Scenario

A typical supercomputer has thousands of sensors recording hardware data (among other things) such as voltages, temperatures, error codes, etc… Abnormal temperature readings are particularly indicative of failure (or potential failure). We are going to process a raw stream of temperature sensor data to get experience working with both single event and windowed streaming processing.

Assignment

Using the provided Java temperature data simulator, we will stream raw data into Kafka. From there you will practice using Spark to work with the stream of data.

Input

Use the (provided) temperature simulator to stream simulated data into Kafka (wget to VM is easiest)

https://bigdata220w18.blob.core.windows.net/blobs/temperatureSimulator.jar

java -jar temperatureSimulator.jar KAFKA_HOST KAFKA_PORT TOPIC

e.g. java -jar temperatureSimulator.jar sandbox.hortonworks.com 6667 raw_temps

Input data is a JSON string of the following schema:

```
{ “server” : String,
  “timestamp”: Timestamp,
   “sensors” :
       {sensor1 : Integer,
        sensor2 : Integer,
          …
        sensor10 : Integer
       }
}
```

Example Input:

```
{"server" : "c20-6c0s0", "timestamp" : "2018-03-03 01:23:03",
"sensors" : {"sensor1" : 24, "sensor2" : 35, "sensor3" : 24,
"sensor4" : 21, "sensor5" : 23, "sensor6" : 38, "sensor7" : 23,
"sensor8" : 22, "sensor9" : 24, "sensor10" : 35}}
```

```
{"server" : "c20-6c1s0", "timestamp" : "2018-03-03 01:23:03",
"sensors" : {"sensor1" : 27, "sensor2" : 38, "sensor3" : 27,
"sensor4" : 22, "sensor5" : 27, "sensor6" : 41, "sensor7" : 27,
"sensor8" : 22, "sensor9" : 28, "sensor10" : 43}}

{"server" : "c23-7c2s7", "timestamp" : "2018-03-03 01:23:03",
"sensors" : {"sensor1" : 31, "sensor2" : 40, "sensor3" : 31,
"sensor4" : 22, "sensor5" : 30, "sensor6" : 41, "sensor7" : 30,
"sensor8" : 25, "sensor9" : 31, "sensor10" : 40}}

{"server" : "c22-7c2s3", "timestamp" : "2018-03-03 01:23:03",
"sensors" : {"sensor1" : , "sensor2" : , "sensor3" : , "sensor4" : ,
"sensor5" : 28, "sensor6" : , "sensor7" : 28, "sensor8" : , "sensor9"
: , "sensor10" : }}
```

Task

Write a Spark streaming application(s) to connect to the stream and do the following tasks. Initially I wanted to write to another Kafka topic but I may have encountered a bug when working with the assignment so I'll make that optional.

- Archive all events to disk (local or HDFS) or just stream to console (or both)
- Write any invalid records to console or to disk (or both)
    - An invalid record is defined as any sensor value being 0 or negative
        - Note that some sensor readings might be null, that is valid input
        - Can include the entire record as the value, no key needed
- Of the records that are valid (i.e. NOT invalid as defined before), write the average value seen per server for sensors 1 and 2 to the console.

Bonus Tasks

- Do #3 from required only instead write to kafka topics “avg1” and “avg2” respectively. Use the server as the key and the average as the value. If you encounter an error about casting when trying to start the query, just submit the code you were trying to execute.
- Of the records that are valid (i.e. NOT invalid as defined before), write the average value seen over a window of 5 minutes sliding every 1 minutes per server for sensors 1 and 2 to either console or kafka topic “windowAvg1” and “windowAvg2” respectively. Use the server as the key and the average as the value if writing to Kafka and if gett an error about casting when trying to start the query, just submit the code you were trying to execute.

Hints

You will likely find the following two articles helpful from this week's reading

https://databricks.com/blog/2017/04/26/processing-data-in-apache-kafka-with-structured-streaming-in-apache-spark-2-2.html

https://databricks.com/blog/2017/02/23/working-complex-data-formats-structured-streaming-apache-spark-2-1.html

The first explains the low level details when working with Kafka that we didn't have time to get into in class. The second article is a general knowledge one about working with dataframes that will be useful in this assignment.

When reading the data, since we have JSON input data, you will find it easiest to use the from_json function (not covered in class, but in the provided materials).

When writing to kafka, note that the result dataframe must have at least a "value" column, and that is what will be written to kafka as the value field of the record. If you are adding a key, then it would need a column called (unsurprisingly) key.