PULSAR

Fast, Durable, Flexible Pub/Sub based on

Segment-Oriented Architecture

演讲者 / streamlio 翟佳

What is Apache Pulsar?

Durability
Data replicated and synced to disk

Ordering
Guaranteed ordering

Delivery Guarantees
At least once, at most once and effectively once

Geo-replication
Out of box support for geographically distributed applications

Multi-tenancy
A single cluster can support many tenants and use cases

Low Latency
Low publish latency of 5ms at 99pct

Unified messaging model
Support both Topic & Queue semantic in a single model

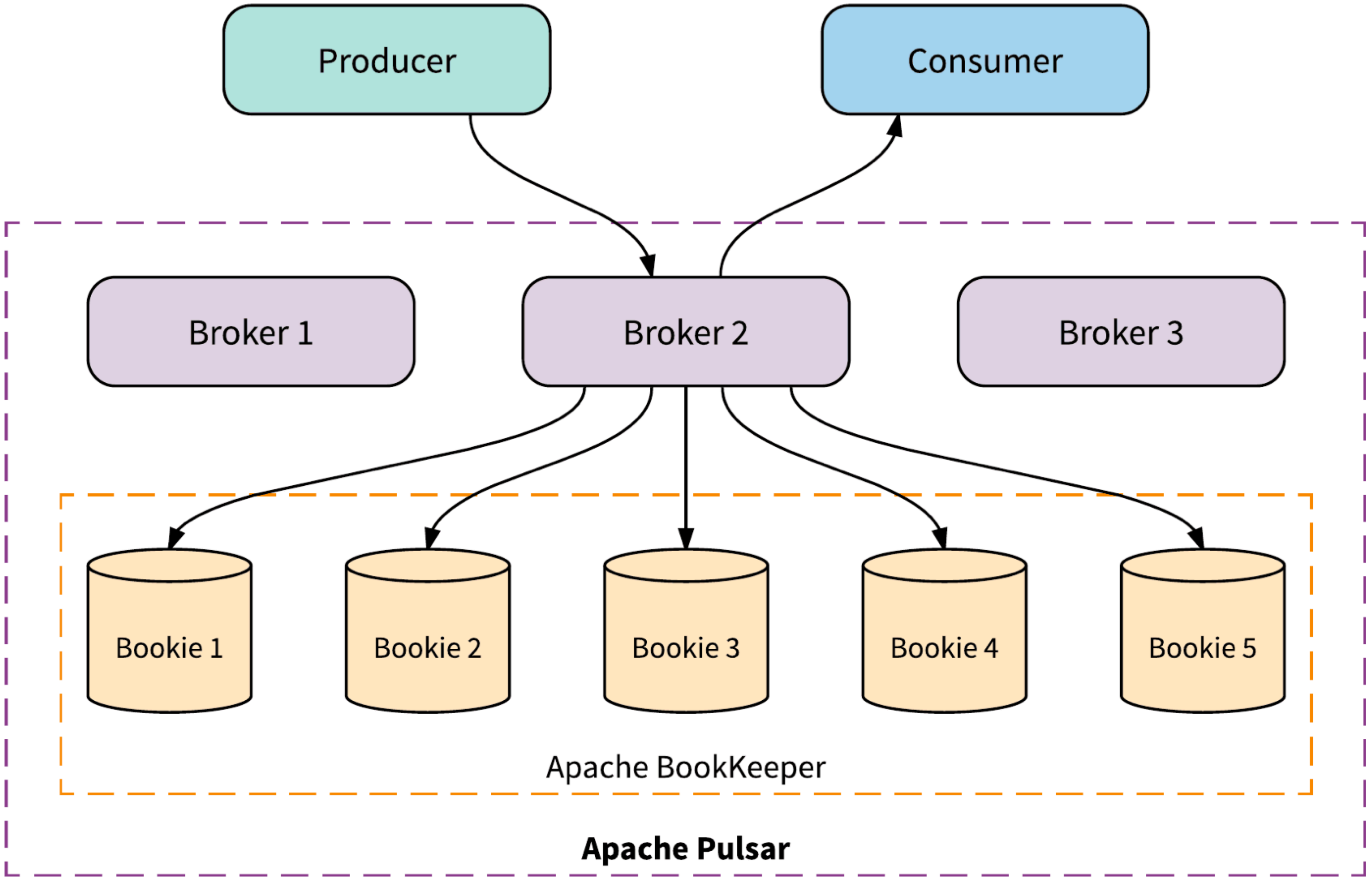
High throughput
Can reach 1.8 M messages/s in a single partition

Highly scalable
Can support millions of topics

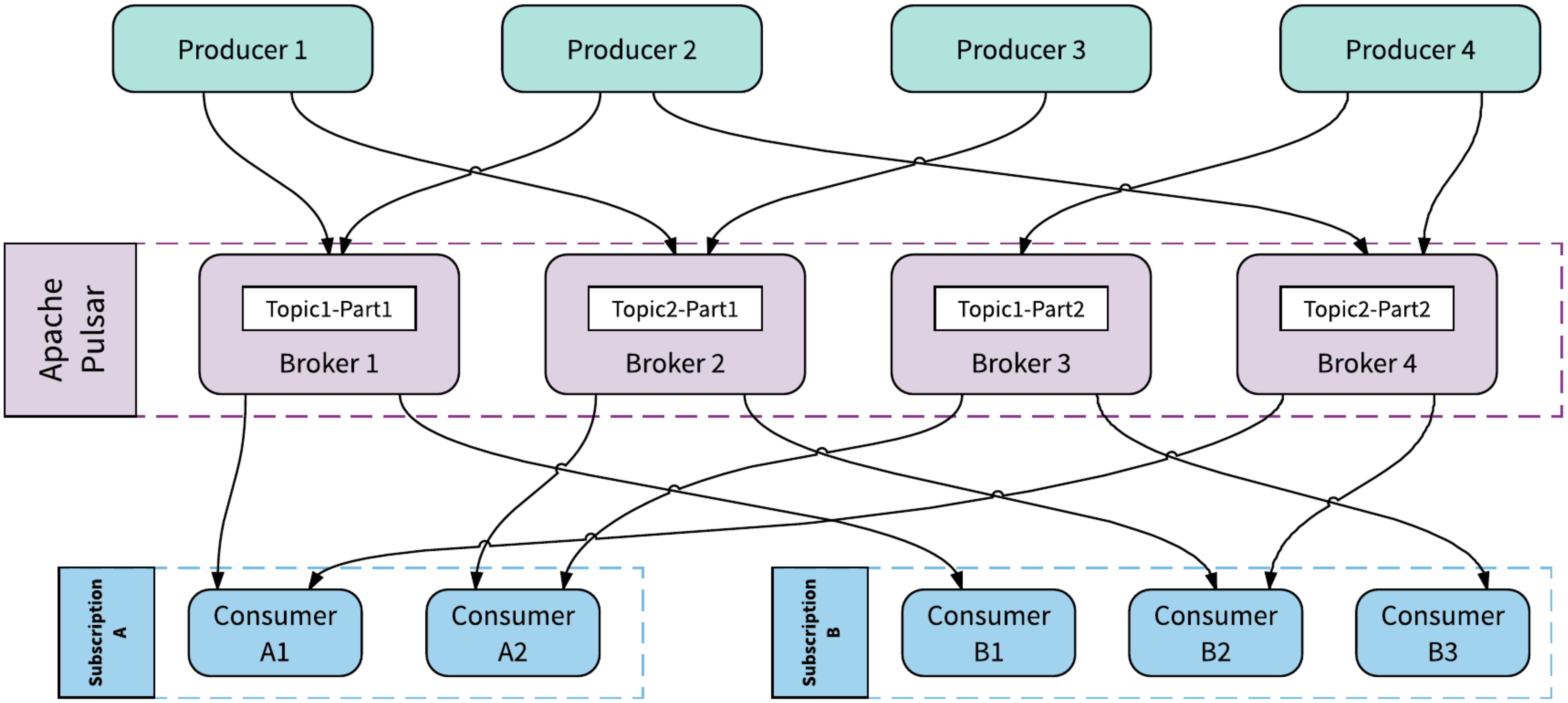
Architecture

Architecture view

- Separate layers between brokers bookies

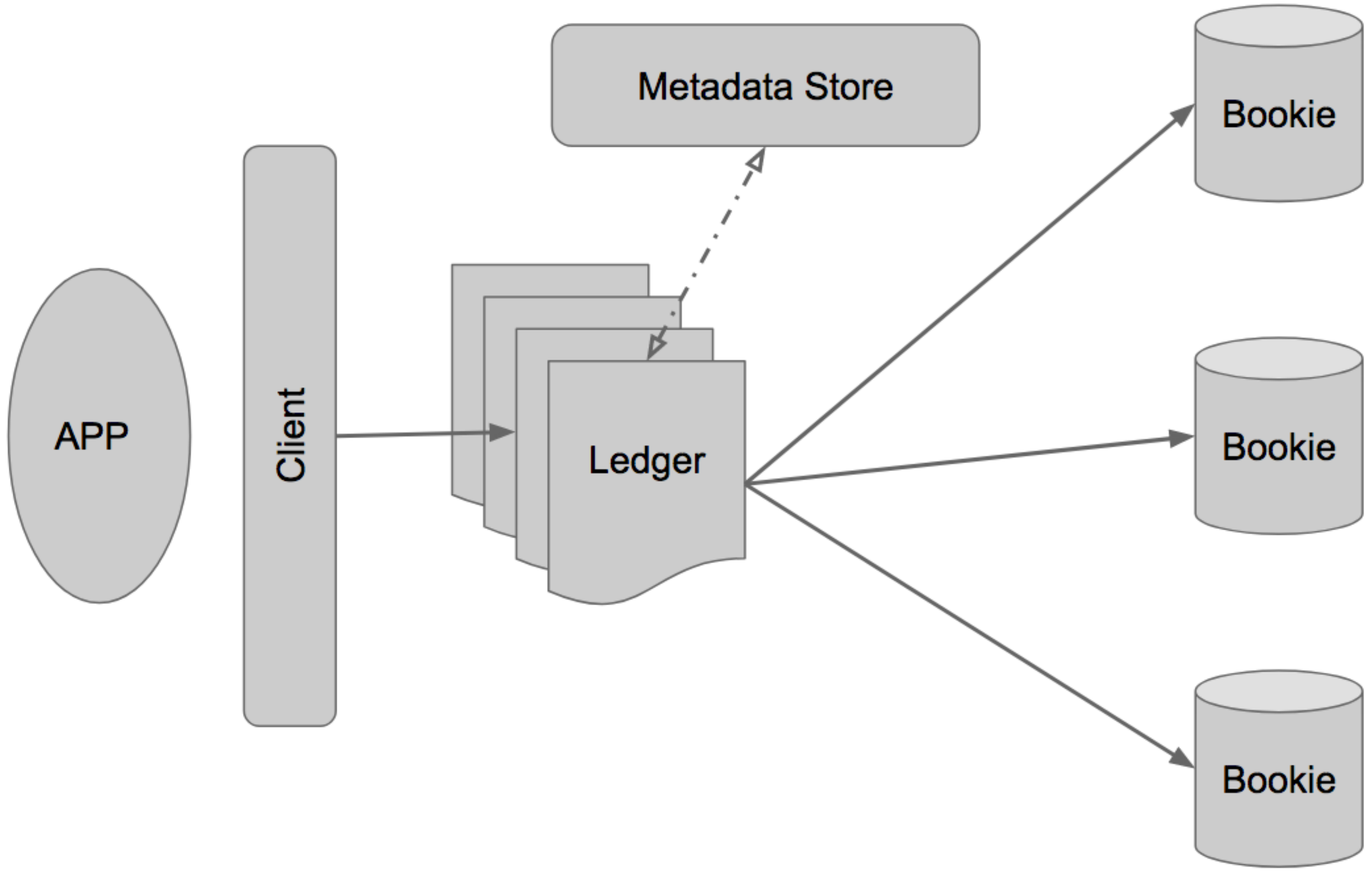


Brokers

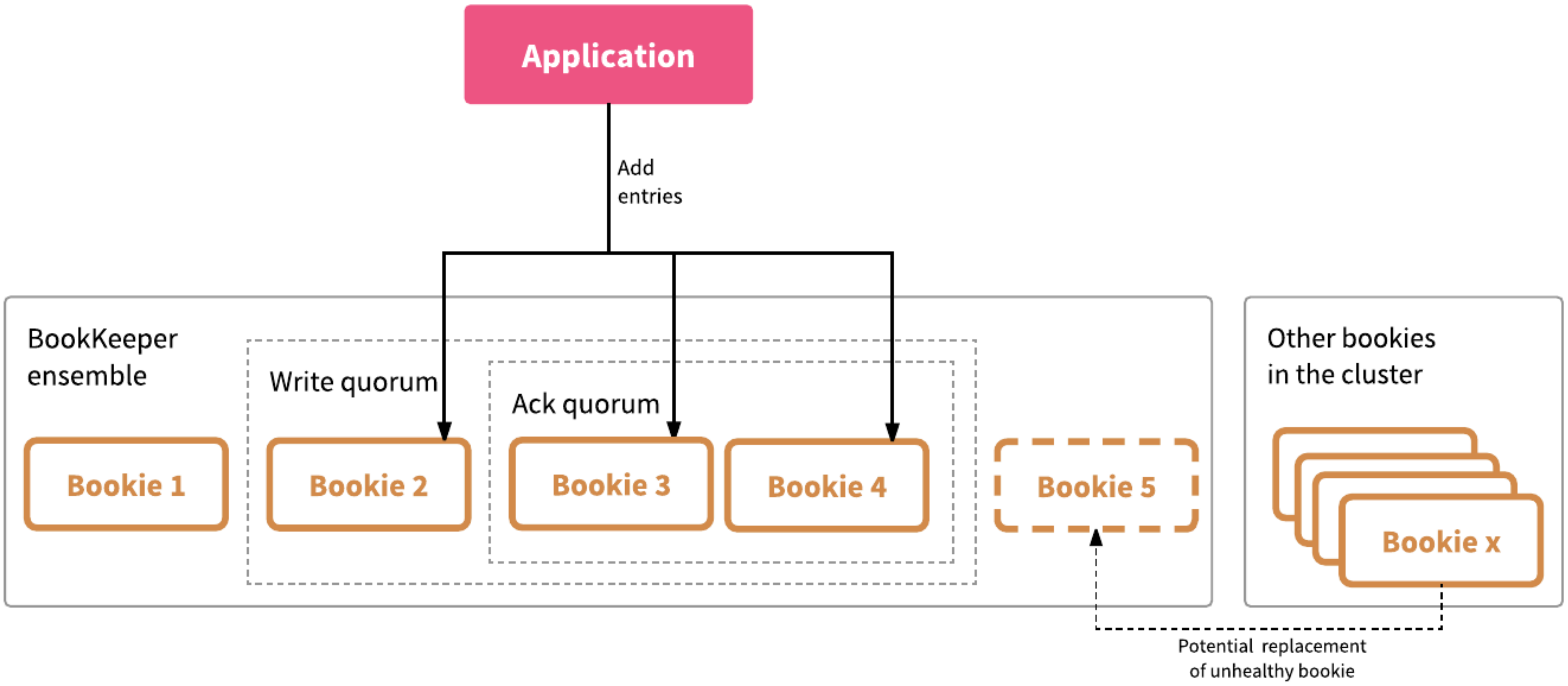

Producer 1
Producer 2
Producer 3
Producer 4
Apache Pulsar
Topic1-Part1
Broker 1
Topic2-Part1
Broker 2
Topic1-Part2
Broker 3
Topic2-Part2
Broker 4
Subscription A
Consumer A1
Consumer A2
Subscription B
Consumer B1
Consumer B2
Consumer B3

Bookies - Apache BookKeeper

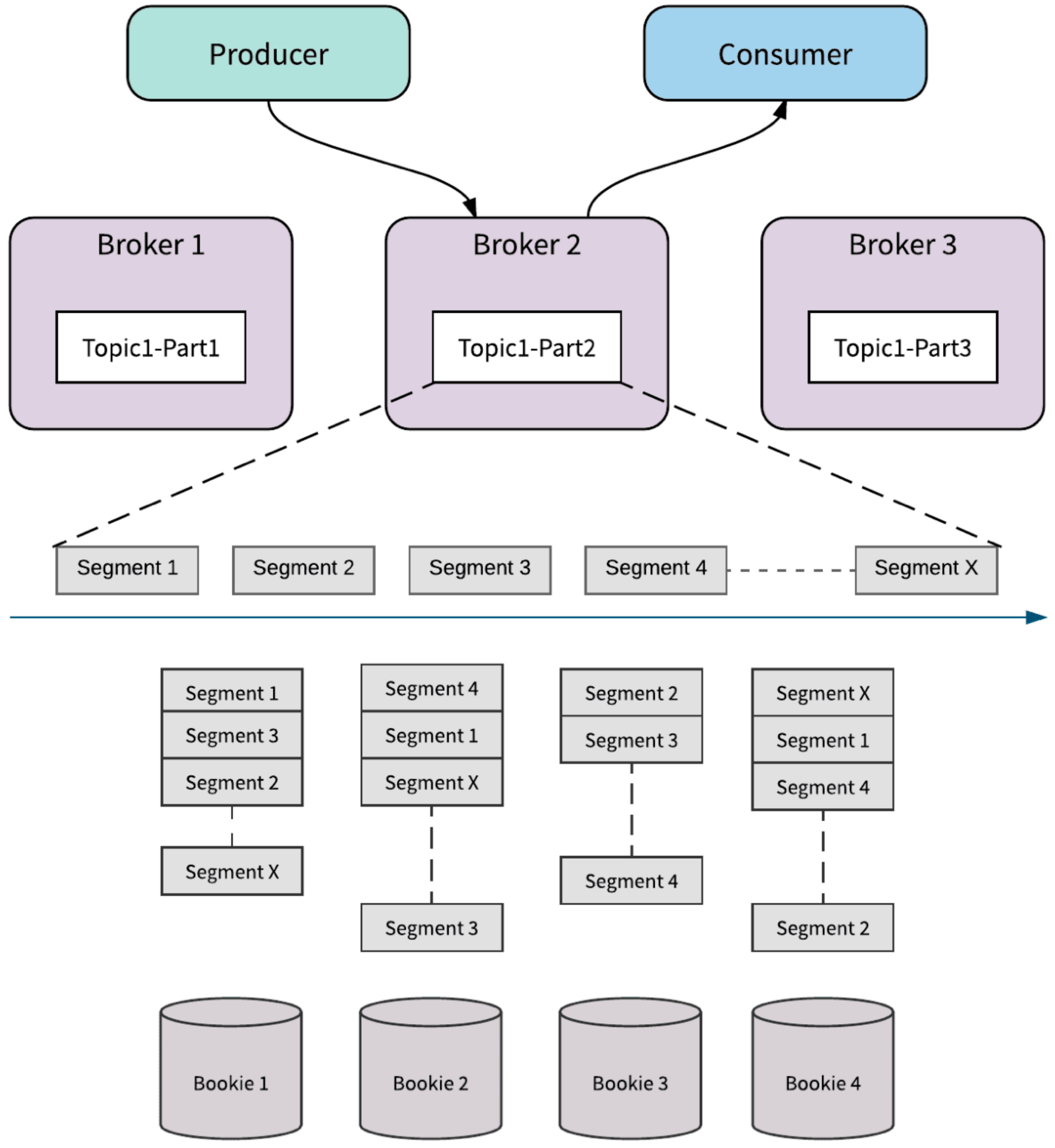
- Durable and Consistent
- I/O Isolation
- High Throughput
- Low Latency



Bookies - Apache BookKeeper



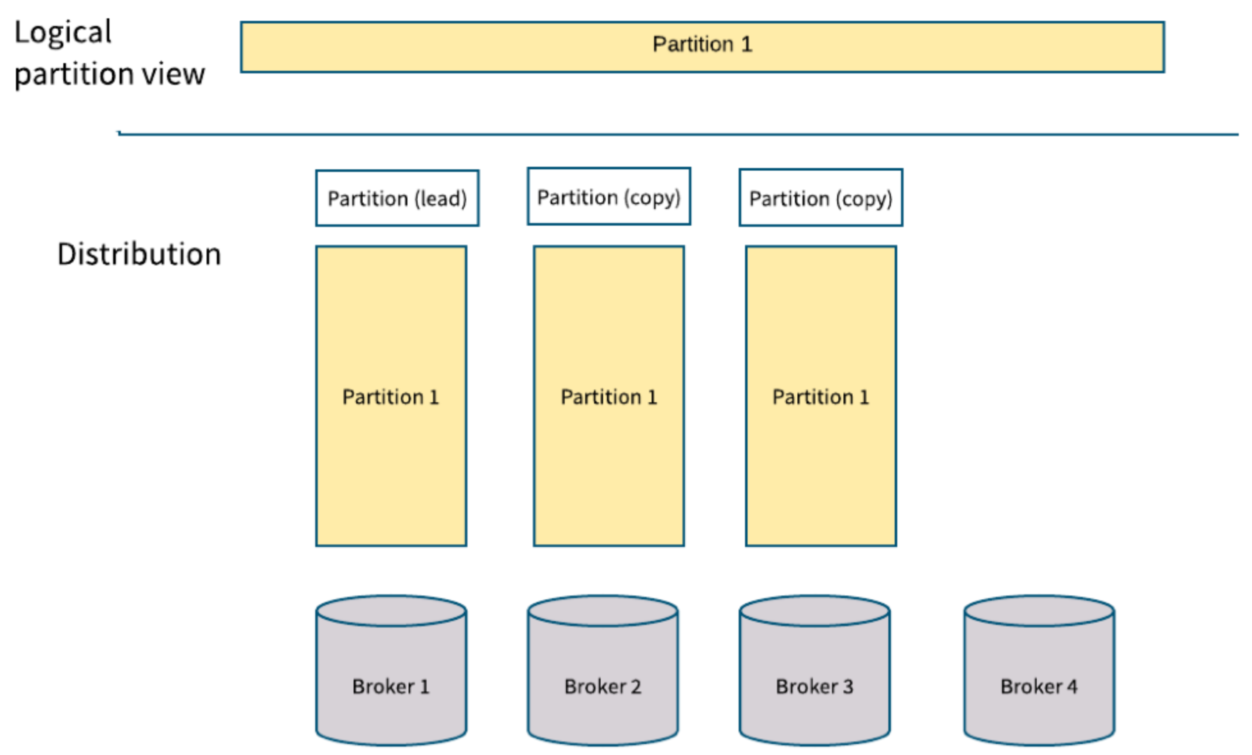
Architecture view



- Unbounded topic partition storage
- Instant scaling without data rebalance
- Independent scalability

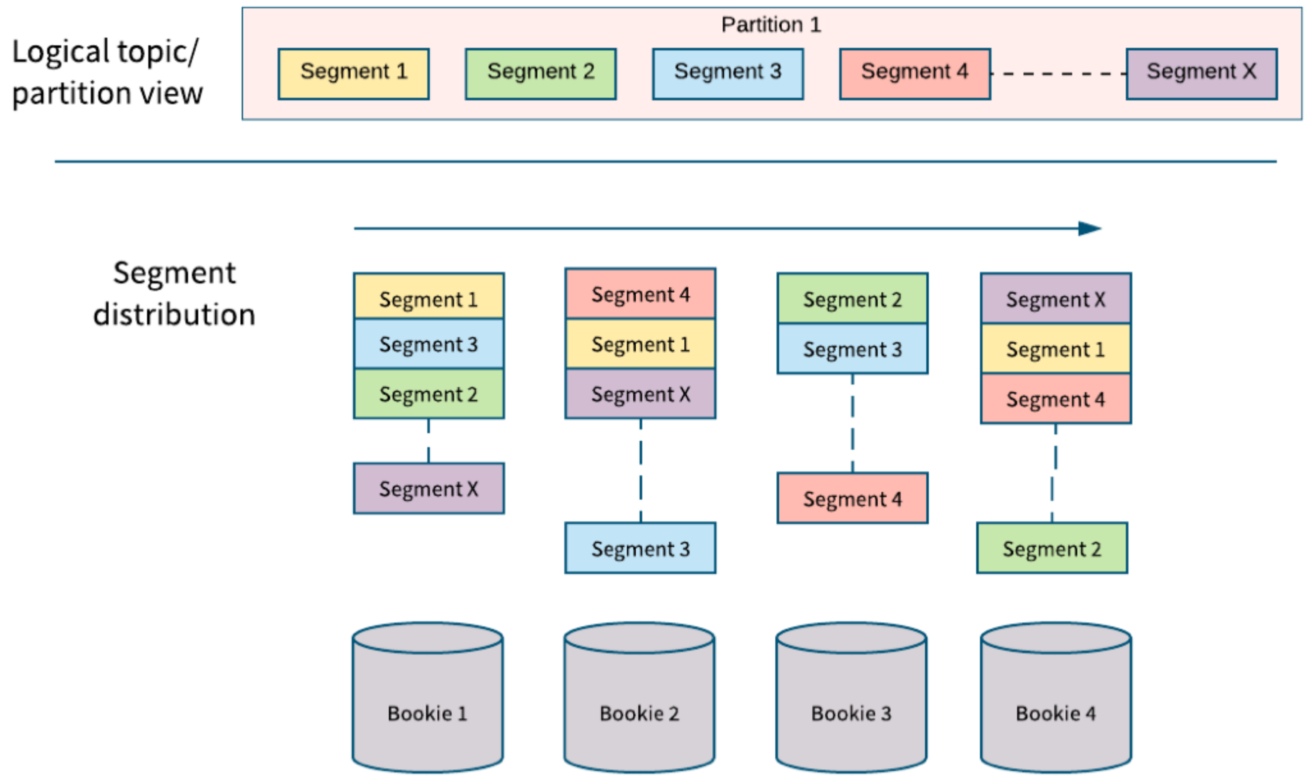
A Compare

Apache Kafka



Kafka Partitions — All log segments are replicated in order across brokers (replication = 3 here).

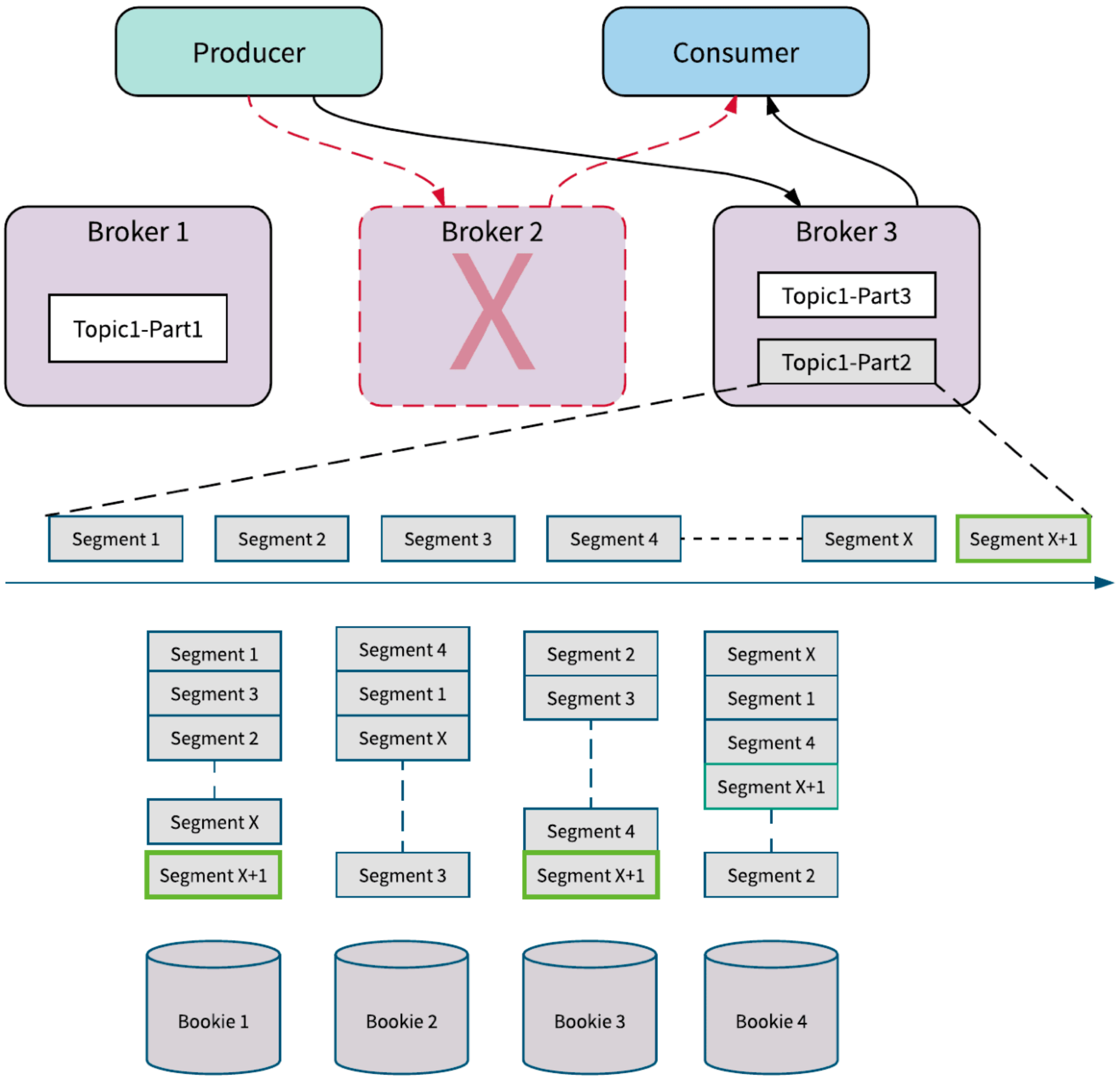
Apache Pulsar/BookKeeper



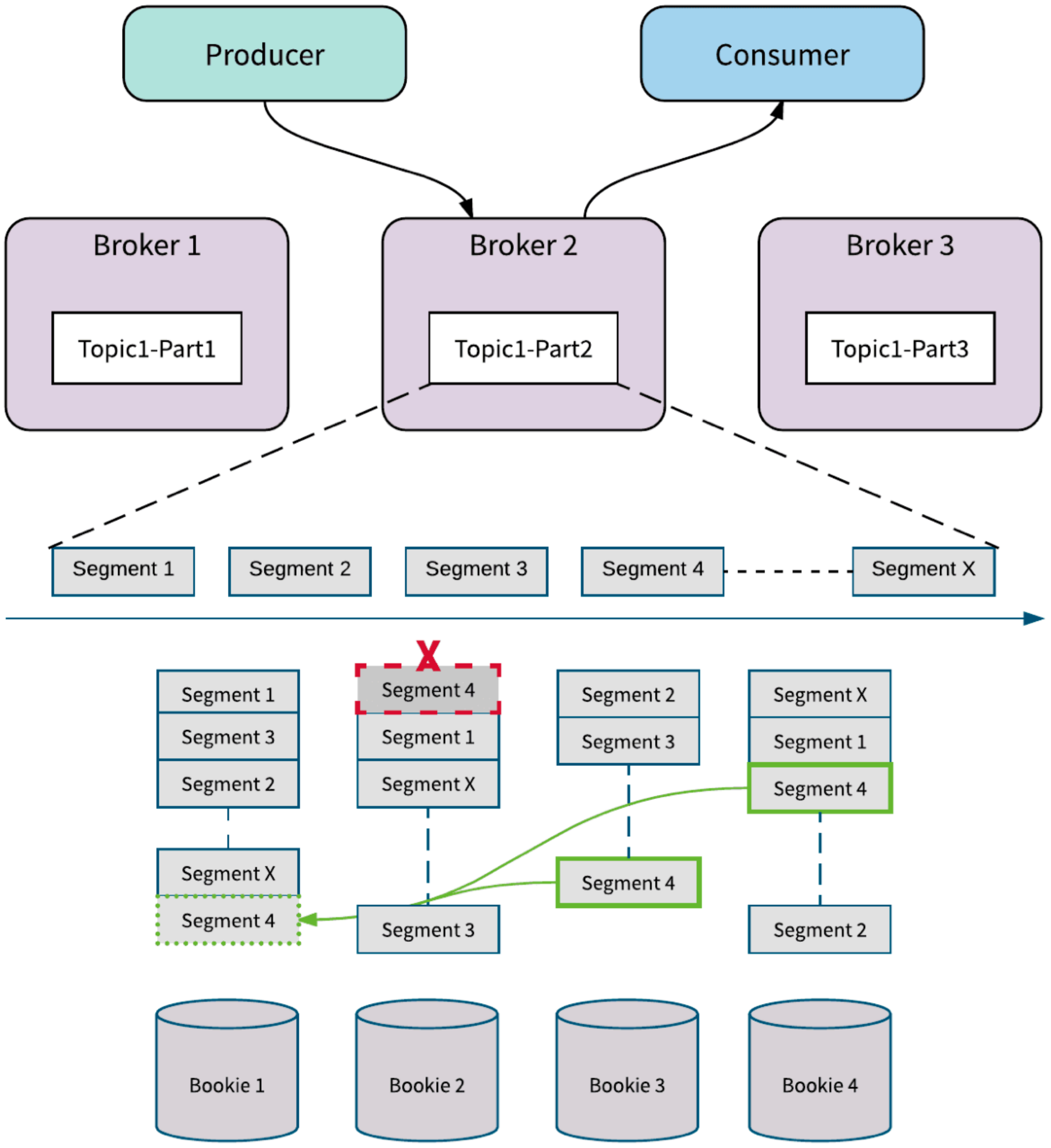
Pulsar/BookKeeper Stream — All log segment are replicated to a configurable number of bookies (replication = 3 here) across N possible bookies (N = 4 here). Log segments are evenly distributed to achive horizontal scalability with no rebalancing.

Benefits

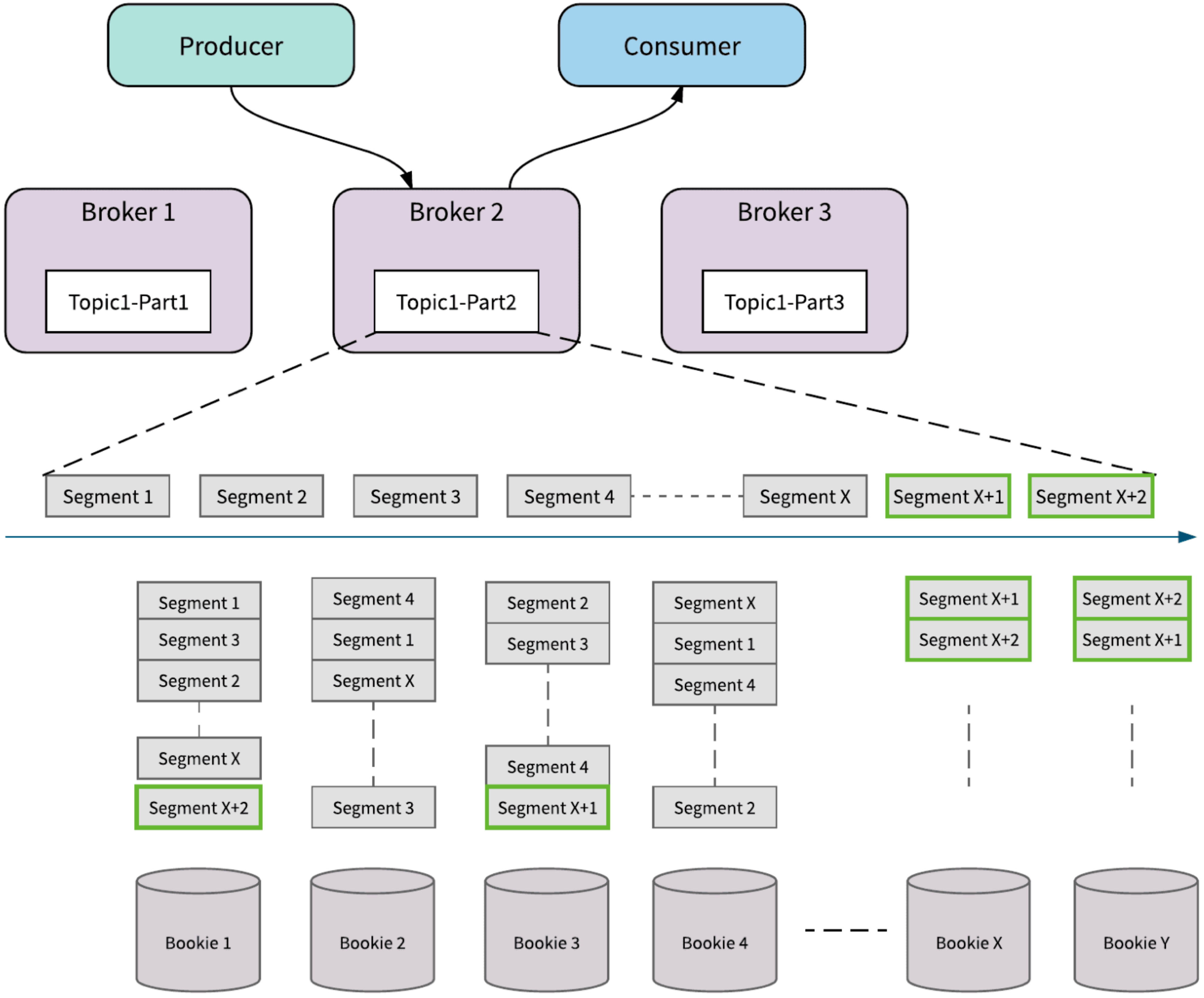
Seamless - broker failure



Seamless - bookie failure



Seamless - cluster expand



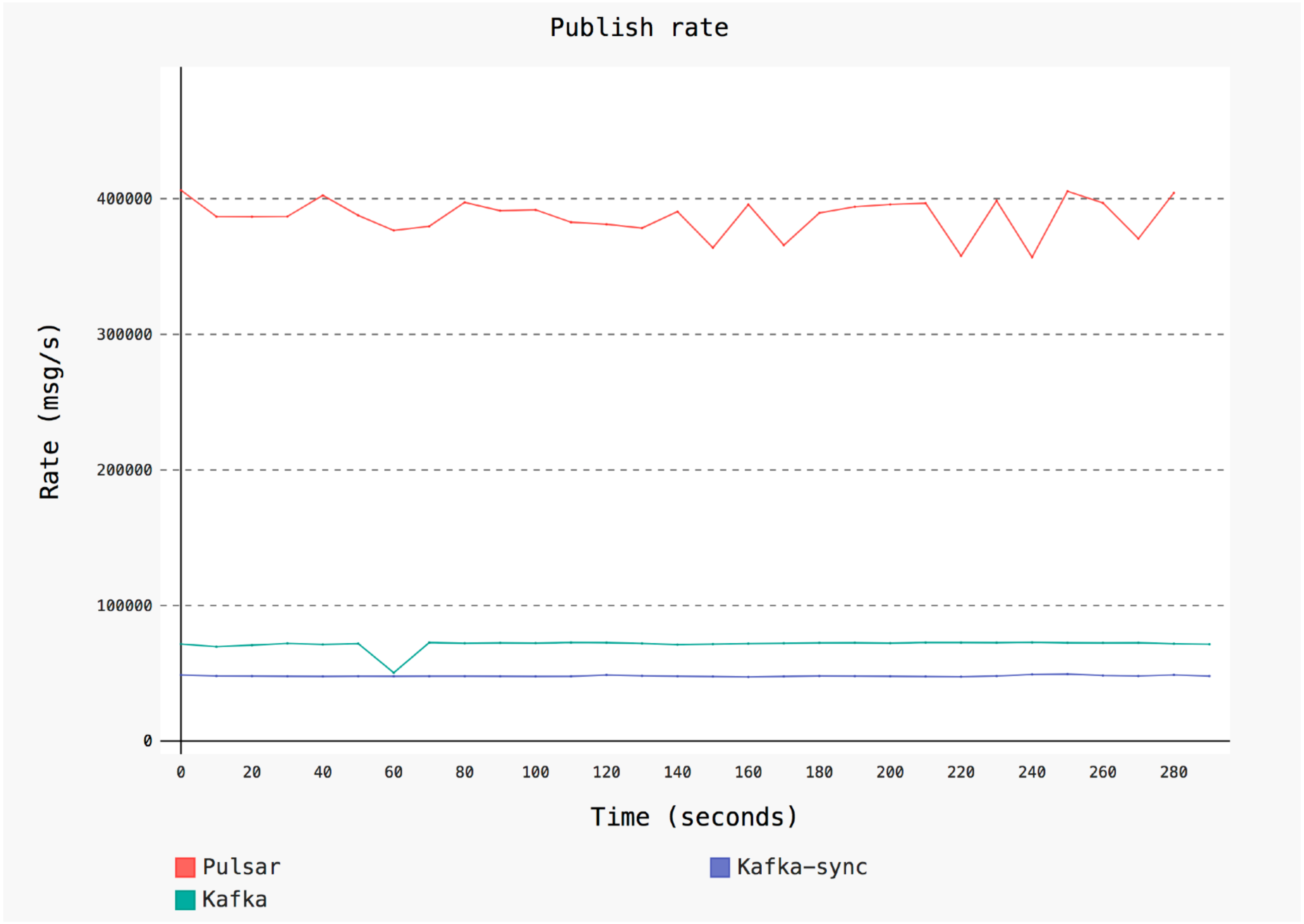
Conclusion

- Unbounded topic partition storage
- Instant scaling without data rebalance
    - Seamless - broker failure recovery
    - Seamless- bookie failure recovery
    - Seamless - cluster expansion
- Independent scalability

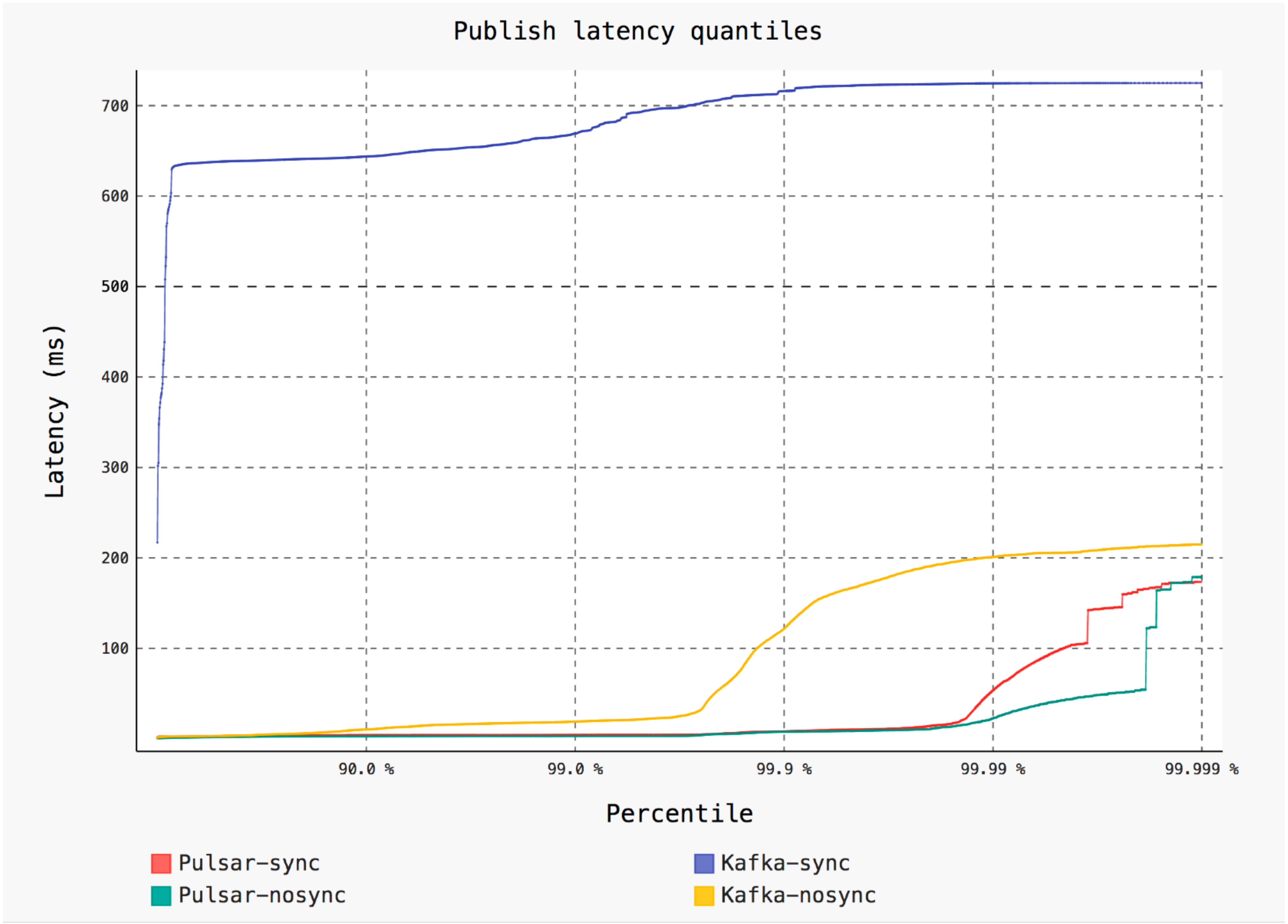
Benchmark

https://github.com/openmessaging/openmessaging-benchmark

Throughput


Publish rate
Rate (msg/s)
400000
300000
200000
100000
0
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
Time (seconds)
Pulsar
Kafka
Kafka-sync

Latency



Publish latency quantiles
Latency (ms)
700
600
500
400
300
200
100
90.0 %
99.0 %
99.9 %
99.99 %
99.999 %
Percentile
🟥 Pulsar-sync
🟩 Pulsar-nosync
🟦 Kafka-sync
🟨 Kafka-nosync

Pulsar Functions

Pulsar Functions

- Lightweight stream processing
- New in Pulsar 2.0
- Currently supports Java and Python

Python

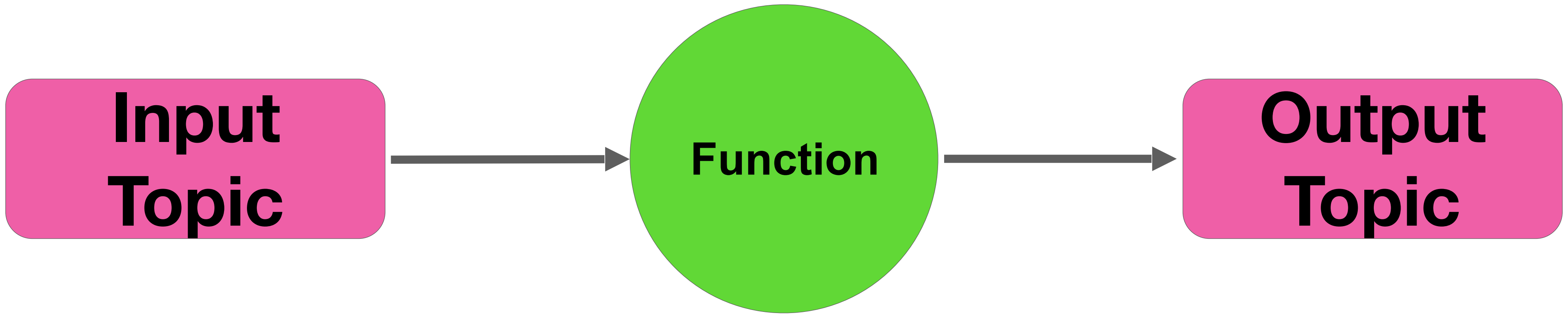
```python
def process(input):
    return input.replace("jia", "anonymous")
```

Java

```java
import java.util.function.Function;

public class Anon implements Function<String,String> {
    @Override
    public String apply(String input) {
        return input.replace("jia", "anonymous");
    }
}
```

Pulsar Functions



Python

```
# pulsar-admin functions create \
    —py anon.py --className anon \
    --fqfn lc3-tenant/demo/anony \
    --inputs persistent://lc3-tenant/demo/input \
    --output persistent://lc3-tenant/demo/output
```

Java

```
# pulsar-admin functions create \
    —jar anon.jar --className Anon \
    --fqfn lc3-tenant/demo/anony \
    --inputs persistent://lc3-tenant/demo/input \
    --output persistent://lc3-tenant/demo/output
```

Curious to Get More

- Apache Pulsar : http://pulsar.incubator.apache.org
- Apache BookKeeper : http://bookkeeper.apache.org
- Technical Blog : https://streaml.io/blog/
- Twitter: @apache_pulsar @asfbookkeeper
- slack:
    - https://apache-pulsar.herokuapp.com/
    - https://apachebookkeeper.herokuapp.com/

streamlio

Thanks!