ISSN (e): 2250 – 3005 || Volume, 08 || Issue, 7|| July – 2018 ||
International Journal of Computational Engineering Research (IJCER)

Dimensionality Reduction for High Variant Data Using Spark with RDD Transformation

D. Ashok Kumar[1], P. Velayudham[2]
1 Head, Department of Computer Science, Government Arts College, Trichy-620022.
2Department of Computer Science, Government Arts College, Trichy-620022.
Correspondence Author: D. Ashok Kumar

ABSTRACT
Clustering is a classification of patterns into small clusters. The main deviations of clustering techniques are quality of clusters. Papulation growth increase the volume of data more and more the classification is suffer from technically and mechanically The technical advancement to solve the high dimensional problem with the help of MapReduce Tool. The Mapper and Reducer is a framework for currently used the de facto standard in both industry and academia for petabyte scale data analysis. There are typical MapReduce computation is to contribute to visualize large volume of data sets. At the same time the processing complexity growing in another hand. The author proposed new techniques to visualize the large volume of data with the help of scale programming with spark. The spark is one of the Hadoop key techniques to store data on a single machine and must be processed in parallel. But the existing MapReduce algorithms works on distributed fashion to reduce the volume of big scale data sets at the same time the reduction process takes large time and high cost and lots of storage. We proposed new algorithms frame work for Spark with RDD transformation similar to MapReduce framework except the quality of reduction and in-memory process within the expected time. The existing MapReduce technique only process the batch files, but the proposed techniques process the batch files, interactive, iterative and streaming file processing system.

KEYWORDS: Classification, MapReduce, Spark, RDD, Reduced Dimension.

Date of Submission: 22-06-2018 Date of acceptance: 07-07-2018

I INTRODUCTION

The digital world generates the massive amount of data for every single mints at the same time people need to track the required data easily for their fast moving multipurpose expected life style, The researchers always contribute their efficiencies and knowledge to innovate the modern techniques to analysis the large volume data. The data analysis has grown at an astonishing rate in recent years. The technology growth increases the data volume required storage and machine capacity also increased for every 10 years. The recent review for growth of technology is the CPU speeds increase every ten years, in year 1990 is 44 MIPS at 40 MHz, but in year 2000 it become 3,561 MIPS at 1.2 GHz and the year 2010 it becomes hundred times faster that is 147,600 MIPS at 3.3 GHz. The require storage growths also a backbone of technology for example the RAM Memory in year 1990, 64KB, in year 2000, 64MB, and the year 2010 , 8.32GB. The system disk capacity improvements in every ten years, the year1990 – 20MB, in year 2000 - 1GB, and the year 2010, 1TB. Due to this growth of data the researcher turn into parallel algorithms for data processing. The MapReduce [3], and its open source implementation Hadoop [20], have emerged as the standard platform for large scale distributed computation. The MapReduce perform the process up to petabytes of data. The Hadoop and their method used to analysis the massive amount of data. The visualization of large scale data is an important role of data analysis. The MapReduce algorithms to visualize the d dimensional data it required O(d) rounds, where d is the dimension of the inputs . Every round in a MapReduce computation can be very expensive and take more time because it often requires a massive amount of data (on the order of terabytes) to be transmitted from one set of machines to another. This is usually the leading cost in a MapReduce computation.[3] Therefore minimizing the number of rounds is essential for efficient MapReduce computations. The spark is one the leading tool for analysis of big data and their methods used to reduce the computation cost and time, and efficiently reduce the large variant dimensions and easy to visualize the user expectations.

Apache Spark is a general purpose cluster computing engine which is very fast and reliable. This system provides Application programing interfaces in various programing languages such as Java, Python, Scala. Spark is a cluster computing system from Apache with incubator status; this tool is specialized at making data analysis faster, it is pretty fast at both running programs as well as writing data. Spark supports in-memory computing, that enable it to query data much faster compared to disk-based engines such as Hadoop, and also it offers a general execution model that can optimize large scale data. Spark is advance and highly capable upgrade to Hadoop aimed at enhancing Hadoop ability of cutting edge analysis. Spark engine functions quite advance and different than Hadoop. Spark engine is developed for in-memory processing as well a disk based processing. This in-memory processing capability makes it much faster than any traditional data processing engine. For example project sensors report, logistic regression runtimes in Spark 100x faster than Hadoop Map Reduce. This system also provides large number of impressive high level tools such as machine learning tool MLib, structured data processing, Spark SQL, graph processing took GraphX, stream processing engine called Spark Streaming, and Shark for fast interactive question device. The above tools are organized into the figure 1 shown in below.

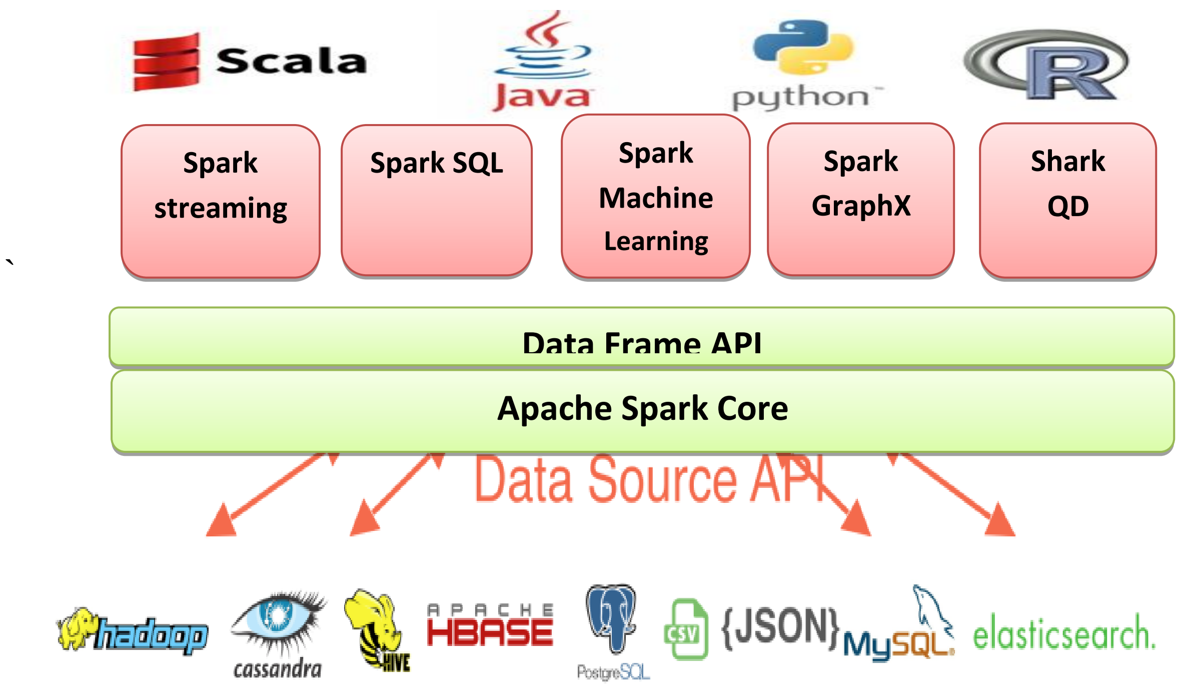

Figure -1 Apache Spark System Tools

II. LITERATURE REVIEW

II. 1. Nature of Big data

A very popular description for the exponential growth and availability of huge amount of data with all possible variety is popularly termed as Big Data[1][9][10][11][12]. This is one of the most challenge terms today's automated world and perhaps big data is becoming of equal importance to business and society as like the Internet usage. The people widely believed and proved that more data leads to more accurate analysis, and the more accurate indicate the analysis could lead to more legitimate, timely and confident decision making, as a result, better judgment and decisions more likely means higher operational efficiencies, reduced risk and cost reductions[2][7]. Big Data researchers visualize big data as follows:

i). Volume - This is the one of the most important factors for big data. Data volume is multiplying to various factors. Organizations and governments has been recording transactional data for decades, social media continuously pumping steams of unstructured data, automation, sensors data, machine to-machine data and etc. Formerly, data storage was itself an issues, but thanks to advance and affordable storage devices, today, storage itself is not a big challenge but volume still contributes to other challenges, such as, determining the relevance within massive data volumes as well as collecting valuable information from data using analysis[4] .

ii). Velocity- Volume of data is a big challenge at the same time which it is increasing is a serious challenge to be dealt with time and efficiency. The Internet streaming, RFID tags, automation and sensors, robotics and much more technology facilities, are actually driving the need to deal with huge pieces of data in real time. So velocity of data increase is one of big data challenge[6][8][9] with standing in front of every big organization today.

iii). Variety - Rapidly growing huge volume of data is a big challenge but the variety of data is bigger challenge. Data is growing in variety of formats, structured, unstructured, relational and non-relational, different files systems, videos, images, multimedia, financial data, aviation data and scientific data etc. Now the challenge is to find means to correlate all variety of data timely to get value from this data. Today huge numbers of organizations are striving to get better solutions to this challenge [4][13].
iv). Variability - Rapidly growing data with increasing variety is what makes big data challenging but up's and downs in this trend of big data flow is also a big challenge, social media response to global events drives huge volumes of data and it is required to be analyzed on time before trend changes. Global events impact on financial markets, this overhead increase more while dealing with un-structured data [5][12].
v). Complexity - The above factors make big data a really challenge, huge volumes, continuously multiplying with increasing variety of sources, and with unpredicted trends. Despite all those facts, big data much be processed to connect and correlate and create meaningful relational hierarchies and linkages right on time before this data go out of control. This pretty much explains the complexity involved in big data today [5]. To precise, any big data repository with following characteristics can be termed big data[6].
vi). Accessible - highly available commercial or open source product with good usability. Central management and scoring, Distributed redundant data storage, Extensible - basic capabilities can be augmented and altered, Extremely fast data insertion, Handles large amount as a petabyte of data[8] , Hardware agnostic, Inexpensive, Parallel task processing , and Provides data processing capabilities.

III. METHODOLOGY OF MAPREDUCE ALGORITHM

The MapReduce algorithms are perform in a two-step process. First step selectively drop, or filter the part of the inputs with the goal of reducing the problem size in to small enough to fit into a single machine memory. In the second stage the algorithms compute the final answer on this reduced input. The technical challenge is to choose enough points to drop but still be able to compute either an optimal or provably near optimal solution. The filtering step differs in complexity depending on the problem and takes a few slightly different forms.

The algorithmic process shows that the salient features of the MapReduce computing paradigm [13]. The input, and all intermediate data, is stored in key-value pairs and the computation proceeds in rounds. Each round is split in to three consecutive phases: map, shuffle and reduce. The map phase the input is processed one tuple at a time. All key-value pairs emitted by the map phase which have the same key are then aggregated by the MapReduce system during the shuffle phase and sent to the same machine. Finally each key, along with all the values associated with it, are processed together during the reduce phase. Since all the values with the same key end up on the same machine, one can view the map phase as a kind of routing step that determines which values end up together. The key act as address of the machine and the system makes sure all of the key-value pairs with the same key are collected on the same machine. We can combine the reduce and the subsequent map phase. Look in the computation for every round each machine performs some task in the set of key-value pairs assigned to it (reduce phase), and then designates which machine each output value should be sent to in the next round (map phase). The shuffle ensures that the data is moved to the right machine, after which the next round of computation can begin. In this simpler model, we shall only use the term machines as opposed to mappers and reducers. . The overview MapReduce system process is shown in figure 2.

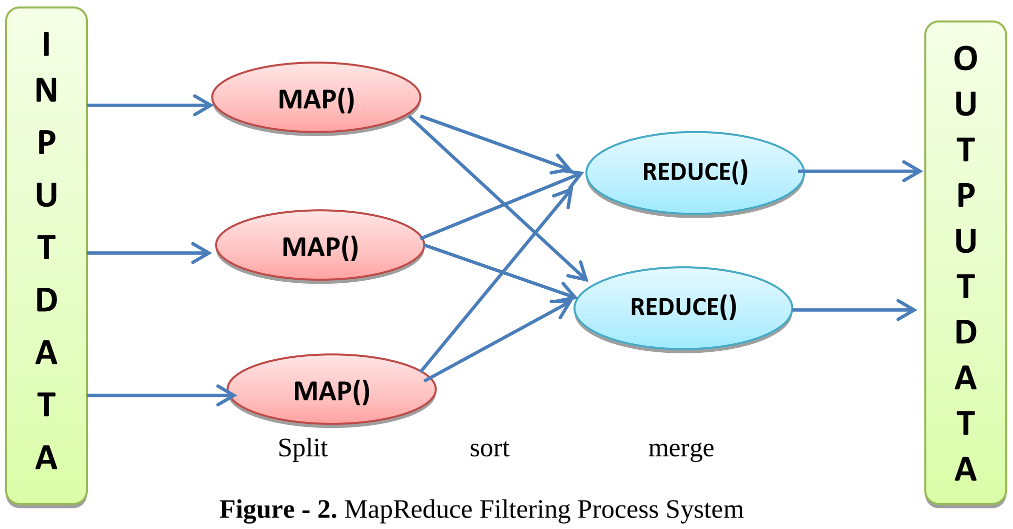

Figure - 2. MapReduce Filtering Process System

III. 1. HADOOP vs SPARK

Hadoop is a distributed file system and data processing engine that is designed to handle extremely high volumes of data in any structure. Hadoop is the combination of Hadoop File System and the MapReduce infrastructure. Spark is an in-Memory cluster computing techniques for iterative and interactive applications. The spark extends the MapReduce model to better support to data analytic applications. The spark enhance the programmability with the help of integrate the scale programming and its interpreter. Spark is popular programming models for clusters transform data flowing from stable storage to stable storage. The benefits of spark are the runtime can decide where to run tasks and can automatically recover from failures. Spark is an expressive computing system, to overcome the limited of map-reduce model. It facilitate system memory to avoid saving intermediate results to disk and cache data for repetitive queries (e.g. for machine learning) and more compatible with Hadoop. Spark is 100 times faster than Hadoop if fits in memory else 10 times faster. Spark performs the in-memory computation and advanced job execution engine of stages. Hadoop MapReduce creates a DAG with exactly two predefined stages called Map and Reduce for every jobs. In case of MapReduce perform multiple jobs required to process complex data here there is no optimization. But DAG in Spark can contain any number of stages with optimization of execution time.
MapReduce is the batch oriented computation system but in spark is batch as well as iterative and interactive and stream oriented computations. The spark Retain the attractive properties of MapReduce Fault tolerance (for crashes & stragglers), Data locality, and Scalability with the help of Resilient Distributed Datasets(RDD), and augment data flow model .The RDD only read the partition collection of records. The RDD transform includes map, filter and join. The RDD action includes count, collect and saves. RDD is immutability because it never changes. The RDD Application process flow shown in figure 3.

Spark Applications runs in four modes: First the standalone local mode, where all Spark processes are run within the same Java Virtual Machine (JVM) process. Second the standalone cluster mode, using Spark's own built-in job-scheduling framework Using Mesos, Third a popular open source cluster computing framework Using YARN (commonly referred to as Next-Gen MapReduce). Fourth the Hadoop related cluster computing and resource scheduling.

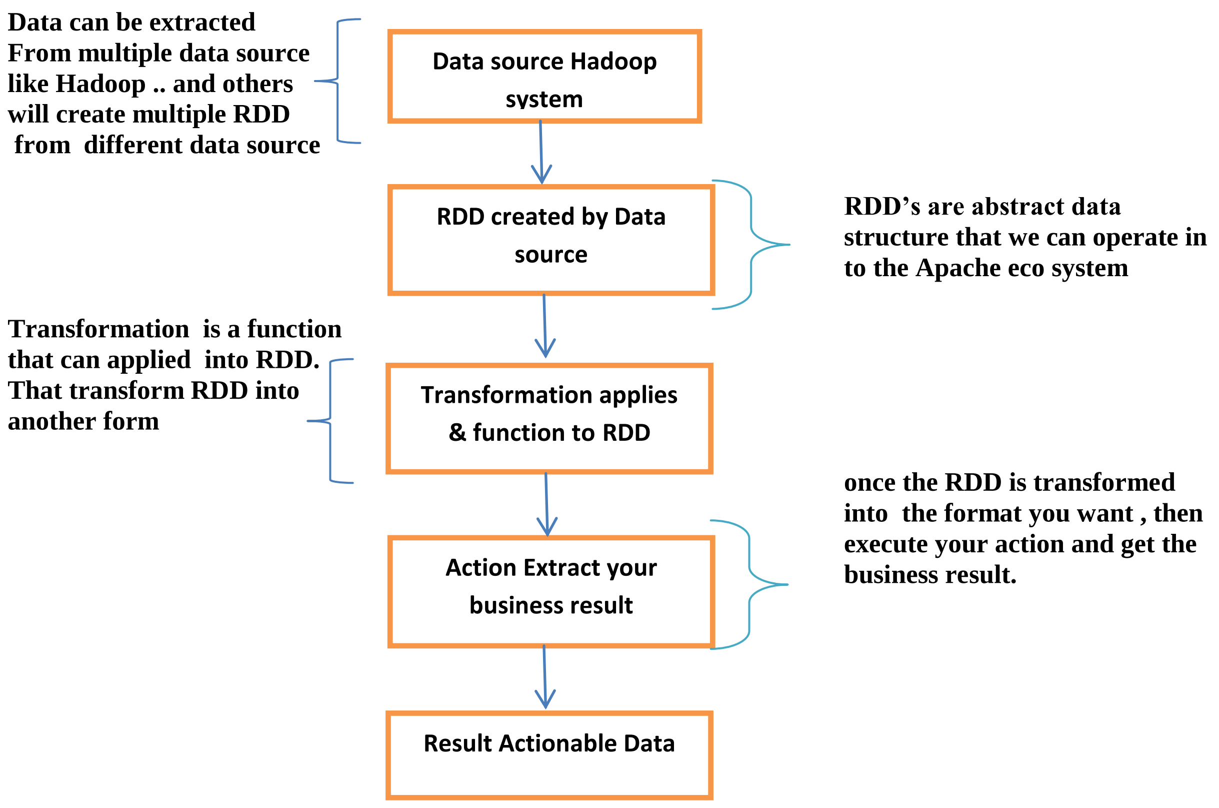
Data can be extracted From multiple data source like Hadoop .. and others will create multiple RDD from different data source
Data source Hadoop system
RDD created by Data source
RDD's are abstract data structure that we can operate in to the Apache eco system
Transformation is a function that can applied into RDD. That transform RDD into another form
Transformation applies & function to RDD
once the RDD is transformed into the format you want , then execute your action and get the business result.
Action Extract your business result
Result Actionable Data

Figure-3 Spark Application Process Flow

III. 2. SPARK-- RDD FILE OPERATIONS

The RDD perform the transformation, Lazy operations and Actions. The Transformation - Create a new dataset from an existing dataset, The Lazy operation - Operates when an action is run on it for eg: map, filter, flat map etc. The Actions Compute values, Return values or write O/P to external storage , Earlier transformations are applied to RDD, Since transformations are lazy operation. Example: Collect, Count, ForEach, top, reduce etc. The transformation operations are shown in Figure 4.

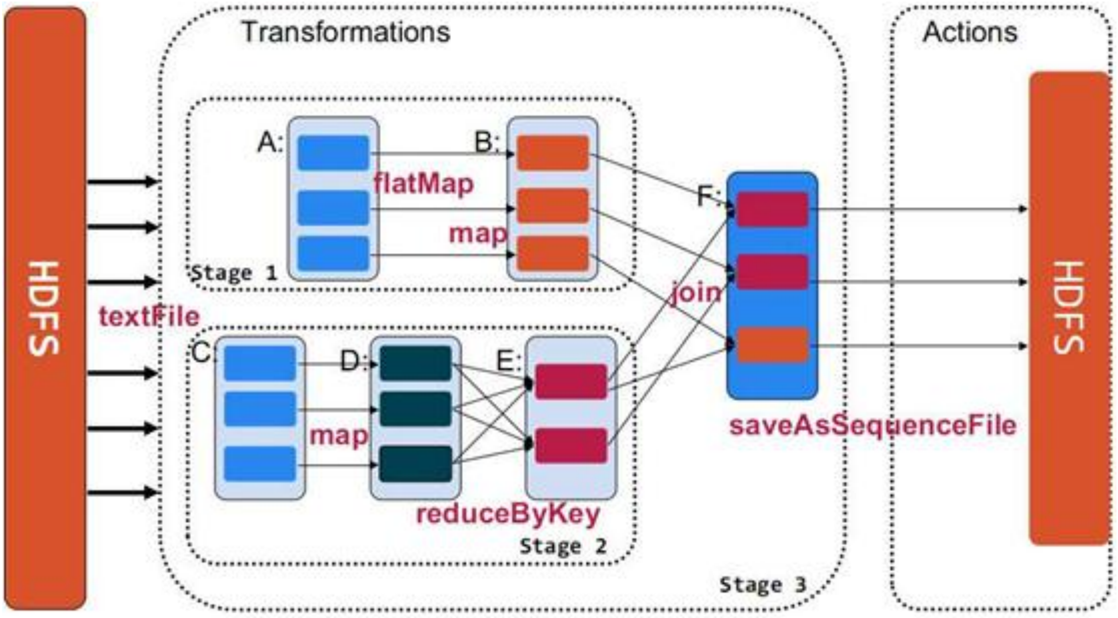

Figure-4 - RDD Transformation Operations.

III. 3. EXISTING SYSTEM

The Existing MapReduce model consists of better reduction process. The MapReduce filtering algorithms are perform in a two-step process. First step selectively drop, or filter, parts of the input with the goal of reducing the problem size to small enough to fit into a single machine's memory. In the second stage the algorithms compute the final answer on this reduced input. The technical challenge is to choose enough edges to drop but still be able to compute either an optimal or provably near optimal solution. The filtering step differs in complexity depending on the problem and takes a few slightly different forms. The MapReduce system required more rounds to complete a task that is why it required more time , more cost and more system memory. Main Memory in each round the total memory used by a single machine is at most O(N1−) bits. Total Memory: The total amount of data shuffled in any round is O(N2−2) bits .The number of rounds is a constant.

III. 4. PROPOSED SYSTEM

Hadoop is a distributed file system and data processing engine that is designed to handle extremely high volumes of data in any structure. Hadoop consists of Hadoop file system and MapReduce infrastructure Extend the MapReduce model to better support two common classes of analytics applications that are one for Iterative algorithms for machine learning and Interactive data analytics methods using spark. Spark RDD enhance the programmability to Integrate into Scala programming and interactive use from Scala interpreter. The Spark RDD transformation is best compare to MapReduce batch proceeding reduction in the way of in-memory batch processing as well as streaming, and interactive file processing system. It transform the large variant data within less time, low cost and capable to process in- memory system.

Proposed system is organized as follows. The Section IV Describe the MapReduce Algorithms and their deviations. The Section V Describe the Proposed RDD Transformation and Reduction process. The Section VI Describe the Experimental Results and Conclusion.

IV.MAPREDUCE ALGORITHMS

IV. I. Minimum Cut Algorithm

The previous MapReduce algorithms, the filtering was done by dropping certain edges, this algorithm filters by contracting edges. Contracting an edge, will obviously reduce the number of edges and may either keep the number of vertices the same(we contracted a self-loop), or reduce it by one. To compute the minimum cut of a graph we appeal to the contraction algorithm. The algorithm has a well-known property that the random choices made in the early rounds succeed with high probability; where as in the later rounds have a much lower probability of successor. We adventure this property by showing how to filter the input in the first phase (by contracting edges) so that the remaining graph is definite to be small enough to fit onto a single machine, yet large enough to ensure that the failure probability remains bounded. Once the filtering phase is complete, and the problem instance is small enough to fit onto a single machine, we can employ any one of the well-known methods to find the minimum cut in the filtered graph. We decrease the failure probability by running several executions of the algorithm in parallel, thus ensuring that in one of the copies the minimum cut survives this filtering phase. The author can show that even with many parallel invocations we will not violate the machine budget.

.

MapReduce Algorithm1- MinCut(E)

```
1: for i = 1 to nδ1 (in parallel)do
2: tag e ∈ E with a number re chosen uniformly at random from [0,1]
3: t ←Findt(E,0,1)
4: Ei ←Contract(E,t)
 5: Ci ←min cut of Ei
6: endfor
7: return minimum cut over all Ci
```

IV. 2. Find Algorithm

The pseudocode for the algorithm to find the correct threshold is given below. The algorithm invokes the copies of the connected components, each of which uses at most nc−machines, with n1+memory.

MapReduce Algorithm-2 - Findt(E,min,max)

```
1: {Uses nδ2+c/ machines.}
2: γ ← max−min nδ2
3: for j = 1 to nδ2 (in parallel)do
4: τj ← min+jγ
5: Ej ←{e ∈ E | re ≤ τj}
6: ccj ←number of connected components in G = (V,Ej)
7: endfor
8: ifthere exists a j such that ccj = nδ3 then
 9: return j 10: else 11: return Findt(E,τj,τj+1) where j is the smallest value s.t. ccj < nδ3,ccj+1 > nδ3 12: endif
```

MapReduce Algorithm 3 . Contraction Algorithm

state the contraction algorithm and prove bounds on its performance.

```
Algorithm3 Contract(E,t)
1: CC ←connected components in{e ∈ E | re ≤ t}
2: let h : [n] → [nδ4] be a universal hash function
3: map each edge (u,v) to machine h(u) and h(v)
4: map the assignment of node u to its connected component CC(u), to machine h(u)
5: on each reducer rename all instances of u to CC(u)
6: map each edge (u,v) to machine h(u) + h(v)
7: Drop self-loops (edges in same connected component)
8: Aggregate parallel edges
```

V. PROPOPSED ALGORITHM

The Spark RDD transformation is best compare to MapReduce batch proceeding reduction in the way of in-memory batch processing as well as streaming, and interactive file processing system. It transforms the large variant data within less time, low cost and capable to process in system memory. The proposed system each round the total memory used by a single machine is at most O(1) bits: The total amount of data shuffled in any round is O(N) bits . The computation time is very less compare to existing system..

Proposed Algorithm 1- Spark RDD Transformation Algorithm

```
1.Read the text from the SC context.
rddFromTextFile=sc.textFile("/home/ponny/a.txt")  rddFromTextFile.collect()
2.Action – Collect() // used for  retrieve the data into drive Spark Context – SC.
3.Action – Take() //Take (n) : Returns n elements from RDD the Number from any partition of RDD .
4. Action – Count() //Return number of elements in the current RDD
5.Action – Reduce() // Reduce the same data type values
6.Action - foreach //Takes a method for executing each element in the RDD.
7.Transformation – Map() //Takes a function and applies to each element in the RDD.
mappedData=distData.map(a.txt)
mappedData.collect()
8,Transformation – Filter  //Filter based on condition
filteredData=distData.filter(a.txt, condition)
filteredData.collect()
9.Key-Value Transformations
m = sc.parallelize([('Apple', 1), ('Orange',2), ('Apple',5)])
d=m.reduceByKey(a,text x,y:x+y)
10.Create key value pair RDD
map(transformation) //Converts RDD into mapped key value pair
 m.groupByKey().map(a,txt, x : (x[0], list(x[1]))).collect()
```

Proposed Algorithm 2: Spark: RDDFind Algorithm

Let's compute the preceding metrics: find to count the number of purchases in specified products and Find the best product using spark RDD transformation techniques.
Now have an RDD, where each record is made up of (user, product, and price), we can compute various interesting metrics for our store, such as the following ones:
• The total number of purchases
• The number of unique users who purchased
• Our total revenue
• Our most popular product.

```
1. val numPurchases = data.count()
 // let's count how many unique users made purchases
2. val uniqueUsers = data.map{ case (user, product, price) => user •
    }.distinct().count()
 // let's sum up our total revenue
 3.val totalRevenue = data.map{ case (user, product, price) => price. •
   toDouble }.sum()
 // let's find our most popular product
 4. val productsByPopularity = data
 5.   map{ case (user, product, price) => (product, 1) }
 6.  reduceByKey(_ + _)
 7.  collect()
.8.  sortBy(-_._2)
9.  val mostPopular = productsByPopularity(0)
```

Step 9 to compute the most popular product is an example of the Map/Reduce pattern made popular by Hadoop.
First, we mapped our records of (user, product, price) to the records of (product, 1). Then, we performed a reduceByKey operation, where we summed up the 1s for each unique product.
Collect: This returns the results of the computation to the driver program as a local Scala collection.

```
println("Total purchases: " + numPurchases)
println("Unique users: " + uniqueUsers)
println("Total revenue: " + totalRevenue)
println("Most popular product: %s with %d purchases". format(mostPopular._1, mostPopular._2)) • } •
# let's count the number of purchases
 numPurchases = data.count()
 # let's count how many unique users made purchases
 uniqueUsers = data.map(lambda record: record[0]).distinct().count()
 # let's sum up our total revenue
 totalRevenue = data.map(lambda record: float(record[2])).sum()
# let's find our most popular product
 products = data.map(lambda record: (record[1], 1.0)).reduceByKey(lambda a, b: a + b).collect()
 mostPopular = sorted(products, key=lambda x: x[1], reverse=True)[0]
 print "Total purchases: %d" % numPurchases
 print "Unique users: %d" % uniqueUsers
 print "Total revenue: %2.2f" % totalRevenue
 print "Most popular product: %s with %d purchases" % (mostPopular[0], mostPopular[1])
```

VI. PERFORMANCE EVALUATION

The Spark RDD transformation is best compare to MapReduce batch proceeding reduction in the way of in-memory batch processing as well as streaming, and interactive file processing system. It transforms the large variant data within less time, low cost and capable to process in system memory. The RDD perform the transformation, Lazy operation and Action. The Transformation Create a new dataset from an existing dataset, The Lazy operation. Operates when an action is run on it for eg: map, filter, flat map ...The Actions – Compute values , Return values or write O/P to external storage , Earlier transformations are applied to RDD. The proposed system each round the total memory used by a single machine is at most O(1) bits: The total amount of data shuffled in any round is O(N) bits .. The computation time is very less compare to existing system. Thus the above evaluations the proposed system is outperform from the existing MapReduce system.

The proposed model is validated using four parameters namely the Accuracy of the classifier, Area under ROC Curve, Sensitivity and Specificity.

TP (True Positive): The number of examples correctly classified to that class.

TN (True Negative): The number of examples correctly rejected from that class.

FP (False Positive): The number of examples incorrectly rejected from that class.

FN (False Negative): The number of examples incorrectly classified to that class.

$$\text{Accuracy} = \frac{T\,P + TN}{\text{Tot. No of Instances}}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{T\,N}{T\,N + FP}$$

TABBLE-1 CLASSIFICATION ACCURACY FOR REDUCTION ALGORITHMS

Classification Performance		Accuracy	Sensitivity	Specificity	Error Rate	Processing Time	Complexity
MapReduce Algorithm	MinCut	98	0.6	± 1.50	± 2.0	0.50345	O(N-1)
	Find	98.2	0.50	± 0.7	± 1.8	0.50321	O(N-2)
	Cont	98.1	0.52	± 0.5	± 2.0	0.40251	O(N-2)
Spark RDD Algorithm	RDDT	99	0.3	± 0..2	± 1.0	0.050251	O(N)
	RDDF	99.1	0.35	± 0.3	±0 .80	0.050267	O(N)

Experimental results show that MinCut had a poor classification performance compared with the other two MapReduce Find and Contraction Algorithmic schemes, and the classification accuracy of RDDT and RDDF was very close to each other. RDDF is a good tool for dimensionality reduction; RDDT and RDDF are close to considers the class sensitivity when it learns from the training samples to obtain a linear optimal matrix. Thus, RDDF and RDDT had a higher accuracy than MinCut and Contraction algorithm for complex computation to the reduction process; the processing time of these classification does in have big difference.

ROC describes the tradeoff between Sensitivity and Specificity, as well as the performance of the classifier, can be visualized and studied using the Receiver Operating Characteristic (ROC) curve shows in figure 5.

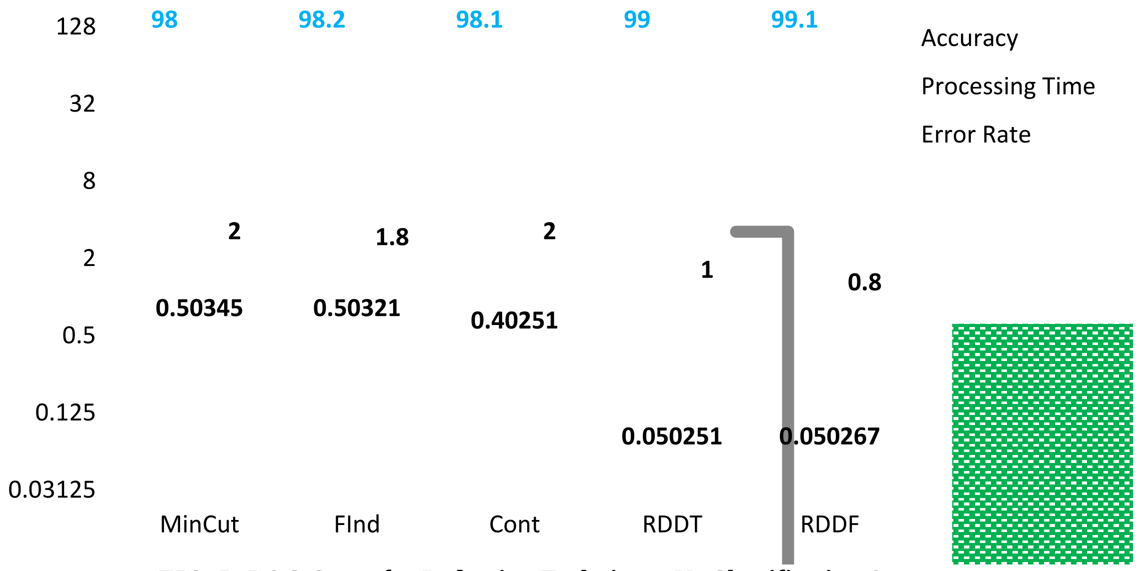

FIG. 5: ROC Curve for Reduction Techniques Vs Classification Accuracy

VI. CONCLUSION

Data Reduction is critical process for large variant data analysis and classification Techniques. The Existing MapReduce Techniques are more efficient for high dimensional data sets like big data sets. But the round strip process is more time consuming and intermediate machine transformation required more memory it means high memory usage MapReduce processed the. parallel processing with high fault tolerance. Due to this inefficient behavior of MapReduce techniques The author proposed a new techniques called spark with RDD transformation.. It is a well know reduction techniques for high dimensional data sets. it solve all the misbehavior of MapReduce Techniques and apply for Bach processing file as well as interactive and streaming file processing. We proposed RDD based algorithms RDDT and RDDF. These two transformation algorithms outperform for existing MapReduce Algorithms for its efficient classifications and processing time and required storage and error rates.. A significant part of the improvement in the running time comes from the fact that the proposed algorithms do a much better job of distributing the computation evenly across the machines, even in the presence of large skew in the input data. This allows them to avoid the "curse of the last reducer", where by the majority of machines in a distributed job finish quickly, and a single long running job determines the overall running time. The two algorithms we propose also diff erent theoretical behavior and exploit diff erent aspects of the spark infrastructure

REFERENCES

[1]. Parmita Mehta, Sven Dorkenwald, Dongfang Zhao, Tomer Kaftan, Alvin Cheung, Magdalena Balazinska, Ariel Rokem, Andrew J. Connolly, Jacob VanderPlas, and Yusra AlSayyad. "Comparative Evaluation of BigData Systems on Scientific Image Analytics Workloads". In: PVLDB 10.11 (2017), pp. 1226–1237.
[2]. Ovidiu-Cristian Marcu, Alexandru Costan, Gabriel Antoniu, and Mar´ıa S. P´erez-Hern´andez. "Spark Versus Flink: Understanding Performance in Big Data Analytics Frameworks". In: 2016 IEEE International Conference on Cluster Computing. 2016, pp. 433–442.
[3]. Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In Proceedings of SODA, pages 938–948, 2010.
[4]. Mayer-Schonberger V, Cukier K (2013) Big data: a revolution that will transform how we live, work, and think. Eamon Dolan/Houghton Mifflin Harcourt .
[5]. Jimmy Lin and Chris Dyer. Data-Intensive Text Processing with MapReduce. Number 7 in Synthesis Lectures on Human Language Technologies. Morgan and Claypool, April 2010.
[6]. Blake Irving. Big data and the power of hadoop. Yahoo! Hadoop Summit, June 2010.
[7]. HeY, LeeR, HuaiY, ShaoZ, JainN, ZhangX, XuZ. RCFile: A fast and spaceecient data placement structure in MapReduce-based warehouse systems. In: 27th International Conference on Data Engineering. IEEE; 2011b. p. 1199–208
[8]. Gantz J, Reinsel D (2011) Extracting value from chaos. IDC iView, pp 1–12 .
[9]. Fact sheet: Big data across the federal government (2012). http:// www.whitehouse.gov/sites/default/files/microsites/ostp/big data fact sheet 3 29 2012.
[10]. Lohr S (2012) The age of big data. New York Times, pp 11 .
[11]. Manyika J, McKinsey Global Institute, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers AH (2011) Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute.

[12]. Mayer-Sch¨onberger V, Cukier K (2013) Big data: a revolution that will transform how we live, work, and think. Eamon Dolan/Houghton Mifflin Harcourt.

[13]. Almeida, J.S. Grüneberg, A., Maass, W. and Vinga, S. (2012) 'Fractal MapReduce decomposition of sequence alignment', Algorithms for Molecular Biology, Vol. 7, No. 12, pp.1–12