Survey on Data Processing and Scheduling in Hadoop

Somya Singh
Department of Computer Science and Engineering
ASET, Amity University
Noida, India

Neetu Narayan
Department of Computer Science and Engineering
ASET, Amity University
Noida, India

Gaurav Raj
Department of Computer Science and Engineering
ASET, Amity University
Noida, India

ABSTRACT
There is an explosion in the volume of data in the world. The amount of data is increasing by leaps and bounds. The sources are individuals, social media, organizations, etc. The data may be structured, semi-structured or unstructured. Gaining knowledge from this data and using it for competitive advantage is the primary focus of all the organizations. In the last few years Big Data has found its way in almost every field, from government to private sectors, industry to academia. The major challenges associated with Big Data are data organization, modeling, data analysis and retrieval. Hadoop is a widely used software framework used for the large scale management and analysis of data. The main components of Hadoop: HDFS and MapReduce, enable the distributed storage and processing of data over a large number of commodity servers. This paper provides an overview of MapReduce and its capabilities and discusses the related issues.

General Terms
Big Data, Hadoop, HDFS, MapReduce

Keywords
MapReduce, Scheduling

1. INTRODUCTION
There has been an immense increase in the volumes of data in the past few years. The major sources of growth of data are the individuals and their increased use of media and the Internet. In the present era all the activities of the individuals on the Internet is recorded. This is done by most of the companies to study the customer's search patterns and understand their demands. All activities of the individuals like online transactions, searches, etc are recorded and analyzed. Business data is analyzed to understand the customer better, for promoting the brands and minimizing the risks and thereby enhancing the productivity. Other sources of data are the public and private sector organizations which store the information of their customers and the market in order to plan business strategies. This has brought the concept of Big Data into picture. Big Data has found its way in almost every field-industry, healthcare, banking, insurance, government, telecommunications consumer products and businesses [11].

With the increasing volume of data, a number of issues come up. Firstly, it is difficult to deal with huge amounts of data. Storing and handling of large amounts of data needs to be taken care of. The second issue that comes up is to decide what is to be stored and what is important to us. The third issue to extract the important data and analyze it in a way to put it to our best advantage.

In 2001, Doug Laney, the META Group (now Gartner) analyst defined big data in three dimensions, i.e.

- Volume: implies to the huge amount of data, which continues to increase with time.
- Velocity: refers to the speed at which data grows.
- Variety: refers to the wide range of data types (structured and unstructured) [3,11].

In 2012, Gartner updated its definition as follows: "Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization"[3,11].

2. HADOOP
Doug Cutting, the creator of Apache Lucene developed Hadoop and named it after his son's toy elephant. Apache Hadoop is an open source project that allows the processing of huge amount of data on a cluster of thousands of commodity servers. It can be scaled up to thousands of machines. These machines do not share memory and each offers local computation and storage. Hadoop replicates the data on multiple servers and takes control over the parallel processing of user's jobs on these servers. Hadoop is specially designed to work with huge amounts of data. It does not work well when the data is small.

Hadoop has two main components:

a) HDFS - the Hadoop Distributed File System is a file system that is capable of efficiently storing and managing petabytes of data. The files are stored on the commodity clusters. The data files are broken down into blocks each of which is replicated on more than one (usually three) servers. This ensures that the data is not lost even if a node containing data fails. It also checks for server failures and other hardware problems. This replication of data provides fault-tolerance. This way any block can be read from any one of the server. Thus it also helps in enhancing the performance. It continually monitors the servers in a cluster and also the data blocks and thus ensures data availability. In Hadoop, checksums are given to all the data splits. Whenever data is read from a block, its checksum is verified. This way it finds out if any block is damaged or not. In case any block is damaged, it is restored from its replicas. This way it provides scalable, inexpensive, reliable and fault-tolerant storage [1,2].

b) MapReduce - It is the framework that handles the parallel processing of data on the cluster. The user applications are divided into smaller tasks which are executed on different nodes in a cluster.

3. MAPREDUCE

MapReduce is a programming model that is used for the processing and analysis of large data sets. It is a parallel data processing system. It is capable of processing terabytes of data on thousands of servers efficiently. It is capable of handling multi-structured data files. The processing of large data files is distributed across the cluster of machines. It copies the data across multiple servers, runs various tasks in parallel and manages all communications and data transfers between various parts of the system. It provides abstraction to the messy details of parallelization and supports simple computation for deep data analysis. This helps inexperienced programmers to utilize the resources of a large distributed system easily. The MapReduce computation requires only two simple programs to be written by the user- the Map() function and the Reduce() function and the programmer doesn't have to bother about the details of parallelization of work or the data distribution and its management [1,2,9,10].

3.1 Map and Reduce Procedures

MapReduce performs processing in two phases: the Map phase and the Reduce phase. The input-output of both these phases are key value pairs[9]. The programmer writes two functions: the map function and the reduce function.

Map() procedure: It is a simple procedure written by user. It takes the input data set and gives as output a set of intermediate key/value pair. Its performs filtering of the input data set and sorts them. Its main function is to find all the intermediate values and associates them with an intermediate key. The output of the Map() procedure is passed to the Reduce() function as input.

Reduce() procedure: It is also written by the user. It takes as input the intermediate key I produced by the Map() function and the set of values for that key. These keys are merged together which results in a smaller set of values. The reduce workers can begin executing as soon as any map worker finishes its execution and passes its output. This means that the reduce workers don't have to wait for the completion of all map workers. This helps in achieving better throughput [2,9].

3.2 Implementation

MapReduce can be implemented in a number of ways. The right implementation depends upon the environment in which it is to be used. For example, in some cases small shared-memory machine may be needed while in other cases larger collection of networked machines may be required. The most common implementation uses hundreds of machines with at least 2-4 GB of memory connected via Ethernet. These machines are usually dual processors running Linux. The commodity network has 1 gigabit/second bandwidth at machine level. IDE disks are attached to each of the machines to provide storage. The data on these disks is managed by the distributed file system (HDFS). To ensure reliability and availability of data, the data is replicated at least three times [9].

3.3 Execution Overview

A job consists of a MapReduce program and input data set. MapReduce consists of a software component called the Job Client. The Job Client sends the job to the Job Tracker. The Job Tracker then chooses the servers that will run that job. The Job Tracker consults the NameNode to look up the location of all the blocks that make up a file required by a job. And then it instructs each of the servers to perform the user's analysis.

1. The input data is divided into a set of M smaller parts. Each sub part is of size 16-60 MB per split. These copies are then distributed to different servers for processing.

2. There is a Master Node which is nothing but one of the copies of the program. The rest of the nodes are the Worker nodes. The Master node consists of the Job Tracker, Task Tracker, NameNode and DataNode. The worker node consists of Data Node and Task Tracker. The master nodes distribute the data obtained from the client to the HDFS and assign the tasks to Map-Reduce layer. It looks up for the idle worker nodes and assigns the map task or reduce task to them.

3. The worker node that is given a map task reads the input data and decomposes key pairs. These entity pairs are fed to the Map function as input. They are also buffered in memory.

4. The system writes the buffered intermediate entity pairs to local disk in some time intervals. The locations of these buffered entity pairs on the local disk are passed to the master. The master then forwards the location information to the reduce workers.

5. The data written by the map workers is read by the reduce workers from the local disks. And then it performs the sorting of the intermediate entity pairs.

6. The reduce worker then sends the entity and its intermediate pair to the Reduce function written by user.

7. When the entire map and reduce tasks have been finished, the master node wakes up the user program and the output of the map-reduce execution is handed over to the user [9].

4. SCHEDULING ISSUES IN MAPREDUCE

Hadoop is a cluster of multiple machines. Managing the execution of tasks on the multiple machines is the primary aim of scheduling. There are three main factors that affect scheduling decision such as locality, synchronization and fairness constraints. These are discussed below:

1. Locality

Locality is an important factor which needs to be kept in mind before designing a scheduling algorithm. Locality is nothing but the distance between a node which has been assigned the task and the node which contains the input data required by the task. If the input data is closer to the computation node it will be quickly available thereby increasing the throughput of the system. In ideal case the data is available at the computation node itself. This is called node locality. When the data node and computation node are on the same rack, it is called rack locality. Locality greatly affects the cost of data transfer and the performance in a shared cluster environment because of the limited bandwidth. Some schedulers assign the task to a node which is not far away from the node containing the input data [8].

2. Synchronization

The Map processes transfer the intermediate key pairs to the reduce processes as input. This process is called synchronization. Synchronization also affects the performance

of the system. The intermediate outputs are sent to the reduce processes only after all the map processes have been completed. Thus if any of the map process is running slow, the other processes have to wait for its completion. This increases the overall processing time thereby, causing a degradation in the throughput. There may be a decrease in throughput because of some other factors as well. Like a node fails, the other nodes have to wait until the failed node finishes its task again. Moreover in a heterogeneous environment, every node has different execution time. The network bandwidth may also be different. So, all the nodes do not perform at the same rate. Thus some nodes may have to wait for the other nodes to complete execution [8].

3. Fairness

There are a number of map and reduce tasks performing at the same time. At times, there may be a situation where a map-reduce job with heavy workload takes over the processing units for a long time. The smaller jobs have to wait for a very long period of time. Thus the short computation jobs do get a fair share of cluster resources and their execution time is increased. Synchronization also affects fairness. For example, if a map process delays too long, the nodes remain idle as the reduce jobs keep waiting for it to finish and this leads to the starvation of other jobs [8].

5. SCHEDULING IN HADOOP

1. Default FIFO Scheduler

The default scheduler of Hadoop uses a FIFO queue. This scheduler works irrespective of the heterogeneity of Hadoop cluster components. A job is partitioned into smaller tasks. These tasks are loaded into a queue. When the free slots become available the tasks are assigned in FIFO order. Every job uses the entire cluster. Thus the other jobs in the queue have to wait for their turn. There is no concept of prioritizing the jobs [1,13].

2. Fair Scheduler

The Facebook group developed the Fair Scheduler to manage large amount of data with their Hadoop cluster. Fair scheduling aims to give a fair share of the resources to every user. In this the jobs are divided into pools and each pool is allocated a fair share of the resources i.e., the Map and Reduce slots. The jobs assigned in a pool can be scheduled either in a FIFO order or by using fair sharing. This method assigns resources to jobs in such a way that all the jobs at that time get approximately the same share of resources. If at a time there is only one job running then that job can use the entire cluster resources. When new jobs are submitted the idle slots are allocated to them in such a way that each job gets the same CPU time to execute [1,13].

The Fair Scheduler also gives the provision of pre-emption. If a pool does not receive the minimum number of resources for some time, the tasks of other pools that are over capacity are killed and the slots are given to those that are under capacity. The most recently launched jobs are selected for killing from the over allocated pools so that the wasted computations are less. Preemption ensures that the jobs are not starved and that the small jobs get to finish on time, unlike in FIFO scheduling.

Moreover, priorities can also be assigned to certain jobs in pools so that the jobs are scheduled according to their priorities. Priorities are used to govern the fraction of the total computation time that every job will get [1,13].

3. Capacity Scheduler

Capacity Scheduler was developed at 'Yahoo' [9]. It works best for large Hadoop systems which have a large number of resources which are to be fairly allocated among the users. In the Capacity Scheduler the user jobs are submitted into queues and each queue is allocated a fraction of the total resource capacity. All the jobs assigned in a queue can use the entire capacity of resources given to the queue. The free capacity of a queue is shared by the other queues. When any Map or Reduce slot becomes free, the queue which has the least load is selected and the oldest remaining job is picked for execution from this queue.

The capacity scheduler does not support preemption. The total resource capacity is distributed among the queues even if it is beyond their capacity. But if new queues request for the resources, the scheduler waits for the jobs running on the resources to finish and then allocates these resources to the new queues. This helps in the efficient and elastic use of the resources. Within a pool the jobs can also be prioritized i.e., the job with higher priority will be executed first. However, this feature is turned off by default.

The capacity scheduler also supports resource based scheduling where a task can explicitly specify the resources (like RAM and virtual memory) required by it. The task is then allocated to the node which has the required amount of resources [1,13].

4. Deadline Constraint Scheduler

The Deadline Constraint Scheduler focuses on the deadlines for scheduling the jobs. It has two main proposed components:

a) Job execution cost model- It deals with the distribution of data and handles different parameters like runtime of map and reduce tasks, input data sizes, etc.

b) Constraint based scheduler- This component takes the user deadlines as input and performs the scheduling.

The system works on the assumption that the reduce tasks begin only after the execution of all map tasks is finished and same amount of input data is given to each reduce node. The proposed job execution cost model handles the scheduling of the jobs. It schedules a job only if its deadlines can be met. The scheduling is independent of the number of jobs submitted to the cluster. When a new job is submitted to the cluster its schedulability test is performed to see if it is possible to finish the job within the specified deadline. The number of free slots at that time is checked. The minimum number of map and reduce tasks required in that job are also estimated and if it is less than the available free slots, then the job is enlisted for scheduling [13].

5. Resource Aware Scheduling

In Hadoop the Job Tracker is responsible for taking all the scheduling decisions. It splits the input data and assigns the work to the Task Trackers. The Task Trackers then execute the tasks. The Job tracker takes care of the running jobs and maintains a record of the tasks assigned to the Task Tracker

and the state of each Task Tracker. Each Task Tracker has a fixed number of computation slots allocated to it. The idea of the Resource Aware Scheduling is that the Task Trackers keep a track of the utilization of the resources (CPU utilization, channel, I/O, etc). There are two Resource Aware Scheduling mechanisms:

a) *Dynamic Free Slot Advertisement*- In this approach the available number of free computation slots is estimated dynamically. This helps in improving the throughput of the system.

b) *Free Slot Priorities/Filtering*- In this approach the maximum number of computation slots is configured. The free Task Tracker slots are buffered in memory for some time and then they are advertised according to the resource availability. This means that the Task Tracker nodes which have higher resource availability are advertised first for scheduling. This helps in achieving higher throughput because the tasks are not scheduled based on next available free slot but on the basis of higher resource availability [13].

6. CONCLUSION

In order to handle multiple nodes in a shared cluster environment an efficient scheduling algorithm is required. The scheduling techniques that exist today provide a tradeoff between the three issues of scheduling- locality, fairness and synchronization. There is no scheduling that is capable of resolving all three at the same time. If we try to resolve the synchronization issue then it is not possible to achieve fairness. The reduce operation begins after the completion of all map tasks. So the reduce tasks need to sit idle until all map tasks finish execution. This means that the resource utilization is poor.

In a scenario where every task is given equal number of resources, if the data is not present on the same server itself (i.e., no locality) then the task has to fetch the data from the other servers. This is an unnecessary delay caused which increases the processing time and decreases the throughput.

The FIFO Scheduler is simple to implement and is efficient for small systems. But if it is deployed on large systems it can degrade the throughput and the performance of the system. It is not applicable to the environment where the data is shared among multiple users. In the Fair Scheduler the throughput is good enough but the feature of data locality is hampered. The Deadline Constraint Scheduler takes parameters like deadlines as input. This improves the performance.

With the rise of Big Data, Hadoop has emerged as a new research field. The Hadoop architecture and its schedulers are evolving. Its new versions have been released with enhanced features and functionalities. Many new scheduler improvement ideas have been suggested by researchers and a lot of further research work is being done in this area. The Hadoop systems being deployed are user specific and hybrid scheduling approaches have come up that are best suited to the user's needs. The hybrid approaches combine the features of two or more scheduling techniques to offer their combined specialty. These approaches are developed with a motivation of achieving best resource utilization and average computation time. The development of a scheduler which is capable of taking into account all aspects of scheduling and heterogeneity levels is still awaited.

7. REFERENCES

[1] Heger, A.D. Hadoop Design, Architecture & MapReduce Performance. DHTechnologies.

[2] Olson, M. 2010 Hadoop: Scalable, Flexible Data Storage and Analysis. Cloudera**,** IQT Quarterly.

[3] Doug, L. 2001 3D Data Management: Controlling Data Volume, Velocity and Variety. Meta Group, File 949.

[4] White, C. 2012 MapReduce and the Data Scientist. BI Research.

[5] Einav,L. and Levin, J. 2013. The Data Revolution and Economic Analysis. In Proceedings of the NBER Innovation Policy and the Economy Conference, Stanford University and NBER.

[6] White, T. Hadoop: The Definitive Guide. 3rd Edition, O'Reilly.

[7] Zhiqiang ,M. L. G. The Limitation of MapReduce: A Probing Case and a Lightweight Solution.

[8] Yoo, D. and Sim K. M. 2011. A comparative review of Job Scheduling for MapReduce. In Proceedings of IEEE CCIS2011.

[9] Dean, J. and Ghemawat, S. 2010. MapReduce: Simplified Data Processing on Large Clusters. Google Inc.

[10] Dean, J. and Ghemawat, S. 2010. MapReduce: A Flexible Data Processing Tool. Communications of the ACM.

[11] Big Data, http://en.wikipedia.org/wiki/Big_data.

[12] Apache Hadoop, http://hadoop.apache.org/.

[13] Rao, B. T. and Reddy. L.S.S. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments. IJC A, 2011.

[14] Haoop, https://en.wikipedia.org/wiki/Apache_Hadoop.