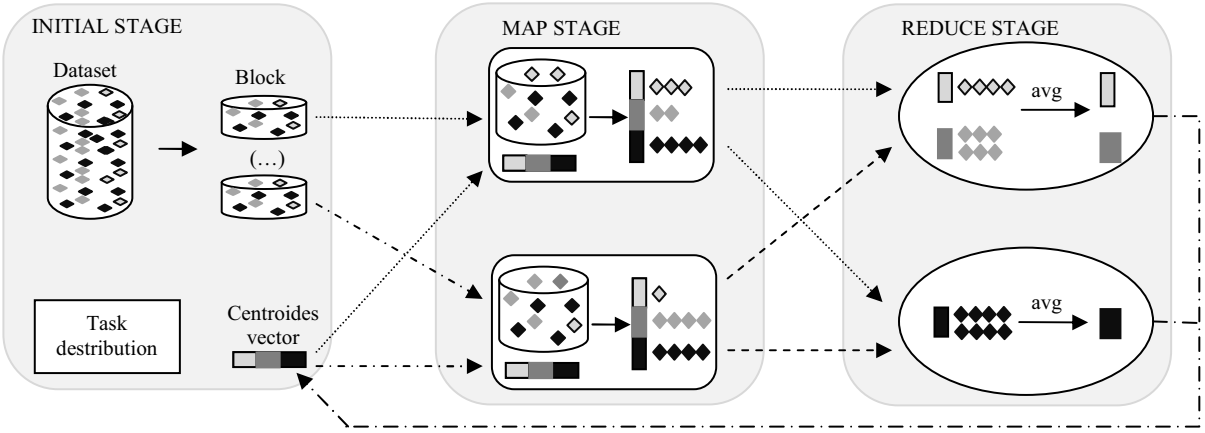

Figure 1 – The three stages of a K-means job.

- Map stage: calculates the distances between samples and centroids; match samples with the nearest centroid and assigns them to that specific cluster. Each map task is processed with a data block.

Reduce stage: recalculates the centroid point using the average of the coordinates of all the points in that cluster. The associated points are averaged out to produce the new location of the centroid. The controids configuration is feedback into the Mappers. The loop ends when the centroids converge.

After finishing the job Mahout provides the K-means time, the centroids and the clustered samples.

Quality of clustering

An important problem in clustering is how to decide what the best set of clusters is for a given data set, in terms of both the number of clusters and the membership of those clusters [17].

One simple inter-cluster quality measure was used in this work: *the sum of the distances squared between centroids* (Q_1) [18].

$$Q_1 = \sum_{1 \le i < j < k} d^2(c_i, c_j) \qquad \text{(eq. 1)}$$

where: c_i is the centroid of a cluster c_i;
d is the distance metric;
and k is the number of clusters.

In our study, we do not aim to reach the optimal cluster configuration. Instead, we want to check if the quality of the Mahout K-means is maintained between the tests. As so, we chosen this measure since it is not computationally expensive and easy to implement.

Related work

At this date we could not find any intensive tests of Mahout in the literature. However there are two interesting works with scaling tools for data mining using MapReduce [19, 20].

III. EXPERIMENTS

The dataset

To evaluate the performance of multi node versus single node we used a dataset from the 1999 kdd cup[4]. This dataset was gathered by Lincoln Labs: nine weeks of raw TCP dump data were collected from a typical US air force LAN. During the use of the LAN, several attacks were performed on it. The raw packet data were then aggregated into connection data. Per record, extra features were derived, based on domain knowledge about network attacks. There are 38 different attack types, belonging to 4 main categories. It has 4,940,000 records, with 41 attributes and 1 label (converted to numerical. A 1.1 GB dataset was used. This file was randomly segmented into smaller files. For the data preprocessing we developed a Java software.

The Machines

We ran our tests using Amazon EC2 m1.large instances with the following characteristics: 7.5 GB memory; 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each); 850 GB of storage; 64-bit platform; high I/O Performance.

All the instances were launched using the same Amazon Machine Image (AMI). We adapted an existing Cloudera AMI[5] to our specific needs and stored on Amazon S3. Our AMI uses Mahout 0.4 and Hadoop 0.20.2+237 and we added scripts for automatic cluster configuration. This way the user just needs to launch EC2 instances with this AMI, run a script on the master node and the cluster became ready for use. The HDFS configuration used was the default one, where the block size is 64 MB and the replication factor 3.

Jobs

To study how the performance changes with the scalability, K-means jobs were setup to guaranty that the algorithm will converge. K-means can produce different results depending on the position of the initial centroids. To

[4] http://www.inf.ed.ac.uk/teaching/courses/dme/html/datasets0405.html
[5] cloudera-hadoop-ubuntu-20090623_x86_64

avoid this issue, the same initial centroids were used for all experiments. Each test was repeated 3 times to ensure the results were not affected by Amazon fluctuations.

The Mahout K-means algorithm was executed using the following parameters: Euclidean distance as the similarity measure on experiments A and B; Squared Euclidean on C. All experiments were set with 150 maximum interactions, and 11 clusters.

To guarantee that the quality of the Mahout K-means was maintained the equation 1 was applied.

IV. RESULTS AND INTERPRETATION

A – How does Mahout scale with the file size?

It can be observed on table 1 that a 5 multi node has worse performance than the single one for small datasets. With a 66 MB file (6%) the distribution process is not compensated by the overhead of the initial step. Given that the HDFS block size is 64 MB, only two tasks will be executed. Since each machine is a dual core there is no gain expected for a file smaller than 128 MB. The gain is already present for a file slightly bigger than 128 MB (12% scenario). The overhead is largely compensated as soon as the dataset grows. The gain in performance reaches 351% for a 1.1 GB file.

Data File (%)	kmeans MN (sec)	kmeans SN (sec)	Iterations to converge	Gain (%)
6	43.9	41.3	72	-6%
12	45.3	52.8	78	16%
25	46.4	88.5	72	91%
50	48.6	149.1	71	207%
100	70.3	316.7	56	351%

Table 1. Times per iteration and gain with the file size. MN - 5 Multi node. SN - Single node.

Figure 2 shows that when the data file is increased, the 5 node time enlarges very slowly. This seems to indicate that the initial step has a considerable footprint. For this specific dataset and set of fixed initial centroids, the K-means needed fewer interactions to converge as the file became bigger. Since the k is the same this is an expected behavior. Given that there are more samples per cluster, the averages of centroids are more stable and the algorithm converges with less interactions. However, each iteration is more computer demanding. On the other side, more iterations means more communication between the mappers and the reducers. On a cluster this implies more network usage, which can signify a bottleneck.

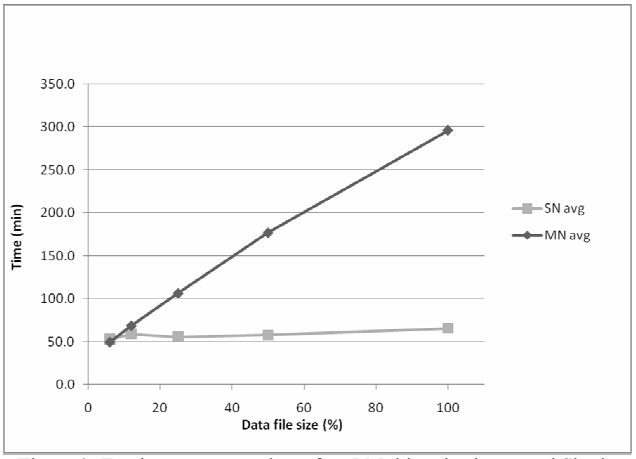

Figure 2. Total convergence times for: 5 Multi node cluster and Single node. File size: 1.1 GB.

B – How does Mahout scale with the number of nodes?

Mahout decreases the clustering time as the number of nodes increases. For 20 nodes it has reduced to less than one fifth. In figure 3, it can be noticed that the cluster time diminishes with the number of nodes in a non linear way.

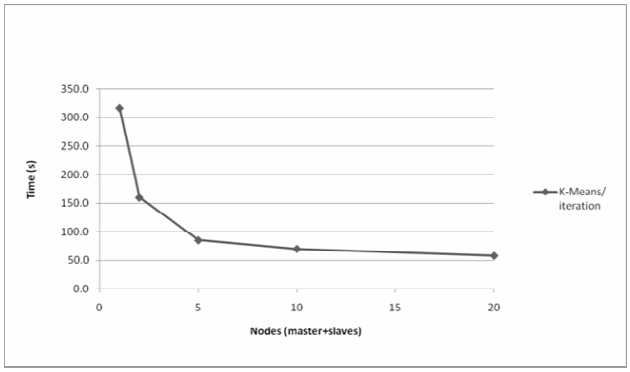

Figure 3. Runtimes per iteration with the number of nodes. File size: 1.1 GB.

One can verify on table 2 that the gain is almost 100% between 1 and 2 nodes. However, it increases only 444% between 1 and 20 nodes. This happens due to the size of the dataset, which seems to be too small for a 20 nodes. The file is divided into 18 blocks. It was noticed 2 idle machines in the 20 nodes cluster. For this dataset a possible improvement could be forcing a higher number of tasks.

It is interesting to notice that when the number of nodes increases, the CPU usage diminishes and the network usage enlarges. This can be clearly seen comparing figure 4, 5 and 6, 7. Those graphics were obtained from Amazon Cloudwatch and correspond to the monitoring of a slave node. They compare the situation between a 5 and a 10 nodes cluster executing K-means with the full dataset. The CPU idle time on the 10 nodes scenario is noticeable. It indicates that the instance was waiting for the results from the other nodes. The network was continually used around

84 Mbps in this scenario while it was interrupted on the 5 nodes. The existence of extremely high peaks was present when the cluster was not executing a job. As so, they should be related to Amazon and not to the experiment itself. Both graphs show how intensive was the network usage. They also point towards a saturation of the network usage.

Nodes	K-means (min)	Gain (%)
1	296	
2	149	98%
5	80	271%
10	66	351%
20	54	444%

Table 2. Converging times varying the number of nodes. File size: 1.1 GB. Gain comparing with 1 node.

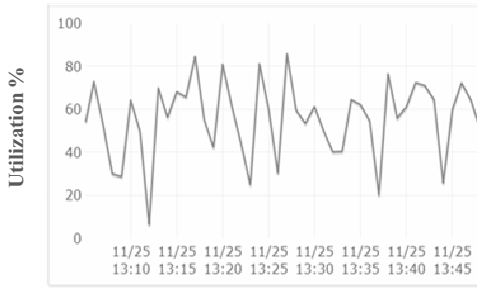

Figure 4. CPU utilization per slave node. xx axis represents the time when the experiment occured.5 nodes. File size: 1.1 GB.

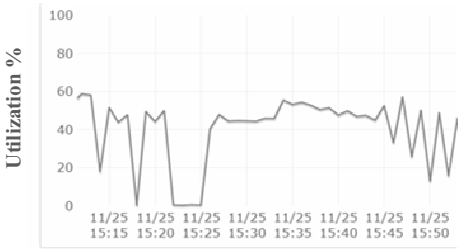

Figure 5. CPU utilization per slave node. 10 nodes. File size: 1.1 GB.

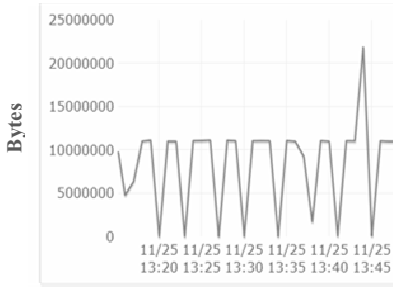

Figure 6. Network usage per node. 5 nodes. File size: 1.1 GB.

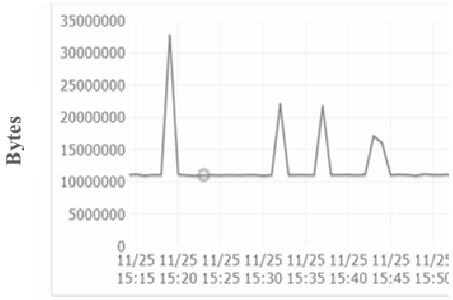

Figure 7. Network usage per node. 10 nodes. File size: 1.1 GB.

C – How is Mahout influenced by the distance measure?

The Euclidean Squared distance metric uses the same equation as the Euclidean distance metric, but does not take the square root. As a result, clustering with the Euclidean Squared distance metric should be considerably faster than clustering with the regular Euclidean distance. However, the experiment shows just a small improvement.

Distance measure	K-means (min)	Iteractions
Euclidean	80	56
Squared Euclidean	79	56

Table 3. Converging times using different measures. Cluster size 5. File size: 1.1 GB.

As K-means in table 3 show the squared Euclidean reduces only 1 minute to 80. The difference between the distances would reduce each map and reducer times. This evidence together with others presented in this work suggests that the first phase and the communication between the machines can be relevant for the performance of Mahout. It also implies that the reducer may have a significant impact on each iteration time.

V. FUTURE WORK

In the future the following influences on Mahout K-means will be studied:

- bigger datasets;
- increase in the number of tasks;
- increase in the number of reducers;
- one machine with n cores vs n machines with 1 core.

VI. CONCLUSIONS

In this paper we tested how Mahout K-means scale. For that purpose we have conducted several experiments

varying the dataset size and the number of nodes. The influence of two different distance measures was also tested.

We concluded that Mahout can be a promising tool for K-means clustering. It allows the clustering of large datasets, is scalable and it produces significant gains in performance.

Using Mahout with small files is not always the best option. There is an overhead due to replication and distribution of the data blocks. The overhead is largely compensated as soon as the dataset grows.

Increasing the number of nodes reduces the execution times. However, for small files it can lead to an underutilization of each machine's resources.

The converging times do not depend largely on the distance measure.

As expected the quality of Mahout K-means clustering was consistent between the several experiments.

REFERENCES

[1] Fredrik Farnstrom, J.L.a.C.E.: 'Scalability for Clustering Algorithms Revisited', SIGKDD Explorations, 2000, 2, pp. 51-57

[2] Jiawei Han, M.K., Jian Pei: 'Data Mining: Concepts and Techniques' (Elsevier, 2006, 2nd edn. 2006)

[3] Osei-Bryson, K.-M.: 'Assessing Cluster Quality Using Multiple Measures - A Decision Tree Based Approach in The Next Wave in Computing', The Next Wave in Computing, Optimization, and Decision Technologies, 2005 29, (VI), pp. 371-384

[4] Kun Niu, C.H., Shubo Zhang, and Junliang Chen: 'ODDC: Outlier Detection Using Distance Distribution Clustering'. Proc. PAKDD 2007 Workshops, 2007 2007 pp. Pages

[5] Austin, V.J.H.a.J.: 'A Survey of Outlier Detection Methodologies', Artificial Intelligence Review, 2004, 22, pp. 85-126

[6] A.K. JAIN, M.N.M.a.P.J.F.: 'Data Clustering: A Review', ACM Computing Surveys, 1999, 31, No. 3, September

[7] S. B. Kotsiantis , P.E.P.: 'Recent Advances in Clustering: A Brief Survey', WSEAS Transactions on Information Science and Applications, 2004, 1, pp. 73 - 81

[8] Abbas, O.A.: 'Comparisons Between Data Clustering Algorithms', The International Arab Journal of Information Technology, Vol 5., No. 3, July 2008, 2008, 5, (3), pp. 320-325

[9] Singla, B., Yadav, K., and Singh, J.: 'Comparison and analysis of clustering techniques', in Editor (Ed.)^(Eds.): 'Book Comparison and analysis of clustering techniques' (2008, edn.), pp. 1-3

[10] YING ZHAO, G.K.: 'Hierarchical Clustering Algorithms for Document Datasets', Data Mining and Knowledge Discovery, 2005, 10, pp. 141-168

[11] Frigui, H., and Krishnapuram, R.: 'Clustering by competitive agglomeration', Pattern Recognition, 1997, 30, (7), pp. 1109-1119

[12] Pang-Ning Tan, M.S.a.V.K.: 'Introduction to Data Mining ' (Addison-Wesley Longman Publishing Co., Inc., 2005, 1st edn. 2005)

[13] C. IMMACULATE MARY, D.S.V.K.R.: 'Refinement of clusters from k-means with ant colony optimization', Journal of Theoretical and Applied Information Technology 2009, 9. No. 2

[14] Dean, J., and Ghemawat, S.: 'MapReduce: simplified data processing on large clusters', Commun. ACM, 2008, 51, (1), pp. 107-113

[15] White, T.: 'Hadoop: The Definitive Guide' (O'Reilly Media, 2009)

[16] Sean Owen, R.A., Ted Dunning and Ellen Friedman: 'Mahout in Action' (Manning Publications, 2010. 2010)

[17] Raskutti, B., and Leckie, C.: 'An Evaluation of Criteria for Measuring the Quality of Clusters'. Proc. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence1999 pp. Pages

[18] Rayward-Smith, Q.H.N.a.V.J.: 'Internal quality measures for clustering in metric space', Int. J. Business Intelligence and Data Mining 2008, 3, (1)

[19] Shang, W., Adams, B., and Hassan, A.E.: 'An experience report on scaling tools for mining software repositories using MapReduce'. Proc. Proceedings of the IEEE/ACM international conference on Automated software engineering, Antwerp, Belgium pp. Pages

[20] Cheng T. Chu, S.K.K., Yi A. Lin, Yuanyuan Yu, Gary R. Bradski, Andrew Y. Ng, Kunle Olukotun: 'Map-Reduce for Machine Learning on Multicore', NIPS, 2006, pp. 281-288