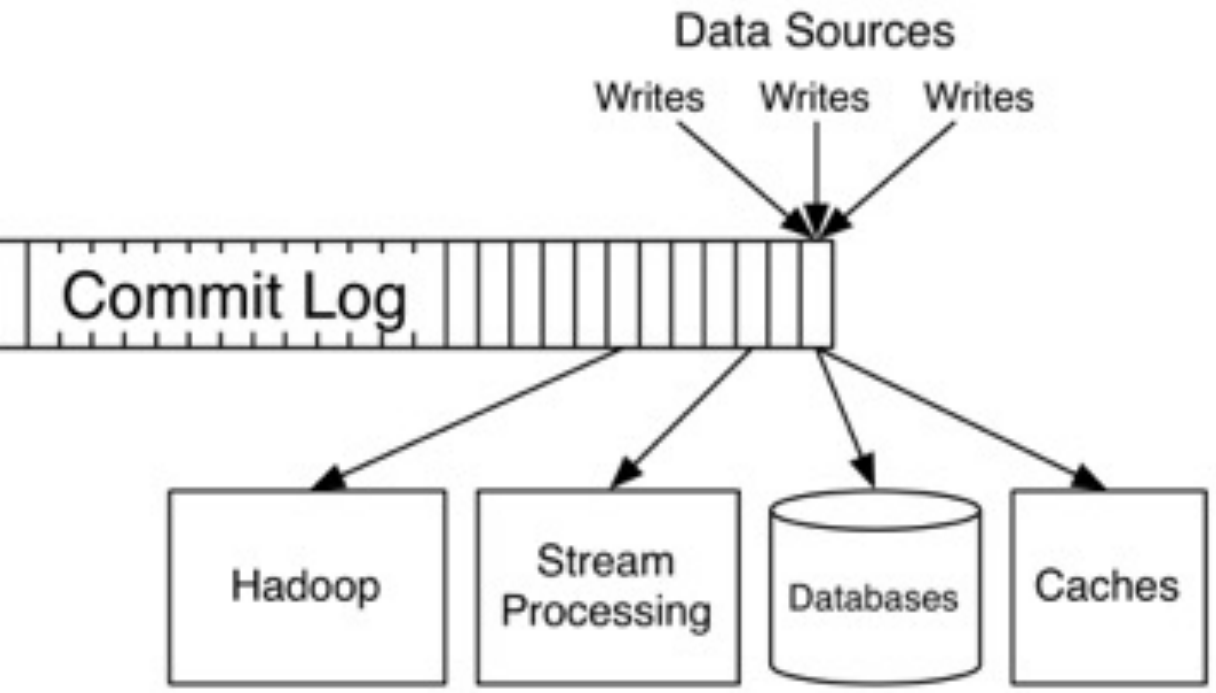
Why is My Stream Processing Job Slow?

Xavier Léauté, Software Engineer

Gwen Shapira, Principal Data Architect

Kafka 101



Distributed

Scalable

Fault-Tolerant

Partitioned + Replicated Log

Ordering guarantees

Consumers advance independently

Exactly-once delivery

Transactional commits


What people think of Stream Monitoring

What our typical experience is

Real Customer Experiences

Real Customer Experiences

Client Side

Broken Streaming Job / App

Real Customer Experiences

Client Side	**Broken Streaming Job / App**
End-to-End	**Slow Replication**

Your Kafka stream job stopped humming… now what?

What we check

- [x] Consumer Lag
- [ ] Partition Assignment
- [ ] Partition Skew
- [ ] Client Logs
- [ ] GC Log
- [ ] Metrics
  - [ ] Request Latencies
  - [ ] Commit Rates
  - [ ] Group Rebalancing
- [ ] Basic Tuning
  - [ ] Batch Sizes
  - [ ] Commit Rate
- [ ] Application Profiling

The Newbie - During an incident...

GC Logs? Metrics?
How do I get those?

I'll just change some configs
and reboot everything.

Consumer Lag

Bad Capacity Allocation

```
kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group
fast-data-reader

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID
fast-data 1          8661694         8703404         41710      myapp-1
fast-data 3          8577975         8616490         38515      myapp-2
fast-data 0          4902354         8741872         3839518    myapp-3
fast-data 2          4922614         8621757         3699143    myapp-3
```

Bad Capacity Allocation

```
kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group
fast-data-reader

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID
fast-data  1          8661694         8703404         41710      myapp-1
fast-data  3          8577975         8616490         38515      myapp-2
fast-data  0          4902354         8741872         3839518    myapp-3
fast-data  2          4922614         8621757         3699143    myapp-3
```

Bad Capacity Allocation

```
kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group
fast-data-reader

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID
fast-data 1           8661694         8703404         41710      myapp-1
fast-data 3           8577975         8616490         38515      myapp-2
fast-data 0           4902354         8741872         3839518    myapp-3
fast-data 2           4922614         8621757         3699143    myapp-3
```

Watch for Partition Skew

```
kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group
fast-data-reader

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID
fast-data 1          8661694         8703404         41710      myapp-1
fast-data 3          8577975         8616490         38515      myapp-2
fast-data 0          4902354         8741872         3839518    myapp-3
fast-data 2          4922614         8621757         3699143    myapp-3
```

Watch for Partition Skew

```
kafka-consumer-groups --bootstrap-server localhost:9092 --describe --group
fast-data-reader

TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID
fast-data  1          8661694         8703404         41710      myapp-1
fast-data  3          8577975         8616490         38515      myapp-2
fast-data  0          4902354         8741872         3839518    myapp-3
fast-data  2          4922614         8621757         3699143    myapp-3
```

Not all partitions are created equal

Important for

- Keyed topics
- Custom partitioned topics

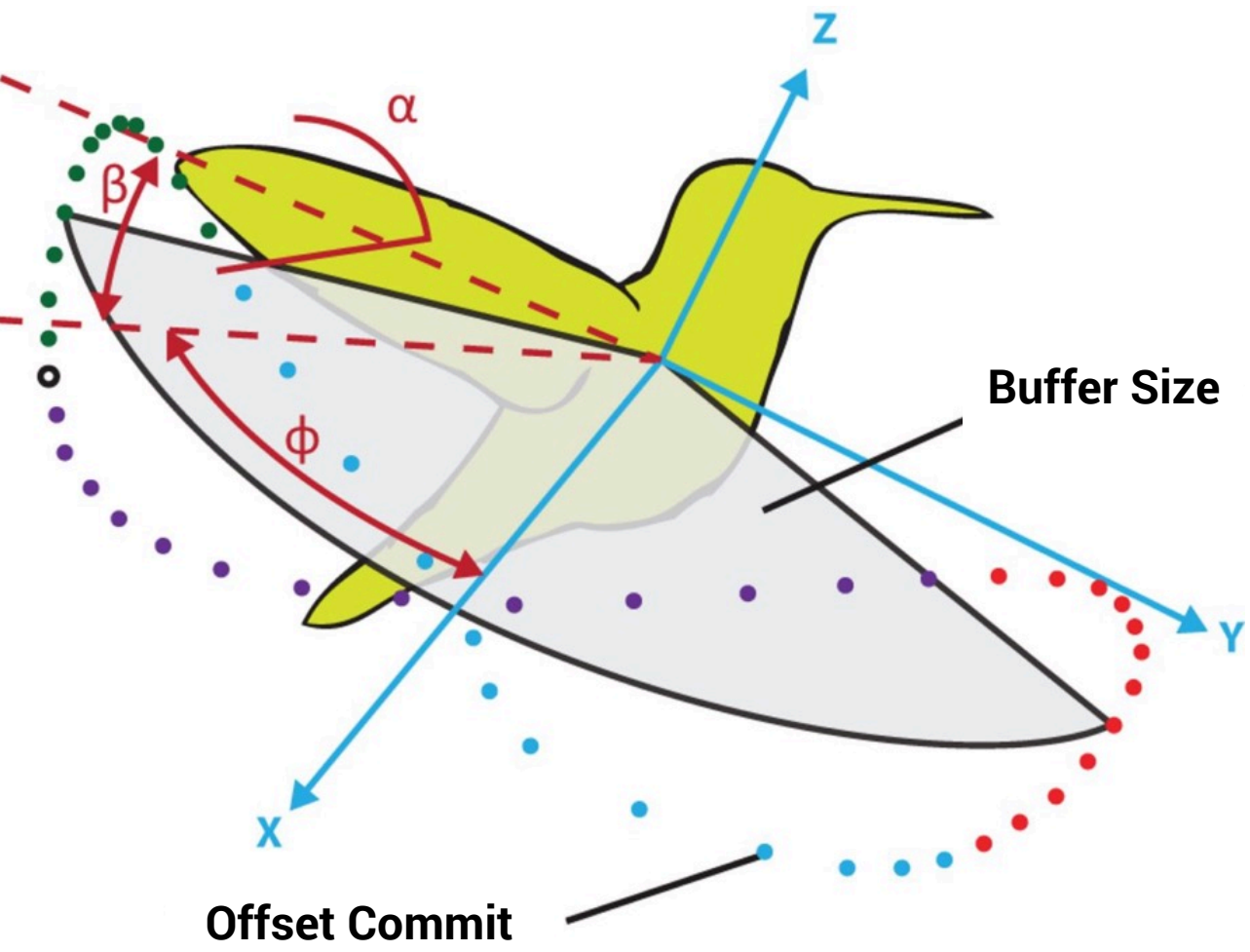
Early warning signs

- some partitions lagging
- uneven CPU / Network usage

Typical cause

- skewed key distribution in your data
- bad joins (null keys)
- imbalance across brokers

Clients have metrics too!



Start with the basics **GC / CPU / Network**

General Slowness

- Consumer or Producer Side?
- Global Request Latencies

Some partitions still lagging

- Per Broker metrics (bad node / network)
- Per Topic metrics (data / tuning)

Turn up the log level

Time for some profiling

https://github.com/jvm-profiling-tools/async-profiler
https://github.com/brendangregg/FlameGraph

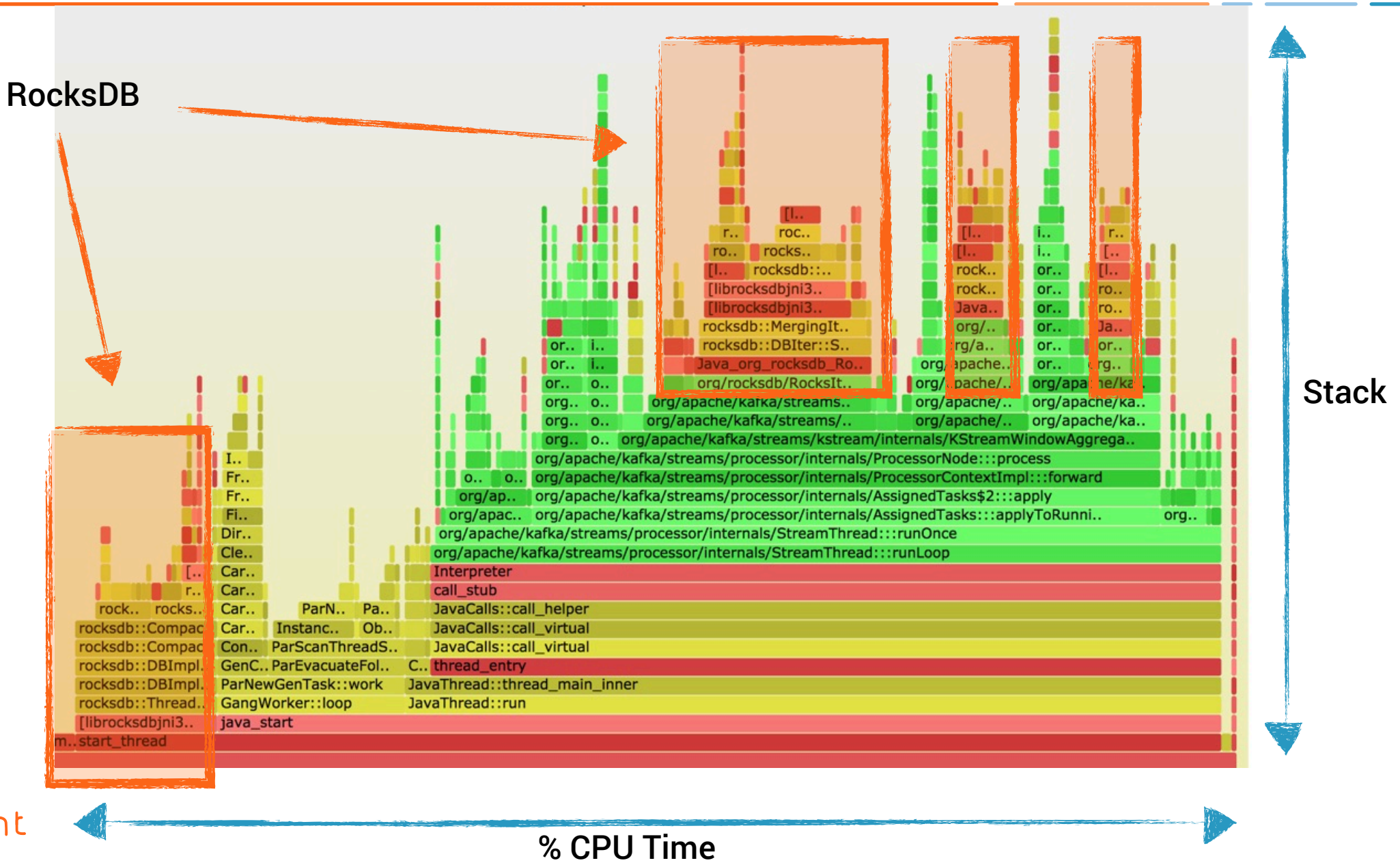
```
./profiler.sh -d 30 -f flamegraph.svg <pid>
```

To impress your coworkers
https://github.com/Netflix/flamescope

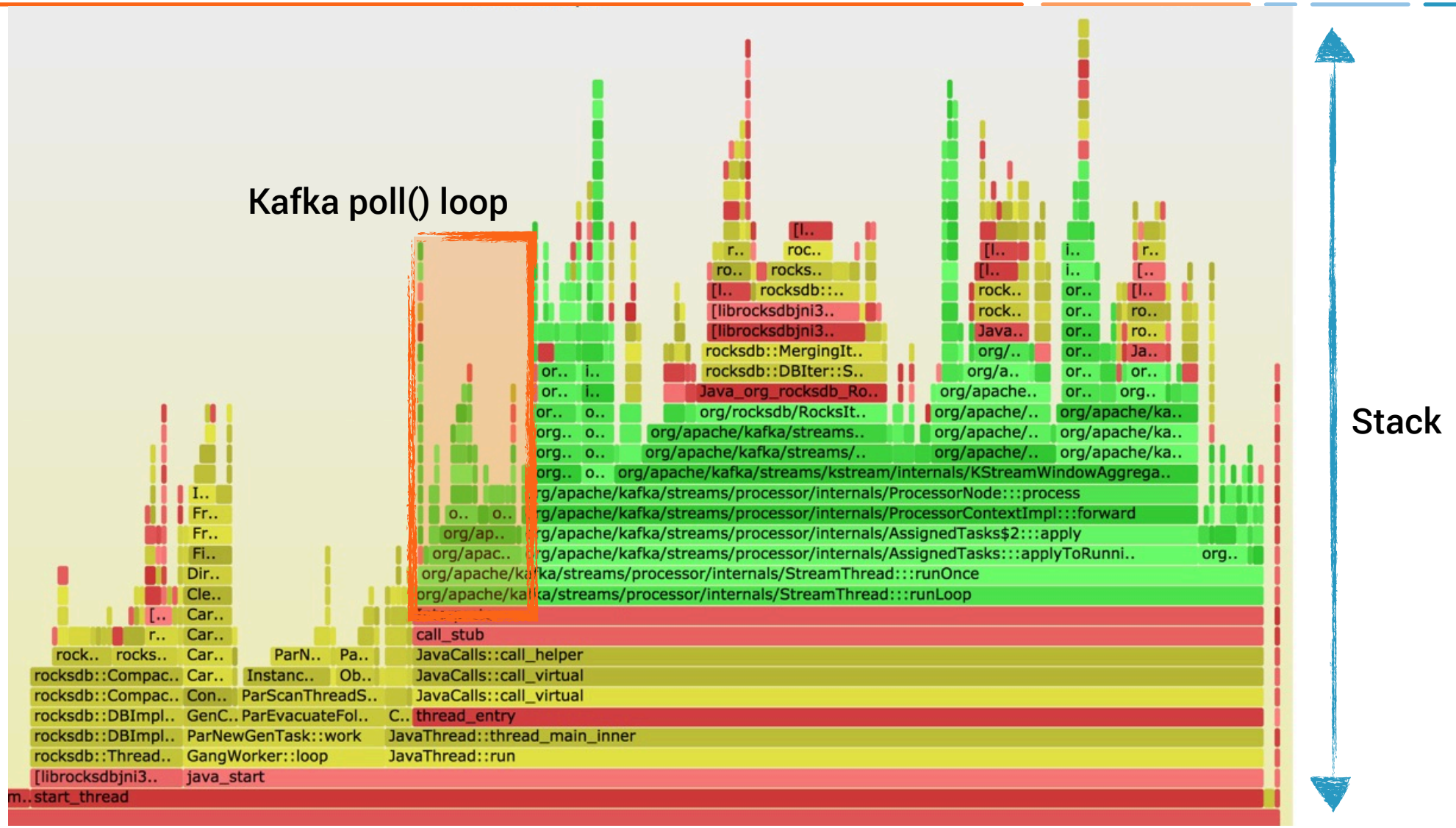
Here's where your CPU cycles went

Here's where your CPU cycles went

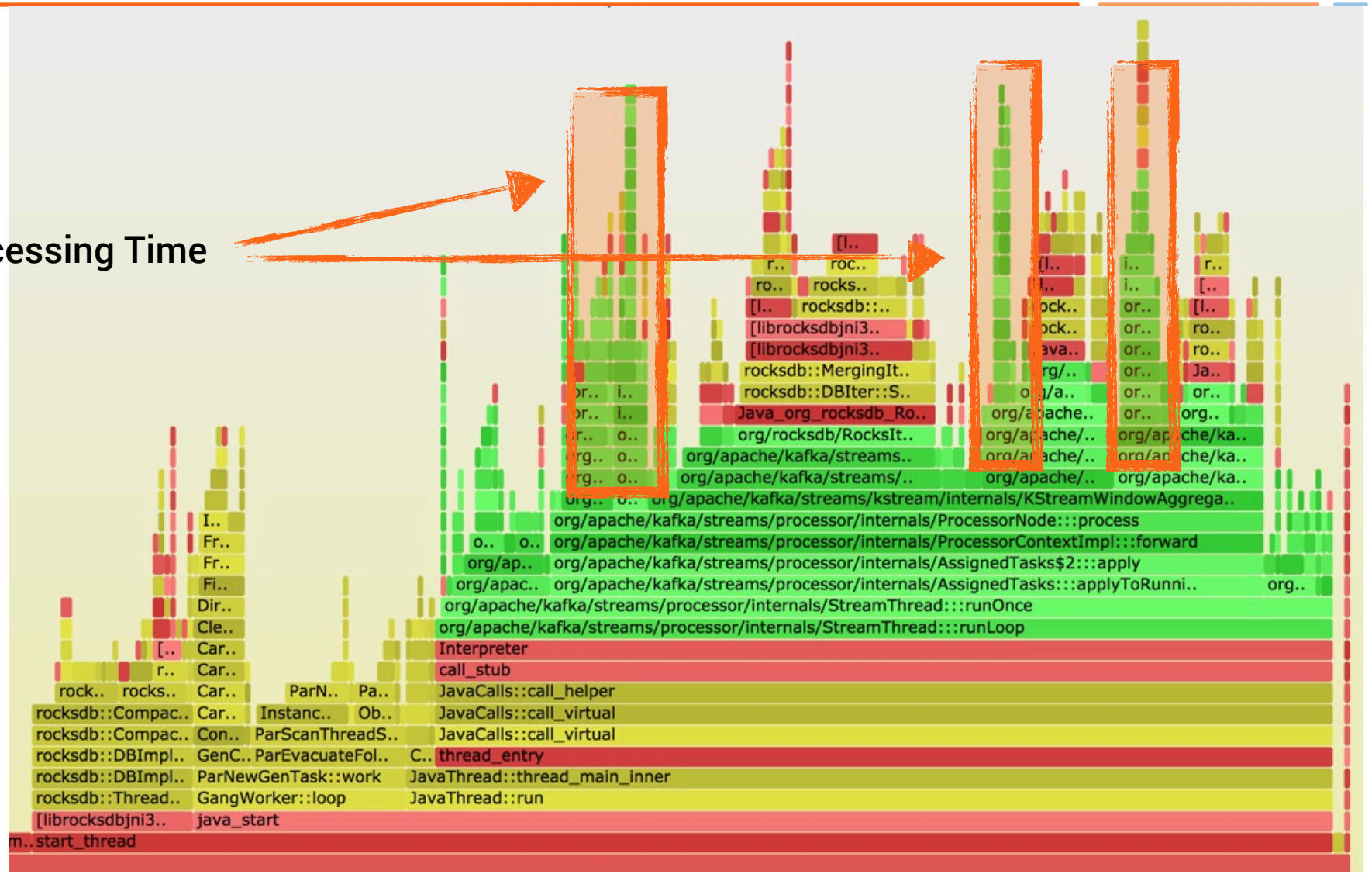
Here's where your CPU cycles went



Here's where your CPU cycles went



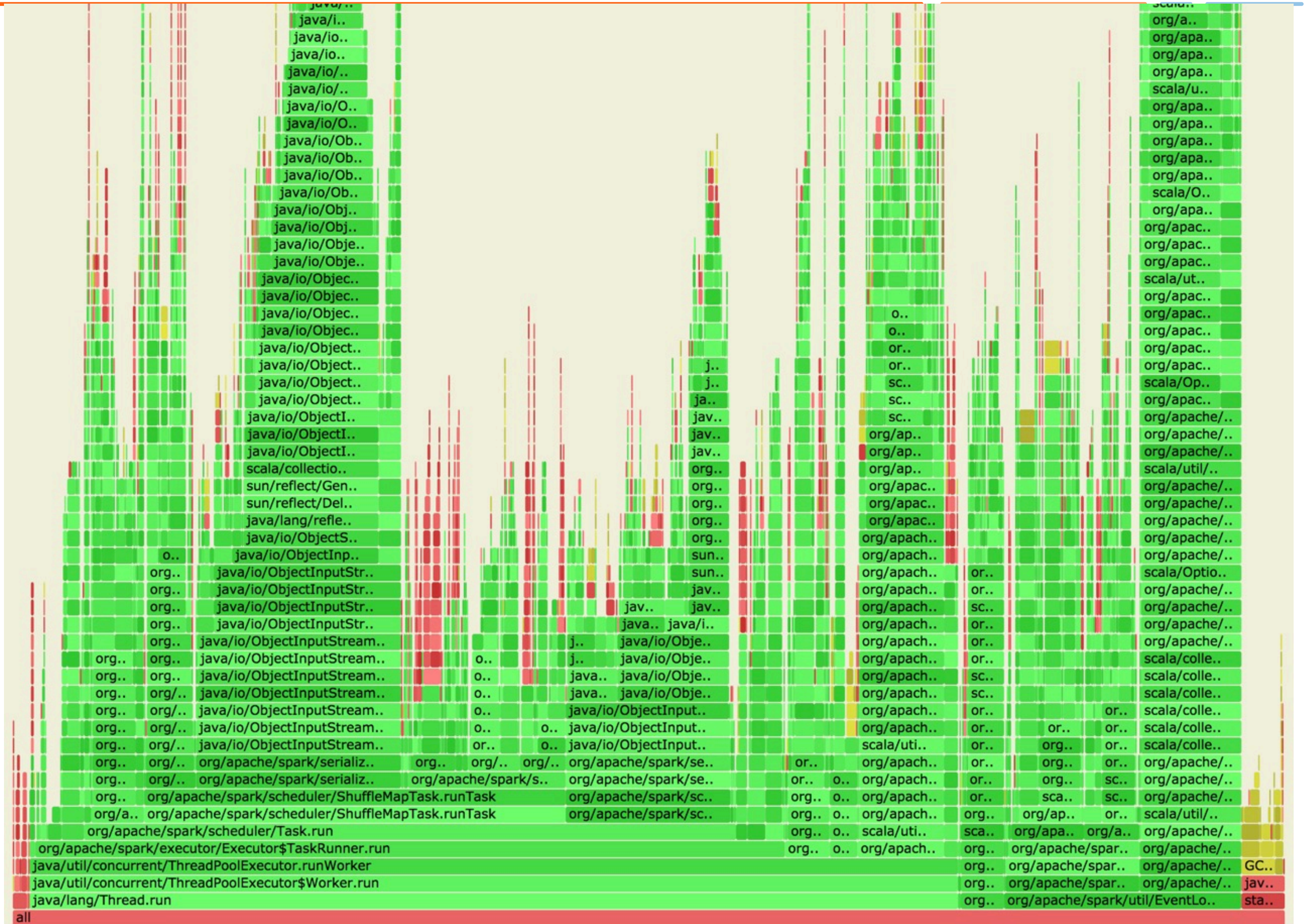
Here's where your CPU cycles went



Spark Streaming Clickstream Example (using Kafka)

```
val pageViews = KafkaUtils.createDirectStream[String, String](
  streamingContext, LocationStrategies.PreferConsistent,
  ConsumerStrategies.Subscribe[String, String](topicsSet, kafkaParams))
  .map { record => PageView.fromString(record.value) }
// Return the rate of error pages (a non 200 status) in each zip code over the last 30 seconds
val statusesPerZipCode = pageViews.window(Seconds(30), Seconds(2))
                                  .map(view => (view.zipCode, view.status))
                                  .groupByKey()
val errorRatePerZipCode = statusesPerZipCode.map{
  case(zip, statuses) =>
    val normalCount = statuses.count(_ == 200)
    val errorCount = statuses.size - normalCount
    val errorRatio = errorCount.toFloat / statuses.size
    if (errorRatio > 0.05) {
      "%s: **%s**".format(zip, errorRatio)
    } else {
      "%s: %s".format(zip, errorRatio)
    }
}
errorRatePerZipCode.print()
```

Spark Streaming Clickstream Example (using Kafka)



Spark Streaming Clickstream Example (using Kafka)

Scheduler
Event Loop

Spark Streaming Clickstream Example (using Kafka)

Shuffle Writes

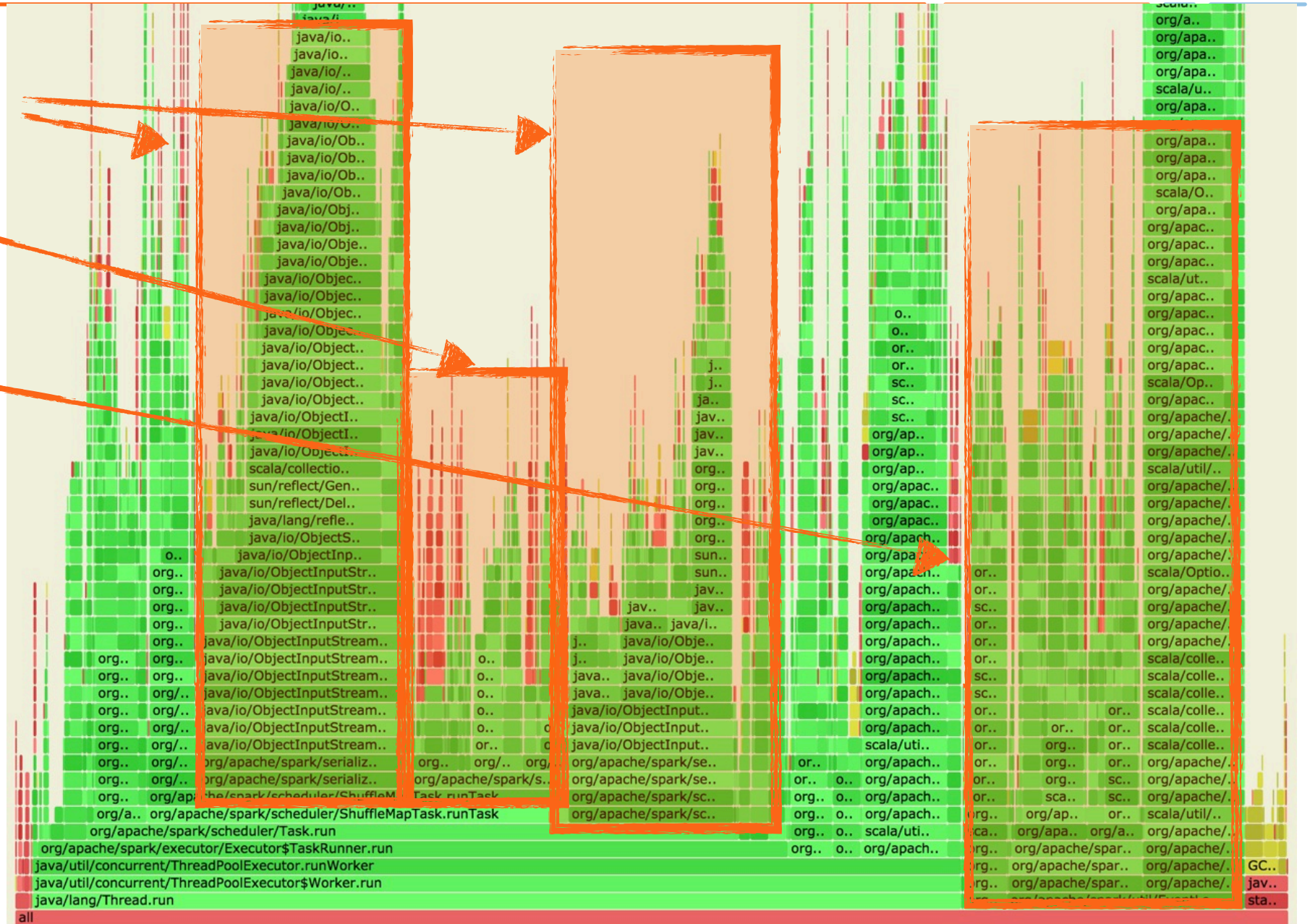
Scheduler Event Loop

Spark Streaming Clickstream Example (using Kafka)

30% deserialization

Shuffle Writes

Scheduler
Event Loop



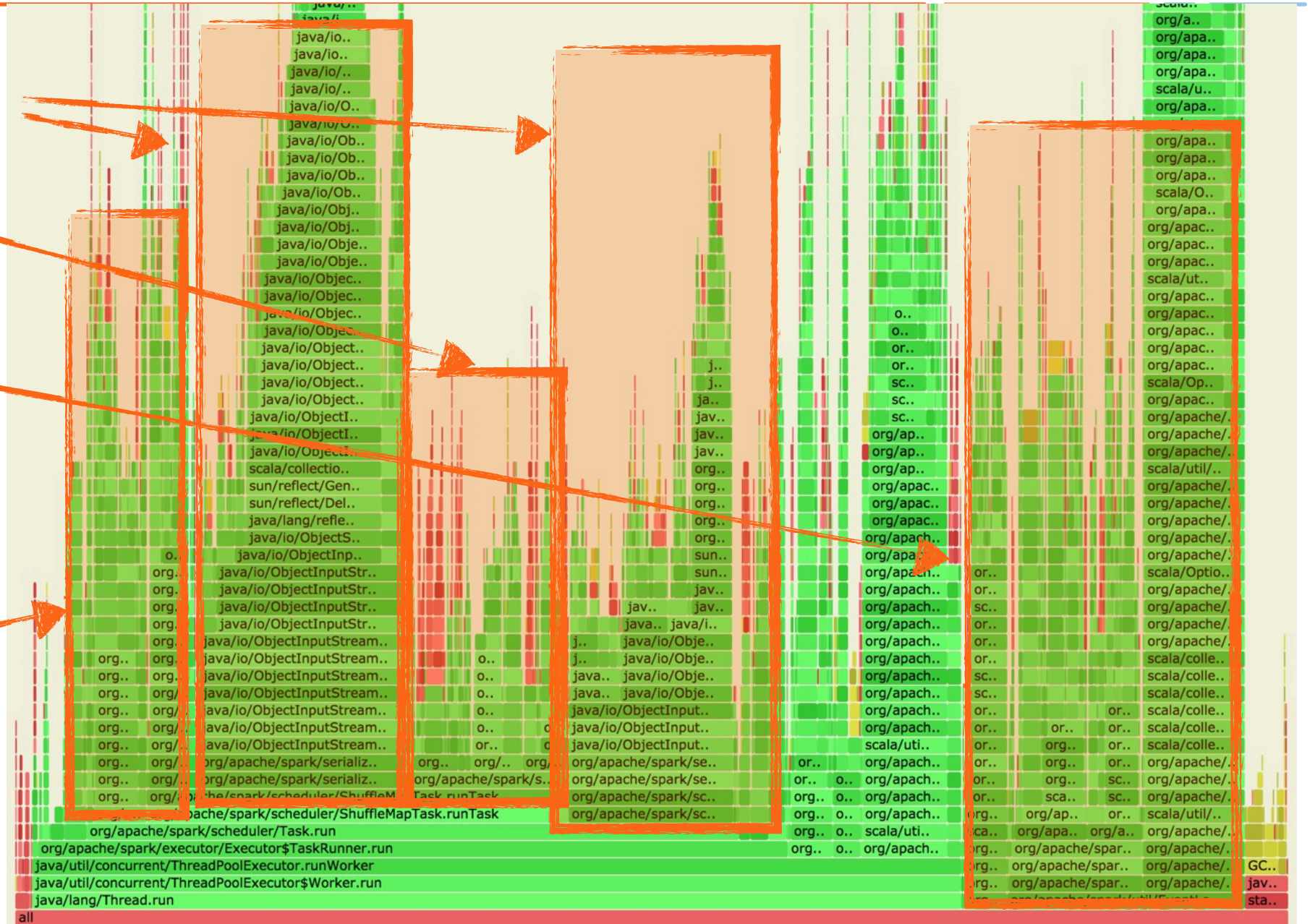
Spark Streaming Clickstream Example (using Kafka)

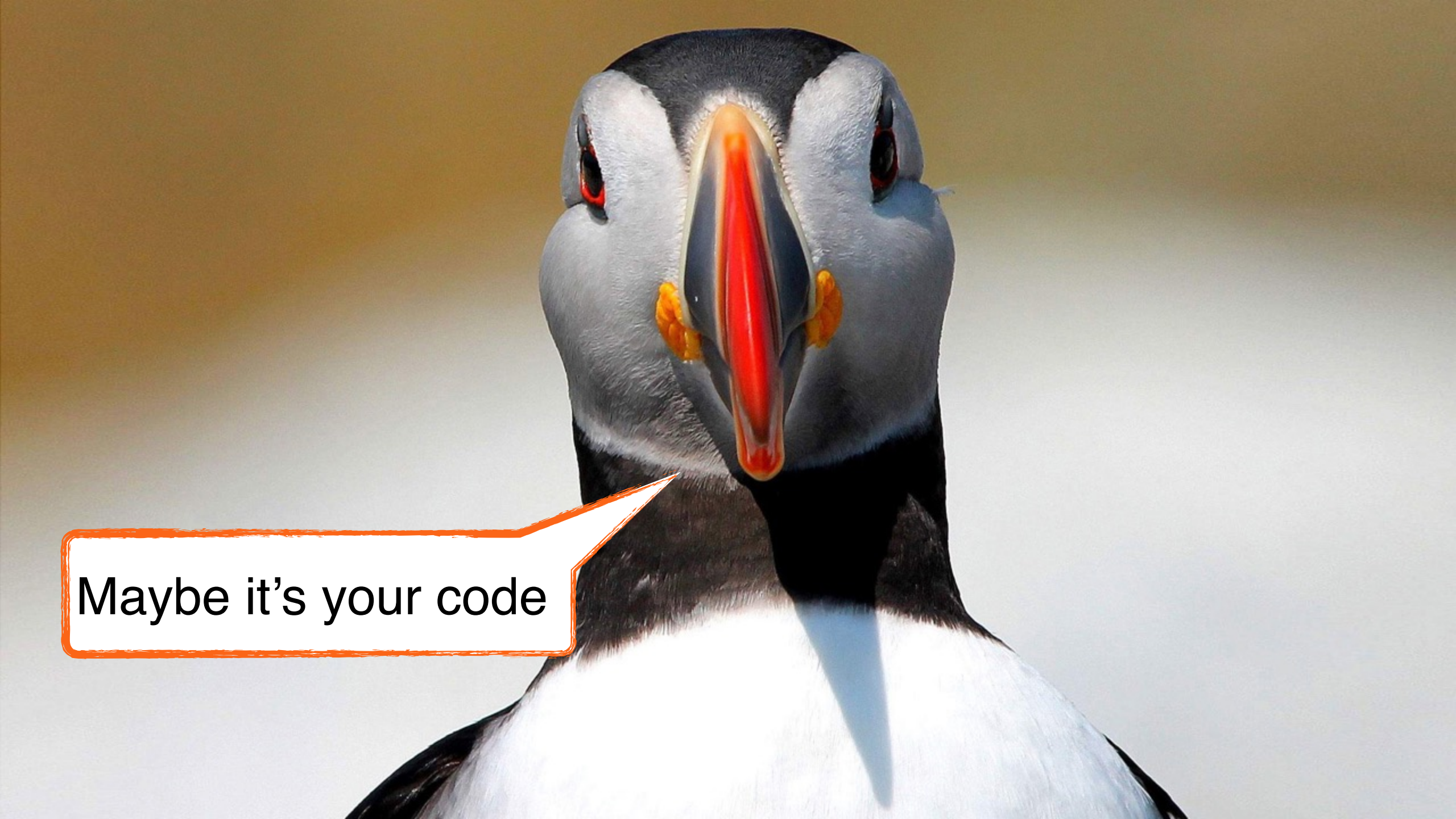
30% deserialization

Shuffle Writes

Scheduler
Event Loop

Read from Kafka
& Processing




Maybe it’s your code

Let's commit, just to be safe, right?

Common beginner mistake

Commit only as needed
- keep recovery short
- maximize throughput

Metrics to validate
- commit-rate
- commit-latency-avg

Right-size your batches

Right-size your batches

Bigger Batches

- increase throughput
- improve compression

Right-size your batches

Bigger Batches

- increase throughput
- improve compression

Small enough (<< 10MB) to keep GC low

Right-size your batches

Bigger Batches

- increase throughput
- improve compression

Small enough (<< 10MB) to keep GC low

batch.size + **linger.ms**

Right-size your batches

Bigger Batches

- increase throughput
- improve compression

Small enough (<< 10MB) to keep GC low

batch.size + **linger.ms**

don't forget!

Right-size your batches

Bigger Batches

- increase throughput
- improve compression

Small enough (<< 10MB) to keep GC low

batch.size + **linger.ms**

don't forget!

Watch

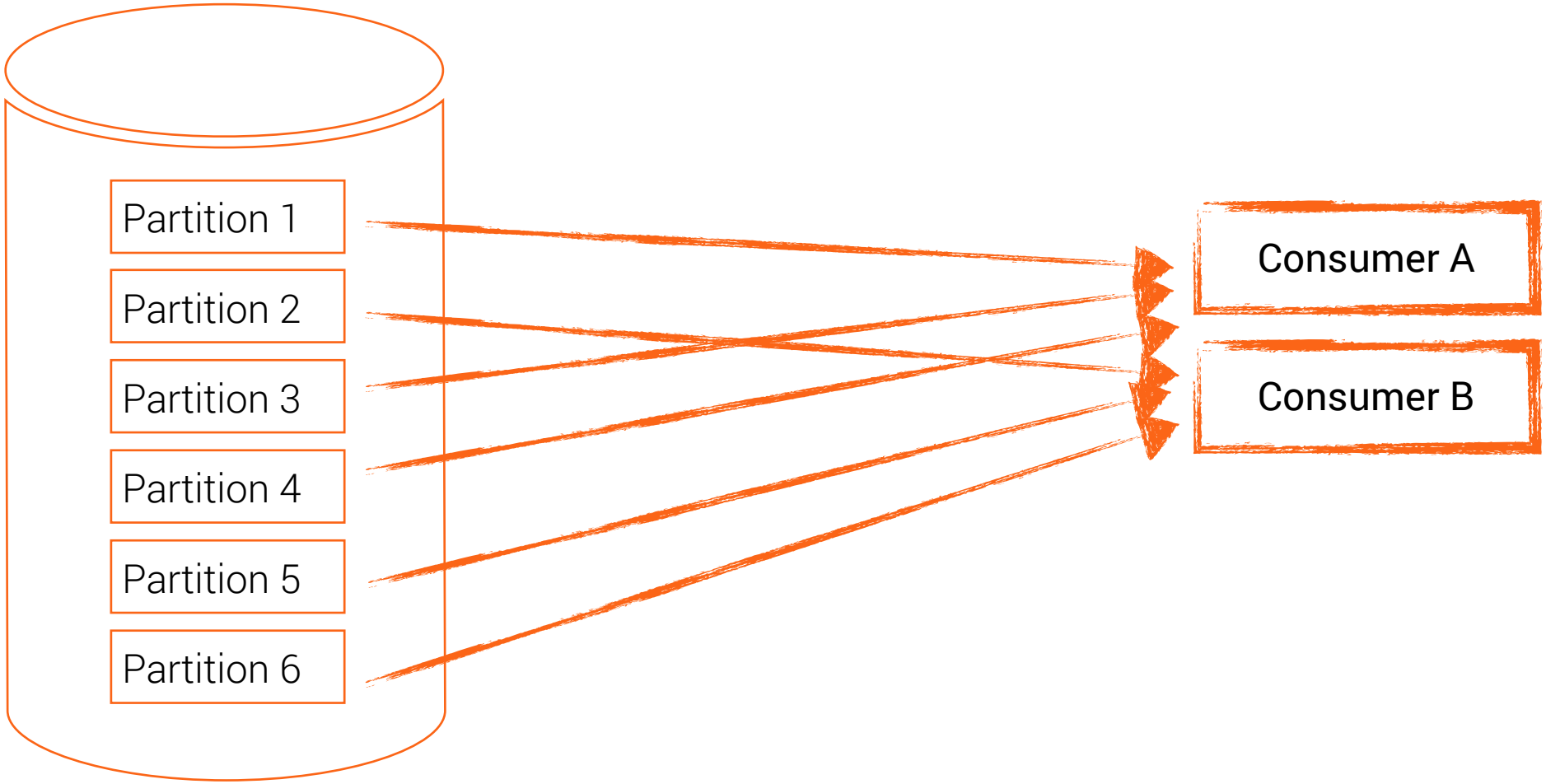
- request-rate
- request-latency-avg
- compression-rate

My app keeps rebalancing

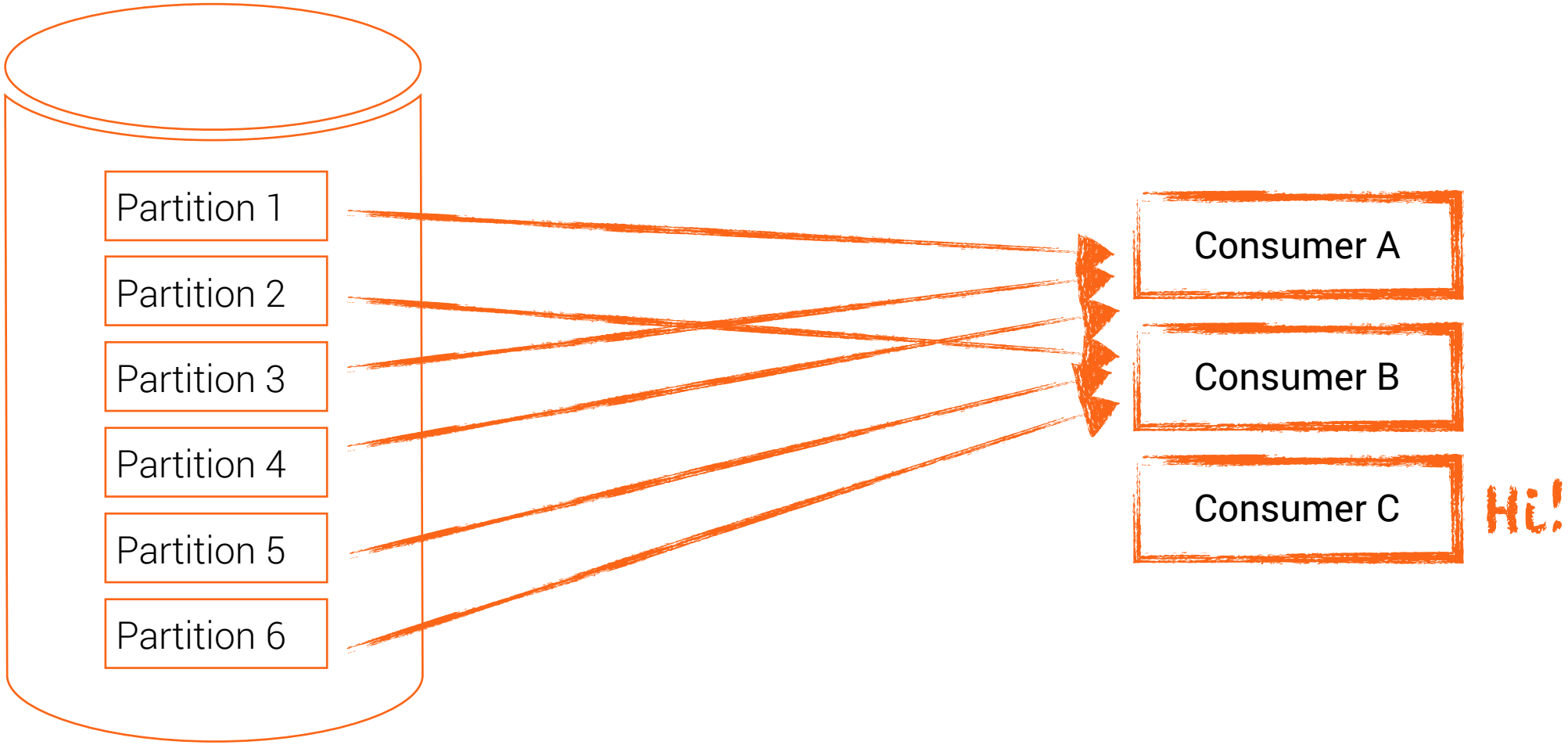
Symptoms

- low throughput
- high network chatter
- consumer logs galore
- no progress
- hanging

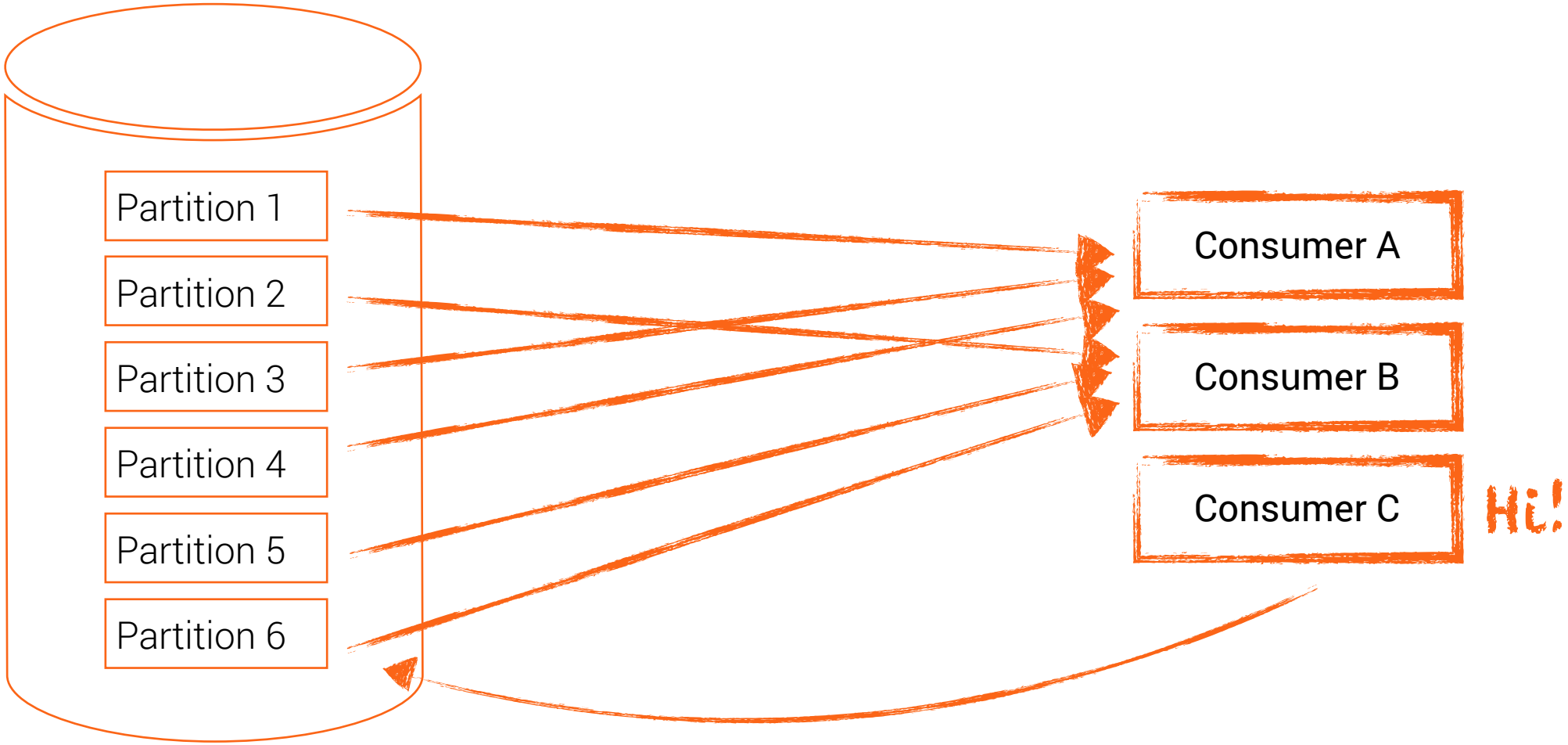
Kafka Consumer Group Rebalancing 101



Kafka Consumer Group Rebalancing 101

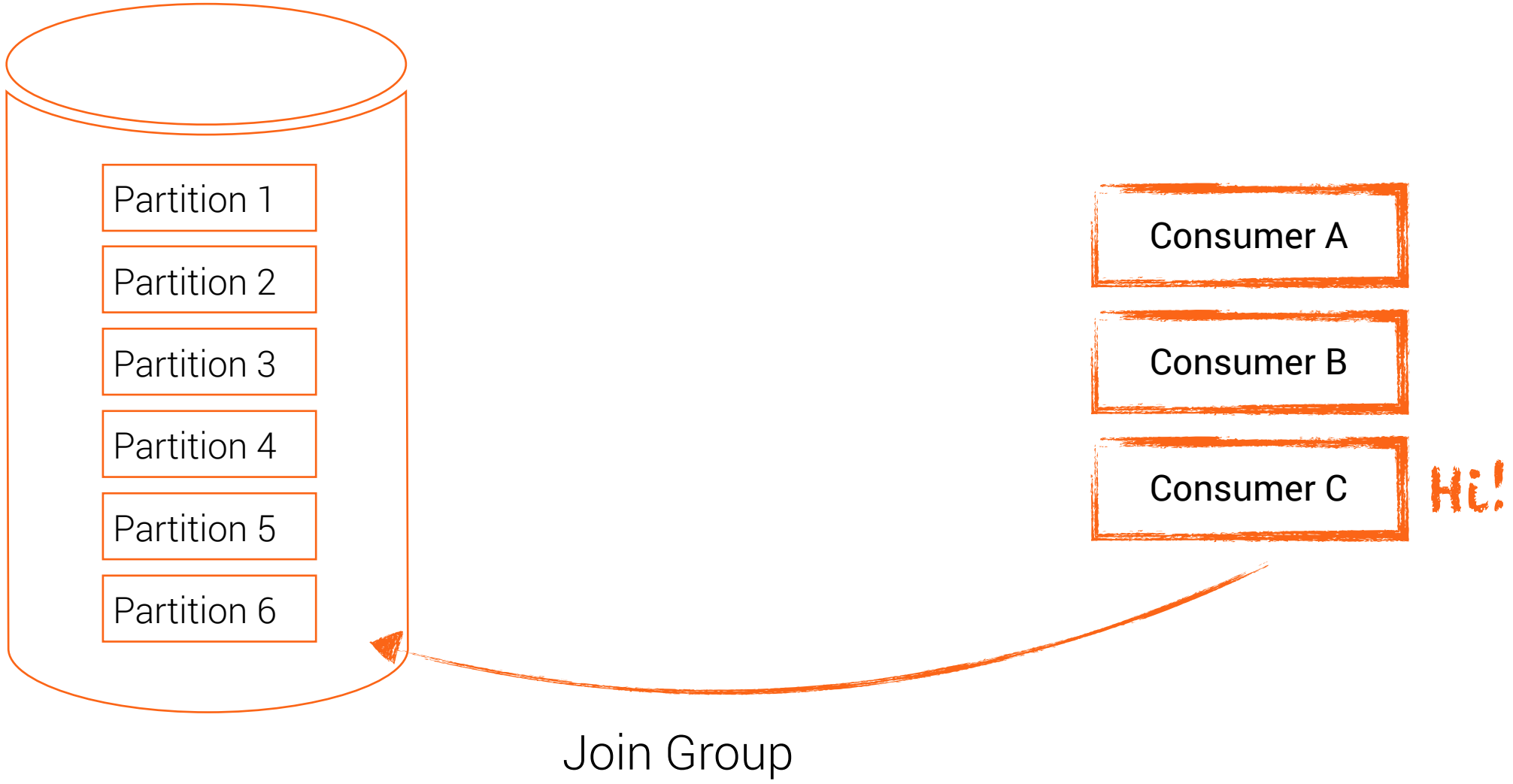


Kafka Consumer Group Rebalancing 101

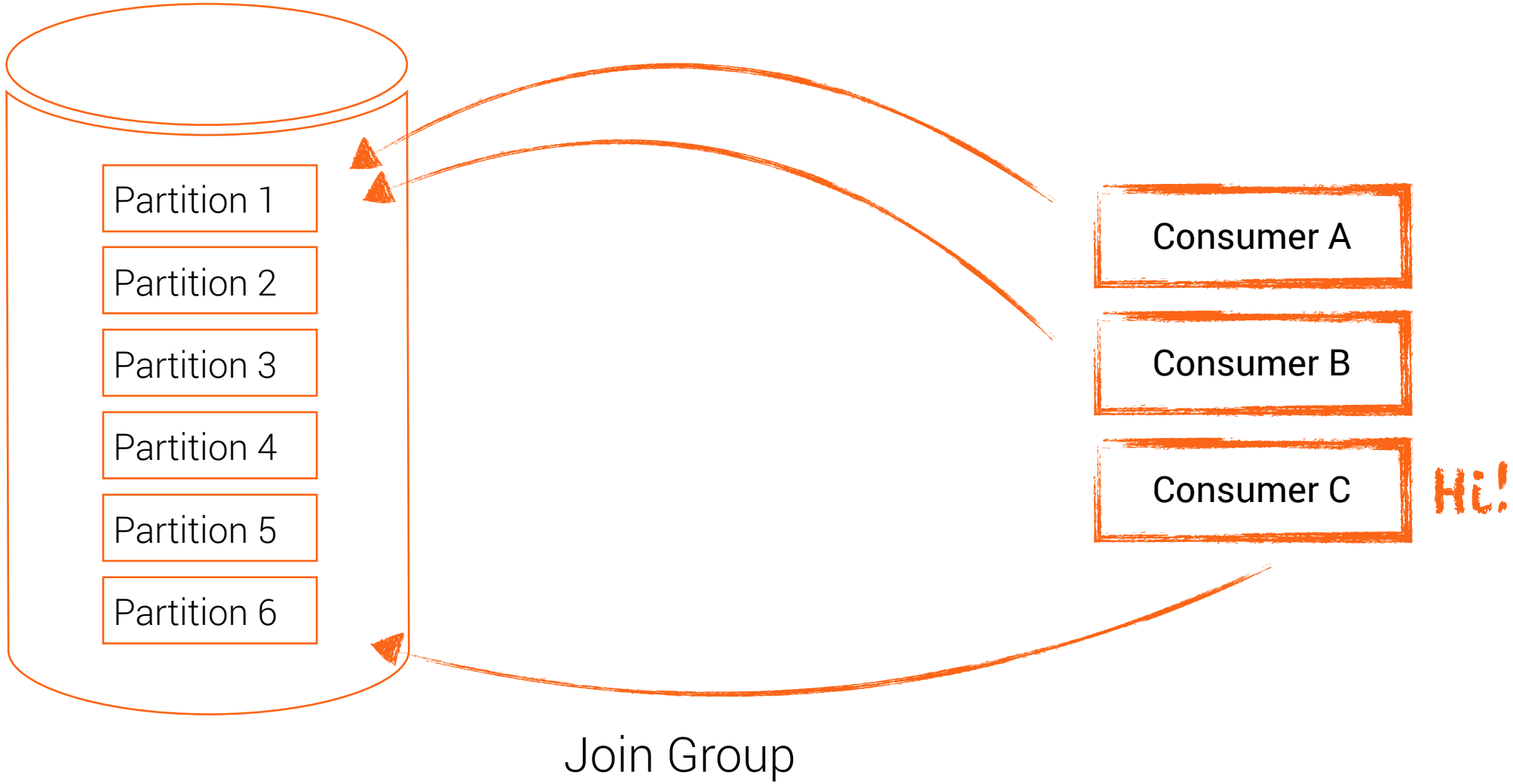


Join Group

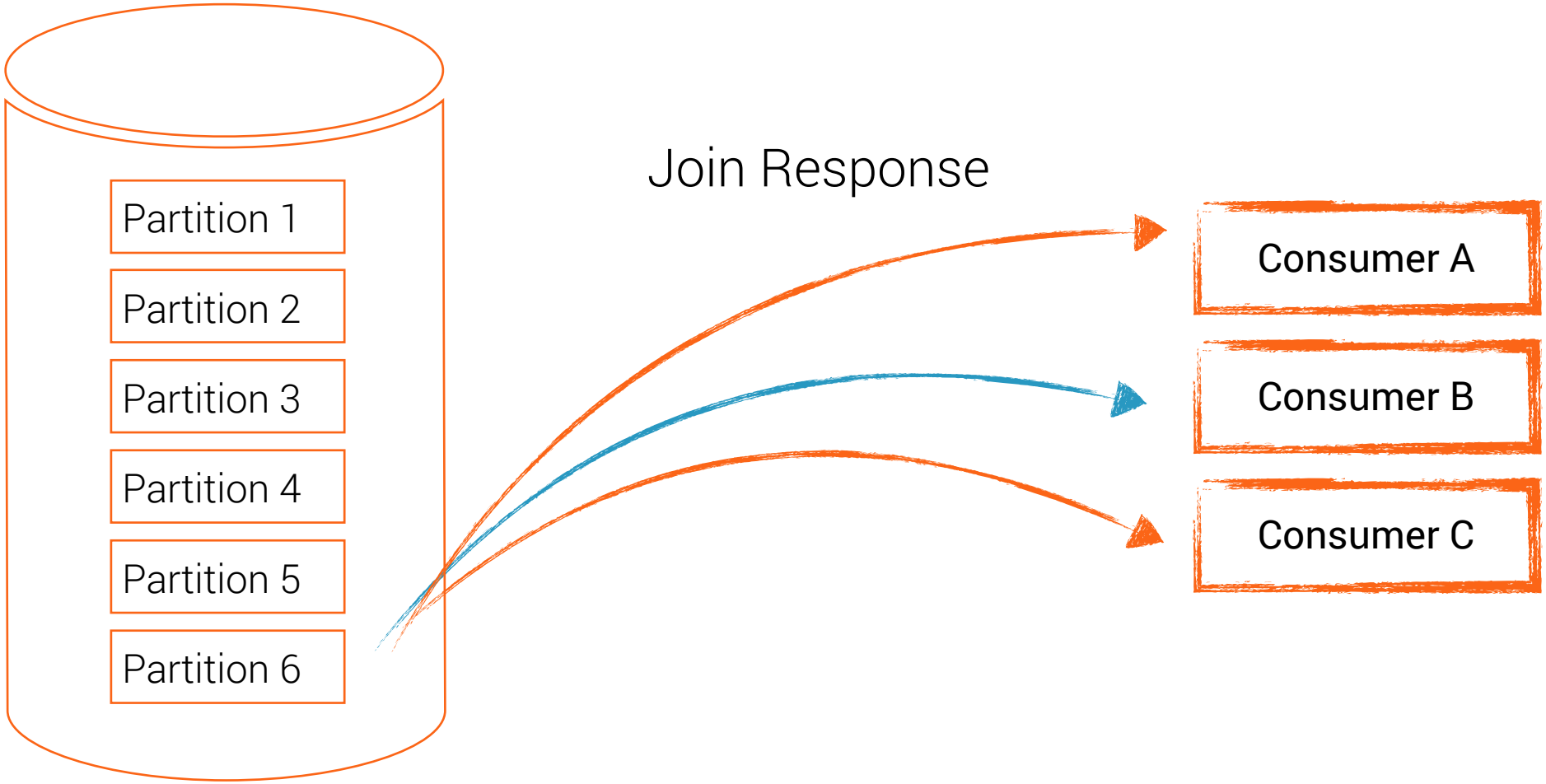
Kafka Consumer Group Rebalancing 101



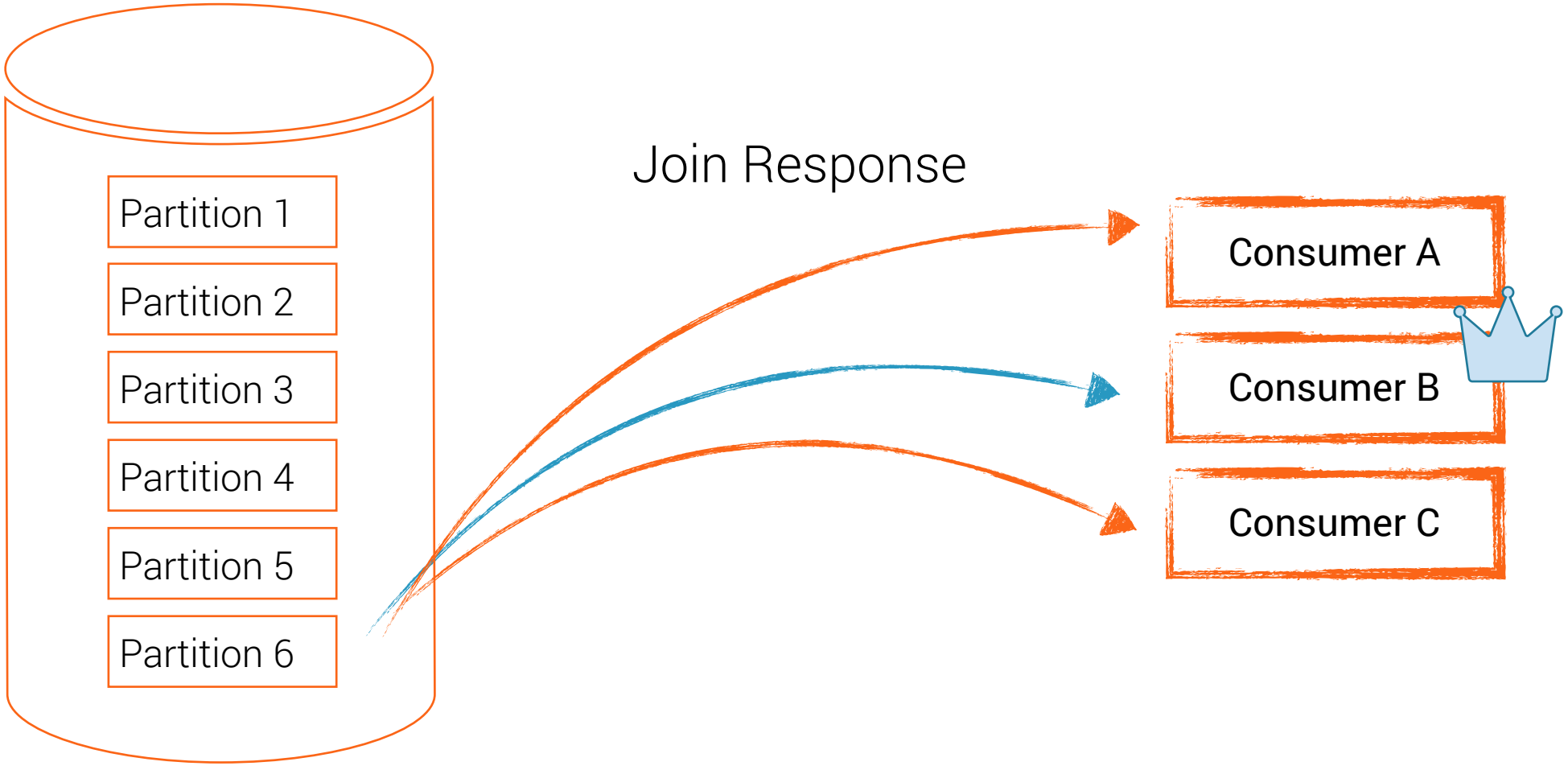
Kafka Consumer Group Rebalancing 101



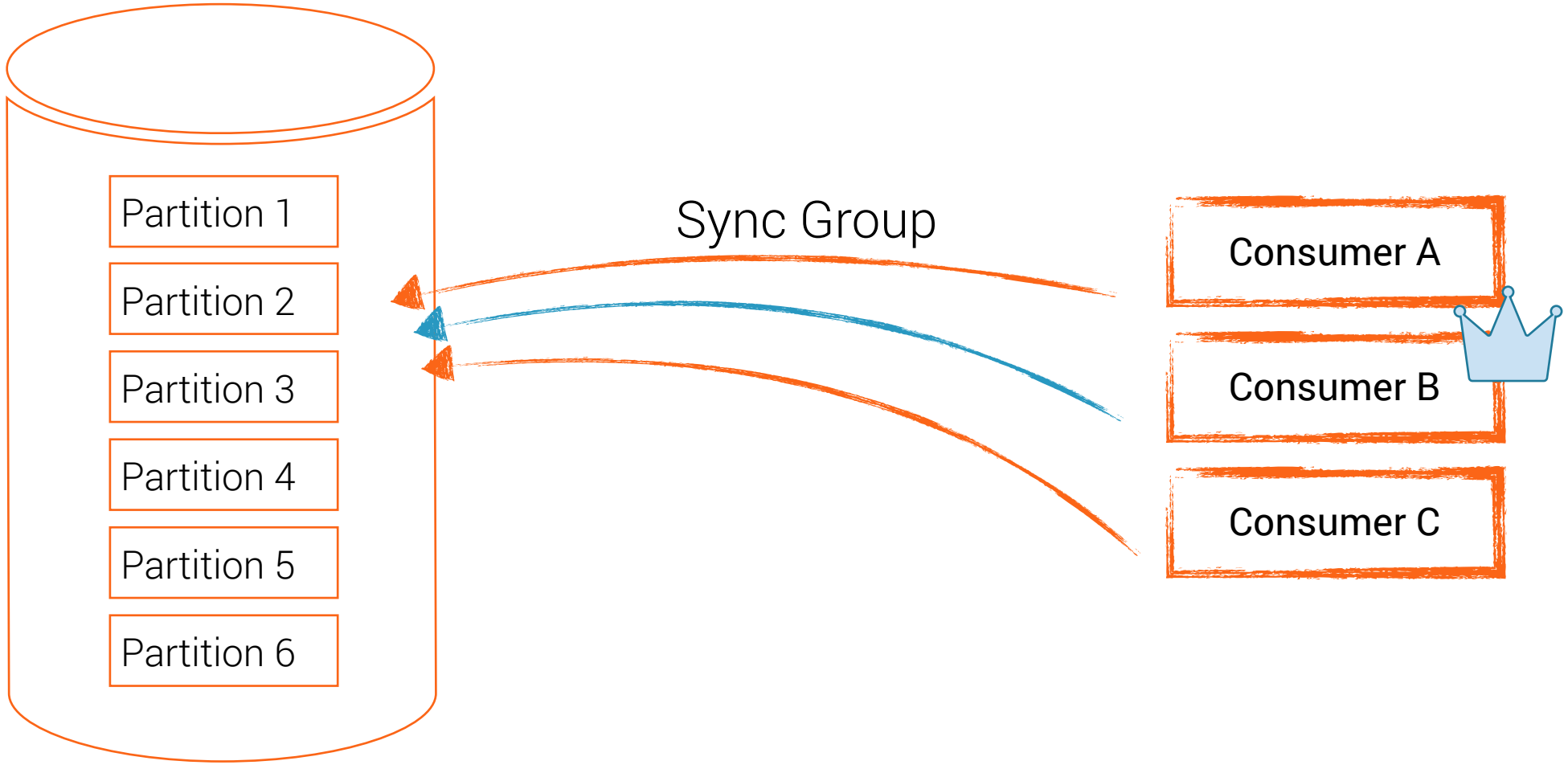
Kafka Consumer Group Rebalancing 101



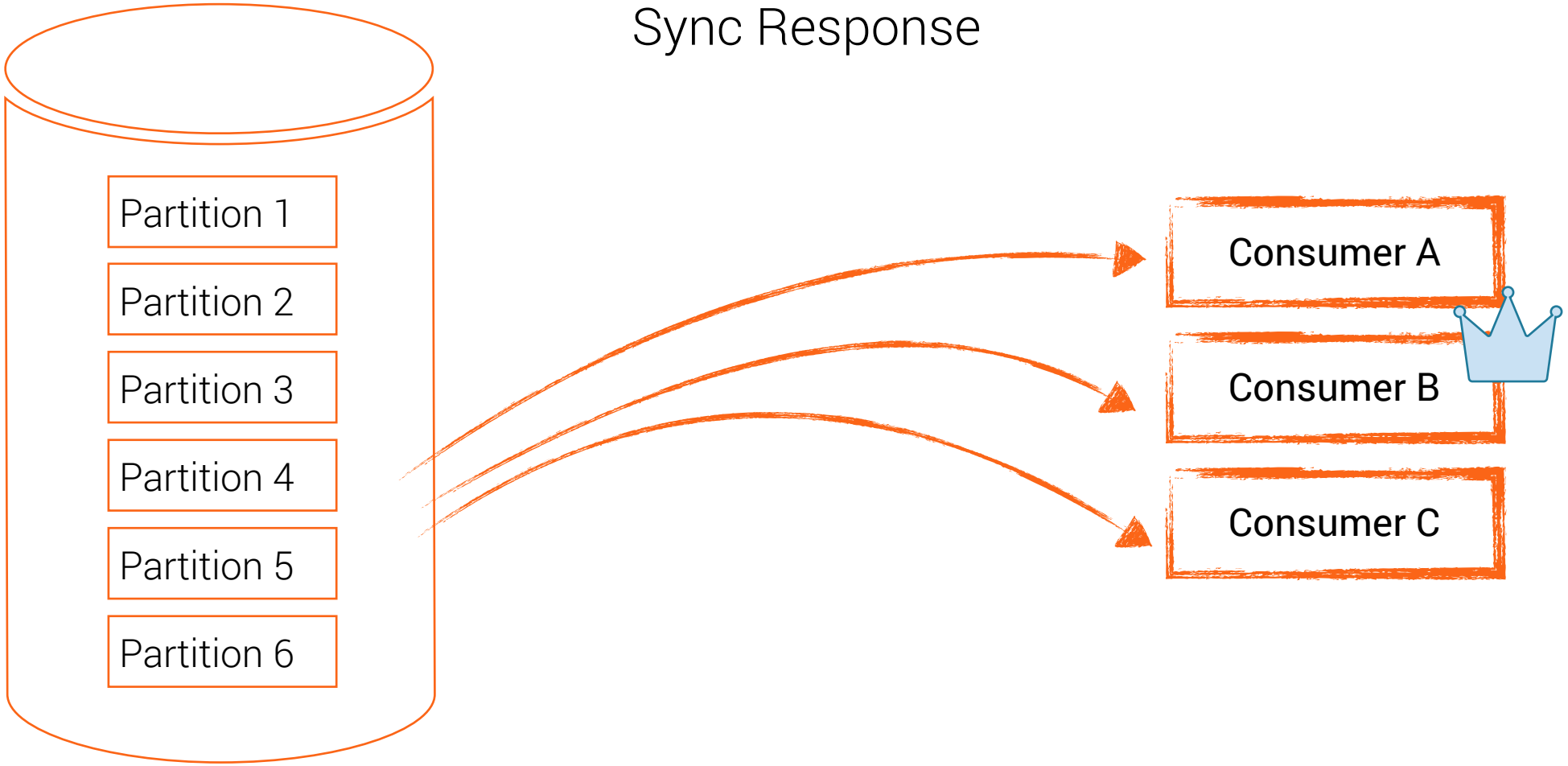
Kafka Consumer Group Rebalancing 101



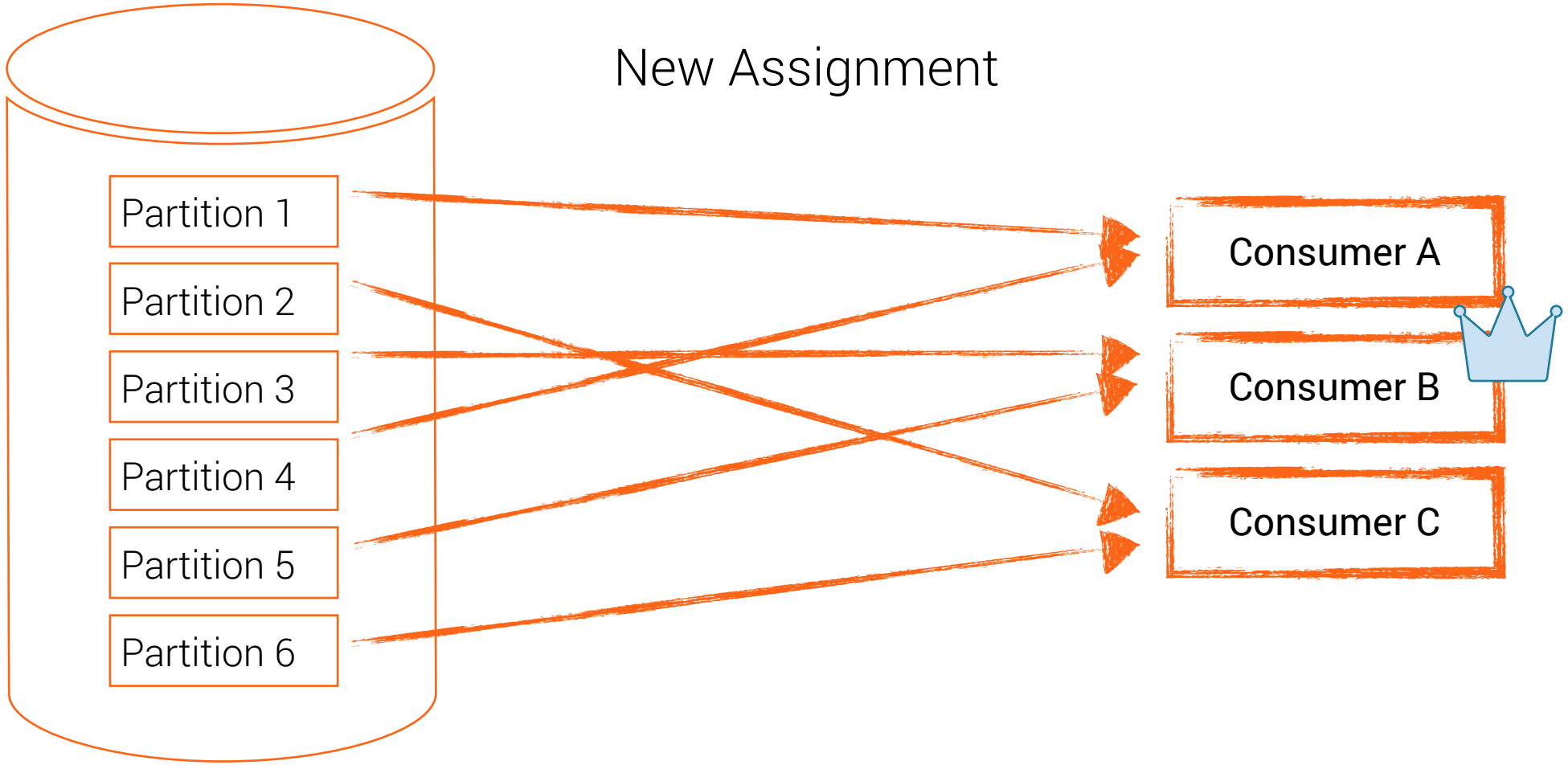
Kafka Consumer Group Rebalancing 101



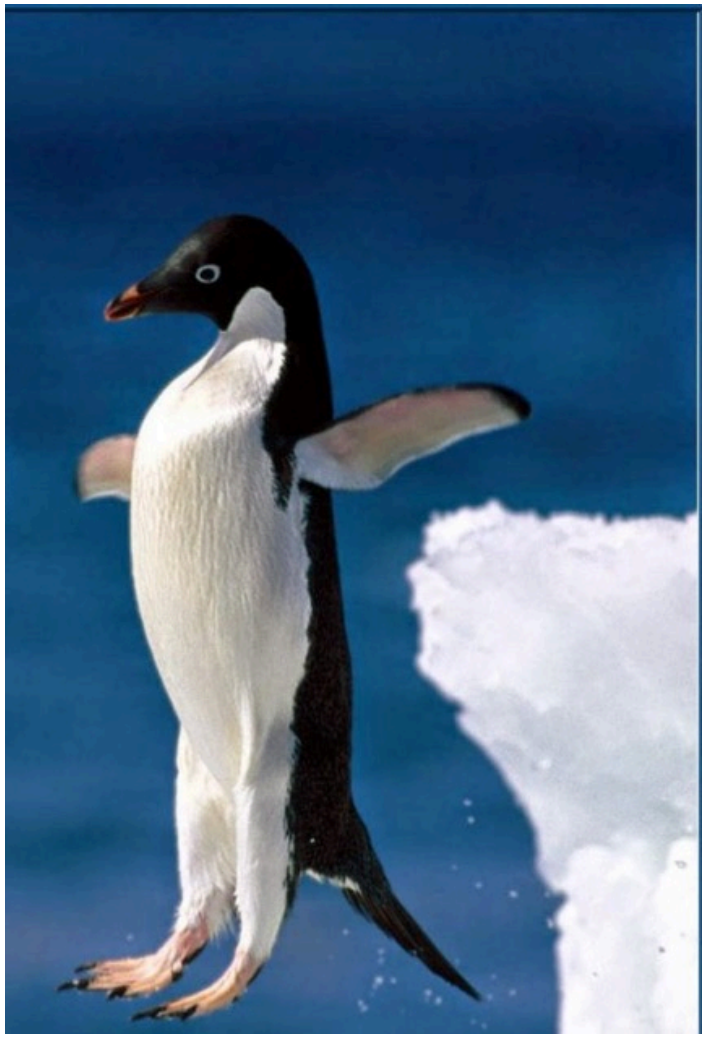
Kafka Consumer Group Rebalancing 101



Kafka Consumer Group Rebalancing 101



Restoring a Happy Balance



Timing Issues

- long GC pauses (tens of seconds)
- infrequent calls to `poll()`
- timeouts too short?

- flaky network
- 1 bad machine affects the entire group

Watch

- join-rate
- sync-rate

Competent Users

- Monitor Consumer Lag
- Lookout for Partition Skew
- Commit Offsets Sparingly
- Collect Logs
- Understand how to tune Batch Sizes

Kafka Pros

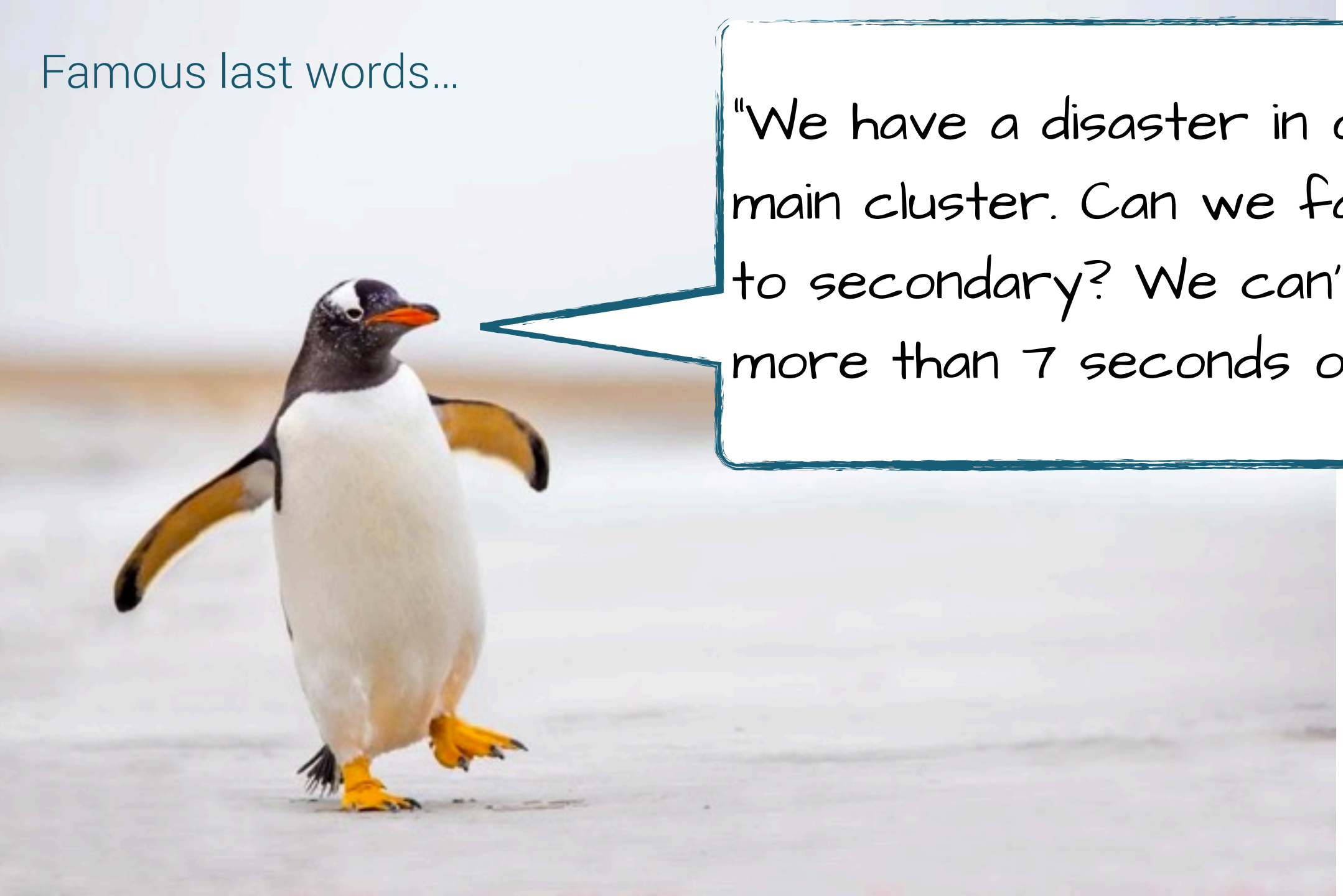
- Watch Group Partition Assignment
- Monitor Client Metrics
- Understand Consumer Rebalancing
- Profile their applications
- Distinguish Client/App/Broker problems

~~Replication~~ Everything is Slow

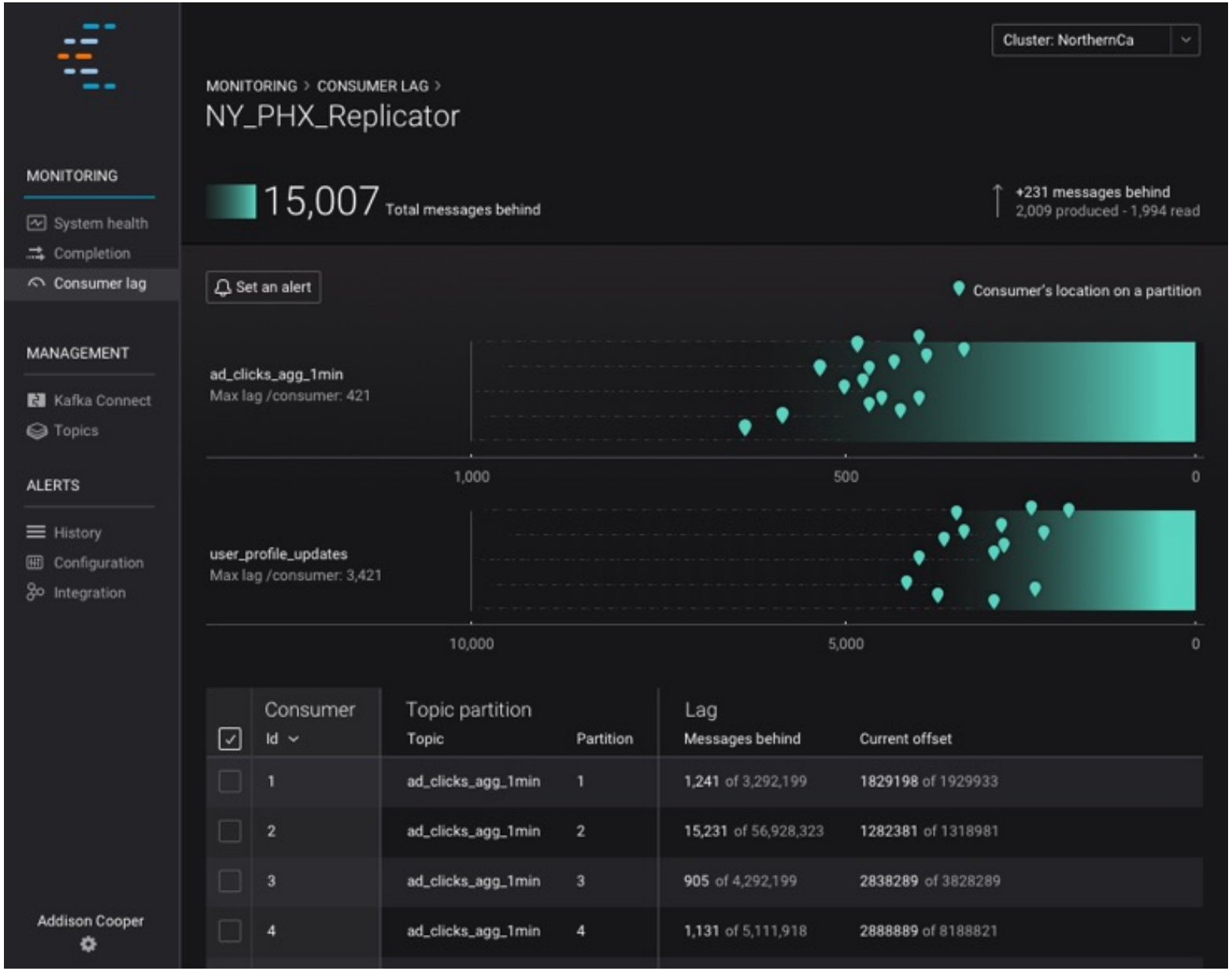
Famous last words…

"You just consume, and produce. How hard can this be?"

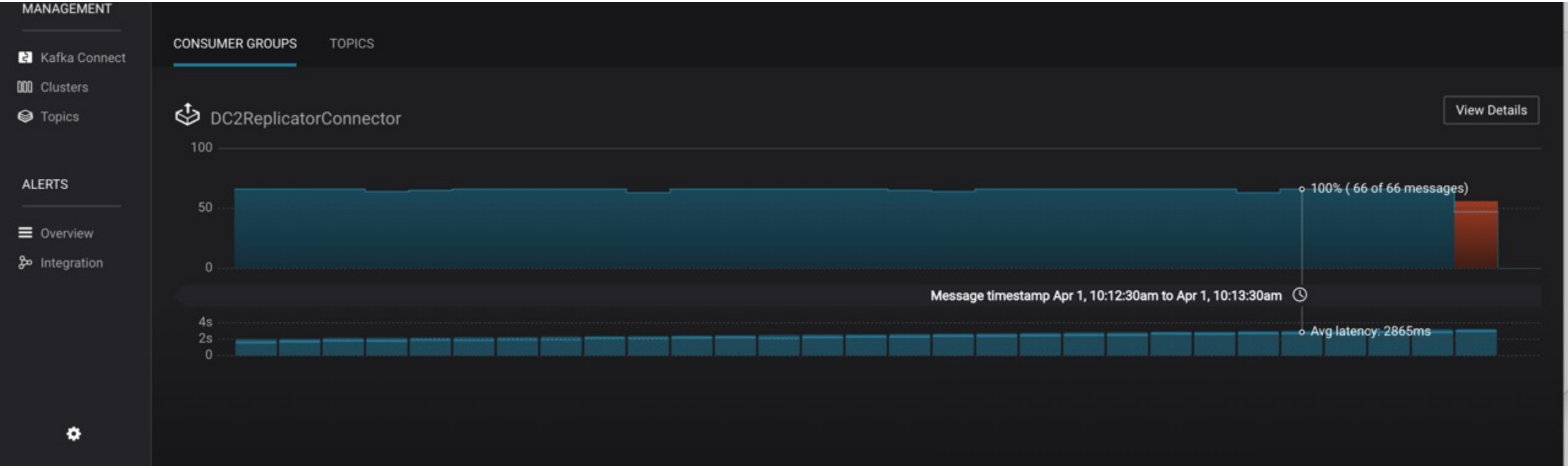
Famous last words…



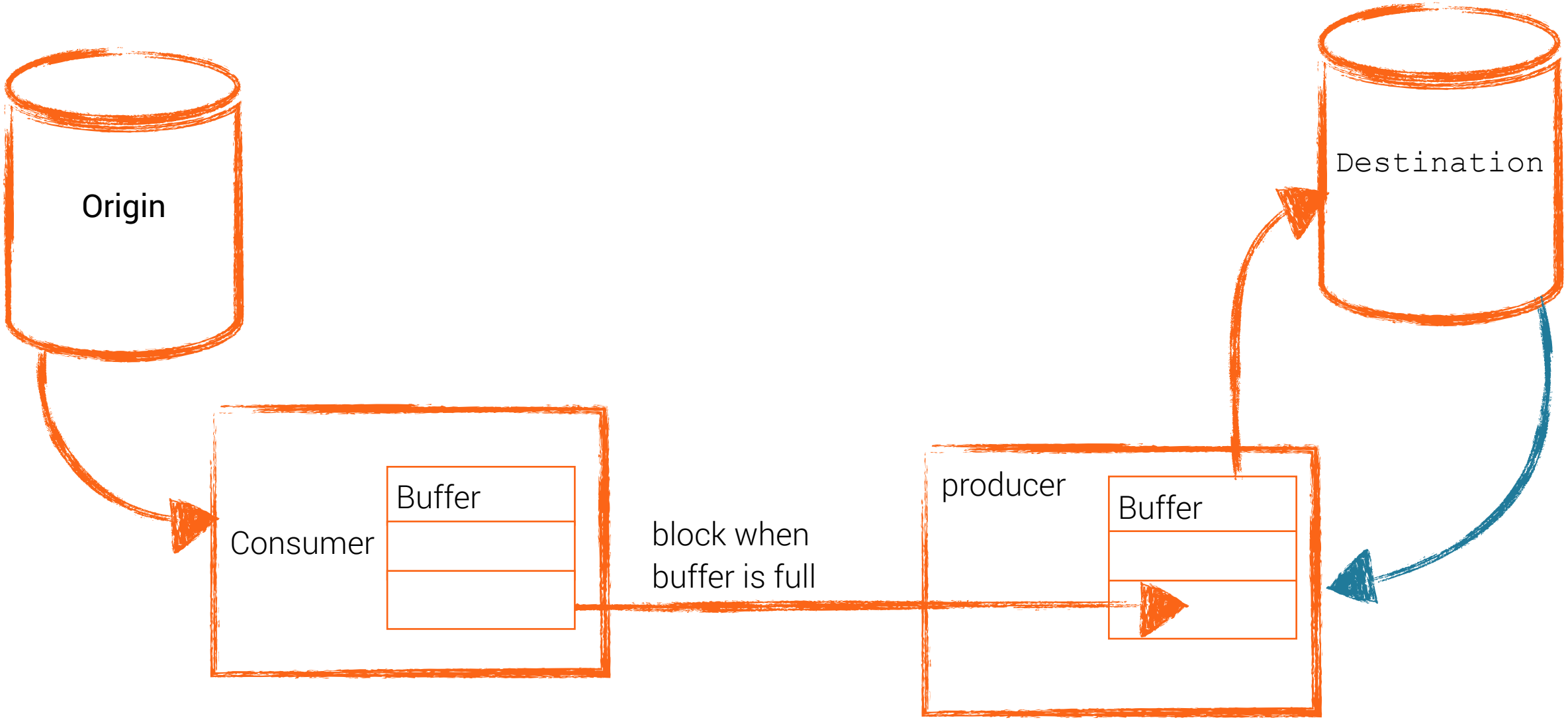
Monitor Replication Lag - In messages



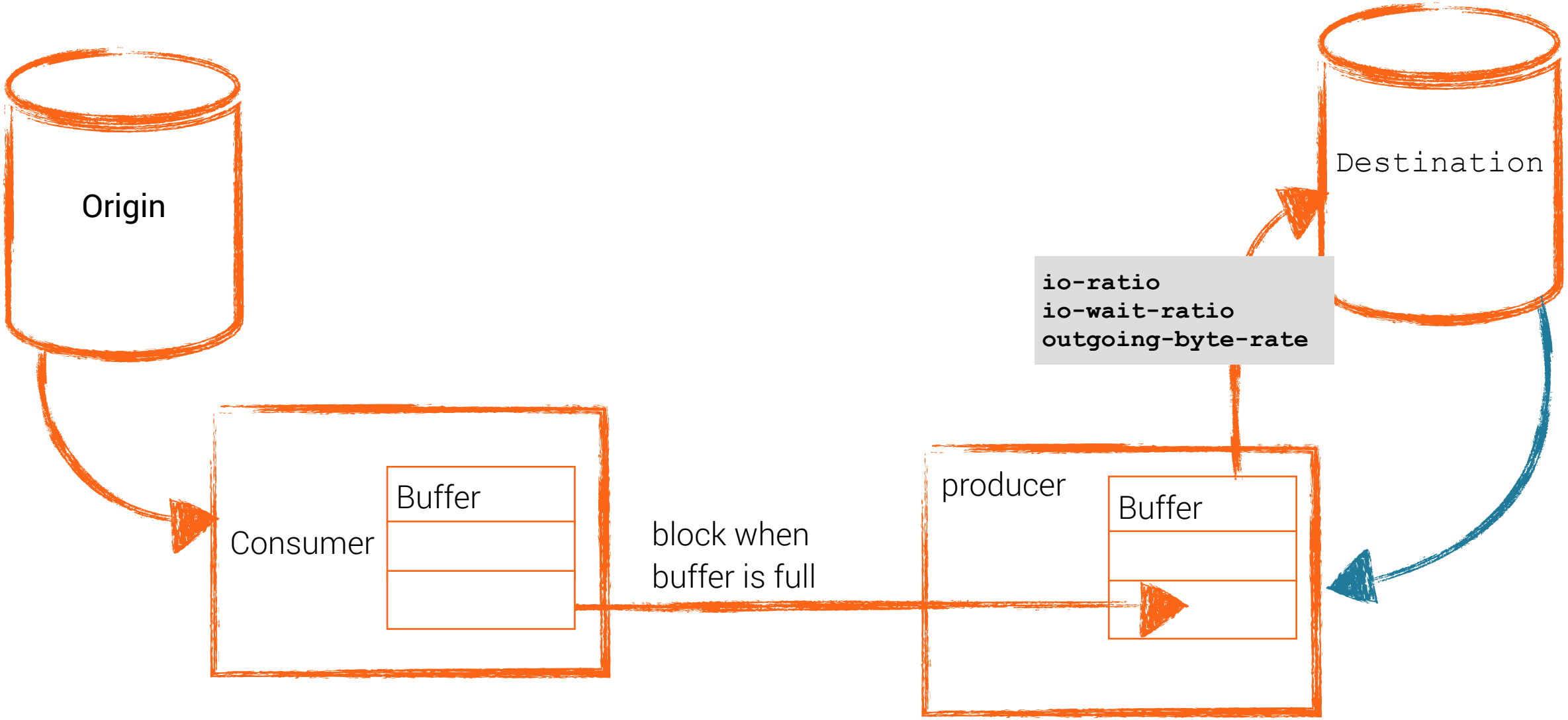
Monitor Replication Lag - or in seconds…



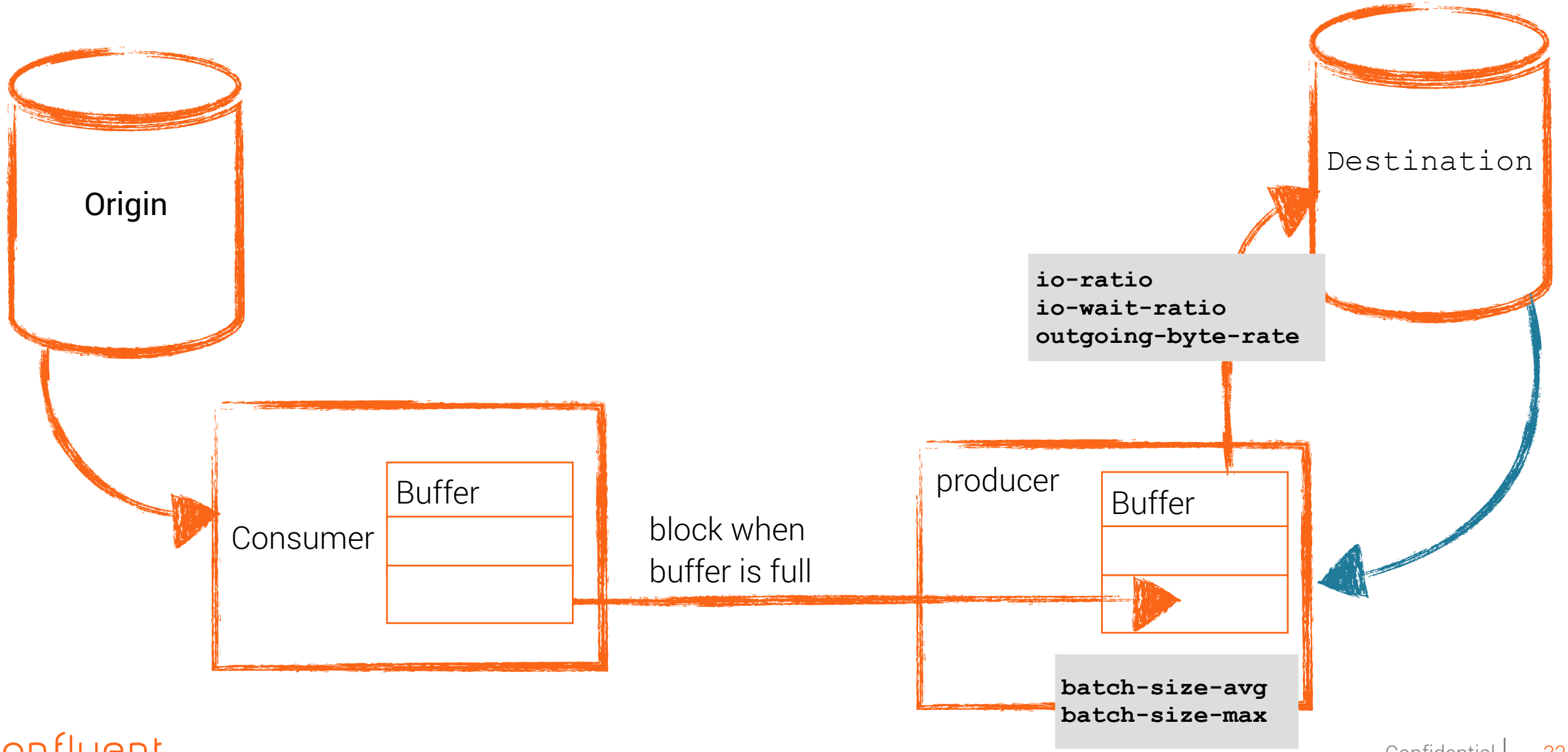
Simple and elegant design



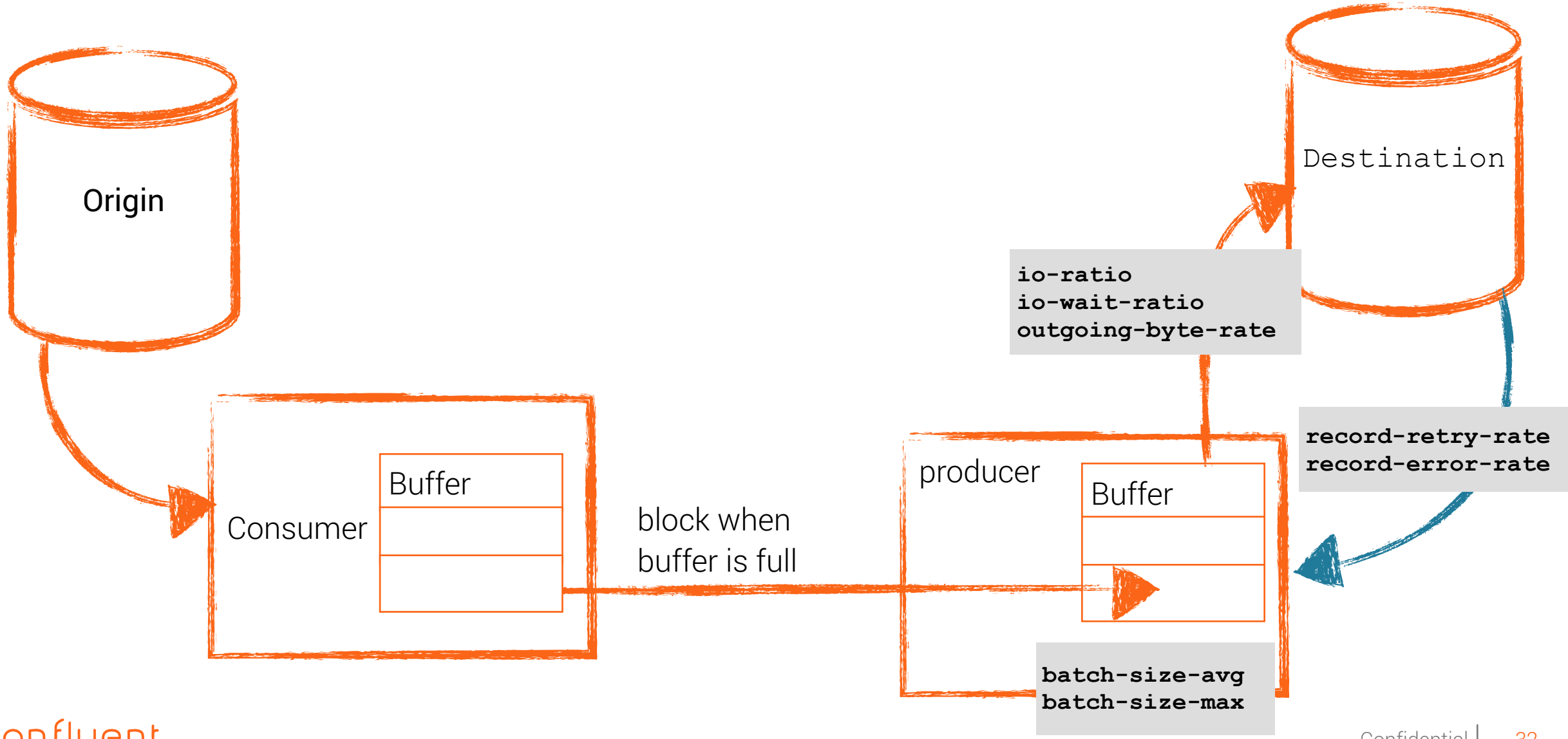
Simple and elegant design



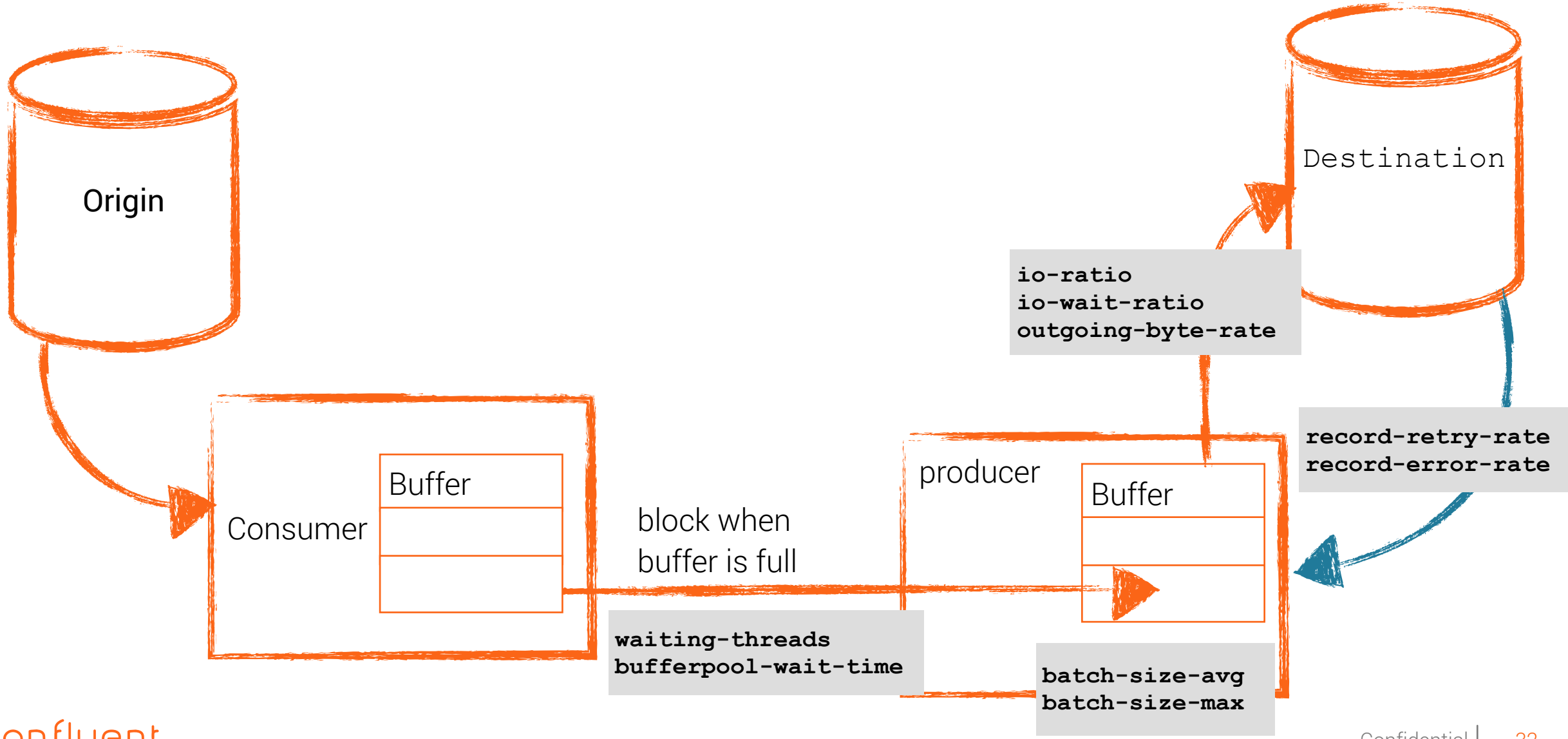
Simple and elegant design



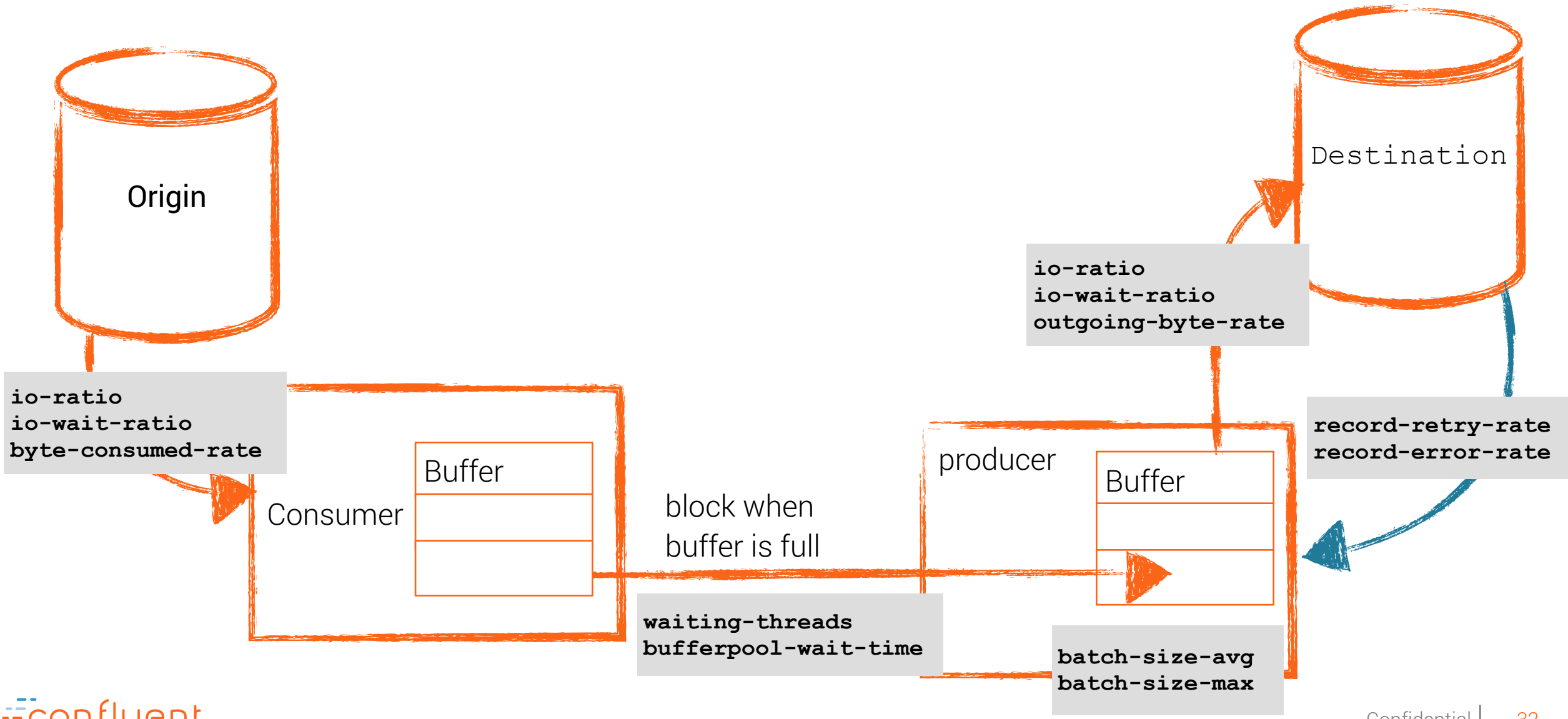
Simple and elegant design



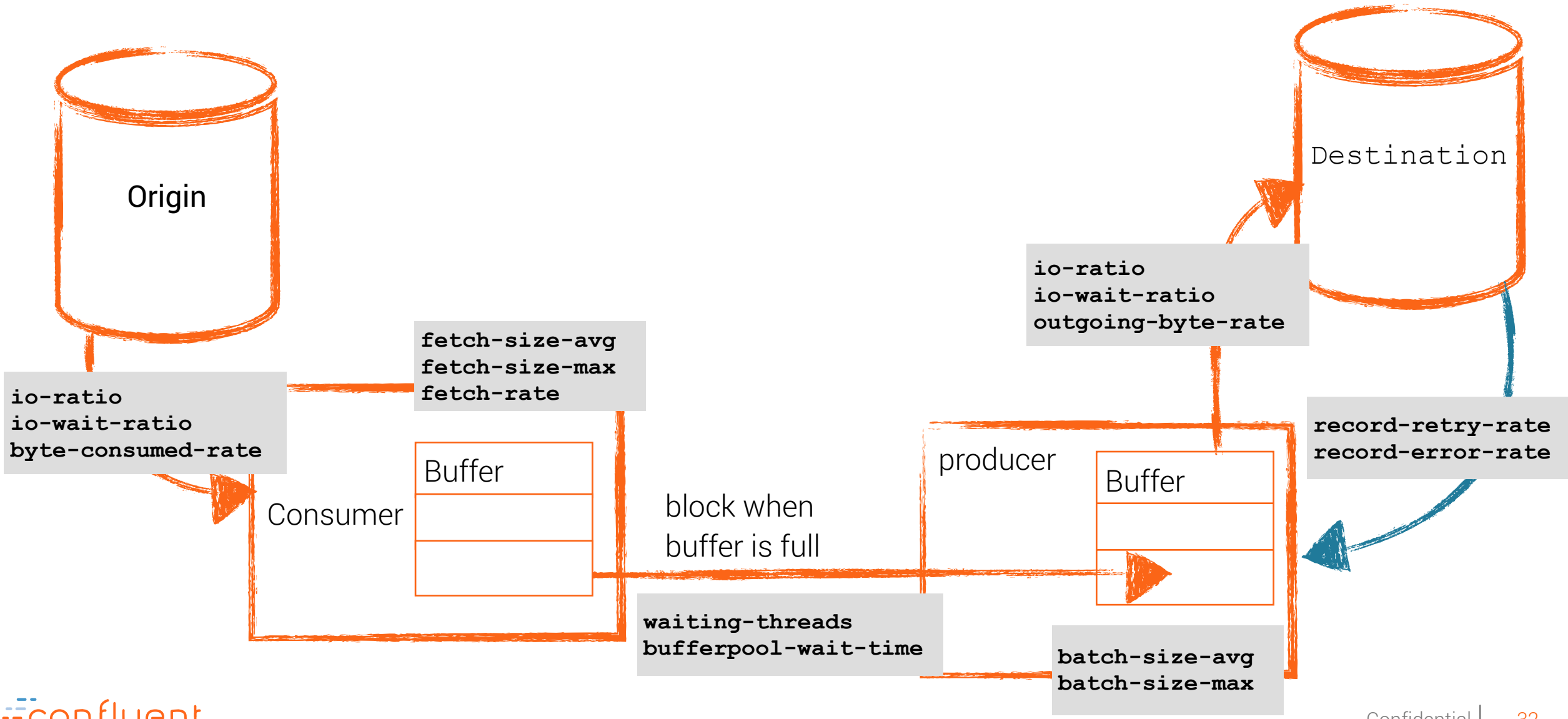
Simple and elegant design



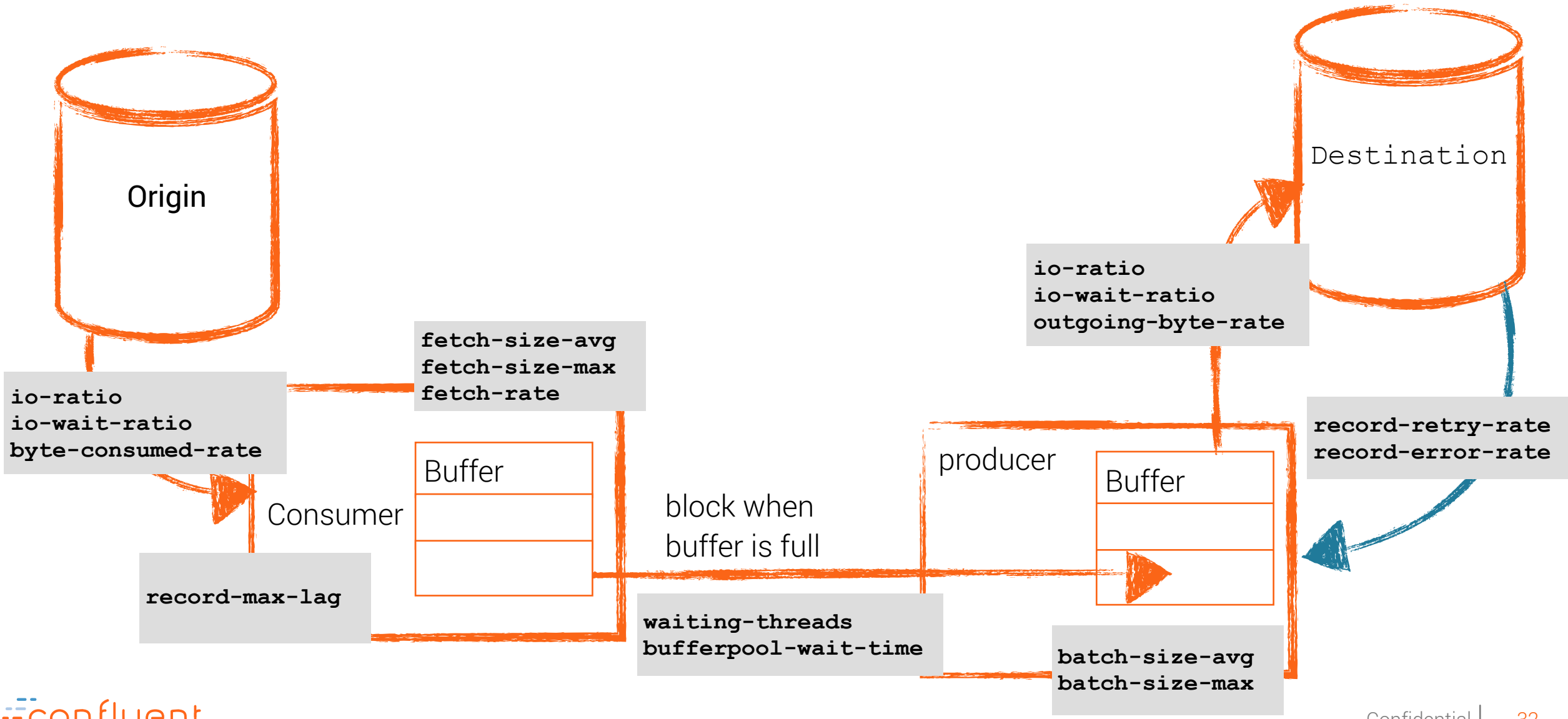
Simple and elegant design



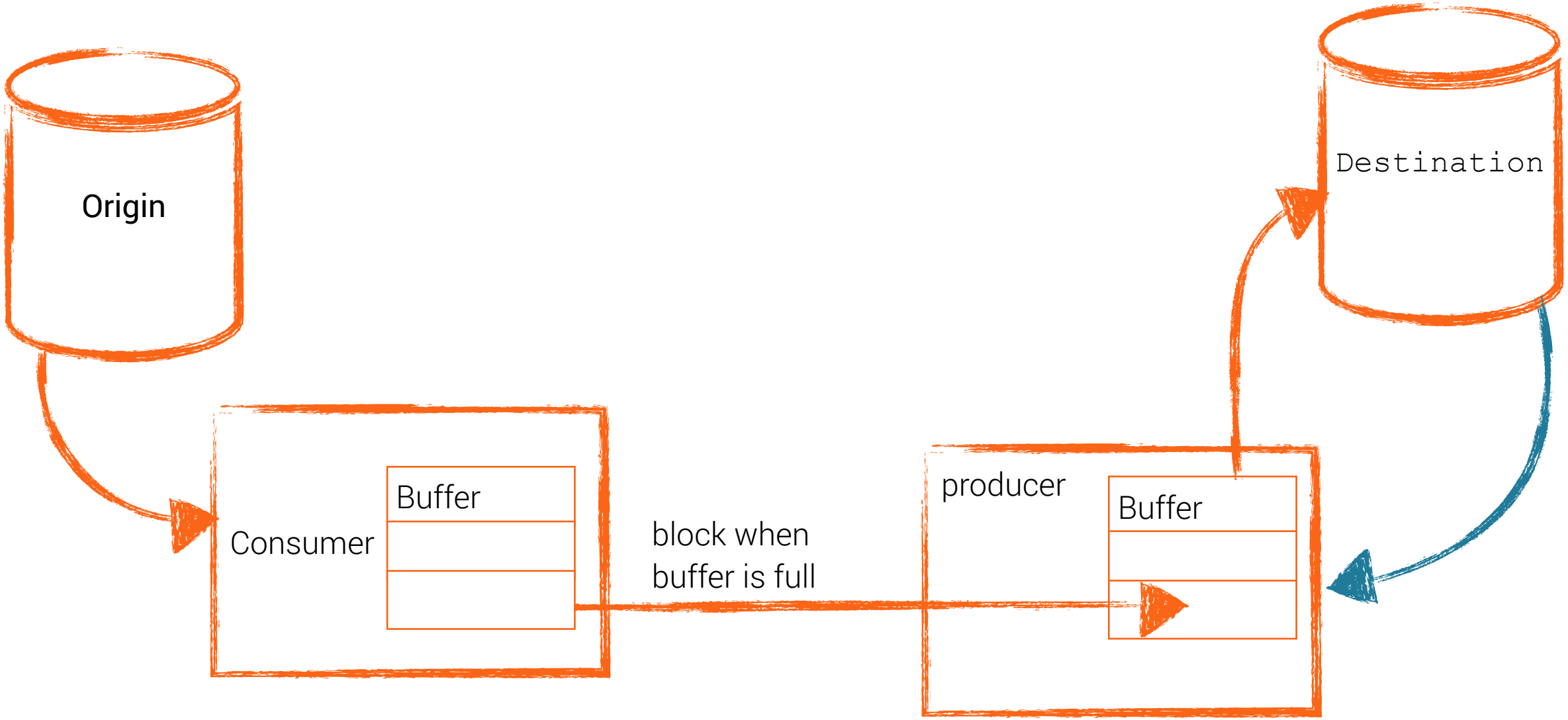
Simple and elegant design



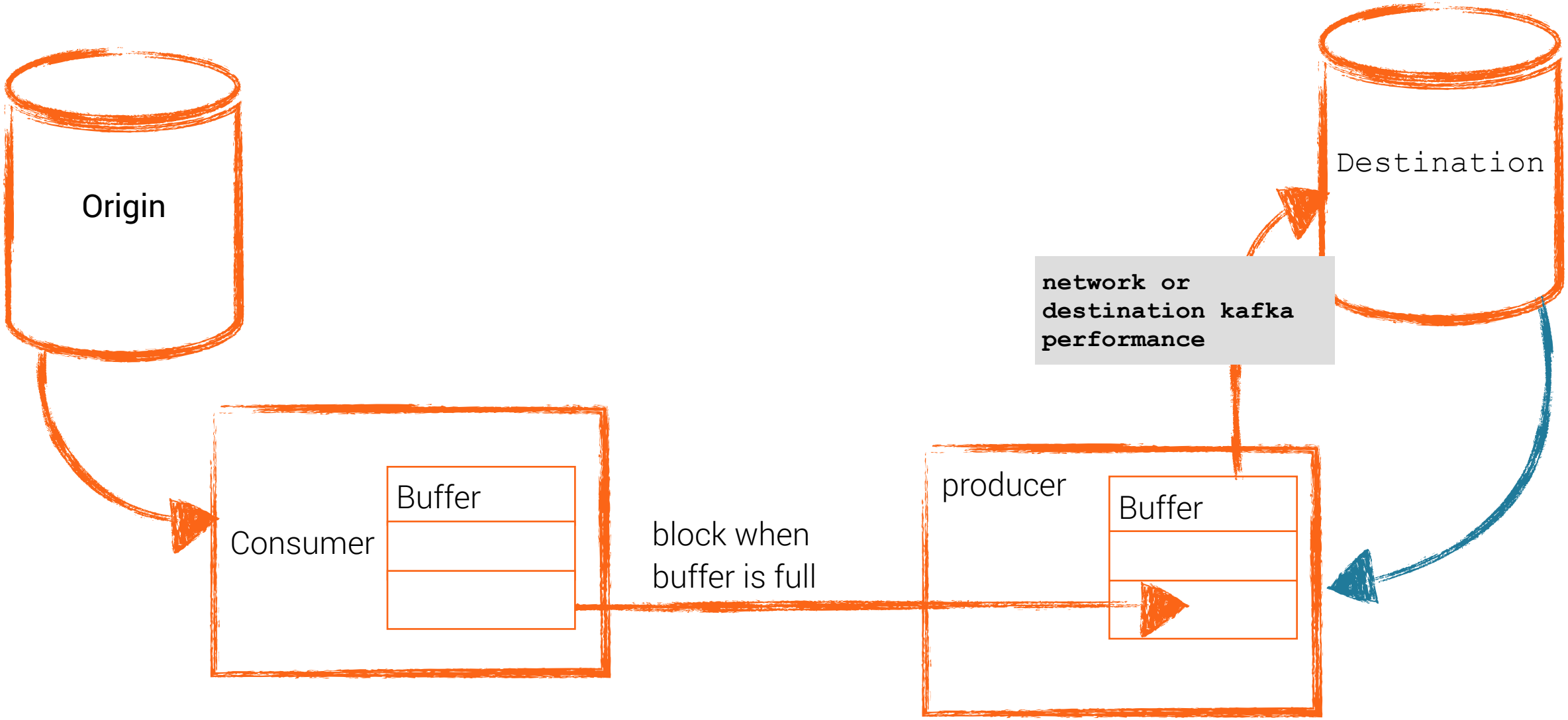
Simple and elegant design



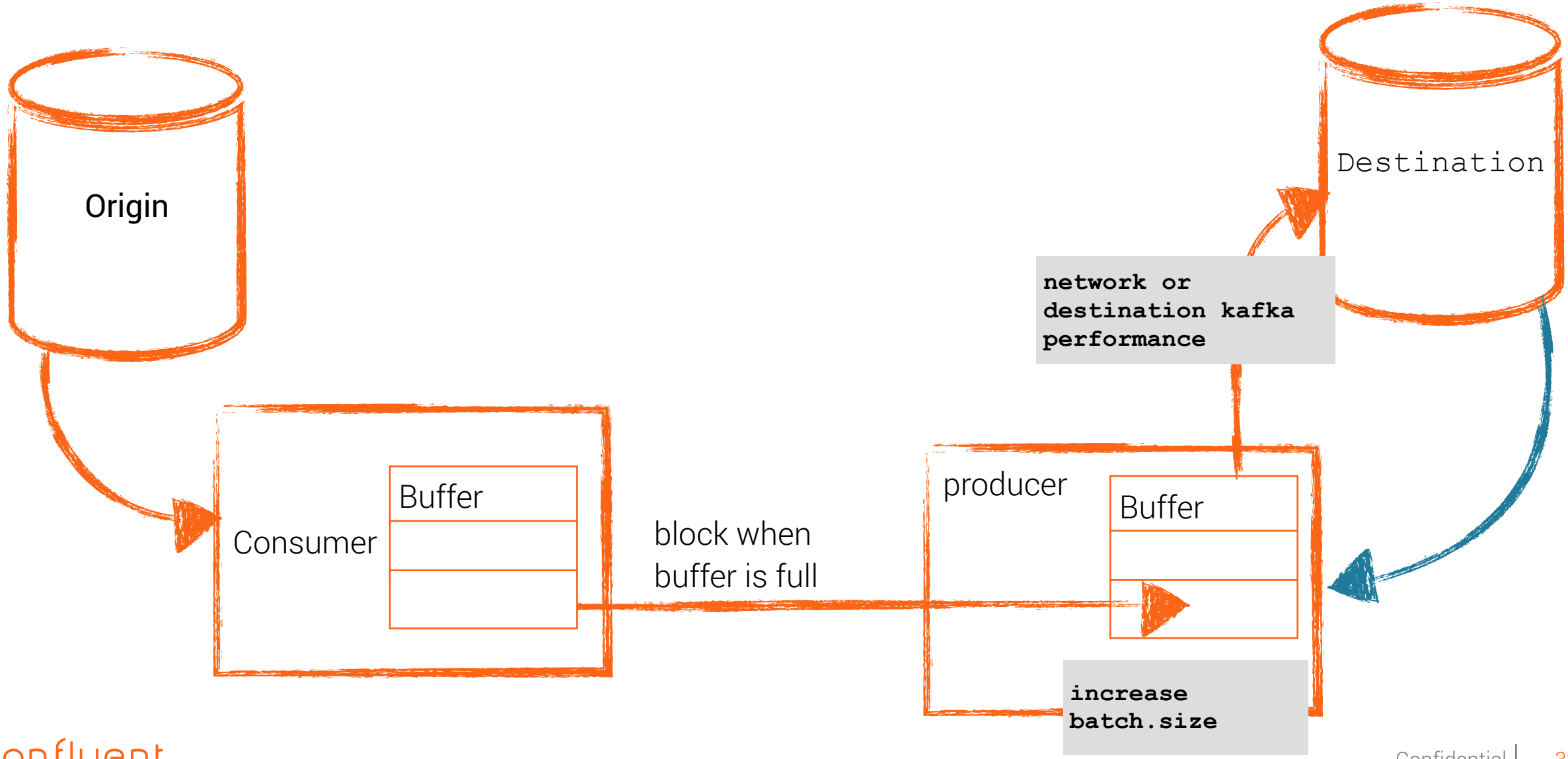
Simple and elegant design



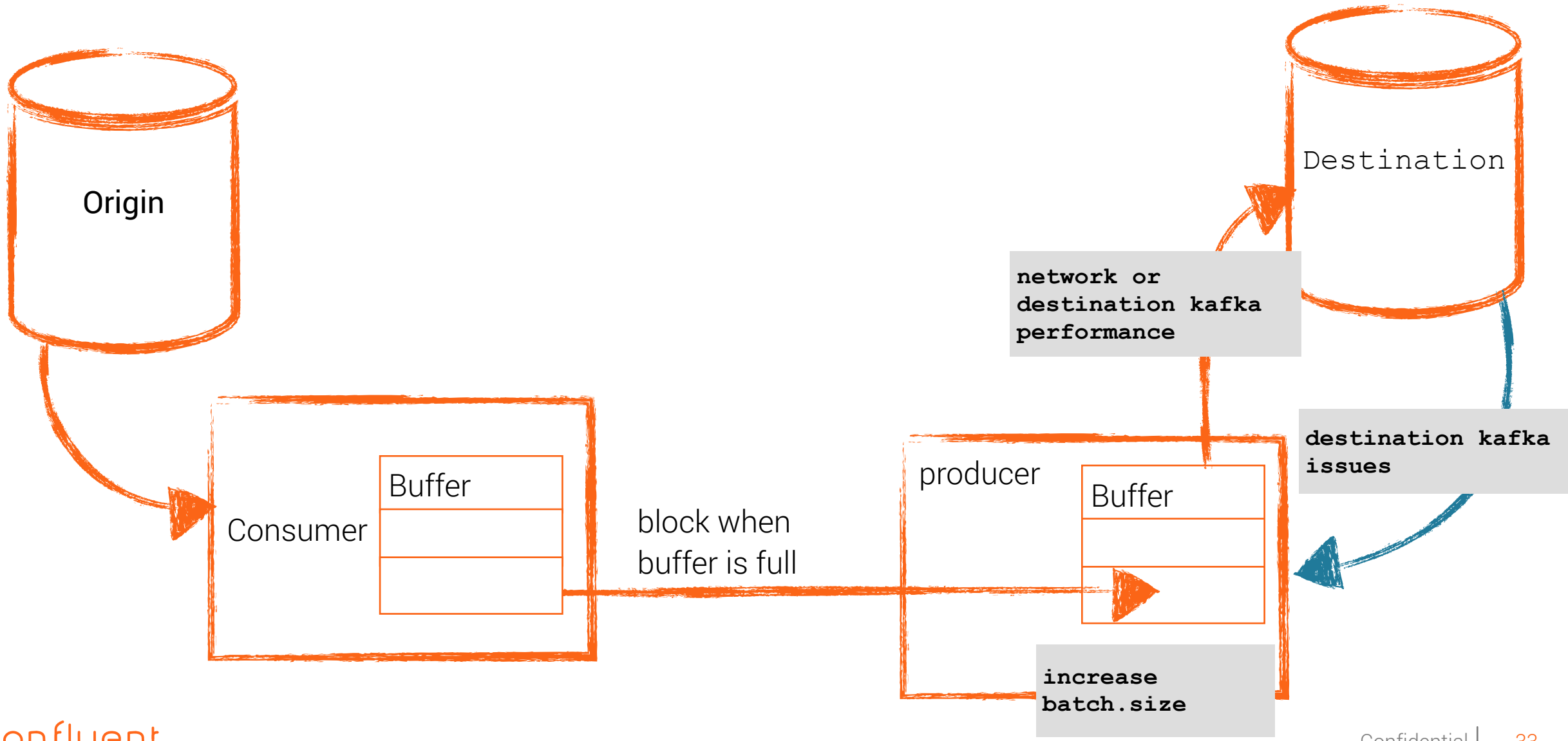
Simple and elegant design



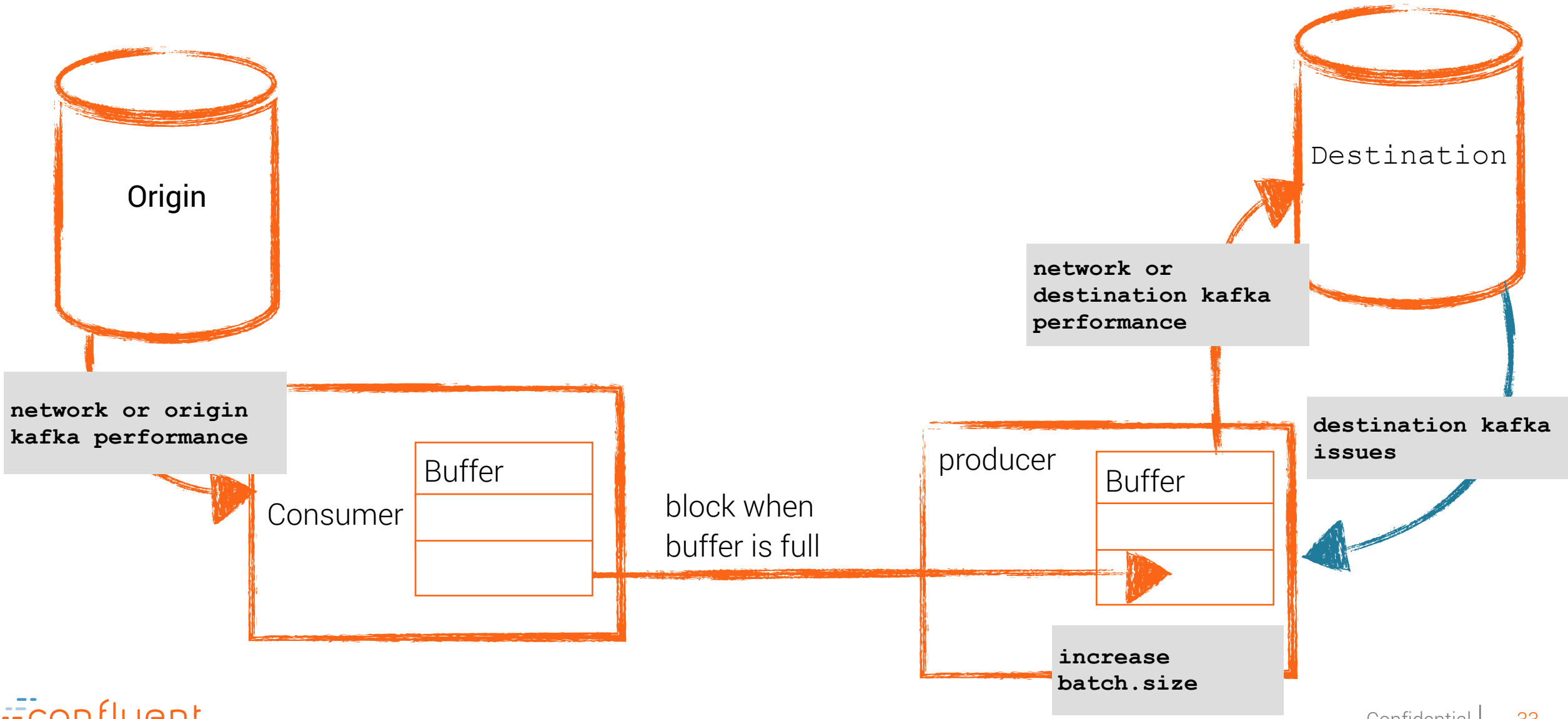
Simple and elegant design



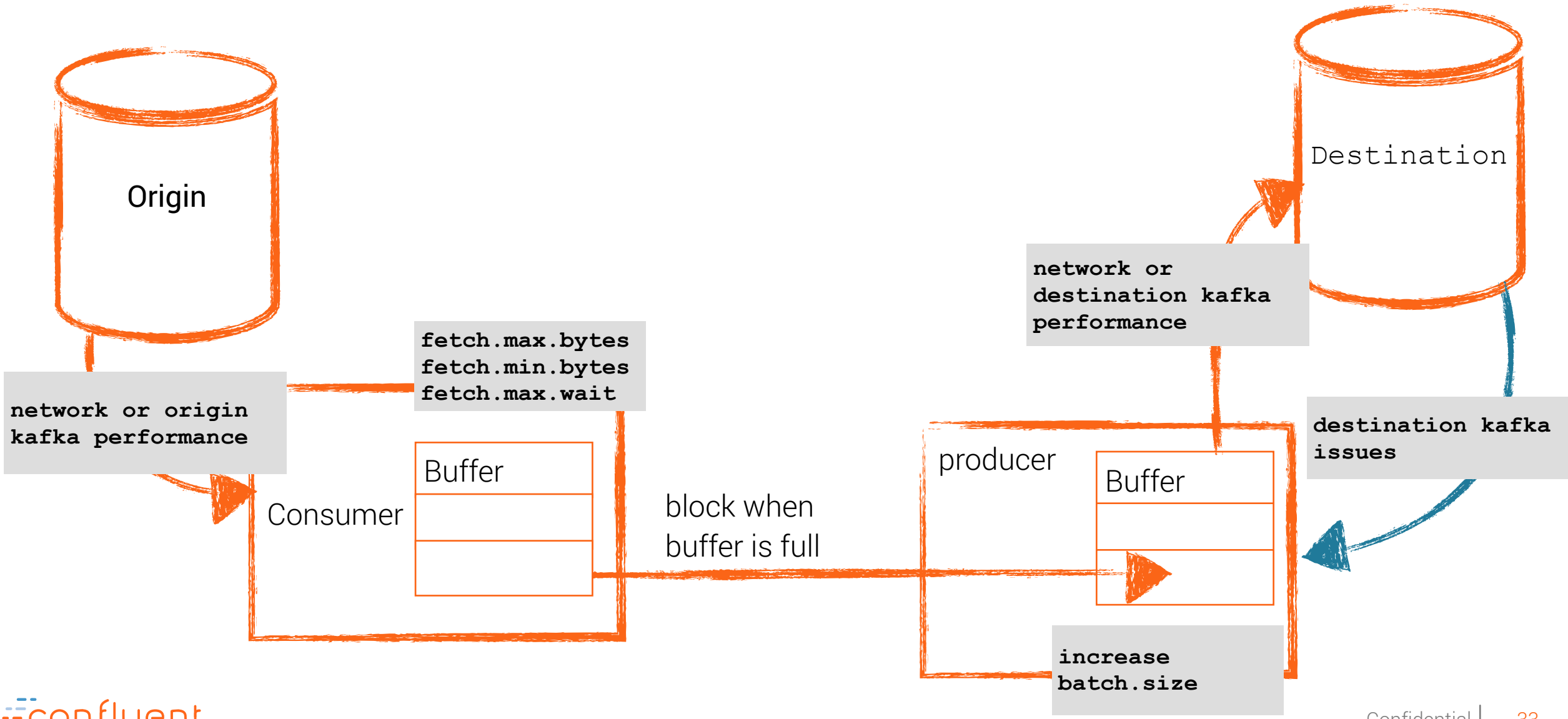
Simple and elegant design



Simple and elegant design



Simple and elegant design



Network Tuning

- WAN has high latency. We deal with it.
- Compute buffer size to match: https://www.switch.ch/network/tools/tcp_throughput/
- `send.buffer.bytes` and `receive.buffer.bytes` on producer, consumer, brokers
- OS tuning: https://wwwx.cs.unc.edu/~sparkst/howto/network_tuning.php

  `net.core.rmem_default, net.core.rmem_max, net.core.wmem_default, net.core.wmem_max`
- Enable logging to check if this had any effect:

  `log4j.logger.org.apache.kafka.common.network.Selector=DEBUG`
- Additional tips in our docs

Competent users

- Monitor consumer lag
- Add processes when things are slow
- Automate deployment

Kafka Pros

- Monitor time lag
- Collect client metrics
- Knows which side to blame
- Know which configs to tune
- Tunes the network over the WAN

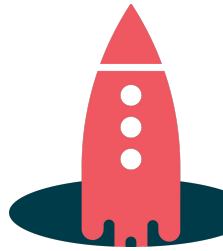
Resources and Next Steps

https://github.com/confluentinc/cp-demo

https://www.confluent.io/download/

https://www.confluent.io/blog

https://slackpass.io/confluentcommunity



Thank you!

@gwenshap
gwen@confluent.io

@xvrl
xavier@confluent.io

confluent