An Efficient Data Placement Algorithm For Hadoop Heterogeneous Clusters

Mr. N. Velmurugan, Associate Professor
Department of Information Technology
K.L.N. College of Information Technology
Madurai, India.
e-mail id: nvel_murugan@yahoo.com

Ms. C. Suhanya Devi, PG student
Department of Information Technology
K.L.N. College of Information Technology
Madurai, India
e-mail id: suhanya29@gmail.com

Abstract—Cloud computing is one of the parallel distributed computing system that has become a frequently used computer application. MapReduce is an present programming model used in cloud computing. Hadoop is an open-source application of the MapReduce model. The current Hadoop implementation assumes that every node in a cluster has the same computing capacity and that the tasks are data-local, which may increase extra above and reduce MapReduce performance. This paper proposes a scheduling algorithm resolve the job execution problem in the heterogeneous environment. The proposed method can dynamically adapt and balance data stored in each node based on the computing capacity of each node in a heterogeneous Hadoop cluster. The proposed method can decrease data transfer time to achieve improved Hadoop performance. The experimental results show that the scheduling can decrease the time of execution and improve Hadoop performance in a heterogeneous cluster.

Keywords—Hadoop, MapReduce, Heterogeneous

1. INTRODUCTION:

In Hadoop, there is a single master managing a number of slaves. The controlling node consists of a NameNode, DataNode, JobTracker, and TaskTracker. A slave node acts as both a DataNode and TaskTracker, though it is conceivable to have data only worker nodes, and compute-only worker nodes. Namenode holds the filesystem metadata. The files are broken up and spread over the datanodes. JobTracker schedules and manages jobs. TaskTracker executes the individual map and reduce functions. If a machine fails, Hadoop continues to operate the cluster by shifting work to the remaining machines. The input file, which resides on a distributed filesystem throughout the cluster, is split into even-sized large piece replicated for fault-acceptance. Hadoop divides each MapReduce job into a set of tasks. Each large piece of input is processed by a map task, which outputs a list of key-value pairs. In Hadoop, the shuffle phase occurs as the data is processed by the mapper. During execution, each mapper hashes the key of each key/value pair into bins, where each bin is associated with a reducer task and each mapper writes its output to disk to ensure fault acceptance. Since Hadoop assumes that any mapper is equally likely to produce any key, each reducer may possibly receive data from any mapper. The intermediate key/value pair from the map task is passed on to a partitioner which in turn calls the partitioner function. It takes as input the key/value pair and returns the reducer to which this key/value pair should be sent. In Hadoop, the default partitioner is HashPartitioner, which hashes a record key and the number of reducers to determine which partition (and thus which reducer) the record belongs to. The number of partitions is equal to the number of reduce tasks for the job. The amount of data received from each mapper to a reducer and the total size of data to be processed by the reduce task will only be known after the map tasks complete execution. This leads to a load imbalance because the reducer roles are fixed before the map tasks start in the current implementation.

Many cloud applications largely assume a homogeneous environment. Hadoop is an open source implementation of MapReduce, a framework for big data analysis that runs on a cluster of nodes. A Hadoop job is consists of a number of tasks that run on nodes concurrently. When Hadoop schedules a task of a job, it assumes that it takes about the same time to process a task regardless of where it runs. It reflects network connectivity by giving preference to tasks that access local data over these access remote data, but does not consider the alteration of...
computing ability of nodes. However, in a heterogeneous environment, some tasks run faster on a particular node than others. In addition, it is not straightforward to guarantee fairness among multiple jobs in heterogeneous environments. The value of a particular node depends on the job of which task is assigned to the node. Hence, simply adding up the number of nodes or relative speed that are used by a job does not account for the job's share. Thus, such differences in the value of nodes depend on the job. We call this job affinity. It must be considered when we want a job scheduler to ensure that each job receives its fair share of processing resources.

2. RELATED WORK:

Related work discussed about a MapReduce operation, submitting a MapReduce job

2.1 MAPREDUCE OPERATION

MapReduce operation consists of two functions. The Mapper function reads a stream of data and parses it into intermediate (key, value) pairs. When that is complete, the Reducer function is called once for each unique key that was generated by Map and is given the key and a list of all values that were generated for that key as a parameter. The keys are presented in sorted order. File placed in HDFS, is broken down into blocks. These blocks are then replicated across the different nodes (Data Nodes/ Slave Nodes/ Chunk Servers) in the cluster. File location is maintained by the NameNode by a corresponding entry of it location. Then the word count operation is done on the Created cluster by using a given mapper and reducer methods. In this algorithm each node i uses Hash function to find out its immediate neighbour. After finding the neighbour, each node calculates its neighbour node metrics such as CPU utilization or network traffic etc. Based on this neighbour node monitoring, if the neighbour node is underutilized corresponding node i can transfer its load to the neighbour node to balance the load equally. If the neighbour node is over utilized, then it automatically adds one data node to manage the traffic. And also make the alert to the user about the rebalancing operation.

2.2 Elastic Mapreduce (Emr)

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process huge volumes of data. Amazon EMR uses Hadoop, an open source framework, to allocate your data and processing across a sizable cluster of Amazon EC2 instances. Amazon EMR is used in a change of applications, containing log analysis, web indexing, data warehousing, machine learning, economic analysis, scientific model, and bioinformatics. Customers launch millions of Amazon EMR clusters every year.

2.3 SUBMITTING A MAPREDUCE JOB

The process of running a MapReduce job on Hadoop consists of 8 major steps.

**Step 1:** The MapReduce program you've written tells the JobClient to run a MapReduce job.

**Step 2:** This sends a message to the JobTracker which produces a unique ID for the job.

**Step 3:** The JobClient copies job resources, such as a jar file containing a Java code you have written to implement the map or the reduce task, to the shared file system, usually HDFS.

**Step 4:** The resources are in HDFS, the JobClient can tell the JobTracker to start the job.

**Step 5:** The JobTracker does its own initialization for the job. It calculates how to split the data so that it can send each "split" to a different mapper process to maximize throughput.

**Step 6:** It retrieves these "input splits" from the distributed file system.

**Step 7:** The TaskTrackers are continually sending heartbeat messages to the JobTracker.

**Step 8:** The JobTracker has work for them, it will return a map task or a reduce task as a response to the heartbeat. The TaskTrackers must to obtain the code to execute, so they get it from the shared file system.

**Step 9:** Then they can launch a Java Virtual Machine with a child process running in it.
Step 10: This child process runs your map code or your reduce code.

3. PROPOSED SYSTEM
In proposed system Resource allocation and job scheduling using hadoop cluster. The job scheduling algorithm mostly using in heterogeneous environments. In this heterogeneity-aware scheduler method uses an advance algorithm of the job scheduling method. This algorithm the reduce the transaction time and increase the performance. The result heterogeneity aware scheduler can decrease the time of execution and improve the hadoop performance in a heterogeneous cluster.

3.1 Heterogeneous Cluster process
Heterogenous task of job A runs three times faster on a slot of N1 than N2, whereas job B runs on N1 as fast as on N2. Figure illustrates how they will be scheduled if the cluster runs only one job. White and gray cells represent the slots of N1 and N2, respectively.

The letter in each cell indicates the job occupying the slot. The graph below shows the progress of each job over time. he graph at the bottom-left shows the progress of each job over time. It is drawn only to the point at which there are enough tasks to schedule. Note that the translucent lines describes the progress each job would make if they occupied the whole cluster without sharing. The graphs on the right side track the change in the slot share and the progress share of each job. Even though each job occupies the same number (three) of slots at all times, the progress share of job A often falls below its fair share (0.5) because many tasks of job A run on slots of N2, which is not suitable for the job. As a result, job A is making less than half progress compared to the job that occupies the whole cluster. Obviously, job A is under-served in this scenario even though the job received enough slot-share. In contrast, Figure shows an example of progress share-based scheduling. The graph at the bottom-left shows that both jobs A and B are making more Progress than they might otherwise.

3.2 Scheduling based on slot share
3.3 Scheduler

To calculate the progress share of each job, the MapReduce system should be aware of the per-slot computing rate (CR). To that end, each job goes through two phases: calibration and normal. When a job is submitted, it starts with the calibration phase. In this phase, at least one map task is launched on each type of node. By measuring the completion time of these tasks, the scheduler can determine the CR. Once the scheduler knows the CR, the job enters the normal phase. During the normal phase, the scheduler works similar to the Hadoop fair scheduler. When a slot becomes available, a job of which the share is less than its minimum or fair share is selected. However, if there is another job with a significantly higher computing rate on the slot, the scheduler chooses that job to improve overall performance. This is similar to the “delay scheduling” mechanism in the Hadoop fair scheduler. By using progress share, the scheduler can make an appropriate decision. As a result, the cluster is better utilized. In addition, each job receives a fair amount of computing resources.

3.1 Number of pending jobs

4. CONCLUSION

This paper proposes a this heterogeneity-aware scheduler algorithm for the heterogeneous cluster. The Hadoop default data placement strategy is supposed to
be applied in a homogeneous environment. The proposed heterogeneity-aware scheduler algorithm is based on the different computing capacities of nodes to allocate data blocks and improve data locality and increase the performance.

5. REFERENCES