**SF Crime Statistics with Spark Streaming**

**Step 3 :**

1. How did changing values on the SparkSession property parameters affect the throughput and latency of the data?

For ingesting data in real time from a streaming service is utmost important to maintain coordination between the batch size (number of events received each stream) , frequency of streams received and time consumed in ingesting/processing that dataset.

The throughput I/O can be altered by increasing the number of input Kafka streams (parallelism) or by increasing spark.default.parallelism configuration property.

maxRatePerPartition : It is the maximum rate at which each Kafka partition will be read by the API. (messages/sec)

Keeping right coordination between batch size and time interval each batch must be received can help optimizing throughput and latency

maxOffsetsPerTrigger

Memory optimization is another way to handle the throughput and latency

The latency can be managed by altering the execution mode and running Spark in standalone mode.

1. What were the 2-3 most efficient SparkSession property key/value pairs? Through testing multiple variations on values, how can you tell these were the most optimal?

The main objective of any streaming service is to process the data in real time – optimized throughput , memory allocation , processing time , fault tolerant and low latency.

The 2-3 most efficient SparkSession property key/value pairs are:

1. spark.default.parallelism or increase number of input Kafka streams – with these configurations we can avoid any wait time or bottlenecks in data ingestion
2. spark.streaming.blockinterval or inputStream.repartition – with this configuration we can avoid overlapping or extra wait time between streams received.
3. spark.streaming.receiver.writeAheadLog.enable or use of watermark – keeps our data fault tolerant.

Ref:

<https://spark.apache.org/docs/2.1.0/structured-streaming-kafka-integration.html>

<https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html>