IMPLEMENTATION OF P-PIC ALGORITHM IN MAP REDUCE TO HANDLE BIG DATA

Jayalatchumy D[1], Thambidurai. P[2]

Abstract

Clustering is a process of grouping objects that are similar among themselves but dissimilar to objects in others. Clustering large dataset is a challenging resource data intensive task. The key to scalability and performance benefits it to use parallel algorithms. Moreover the use of Big Data has become very crucial in almost all sectors nowadays. However analyzing Big data is a very challenging task. Google's Mapreduce has attracted a lot of attention for such applications that motivate us to convert sequential algorithm to Mapreduce algorithm. This paper presents the p-PIC with Mapreduce, one of the newly developed clustering algorithms. P-PIC originated from PIC though scalable and effective finds it difficult to fit, and works well only for low end commodity computers. It performs clustering by embedding data points in a low dimensional data derived from the similarity matrix. The experimental results show that p-PIC can perform well in MR framework for handling big data. It is very fast and scalable. The results show that the accuracy in producing the clusters is almost the same in using Mapreduce framework. Hence the results produced by p-PIC in mapreduce are fast, scalable and accurate.

Keywords: *p-PIC, MapReduce, Big data, clustering, HDFS*

--***--

1. INTRODUCTION

Clustering is the process of grouping a set of abstract objects into classes of similar objects. Survey papers[1,2] provide a good reference on clustering methods. Sequential clustering algorithms work well for the data size that is less than thousands of data sets. However, the data size has been growing up very fast in the past decade that leads to the growth of big data. The characteristics volume, velocity and variety are referred to as big data by IBM. Big data is used to solve challenges that doesn't fit into conventional relational database for handling them The techniques to efficiently process and analyze become a major research issue in recent years. One common strategy to handle the problem is to parallelize the algorithms and to execute them along with the input data on high-performance computers. Compared to many other clustering approaches, the major advantage of the graph-based approach is that the users do not need to know the number of clusters in advance. It does not require labeling data or assume the number of data clusters in advance. The major problem of the graph-based approach is that it requires large memory space and computational time while computing the graph structure.

Moreover, all the pairs must be sorted since the construction of the graph structure must retrieve the most similar pair of the data nodes. This step is logically sequential and thus hard to be parallelized.. Therefore, to parallelize the graph-based approach is very challenging. One popular modern clustering algorithm is Power iteration clustering[5]. PIC replaces the Eigen decomposition of the similarity matrix required by spectral clustering by a small number of matrix–vector multiplications, which leads to a great reduction in the computational complexities.

The PIC algorithm has slightly better capability in handling large data. However, it still requires both the data and the similarity matrix fit into computer memory, which is infeasible for massive datasets. These practical constraints motivate us to consider parallel strategies that distribute the computation and data across multiple machines. Due to its efficiency and performance for data communications in distributed cluster environments, the work was done on MPI as the programming model for implementing the parallel PIC algorithm. Hadoop is an open source software framework that is under the Apache foundation [6]. It can take large amounts of unstructured data and transform it into a query result that can be further analyzed with business analytic software. The power of Hadoop is the built-in parallel processing and ability to scale in a linear fashion to process a query against a large data set and produce a result. The rest of the paper is organized as follows section II describes about the architecture of MapReduce. In section III and IV the PIC and p-PIC algorithms has been discussed. The algorithm design is designed in section V. In section VI the configuration of Hadoop followed by experimental results are discussed.

2. ARCHITECTURE OF MAPREDUCE

Hadoop is a Java-based software framework that enables data and can be installed in commodity linux clusters. Hadoop enables applications to work with thousands of nodes and terabyte of data, without concerning the user with too much detail on the allocation and distribution of data and calculation. . Hadoop comes with its own file system called the Hadoop Distributed File System (HDFS) and a strong infrastructural support for managing and processing huge petabytes of data. Each HDFS cluster consists of one unique server called the Namenode that manages the namespace of the file system, determines the mapping of blocks to Datanodes, and regulates access. Each node in the HDFS cluster is a Datanode that manages the storage attached to it. The datanodes are responsible for serving read and write requests from the clients and performing block creation, deletion and replication instructions from the Namenode.

There are many open source projects built on top of Hadoop. Hive is a data warehouse framework [17] used for ad-hoc querying and complex analysis. It is designed for batch processing. Pig [16] is a data flow and execution framework that produces a sequence of MapReduce programs. Mahout [14] is used to build scalable machine learning libraries that focus on clustering, classifcation, mining frequent item-sets and evolutionary programming. Pydoop is a python package that provides an API for Hadoop MapReduce and HDFS.

MapReduce is a programming framework [9] to process large scale data in a massively parallel way. The main benefit of mapreduce is the programmer is oblivious of the details related to the data storage, distribution, replication, load balancing. The programmer must specify two functions, a map and a reduce[9]. The typical framework is as follows (15
(a) The map stage passes over the input file and outputs (key, value) pairs;
(b) The shuffling stage transfers the mappers' output to the reducers based on the key;
(c) The reduce stage processes the received pairs and outputs the final result. (paper read2.pdf(secue))

The mapping operation can be performed in parallel, when each mapping operation is independent of the other. The parallelism helps to overcome failure, when one node fails the work is assigned to other nodes as the input data is still available. The data structure (key, value) pair is used to define both map and reduce methods[3].

$$\text{Map}(\text{key1},\text{value1}) \rightarrow \text{list}(\text{key2},\text{value2})$$

The Map function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. The Reduce function is then applied in parallel to each group.

$$\text{Reduce}(\text{k2}, \text{list}(\text{v2})) \rightarrow \text{list}(\text{v3})$$

The reduce method will provide either one value or empty return. The phases which performs map and reduce need to be connected. The architectural framework is shown in fig 1.

3. POWER ITERATION CLUSTERING

One popular modern clustering algorithm is Power iteration clustering. PIC replaces the Eigen decomposition of the similarity matrix required by spectral clustering by a small number of matrix–vector multiplications, which leads to a great reduction in the computational complexities. PIC finds only one pseudo-eigenvector, which is a linear combination of the eigenvectors in linear time. The effort required to compute the eigenvectors is relatively high, O(n3), where n is the number of data points. (PIC) [13] is not only simple but is also scalable in terms of time complexity, O(n) [13]. A pseudo-eigenvector is not a member of the eigenvectors but is created linearly from them. Therefore, in the pseudo Eigen vector, if two data points lie in different clusters, their values can still be separated.

Given a dataset X = (x1; x2…..x n) , a similarity function s(xi; xj) is a function where s(xi, xj) = s(xj ; xi) and s >= 0 if i≠ j, and, s = 0, if i = j. An affnity matrix A€ Rnxn is defined by Aij = s (xi; xj).

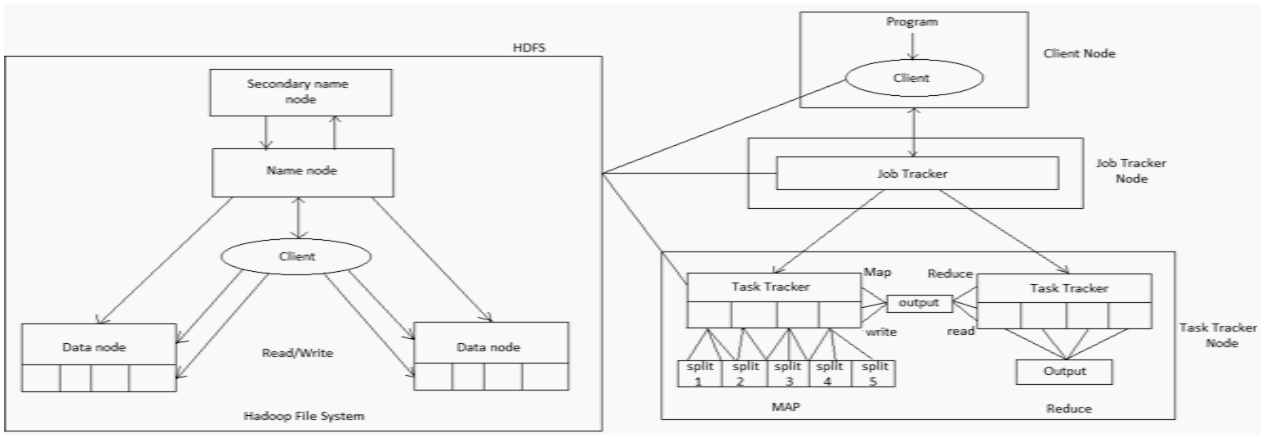

Fig 1: Architecture of a MapReduce Framework

The degree matrix D associated with A is a diagonal matrix with dij = ∑j Aij: A normalized affinity matrix W is defined as D-1A. Thus the second-smallest, third-smallest,. . . , kth smallest eigenvectors of L are often well-suited for clustering the graph W into k components. The main steps of Power iteration clustering algorithm are described in Fig 2:

- ➢ **Calculate the similarity matrix of the given graph.**
- ➢ **Normalize the calculated similarity matrix of the graph, $W=D^{-1}A$.**
- ➢ **Create the affinity matrix A € R^{n*n}, W from the normalized matrix, obtained by calculating the similarity matrix.**
- ➢ **Perform iterative matrix vector multiplication is done V^{t+1}**
- ➢ **Cluster on the final vectors obtained.**
- ➢ **Output the clustered vector.**

Fig 2: Power Iteration Clustering

4. P-PIC

The PIC algorithm has slightly better capability in handling large data. However, it still requires both the data and the similarity matrix fit into computer memory, which is infeasible for massive datasets. These practical constraints motivate us to consider parallel strategies that distribute the computation and data across multiple machines [5]. Due to its efficiency and performance for data communications in distributed cluster environments, the work was done on MPI as the programming model for implementing the parallel PIC algorithm. There are several different parallel programming frameworks available [7]. The message passing interface (MPI) is a message passing library interface for performing communications in parallel programming environments [5]. Due to its efficiency and performance for data communications in distributed cluster environments, the work was done on MPI as the programming model for implementing the parallel PIC algorithm. The algorithm for parallel PIC is described in fig 3[5]

- ➢ **Get the starting and end indices of cases from master processor.**
- ➢ **Read in the chunk of case data and also get a case broadcasted from master.**
- ➢ **Calculate similarity sub-matrix, A_i ,a n/p by n matrix.**
- ➢ **Calculate the row sum, R_i , of the sub-matrix and send it to master.**
- ➢ **Normalize sub-matrix by the row sum, $W_i = D_i^{-1} A_i$.**
- ➢ **Receive the initial vector from the master, v^{t-1}.**
- ➢ **Obtain sub-vector by performing matrix- vector multiplication, $v_i^t = \gamma W_i v^{t-1}$.**
- ➢ **Send the sub-vector, v_i^t , to master.**

Fig 3: Parallel- Power Iteration Clustering

5. ALGORITHM DESIGN FOR P-PIC IN MAPREDUCE

The algorithm discussed describes the procedure for implementing the parallel power iteration clustering using the hadoop mapreduce framework. The parallelism concept of mapreduce comes into picture. Map function is forked for every jobs. These maps are run in any node under distributed environment configured under Hadoop configuration[5]. The job distribution is done by the Hadoop system and files datasets are required to be put in HDFS. The dataset is scanned to find the entry and the frequency is noted. This is given as output to the reduce function in the reduce class defined in Hadoop core package. In the reducer function each output is collected. The comparison of flowchart for the p-PIC and p-PIC in MapReduce framework are show in the fig 4 and 6.

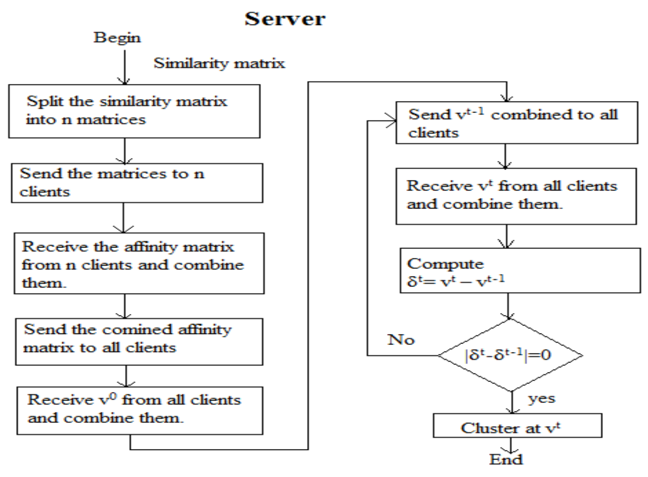
Server
Begin
Similarity matrix
Split the similarity matrix into n matrices
Send the matrices to n clients
Receive the affinity matrix from n clients and combine them.
Send the comined affinity matrix to all clients
Receive v^0 from all clients and combine them.
Send v^{t-1} combined to all clients
Receive v^t from all clients and combine them.
Compute $\delta^t = v^t - v^{t-1}$
$|\delta^t - \delta^{t-1}| = 0$
No
yes
Cluster at v^t
End

Fig 4: Flowchart for p-PIC using MPI

The pseudocode for the p-PIC using the Mapreduce environment for the Map function and the Reduce function is shown in fig 4 and 5.

```
Input: set $x=\{x_1,x_2,\ldots,x_n\}$ representing data set and similarity
function $s(x_i,x_j)$
                    let A[n,n] be the similarity matrix,
                     D[n,n] - diagonal matrix,
                     N[n,n] - normalized similarity
                              matrix and
                     R the row sum of normalized
                     matrix
          for i=1 to n do
                    D[i,i]=0;
                           for j=1 to n do
                     A[i,j]= $s(x_i,x_j)$
                     D[i,i]=D[i,i]+A[i,j]
               repeat
                     $N=D^{-1}A$
          produce output record <R>
      end function
```

Fig 4: Pseudocode for MAP function

```
input: constant $\gamma$ represents the normalizing constant
      let v be the vector
      let $\delta$ be the velocity
       $v^0 = R/||R||_2$
          loop
                $v^t=\gamma Wv^{t-1}$
                $\delta^t=|v^t-v^{t-1}|$
            until ($\delta^t$!=0)
      produce output record <$v^t$>
end function
```

Fig 5: Pseudocode for REDUCE function

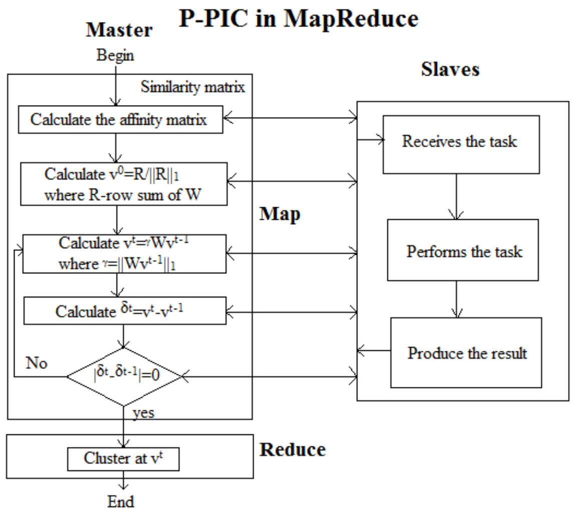
P-PIC in MapReduce
Master
Begin
Similarity matrix
Calculate the affinity matrix
Calculate $v^0=R/||R||_1$ where R-row sum of W
Calculate $v^t=\gamma Wv^{t-1}$ where $\gamma=||Wv^{t-1}||_1$
Calculate $\delta^t=v^t-v^{t-1}$
No
$|\delta^t-\delta^{t-1}|=0$
yes
Cluster at v^t
End
Map
Reduce
Slaves
Receives the task
Performs the task
Produce the result

Fig 6: Flowchart for p-PIC using MPI

6. CONFIGURATION OF HADOOP:

The hadoop setup for multimode configuration and single node is as follows. The prerequisites for the hadoop installation requires java to be installed on the system with JDK up and running. 2 Laptops (nodes) with SSH enabled and password-less logins. Install release 0.20.2. Operating system may be ubuntu or fedora.

Single node set up in hadoop[19]

- ❖ Install and configure hadoop
- ❖ Change the properties of conf/core-site.xml
- ❖ Change the properties of conf/hdfs-site.xml
- ❖ Change the properties of conf/mapred-site.xml
- ❖ Set JAVA-HOME in conf/hadoop-env.sh
- ❖ Setup passwrodless ssh'
- ❖ Format the namenode
- ❖ Finally start the job and execute the program

Multimode setup:

- ❖ Choose one of the node to be the primary Name Node (for HDFS) and Job Tracker node (for MR jobs)
- ❖ Enter the Primary node`s IP in conf/core-site.XML for the property "fs.default.name" (on both the nodes)[19]
- ❖ Enter the Primary node`s IP in conf/masters (only on the primary node)
- ❖ Enter both the nodes IPs in conf/slaves (only on the primary node). Both the nodes will act as Data(HDFS) and Task (MR) nodes
- ❖ Set the value of property "dfs.replication" to 2 (since we have 2 nodes, its better to replicate the HDFS blocks) in conf/hdfs-site.xml (on both the nodes)

7. EXPERIMENTAL RESULTS

The effectiveness of the original PIC for clustering has been discussed by Lin and Cohen [13]. The scalability of p-PIC have been shown by Weizhong Yana[5]. Hence in this paper we will focus on demonstrating the scalability of our parallel implementation of the PIC algorithm in the MR framework . To demonstrate the data scalability of p-pic over the framework , we implement our algorithm over a number of synthetic dataset of many records. We also created a data generator to produce a dataset used in our experiment. The size (n) of the dataset varies from 10000 to 100000 number of rows. We performed the experiment on local cluster. Our local cluster is HP Intel based and the number of nodes is 6. The stopping criteria for the number of clusters created are approximately equal 0. In this paper we used speed up(execution time) as the performance measure for implementing the mapreduce framework.

8. COMPARISONS OF PIC, P-PIC IN MPI AND HADOOP FRAMEWORK

We present performance results of mapreduce in terms of speedups on different no of processors and the scalability of algorithm with different database size is found. We compare p-pic with MapReduce performance has been increased along with the scalability. Figure 5 shows the time executed for various sizes of datasets. The data size is measured in MB and the time taken is calculated in milliseconds. The graph gives a comparison of PIC, p-PIC and p-PIC in Mapreduce Framework.

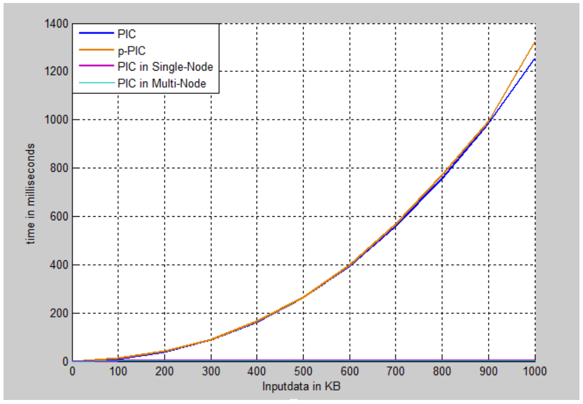

Fig 5: Comparison of speedup for PIC, p-PIC and p-PIC in MR

PIC	p-PIC	Sinle node	Multinode
10	13	0.312	0.152
39	41	0.326	0.157
89	91	0.339	0.161
163	167	0.342	0.168
265	266	0.356	0.173
396	401	0.368	0.179
558	566	0.373	0.184
756	770	0.379	0.188
985	992	0.385	0.191
1257	1324	0.391	0.196

Fig 6: Data (MB) vs Execution time (ms)

9. CONCLUSIONS

In this paper we have applied the MapReduce technique to p-PIC and have generated cluster for dataset of various data size. The results have been compared with sequential PIC and p-PIC using MPI. The paper shows that p-PIC has been successfully clustered on commodity hardware. The results show that the clusters formed using MapReduce is almost same as that of the other algorithms. The performance has been increased to a greater extend by reducing the execution time. It has been observed the performance increases with the increase in the data size. Hence it is more efficient for high dimensional dataset.

FUTURE WORK

When a single node causes whole system to crash and fail such node are known as single point failure nodes. In faults tolerance system its primary duty is to remove such nodes which causes malfunctions in the system [8]. Fault tolerance is one of the most important advantages of using Hadoop. It not only provides faults tolerance by detecting faults and also provides mechanism for overcoming them [11]. As a future work we can address how fault tolerance can be compared using MapReduce and with other frameworks.

REFERENCES

[1] Jain, M. Murty, and P. Flynn, “Data clustering: A review”, ACM Computing Surveys 31 (3) (1999) 264–323.

[2] Xu, R. and Wunsch, D. “Survey of clustering algorithms”, IEEE Transactions on Neural Networks 16 (3) (2005).

[3] Kyuseok Shim, “MapReduce Algorithms for Big Data Analysis”, Proceedings of the VLDB Endowment, Vol. 5, No. 12,Copyright 2012 VLDB Endowment 21508097/12/08.

[4] Chen,W.Y , Song, Y, Bai,H. and Lin. C, “Parallel spectral clustering in distributed systems”, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (3) (2011) 568–586

[5] 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, March 2013

[6] Apache Hadoop, http://hadoop.apache.org/

[7] W. Zhao, H. Ma, Q. He, “Parallel K-means clustering based on MapReduce”, Journal of Cloud Computing 5931 (2009) 674–679.

[8] H. Gao, J. Jiang, L. She, Y. Fu, “A new agglomerative hierarchical clustering algorithm implementation based on the map reduce framework”, International Journal of Digital Content Technology and its Applications 4 (3) (2010) 95–100.

[9] Anjan K Koundinya1,Srinath N K, et.al. , “Map/Reduce Design and Implementation Of Apriori algorithm For Handling Voluminous Data-Sets”, Advanced Computing: An International Journal,Vol.3, No.6, November 2012 Doi : 10.5121/Acij.2012.3604 29.

[10] Kyuseok Shim, “MapReduce Algorithms for Big Data Analysis”, Proceedings of the VLDB Endowment, Vol. 5, No. 12,Copyright 2012 VLDB Endowment 21508097/12/08.

[11] Ping ZHOU , Jingsheng LEI, Wenjun YE , “Large-Scale Data Sets Clustering Based on MapReduce and

Hadoop" Journal of Computational Information Systems 7: 16 (2011) 5956-5963.
[12] Robson L. F. Cordeiro ," Clustering Very Large Multi-dimensional Datasets with MapReduce, "KDD'11, August 21–24, 2011, San Diego, California.
[13] Frank Lin frank, William W.," Power Iteration Clustering",International Conference on Machine Learning, Haifa, Israel, 2010.
[14] Hadoop Page on Mahout. http://mahout.apache.org/, 2011.
[15] Hadoop Page on Disco. http://discoproject.org/, 2011.
[16] Hadoop Page on Pig. http://pig.apache.org/, 2011.
[17] Hadoop Page on Hive. http://hive.apache.org/, 2011.
[18] F. Lin, W.W. Cohen, Power iteration clustering, in: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010.
[19] Apache Hadoop, http://hadoop.apache.org/