CS6240
Assignment 2
Ajay Vardhan

Running times are calculated by subtracting the start and end times in the syslog.

EMR first run running times:

Program	**Running time** (seconds)
No Combiner	76
Combiner	78
In Mapper Combiner	70

EMR second run running times

Program	**Running time** (seconds)
No Combiner	74
Combiner	77
In Mapper Combiner	71

Was the Combiner called at all in program Combiner? Was it called more than once per Map task?

Yes. The combiner was called in the program. This can be verified by checking the Combine input records and Combine output records values in the syslog file. The number of combine input records is equal to the number of map output records.

Since the combiner has the same number of input records as the map emitted, it is not possible that the combiner is called multiple times for the same data. And since we never know when a combiner is called in the program, we can't accurately calculate the number of times it was called.

What difference did the use of a Combiner make in Combiner compared to NoCombiner?

The combiner significantly reduced the number of input records processed by the reducer. Since most the records were combined before the records were sent to the reducer, the workload of the reducers decreased a lot. The number of I/O operations were also reduced since the map didn't emit each and every record to the reducer. The buffered records were processed by the combiners and the aggregated records were only emitted to the reducers which saves a lot of I/O and the reducer workload. We will see a difference in running time when working on larger datasets.

Was the local aggregation effective in InMapperComb compared to NoCombiner?

Yes. The local aggregation definitely improved the processing speed. As we can see from the running times, the InMapperComb program as taken lesser time than NoCombiner. We will see better differences when processing even larger datasets. The number of records processed by the reducer is reduced here as well. Since the records were aggregated in the mapper, before being transferred, the I/O processes were also significantly reduced which made the lesser running time possible.

Which one is better, Combiner or InMapperComb? Briefly justify your answer.

InMapperComb is definitely better since the data is aggregated even before it is generated. Combiner waits until the data is generated and then processes it. InMapperComb also guarantees that all the data is aggregated before it is sent out to the reducer but we never know if a combiner will be excited or not. The number of records that the Mapper emitted is lesser. The number of I/O operations is also lesser for InMapperComb. Overall, InMapperComb is better than Combiner.

How do the running times and accuracy of these MapReduce programs compare to the sequential implementation of per-station mean temperature?

Both the programs had the same results. There wasn't any problem with the correctness but the sequential execution ran much faster than the MapReduce programs. This is because we are using a very small dataset, there is no transfer of data anywhere and all the data are processed altogether. The Mapreduce programs running in EMR have a lot of data transfers and distributed system management. This will affect for small datasets. Since MapReduce programs are designed for very large datasets, we will see drastic difference in processing time between MR and sequential programs. If the data is too large for a single processor to process, the MR programs will easily outrun the sequential program since there is no parallelism there.

Secondary sort EMR running time – 51 seconds