ISSN 2395-1621

Big Data Processing Using Hadoop

#1Krutika Bhalekar, #2Shital Birajdar, #3Pritam More, #4Arati Pawar, #5Prof. R.A.Kulkarni

1krutikabhalekar9@gmail.com
2birajdar.shital15@gmail.com
3pritamsmore555@gmail.com
4arativpawar94@gmail.com

ABSTRACT

Scheduling with Size-based recognized as an effective approach to guarantee fairness, priorities, parsing, job queuing and near-optimal system response times. We present a scheduler introducing this technique to a real, multi-server, complex and widely used system such as Hadoop. Scheduling requires a priori job size information, validation, processing, and retrievals. Scheduling builds such knowledge by estimating it on-line during job execution. Our scheduler, which is based on realistic workloads generated via a standard benchmarking suite, pinpoint at a significant decrease in system response times with respect to the widely used Hadoop scheduler, and show that our Scheduler is largely tolerant to job size estimation errors with hadoop as database.

Keywords: Big Data Problem, Hadoop cluster, Hadoop Distributed File System(HDFS), Parallel Processing, Map Reduce, Hadoop Components.

ARTICLE INFO

Article History

Received 30th March 2016

Received in revised form :

1st March 2016

Accepted : 2nd April 2016

Published online :

4th April 2016

I. INTRODUCTION

The large-scale data analytics, fostered by parallel processing frameworks such as Map Reduce has created the need to manage the resources of compute clusters that operate in a shared, multi-tenant environment. Within the same company, many users share the same cluster because this avoids redundancy and may represent enormous cost savings.

Initially designed for few and very large batch processing jobs, data-intensive scalable computing frameworks such as Map Reduce are nowadays used by many companies for production, recurrent and even experimental data analysis jobs. This heterogeneity is substantiated by recent studies that analyze a variety of production-level workloads. An important fact that emerges from previous works is that there exists a stringent need for short system response times. Data exploration, preliminary analysis and algorithm tuning on small datasets often involve interactivity, in the sense that there is a human in the loop seeking answers with a trial-and-error process. In addition, workflow schedulers It has possible to minimize starvation by efficient utilization of maximum resources in the hadoop.

II. LITERATURE REVIEW

A. J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in Proc. of USENIX OSDI, 2004

Essentially, size-based scheduling adopts the idea of giving priority to small jobs: as such, they will not be slowed down by large ones. The Shortest Remaining Processing Time (SRPT) policy, which prioritizes jobs that need the least amount of work to complete, is the one that minimizes the mean sojourn time (or response time), that is the time that passes between a job submission and its completion Policies like SRPT may however incur in starvation: if smaller jobs are continuously submitted, larger ones may never get scheduled. In order to avoid starvation, a common solution is to perform job aging: virtually decreasing the size of jobs waiting in the queue, in order to make sure that they will be eventually scheduled.

B. K. Ren et al., “Hadoop’s adolescence: An analysis of Hadoop usage in scientific workloads,” in Proc. of VLDB, 2013.

Compares Processing State (PS) with the SRPT scheduling discipline with an illustrative example: in this case, two small jobs – j2 and j3 – are submitted while a large job j1 is running. While in PS the three jobs run (slowly) in parallel, in a size-based discipline j1 is preempted: the result is that j2 and j3 complete earlier. It is worth noting that, in this case, the completion time of j1 does not suffer from preemption: somewhat counter to intuition, this is often the case for SRPT-based scheduling.

job size distribution is very skewed, ranging from few seconds to several hours. These sizes are difficult to obtain a priori, even though various recent works tackle the task of estimating Map Reduce job sizes ; in addition, evaluate the impact of estimation errors on size based scheduling for synthetic traces. for example, by overestimating the size of a job – is, in the long run, “corrected” by decreasing job size through aging. We verify our claim by implementing a common aging policy, where the remaining processing time is decreased by the amount of work performed in a virtual PS scheduler. This technique, which we label Shortest Remaining Virtual Time (SRVT), results (in the absence of estimation errors) in scheduling jobs in series, following the order in which they would complete with the virtual PS.

C. Y. Chen, S. Alspaugh, and R. Katz, “Interactive query processing in big data systems: A cross-industry study of MapReduce workloads,” in Proc. of VLDB, 2012.

Hadoop is a popular open source implementation of the MapReduce programming model for cloud computing. However, it faces a number of issues to achieve the best performance from the underlying systems. These include a serialization barrier that delays the reduce phase, repetitive merges, and disk accesses, and the lack of portability to different interconnects. To keep up with the increasing volume of data sets, Hadoop also requires efficient I/O capability from the underlying computer systems to process and analyze data. We describe Hadoop-A, an acceleration framework that optimizes Hadoop with plug-in components for fast data movement, overcoming the existing limitations. A novel network-levitated merge algorithm is introduced to merge data without repetition and disk access. In addition, a full pipeline is designed to overlap the shuffle, merge, and reduce phases. Our experimental results show that Hadoop-A significantly speeds up data movement in MapReduce and doubles the throughput of Hadoop. In addition, Hadoop-A significantly reduces disk accesses caused by intermediate data.

III.RELATED WORK

Big data is an unrolling concept which defines the typical large amount of unstructured, semi-structured and structured data. Big data doesn’t specify the derived data when speaking about the Zeta bytes, petabytes and Exabyte’ssize of data.

Nowadays Big data is expanding around us every single minute.Its increasing factor is the use of internet processand social media generates it. Big data comes from multi sources at an alarming volume, variety and velocity [1].

Big data plays important role in academics as well as industrial sources where the data generation ratio is at the peak. Relational database cannot be use always because it has certain limits so data scientist developed the concept of big data which can be used in efficient manner to accept challenges of bulk data storage.

Mike Guiltier, Forrester Analyst, proposes a definition that attempts to be pragmatic and actionable for IT professionals: “Big Data is the frontier of a firm’s ability to store, process, and access (SPA) all the data it needs to operate effectively, make decisions, reduce risks, and serve customers” (Guiltier, December 2012).3V’S of Big Data.

1.Velocity:-Analysis of the Batch process, Real Time process, datastreaming, when one takes chunks of data, it submits the job then there may be certain delay from the server.

[1][2]

2. Volume:-The size of data files is increasing with high ratio. The data size .The storage area is also increased from megabytes to Giga bytes, Zeta Bytes, petabytes and exabytes.Eg.Facebook generates around 500Tb of Data every single day.

3. Variety:-In the era of internet the variety of data is changing day by day. It follows with structured data like tables and unstructured like videos, xml, audio etc.

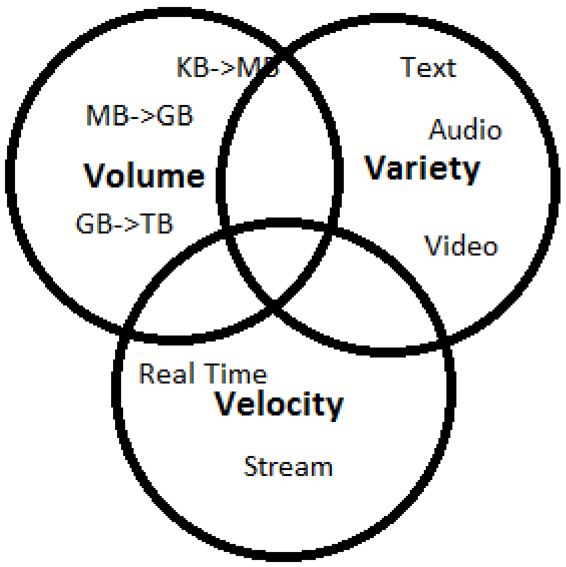

Fig. 1.Big Data Issues

Differentiate Between Big Data and DBMS.

1. Parallel database are actively querying the large data sets.

2. DBMS supports only structured data where as Big Data supports structured, semi-structured and unstructured data.

3. Parallel database are relational paradigm of rows and columns.

Pillars of Big Data

1. Text: The nature of text with all types of structures, unstructured and semi-structured.

2. Tables: Table form-rows and columns.

3. Graphs: Degree of separation, subjectpredicates, objectprediction, semantic discovery. [6]

Hadoop

Hadoop is a framework which is open-source for processing and storing big data in a distributed way on large clusters. It processes two tasks: large storage of data and processes faster with data.

Framework: - It is to develop and to run software applications provides– programs, tool sets, connections, etc.**[7]**

Open-source software:-It is downloaded and can be used to develop the commercial software. Its broad [7] and open network can be easily, managed by the developers.

Distributed Form: - Data is stored on multiple computers and divided in to equal chunks on the parallel connected nodes.

Large storage. The Hadoop framework storeslarge amount of data by dividing into separate blocks and stored on clusters.

Hadoop includes:-

1. Hadoop Distributed File System (HDFS):-It manages the large amount of data that is stored in distributed fashion.

2. Map Reduce:-a framework and a programming model for distributed processing [6] [7] on parallel clusters.

HADOOP STRUCTURE

1. Hadoop Distributed File System(HDFS) Structure.

The HDFS is a distributed file system developed to execute on commodity hardware.The feature of HDFS is fault-tolerance and to be deployed on low-cost hardware. HDFS provides great access to data and is suitable for applications that have large data sets. [5].

The distribution of files is divided into chunks of 64MB size.

The HFDS architecture contains a unit Name Node, multiple Data Nodes;it can also be stated as master slave like architecture. Name Node manages the mapping of file system namespace and regulates access to the block of files. Data Nodes are responsible for the process of read and write from the clients. It also performs operations like creation of blocks and deletion of blocks. [hadoop.apache]

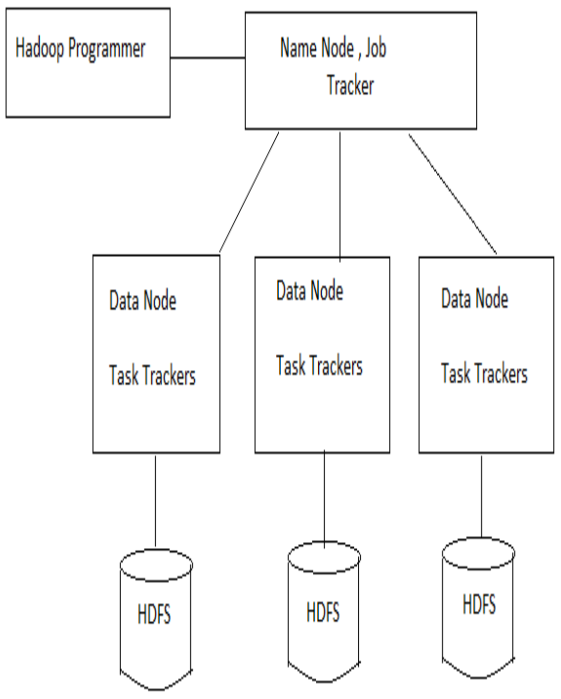

Fig. 2.Hadoop Architecture.

Map reduce

• Map Reduce is a programming model and software framework first developed by Google (Google's Map Reduce paper submitted in 2004). Intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner, Petabytes of data, Thousands of nodes. Computational processing occurs on both: ▫ Unstructured data:file system ▫ Structured data:database. Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines [4]

Hadoop Scheduling:

Hadoop implements the schedulers where resources are assigning to the jobs. Traditional scheduling gives us the scenario that all algorithms are not same and might not be as effective and dependent completely.We have to study various schedulers available in Hadoopand how relevant they are as compared to traditional schedulers in terms of performance, throughputs, time consuming etc. First in First out (FIFO) scheduler is a default scheduler which considers e order for submission of the jobs to get executed.

I. FIFOScheduler.

FIFO is traditional default scheduler which performs with the use ofqueue. Job is divided into several tasks and then loads to free slots of the queue on the task tracker. In filo the job have to wait for execution due to acquisition of clusters take place this leads to wait for other jobs for their turn. Shared clusters have ability for offering resources to users.

II. Fair Scheduler.

Fair scheduler groups jobs into "pools" and it allots each pool a guaranteed minimum share withsegment more capacity equally between pools. Facebook first develop the concept of fair scheduler to manage the access to their Hadoop and get the subsequent relation with the Hadoop environment. Poolshave minimum mapping slots andminimum reduced slots. The Fair Scheduler supports

Preemption, so if a pool has not received its fair share for acertain period of time, then the scheduler will kill tasks inpools running over capacity in order to give the slots to the pool running under capacity [8].

III. Capacity Scheduler.

Capacity scheduler organizes jobs into queues. It shares as percent of cluster. FIFO scheduling within each queue and it supports preemption. Yahoo developed the capacity scheduleraddresses a usage scenario where the number of users islarge, and there is a need to ensure a fair allocation of computation resources amongst users.

[3][6]When a Task Tracker slot becomes free, the queue with thelowest load is chosen, from which the oldest remaining jobis chosen. A task is then scheduled from that job.

IV. CONCLUSION

Our work was motivated by the realization that MapReduce has evolved to the point where shared clusters are used for a wide range of workloads, which include a non-negligible fraction of interactive data processing tasks. As a consequence, we have witnessed the raise of deployment best practices in which long sojourn times – due to a fair sharing of resources among competing jobs – were compensated by over dimensioned Hadoop clusters. In addition, we remarked that efficient cluster utilization could be approximated through a tedious manual exercise, involving the creation of static resource pools to accommodate workload diversity and an important tuning effort. To overcome such limitations, in this work we set off to study the benefits of a new scheduling discipline that targets at the same time short sojourn times and fairness among jobs.

We thus proposed a size-based approach to scheduling jobs in Hadoop, which we called HFSP. Our work brought up several challenges: evaluating job size on-line without wasting resources, avoiding job starvation both on small and large jobs, and guaranteeing short sojourn time despite estimation errors were the most noteworthy. We solved these problems in the context of a multi-server system using virtual time and aging, that is built to be tolerant to failures, scale-out upgrades, and supports the composite job structure of Map Reduce.

The system implementation is on the hadoop version 1.2.1 which is a basic version of hadoop.The system can be made Scalable by using the hadoop version 2.X where some of the drawbacks can be overcome in 2.X

REFRENCES

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in Proc. of USENIX OSDI, 2004.

[2] Y. Chen, S. Alspaugh, and R. Katz, “Interactive query processing in big data systems: A cross-industry study of MapReduce workloads,” in Proc. of VLDB, 2012.

[3] K. Ren et al., “Hadoop’s adolescence: An analysis of Hadoop usage in scientific workloads,” in Proc. of VLDB, 2013.

[4] Apache, “Oozie Workflow Scheduler,” http://oozie.apache.org/.

[5] “Hadoop: Open source implementation of MapReduce,” http://hadoop.apache.org/.

[6] E. Friedman and S. Henderson, “Fairness and efficiency in web server
protocols,” in Proc. of ACM SIGMETRICS, 2003.

[7] L. E. Schrage and L. W. Miller, “The queue m/g/1 with the shortestremaining processing time discipline,” Operations Research, vol. 14,no. 4, 1966.