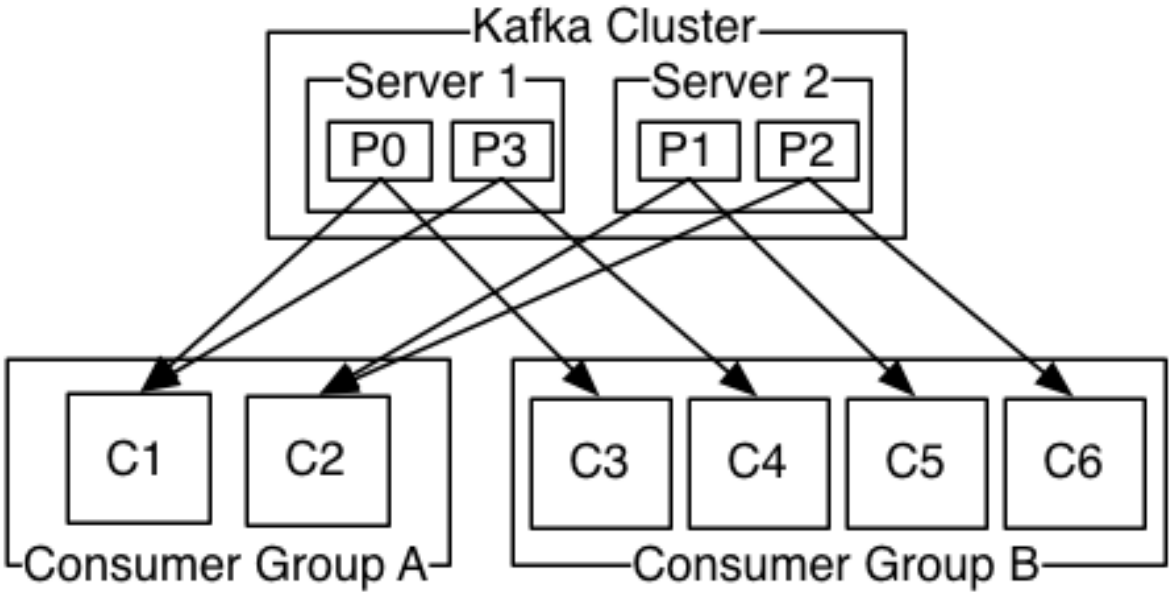
Optimizing Kafka and Flink Consumption

Sample Kafka Setup

3 kafka topics from 3 different kafka clusters, each topic has 10 partitions

Kafka Consumer groups
Each Kafka consumer belongs to a consumer group i.e. it can be thought of as a logical container/namespace for a bunch of consumers. A consumer group can receive messages from one or more topics. Instances in a consumer group can receive messages from zero, one or more partitions within each topic (depending on the number of partitions and consumer instances)

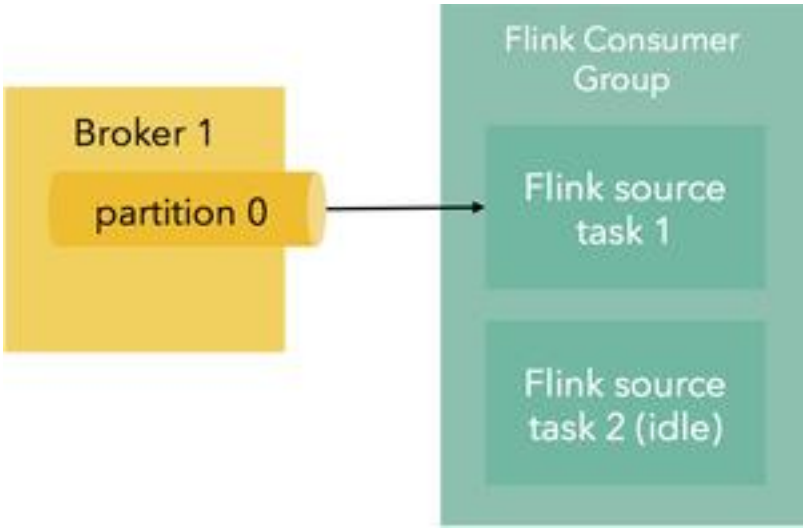


How are Kafka partitions assigned to Flink workers?
In Kafka, each consumer from the same consumer group gets assigned one or more partitions. Note that it is not possible for two consumers to consume from the same partition. The number of Flink consumers depends on the Flink parallelism, meaning that each Flink Task (We roughly consider each Flink Task = Flink slots = Flink Parallelism = Available CPU core) can act as a separate consumer in a consumer group. Also, you should notice that topics are just an abstraction for grouping partitions and data, internally only partitions are assigned to Flink's parallel task instances according to the following pattern.

There are three possible cases:
1. kafka partitions == flink parallelism
This case is ideal since each consumer takes care of one partition. If your messages are balanced between partitions, the work will be evenly spread across Flink operators

2. kafka partitions < flink parallelism

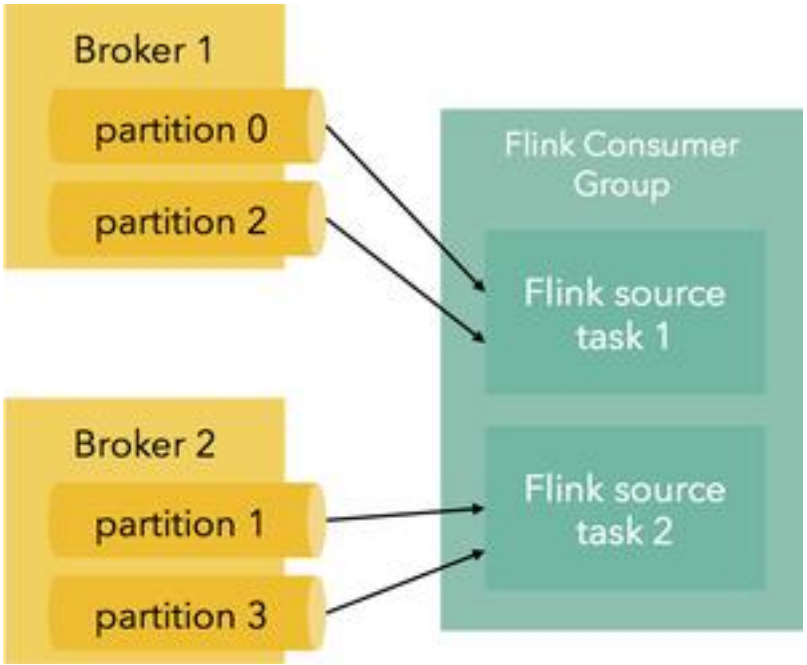
When there are more Flink tasks than Kafka partitions, some of the Flink consumers will just idle, not reading any data:



In this case, where you have higher parallelism than the number of partitions (because you want to make use of it in a future operator), you could do a `.rebalance()` after the Kafka source. This makes sure that all operators after the Kafka source get an even load, at the cost of having to redistribute the data (so there is de/serialization + network overhead).

3. kafka partitions > flink parallelism

When there are more Kafka partitions than Flink tasks, Flink consumer instances will subscribe to multiple partitions at the same time:



In all cases, Flink will optimally assign Tasks to the partitions.

In the example, you can create the Kafka Consumer group using Flink Kafka connector and assign one or more topics to it (using Regex, for example). So if Kafka has three topics including 10 partitions each, assigning 30 slots (core) to Flink Job Manager, you can achieve the ideal case, meaning each consumer(Flink slot) will consume one Kafka partition.