An Approach of RDD Optimization in Big Data Analytics

Athira V
PG Scholar, Department of Computer Science, St. Josephs College of Engineering and Technology, Palai, Kerala, India
E-mail: athira24v@gmail.com

Jeena Thomas
Assistant Professor, Department of Computer Science, St. Josephs College of Engineering and Technology, Palai, Kerala, India
E-mail: jeena.thomas@sjcetpalai.ac.in

Abstract. We live in the information age and new information is produced in each and every second. The information can be worlds data which is responsible for major technological changes that can bring new ways in decision making. Keeping this data for further analysis and computation is a difficult task. Several studies are done in this arena, making it effective for future figuring. Processing or analysing such huge amount of data is a challenging task. All the existing technologies contain certain bottlenecks and performance overheads, diverse challenges like scalability. Spark is the commonly used data analysis framework. Map Reduce is a computing paradigm and a popular model for distributed data analysis. This paper gives a survey about some enormous information technologies, how it will deal with huge data, and the difficulties in existing advances, and has additionally learned about a portion of the execution bottlenecks and preventive techniques, and the concentration at that point moves to the Resilient Distributed Dataset (RDD), and how it is optimized.

1. Introduction
The new media age has witnessed the growing of todays enterprises in an exponential rate day by day, along with the explosion of data and the databases, this has caused a big data problem faced by the industries due to its inability to manage or process this data within the time limit. Data is generated through several social networking sites, as a result of several transactions. The amount of data generated can be structured as well as unstructured [1]. Processing or analysing such huge amount of data is a challenging task. The blowing up of data has created a major challenge in the field of science and engineering. Data sets are fast growing and the conventional methods lacks solutions to manage easily. Existing solutions use files or some based on storing in databases still fails in handling and analysing data properly and to make it use in future. As data sets exceeds the capacity of the system, its analysis gets difficult and performance also gets limited. Data analysis is done by evaluating certain attributes and necessary data is extracted and transformed, types of data varying from simple to complex ones are extracted and perform multiple complex joins to these datasets. Explosion of data size makes it inefficient to store and
process data, hence causing several challenges to preserve it for future computation. Challenges include limited scalability of I/O; scalability factor determines performance of application. Big data is large data that cannot be processed using traditional computing techniques. The volume of data Facebook and YouTube [19] handles will come under the category of big data. The data size varies from terabytes to petabytes of data as the data can be structured or unstructured. Big data is important as in this growing economy data is also growing in abundance due to the abundant uses of social networking sites, mobile and networking. More often there is a need to analyse this data to acquire the required information within a short time.

2. Related Works

Several researches are done aiming the improvement of performance of data analytics framework, most of them failed to understand properly the factors affecting the performance of the system. Kay Ouster hout [6] has developed a big data analytics framework for identifying the performance bottlenecks in a distributed computing framework, and have used it to analyse the performance of spark framework on two sql benchmarks and production workload, and he came to conclusion that cpu is often the bottleneck and by improving the network performance can improve the job completion time by 2 percent. In order to improve the performance hadoop and spark is widely used. Usually identifying performance bottlenecks is a problem due to parallelism, there will be multiple task performing.

Francisco Clemente-Castello and Bogdan Nicolae [7] discusses about enabling big data analytics in the hybrid cloud using iterative map reduce. There are different cloud computing models like private cloud, public cloud and a combination of private and public cloud called Hybrid cloud. Hybrid clouds can be made used in big data analytics and can hence improve the iterating level of map reduce applications. Using cloud can improve the environment like hiring some from off-premises also.

Bodgan Nicolae [9] introduced an approach that identifies and remove the data redundancy. This new approach minimizes the data that gets replicated, and compresses the data which is present above the required level using local storage instead of parallel file systems as it cause for a large amount of related data, and these local storage has limited capacity to hold data and also these local storages are prone to failures also, and if multiple requests arrive it will be difficult to handle in local storage, so to make high availability replication is usually done.

Nicolae [9] has also proposed redundancy elimination and replication as a co-optimized phase. In many of the high performance computing platforms [9] input and output will be the main performance bottlenecks. Yong chen and Xian-He sun [10] has proposed a collective input output strategy called layout aware collective input output (LACIO), which optimizes the input output performance and provides integration. Input output can cause performance degradation causing low latency, LACIO[10] improves the performance of the current systems, it does it by a certain file system calls, and hence improves the input output performance of parallel systems. Big data analytics plays an important role in many fields like science, medicine healthcare etc. [14]. As the data size has increased a lot there needed an efficient method for its computation and its analysis. In memory platforms always face challenges like data shuffling and it will diversely affect the performance and scalability of the system. Commonly used in memory platform is apache spark. There will be massive amount of data transfer that occur and some cases these platforms need to operate with scarce memory also. So to make shuffling easier efficient scheduling of the data transfer is required. There are many solutions exists that gives sub optimal performance and resource usage.

Bodgan Nicolae [11] gives a memory optimized data shuffling, this shuffle strategy dynamically adapts to the computation with minimum memory utilization and Aaron Davidson [8] discusses about optimizing shuffle performance in spark. Spark is a commonly used framework for performing in-memory computation. Shuffle performance in spark should be optimized so
that it performs better than hadoop. There are different bottlenecks available which affect the performance, the existing solutions have identified some bottlenecks. The method has identified bottlenecks and has proposed some alternatives to mitigate the operating system overheads associated with these bottlenecks. It is shuffle file consolidation, the new simple solution has led to two times improvement in overall job completion time.

In map reduce there are map and reduce phases and shuffling occurs in between them. Shuffling is a part of reduce phase and it has a little role to deal with data. Shuffling can cause operating system overheads, and it causes input and output over the network. Several researches are done to avoid the operating system overhead, since shuffle phase is not connected to the semantics of the data, additional storage can significantly reduce the overhead due to the data transferred. Reducer will combine the results with the use of combiners.

Xiaoyi Lu and Dipti Shankar [11] talks about enhancing features on spark to improve the performance in shuffle phase. RDMA [11] based apache spark on high performance network. Apache spark is the most commonly used method for inmemory processing of real time data, in apache spark data can be loaded in memory and can query it repeatedly. It gives an abstraction of RDD, as it supports data lineage. Shuffle phase caused overhead and many research works are going on to mitigate it. Xiaoyi Lu [11] has proposed a data transfer strategy in the shuffle phase, which consumes only least amount of memory, it has given a design of RDMA based Apache spark and also has proposed block transfer service plugin which supports three shuffle scheme sort, hash and tungsten sort in spark. It is expected that in future can remove all the bottlenecks in spark and introduce methods to improve performance.

| Author           | Proposed Approach                              | Aim                                               | Findings                                                                 |
|------------------|------------------------------------------------|--------------------------------------------------|--------------------------------------------------------------------------|
| 1) Key Ouster Host | A big data analytics framework                 | To identify performance bottlenecks in distributed computing framework | • CPU is the main bottleneck  
• Improving network performance improves job execution time |
| 2) Rodger Nicolson | Approach done to minimize data that gets replicated | To identify and remove data redundancy             | • Compressing data that exceed the storage level                         |
| 3) Youg Chou     | LACIO                                         | Optimizing input output performance and to provide integration | • Improves the performance of system  
• Include certain file system calls                                |
| 4) Aaron Davidson | Proposed Shuffle file consolidation            | To optimize shuffle performance                   | • Improved overall job execution time                                    |
| 5) Xiaoyi Lu     | A data transfer strategy in shuffle phase      | To remove all the bottlenecks.                    | • Consumes only less amount of memory  
• Support shuffle sort, hash and tungsten sort                       |

Figure 1. Review of Literature

3. Big data Technologies

3.1. Hadoop

Hadoop [16] is a fault tolerant, cost effective, flexible and scalable computing solution. Most of the industries use hadoop to analyse their dataset. It is an open source software, and allows distributed processing of large datasets across clusters of computer. It is a simple programming model, it involves HDFS (Hadoop Distributed File System) [16], which is a distributed file system providing fault tolerance and it runs on a hardware. It store large datasets across
multiple clusters of computer. It involves a master/slave architecture [18]. Its design is based on the Google File System [2]. It supports high-streaming read performance. It has a block structured file system, where files are divided into blocks, distributed and stored across different hadoop clusters. It has an open source implementation. There is a job tracker node also known as master node and a Task Tracker node called worker node. Client interact with job tracker, job tracker takes jobs from client and necessary operations according to the client request, decomposes the jobs into certain tasks. Task tracker contains slots and each slot will contain task. Task tracker receives task from job tracker and executes finally, and after completion it send notification back to the task tracker using a heartbeat message. If certain tasks are failed it executes the failed task again. Namenode manages and provides access to files by clients. HDFS replicates blocks to avoid missing of particular block. Data node sends feedback as heartbeat message to the namenode.

3.2. Map Reduce

Map reduce is a computing paradigm,[13] and a popular model for distributed data analysis, and it provides simplified data processing on large clusters. Its a popular model of its easiness to use, a programmer who does not have any experience in distributed computing systems can make use of this model. It provides functionality like fault tolerance [13], load balancing, hiding details of parallelization [2] etc. Programmers use map reduce to compute different programs relating to different data types. Problems written in Map Reduce are automatically parallized. Map Reduce provides scalable solution to process large amount of data. Map Reduce involves two parts a mapper part and a reducer part, mapper part will split up the data, and gives it as a particular key/value pair, this intermediate key/value pair is the input given to reducer, reducer combines this which forms the output. Result of a Map Reduce jobs are stored in a distributed file system.

3.3. Issues in Map Reduce

Hadoop/MapReduce is designed for batch processing of huge volumes of data but is not optimized for iterative computations, interactive mining and stream processing due to redundant data [24]. The main problem is that both iterative and interactive applications need data sharing across multiple MapReduce steps. In MapReduce the data sharing between parallel operations is done by writing the intermediate results to distributed file system HDFS which adds overhead due to disk I/O, redundant and wasteful processing and data replication [25]. It also suffers from various Configuration and Automation issues which require numerous configuration parameters to set when deploying a Hadoop MapReduce cluster. These Programming Model issues makes it unsuitable for machine learning algorithms and graph processing which often require iterations or incremental computations [25].

4. The Need of RDD

(i) Issue with Small Files
Hadoop isn’t suited for little information. (HDFS) Hadoop distributed file system does not have the capacity to productively bolster the irregular perusing of little documents as a result of its high limit plan.

(ii) Slow Processing Speed
MapReduce requires a great deal of time to play out these assignments in this manner expanding dormancy. Information is conveyed and handled over the bunch in MapReduce which builds the time and lessens preparing speed.

(iii) Support for Batch Processing only
Hadoop supports batch processing only, it does not process streamed data, and hence overall performance is slower. MapReduce framework of Hadoop does not leverage the memory of the Hadoop cluster to the maximum.

(iv) No Real-time Data Processing
Apache Hadoop is intended for group handling, that implies it take a gigantic measure of data as input, process it and create the outcome. Despite the fact that bunch preparing is extremely proficient for handling a high volume of information, yet relying upon the measure of the information being prepared and computational energy of the framework, a yield can be postponed altogether. Hadoop isn’t reasonable for Real-time information preparing.

(v) No Caching
Hadoop is not efficient for caching. In Hadoop, MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop.

(vi) Lengthy Line of Code
Hadoop has 1,20,000 line of code, the quantity of lines creates the quantity of bugs and it will require greater investment to execute the program.

5. Spark RDD and basic operations in RDD
Resilient Distributed dataset (RDD) [3] is a memory abstraction, in memory computations can be performed using RDD, iterative algorithm and interactive data mining tools cannot handle the computing frameworks efficiently, RDD came motivated from it. Performance can be improved by keeping the data in memory, RDD provides fault tolerance. RDD provides coarse grained transformations rather the fine grained one. RDD is implemented in spark which is a commonly used computing platform. Map reduce has been used widely for data analytics where user can write programs in a distributed way in a fault tolerant manner. RDD allows data reuse. It allow results to remain the intermediate memory. Defining a programming interface is a challenge in RDD. RDDs apply same operation to many data item as it is based on coarse grained transformation. In RDD lost data can be acquired quickly, as it stores data in its lineage. RDD can be created by using data in a stable storage and using other RDD. Spark expose RDD via language interpreted API. Spark computes RDD lazily. Persist method [3] is use to make use of RDD for future operations. Spark keeps RDD in memory and it also distribute to disks in case of shorter memory.

In case of distributed computing datasets is split into smaller nodes and divided among to achieve speedup and to improve efficiency. Bodgan Nicolae [12] proposed a novel coded data delivery scheme in case of no excess storage, this new coded scheme exploits new coding opportunity called leftover combining to reduce communication overhead. Resilient distributed datasets is an important concept in spark. Spark RDD is fault tolerant and it can perform computations in a parallel manner. Consider RDD as a machine language, dataset and dataframe comes with RDD optimization, based on the distribution of the system. RDD will compute in case of two, four and eight distributed systems. An optimized RDD is developed from dataset and dataframe as a result of computer based optimization. Dataset and dataframe is based on dynamic language principle and can take python, scala, java and R.

Bigdata hadoop system is a distributed file based system. Data shuffling is a high I/O operation, it will have a input, map, reduce and an output, in between there is a shuffle phase. It is a store and forward pattern. While doing store and forward pattern there can be a lot of input output operation, to avoid this shuffling data has to go into a primary storage. In bigdata high input output happen as it has to go into a muliple file storage system and it causes a high store and forward mechanism. To do a I/O based lookup need to remove going to a primary storage. In a task to add six numbers from 1 to 6 , consider all these numbers are in some files
read one value save it read another value save separately as a two process, adding happens in between a shuffle phase, and in some other file storage. Every action or operation taken need a input output bound operation, all these operations takes place in the memory, it will be saved in systems primary memory. Do all these operations without looking secondary memory.

The task performed is a primary memory lookup operation, its a DAG operation. A DAG structure will be computed for every operation. In spark this operation is done using RDD, it will have different computers for doing different operations and all these will be attached with primary memory, next figure 3 shows RDD distribution. RDD is immutable, RDD contains data in its lineage, hierarchy of data, will be saved in the environment. In RDD what ever data is present will remain and cannot edit it each and every time. There is no chance for the data to get changed. If x=abc, there there is no chance for x to become x= xyz. Each and everytime no need to go and look whether it is changed. It will be stored in memory, therefore data shuffling is not required. Spark 6.0 is used for RDD optimization.

| Table 1. Hadoop vs Spark |
|--------------------------|
| **Hadoop**               | **Spark**               |
| Data Storage             | The data is stored in the disc. | The data is stored in-memory. |
| Computing                | Computing is based on the disc. | Computing relies on RAM. |
| Fault Tolerance          | Fault tolerance is done through replication. | Fault tolerance is done through RDD. |
| Design Aspect            | Batch processing only. | Interactive query processing |
| Economic Aspect          | Less costly in comparison to spark. | More costly. |

5.1. Optimizing RDD

From the distributed store try to query a distributed data set using RDD and spark SQL. No matter which language or which API is used, or whether it is written in java, scala, sql, or dataset and dataframe, first thing happens is constructing a logical plan that will tell the structure of computation. Instructions can be like take the data read it from a particular file, do a join, do a filter, do dataframe operations on static data etc. It is easy to understand as a batch.

In the physical planning spark automatically runs queries in streaming fashion, it will be done continuously. Logical plan we will understand the structure of computation and in physical plan we will understand how it is going to perform after optimization. We use the catalytic optimizer and take the accurate plan and turn it into an incremental query plan. It is incremented by spark. There are four stages of plan,

(i)Parsed Logical Plan:It is the logical plan, applied by the spark optimize for query optimization.
(ii) Analyzed logical plan: It is the logical plan combined with catalog information, which is combined with name of datasets and types.

(iii) Optimized Logical Plan: It is applied to number of rows to simplify this query, we will do filter operation and then combine.

(iv) Physical Plan: Regression model is used for analyzing cost, cost model is similar to regression cost model.

In RDD there are lot of optimization plan, it can be different DAG operations and cost is applied for all. These DAG sets are created based on the datasets. Data will be applied and find which one will have least cost and that DAG will be selected and RDD will be created. Here RDD creation will be an optimization. Data is in table and it has attributes. If these attributes are unresolved there will be some rules to do that called catalytic rules. Developers will optimize this RDD based on these attributes, and then optimization will takes place using Catalytic Optimizer. Dataframe is data organized in tables or named columns. It is designed in such a way that to make data processing easier. Dataframes will allow developers to make changes in the data. Dataframe uses the catalyistic tree transformation framework. It will analyze the logical plan, and optimize it. For analysis it uses catalytic rules, and using that rules it resolve attributes that are not resolved. It will generate many physical plans, physical operators help for that, and it select a plan among them using the cost model.

6. Conclusion

Big data technologies takes a secondary file fetch for data shuffling, and faces many challenges in different cases like handling small files, batch processing, real time data processing and caching. To solve this difficulties spark takes an in-memory RDD based shuffling. All these comes under distributed computing events. Input output optimization leads to increase in data fetch and computation. In big data it takes a long time for data fetching, and results in time loss. So there is a need to minimize the input output computation. For that, every computation should be performed in primary memory. Encouraged by these results, we plan to broaden the scope of our work in future efforts. We analyzed the basic mechanism of hadoop and spark, and the various bottlenecks in the input-output mapping. For that, we considered the Resilient Distributed Dataset approach in both platforms. Based on the analysis, Spark found superior over the Hadoop. But there are various technologies like Apache Flink that are overtaking Spark. Like stream processing is much better using Flink then Spark as it is real time processing- 4G of Big Data. As a future analysis we can learn feature wise differences between Apache Spark vs Apache Flink to understand which is better and how.

7. References

[1] Tony Hey, Stewart Tansley, Kristin Tolle 2009 The Fourth paradigm data intensive scientific discovery, Microsoft Reserch version 1.1
[2] Jeffrey Dean and Sanjay Ghemawat 2004 Map Reduce: Simplified data processing on large clusters Communications of the ACM, vol. 51, no. 1, pp. 107113.
[3] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das 2012 Resilient Distributed Datasets: A fault tolerant abstraction for in-memory cluster computing. in NSDI12: The 9th USENIX Symposium on Networked Systems Design and Implementation, San Jose, USA, 2012, pp. 15 28.
[4] Bodgan Nicolae 2013 Understanding vertical scalability of i/o virtualization of mapreduce workloads: challenges and opportunities. in BigDataCloud 13: 2nd Workshop on Big Data Management in Clouds (held in conjunction with EuroPar13), Aachen
[5] Zujie ken, weisong shi 2012 Workload characterization on a production hadoop clusters: A case study on Taobao,978-1-4673-4583-3/12/IEEE
[6] Kay ouster hout 2015 Making sense of performance in data analytics frameworks in NSDI15: The 12th USENIX Conference on Networked Systems Design and Implementation, Oaklend, USA, 2015, pp. 293307.
[7] Francisco Clemente-Castello, Bogdan Nicolae 2015 Enabling bigdata analytics in the hybrid cloud using iterative mapreduce. IBM Research
8

[8] Aaron Davidson, 2013 Optimizing shuffle performance in spark. University of California, Berkeley - Department of Electrical Engineering and Computer Sciences, Tech. Rep.

[9] Bodgan Nicolae 2015 leveraging naturally distributed data redundancy to reduce collective i/o replication overhead in IPDPS 15: 29th IEEE International Parallel and Distributed Processing Symposium, Hyderabad, India, 2015, pp. 1023-1032.

[10] Yong chen, Xian-He Sun, 2015 LACIO: A new collective i/o strategy for parallel i/o systems.

[11] Xiao Yi Lu, Dipti Shankar, 2016 High performance design of Apache Spark with RDMA and its benefits on various workloads.

[12] Bodgan Nicolae, 2015 Towards memory optimized data shuffling patterns for big data analytics in CCGrid16: 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Cartagena, Colombia, 2016, pp. 409-412, short Paper.

[13] Yong chen, Xian-He Sun, 2015 On the worst-case communication overhead for distributed data shuffling.

[14] Bodgan Nicolae 2016 Leveraging adaptive i/o to optimize collective data shuffling patterns for big data analytics.

[15] V Srinivas Jonnalagadda, P Srikanth, 2015, A Review Study of Apache Spark in Big Data Processing.

[16] Aditya B. Patel, Manashti Birla, Addressing big data problem using hadoop and map reduce.

[17] HDFS Architecture Guide, available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf

[18] Saeed Shahrivari, 2014 Beyond Batch Processing: Towards Real-Time and Streaming Big Data.

[19] Professional Hadoop Solutions, https://www.wiley.com/en-us/professional_hadoop_solutions-p-9781118611937

[20] Vasiliki Kalavri, Vladimir Vlassov, 2013 MapReduce: Limitations, Optimizations and Open Issues 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communication, 2013, pp. 1031-1038, doi: 10.1109/TrustCom.126.

[21] Mansi Shah1, Vatika Tayal2, 2015 Fast Analysis of Sensor Data over MapReduce using Spark. International Journal of Advance Engineering and Research Development Volume 2, Issue 5.