DATA PROCESSING WITH MAPREDUCE

MUKUNDKUMAR JHA, DEEPAK KHUSHWAHA, NAWAZ ASLAM, AMARJEET SINGH YUMNAM

***Abstract*— Attribute reduction for big data is viewed as an important preprocessing step in the areas of machine learning, Information mining and pattern recognition. MapReduce is useful programming model for data-intensive distributed parallel computing. In this paper, a K-mean clustering based MapReduce is proposed. This approach will help to do classification more accurately. Our model with mapreduce operations allows us to parallelize large computations effortlessly.**

***Index Terms*—Big-data; K-Mean; MapReduce; Parallel Processing.**

I. INTRODUCTION

In past decade, this a huge growth in the field of technology. With this growth the amount of data available has grown exponentially. This gives a need for finding useful information in large data. MapReduce is used by Google as a Parallel programming model and a framework for processing big-data. As data clustering has attracted a major quantity of analysis attention, several clustering algorithms have been proposed within the past decades. However, the enlarging data in applications makes clustering of terribly massive scale of knowledge a difficult task.

Apache Hadoop being a well-liked data processing platform for data, several data processing algorithms are migrating towards Hadoop.

II. RELATED WORKS

Many systems provide limited programming models and used the restrictions to put the computation mechanically. for instance, associate degree associative operate are often computed over all prefixes of associate degree N part array in ln(N) time on N processors exploitation parallel prefix computations [1, 5, 7]. MapReduce are often thought-about a simplification and distillation of a number of these models supported our expertise with massive real-world computations. Extra significantly, the greater part of the data process frameworks have alone been constrained on littler scales and leave the most purposes of taking care of machine disappointments to the architect.

Our neighborhood optimization attracts its inspiration from techniques like active disks [6, 8], wherever computation is pushed into processing parts that are near native disks, to scale back the number of knowledge sent through I/O network.

The sorting facility that's a locality of the MapReduce library is similar operating to NOW-Sort [2]. Supply machines (map

employees) partition the information to be sorted and send it to 1 of R scale back workers. every scale back employee types its knowledge regionally (in memory if possible). in fact NOW-Sort doesn't have the user-definable map and scale back functions that build our library wide applicable.

BAD-FS and TACC [4] are 2 different systems that have self-assurance re-execution as a mechanism to implement fault tolerance.

The original paper features an additional complete treatment of connected work [3].

Density-based ways assume that a cluster may be a knowledge house region within which the component distribution is dense. every region might have associate degree capricious form and also the parts within it should be every which way distributed. A cluster is separated from the others by regions of density, whose points area unit thought of as noise. The algorithms use own heuristics to spot dense and non-dense regions, typically wishing on user-defined density thresholds. Examples of such algorithms area unit circle, COPAC, P3C etc.

III. MAPREDUCE

MapReduce may be a framework for process parallelization issues across large datasets employing a sizable amount of computers, grouped to form cluster (if all nodes area uses similar hardware unit and are on constant native network) or a grid (if the nodes area unit shared across geographically distributed systems, and utility a lot of heterogeneous hardware). Machine procedure happens on a data keep either in an exceedingly filesystem or in an exceedingly data. MapReduce takes fortunate thing about region of learning; procedure it on or near to the capacity ownership to scale back the hole over that it ought to be transmitted.

a) Map step

The master node divides the input/problems into sub-problems, and distributes the sub-problem to nodes. a piece node may try this over again successively, leading to multi-level tree structure. Slave node processes the data, and passes result back to its master node.

b) Reduce step

Master node collects the solutions of all the sub-problems and combines them in an exceedingly thanks to create the output – the solution to the matter it had been originally creating a shot to unravel.

Exchange approach to appear at MapReduce is, As a 5-stage distributed and parallel processing:

- Prepare the Map() input – the "MapReduce system" designates Map processors, assigns the input key worth K1 that each processor can work, and provides that processor with all the computer file associated with that key worth.
- Run the user-provided Map() code – Map() is run exactly once for each K1 worth and generates output organized by K2

ISSN: 2278 – 7798

- Shuffle the Map output to the size back processors–the MapReduce system designates reduce processors, assigns the K2 key value every processor to figure and provides that processor with all the Map-generated data associated with that key worth.
- Run the user-provided Reduce() code – Reduce() is run exactly once for each K2 key value produced by the Map step.
- turn out the ultimate output – the MapReduce system collects all the output, and , and sorts it using K2 to produce the final outcome.

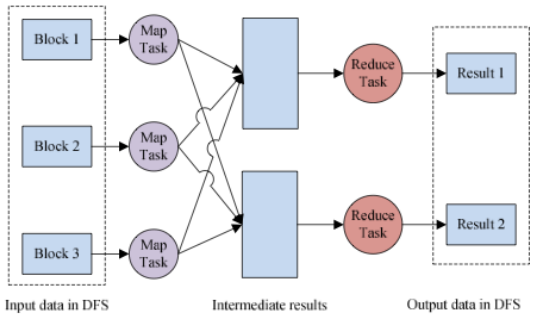

Fig. 1: MapReduce programming model

IV. K-MEANS CLUSTERING

K-means Clustering, k-means gathering is a method for vector quantization, at first from sign changing, which is standard for group examination in data mining. k-means grouping intends to group n sample into k bunches in which every perception fits in with the group with the closest mean, serving as a model of the group. This results in a dividing the information space into Voronoi cells.

The issue is computationally troublesome (NP-hard); of course, there are profitable heuristic estimations that are normally used and meet quickly to a close-by perfect. These are normally like the desire augmentation calculation for mixtures of Gaussian circulations by means of an iterative refinement methodology utilized by both calculations. Also, group focuses to model the information; on the other hand, k-means grouping has a tendency to discover groups of tantamount spatial degree, while the desire augmentation instrument permits bunches to have diverse shapes.

V. PROPOSED MODEL

A K-mean based MapReduce is being proposed. The algorithm for map and reducer are as follow:

Algorithm 1: Map (key, value)

Input: Variable centers, the key, sample values

Output: <key1, Value1> pair, where key1 is index of nearest center point and Value1 is sample values.

- Read center value from sequence file.
- Iterate for input key/value over centers.
- Measure Euclidean separation in the middle of center and sample value and store nearest center as key and sample value as value.
- Write center and its value in file system.
- End.

Algorithm 2: Reduce (Key, Value)

Input: key is the center; value is the sample values associated with each center

Output: <key2, Value2>, where key2 is the new cluster center and Value2 is the assigned sample values to cluster.

- Read key and its value from file
- Iterate over each sample values for the manipulating new center as an average of sample values of that cluster center.
- Compose new focus and it comparing values in sequential file.
- Compare between old center and new center
- If centers are not equal, increase center update counter and repeat above three step.
- Else come out of the loop and write the final list of center and its value in file system.
- End.

Once this clustering is over a MapReduce is performed over each cluster formed in the above steps. K-Mean provide cluster on small scale date such as 500 MB – 1 GB but consume time as it run sequentially. By using MapReduce framework for k-Mean, processing time and classification quality.

VI. RESULTS

K-mean clustering has been implemented in both java and in MapReduce on a single node cluster with a configuration of 2gb physical memory and dual core processor. By calculating the processing time of each sample data we perform analysis. The processing is done on a single node.

The processing time on a single node given bellow:

	No of record	time(millisecond)	
		Java	MapReduce
1	1300	111	9874
2	2600	204	11954
3	10500	486	10928
4	21000	480	11670
5	42000	584	12007

ISSN: 2278 – 7798

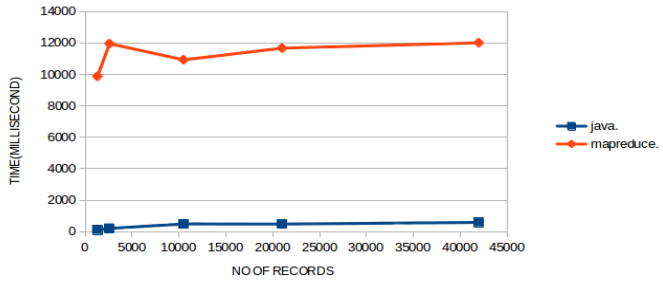

Fig. 2 processing time vs. no of records

MapReduce is implemented on a single node, the results of first mapreduce is stored in a file thus much time is consumed by mapreduce in reading records from file while iterating the record. Since we are using a single node cluster we can further more improve the processing time adding more node in the cluster.

VII. CONCLUSION

As data clustering has attracted a significant no of analysis attention, several clustering algorithms are proposed within the past decades. However, the enormously increasing data in applications makes clustering of terribly massive scale of data a challenging task. During this paper, we tend to propose parallel k-means clustering algorithm based on MapReduce that has been wide embraced by each domain and business. We tend to use acceleration, scale-up and speed-up to evaluate the performances of our proposed algorithm.

REFERENCES

[1] Blelloch, G. E. 1989. Scans as primitive parallel operations. IEEE Trans. Comput. C-38, 11.

[2] Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., Culler, D. E., Heller- stein, J. M., and Patterson, D. A. 1997. High-performance sorting on networks of workstations. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. Tucson, AZ.

[3] Dean, J. and Ghemawat, S. 2004. MapReduce: Simplified data processing on large clusters. In Proceedings of Operating Systems Design and Implementation (OSDI). San Francisco, CA. 137-150.

[4] Fox, A., Gribble, S. D., Chawathe, Y., Brewer, E. A., and Gauthier, P. 1997. Cluster-based scalable network services. In Proceedings of the 16th ACM Symposium on Operating System Principles. Saint-Malo, France. 78-91.

[5] Gorlatch, S. 1996. Systematic efficient parallelization of scan and other list homomorphisms. In L. Bouge, P. Fraigniaud, A. Mignotte, and Y. Robert, Eds. Euro-Par'96. Parallel Processing, Lecture Notes in Computer Science, vol. 1124. Springer-Verlag. 401-408

[6] Huston, L., Sukthankar, R., Wickremesinghe, R., Satyanarayanan, M., Ganger, G. R., Riedel, E., and Ailamaki, A. 2004. Diamond: A storage architecture for early discard in interactive search. In Proceedings of the 2004 USENIX File and Storage Technologies FAST Conference.

[7] Ladner, R. E., and Fischer, M. J. 1980. Parallel prefix computation. JACM 27, 4. 831-838.

[8] Riedel, E., Faloutsos, C., Gibson, G. A., and Nagle, D. Active disks for large-scale data processing. IEEE Computer. 68-74.

Mukundkumar Jha received the M. Tech degree in software technology from V.I.T University in 2015. Area of interest: Big-Data analytics, Cloud Computing, Data Mining, Machine Learning.

Deepak Kushwaha received the M. Tech degree in software technology from V.I.T University in 2015, and is currently working toward the PhD degree in the College of Computer Science,V.I.T university. His research interests include data privacy protection, Data mining, and personalized information retrieval.

Nawaz Aslam received the M. Tech degree in software technology from V.I.T University in 2015, and is currently working toward the PhD degree in the College of Computer Science,V.I.T university. His research interests include data privacy protection, Data mining based on cloud computing.

Amarjeet Singh Yumnam received the M. Tech degree in software technology from V.I.T University in 2015, Area of Interest: Cloud Computing, Data Mining.

ISSN: 2278 – 7798