# Big Data Technologies 774/874

## Post Block Assignment 1: Streaming

## Overview

### Part 1

- To apply some of the concepts from the lectures and tutorials.
- To consider implementations and architectures in the streaming arena.

### Part 2

- Perform hands-on processing tasks using streaming technologies.

### Submission

- A form will be provided on SUNLearn for completion of this assignment.
- Please submit your code blocks for code related questions (text areas will be included on SUNLearn).

## Part 1: Streaming and Big Data Processing

**Question 1: Beam [21]** Your organisation wishes to add streaming capabilities to their big data analytics and processing stack. They have a few near real-time usecases they wish to work on and have on-premises compute cluster capabilities. They are not planning to migrate to the cloud in the forseeable future. The most likely candidate for processing has been selected as Apache Beam, but now a decision as to which distributed backend to use is being considered. Select two prominent Beam runners and,

1. motivate why you chose these runners over others [2],
2. compile a table of relevant features (see vs-dbms lecture for how we did it with NoSQL) by which to compare them; provide descriptions and background on the relevance of those features in your own words (for each technology) [15],
3. make a recommendation as to which technology to choose based on your comparison [2],
4. lastly, do you support the choice of Apache Beam as the processing tool in general? Motivate your answer. [2]

**Question 2: Kafka [5]** Consider a fraud detection implementation in Kafka using KSQL.

1. Describe what this SQL is doing [3]:

```
CREATE STREAM ATM_TXNS (
        account_id VARCHAR,
        atm VARCHAR,
        location STRUCT<lon DOUBLE,
                        lat DOUBLE>,
        amount INT,
        transaction_id VARCHAR)
    WITH (
        KAFKA_TOPIC='atm_txns',
        VALUE_FORMAT='JSON');
```

2. Why is it necessary to join two streams together in their implementation? [2].

**Question 3: Out of Order/Late Arriving [3]** Consider a usecase where you may not tolerate data loss (e.g. billing internet usage at an ISP). Packets/events may arrive out-of-order, but with a time difference between event time and processing time of at most *10s*.

1. What windowing options should you choose in Beam. Briefly explain your answer? [3]

**Question 4: Architecture [12]** Provide a critical evaluation of the Lambda and Kappa architectures for big data platforms (e.g. What are the differences between these approaches? What are the technologies used and how are they used? What kinds of workloads do they serve? Where is data science performed within these systems?) [12]

## Part 2: Processing with Beam

**Question 1: Merge PCollections [9]** Consider the data from the streaming tutorial. The second file received (*orders.csv*) contains a data excerpt from the retailer's orders database. Combine the 2 input files (the other being *users.csv*) in order to provide insights into the average ordering profiles of their customers.

Expand your pipeline from the tutorial to do the following:

1. Merge the 2 input files [3]
2. Perform a transformation that determines the average number of orders for female and male customers respectively [3]
3. Perform a transformation that groups users into 4 equal age groups (15-25, 25-35, 35-45, 45-55) and determine the total number of orders placed by customers in each age group [3]

## For the master's level, Big Data Technology (Eng) 874 (not 774), students

**Question 1: Stream Aggregation [14]** You are working at an ISP and you have high volumes of customer website related data streaming into your infrastructure (URLs visited, and bytes used, for the sake of this example). You wish to aggregate them in near real-time to reduce the data volumes for easier analysis. A record is:

| user_id | fullname | url | timestamp | bytes |
|---|---|---|---|---|
| 01f4f1c2 | Frank Ortmann | www.google.com/colab | 2020-09-15 12:36:19 | 212 |

To reduce data volumes you need to group by `user_id` and `url` within a time window, aggregating the bytes used. Your streams will consume from a PubSub emulator (emulating a true PubSub queue in the cloud) which will be publishing user usage data.

To setup your emulation environment, follow the instructions here, then

1. Perform the aggregation for a fixed time window of 60s. [8]
2. The data stream contains late data. Explain why these data would currently be lost. [2]
3. Implement a strategy to include late data in your output, and compare the output to that of (1). Briefly discuss the difference (if any), as well as any computing implications. [4]