Distributed Fraud Detection System

Luca Arduini
Federico Casu
Daniel Deiana

Going Distributed w/ Apache Kafka (pt.1)

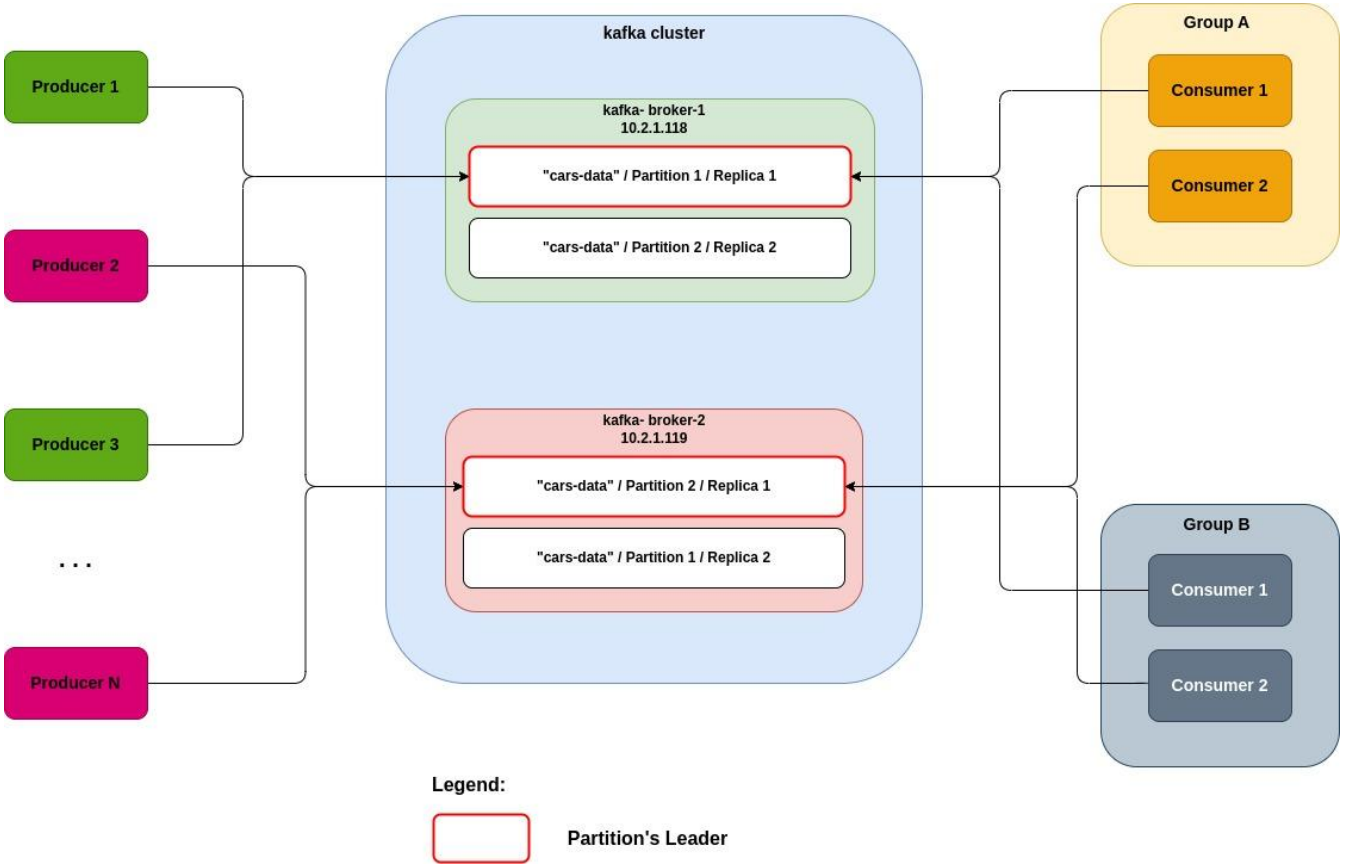
Partitioning

- Kafka topics are divided into partitions.
- Producers publish messages to specific partitions within a topic.
- Kafka assigns a message to a partition based on a *round robin* strategy.

Replication

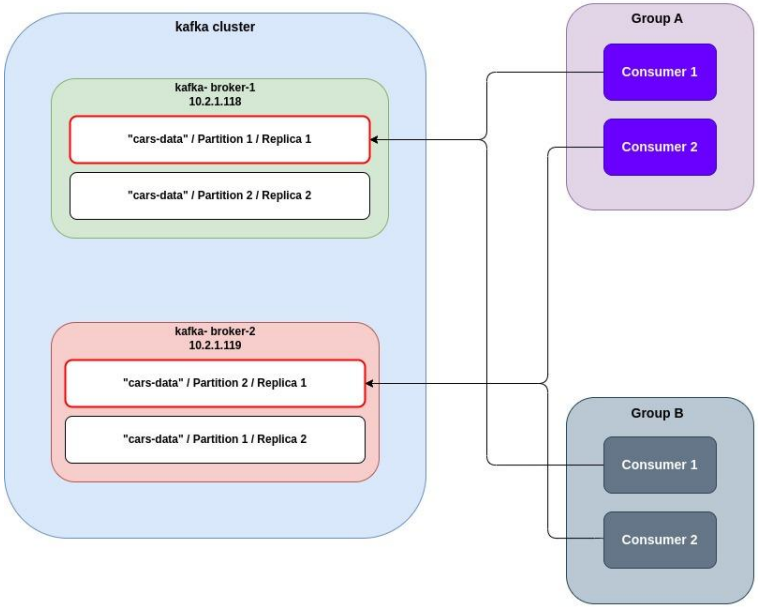
- Partitions are replicated across multiple brokers to ensure fault tolerance and high availability.
- In our system, each partition has *one* leader and *one* replica.
- The leader is responsible for handling all read and write requests for the partition.
- The leader acts as load balancer within a partition (w.r.t. *reads*).

Going Distributed w/ Apache Kafka (pt.2)

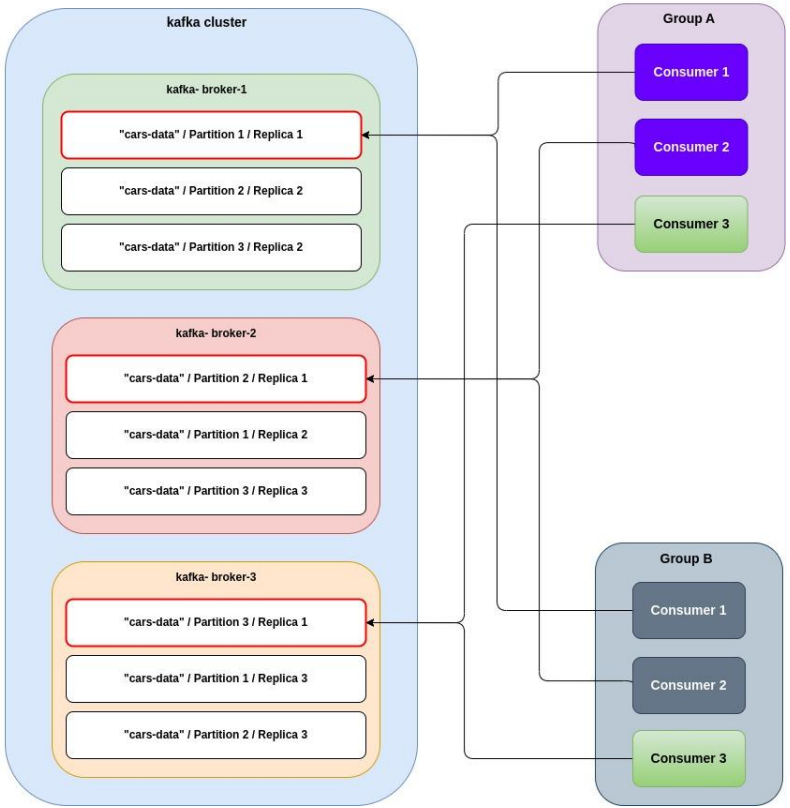


Kafka Consumers: Dynamic Allocation Strategy (pt. 1)

From 2 Partitions and 2 Consumers per Group ...



... To 3 Partitions and 3 Consumers per Group ...



Kafka Consumers: Dynamic Allocation Strategy (pt. 2)

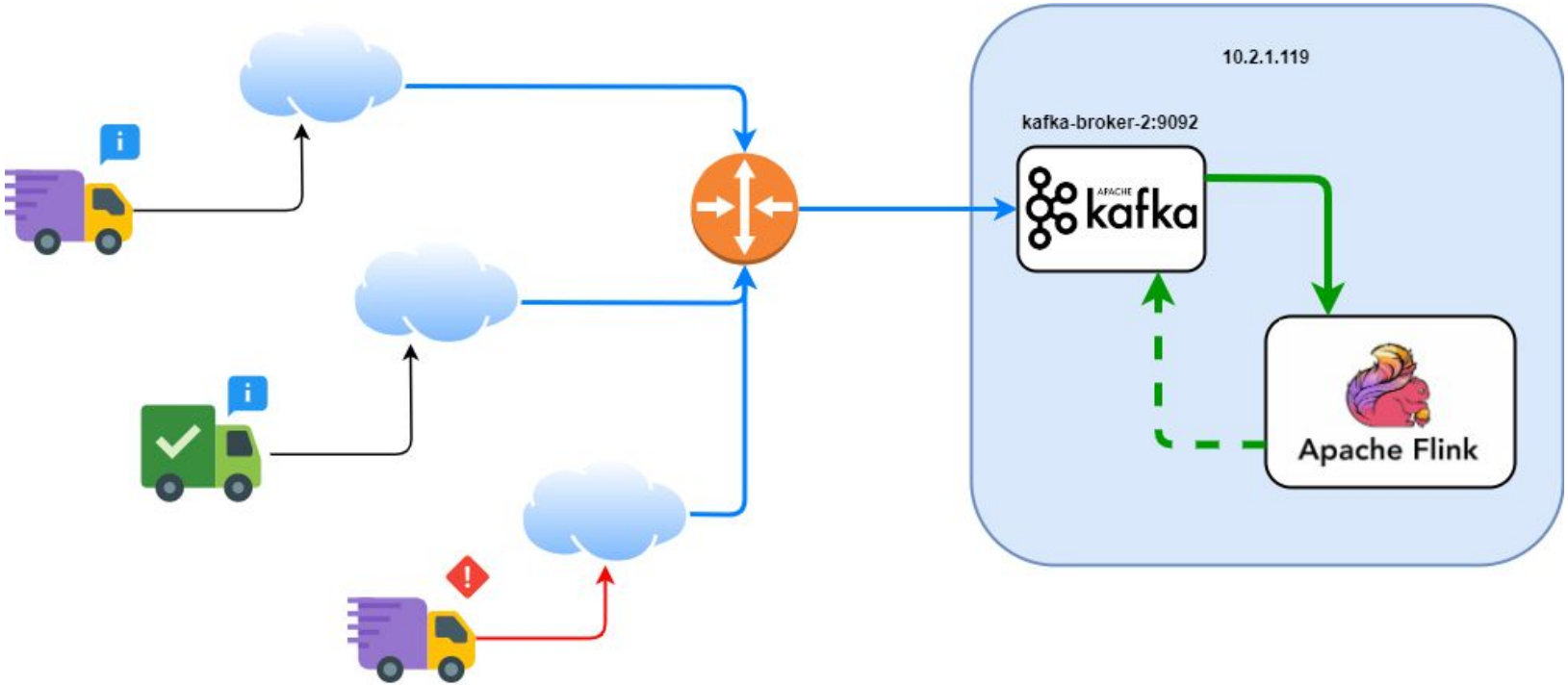
Why number of partitions = number of consumers?

- With fewer partitions than consumers, some consumers will be left idle as they won't have any partitions to consume from.
- With fewer consumers than partitions, some partitions will have more than one consumers and some others will have just one consumer.

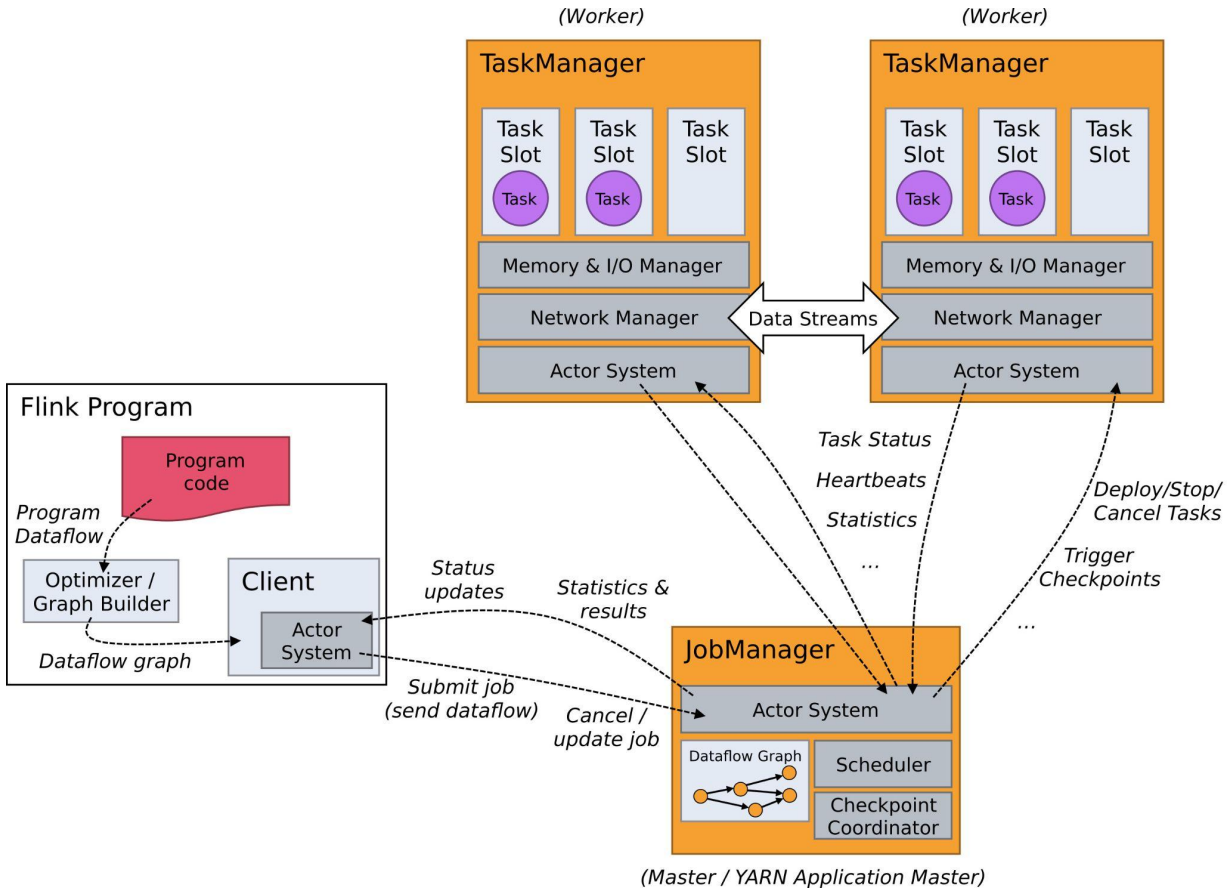
In both cases, the workload is not evenly distributed!

number of partitions = number of consumers ensures the best solution in terms of resources utilization and work load balancing.

Apache Flink what?

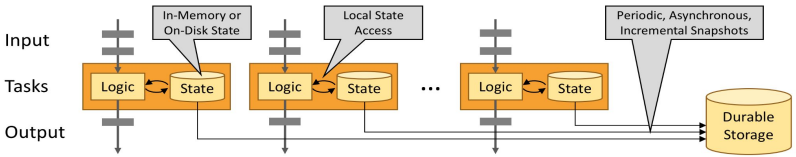


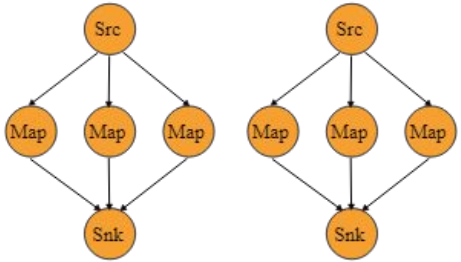
Flink cluster architecture



How flink manages state

- Using attributes of the input data as keys we can create many parallel keyed streams.
- During the computation we can also save the state for a particular keyed stream.





Problems related to distribution

- The use case we imagined is the one of a **large scale** event processing:
  - We partitioned topics data among kafka nodes to address scalability.
  - We also have replicas for each partition to address availability.

- However, we **need to scale also the computing part** and address availability problems related to it:
  - The proposal is to replicate also the Flink Jobmanager along with taskmanagers across multiple machines .

Event Processing and Fraud Detection with Flink

- **Speeding Violation**
  - We retrieve the speed limit for a given geographical location using OpenStreetMap's Overpass API.
  - Using the GeoTools library for accurate Earth surface distance calculation.

- **Inactivity Violation**

- **Frauds**
  - We check for consequently large number of transactions in the stream (no state required) and for multiple transaction in a short period of time (keyed state is required)

Events Processing Pipeline (Kafka + Flink)