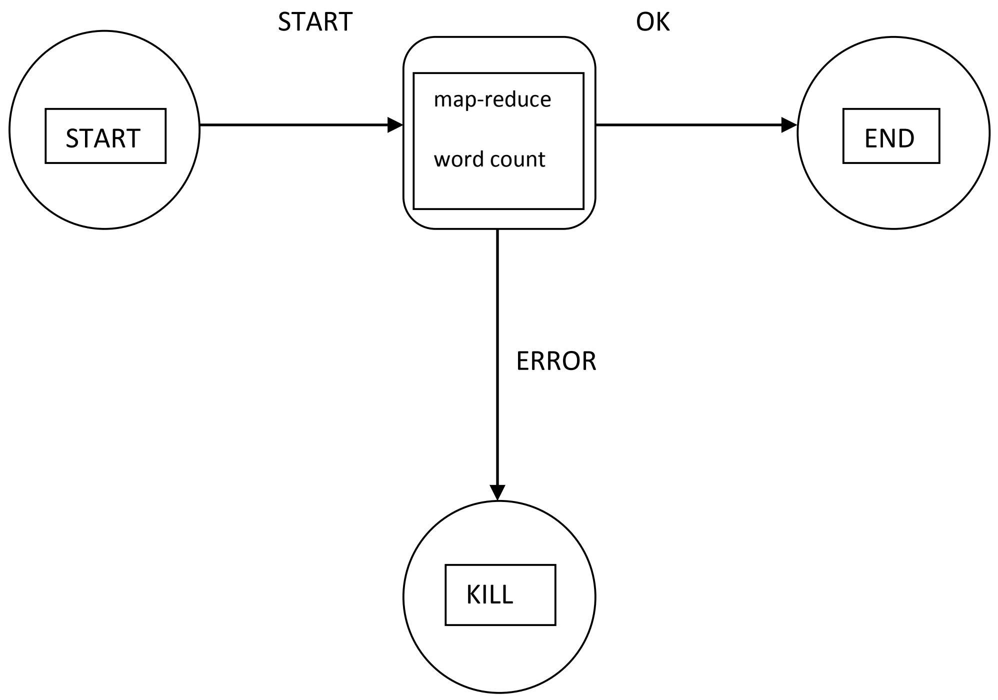
OOZIE WORKFLOW DIAGRAM:



Algorithm for Mappers and Reducers

1. The input data can be divided into n number of chunks depending upon the amount of data and processing capacity of individual unit.

   partition (k’, number of partitions) ? partition for k’

   Often a simple hash of the key, e.g., hash(k’) mod n

   Divides up key space for parallel reduce operations

2. Next, it is passed to the mapper functions. Please note that all the chunks are processed simultaneously at the same time, which embraces the parallel processing of data.

   map (k, v) ? <k’, v’>*

3. Shuffling leads to aggregation of similar patterns.

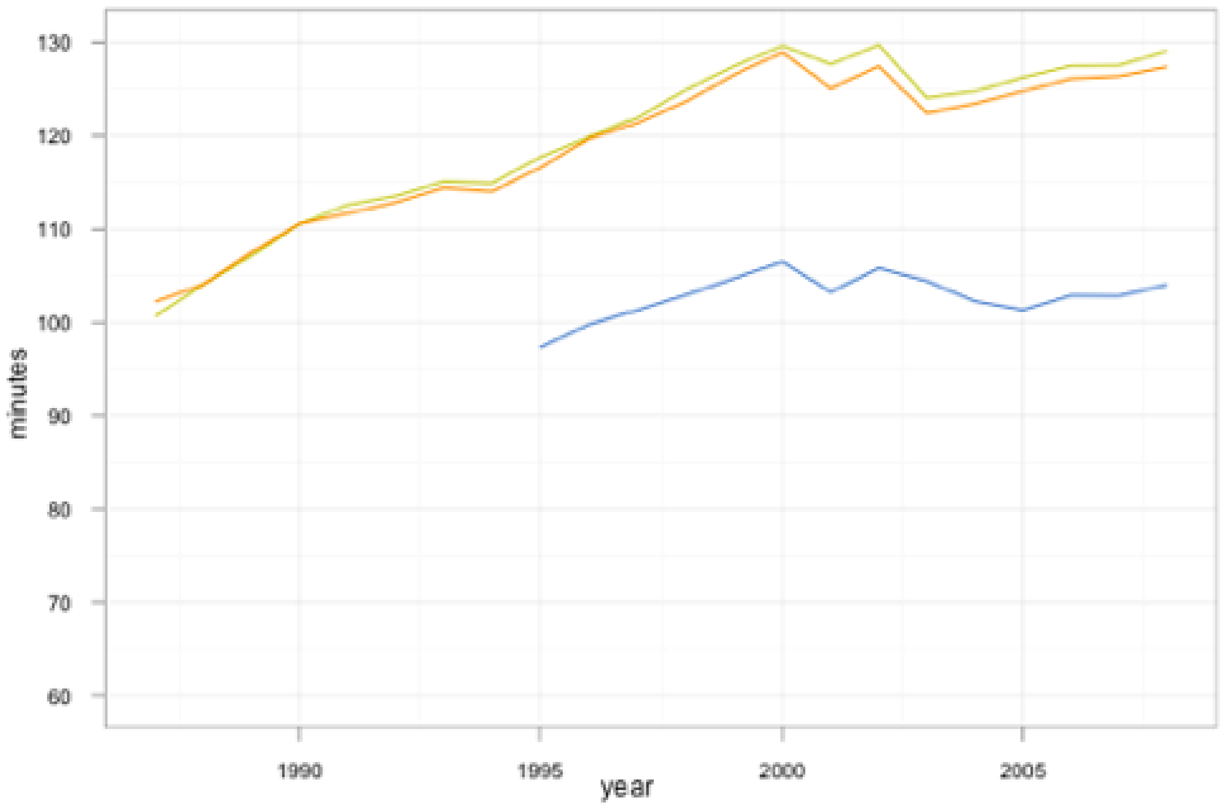
4. Finally, reducers combine them all to get a consolidated output as per the logic.

   reduce (k’, v’) ? <k’, v’>*. All values with the same key are sent to the same reducer.

   Mini-reducers that run in memory after the map phase.

   Used as an optimization to reduce network traffic.

5. This algorithm embraces scalability as depending on the size of the input data, we can keep increasing the number of the parallel processing units.


130
120
110
100
90
80
70
60
minutes
1990
1995
2000
2005
year