Lab 3 – Map reduce algorithm

Java implementation of Map Reduce :

Word Count map reduce algorithm searches for all the keys and adds a value of 1 to it, irrespective of previous occurrence.

Current algorithm removes the need of reduce class and computes the value dynamically by adding the value to the existing map key and sorts the map based on most number of occurrences.

Output is a sorted map with word and number of occurrences without the need of reducer.

Above algorithm is not suitable for HDFS. This is just an implementation of Mapreduce in java.

Algorithm change for optimization :

Simple word count example doesn't use in-class or typical combiner.

Typical combiner reduces the distributed map in a node, which will make the work of reducer easy. But typical combiner has a buffer limit for example consider the below map:

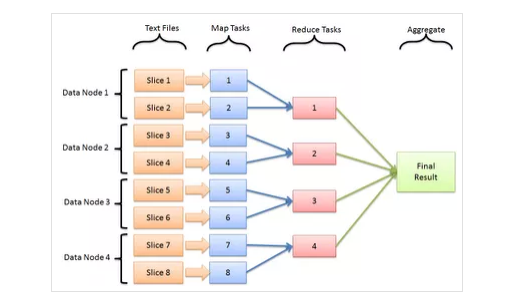
<560, 1>,<551,4>,<560,3>,<560,6>

If the buffer size be 3, the typical combiner will give a result of <560,4>,<551,4>,<560,6>

We can implement an in-class combiner which has a buffer to reduce all the map in a node, which in turn reduces the task of main reduce class and optimizes the problem.

This can be implemented only in a distributed cluster, a single node cannot take advantage of this method.

This can be achieved by having a temporary hash Map as buffer for the node . Map in that node can be reduced with the help of buffer, and sent to main reducer class for final reduction.



Above layout explain the HDFS system. Having a combiner in individual map tasks would reduce the work of reduce tasks, making the algorithm more efficient.

References

<http://www.javacodegeeks.com/2012/05/mapreduce-for-dummies.html>