Implementation of Extended MapReduce for Emerging BigData Analytics

Nayana N Kumar, Jayashree L K
Department of Computer Science,
Vemana Institute of Technology, Bengalaru-34, Karnataka
nayana.nayu14@gmail.com, jayashree_nov26@yahoo.com

ABSTRACT-Now a day‘s there is a need for managing huge amount of data in a faster rate is due to modern internet applications. As new upcoming data are arriving continuously, the result of data mining application becomes out of date over a period of time. This is one of the challenging jobs for computer organization to come out with new techniques and idea for handling data processing on large datasets at optimum response times. To manage vast amount of data it is required to refresh the results of mining which avoids the re computation cost from scratch. MapReduce is a technique which works based on several processors to provide automatic parallelization and distribution of computation. MapReduce framework and its open-source implementation Hadoop provides large computation environment for analysis of large-scale dataset processing and handling of dataset. This paper is focused on framework called ExtendedMapReduce, to manage the iterative operations by using map and reduce functions automatically without the user‘s involvement. After number of iterations small amount of updates may propagates to affect larger number of intermediate states. Which helps in improving the job running time and decreases the Incremental interactive processing time of refresh the result of big data.

Keywords –***Data mining, ExtendedMapReduce, Incremental interactive process, Hadoop, MapReduce***

I. INTRODUCTION

Many industries and organization uses big data processing technique, which is one of the trusted technologies to handle increasing amount of huge data. To improve the quality of existing services and to provide attractive new service it is important to extract important and valuable information from vast data sets, such as analysis of web-data, processing of logs and click analysis. Evaluating the huge amount of data obtained from different sources create a big challenges to the fields of science, mainly those involving massive-scale simulations and sensor networks.Modern Internet applications have created a need to manage immense amounts of data quickly. For example in social networking sites, the data produced by the user is increasing very fast every year. Big data is one of the popular techniques to take business decisions and to give better quality services. The information which is sorted and filed in the server of the organization was just data until yesterday .Suddenly the term BigData became famous and now the data filed in the company is nothing but BigData.

Involving different devices and applications data is produced which is called BigData. BigData is important in giving more accurate analysis, which may direct to more actual decision making and resulting in reduction in cost, decreased risk for business and greater operational efficiencies. To keep the extracted data up to date, periodical refreshing of mining computation is required in many situations. For analyzing BigData many frameworks have been designed. MapReduce [1] is one of the simple, generalized, framework built on Hadoop which is used for several productions. MapReduce implementation is done in large collection of computers to perform common calculations on large scale data effectively. This supports computation during hardware failure.

This paper is mainly focused on improving MapReduce technique. It supports incremental iterative process to timely accommodate new changes to the underlying data sets.

ExtendedMapReduce is advanced MapReduce technique and most sophisticated iterative computation to support key-value pair level incremental processing and extensively used in data mining applications. ExtendedMapReduce supports advanced approach called Incremental processing [2] [3], to refreshing mining results. Given the size of the input BigData it becomes very difficult to return the whole computation from scratch. Incremental processing technique takes new data from large data set, consider this as state as implicit input and combines it with new data.

II. MAP REDUCE BACKGROUND

MapReduce is one of the trusted techniques of computation; it handles large scale computations which help to stand the hardware faults. MapReduce has two main functions, called Map and Reduce. It splits the input data-set into independent chunks, and it is processed completely in parallel manner. For MapReduce computations refer Fig.1.

MapReduce supports the parallel execution, coordination of tasks that execute Map or Reduce, and also in handling one of the task that fails to execute. Key-Value pairs <K, V> sequences are turned into chunks by map task. By using input data, key-value pairs are produced, which is determined by the code written by the user for the Map function.

Each map task contains key-value pairs, are composed by a master controller and sorted by key. In reduce stage keys are divided among all, so the same reduced task wind up with same key with all key-value pairs. The reduce task processed on one key at a time and join all the values linked with that key in some way. For the reduce function the way of grouping of values is done by the code written by the user.

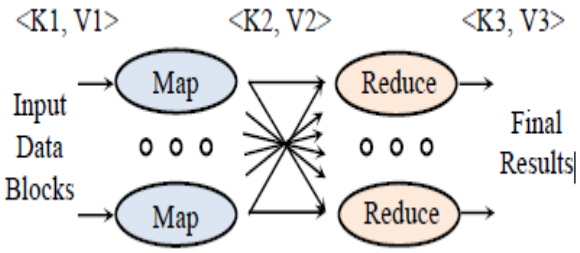

Figure 1.MapReduce computation

III. SYSTEM ARCHITECTURE

The system architecture of ExtendedMapReduce is shown in Fig.2.The implementation of ExtendedMapReduce is executed on a huge dataset and is extremely scalable. And this system can processes many terabytes of data on thousands of machines. Let us consider online data sets, by using ETL process (Extract, transform and Load), it reads the data and sends to apache Solr [4].Apache Solr is one of the open source API (Application Program Interface),is also called as NO-SQL Database, it is used as search engine for big data and accommodate data in XML(Extensible Markup Language). The performance analysis and difference with basic MapReduce and ExtendedMapReduce is shown by using K-Means Algorithm with Map Reduce Algorithm. Then for offline data set, PageRank algorithm along with MapReduce algorithm is used. For example in student database in stand Ford University the analytics is done on the student data along with iterative Algorithm implementation.

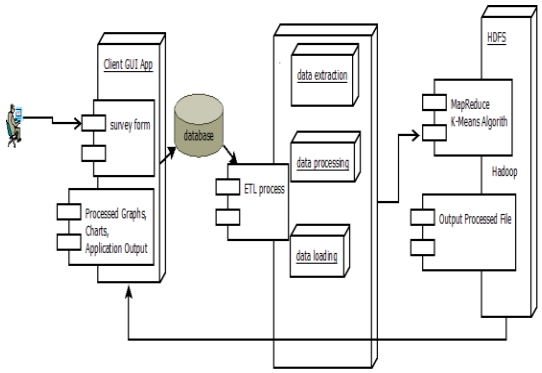

Figure 2.System Architecture

IV. IMPLEMENTATION

Analyzing Iterative Computation

PageRank [5] is a well suited web graph ranking algorithm. For each vertex in a graph ranking score will be computed by PageRank. The MapReduce job iteration is performed only after initializing all ranking scores which is shown in algorithm 1. In hyperlinked set of web pages, the PageRank calculates the numerical value for each element which reflects the probability that a random surfer will access that page.The method of PageRank can be considered as a Markov Chain [6] which needs iterative calculations to converge. Based on values calculated in the previous iteration, iteration of PageRank calculates the new access probability for each webpage. This process will continue till, the number of present iterations is larger than predefined maximum iterations, or the Euclidian distance between rank values in two subsequent iterations is less than a predefined threshold that handle the accuracy of the output results.

PageRank and MapReduce well suited with each other. Working procedure of these is as given below. It initiate with big data set called as D that has been divided into number of blocks i.e D1, D2…Dm. These blocks are circulated across different machines, such that each blocks on one machine. Let us consider Di is on machine i. These blocks are also replicated, MapReduce can ignore this. The main lamination to consider is that each machines has less memory compared to D. Now proceeds in rounds, each with 3 steps.

1. **Mapper:** Translate all d ∈ D to (key (d), value (d))

2. **Shuffle:** Moves all (k, v) and (k 0, v 0) with k = k 0 to same machine.

3. **Reducer:** Transforms {(k, v1), (k, v2) . . .} to an output D0 k = f(v1, v2, . . .).

Combiner: If one machine had multiple number of key value pairs i.e. (k, v1), (k, v2) with same key k, than reduce will perform before shuffle.

PageRank on MapReduce: v1 here is a first step. Break M into k vertical stripes M = [M1 M2 . . . Mk] so each Mj fits on a machine. Break q into q T = [q1 q2 . . . qk] (a horizontal split), again so each qj fits on a machine with M_j (This can be assumed how the data is stored, or can be done in a earlier round of MapReduce if not.) Now in each round:

• **Mapper:** j → (key= j 0 ∈ [k]; value = row r of M_j ∗ q_j)

• **Reducer:** adds values for each key i to get qi+1[j] ∗ β + (1 − β)/n.

The output of each mapper is considered as whole vector q_{i+1} or length n, each stripe M_j has n/k full columns. This process is feasible because qi+1 on has as many non-zero entries as Mj. However, it is not getting that much out of the combiner phase. It will see next how this can be improved.

PageRank on MapReduce: v2

Let √ k and tile M into ` × ` blocks

M = M1,1 M1,2 . . . M1,` M2,1 M2,2 . . . M2,` . . . . . M`,1 M`,2 . . . M`,`

• **Mapper**: Each of k machines get one block Mi,j and get sent q_i for i ∈ [`]. • Reducer: On each row i_0 adds Mi,j qi to q[i_0]. Then does q+ [i_0] = q[i_0]β + (1 − β)/n.

K-means Clustering

K-means [7] is the most common and well used algorithm in clustering method. It takes input as parameter k and split a set of objects into k clusters. The result of intra-cluster is high and inter cluster similarity is low. The similarity of cluster is calculated using mean value of objects in the cluster.

The algorithm contains following steps. Firstly, it randomly chooses k objects from the entire objects, it represents the

initial cluster centers. Based on the distance between the object and the cluster center the remaining object is assigned to the cluster to which it is the most similar. The new mean value for each cluster is calculated. This process iterates until the criterion function converges. In this algorithm, calculation of distances is considered the most intensive calculation. For iteration it is required to compute total distance (nk), where n is a number of objects and k is the number of clusters being created. It is not relevant to compute distance between one object with the centers to the distance computations between other objects with the corresponding centers. So distance computations between different objects with centers executed parallel.

In iteration, the new centers, used in the next iteration, should be updated. Hence the iterative procedures serially executed.

To identify and handle the input and output of the implementation is the first step in designing the MapReduce for K-Means. The input is provided as a key-value pair, where _key‘ is the center of cluster and _value‘ is the serializable implementation of vector in the data set. Once it set the cluster and selects the centroids, and defined the data vectors that are to be clustered properly, arranged in two files then the K-Means clustering technique can be used along with Map and Reduce technique. The input directory of HDFS (Hadoop Distributed File System) prior to Map routine is contains set of centers .They form a _key‘ field in the pair.

Mapper routine [8] is coded with the instruction needed to calculate the between the given data set and cluster center fed as a pair. The Mapper computes the distance between the vector value and every cluster centers mentioned in the cluster set. At the same time it also keeps track of the cluster which provide closet vector. After completion of calculation of distance, the vector is assigned to the cluster which is nearest.

It needs two main file to implement the Map and Reduce .One that houses the clusters with their centroids and another is houses the vectors to be clustered. Once Mapper is triggered the specified vector is set to the cluster that it is closest related to. After completion of this task, the recalculation is done on centroid of that, particular cluster. The reduce routine perform recalculation and prevent creations of clusters with extreme sizes (cluster having too less data vectors or a cluster having too many data vectors) by restructuring the cluster At the final stage, the centroid of the given cluster is updated, and re written the new set of vectors and clusters on disk which is ready for the next iteration. After understanding of what the input, output and functionality of the Map and Reduce routines it design the Map and Reduce classes by following the algorithm discussed below.

Algorithm 1. map (key, value)

Input: Global variable centers, the offset key, the sample value

Output: pair, where the key‘ is the index of the closest center point and value‘ is a string comprise of sample information

1. Construct the sample instance from value;
2. minDis = Double.MAX VALUE;
3. index = -1;
4. For i=0 to centers.length do dis= ComputeDist(instance, centers[i]); If dis < minDis { minDis = dis; index = i; }
5. End For
6. Take index as key‘;
7. Construct value‘ as a string comprise of the values of different dimensions;
8. output < key , value > pair; 9. End

Algorithm 2. reduce (key, V)

Input: key is the index of the cluster, V is the list of the partial sums from different host Output: < key , value > pair, where the key‘ is the index of the cluster, value‘ is a string representing the new center

1. Initialize one array record the sum of value of each dimensions of the samples contained in the same cluster, e.g. the samples in the list V ;
2. Initialize a counter NUM as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){ Construct the sample instance from V.next(); Add the values of different dimensions of instance to the array NUM += num;
4. }
5. Divide the entries of the array by NUM to get the new center‘s coordinates;
6. Take key as key‘;
7. Construct value‘ as a string comprise of the center‘s coordinates;
8. output < key , value > pair;
9. End

V. RESULTS

Finally comparing difference between Normal MapReduce and ExtendedMapReduce for different applications like K-means clustering and PageRank the Overall performance analysis of ExtendedMapReduce is as shown in Fig. 3.In every iteration, map function avoids reading and parsing the structure data by splitting the structure and state data. With incremental processing, the performance will improved in ExtendedMapReduce, so reducing the Normal MapReduce by 98%. When compared to Normal MapReduce, the advanced ExtendedMapReduce shuffles and input changes will affect the intermediate kv-pairs from the Map instances. Thereby further improving the shuffle time, achieving 95% reduction of NormalMapReduce time. For the sort stage, ExtendedMapReduce sorts the little number of kv-pairs from the changed Map instances, thus eliminating almost all sorting cost of NormalMapReduce. For the Reduce stage, iterMapReduce cuts the run time of Normal MapReduce by 80% because it does not require combining the updated state data and the structure data.

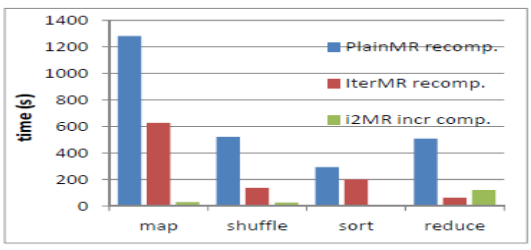

Figure 3 Performance of ExtendedMapReduce.

CONCLUSION

The ExtendedMapReduce is advanced MapReduce technique that supports the iterative processing for large datasets and give solution to the issues which arises during MapReduce implementation of iterative processing. It offers an environment and a model to the programmers for designing and to perform explicitly for iterative algorithms and moreover suggesting the concept of persistent tasks to carry out the iterative computation to keep away from continually creating, destroying, and scheduling tasks. In same iteration, it can also perform asynchronous execution of tasks, to accelerate the processing speed.

FUTURE ENHANCEMENT

As a result of the incremental processing, the MRBGraph file may contain multiple segments of sorted chunks, each resulting from a merge operation. So this situation needs to be improved and there is a need to enhance the query algorithm with a multi-window technique to efficiently process the multiple segments.

REFERENCES

[1] J. Dean and S. Ghemawat, ―Mapreduce: simplified data processing on large clusters,‖ *in Proc. of OSDI '04, 2004.* [2] D. Peng and F. Dabek, ―Large-scale incremental processing using distributed transactions and notifications,‖ in *Proc. Of OSDI '10, 2010, pp. 1–15*.

[3]P.Bhatotia, AWider R and R.Pasquin,‖Incoop:Mapreduce for incremental computations‖,in *Proc. Of SOCC '11,2011*

[4] Apache giraph. http://giraph.apache.org/.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mc-Cauley, M. J. Franklin, S. Shenker, and I. Stoica, ―Resilient distributeddatasets: A fault-tolerant abstraction for. in-memorycluster computing,‖ in *Proc. of NSDI '12, 2012*.

[6]S. Ewen, K. Tzoumas, M. Kaufmann, and V. Markl, ―Spinning fast iterative data flows,‖ PVLDB, *vol. 5, no. 11, pp. 1268–1279, 2012.*

[7] Y. Zhang, Q. Gao, L. Gao, and C. Wang, ―imapreduce: A Distributed computing framework for iterative computation,‖ J. Grid Comput., *vol. 10, no. 1, 2012.*

[8] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, ―Haloop: efficient iterative data processing on large clusters,‖ PVLDB, *vol. 3, no. 1-2, pp. 285–296, 2010.*

Biographies and Photographs

Nayana N Kumar received the B.E. degree in 2014 from Visvesvaraya Technological University. Currently, she is an M.Tech. Candidate in computer science, Vemana IT Bengaluru Karnataka. Her research interests include Data mining and BigData processing.

Jayashree L K is an Asst. professor at Vemana IT. And working from past 15 years in Dept. of CSE, Vemana IT Bengaluru Karnataka. Her current research interests are data ware housing and analysis of BigData.