POLITECNICO
MILANO 1863

Apache Kafka

Alessandro Margara
alessandro.margara@polimi.it
https://margara.faculty.polimi.it

Exercise 1

- Implement a system with two (basic) consumers for the same topic
  - Consumer C1 prints records on the standard output
  - Consumer C2 processes each value (e.g., removes upper case letters) and stores the result to a new topic
    - Without delivery guarantees

- Consider a single instance for each consumer
  - What happens if one consumer fails?
  - What happens if you restart it?

if consumer C2 fails, consumer C1 won't receive any message. when we restart C2, we need to wait some seconds so that the broker can understand that the old C2 has failed and so the partition has to send to the new C2.

Exercise 2

- Now assume that you realize that consumer C2 is too slow
  - It cannot sustain the rate of messages added to the input topic

- How can you improve the performance of the system?

we can add another C2 instance in the same group of the previous C2. to test it we create two class for C2 (which are equal) and increment the waitBetweenMesage of the two C2 instances (to simulate C2 slow). we need to specify that we want two partitions for topicA

- Experiment with the system
  - What happens if one consumer fails?
  - What happens if you start multiple consumers?

1. the producer still publishes on both the topics. the partitions will be reassigned automatically for the remaining consumers.
if we start multiple consumers, then every consumer will get linked to a partition until we finish the partitions. (3 partitions and 4 consumers means that one consumer doesn't get any message)

Exercise 3

- Now you want C2 to guarantee exactly-once semantics
  - Each input message should be delivered to the output topic once and only once

we implemented this with an atomic forwarder that uses transactions.

- Experiment with the system
  - What happens if one consumer fails?
  - What happens if you start multiple consumers?

if the forwarder fails and then rejoins, then it starts forwarding the messages where it last stopped working

Exercise 4

- Modify C2 to store and forward the overall number of messages received for each key

- Consider a single instance of C2
  - What happens in the case of failure?
  - Does your implementation guarantee exactly-once semantics?

if the forwarder fails, the number of messages for each keys gets lost, so when it rejoins, then it will start from 0.
this means that this implementation is not exactly once. it works like an at most once semantics.

- How do your answers change in the case of multiple instances?