REPORT 2 DECEMBER 2017

Experimental setup:

All experiments are performed on wrangler using
8 concurrent producers which run on a single wrangler node.
Producer:

- 8 producers
- Each one produces total of 22223 messages/records
- Message size is 5000K 3-dim points which is equivalent to an average of 310KBs
- Each one produces on average 1450 messages/minute
- Therefore, all together producer ~11600 messages/min
- 3.5MB/s/ second

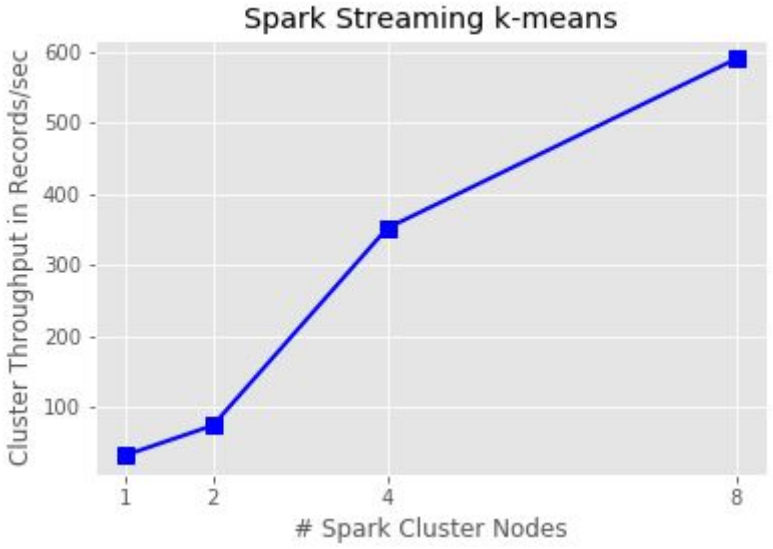
Consumer (Spark):

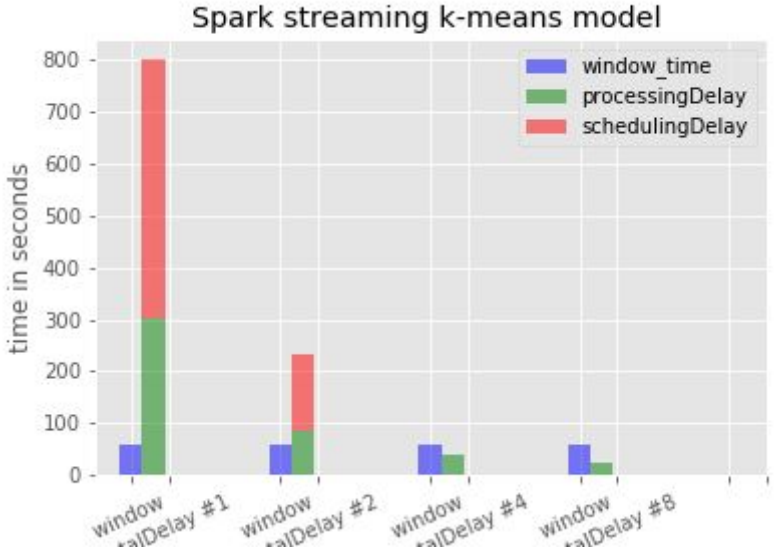
- Runs streaming k-means application
- Ratio of kafka partitions/cores = 1
- Window size is 60 seconds
- Each minibatch handles all the incoming messages, therefore is the data through- in is not the bottleneck.

At this experiment I vary the number of spark nodes and calculate the Processing throughput/#minibatch of the spark cluster.

Processing throughput = (Number of Records) / (totalDelay)
totalDelay = schedulingDelay + processingDelay
SchedulingDelay: time to partition and schedule the new databatch from the moment they arrive until the moment the processing starts. In order to start processing the new minibatch the previous should have finished.

processsingDelay: time to process the data batch





- By adding spark nodes the cluster throughput is increasing. The rate is increasing exponentially from 2 to 4 nodes because the scheduling delay disappears.
- Optimal configuration is with 4 spark nodes, where the processing throughput is ~ 90 records/seconds

- There is no point in increasing more than 4 nodes the number of cluster nodes because the data processing takes less than the window size(60 seconds), which means than the consumer fininishes the data processing and waits

- Next measurements :
    - Increase the produce rate from 3.5MB/s to 7MB and iterate.