Large data computing using Clustering algorithms based on Hadoop

Samrudhi Tabhane, Prof. R.A.Fadnavis

Dept. Of Information Technology,YCCE, email id: sstabhane@gmail.com

Abstract— The Hadoop Distributed File System (HDFS) is designed to store large data sets reliably and to stream those data sets at high bandwidth to user applications. In a large cluster, thousands of servers and host are directly attached and execute user application tasks. By distributing storage and computation across many servers, the resource can grow with demand while remaining economical at every size. Hadoop is a popular opensource implementation of MapReduce for the analysis of large datasets. To manage storage resources across the cluster, Hadoop uses a distributed user-level file system. This paper analyzes the performance of two major clustering algorithms K-means and DBSCAN on Hadoop platform and uncovers several performance issues. The experimental result demonstrates that K-means clustering algorithm is more efficient than DBSCAN algorithm based on MapReduce. Experimental results also show that DBSCAN algorithm based on MapReduce alleviates the problem of time delay caused by large data sets.

Keywords— MapReduce, Hadoop, Clustering, K-means, DBSCAN, HDFS

INTRODUCTION

Apache Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. Hadoop was derived from Google's MapReduce and Google File System (GFS) papers. The Hadoop framework transparently provides both reliability and data motion to applications. Hadoop implements a computational paradigm named MapReduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides a distributed file system that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both map/reduce and the distributed file system are designed so that node failures are automatically handled by the framework.

In a larger cluster[1] the HDFS is managed through a dedicated NameNode server that hosts the filesystem index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, so preventing filesystem corruption and reducing loss of data. Similarly, job scheduling can be managed by a standalone JobTracker server. In clusters where the Hadoop MapReduce engine is deployed against an alternate filesystem, the NameNode, secondary NameNode and DataNode architecture of HDFS is replaced by the filesystem-specific equivalent In a Hadoop cluster, data is distributed to all the nodes of the cluster as it is being loaded in. The Hadoop Distributed File System (HDFS) will split large data files into chunks which are managed by different nodes in the cluster. In addition to this each chunk is replicated across several machines, so that a single machine failure does not result in any data being unavailable. An active monitoring system then re-replicates the data in response to system failures which can result in partial storage. Even though the file chunks are replicated and distributed across several machines, they form a single namespace, so their contents are universally accessible. Data is conceptually record-oriented in the Hadoop programming framework. Individual input files are broken into lines or into other formats specific to the application logic. Each process running on a node in the cluster then processes a subset of these records. The Hadoop framework then schedules these processes in proximity to the location of data/records using knowledge from the distributed file system. Since files are spread across the distributed file system as chunks, each compute process running on a node operates on a subset of the data. Which data operated on by a node is chosen based on its locality to the node: most data is read from the local disk straight into the CPU, alleviating strain on network bandwidth and preventing unnecessary network transfers. This strategy of moving computation to the data, instead of moving the data to the computation allows Hadoop to achieve high data locality which in turn results in high performance.

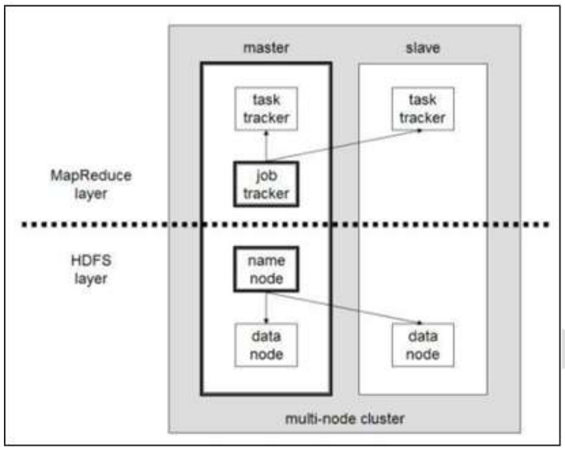


Fig. 1 HDFS setup

MAPREDUCE PROGRAMMING MODEL

MapReduce Programming Model MapReduce is a software framework proposed by Google, which is a basis computational model of current cloud computing platform. Its main function is to handle massive amounts of data. Because of its simplicity, MapReduce can effectively deal with machine failures and easily expand the number of system nodes. MapReduce provides a distributed approach to process massive data distributed on a large -scale computer clusters. The input data is stored in the distributed file system (HDFS), MapReduce adopts a divide and conquer method to evenly divide the inputted large data sets into small data sets, and then process on different node, which has achieved parallelism. In the MapReduce programming model[2] data is seen as a series of key value pairs like,the workflow of MapReduce consists of three phases: Map, Shuffle, and Reduce. Users simply write map and reduce functions. In the Map phase, a map task corresponds to a node in the cluster, as the other word, multiple map tasks are be running in parallel at the same time in a cluster. Each map call is given a key-value pair (k1, v1) and produces a list of (k2, v2) pairs. The output of the map calls is transferred to the reduce nodes (shuffle phase). All the intermediate records with the same intermediate key (k2) are sent to the same reducer node. At each reduce node, the received intermediate records are sorted and grouped (all the intermediate records with the same key form a single group). Each group is processed in a single reduce call. The data processing can be summarized as follows:
Map (k1, v1) −→ list (k2, v2)
Reduce (k2,list(v2))−→list(k3,v3)

K-MEANS CLUSTERING ALGORITHM

K-Means is a simple learning algorithm for clustering analysis. The goal of K-Means algorithm is to find the best division of n entities in k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized the k-means algorithm is used for partitioning where each cluster's Centre is represented by the mean value of the objects in the cluster [3] K-means Pseudo code:

1. Begin with n clusters, each containing one object and we will number the clusters 1 through n.

2. Compute the between-cluster distance D(r, s) between-object distance of the two objects in r and s respectively, r, s =1, 2… n. Let the square matrix D = (D(r, s)). If the objects are represented by vectors, we can use the Euclidean distance.

3. Next, find the most similar pair of clusters r and s, such that the distance, D(r, s), is minimum among all the pairwise distances.
4. Merge r and s to a new cluster t and compute the between-cluster distance D(t, k) for any existing cluster k ≠ r, s . Once the distances are obtained, delete the rows and columns corresponding to the old cluster r and s in the D matrix, since r and s do not exist anymore. Then add a new row and column in D corresponding to cluster t. 5. Repeat Step 3 a total of n − 1 times until there is only one cluster left.

K-MEANS ALGORITHM BASED ON MAPREDUCE

As the analysis above, PKMeans algorithm needs one kind of MapReduce job. The map function performs the procedure of assigning each sample to the closest center while the reduce function performs the procedure of updating the new centers. In order to decrease the cost of network communication, a combiner function is developed to deal with partial combination of the intermediate values with the same key within the same map task. In Map-function the input dataset is stored on HDFS [16] as a sequence file of<key, value>pairs, each of which represents are cord in the dataset. The key is the offset in bytes of this record to the start point of the data file, and the value is a string of the content of this record. The dataset is split and globally broadcast to all mappers. Consequently, the distance computations are parallel executed. For each map task, PKMeans construct a global variant centers which is an array containing the information about centers of the clusters. Given the information, a mapper can compute the closest center point for each sample. The intermediate values are then composed of two parts: the index of the closest center point and the sample information. The pseudo code of map function is shown in Algorithm

DBSCAN CLUSTERING ALGORITHM

DBSCAN Clustering Algorithm The purpose of clustering algorithm is to convert large amounts of raw data into separate clusters in order to better and faster access. DBSCAN and Kmeans are two major algorithm for processing clustering problem. DBSCAN is a density-based clustering algorithm , which can generate any number of clusters, and also for the distribution of spatial data. KMeans algorithm is based on the prototype, which can find the approximate class for the given value. Compared to K-means algorithm, DBSCAN does not need to know the number of classes to be formed in advance. It can not only find freeform class, but also to identify the noise points. Class is defined as a collection contains the maximum number of data objects which density connectivity in DBSCAN algorithm. The idea of the algorithm is for all of the unmarked objects in data set D, select object P and marked P as visited. Region query for P to determine whether it is a core object. If P is not a core object, then mark it as noise and reselect another object that is not marked. If P is a core object, then establish class C for the core object P and generals the objects within P as seed objects to region query to expanding the class C until no new object join class C, clustering process over. That is when the number of objects in the given radius (ε) region not less than the density threshold (MinPts), then clustering. Because of taking the density distribution of data object into account, so it can mining for freeform datasets. The following are related terms, definitions [4] of DBSCAN algorithm:

- The area in ε of the specified object P is called ε region of P.
- If the number of objects in the ε region of object P is not less than the given density Threshold MinPts, say P is a core object.
- The given dataset D, if object P is included in the ε region of object Q, and Q is a core Object, say P starting from Q is directly density-reachable.
- Object list {P1, P2... Pn}, P1 = Q, Pn = P, for Pi ∈ D (1 ≤ i ≤ n), if Pi +1 starting from Pi is directly density-reachable on ε and MinPts, then P starting from Q is density-reachable on ε and MinPts.
- For the object O in the dataset D, if object P starting from O is density-reachable on ε and MinPts and object Q starting from O is also density-reachable on ε and MinPts, then call P to Q is density-linked on ε and MinPts.
- If object P does not belong to any class, P is said to noise. DBSCAN discover classes by querying the ε region of unmarked objects in the dataset. If the ε region of an object P has the number of objects is more than MinPts, then create a new class based on the core object P, all of the objects in the ε region of P and P are belong to the same class. Objects in the ε region of P are treated as seeds and region query for these seeds to expanding the class until there is no new object join class.

Algorithm pseudo-code is as follows:

Input: dataset D, radius Eps, density threshold MinPts

Output: class C

```
1. DBSCAN（D, Eps, MinPts）
2. Begin
3. Init C=0; // the number of classes is initialized to 0
```

```
4. for each unvisited point p in D
5. Mark p as visited; //marked P as accessed
6. N = getNeighbours (p, Eps);
7. If sizeOf (N) < MinPts then
8. Mark p as Noise; //if sizeOf (N) < MinPts then mark P as noise
9. Else
10. C= next cluster; //create a new class C
11. Expand Cluster (p, N, C, Eps, MinPts); //expand class C
12. End if
13. End for
14. End
```

DBSCAN algorithm has two parameters: the radius ε and density threshold MinPts, Determination of parameters determine the final clustering result. On the computational complexity, if using spatial index, it can be analyzed the time complexity of DBSCAN is O (nlogn), n is the number of objects in the database. Otherwise, the time complexity rises to O (n2) [5]. As we know, DBSCAN algorithm firstly determine whether an object is the core object, if it is, then continue to expand the class with the object as the center. In this process, with the increasing of core objects, the objects which are not marked are stored in memory, if there is a very large clustering in database, it will require a lot of memory space to store the core object information, and will lead to high I/O overhead, as a result, the clustering speed will be seriously affected. So this paper proposes a parallel DBSCAN clustering algorithm based on MapReduce to reduce I/O overhead and improve the clustering speed.

DBSCAN ALGORITHM BASED ON MAPREDUCE

In DBSCAN algorithm extending class is by verifies whether the given object in the dataset D is a core at the specified radius ε. The Algorithm takes a large part of the time spending on the region query of object. When the dataset is very large, the inputted data objects are many, serial DBSCAN algorithm to determine whether each of object is core object will consume a high I/O overhead. Researchers found that the determination of core object can be parallelized, and we can get a conclusion by analyzing the algorithm that if an object exists in two different classes, and it is a core object, then these two classes can be merged into a new class. Otherwise the object belongs to one of classes, there is no relationship between the two classes. This paper introduces a new concept - sharing object sharing object Core objects P and Q belong to different classes, if the object O starting from P, Q are directly densityreachable, then called object O is a sharing object. And if O is a core object, called O is a sharing core object. If there exist a sharing core object in different classes, these classes can be combined into a new class. As shown in Figure 3, object O is a sharing core object: Figure 3. Clustering of Overlapping Objects in Region Obviously we can take advantage of the MapReduce programming model to parallelize the whole process to save clustering time and resources. The basic idea of
DBSCAN algorithm based on MapReduce is divided into four steps [6]:

1)Step one: The data in the dataset cut into small blocks which are equal size.

2) Step two: the blocks are distributed to the nodes in the cluster, so that all of nodes in the cluster can run the Map function of themselves in parallel to calculate and process those blocks.

3) Initial Cluster. For any unlabeled object p in the dataset, using Map-Reduce parallel programming model to calculate the number of objects in its region to determine if it is a core object. If is, P and all the objects in the region of P constitute an initial class (C), and marked those objects with the same cluster identifier (Cid). Conversely, if p is not a core object, and there is no other object in its region, marked P as noise. Otherwise, detect whether there has a core object q in the region of non-core object p. If has, given the object p and the objects in the region of q with the same cluster identifier. Repeat until all of the objects in the dataset are identified. After the process is completed, get an initial class clusters and a noise set.

4) Merge Result. Class merging is to consider these objects which exist in more than two classes. If there is a sharing core object, merging the two classes. Else classify the object as the proximity side. It will be given a new class name if there have different classes are combined. When there is no object exist in different classes, mergering initialize class completed. 3. Step three: merge the result of each processor. 4. Step four: Output clustering results. Figure 3 shows the ideas of DBSCAN algorithm based on MapReduce, the data

exchange format of each stage is, and the object identifier Oid as the key, value is filled with (core_tag, used_tag, Cid, x, y, and z). Core_tag indicates whether it is a core object, used_tag identify whether there had been clustering, Cid means class logo .Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

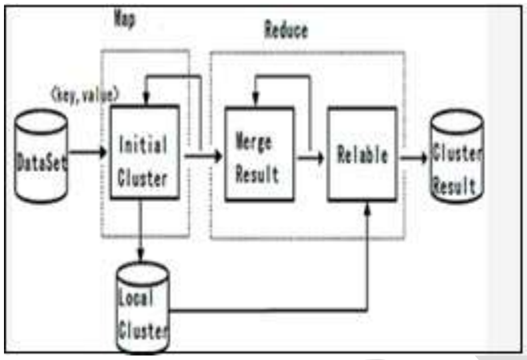
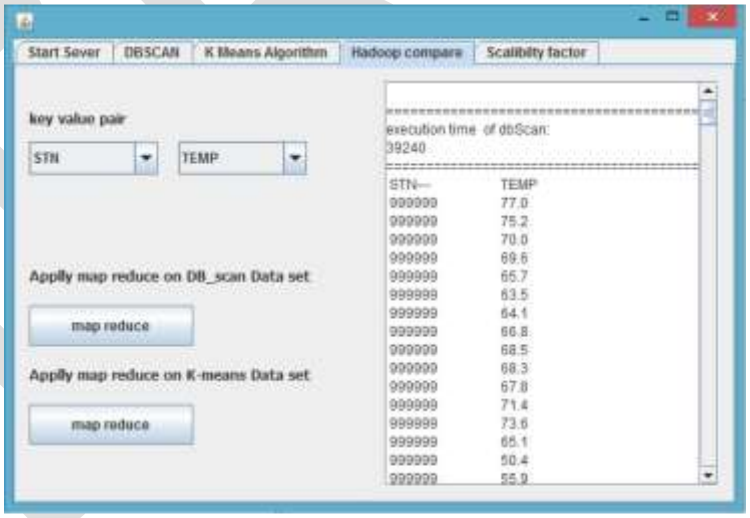


Fig.3 DBSCAN with MapReduce

EXPERIMENTAL RESULTS

For the test, we have taken a data set and selected a key value pair. After implementing DBSCAN algorithm based on MapReduce we get the following result. The output in fig 4. Displays the execution time taken by DBSCAN algorithm on Hadoop MapReduce Framework. Similarly, after implementing k-means clustering algorithm on Hadoop for same data set and same key value pair we get the execution time as shown in fig.5. Table I summarizes the execution time taken by both the algorithms on Hadoop

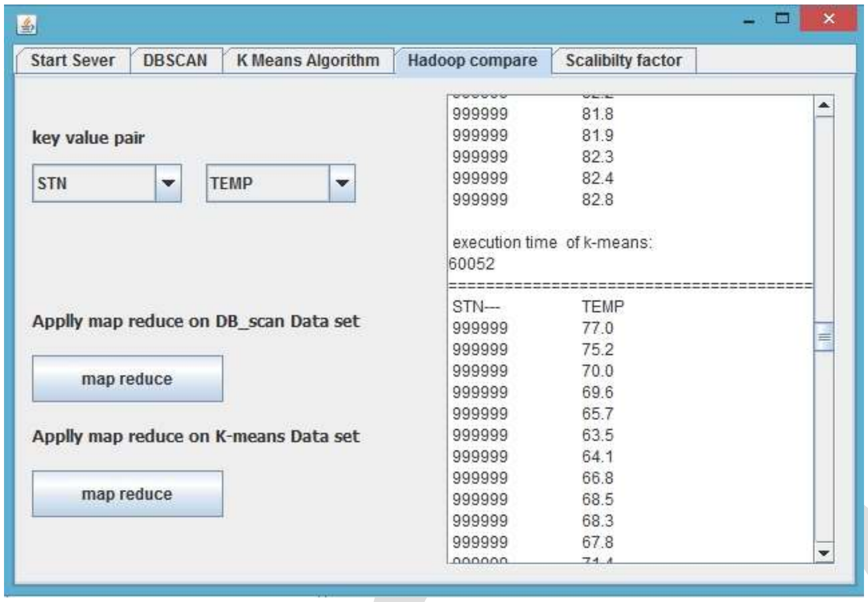
Experiment 1: Execution time taken by K-Means algorithm and DBSCAN clustering algorithm based on Hadoop for the same documents.



Key value pair: STN, TEMP

```
execution time  of dbScan:
39240
STN---          TEMP
999999          77.0
999999          75.2
999999          70.0
999999          69.6
999999          65.7
999999          63.5
999999          64.1
999999          66.8
999999          68.5
999999          68.3
999999          67.8
999999          71.4
999999          73.6
999999          65.1
999999          50.4
999999          55.9
```

Fig. 4 Execution time taken by DBSCAN



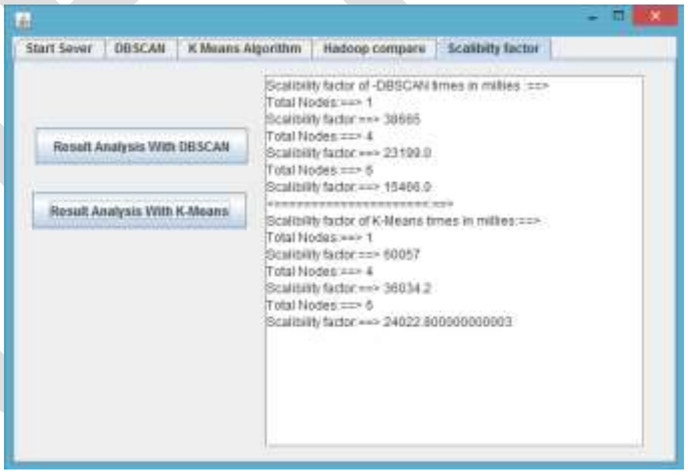
```
999999      81.8
999999      81.9
999999      82.3
999999      82.4
999999      82.8

execution time  of k-means:
60052
=====================================
STN---      TEMP
999999      77.0
999999      75.2
999999      70.0
999999      69.6
999999      65.7
999999      63.5
999999      64.1
999999      66.8
999999      68.5
999999      68.3
999999      67.8
```

Fig.5 Execution time taken by K-means

Table 1 Execution time table

Clustering Algorithm	DBSCAN algorithm	K-means algorithm
Execution time in seconds(s)	39.240	60.052

Experiment 2: As the number of nodes increases the time taken by each algorithm is as follows



```
Scalibility factor of -DBSCAN times in millies :==>
Total Nodes:==> 1
Scalibility factor:==> 38665
Total Nodes:==> 4
Scalibility factor:==> 23199.0
Total Nodes:==> 6
Scalibility factor:==> 15466.0
*************************:==>
Scalibility factor of K-Means times in millies:==>
Total Nodes:==> 1
Scalibility factor:==> 60057
Total Nodes:==> 4
Scalibility factor:==> 36034.2
Total Nodes:==> 6
Scalibility factor:==> 24022.800000000003
```

Fig. 6 Scalability Factor based on number of nodes

Table 2 Time taken by both algorithm as number of nodes are increased

Number of nodes	2	4	6
Time taken by DBSCAN algorithm in seconds(s)	38.665	23.199	15.466
Time taken by K-means algorithm in seconds(s)	60.057	36.034	24.022

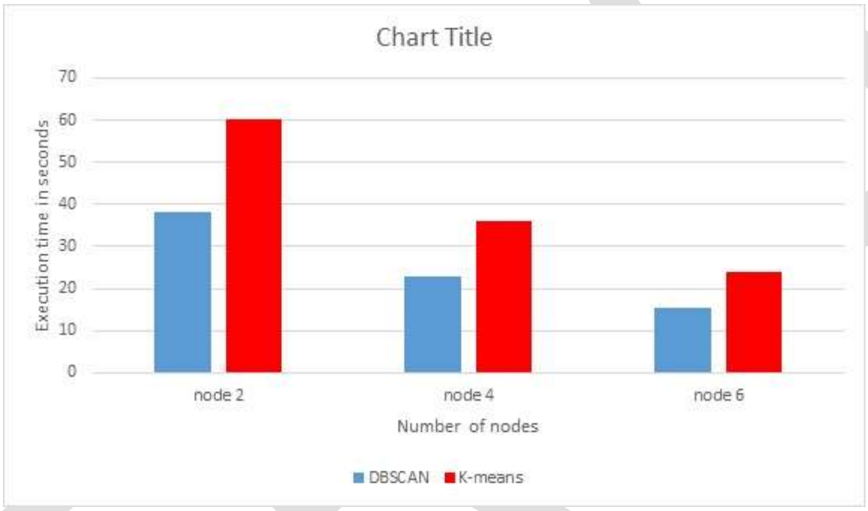


Fig. 7 Execution time taken by both algorithms as number of nodes are increased

CONCLUSION

DBSCAN and K-means are two major algorithm for processing clustering problem. These algorithms can automatically cluster the data making full sense of the Hadoop cluster performance.

The main deficiency with K-means clustering algorithm is we have to set the number of clusters to be generated in advance and it is also sensitive to noise and outlier data. Compared to K-means algorithm, DBSCAN does not need to know the number of classes to be formed in advance, it can not only find freeform classes but also identify noise points.

As per the results obtained K-means clustering algorithm performed on Hadoop alleviates the problem of time delay caused by increasing the number of nodes.

REFERENCES:

[1] Aditya Jadhav, Mahesh Kukreja "Parallel Data Mining and Assurance Service Model Using Hadoop in Cloud" International Journal on Advanced Computer Engineering and Communication Technology (IJACECT),ISSN (Print): 2278-5140, Volume-1, Issue – 2, 2012

[2] Xiufen Fu, Shanshan Hu and Yaguang Wang "Research of parallel DBSCAN clustering algorithm based on MapReduce" International Journal of Database Theory and Application Vol.7, No.3 (2014), pp.41-48

[3] K Means Clustering Algorithms https://onlinecourses.science.psu.edu/stat857/node/ 108

[4] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data", Data & Knowledge Engineering, vol. 60, no. 1, (2007), pp. 208-221.

[5] C. Böhm, K. Kailing and P. Kröger, "Computing clusters of correlation connected objects", Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, (2004), pp. 455-466

[6] Xiufen Fu, Shanshan Hu and Yaguang Wang "Research of parallel DBSCAN clustering algorithm based on MapReduce" International Journal of Database Theory and Application Vol.7, No.3 (2014), pp.41-48

[7] "Data intensive computing." [Online]. available: http://en.wikipedia.org/wiki/data intensive computing

[8] "MapReduce: simplified data processing on large clusters" Jeffrey dean and Sanjay Ghemawat. Commun. Acm, 51(1):107–113, 2008.

[9] Aditya B. Patel, Manashvi Birla, Ushma Nair (6-8 Dec. 2012, "Addressing Big Data Problem Using Hadoop and Map Reduce"

[10] Yu Li; Wenming Qiu; Awada, U; Keqiu Li, (Dec 2012)," Big Data Processing in Cloud Computing

[11] Garlasu, D.; Sandulescu, V.; Halcu, I.; Neculoiu, G. ;,(17-19Jan. 2013),"A Big Data implementation based on Grid Computing", Grid Computing

[12] Sagiroglu, S.; Sinanc, D., (20-24 May 2013),"Big Data: A Review"

[13] Grosso, P.; de Laat, C.; Membrey, P., (20-24 May 2013)," Addressing big data issues in Scientific Data

[14]] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, and G. Fox. Twister:"A runtime for iterative MapReduce." In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, pages 810818. ACM, 2010

[15] S. Ibrahim, H. Jin, L. Lu, L. Qi, S. Wu, and X. Shi."Evaluating mapreduce on virtual machines: The hadoop case". Cloud Computing, pages 519, 2009

[16] W. Zhao, H. Ma, and Q. He. "Parallel k-means clustering based on mapreduce." Cloud Computing, pages 674, 2009