Practical Assignment II (KAFKA)

1 Introduction

The practical assignment II is focused on using Apache Kafka´s Quality Attributes.

2 Description

The project is about a simulation of processing data from sensors in a Kafka cluster. There will be several Use Cases and for each one a solution must be built.

The file sensor.txt contains all the data collected from several sensors. The format is as follows:

XXXXX -> string containing the sensor ID
ZZZ.ZZ -> real number containing a temperature in ºC.
YYYYYY -> integer time stamp

The data is ordered by its timestamp.

3 Use Cases (UC)

General Requirements

- A Kafka Cluster, beyond Zookeeper, includes:
  - 1 to maximum 6 Brokers
  - 1 Topic named as Sensor
  - 1 to maximum 6 Partitions
  - 1 to maximum 3 replicas with. You must choose the advisable value of min.insync.replicas
- Beyond the Kafka Cluster (Brokers and Zookeeper) there are at least three additional processes:
  - PSOURCE:
    - responsible for reading sensor data (records) from the file sensor.txt and sending it to PProducer according to the requirements of each UC
    - there can be only one Thread responsible for reading the data from sensor.txt
    - there must be one Thread sending data for each Kafka Producer
    - Mandatory:
      - data sent to PProducer relies on Java Sockets
      - all Java Threads must run inside the process PSource
      - PSource contains one process only: PSource.java

- PPRODUCER:
  - responsible for receiving records from PSOURCE and sending them to a Kafka Cluster according to the requirements of each UC
  - may include one or more Kafka Producers
  - each Kafka Producer must be implemented as a Java Thread
  - data received from PSource relies on Java Sockets
  - PProducer plays the role of a server and provides one server socket only.
  - all Java Threads must run inside the process PProducer
  - PProducer contains one process only: PProducer.java
  - one GUI by Kafka Producer
  - at least, each GUI must provide:
    - all the records received from PSOURCE
    - total number of records received
    - total number of records received by sensor ID
- PCONSUMER:
  - responsible for reading records from the Kafka Cluster and to process them according to the requirements of each UC
  - may include one or more Kafka Consumers
  - may include one or more Group Consumers
  - one GUI by Kafka Consumer and/or by Group Consumer
  - each Kafka Consumer must be implemented as Java Thread
  - all Java Threads must run inside the process PConsumer
  - PConsumer contains one process only: PConsumer.java
  - at least, each GUI must provide:
    - all the records received from the Kafka Cluster
    - total number of records received
    - total number of records received by sensor ID

- Each Use Case must be implemented in a way it can be demonstrated separately from the remaining UCs
- Create one Java project only (PA2_GXX, XX is your group id) and for each UC:
  - create one Package named as UCn (n is the UC id)
  - 3 processes: PSOURCE, PPRODUCER and PCONSUMER
  - Scripts to launch the Kafka Cluster
- Whenever necessary, concurrency must be used to improve performance
- In each Kafka Producer class and in each Kafka Consumer class it is mandatory to instantiate a Java Properties object and set into it all the required properties
- If a property (from a Kafka Producer or Kafka Consumer) is important you must set its value explicitly even if you intend to use the default value
- You cannot use: idempotence property and Kafka Transactions.
- The solution for each UC must be considered the optimal one

- You must use the KAFKA libraries published in the e-learning platform
- You cannot use MAVEN

UC_1

PSOURCE:

Role:	records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role:	records are displayed and sent to the Kafka Cluster
Performance:	default: do not forget to set the default values explicitly
Data ordering:	PCONSUMER needs to receive and process the records in their original order as in sensor.txt
Data loss:	records can be lost
Constraints:	one Kafka Producer only

PCONSUMER:

Role:	records are read from the Kafka Cluster and displayed
Data ordering:	check if records keep their original order
Data duplication:	any
Constraints:	one Kafka Producer only

UC_2

PSOURCE:

Role:	records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role:	records are displayed and sent to the Kafka Cluster
Performance:	minimize latency
Data ordering:	PCONSUMER needs to receive and process records by sensor ID and then ordered as in sensor.txt
Data loss:	some records can be lost, but minimize the possibility of losing all records of a sensor ID
Constraints:	6 Kafka Producers

PCONSUMER:

Role:	records are read from the Kafka Cluster and displayed
Data ordering:	each Kafka Consumer must check if records are of one sensor ID only and if they keep their relative original order as in sensor.txt
Data duplication:	records can be reprocessed
Constraints:	6 Kafka Consumers

UC_3

SOURCE:

Role:	records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role: records are displayed and sent to the Kafka Cluster
Performance: maximum throughput
Data ordering: records can be reordered
Data loss: minimize the possibility of losing records but while minimizing the impact on the overall performance
Constraints: 3 Kafka Producers

PCONSUMER:

Role: records are read from the Kafka Cluster and displayed
Data duplication: records can be reprocessed but try to avoid some degree of reprocessing
Constraints: 1 Consumer Group with 3 Kafka Consumers

UC_4

SOURCE:

Role: records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role: records are displayed and sent to the Kafka Cluster
Performance: any
Data ordering: records cannot be reordered
Data loss: minimize the possibility of losing records
Constraints: 6 Kafka Producers

PCONSUMER:

Role: records are read from the Kafka Cluster and displayed
Constraints:

UC_5

SOURCE:

Role: records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role: records are displayed and sent to the Kafka Cluster
Constraints: 6 partitions

PCONSUMER:

Role: records are read from the Kafka Cluster and the maximum and minimum temperatures are displayed and computed following the Voting Replication tactic
Data reprocessing: records can be reprocessed
Constraints: 3 Group Consumers each with 3 Consumers

UC_6

SOURCE:

Role: records are read from sensor.txt and sent to PPRODUCER such a way an optimal solution can be implemented

PPRODUCER:

Role:	records are displayed and sent to the Kafka Cluster
Constraints:	6 partitions

PCONSUMER:

Role:	records are read from the Kafka Cluster by a Consumer Group with 3 Consumers the average temperature is displayed and computed following the Voting Replication tactic
Data reprocessing:	minimize the possibility of reprocessing records in case a Kafka Consumer crashes

4 Assignment Submission

- Project and folder root name must be: PA2_GXX, where XX is your group number;
- Each project must be compressed with **7zip** and sent to omp@ua.pt till 2021 May 16 09:00;
- Do not send Kafka libraries but send any additional library required to run your project;
- Report (PDF):
    - For each UC:
        - what is implemented and working correctly
        - what is not implemented or not working correctely
- Assessment Rules for Grading:
    - Each UC 3 points - 3 x 6 = 18 points;
    - Java Doc – 1 point
    - Report – 1 point