Big Data Processing: Improve Scheduling Environment in Hadoop
Bhavik.B.Joshi
LDRP-ITR, Kadi Sarva Vidhyalay, Gujarat, India

Abstract—Now a days, dealing with datasets in the order of terabytes or even petabyte is a reality. Therefore, processing such big datasets in an efficient way is a clear need for many users. In this context, Hadoop Map Reduce is a big data processing framework that has rapidly become the de facto standard in both industry and academia. The main reasons of such popularity are the ease-of-use, scalability, and failover properties of Hadoop Map Reduce. However, these features come at a price: the performance of Hadoop Map Reduce is usually far from the performance of a well-tuned parallel database. Therefore, many research works (from industry and academia) have focused on improving the performance of Hadoop Map Reduce jobs in many aspects. For example, researchers have proposed different data layouts; join algorithms, high-level query languages, failover algorithms, query optimization technique, and indexing techniques. We will point out the similarities and differences between the techniques used in Hadoop with those used in parallel databases. And try to improving scheduling environment in Hadoop.

Key words: Big Data, Hadoop, Map Reduce, Scheduling algorithms, SARS algorithm.

I. INTRODUCTION
Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Big data is a broad term for data sets so large or complex. Those traditional data processing applications are inadequate. Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, search, sharing, storage, transfer, visualization, and information privacy. But having data bigger it requires different approaches, Techniques, tools and architecture [1] [2].

Big Data generates value from the storage and processing of very large quantities of digital information that cannot be analyzed with traditional computing techniques. [1] [10].

A. Architecture of Hadoop

There are two kinds of nodes in Hadoop first one are master node and second one is the slave nodes. Master node contains job trackers and slave nodes contain task trackers. [7] [8]

Map Reduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. Hadoop Map Reduce (Hadoop Map/Reduce) is a software framework for distributed processing of large data sets on compute clusters of commodity hardware. [1][9]. It also uses the HDFS (Hadoop distributed file system).

Hadoop Distributed File System (HDFS): a distributed file system that store large amount of data with high throughput access to data on clusters. A Map Reduce program is composed of a Map () procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name).Reduce () procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "Map Reduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshaling the distributed servers, running the various tasks in parallel. Managing all communications and data transfers between the various parts of the system and providing for redundancy and fault tolerance. [7][8]

B. Working of Map Reduce

The model is inspired by the map and reduces functions commonly used in functional programming although their purpose in the Map Reduce framework is not the same as in their original forms. The key contributions of the Map Reduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once.[1][2][4][6]

C. Hadoop Map Reduce Scheduling and Important Issues

Hadoop allows the user to configure the job submit it and control its execution and query the State.

Every job consists of independent tasks and all the tasks need to have a system slot to run. In Hadoop all scheduling and allocation decisions are made on a task and node slot Level for both the map and reduce phases.
Big Data Processing: Improve Scheduling Environment in Hadoop

The Hadoop scheduling model is a Master/Slave (Master/Worker) cluster structure. The master node (Job Tracker) coordinates the worker machines. Task Tracker. Job Tracker is a process which manages jobs, and Task Tracker is a process which manages tasks on the corresponding nodes. The scheduler resides in the Job Tracker and allocates Task Tracker resources to running tasks: Map and Reduce tasks are granted independent slots on each machine.

In the fact, Map Reduce Scheduling system takes on in six steps: first, User program divides the Map Reduce job. Second, master node distributes Map Tasks and Reduce Tasks to different workers. [1][3][4] [5]

Third, Map Tasks reads in the data splits, and runs map function on the data which is read in. Fourth, Map Tasks write intermediate results into local disk. Then fifth, Reduce Tasks read the intermediate results remotely, and run reduce function on the intermediate results which are read in. Finally, These Reduce Tasks write the final results into the output files. [1][3][4][5]
D. Issues in Scheduling in Hadoop

There are three important scheduling issues in Map Reduce such as locality, synchronization and fairness.

(D.1) Locality

Locality is a very crucial issue affecting performance in a shared cluster environment, due to limited network bisection bandwidth. It is defined as the distance between the input data node and task-assigned node. The concept of data locality in Map Reduce is a time when scheduler tries to assign Map tasks to slots available on machines in which the underlying storage layer holds the input intended to be processed.[1][3][6][7]

(D.2) Synchronization

Synchronization is the process of transferring the intermediate output of the map processes to the reduce processes as input is also considered as a factor which affects the performance. In generally, reduce process will be start a time when Map Process is absolutely finished. Due to this dependency, a single node can slow down the whole process, causing the other nodes to wait until it is finished. So, Synchronization of these two phases (Map and Reduce phase), made up essential help to get overall performance of Map Reduce Model.[6][7]

(D.3) Fairness

A map-reduce job with a heavy workload may dominate utilization of the shared clusters, so some short computation jobs may not have the desired response time. The demands of the workload can be elastic, so fair workload to each job sharing cluster should be considered. Fairness finiteness has trade-offs between the locality and dependency between the maps & reduce phases. To solve these important issues many scheduling algorithms have been proposed in the past decades.[1][5][6][7]

II. SCHEDULING ALGORITHMS

Due to the important issues that we described, and many more problems in scheduling of Map Reduce, the Scheduling is one of the most critical aspects of Map Reduce. There are many algorithms to address these issues with different techniques and approaches. Some of them get focus to improvement data locality and some of them implements to provide Synchronization processing. Also, many of them have been designed to minimizing the total completion time. In this section we describe briefly some of these algorithms.[6][7]

A. Analysis of Scheduling Algorithm Based Of Taxonomy, Advantages and Disadvantages

<table>
<thead>
<tr>
<th>Sr.no</th>
<th>Scheduling Algorithm</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FIFO</td>
<td>1. cost of entire cluster Scheduling process is less. 2. simple to implement and efficient.</td>
<td>Designed only for single type of job. Low performance when run multiple Types of jobs. poor response times for short jobs</td>
</tr>
<tr>
<td>2</td>
<td>Fair Scheduling</td>
<td>1. Less complex. 2. Works well when both Small and large clusters. 3. it can provide fast response times for small jobs mixed with large Jobs.</td>
<td>Does not consider the job weight of each node.</td>
</tr>
<tr>
<td>3</td>
<td>Capacity</td>
<td>1. Ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within Queues over large cluster.</td>
<td>1. The most complex Among three schedulers.</td>
</tr>
<tr>
<td>4</td>
<td>hybrid scheduler based on dynamic priority</td>
<td>1. is a fast and flexible Scheduler. 2. improves response time for multi-user Hadoop Environments.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LATE</td>
<td>1. robustness to node Heterogeneity. 2. address the problem of how to robustly perform speculative execution to Maximize performance.</td>
<td>1. only takes action on Appropriate slow tasks. 2. However it does not compute the remaining time for tasks correctly and cannot find real slow tasks in the end. 3. Poor performance due to the static manner in computing the Progress of the tasks.</td>
</tr>
<tr>
<td>6</td>
<td>SAMR</td>
<td>1. decreases the execution Time of map reduce job. 2. improve the overall Map Reduce performance in the heterogeneous Environments.</td>
<td>1. do not consider the data locality management for Launching backup tasks.</td>
</tr>
</tbody>
</table>
Table 1:

III. INTRODUCTION TO SARS SCHEDULING ALGORITHM

Map Reduce is by far one of the most successful realizations of large-scale data-intensive cloud computing platforms. Algorithms, the reduce completion time is decreased sharply. Time, the experimental results illustrate that, when comparing with other When to start the reduce tasks is one of the key problems to advance the Map Reduce performance. The existing implementations may result in a block of reduce tasks. When the output of map tasks becomes large the performance of a Map Reduce scheduling algorithm will be influenced seriously. Through analysis for the current Map Reduce scheduling mechanism, this paper illustrates the reasons of system slot resources waste, which results in the reduce tasks waiting around, and proposes an optimal reduce scheduling policy called SARS (Self Adaptive Reduce Scheduling) for reduce tasks’ start times in the Hadoop platform. It can decide the start time point of each reduce task dynamically according to each job context, including the task completion time and the size of map output. Through estimating job completion time, reduce completion time, and system average response it is also proved that the average response time is decreased by 11% to 29%, when the SARS algorithm is applied to the traditional job scheduling algorithms FIFO, Fair Scheduler, and Capacity Scheduler. [4] [5] [6] [7]

A. Problem Analysis

Hadoop allows the user to configure the job, submit it, control its execution, and query the state. Every job consists of independent tasks, and each task needs to have a system slot to run. Fig. 5 shows the time delay and slot resources waste problem in reduce scheduling. Y-axis in Fig. 5 means the resource slots, which is represented by Map slots and Reduce slots. At first in Fig.5 we can know that Job1 and Job2 are the current running jobs, before the time t2, each job is allocated two map slots to run respective tasks. Because the reduce tasks will begin once any map task finishes, from the duration t1 to t2, there are two reduce tasks from Job1 and Job2 running respectively. Since the execution time of each task is usually not the same, Job2 finishes its map tasks at time t2 and reduce tasks at time t3. Because the reduce tasks of Job1 are not finish in this time t3, as indicated in Fig. 5, there are two new reduce tasks from Job1 started.

Now all the reduce slot resources are taken up by Job1. In Fig. 5, at the moment t4, when Job3 starts, two idle map slots can be assigned to it, and the reduce tasks from this job will then start. However, we can find all reduce slots are already occupied by the Job1, and the reduce tasks from Job3 have to wait for slot release.

The root cause of this problem is that reduce task of Job3 must wait for all the reduce tasks of Job1 to be completed, as Job1 takes up all the reduce slots and Hadoop system does not support preemptive action acquisitely.
scheduling algorithm, which gives a method to estimate the start time of a task, instead of the traditional mechanism where the reduce tasks are started once any map task is completed.

The reduce process can be divided into the following several phases. Firstly, the reduce task requests to read each map output data in the copy phase, which belong to this reduce function in the map output data blocks. [5] [6]

Next, in the sort process, these intermediate data are output to an ordered data set by merging, which are divided into two types. One type is the data in memory. When the data are read from various maps at the same time, the data should be merged as the same keys. The other is as like the circle buffer. Because the memory belonging to the reduce task is limited, the data in the buffer should be written to disks regularly in advance. Next, in the sort process, these intermediate data are output to an ordered data set by merging, which are divided into two types. One type is the data in memory.

When the data are read from various maps at the same time, the data should be merged as the same keys. The other is as like the circle buffer.

V. SARS ALGORITHM (A SELF-ADAPTIVE REDUCE SCHEDULING ALGORITHM)

A. Reduce Task Algorithm

M: the map slots collection;
R: the reduce slots collection;
Q: the job queue.

1) \( MT = \text{obtainMapTask} (M, Q); \)
2) start the map task in MT;
3) obtain KVS as the middle key-value set;
4) for each job \( j \in Q \) do
5) for each key \( k \in \text{KVS} \) do
6) \( S_{\text{MapTask}} = 0; /\text{all maps’ finish time} \)
7) \( S_{\text{MapOut}} = 0; /\text{all maps’ output} \)
8) for each \( I \in [1, m] \) do
9) \( S_{\text{MapTask}+} = t_{\text{mapi}}; \)
10) \( S_{\text{MapOut}+} = m_{\text{outi}}; \)
11) end for
12) //get the starting time of reduce task:
13) \( \text{startreduce} = \text{startmap} + \frac{S_{\text{MapTask}}}{s} - \frac{S_{\text{MapOut}}}{\text{transSpeed} \times \text{copyThread}}; \)
14) if System.currentTime = startreduce then
15) if job exists waiting reduce task then
16) remove a reduce slot from R;
17) start the reduce task;
18) else
19) break;
20) end if
21) else
22) break;
23) end if
24) end for
25) end for

B. Map Task Algorithm

M: the map slots collection;
Q: the job queue.

Ensure:

MT: map task set;

1) \( m: \) the count of current map tasks.
2) for each job \( j \in Q \) do
3) \( \text{count} = 0; \)
4) if job has a waiting map task \( t \) then
5) \( MT = MT \cup \{t\}; \)
6) remove a map slot from M;
7) \( \text{count}++; \)
8) else
9) break;
10) end if
11) end for
12) \( m = \text{count}; \)
13) return MT. [6]
VI. COMPARISON BETWEEN BASIC SCHEDULING ALGORITHMS AND SARS ALGORITHMS ON WORD COUNT

Fig. 7: Comparison between FIFO and SARS algorithms on Word count

Fig. 8: Comparison between FAIR and SARS algorithms on Word count

VII. CONCLUSION

The goal of the improved algorithm proposed in this report is to decrease the completion time of reduce tasks in the Map Reduce framework. This method can decide reduce task start time through running situation of the jobs, and to avoid waiting around of the reduce function. In this paper, the performance of this algorithm is estimated from the job completion time; reduce completion time, and the system average response time. So in SARS algorithm we have to estimate the point at which reduces function starts.

VIII. FUTURE WORK

We saw a multiple, robust scheduling algorithm, SARS, which uses to improve scheduling environment in reduce phase but as per its disadvantage it does not consider mapping phase and add the mapping phase algorithm and analyze the result. And also implement a technique to identify that at which point we start the Reduce function and it benefited the algorithm most. So, now in future we select more efficient mapping algorithm with SARS algorithm and check the result and analysis of it in Hadoop cluster.

REFERENCES


[4] Improving Map Reduce Performance Using Smart Speculative Execution Strateg Qi Chen,Cheng Liu, and Zhen Xiao, Senior Member, IEEE


[7] A Comprehensive View of Hadoop Map Reduce Scheduling Algorithms Seyed Reza Pakize Department of Computer, Islamic Azad University, Yazd Branch, Yazd, Iran

