**Metropolitan State University**

**ICS 411-01 Big Data Storage and Processing**

**Fall 2019 - Assignment 3 Report**

**Case 1:**

Just Mapper and reducer.

**Case 2:**

Mapper, combiner, and reducer.

**Case 3:**

Mapper, partitioner, and reducer.

**Case 4:**

Mapper, combiner, partitioner, and reducer.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Counter Name** | **Case 1** | **Case 2** | **Case 3** | **Case 4** |
| FILE: Number of bytes read | 404720 | 25814 | 2268650 | 2261324 |
| FILE: Number of bytes written | 1095817 | 338337 | 6544308 | 6532036 |
| HDFS: Number of bytes written | 20779 | 20779 | 2178867 | 2178867 |
| Total time spent by all map tasks (ms) | 10763 | 3146 | 3440 | 2358 |
| Total time spent by all reduce tasks (ms) | 4708 | 3068 | 68614 | 70938 |
| Combine input records | 0 | 20103 | 0 | 20103 |
| Combine output records | 0 | 1309 | 0 | 20030 |
| Reduce shuffle bytes | 404720 | 25814 | 2268650 | 2261324 |
| Reduce input records | 20103 | 1309 | 20103 | 20030 |
| Reduce output records | 1309 | 1309 | 20030 | 20030 |
| Spilled Records | 40206 | 2618 | 40206 | 40060 |
| Shuffled Maps | 1 | 1 | 13 | 13 |
| Merged Map outputs | 1 | 1 | 13 | 13 |
| Total committed heap usage (bytes) | 362283008 | 548405248 | 2583691264 | 2457862144 |

*(All lines have differences; explanations for them on the following page.)*

**FILE: Number of bytes read / FILE: Number of bytes written**

Efficiency appears to have improved in Case 2 due to the use of the combiner. This should have been true for Case 4 as well, despite the table showing otherwise.

**HDFS: Number of bytes written**

The difference here appears to be related to the fact that the partitioner is utilizing 13 different reducers.

**Total time spent by all map tasks (ms)**

Map tasks in Case 2 ran for a shorter duration than in Case 1 due to the use of a combiner, which essentially reuses the Reducer class’s functionality to sort and shuffle the data before passing the data to the Reducer for final reduction.

**Total time spent by all reduce tasks (ms)**

Case 3 and Case 4 took longer than Case 1 and Case 2 due to the use of the custom partitioner. The activity of sorting out the data and assigning it to a specific reducer is the likely cause.

**Combine input records / Combine output records**

Only Case 2 and Case 4 have any combiner input records since those were the only cases for which a combiner was used.

**Reduce shuffle bytes**

There were many more reduce shuffle bytes for Case 3 and Case 4, likely due to the use of a partitioner. Data needed to be shuffled to more than just one reducer.

**Reduce input records / Reduce output records**

The combiner helped with these two items in Case 2, but oddly it should have also helped with Case 4, despite the table showing that it didn’t.

**Spilled Records**

There were fewer spilled records (records written to disk) for Case 2 than for Case 1 due to using the combiner. The combiner helped by reducing the amount of data needing to be passed into the partitioner. This should also have been the same for Case 4 versus Case 3, despite the table showing otherwise.

**Shuffled Maps / Merged Map outputs**

These two counters reflect the number of reducers running. The first three columns show a value of 1, and the last column shows 13 because Case 4 represents running the job using a custom partitioner (using 13 reduce tasks).

**Total committed heap usage (bytes)**