Project 2 - Kafka Use Cases

Arquiteturas de Software 2021/2022 - Universidade de Aveiro

Tiago Rainho 92984 - 50%
Diogo Mendonça 93002 - 50%

1. Introduction **1**

2. Graphical User Interface **2**

3. Use Cases **3**
- **UC_1** **3**
- **UC_2** **3**
- **UC_3** **3**
- **UC_4** **4**
- **UC_5** **4**
- **UC_6** **4**

4. Conclusion **5**

1. Introduction

In this work, our goal is to simulate a sensors' data processing system using Kafka Clusters for multiple different use cases (UC), each requiring a distinct solution.

Each data entry goes through 3 entities until it reaches its final stage:

1. PSource - a thread is responsible for reading the data from a .txt file, and another one is responsible for sending aforementioned data to the PProducer through a Java Socket.

2. PProducer - the data is received and then one or more Kafka Producers send said data to the Kafka Cluster, following the respective UC's requirements.

3. PConsumer - one or more Kafka Consumers are in charge of reading the data in the Kafka Cluster, and then the data is processed in order to verify the UC's requirements.

2. Graphical User Interface

For each Kafka Producer and Consumer, it is required to present all the records received, the total number of records received and the total number of records received by sensor ID. To solve this, two GUIs were developed, one for the Producers and one for the Consumers.

The Producers's GUI includes one or more tabs, each for a respective Kafka producer, where the producer's statistics are shown in labels and the received records are shown in a table, as seen in Figure 1.

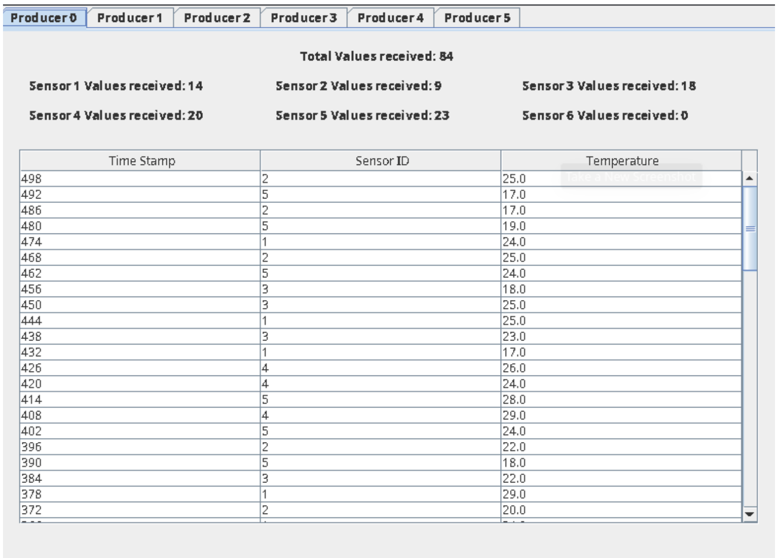

Figure 1: Producers' GUI

The Consumers' GUI is very similar to the previous, but includes the capability of showing consumers groups by aggregating into a tab the tabs respective to a group's consumers, as seen in Figure 2.

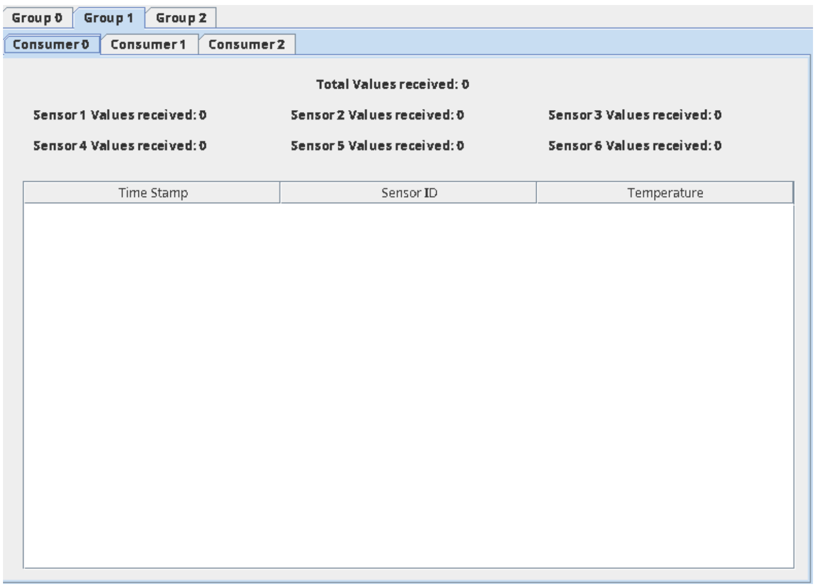

Figure 2: Consumers' GUI

3. Use Cases

Each UC can have different requisites when it comes to data ordering, loss and/or duplication, performance and number of Kafka producers, consumers and consumer groups. In order to fulfill these requisites, each UC defines different Kafka properties for the Kafka brokers, producers and consumers, and may have a slightly altered implementation; all these modifications will be shown and justified below for each UC.

UC_1

In this UC, because records can be lost, a fire-and-forget method was used in the Kafka producers, and the acknowledgements setting was set to 0, so the producer does not wait for a confirmation if the data was received in the cluster. Only 1 broker with 1 partition was used, which allows records to keep their original order more easily, as the Kafka Consumer only gets records from 1 source.

Only 1 Kafka producer and 1 Kafka consumer were used in this UC.

UC_2

In this UC, despite being able to lose some records, the possibility of losing all records of a sensor ID should be minimized, which lead us to use 3 brokers with a replication factor of 3, meaning that if one of the brokers stopped working, there would be 2 copies of each partition.

In order to ease the retrieval of records by sensor ID, 6 partitions were used, one for each sensor. To reduce latency, *linger.ms* and acknowledgements were set to 0.

6 Kafka producer and 6 Kafka consumer were used in this UC, so each consumer will be responsible for reading data of only one partition/sensor, which eases the verification of relative original order.

UC_3

In this UC, the possibility of losing records should be minimized while minimizing the impact on the overall performance, which lead us to use 3 brokers with a replication factor of 2, meaning that if one of the brokers stopped working, there would be a copy of each partition.

To maximize the throughput, the producers' batch size was increased to 100000, the compression type to lz4 and the *linger.ms* to 50.

3 Kafka producers and 3 Kafka consumers (integrated into 1 consumer group) were used in this UC.

UC_4

In this UC, the possibility of losing records must be minimized, so 3 brokers were used with a replication factor of 3.

6 Kafka producers and 9 Kafka consumers (3 consumer groups with 3 consumers each) were used in this UC.

UC_5

In this UC, the maximum and minimum temperatures of each sensor must be calculated using the Voting Replication tactic. This tactic entails that, for a value to be accepted, it must be present in at least half of the Kafka Clusters.

6 Kafka producers, 9 Kafka consumers (3 consumer groups with 3 consumers each) and 3 brokers were used in this UC.

UC_6

In this UC, the average temperature is computed for each sensor.

6 Kafka producers, 3 Kafka consumers (1 consumer groups with 3 consumers each) and 3 brokers were used in this UC.

4. Conclusion

With this project, we could better understand how real-time event driven applications work by using Kafka, an open source, distributed streaming platform.

It was also possible to recognize, in depth, the responsibilities that each Kafka entity has in the task of data processing through a Kafka Cluster, and the multiple configurations available to better suite our implementation to each use case.