|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Counter Name | Case 1 | Case 2 | Case 3 | Case 4 |
| 1.FILE: Number of bytes read | 485132 | 27082 | 485198 | 27148 |
| 2.FILE: Number of bytes written | 1256677 | 340921 | 2834345 | 1920546 |
| 3.HDFS: Number of bytes written | 21812 | 20779 | 21812 | 2165952 |
| 4.Total time spent by all map tasks (ms) | 4130 | 4365 | 4251 | 4520 |
| 5.Total time spent by all reduce tasks (ms) | 3784 | 3450 | 237407 | 241433 |
| 6.Combine input records | 0 | 20103 | 0 | 20103 |
| 7.Combine output records | 0 | 1372 | 0 | 1372 |
| 8.Reduce shuffle bytes | 485132 | 27082 | 485198 | 27148 |
| 9.Reduce input records | 20103 | 1372 | 20103 | 1372 |
| 10.Reduce output records | 1372 | 1309 | 1372 | 1372 |
| 11.Spilled Records | 40206 | 2744 | 40206 | 2744 |
| 12.Shuffled Maps | 1 | 1 | 12 | 12 |
| 13.Merged Map outputs | 1 | 1 | 12 | 12 |
| 14.Total committed heap usage (bytes) | 226627584 | 226627584 | 896339968 | 896339968 |
|  |  |  |  |  |

I number each row and will number them in my explanation on the next page\*

1.For this the only columns that are worse is the columns without the combiner. Not really sure how the file read is smaller in bytes just by havin the combiner because to my knowledge the combiner is coming in after the file has been read and is through the mapper but the combiner sure helps in reducing the data read.

2.Another example of the combiner really pulling its weight although the numbers for each are all over the place the number that sticks out is case 3 and case 2. Case 3 being the highest and I think that is because the partitioner is making more work by having it write to separate files and when you add the combiner back in for case 4 it drops it almost by a million. Case 2 I think is the lowest because it gets rid of the partitioner and adds the combiner which I have already said really pulls its slack.

3.Not really 100% sure why only case 4 is so different, but all I can come up with is the HDFS is having to write a lot more having all the parts in motion.

4.These values are only slightly different and I think it is because it is for the map tasks which is the same being the mapper is the first to get ahold of the data.

5.The time went up for the reducer tasks when adding the partitioner into the picture and I think that is only because of having to write the reduced data to the different files. You can also see that without the partitioner and just the combiner it is the lowest value in the row.

6.The columns are paired into with or without the combiner which makes sense the combine input record goes up when using a combiner.

7.This is the same as above the values go up when having the combiner.

8. What seems to be the pattern so far is when adding the combiner in it really drops the amount of data dealt with.

9.Same as the above when adding the combiner it does what it is supposed to do by being a mini reducer to help the reducer.

10.The smallest value is when just using the combiner without the partitioner although it is not a big change it shows again that the combiner by itself does a good job at what it is made for to help the process.

11.Spillled records are controlled when adding the combiner.

12.Shuffeled maps with partitioner are at the number of partitions set which was 12 vs the regular one file without partitioner.

13.Same as the above answer when data is partitioned you get the number allocated to the partitioner.

14.The heap as stated many times throughout this explanation is when adding the partitioner it really adds the work and data to the work for the MapReduce.