I J C T A, 9(3), 2016, pp. 1453-1459

Analysis of Deflated Power Iteration Clustering on Hadoop Cluster

Jayalatchumy Dhanapal*, Thambidurai Perumal**

Abstract: Big Data has to be stored and processed efficiently to extract knowledge and information from them. Scalability and performance becomes an important issue when dealing with large data intensive computing. There is also a need to address the failure and improve the effectiveness of existing machine learning algorithm. MapReduce is a simplified programming model used for data intensive distributed parallel computing. This motivates us to redesign and convert the existing sequential algorithm to MapReduce algorithms. One such unsupervised sequential algorithm is Deflated Power Iteration Clustering algorithm (DPIC) which uses Schurz deflation technique to compute multiple orthogonal pseudo eigen vectors. Finally they are clustered using K-means. This paper discusses about the implementation of DPIC algorithm over a distributed environment using Apache Hadoop. The experimental results show that DPIC algorithm achieves better performance in MapReduce framework when handling larger datasets. Also, they are proved to be effective and accurate.

Keywords: MapReduce, BigData, Hadoop, Power-Iteration, MapReduce, Performance, Deflation.

1. INTRODUCTION

The data volume is scaling faster than computing resources. Hence managing large datasets is a challenging task. The larger the dataset longer is the time taken for computation. Despite of algorithmic improvements proposed in many serial algorithms, they could not handle large scale problems further slowing down the process [11][12]. It is therefore a growing need to develop efficient parallel data mining algorithms that can run on a distributed system.that leads to powerful platforms. Distributed computing is a technique to solve the computational problems mainly by sharing the computation over network. The first such attempt was made by Google. e.g., Hadoop MapReduce. Numerous notable attempts have been initiated to exploit massive parallel processing architectures as reported in [9]. Cluster analysis is the most important of the data mining techniques. Though traditional clustering methods are still popularly used, they still suffer from curse of dimensionality [26]. Motivated by the need for parallelism and effectiveness in this paper, implementation of the DPIC on MapReduce is performed. While the main effort in this paper is on parallelization strategies and implementation details for minimizing computation and communication costs, additional work has been done on the MapReduce framework.

The rest of the paper is organized as follows. Section II gives a literature review on various clustering methodologies. In Section III we discuss about the deflated PIC algorithm followed by the implementation of DPIC algorithm in section IV. In section V and VI, the mapper/reducer algorithms and its experimental results are shown. Finally section VII concludes the work giving its future directions.

2. LITERATURE REVIEW

There have been many studies conducted on various clustering methodologies [6]. The computational time and memory space occupied by spectral clustering algorithms is very large as there is a need of similarity measure calculations [1][2]. Fowleks et.al [13] proposed new method Nystrom to avoid the calculation of

* Department of CSE/PKIET Pondicherry University. India, *Email: djlatchumy@gmail.com*

** Thambidurai Perumal Department of CSE/PKIET Pondicherry University. India, *Email: pdurai157@gmail.com*

similarity matrices. Dhillion et.al [14] proposed a method that does not use eigen vector. The advantage with these algorithms is that they reduce the computational time but the accuracy ratio and memory usage have not been addressed. To overcome these drawbacks Lin and Cohen [5] proposed a clustering algorithm based on power iteration which replaces the Eigen value decomposition by matrix vector multiplication. In terms of parallel framework many algorithms have been implemented in MPI, a message passing interface [4][15]. Yang [28] used this MPI method to distribute data. MPI is usually used to exploit parallelism across various nodes.

The MPI mechanism increases the communication between machine and the network. MPI is bound in terms of performance when working with large dataset. Weizhong Yan et.al [4] have chosen MPI as the programming model for implementing the parallel PIC algorithm since it is efficient and performs well for communicating in distributed clusters[27].

More recently MapReduce [17] a Google parallel computing framework and its Java version Hadoop have been increasingly used [25]. Weizhong Zhao et.al have proposed Parallel K-means algorithm based on MapReduce that can scale well and efficiently to process large datasets on commodity hardware[10][21]. Qing Liao et.al [22] have proposed an Improved K-means based on MapReduce which improve the performance of traditional ones by decreasing the number of iterations and accelerating processing speed per iteration.

To overcome the efficiencies of existing algorithms for parallel matrix multiplication, Jian-Hua Zheng et.al [20] have presented a processing scheme based on VLC. Xiaoli Cu et.al[23] has proposed a novel processing model in MapReduce to obtain high performance by eliminating the iteration dependency. MapReduce takes advantage of local storage to avoid these issues while working with large datasets [16]. Hadoop supports fault tolerance and hence in this paper the implementation is carried on a parallel framework MapReduce.

3. DEFLATED POWER ITERATIVECLUSTERING

Nowadays, spectral clustering algorithm is an active research areas in machine learning. Deflated Power Iterative Clustering is one among the spectral clustering algorithm which performs well compared to other traditional clustering algorithm like K-Means. The best method for computing the larger eigen vector is the Power Iteration. In this method, the vectors are initiated randomly and it is denoted by v^t. The power method is given as $v^t = \gamma W v^{t-1}$ where, '*W*' is the matrix and γ is a normalizing constant which controls the vector from becoming too large or small [7].

The speed of computing the first pseudo eigenvector is the same for every pseudo eigen vector. Anh Pham The et.al has proposed a sequential method to find *k* mutually orthogonal pseudo-eigen vectors, where *k* is the number of clusters. Instead of finding one pseudo eigen vector like Power Iterative Clustering, Deflated-Power Iterative Clustering (DPIC) finds additional pseudo eigen vector. Each pseudo eigen vector contain additional useful information for clustering.

Deflation is a technique to manipulate the system. After finding the largest eigen value displace it in such a way that the next larger value is the largest value in the system and apply power method. The Weildnet deflation is a single vector technique. Schur decomposition is a technique where the basic idea is if one vector of 2 norm is know it can be completed by (n-1) additional vector to form the orthogonal basis of C^n.

Similarly various studies have been done on various deflation techniques like Weildnet deflation; Hotelling deflation, Projection deflation, and Schur complement deflation. Of all these techniques it has been found that the only methods that find the orthogonality of pseudo eigen vectors is Schur complement deflation. In DPIC the original affinity matrix '*W*' has eigen vectors (x_1, x_2, x_3….x_n) with eigen values

(λ_1, λ_2,…… λ_n). First, from the original affinity matrix W_0, they use power iteration as PIC to find the first pseudo-eigenvector v_1. The algorithm for DPIC is shown in Figure 1.

On applying deflation the effect of v_1 is eliminated on W_0 to obtain the second affinity matrix W_1. From W_1, power iteration is applied again to obtain the second pseudo-eigenvector v_2. The effect of v_2 on W_1 is eliminated again using schurs deflation. This process is repeated k times. Due to multiple pseudo eigen vector computation, DPIC requires more execution time than PIC. This observation gave an idea to implement the DPIC algorithm using MapReduce

Input: A dataset $X = \{x_1, x_2, \ldots..x_n\}$

Step 1. Read the input and construct the similarity matrix.

Step 2. Normalise the obtained matrix by dividing each element by its row sum

Step 3. Compute the initial vector $v_0 = \frac{R}{\|R\|}$

Step 4. Calculate the sub vector $v^t = \gamma W v^{t-1}$ until the stopping criteria is met**//deflation technique**

Step 5. Find the k^{th} vector and then remove the effect of v_1 on W by Schur deflation using

$$W_l = W_{l-1} - \frac{W_{l-1} v / v_l w l-1}{W_{l-1}}$$

Step 6. Perform iteration by incrementing the value until the condition satisfies.

Step 7. Use K-Mean to cluster point v_t.

Output: Cluster $C_1, C_2, \ldots\ldots\ldots C_k$

Figure 1: Algorithm for DPIC

4. MAPREDUCE OPERATING PRINCIPLE

Google introduced Hadoop, a technique to analyze, manipulate and visualize Big Data[18][19]. Hadoop is a software framework that consists of Hadoop kernel, MapReduce, HDFS and other components [8]. The MapReduce has two phases, Map and Reduce. Users specify a map function that processes a <Key/Value> pair to generate a set of intermediate <Key/Value> pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

During processing the HDFS splits the huge data into smaller blocks called the chunks which are 64 MB by default. The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files [13][24]. The data is split in <Key, Value> pair and the Map function is invoked for every <Key, Value> pair.

The output is stored in a buffer. When the buffer gets its threshold size the contents are partitioned and written to the disk. A memory sort is performed and the results are given to the combiner before which the files are merged into a single file and compressed accordingly. As soon as the map is completed the output is copied to buffer of the reduce task. When the threshold level is reached the outputs are merged and written to the disk [8]. The map does the sorting; the output is copied and merged. During reduced phase the reduce function is called for each key of the output that is sorted and written to the HDFS. The execution model of MapReduce is shown in the Figure 2.

5. DPIC IN MAPREDUCE

The first step is to design the MapReduce function to handle input/output. The input is given as <Key, Value> pair where ‘Key’ is the cluster and <Value> is the vector in the dataset. The dataset are stored in the

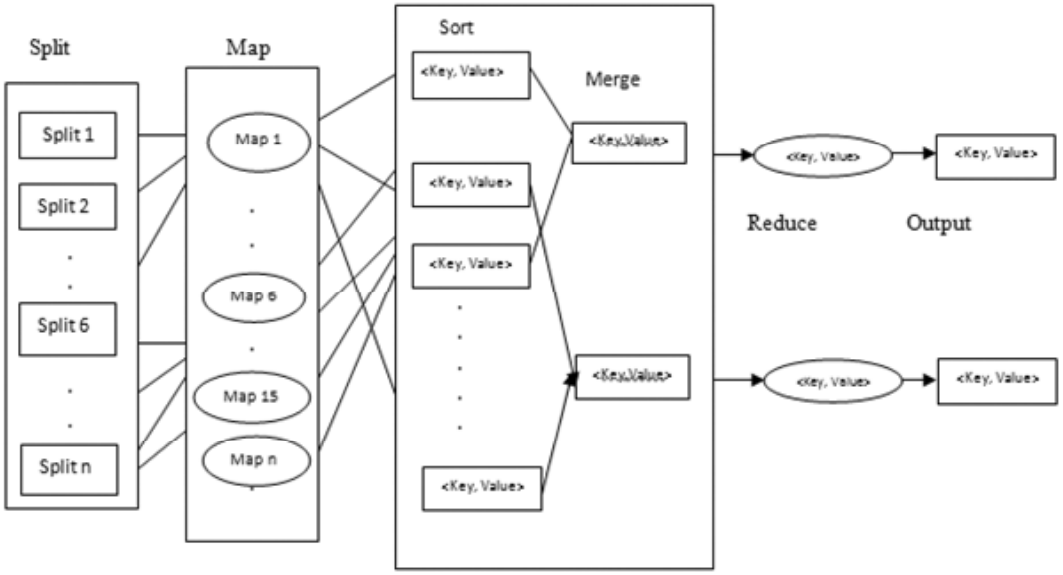

Figure 2: MapReduce Execution Model

Input: Normalized Similarity matrix

Output: Final vectors

Procedure DPIC REDUCEDESIGN

NEW vectors

COMBINE resultant normalized sub matrix from MAP CLASS

Generate the initial vector $v_0 = \frac{R}{\|R\|}$

where *R* is the row sum of *W*.

Find v_1 from W_{l-1}

Remove the effect of v1 on w_{l-1} using

$$W_t = W_{t-1} - \frac{W_{t-1} V_i V_j W_{l-1}}{V_c W_{l-1} V_t}$$

Calculate new vector and new velocity v_t

On increment, check for the convergence and cluster results

WRITE cluster file to output *DIRECTORY*

End Procedure

Figure 3: Algorithm for DPIC Mapper Design

Input: Datasets$\{x_1, x_2 \ldots x_n\}$, splited into sub datasets such as split 1, split 2 …split *n*,

The sub datasets form <Key, Value> lists.

These<Key, Value>lists are theinput to the map function.

Output: Normalized Similarity matrix.

Procedure DPIC MAPDESIGN

Load cluster datasets.

1. $W_1 \leftarrow \{x_1, x_2, \ldots . x_n\}$
2. Initial vector $\leftarrow V_0$
3. $S(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|2^2}{2\sigma^2} \right)$
4. Construct the similarity matrix $A \leftarrow R^{n*n}$
5. Normalize the matrix using
6. $A(i, j) = \frac{A(i, j)}{sum[k-1]\text{^}N}$
7. CREATE output record <A>
8. Call DPIC reduce

End Procedure

Figure 4: Algorithm for DPIC Reducer Design

input directory of the HDFS and this forms the <Key> field in the <Key, Value> pair. The commands need to compute the similarity between the dataset is assigned to the mapper function. The mapper is structured in such a way that it computes similarity and performs normalization.

Once the normalization is processed they are assigned to the reducer. The matrix value computation is done and the vectors are generated. Now the effect of the vector is removed from W. Next the cluster is updated and new vector and velocity is rewritten to the disk which is ready for the next iteration. After

understanding the mapper and the reducer function the algorithm for mapper/reducer is designed and shown in Figure 3 and Figure 4.

6. EXPERIMENTAL RESULTS

In this section the performance of the deflated PIC in MapReduce environment is evaluated with respect to the speedup, scalability and efficiency. The experiment was conducted on a cluster of computers each of which has two, 2.40 GH cores and 4 GB of memory. The Hadoop version is 0.17.0, java 1.6.0.18 are used in MapReduce systems. The data are executed on real datasets with 2, 4,6 and 8 nodes respectively in single machine environment and MapReduce environment and their execution time are shown in Figure 5 and Figure 6.

Test of speedup ratio: The speedup is an initial scale for consideration as it is an important key term to calculate the execution time and improvement in performance. It is used to calculate how many times a parallel algorithm works faster than a serial algorithm. The greater the ratio of speedup, lesser is the time that the parallel processors exploit [3]. This is the reason for parallelization of a sequence algorithm. The speedup is calculated using the formula

$$S = \frac{Ts}{Tp}$$

Where T_s is the execution time of the fastest sequential program and Tp is the execution time of the parallel program. If a parallel program is executed on (p) processor, the highest value is equal to number of processors [3]. In this system every processor needs T_s/p time of complete the job.

$$S = \frac{T_s}{\frac{T_p}{p}} = P$$

The result of speedup ratio performance tests according to the various synthetic datasets are shown in Figure 7. It is clearly seen that as the value of the speedup increases the execution time decreases. In the graph the dataset of different size for different nodes are given along the x axis and its execution time in ms is given along the y axis. Since the time taken for execution of the algorithm decreases we can conclude that there is an increase in speed up. From the figure it is seen that the speedup ratio and the performance is improving drastically as the data size grows (i.e.) as the data increases the speedup performance also increases. When the number of nodes increases it narrows the speedup [3]. It can be illustrated that the efficiency of DPIC in Hadoop platform is greater. The figure shows that the execution time of DPIC is lesser. Hence speedup is more. According to Amdahl's law [3], it is difficult to obtain the speedup of a parallel system with many processors because each program has a fraction á that cannot be parallelized, only the rest of (1-á) gets parallel. So the speedup for parallel program is

$$S = \frac{Ts}{Tp} = \frac{Ts.\alpha + Ts(1-\alpha)}{P} = \frac{P}{\alpha.(P-1)+1}$$

Analysis of Scalability: **T**he parallel efficiency of an algorithm is the speedup of the processor during parallel program execution. The formula [3] to calculate is

Efficiency (E) = S_R /p. where

S_R represents the ratio of speedup, p represents the number of processors in the cluster.

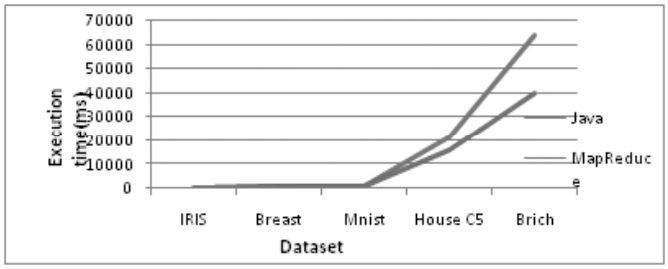

Figure 5: Execution time in Single Machine Environment

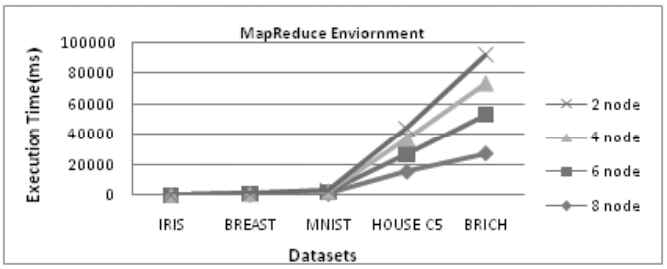

Figure 6: Execution time in MapReduce Environment

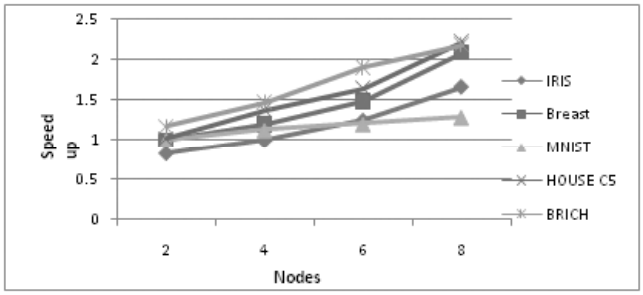

Figure 7: Speedup results for DPIC algorithm different datasets with varying nodes.

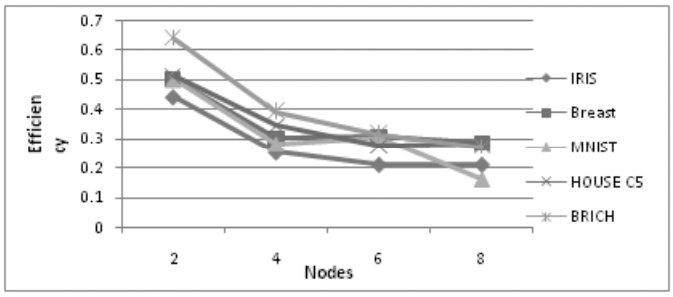

Figure 8: Efficiency curve for DPIC algorithm for different datasets with varying nodes.

Figure 8 shows the efficiency of DPIC for different datasets. From the figure it is seen that [3] as the data size increases, the efficiency is greater, (i.e.) it has better scalability. The graph for relative efficiency of the algorithm for different synthetic datasets from UCI machine repository has been shown below.

7. CONCLUSION

In this study we have applied MapReduce to DPIC algorithm and clustered over large data points of varying synthetic datasets. The algorithm shows an improvement in performance. The computational cost and time complexity has been much reduced using MapReduce. It is noticed that increasing the number of nodes in the mapper affects the performance (i.e.) more the number of nodes better is the performance, Thus we can conclude that MapReduce algorithm is highly scalable and is best suited for clustering of Big Data. The work has to be validated in real time increasing the size of datasets. The performance of the algorithm is to be further investigated in cloud.

References

[1] C. Fowlkes, S. Belongie, F. Chung, J. Malik, “Spectral grouping using the Nystrom method”, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 26.Issue 2. January-2004.

[2] W.Y. Chen, Y. Song, H. Bai, C. Lin, E.Y. Chang, Parallel spectral clustering in distributed systems, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 33. Issue 3 568–586. 2011.

[3] Felician ALECU, Performance Analysis of Parallel Algorithms, Journal of Applied Quantitative methods vol (2) Issue 1.Spring129-134.2011.

[4] Weizhong Yana et al., p-PIC: Parallel power iteration clustering for big data, Models and Algorithms for High Performance Distributed Data Mining. Volume 73, Issue 3.2013

[5] Frank Lin frank, William W, Power Iteration Clustering, International Conference on Machine Learning, Haifa, Israel. June 2010.

[6] Kyuseok Shim, MapReduce Algorithms for Big Data Analysis, Proceedings of the VLDB Endowment, Vol. 5, No. 12, Copyright 2012 VLDB Endowment 21508097/12/08.2012.

[7] http://en.wikipedia.org/wiki/Cosine_similarity

[8] Ran Jin, Chunhai Kou, Ruijuan Liu and Yefeng Li, Efficient Parallel Spectral Clustering Algorithm design for large datasets under Cloud computing, Journal of Cloud computing: Advances, Systems and Applications 2:18. 2013.

[9] Niu XZ, She. K, Study of fast parallel clustering partition algorithm for large dataset. ComputSci 39; 134-15.2012.

[10] Zhao WZ, Ma HF, Shi ZZ, Research on Parallel K means algorithm design based on Hadoop platform, ComputSci 38; 166-176.2011

[11] Unren Che, Mejdl Safran, and Zhiyong Peng, From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, DASFAA Workshops 2013, LNCS 7827, pp. 1–15.2013

[12] M. Ester, H.P. Kriegel, J. Sander, X.W. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD-96, AAAI Press, pp. 226–231.1996.

[13] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[14] S Dhillion, Y. Guan, and B. Kulls, "Weighted graph cuts without Eigen vector: a multilevel approach. IEEE trans. On Pattern analysis and machine intelligence vol 29. Issue 11.1944-1957.2013

[15] Hitesh Ninama, "Distributed data mining using Message Passing Interface" Review of Research [2249-894X] vol: 2 iss: 9.

[16] Ellis H. Wilson III and Mahmut T. Kandemir, Garth Gibson, Exploring Big Data Computational Traditional HPC NAS Storage, International conference on distributed computing systems,2014.

[17] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Communication. ACM, 51(1):107–113,2008

[18] http://www.webopedia.com/TERM/B/big_data.html

[19] http://www.cs.colorado.edu/~kena/classes/5448/s11/presentations/hadoop.pdf

[20] Jian-Hua Zheng, Liang-Jie Zhang, Rong Zhu, Ke Ning and Dong Liu, Parallel Matrix Multiplication Algorithm Based on Vector Linear Combination Using MapReduce, IEEE Ninth World Congress on Services, 978-0-7695-5024-4/13 IEEE DOI 10.1109/SERVICES.2013.67.2013

[21] Weizhong Zhao, Huifang Ma, and Qing He, Parallel K-Means Clustering Based on MapReduce, CloudCom, LNCS 5931, pp. 674–679, Springer-Verlag Berlin Heidelberg.2009

[22] Qing Liao, Fan Yang, Jingming Zhao, An Improved parallel K-means Clustering Algorithm with MapReduce, Proceedings of ICCT2013, 978-1-4799-0077-0/13, IEEE.2013

[23] Xiaoli Cui · Pingfei Zhu · Xin Yang ·Keqiu Li· Changqing Ji, Optimized big data K-means clustering using MapReduce, Springer Science+Business Media New York, Supercomput, DOI 10.1007/s11227-014-1225-72014

[24] http://www.datanubes.com/mediac/HadoopArchPerfDHT.pdf

[25] Yuan Luo and Beth Plale, Hierarchical MapReduce Programming Model and Scheduling Algorithms, Cluster, Cloud and Grid Computing (CCGrid), 12th IEEE/ACM International Symposium doi10.1109/CCGrid.2012.132, 13-16.2012.

[26] Michael Steinbach, LeventErtöz, and Vipin Kumar(2004), The Challenges of Clustering High Dimensional Data, New Directions in Statistical Physics, Book Part IV, Pages pp 273-309, doi 10.1007/978-3-662-08968-2_16

[27] "Where to Deploy Your Hadoop Clusters?" Accenture, June 2013.

[28] Y. Ma, P. Zhang, Y.Cao and L. Guo.Parallel Auto-encoder for Efficient Outlier Detection. In Proceedings of the 2013.