Survey on Task Assignment Techniques in Hadoop

Sarika Patil
Student, ME
Dept. of IT
Pune Institute of Computer Technology,Pune, India

Shyam Deshmukh
Assistant Professor
Dept. of IT
Pune Institute of Computer Technology,Pune, India

ABSTRACT
MapReduce is an implementation for processing large scale data parallelly. Actual benefits of MapReduce occur when this framework is implemented in large scale, shared nothing cluster. MapReduce framework abstracts the complexity of running distributed data processing across multiple nodes in cluster. Hadoop is open source implementation of MapReduce framework, which processes the vast amount of data in parallel on large clusters. In Hadoop pluggable scheduler was implemented, because of this several algorithms have been developed till now. This paper presents the different schedulers used for Hadoop.

General Terms
MapReduce, Scheduling,

Keywords
MapReduce, Task Assignment, Resource Management, Scheduling.

1. INTRODUCTION
As use of internet is growing day by day it leads to handle too much data by Internet service providers. MapReduce [2] is now popular solution for developing large scale distributed data applications. MapReduce framework is used for processing hundreds of terabytes of data. It is used to build fault tolerant and scalable applications.

Number of organizations across the world uses Apache Hadoop [1] which is an open source implementation of MapReduce. If any node crashes, Hadoop reassign task to another node. Hadoop scheduler handles all these tasks. There are number of Hadoop scheduler improvements. Task assignment in Hadoop is important to reduce runtime and improve resource utilization. This paper illustrate the various schedulers for Hadoop.

1.1.Hadoop
Hadoop has been used by many companies including Amazon, Facebook, and Yahoo. Hadoop hides the details of parallel processing, including data distribution to processing nodes. Hadoop includes 1) Hadoop Distributed File System (HDFS) 2) Hadoop MapReduce.

1.1.1 Hadoop Distributed File System
HDFS is a block oriented file system. Individual files are divided into blocks of 64MB. These blocks are stored across machines of a cluster which having data storage capacity. Individual machines in the cluster are called as DataNode. Any file is made of several blocks, and these blocks are not stored on same machine.

The target machines which hold each block are chosen randomly on a block-by-block basis. Thus access to a file may requires the co-operation of multiple machines, but supports the large file size which is larger than a single machine DFS. The Problem of unavailability due to failure of any node in the cluster is solved in HDFS by replicating each block across number of machines, by default it is 3.

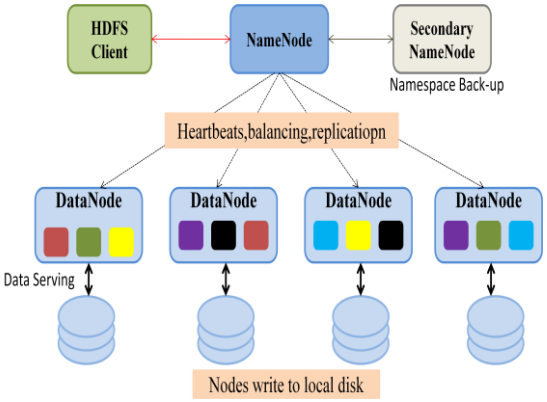

Fig.1HDFS Architecture

As shown in fig. 1, HDFS cluster consist of a single node known as a NameNode, which manages the file system namespace and regulates client access to files. DataNode store data as blocks within files. NameNode is responsible for mapping of data blocks to DataNodes. Also NameNode manages file system operations like opening, closing, renaming files and directories.

The NameNode information must be preserved even after the NameNode machine fails. There are multiple copies of data on NameNode is maintained on number of machines. So that in case of crashes of NameNode these node can be used by other nodes in cluster. These nodes are called as secondary NameNode.

1.1.2 Hadoop MapReduce
MapReduce was originally proposed by Google to handle large scale web search applications. This approach has been proved to be an effective programming approach for developing machine learning, data mining and search applications in data centres.

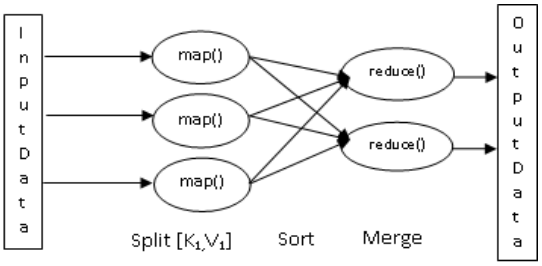

Fig.2 MapReduce

As shown in fig.2 MapReduce programming model consist of data processing functions Map and Reduce. Parallel Map tasks are run on input data which is partitioned into fixed sized blocks and produce intermediate output as a collection of <key, value> pairs. These pairs are shuffled across different reduce tasks based on <key, value> pairs.

Hadoop MapReduce architecture consists of one JobTracker (Master) and many TaskTrackers (Workers) as shown in fig. 3. The JobTracker first determines the split from input data and select some TaskTracker based on their network distance to the data source. Each TaskTracker periodically submits its status report to JobTracker via heartbeat message.

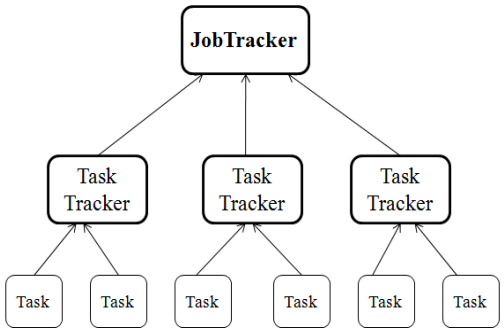

Fig.3 Hadoop MapReduce Architecture

Task or worker failures are handled by relaunching tasks on other nodes. The JobTracker keeps track of heartbeat messages coming from TaskTrackers and uses it in task assignment.

2. RELATED WORK

2.1 Scheduling In Hadoop

For task assignment in Hadoop various scheduling algorithms have been developed. Many research papers such as [4],[5],[6],[7],[8],[10],[13] are focusing on this issue. Hadoop's pluggable schedulers facilitate to design effective schedulers and as per requirements of application.

2.1.1 Default FIFO Scheduler

The default scheduler in Hadoop uses First-In First-Out scheme for scheduling of tasks. When job is submitted, it is divided into individual small tasks, these tasks are queued into the queue. Tasks are assigned to available free slots of TaskTracker nodes. There is also provision for assignment of priorities to jobs, but this is not turned on by default. This approach was simple to implement, but for fair sharing of resources better scheduler is required.

2.1.2 Fair Scheduler

The fair scheduler [4] in Hadoop was developed by Facebook. The main idea behind Fair scheduler was assign resources to jobs such that on average, each job gets equal share of available resources. Therefore the job that requires less time to execute is able to access CPU and other resources with other jobs that require more time to execute. This type of behaviour allows some interactivity among Hadoop jobs. In Fair Scheduler set of pools are created into which jobs are placed for selection, with each pool assigned guaranteed minimum number of map and reduce slots. Free slots in idle pools may be allocated to other pools and excess capacity within a pool is shared among jobs. The Fair Scheduler supports preemption, so if pool has not received its fair share then scheduler will kill tasks in pools running to lower the resource consumption. There is also provision of priority settings for jobs within pool.

2.1.3 Capacity Scheduler

The Capacity Scheduler [5] was originally developed by Yahoo. It was defined for large clusters. In capacity scheduling number of queues are created each with a configurable number of map and reduce slots. Every queue has assigned guaranteed capacity. During execution queues are monitored, if its whole capacity is not used its excess capacity used for another queue temporarily. Capacity scheduling has the ability to prioritize the jobs within a queue. So job with high priority have access to resources earlier than low priority jobs. There is strict access control on queues (queues are tied to a person or organization). These access controls are defined as per queue basis.

2.1.4 Longest Approximate Time to End (LATE)

It is common to any task to continue to progress slowly. This may be due to the load on CPU or slow processes running in background. All tasks need to be executed for completion of a job. The scheduler tries to detect slow running task and starts another equivalent task as backup which is called as speculative execution. If backup task completes faster, then only job performance is improved. Hence speculative execution is just optimization not a feature to ensure reliability of jobs. If task is slow due to errors in code speculative execution is of no use, because same error may affect the speculative task also. This needs to uncover bugs in the code. Also speculative execution relies on some assumptions such as uniform task progress on nodes and uniform computing environment. So this implementation of speculative execution works well in homogeneous clusters. These assumptions not work in heterogeneous environment which are required for real world problems. Zaharia et al [6] proposed an algorithm called Longest Approximate Time to End (LATE). It uses remaining time to complete the execution of task instead of progress made by the task. They have proved the significant improvement in the job response time over the default speculative execution.

2.1.5 Delay Scheduling

Basically Fair Scheduler is developed to allocate fair share of capacity to all users. But in Fair sharing there are two locality problems are identified head-of-line scheduling and sticky slots. The first problem is occurring with respect to small jobs. Whenever job reaches at the head of sorted list, one of its tasks is assigned to next free slot regardless of which node this slot is on. If head-of-line job is small then there are fewer chances for data locality on the node that is given to it. The second locality problem, sticky slots, is that there are chances for a job to be assigned the same slot repeatedly. These problems are occurred because of strict queuing order followed for scheduling job with no local data.

To overcome this problem task needs to be schedule on a node near their input data. Running task on node that contains input data is most efficient, but when this is not possible running task on a node on the same rack is allowed. Delay Scheduling [7], is a solution that temporarily relaxes fairness by forcing job to wait for a scheduling opportunity on a node with local data. The concept of this delay scheduling is that when node requests for new task, if the head-of-line job cannot launch a local task, it is skipped and subsequent job is considered for scheduling. If job is skipped number of times, non-local tasks are allowed to launch to avoid starvation. The main idea behind Delay Scheduling is that although the first slot considered is unlikely to have data for it, tasks finishes so

quickly that some slot with data required for it will free up in the next few seconds.

2.1.6 Dynamic Priority Scheduling

Dynamic priority scheduler [8] provides a way to automatically manage user job QoS based on demand. This scheduler supports capacity distribution dynamically among concurrent users depending on priorities of the users. In this method Map and Reduce slots are allocated on proportional share basis. These time slots can be configured by users as per needs and called as allocation interval. Whenever users are not using guaranteed slots, other users can use these slots. Spending rate is the amount will be payed from the budget per Map or Reduce task. Preemption is supported but can be disabled. While pre-empting task, the shortest running task is killed first. Users claiming higher priority will have to pay for it. The important feature of this method is it limits the free riding and gaming by users. This scheduler easily configured into other scheduler behaviour. When no queues or credits left the scheduler reduces to a FIFO Scheduler.

2.1.7 Deadline Constraint Scheduler

Deadline constraint scheduler [9] takes into consideration deadlines of the jobs and tries to increase system utilization. Deadline requirements in Hadoop based data processing is done by 1) a job execution cost model that considers various parameters like map and reduce runtime, input data size, data distribution, 2) a constraint based Hadoop scheduler that takes user deadlines as part of its input. Estimation model is based on set of assumptions:

- All nodes are homogeneous and unit cost for processing map and reduce node is equal.
- Input data is distributed uniformly.
- Reduce tasks starts after all map tasks have completed.
- The input data is available in HDFS.

Whether job will be scheduled or not is depending on job execution cost model independent of number of jobs running in the cluster. Jobs are scheduled only if deadline can be met. After job is submitted, test for determining deadline is performed. Free slots are computed at the given time or in the future regardless of all the jobs running in the system. If job can be completed within deadline then it is enlisted for scheduling.

2.1.8 Resource Aware Scheduling

The Fair scheduler and Capacity Scheduler are trying to allocate capacity fairly among jobs and users. These scheduler does not consider the resource availability. If data-intensive or processor-intensive task are scheduled on node with slow processors it will overload the node.

Resource Aware scheduling tries to deal with this problem. Scheduling in Hadoop is centralized, and worker initiated. Job Tracker which is master node takes scheduling decisions whereas worker nodes called as TaskTracker are responsible for executing these tasks. The JobTtracker have detailed information regarding status of TaskTracker, currently running jobs and list of tasks allocated to each node. Even though this information is maintained per node basis to reflect the actual processing power available on cluster machines, there is no online modification system available. Hence in this situation congestion on node cannot be avoided by advertising a reduced capacity on node. Hence to overcome these problems basic three resources must be tracked all the time before task scheduling i.e. CPU, memory, disk IO.

The two mechanisms for resource aware scheduling [10] are: 1) Dynamic Free Slot Advertisement - Instead of having fixed number of available computation slots configured on each TaskTracker node, this number can be computed dynamically using resource metrics received from each node. The overall resource availability can be set to minimum availability across all resource metrics. In a cluster that is not running at maximum utilization at all times, there is improve in job response time as no machine is running task in manner such that it leads to resource bottleneck. 2)Free Slot Priorities/Filtering-In this approach, cluster administrator will configure maximum number of compute slots per node at configuration time. The order in which free Task Tracker slots are advertised is decided according to their resource availability. As free slots are available on TaskTracker they are buffered for some small time and advertised in a block. TaskTracker with higher resource availability are presented first for task scheduling. When short jobs takes long time to complete, this will present significant performance gains.

2.1.9 Learning Scheduler

Many organizations schedule periodic Hadoop jobs to pre-processes raw information. Repetitive nature of these applications provides opportunity to use performance data from past runs of the application and integrate that data into resource management algorithms. Because of this reason learning approaches can be applied to scheduler in Hadoop. Learning Scheduler [11] automatically supervised pattern classification for learning the impact of different MapReduce applications on system. Classifier is used to predict the outcome of queued tasks on node utilization. This classifier labels job as good or bad depending on their resource utilization. Only good jobs are considered for task assignment such that it will not overload the node. It uses dynamic and static properties of computational resources for classification. There are job features are like CPU utilization, Memory, Disk IO, Network usage of jobs. These features can be calculated from past execution traces of the job. The node features are of two type Static and Dynamic. Static features include number of processors, processor speed, physical memory, number of disks. Dynamic features include CPU usage, IO rate, Network usage, number of processes running. Utility functions are used to prioritize jobs and for policy enforcement. After task is assigned, effect of task assignment is observed from information contained in the subsequent Heartbeat message from TaskTracker. If TaskTracker is overloaded such task assignment was incorrect. The pattern classifier is trained to avoid such assignment in future.

3.CONCLUSION

To increase performance of Hadoop till now number of variations in scheduling strategies are emerged. Resource aware scheduling is emerging research problem that attracts most of the researchers as current implementation is based on static configuration of slots. Also learning scheduler approach for job classification as good or bad is research area in Hadoop scheduling. Machine learning approaches can be applied to predict node failures in Hadoop. Use of most efficient classifier for classification is also future direction in learning scheduler. This paper summarizes the pros and cons of different scheduling policies developed by different communities.

4. REFERENCES

[1] Hadoop http://hadoop.apache.org/

[2] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters, "In Communications of the ACM, Volume 51, Issue 1, pp.

107-113, 2008.J.Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon,1892, pp.68–73.

[3] B.Thirumala Rao and Dr. L.S.S. Reddy,"Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments "

[4] Matei Zaharia, "The Hadoop Fair Scheduler" http://developer.yahoo.net/blogs/hadoop/FairSharePres.ppt.

[5] Hadoop's Capacity Scheduler, http://hadoop.apache.org/core/docs/current/capacity_scheduler.html.

[6] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleely, Scott Shenker and Ion Stoica, "Delay Scheduling: A Simple Technique For Achieving Locality and Fairness in Cluster Scheduling"

[7] Thomas Sandholm and Kevin Lai. Dynamic proportional share scheduling in hadoop. In JSSPP '10: 15th Workshop on Job Scheduling Strategies for Parallel Processing, April,2010

[8] K. Kc and K. Anyanwu, "Scheduling Hadoop Jobs to Meet Deadlines", in Proc. CloudCom, 2010, pp.388-392.

[9] Mark Yong, Nitin Garegrat, Shiwali Mohan: "Towards a Resource Aware Scheduler in Hadoop" in Proc. ICWS, 2009, pp:102-109

[10] Jaideep Dhok and Vasudeva varma," Using Pattern Classification for Task Assignment in MapReduce"

[11] Jaideep Dhok, Nitesh Maheshwari and Vasudeva Varma,"Learning Based Opportunistic Admission Control Algorithm for MapReduce as a Service"

[12] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/.