Nicolas Escobar (escobarn)
Harsh Reddy (hagandav)

Cloud Computing - Project 1 Report

In this report we are providing details regarding the functionality of our MapReduce project, specifically on the 1) *transformation of data during the computations*, 2) *the data structure used to transfer between Map and Reduce phases* and 3) *how the data flow happens between disk and memory during the computation*.

Transformation of data during the computations, i.e. data type of key, value

Map Task

1. A new dummy key of type Text is generated to be used in the map output
2. The value received as a parameter in the map function is converted to a Float with the Float.parseFloat function.
3. A new instance of a FloatWritable is created with the number of type Float as an argument and passed to the Context.write function as a value parameter to generate the map output.

Reduce Task

1. A new MultipleWritable class that implements Writable is created to handle the composite values that are generated in the reduce function (minimum, maximum, average, standard deviation).
2. A float (or double in case of standard deviation) variable for each calculation is created to store the results of the operations.
3. Once the calculations are done, the results are assigned to the variables of the MultipleWritable class object (result) and written as values of the reduce output.

The data structure used to transfer between Map and Reduce phases

The output that is “transferred” from the Map phase to the Reduce phase is of type *<Text, FloatWritable>*. The Text data type is used for the variable that stores a dummy key and the FloatWritable for the variable that stores the actual floating point number.

The reduce function which is part of the Reduce phase will receive as inputs the dummy key of type Text and a list of FloatWritable values contained in *Iterable<FloatWritable>*. This list will contain the list of floating point numbers that were produced in the Map phase.

Once the reduce function performs the calculations over the list of floating point numbers, the output generated is of type *<Text, MultipleWritable>* where the key is again a dummy variable of type Text and the actual results of the calculations are represented as variables of MultipleWritable type, which implements a Writable interface to be able to display multiple values as a single result.

How the data flow happens through disk and memory during the computation

Map Task
The input data which is stored in HDFS is split into data chunks or “input splits”. Each split is assigned to a map task which runs the map function of our MapReduceStats class. For data locality purposes, Hadoop will try to run the map task on the same node of the input data but there are instances when all nodes having a replica of the map input split data are busy running other tasks (since we are running the application in a standalone fashion, this is not applicable).

The output of our map tasks will be initially written to a local buffer in memory and when it reaches a certain threshold size, data will be spilled to local disk (not to HDFS). Just before writing to disk, the data will be partitioned to reflect the data partitions that will be later sent to the reducers. In case a Combiner is used, then less data will be written to disk and sent to the reducer but we haven't used one in this exercise because it was not a requirement.

In the ‘MapReduceStats’ class we have implemented, the map output will be a dummy blank key with the list of all numbers read from the input file.

Reduce Task
The *Copy phase* of the Reduce task implies fetching the map outputs from the different nodes in the cluster. The threads in the Reduce task are in charge of doing the copy and the data is copied to the JVM’s memory of the Reduce task if it fits, or otherwise it is written to disk.

When all the map outputs have been copied, the Reduce task moves into the *Sort phase* which merges all these outputs in different iterations depending on the merge factor.

The final *Reduce phase* receives from the Sort phase a combination of disk and in-memory data which is used by the reduce function to apply the final merge. In the context of our project, the reduce function is invoked for a single dummy key which contains all the values (floating point numbers) and calculates the maximum, minimum, average and standard deviation. The final output with the result of the calculations of the Reduce phase are stored in HDFS filesystem.

The flow can be observed graphically in the Figure below, obtained from *Hadoop: The Definitive Guide* book.

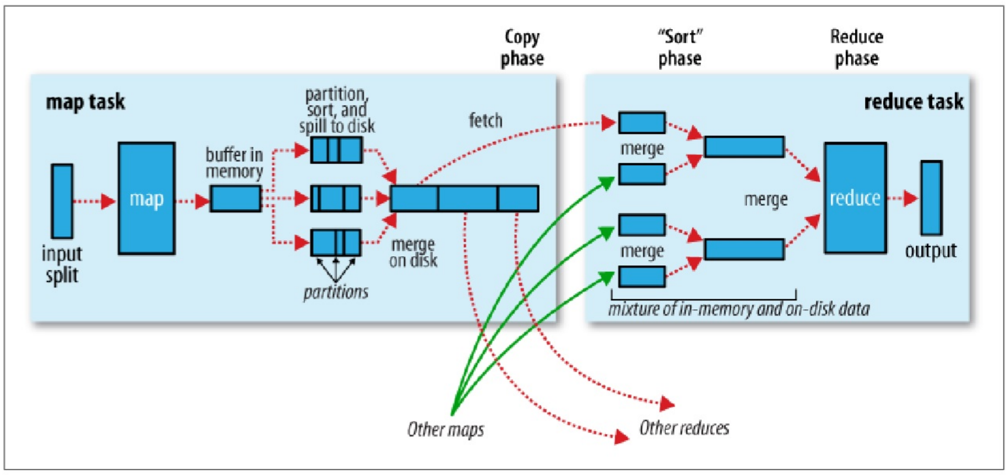

Figure 6-6. Shuffle and sort in MapReduce

References: 1) Hadoop: The Definitive Guide