Performance Improvement in Apache Spark through Shuffling

Nirali Rana, Shyam Deshmukh

***Abstract*— Apache Spark is a fast and general engine for large-scale data processing. Shuffle Phase refers to the partitioning and aggregation of data during an all-to all operations. Spark shuffle performance is improved in Sort-based Shuffle. Spark Shuffle Phase Large number of partitions are created by RDDs and Large number of Shuffle File is created. Discrete Fourier Transformation (DFT) Technique is use for Shuffle File handle and also Store Shuffle File in Shuffle Memory.**

***Index Terms*— Apache Spark; Shuffle; Shuffle Memory.**

I. INTRODUCTION

Apache Spark is an open-source analytics cluster computing framework developed in AMP Lab at UC Berkeley [11].Spark is an implementation of Resilient Distributed Datasets [RDD]. It provides high level APIs in Java, Scala, Python and R. Spark enables applications in Hadoop clusters to run up to 100x faster in memory and 10x faster running on disk. It comes with a built-in set of over 80 high-level operators.

Spark Driver controls the workflow and Spark workers launches executors responsible for executing part of the job submitted to spark driver through cluster node. Spark driver has few components: 1) RDD 2) Scheduler 3) Serializer 4) Shuffle. Spark Worker has two components: 1) Task and 2) Block Manager. RDD represents a partitioned collection of elements that can be operated on in parallel [8]. Spark's scheduler uses representation of RDDs. Spark sterilizer that uses Java's built-in serializer. Shuffle creates a large number of shuffles (M*R).

Shuffle refers to maintaining a shuffle file For each partition which is the same as the number of reduce task R per core C rather than per Map task M.

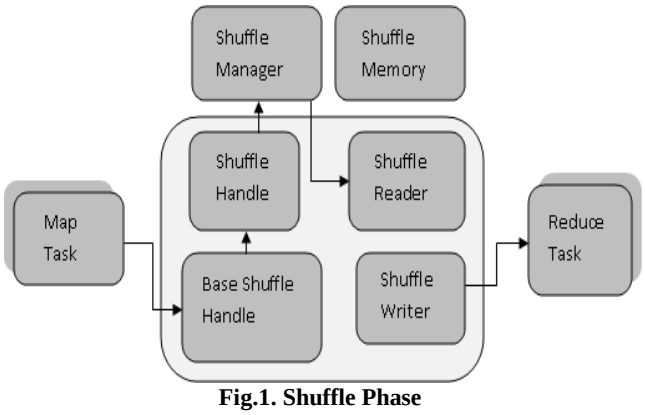

Fig.1. Shuffle Phase

Apache Spark has two types of Shuffle Techniques: Sort-Based and Hash-Based. Hash-based Shuffle is default in Shuffle data, is start in Spark 1.1.0. A Sort Based Shuffle Technique can be more scalable because Sort Based doesn't require writing a separate file for each reduce task from each mapper. Apache Spark Shuffle has few components which is show in figure 1.

Emulation of Hadoop behaviour may be too expensive in Spark mainly because it requires a second full pass over all shuffle data. It would also be more involved in terms of modifications required to Spark. An attempt at this was made by assuming primitive data types and Java objects are used during the shuffle phase.

II. PROPOSED SYSTEM

A. Proposed Methodology

Shuffle Phase is a component of Spark Driver. A shuffle is a communication between one input RDD and an Output RDD. Each shuffle has a fixed number of mappers and a fixed number of reduce partitions. Shuffle writer and Shuffle reader handle the I/O for a particular task, operating on iteration of RDD elements. When a shuffle has an Aggregator, the Shuffle Manager and its readers and writers are responsible for external spilling.

Shuffle File Consolidation by maintaining a shuffle file per core rather than per map task implying all map tasks sharing a core will use the same shuffle file. This gives rise to C*R no. of shuffle files, where C is no. of cores and R is no. of Reducers. The Shuffle file is generated M X R output files, where M is no. of Mappers.

The shuffle memory is a generalization of transposition memory that has been widely used in 2-D Discrete Fourier Transform. The shuffle memory is the first memory system that receives M x N inputs in row major while providing M x R outputs in column major order using only M x R memory space.

Shuffle memory can provide memory efficient for separable 2-D transforms.

$$F(u,v) = \frac{1}{MR} \sum_{x=0}^{M-1} \sum_{y=0}^{R-1} f(x,y) e^{-j^2 \pi (ux/M + vy/R)}$$

A Shuffle Memory of size M x R has the following characteristics:

1) Shuffle Memory receivers an M x R array in row major order while providing an M x R array in column major order.

ISSN: 2278 – 7798

2) A Shuffle memory supports read then write at the same address operation.

Apache Spark has components but in shuffle Memory Manager is manage the size of Shuffle Data. It allocates a pool of memory to task for use of Shuffle Operation. Sort based Shuffle is implemented using some technique but Hash base is default so we can't change in it. Shuffle Memory Manager Use task thread to disk-spilling and spills data-out when task ends all memory will be release by the executor.

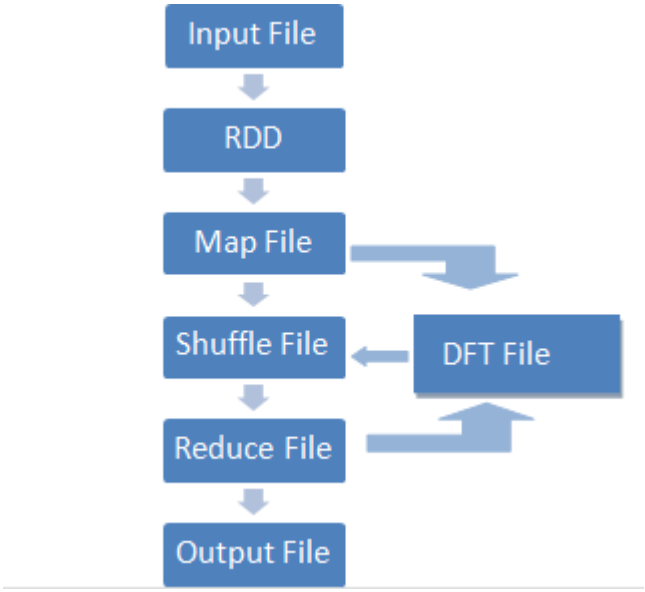

Fig.2. Proposed Method

Take any input file like wordcount.scala in spark example. RDD represents the partitioned collection of elements that can be operated on in parallel. Spark automatically sets the number of map tasks to run on each file according to its size and for distributed reduce operations. It uses RDD's number of partitions. In Between RDD operation and Output file 2-D Discrete Fourier Transformation Technique is applied in shuffle Memory to create large number of Shuffle File using Map and Reduce File. Shuffle File is Store in the Shuffle Memory.

III. RELATED WORK

Apache Spark Sort-based Shuffle Performance is implemented in Spark 1.1.0.So, here we are using Spark 1.1.0 Version and also using Scala Programming Language 10.2.3. Comparison between Haoop and Spark Shuffle Phase is show in Table 1.

	Hadoop Shuffle	**Spark Shuffle**
#Data Size	10GB	10GB
#Time	5min	4min
#Maps	10	15
#Reduces	50	60

Table 1 Comparison between Hadoop and Spark Shuffle

In our Experiment we are using 10GB Data size in Spark and Hadoop. Hadoop takes 5min to execute the data and Spark also take 4min time to execute data. Spark is faster than Hadoop. But we need to more faster than Hadoop. Because more data can take more time to execute. In Big Data every day 1TB data generated. So we need to faster engine like as Spark.

We have implemented Spark Shuffle using Discrete Fourier Technique (DFT). Our experiment is use 2 node and 10GB Data size. It creates large number of partitions using RDD. These partitions are used by Map and Reduce file. Using this DFT Technique we compare the Spark Shuffle and Experimented Spark Shuffle Performance in Table 2.This comparison is proved that using DFT Shuffle Memory Performance is improved in base of Time.

	Spark Shuffle	Experimented Spark Shuffle
#Data Size	10GB	10GB
#Time	4min	2.5min
#Maps	15	20
#Reduces	60	80

Table 2 Comparison between Spark Shuffle and Experimented Spark Shuffle

IV. RESULT

Apache Spark Shuffle is faster than Hadoop. Experimental Result is shown by two performance parameters: CPU Utilization and Shuffle Memory Performance which is shown Figure 3 and Figure 4.

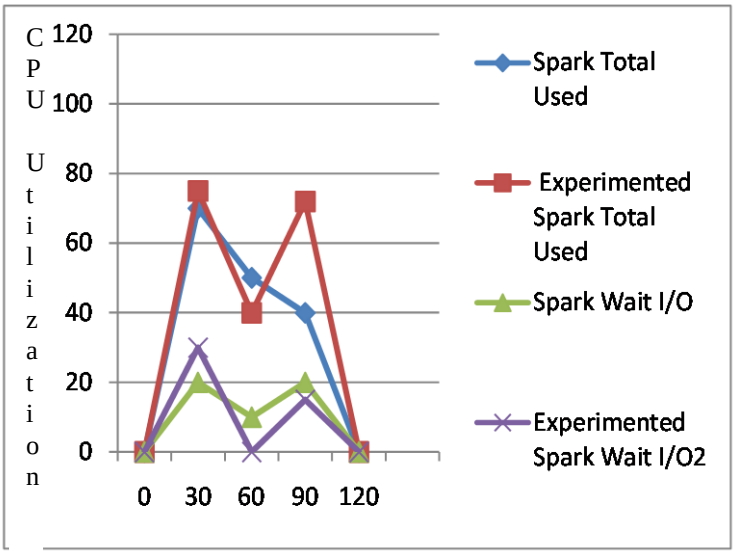

Fig.3. CPU Utilization Performances

CPU Utilization parameter is shows how many total memory use in Spark and Spark Write operation in Memory.

ISSN: 2278 – 7798

Using DFT technique Shuffle File is created in Shuffle Memory and compares the existing spark and proposed spark reading.

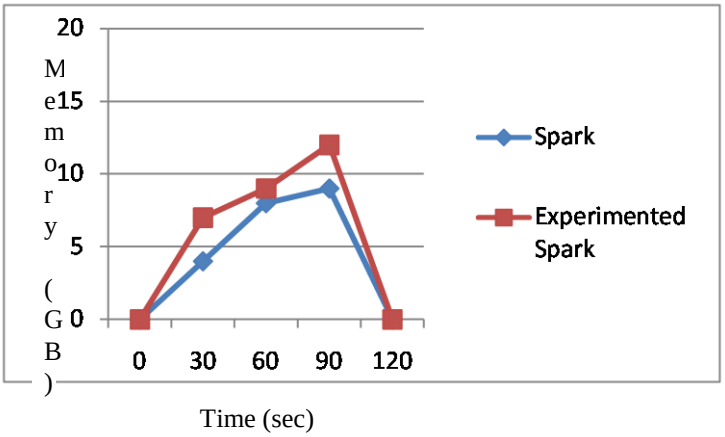

Figure 4 Shuffle Memory Performances

VI. CONCLUSION

Apache Hadoop and Apache Spark both are solving in Big Data problem. But Apache Spark is faster than Apache Hadoop in Shuffle Phase is proved using DFT Technique. Compare Spark Performance using Different Performance parameters like CPU Utilization, Shuffle Memory.

In Future, we are implemented DFT Technique in above 10GB Size of Data.

REFERENCES

[1] Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, Geoffrey C.Fox. A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures.Indiana University, USA, International Congress on Big Data 2014 IEEE.

[2] Aaron Davidson, Andrew Or "Optimizing Shuffle Performance in Spark" UC Berkeley 2013.

[3] Jingui Li, Xuelian Lin, Xiaolong Cui , Yue Ye. Improving the Shuffle of Hadoop Map Reduce. IEEE 5th International Conference on Cloud Computing Technology and Science.

[4] Yanfei Guo, Jia Rao and Xiaobo Zhou, "iShuffle: Improving Hadoop Performance with Shuffle-on-Write", University of Colorado, Colorado Springs, USA.

[5] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, "Spark: Cluster Computing withWorking Sets", University of California, Berkeley 2013.

[6] Kichul Kim, "Shuffle Memory System", University of Seoul, Korea 1999.

[7] Lei Gu, HuanLi,"Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark", Beihang University, Beijing, China 2013 IEEE.

[8] Lei Gu, Huan Li , "Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark", Beihang University, Beijing, China 2013 IEEE.

[9] Shuffle operation in Hadoop and Spark. http://analyticsindiamag.com/shuffle-operation-hadoop-spark/

[10] Apache Spark.http://spark.apache.org/

[11] Apache hadoop. http://apache.hadoop.org

[12] Scala Learn. http://www.scala-lang.org/documentation/

Nirali Rana I completed my Master of Engineering in Wireless & Mobile Computing (Computer Engineering) at Gujarat Technological University Ahmedabad.

Shyam Deshmukh He is an Assistant Professor at Pune Institute of Computer Engineering at Pune.

ISSN: 2278 – 7798