In practice, a Hadoop cluster consists with tens to thousands of slave nodes. These slave nodes are put in server racks in a data center. Network traffic through different racks is known to be more expensive compared to the traffic within a rack.

HDFS (Hadoop Distributed File System), the storage layer of Hadoop, consists with namenode for managing the filesystem and datanodes that actually saves data.

HDFS stores files in the unit of blocks. The default block size is 128MB and it can be changed according to the requirements. A file is broken into blocks and stored in datanodes. By default, each block is replicated to three separate physical machines for fault tolerance. Considering the large number of servers and hard disks used in a Hadoop cluster, hard disk failure is very common. If a block becomes unavailable in a server, that data is copied from a different server.

HDFS is designed to select different datanodes in separate racks when data is written to HDFS. The first replica is placed at a datanode in the local rack as the client that is writing the data. The second replica is placed at a datanode in a different rack from the first replica. The last replica is placed at a different datanode in the same rack as the second replica. This means that in one rack there are two replicas of a given block.

Add “Replica placement” content in P.72

When a HDFS client wants to read a file from HDFS, first, the client contacts the namenode to get a list of datanodes that hold a copy of the first few blocks of the file. This datanodes list is sorted according to the distance to the client for each block. Then the client connects to the closest datanode of each block, according the datanodes list, and reads the data. The important point is that the client only reads block data from the closest datanode, even though it is possible to read the same block data from the rest of the datanodes in parallel and combining them at the client.

Let us consider reading a 128MB block from one datanode only and reading the same block data from two datanodes in parallel and combine them at the client. We consider the processing time at the client side such as combining the data is considerably small compared to data transfer through network. Let network transfer speed be 1Mbit/sec through the network and client reads data from two datanodes in parallel with the ratio of 7:3. It takes (128 x 8)/1 = 1024 seconds to read one block if the client reads data from one datanode only. On the other hand, it takes (128 x 8 x 0.7)/1 = 716.8 seconds if data is read from two datanodes. In this paper we propose 1) a method to calculate the optimal ratio to read data from two datanodes 2) modified version of Hadoop that reads data from multiple datanodes in parallel.

Problem Formulation

When a job runs and data locality is not available, blocks are read from HDFS to the server running that task.

Job scheduler assigns tasks to take the nodes where input data is available. This is called data locality. One data block is replicated on three data nodes by default. Job scheduler assigns a task randomly if all the nodes containing the input data for that task are already busy with other tasks. In the current implementation of Hadoop, the node that was assigned the task gets the data from only the closest node that contains the input data. System administrator predefines the distance between nodes/racks manually. This is called rack awareness in Hadoop. When the task assigned node reads data from the source, it only copies from only one node, where the same data is available in other two nodes. Network distance from the task assigned node for these two nodes might be longer than the nearest node, but it is also possible to utilize these other two nodes to transfer the data needed for the task assigned. Finding the ratio of the data is the problem.