系统架构——分层 + 分片

演讲者 / streamlio 翟佳

Apache Pulsar特性

Durability
Data replicated and synced to disk

Ordering
Guaranteed ordering

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

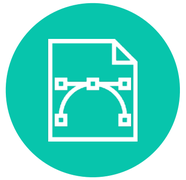
High throughput
Can reach 1.8 M messages/s in a single partition

Low Latency
Low publish latency of 5ms at 99pct

Unified messaging model
Support both Streaming and Queuing

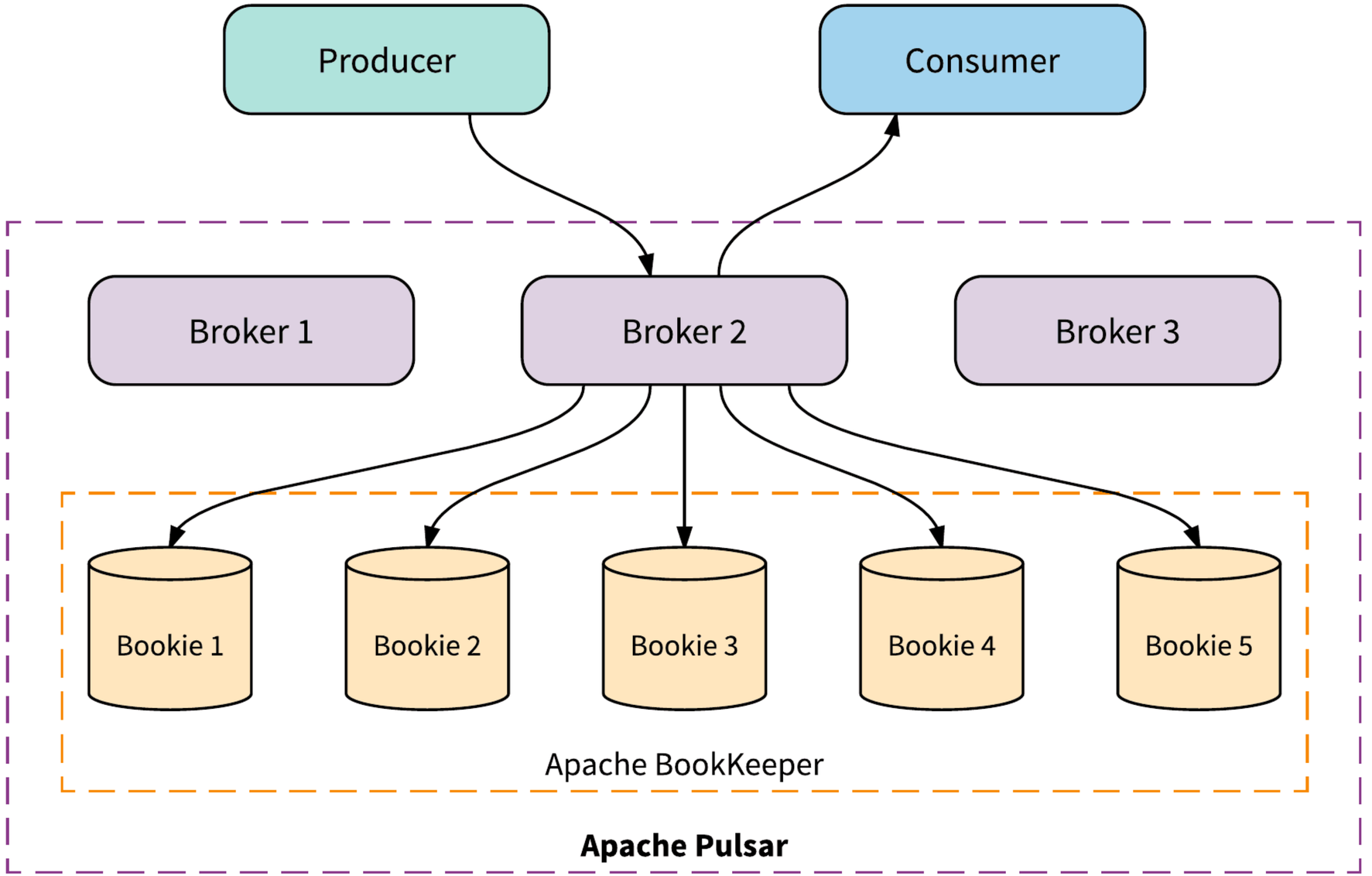
Multi-tenancy
A single cluster can support many tenants and use cases

Geo-replication
Out of box support for geographically distributed applications



Highly scalable & available
Can support millions of topics

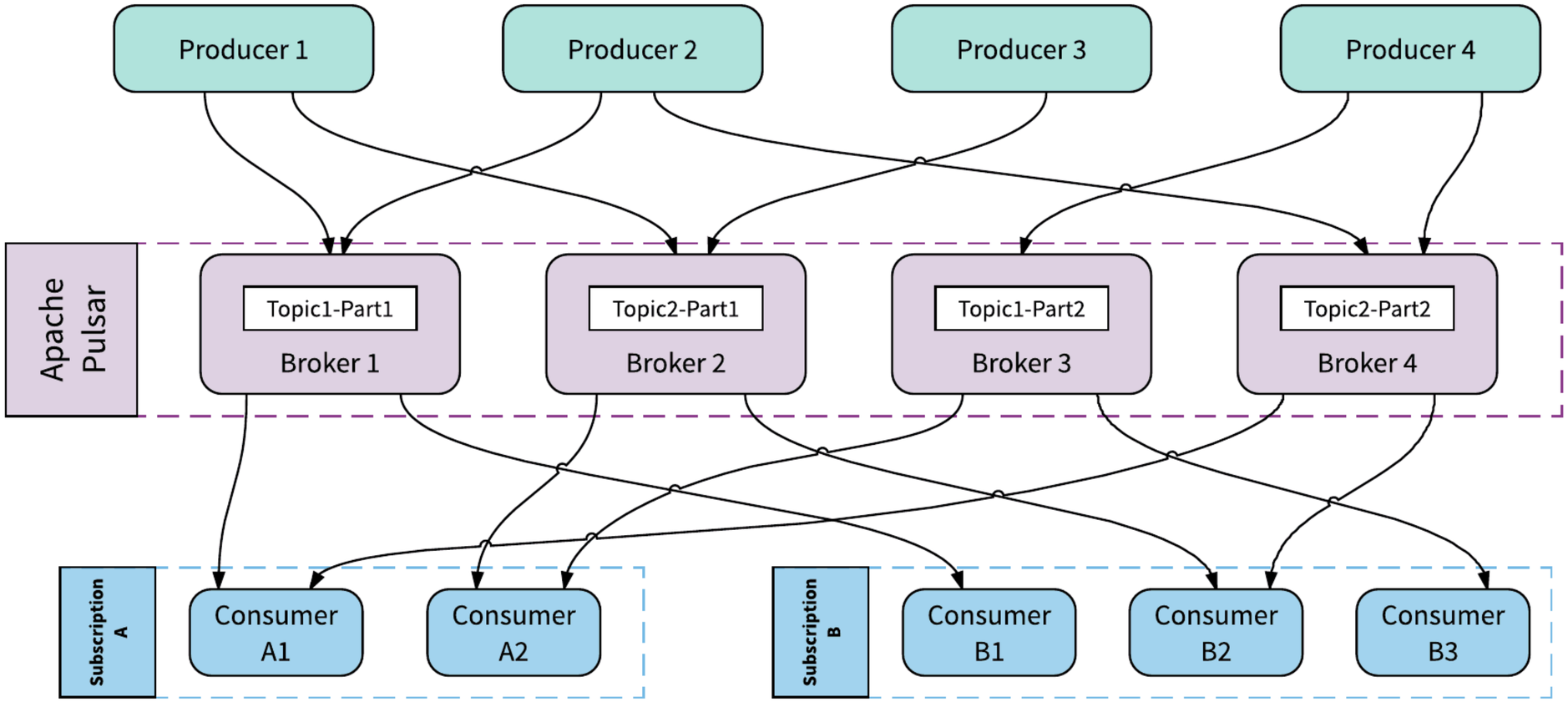
分层架构



分层架构:
Brokers & Bookies

- 独立扩展
- 灵活容错
- 快速扩容

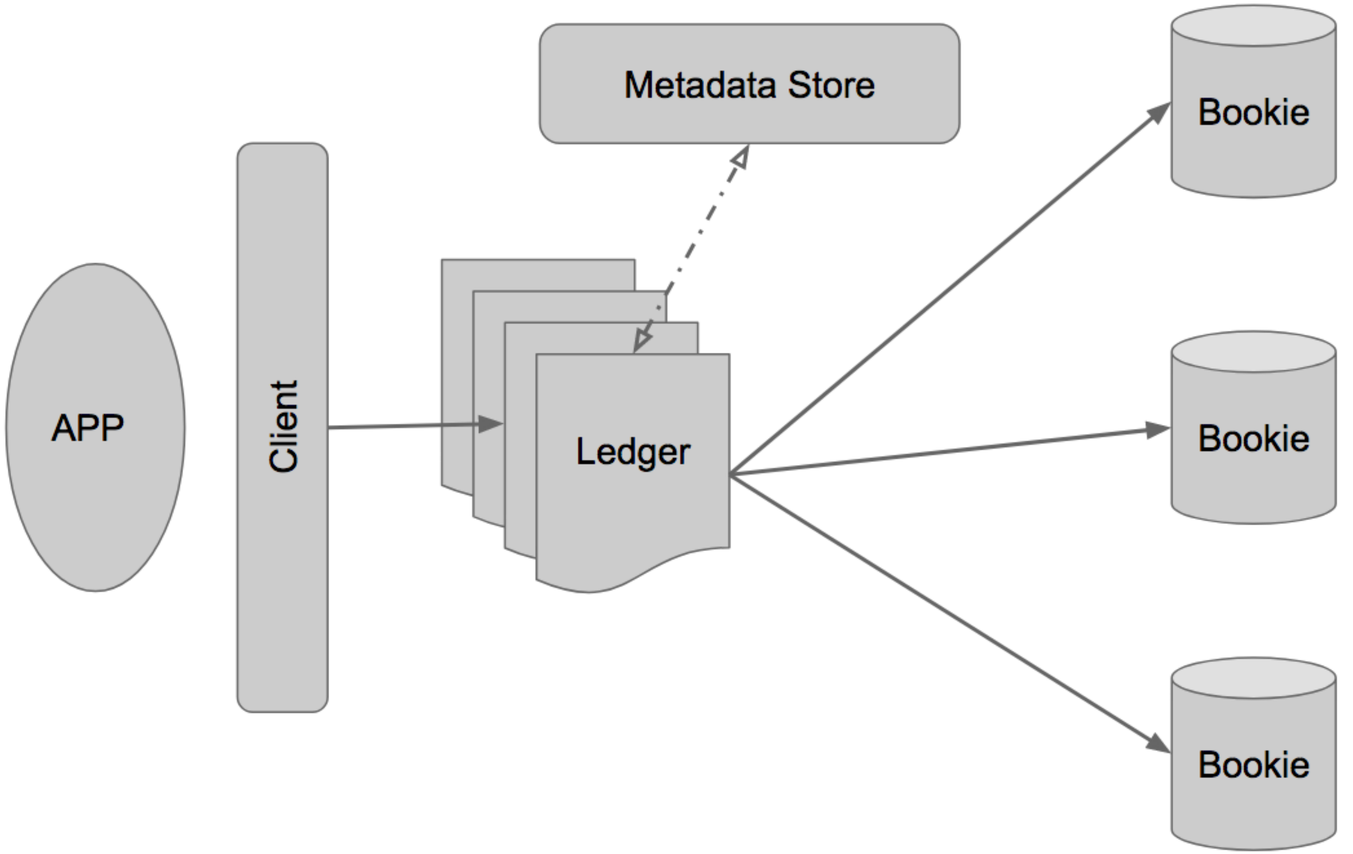
分层架构 — Brokers



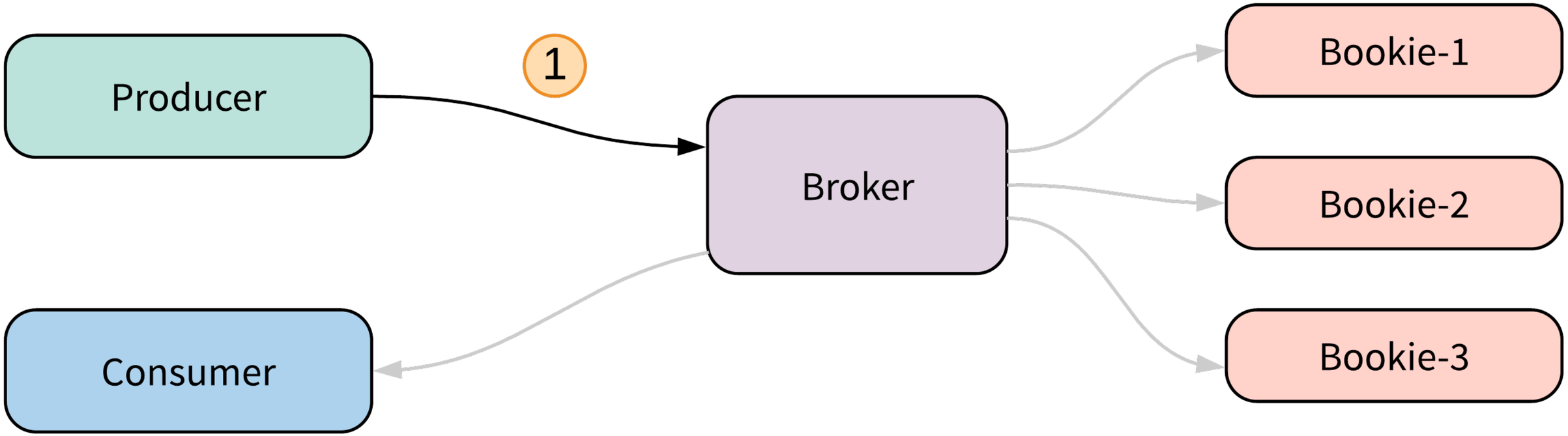
分层架构 — BookKeeper

BookKeeper分布式日志/流存储

- 低延时、高吞吐、持久化
- 强一致 (repeatable read consistency)
- 高可用
- 单节点可以存储很多日志
- I/O隔离

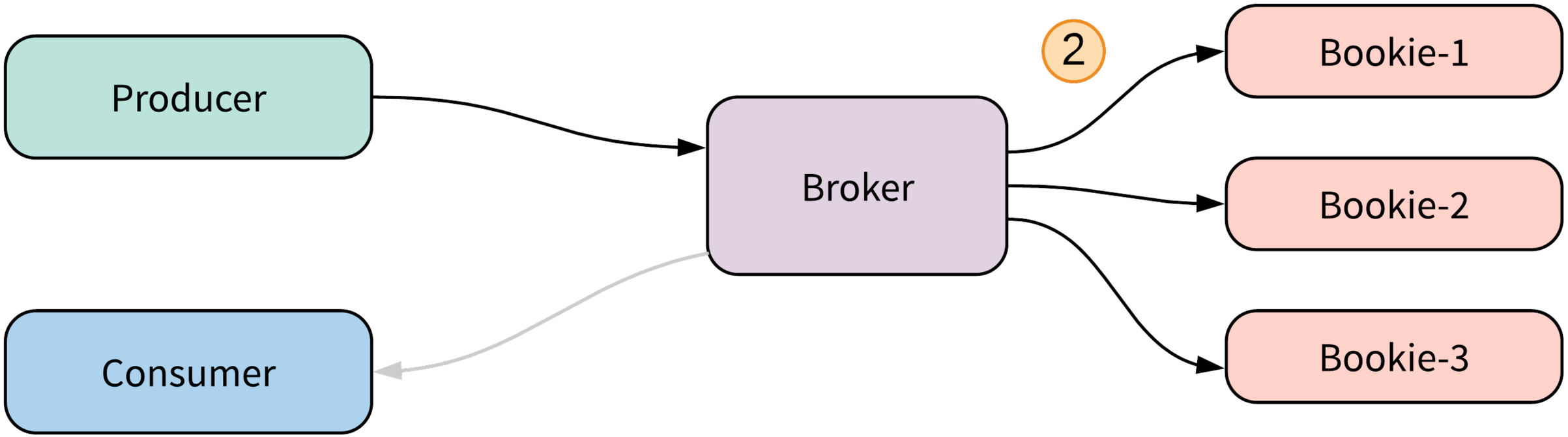


读写流程 (1)



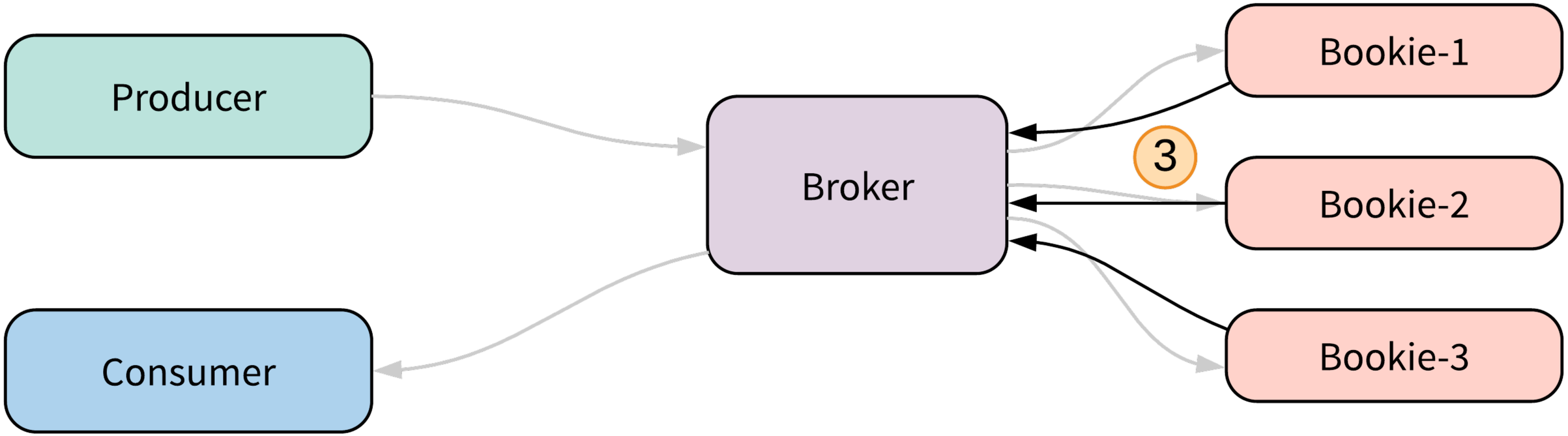
(1) 生产者发送消息给Broker

读写流程 (2)



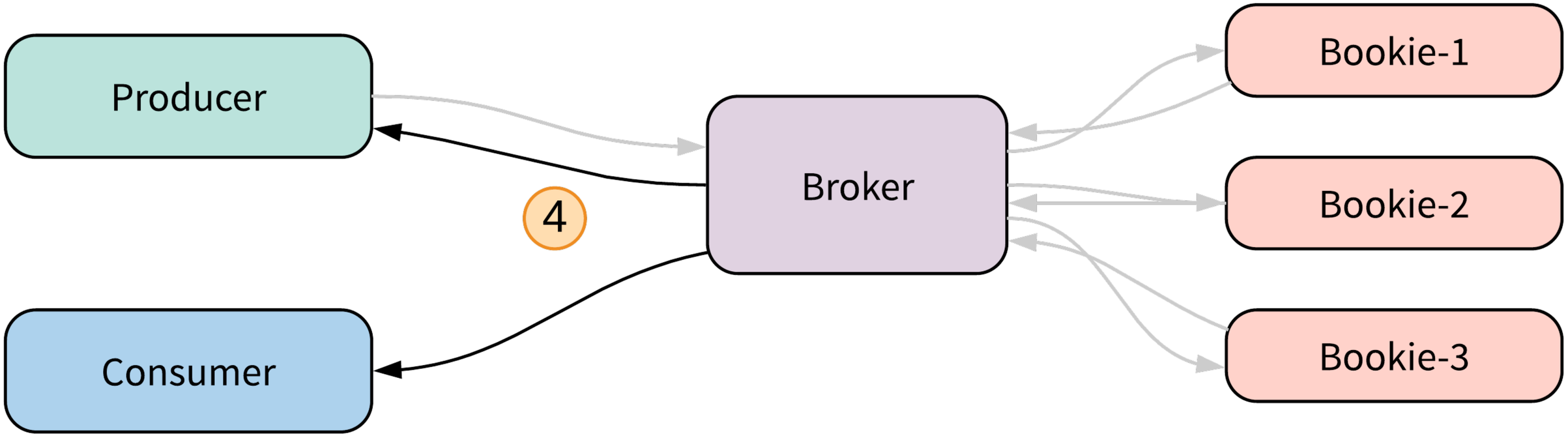
(2) Broker并发写N个副本

读写流程 (3)



(3) Broker等待来自于Bookies的Quorum Acks

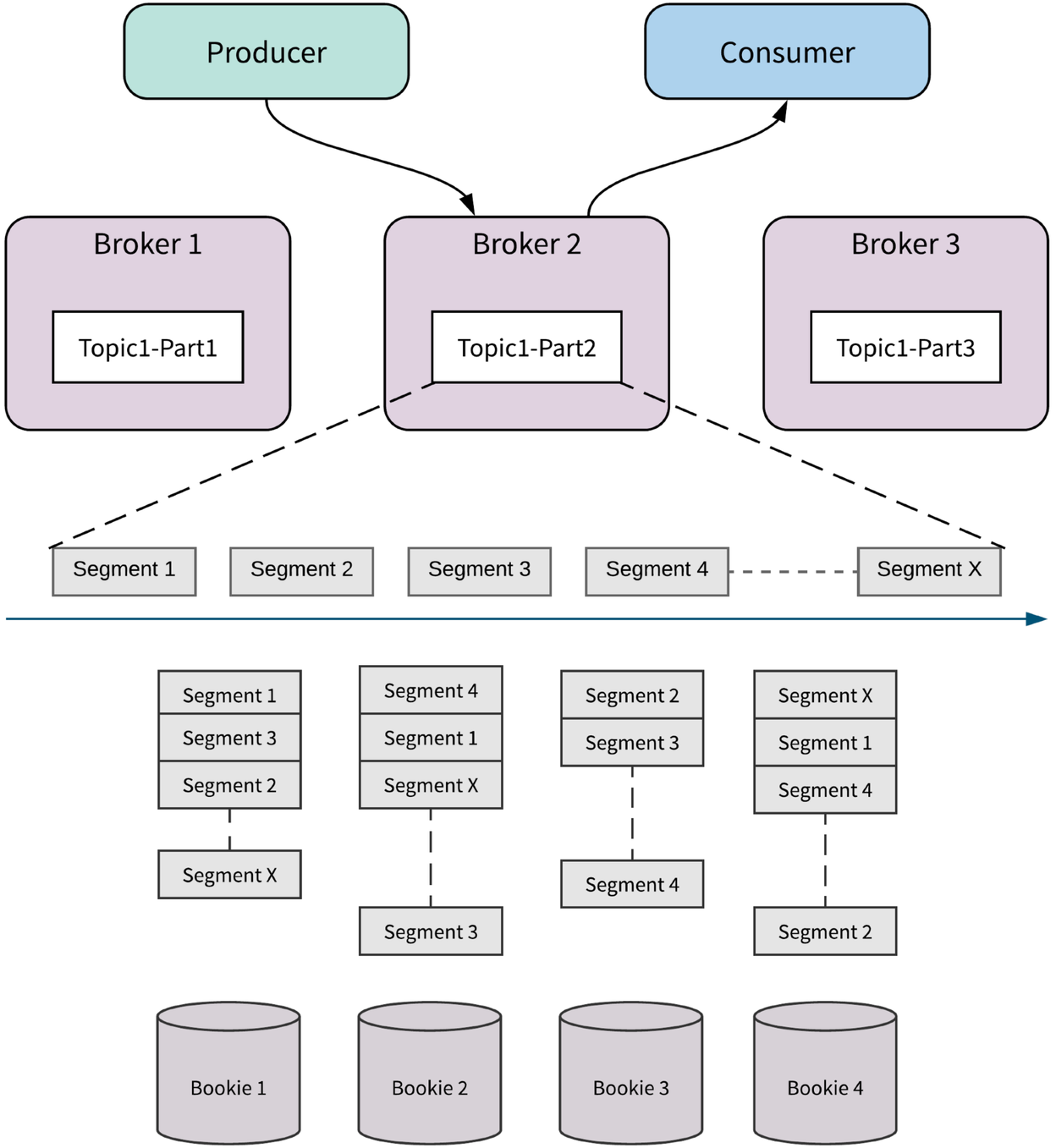
读写流程 (4)



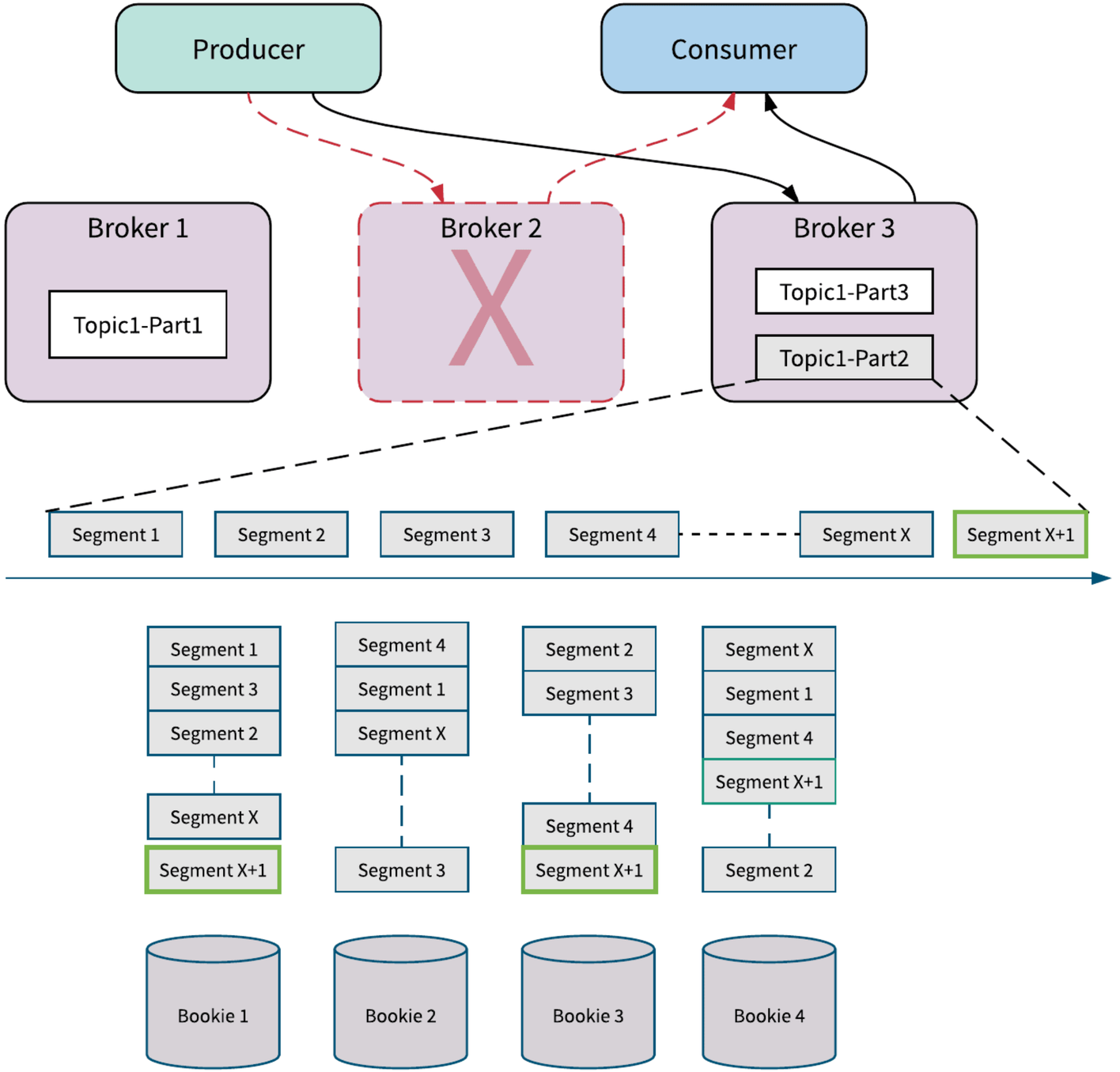
(4) Broker返回确认给生产者，并投递给消费者

分片存储

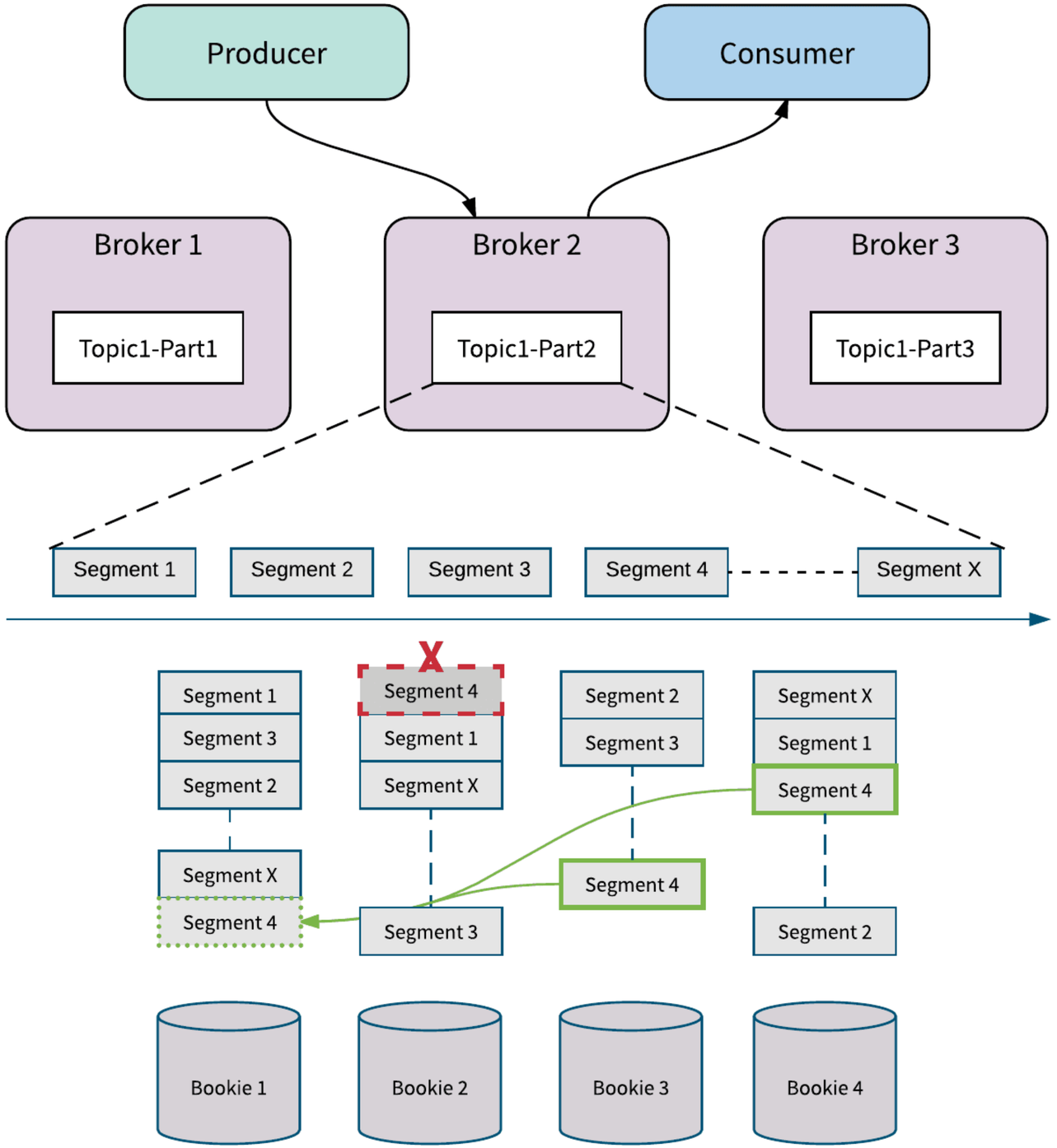
- 分区(Partition)是逻辑上的一个概念
- 分区按照时间或者大小被切成分片(Segment)
- 分片被打散存放到集群中的所有节点



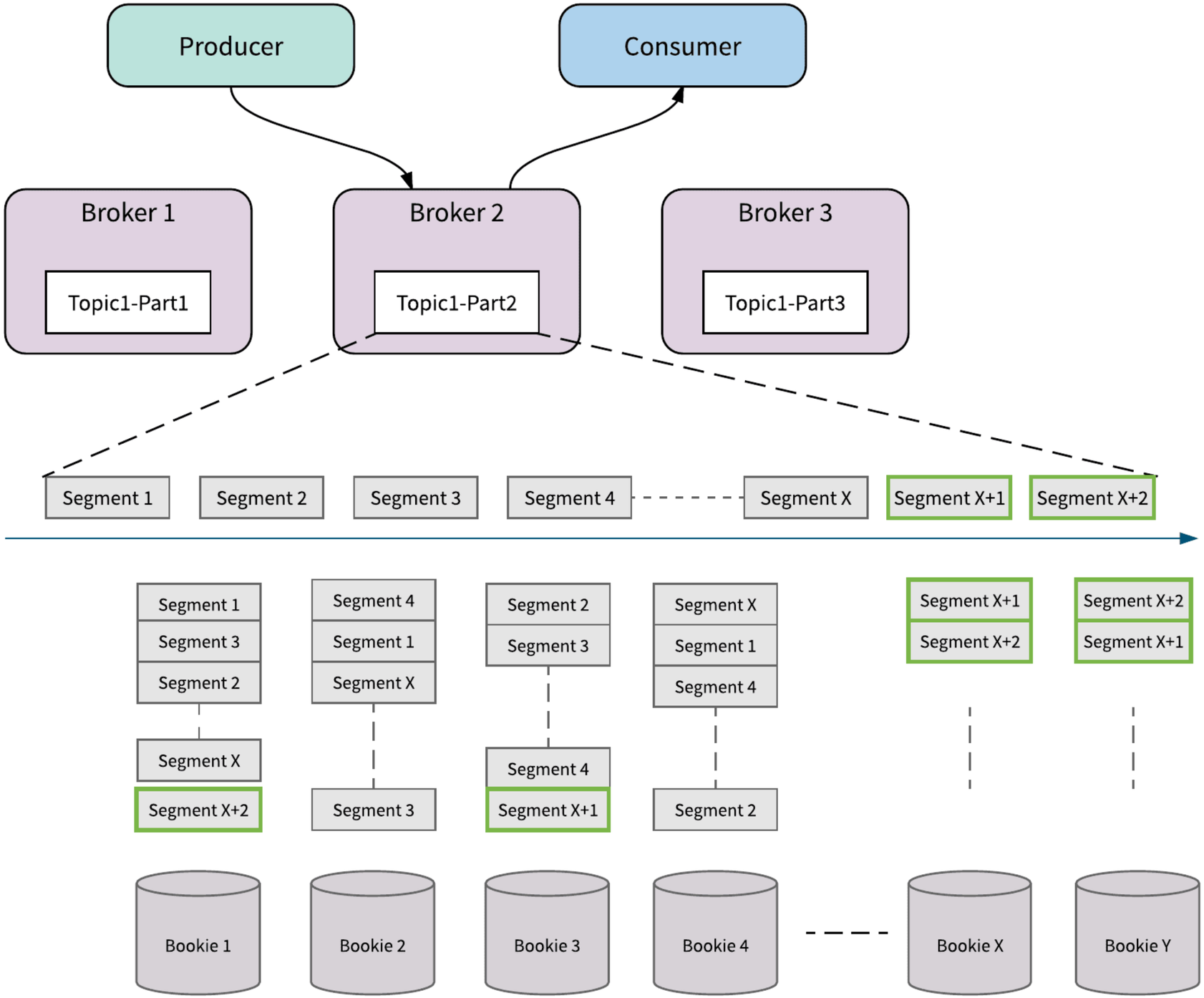
Broker容错


Producer
Consumer
Broker 1
Topic1-Part1
Broker 2
X
Broker 3
Topic1-Part3
Topic1-Part2
Segment 1
Segment 2
Segment 3
Segment 4
Segment X
Segment X+1
Segment 1
Segment 3
Segment 2
Segment X
Segment X+1
Segment 4
Segment 1
Segment X
Segment 3
Segment 2
Segment 3
Segment 4
Segment X+1
Segment X
Segment 1
Segment 4
Segment X+1
Segment 2
Bookie 1
Bookie 2
Bookie 3
Bookie 4

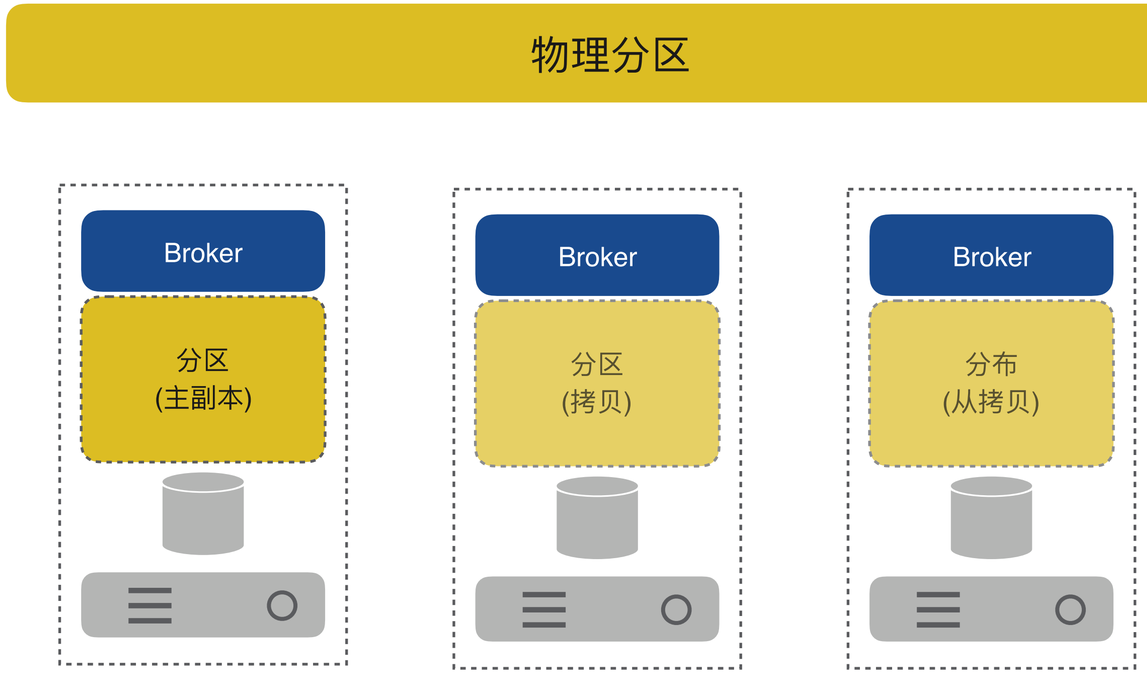
Bookie容错


Producer
Consumer
Broker 1
Topic1-Part1
Broker 2
Topic1-Part2
Broker 3
Topic1-Part3
Segment 1
Segment 2
Segment 3
Segment 4
Segment X
X
Segment 1
Segment 3
Segment 2
Segment X
Segment 4
Segment 4
Segment 1
Segment X
Segment 3
Segment 2
Segment 3
Segment X
Segment 1
Segment 4
Segment 4
Segment 3
Segment 2
Bookie 1
Bookie 2
Bookie 3
Bookie 4

扩容

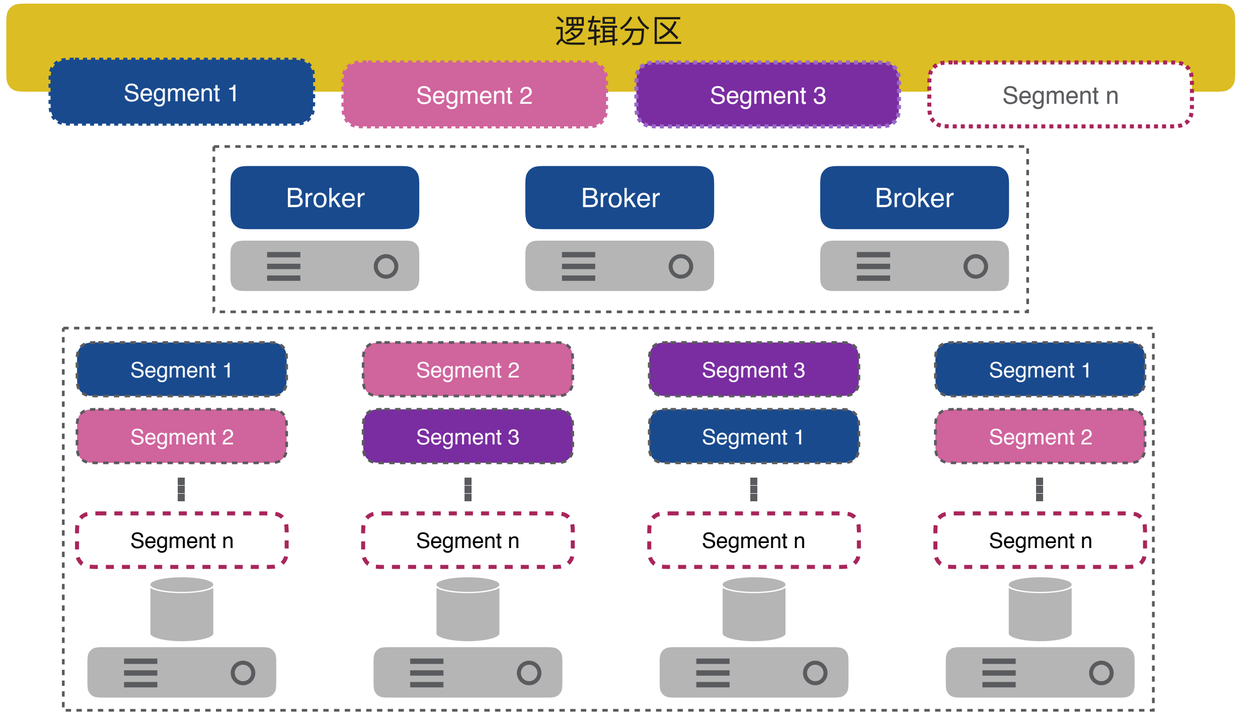

Producer
Consumer
Broker 1
Topic1-Part1
Broker 2
Topic1-Part2
Broker 3
Topic1-Part3
Segment 1
Segment 2
Segment 3
Segment 4
Segment X
Segment X+1
Segment X+2
Segment 1
Segment 3
Segment 2
Segment X
Segment X+2
Segment 4
Segment 1
Segment X
Segment 3
Segment 2
Segment 3
Segment 4
Segment X+1
Segment X
Segment 1
Segment 4
Segment 2
Segment X+1
Segment X+2
Segment X+2
Segment X+1
Bookie 1
Bookie 2
Bookie 3
Bookie 4
Bookie X
Bookie Y

分区 vs 分片



分区架构

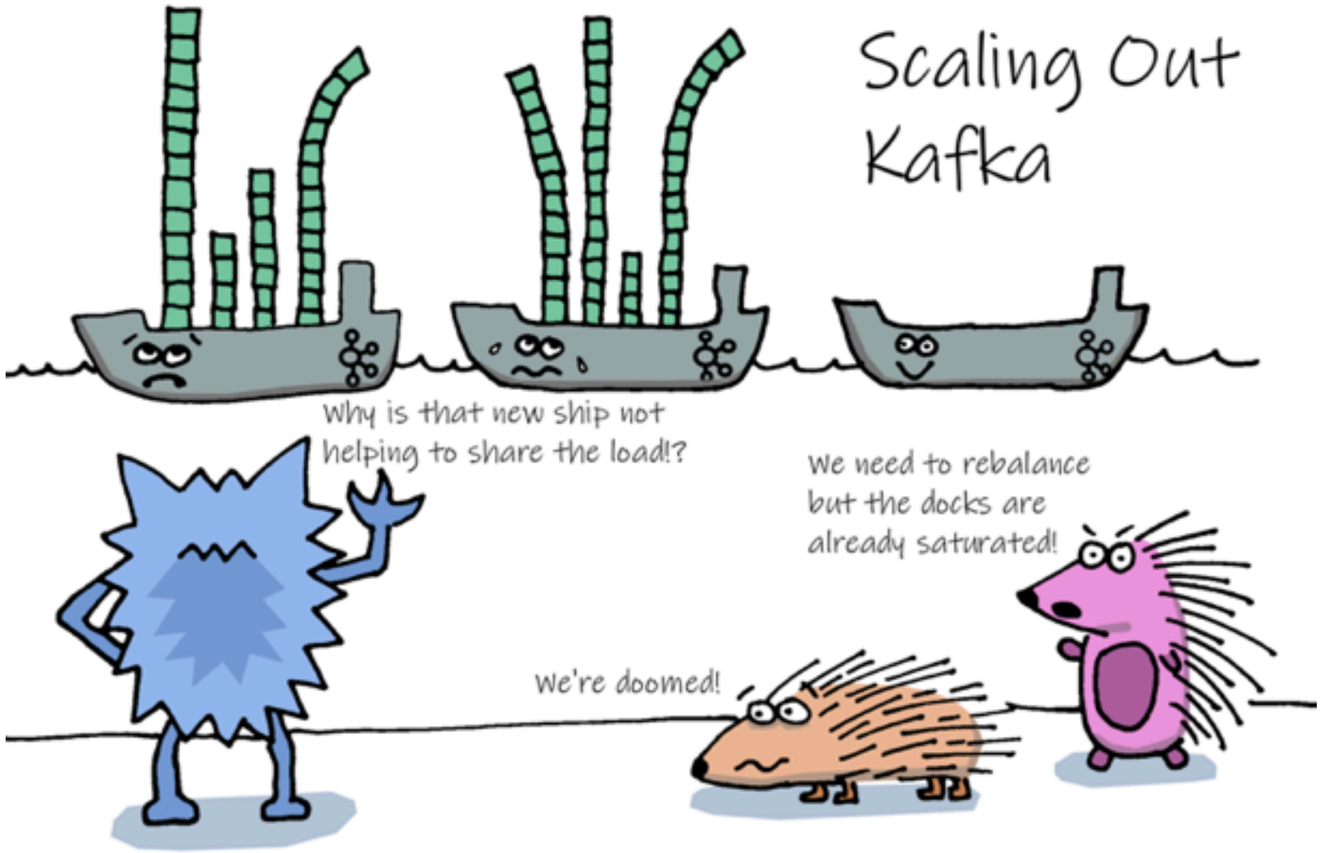
- 物理分区
- 存储和计算紧耦合
- 容错恢复需要拷贝物理分区
- 扩容需要迁移物理分区来达到负载均衡

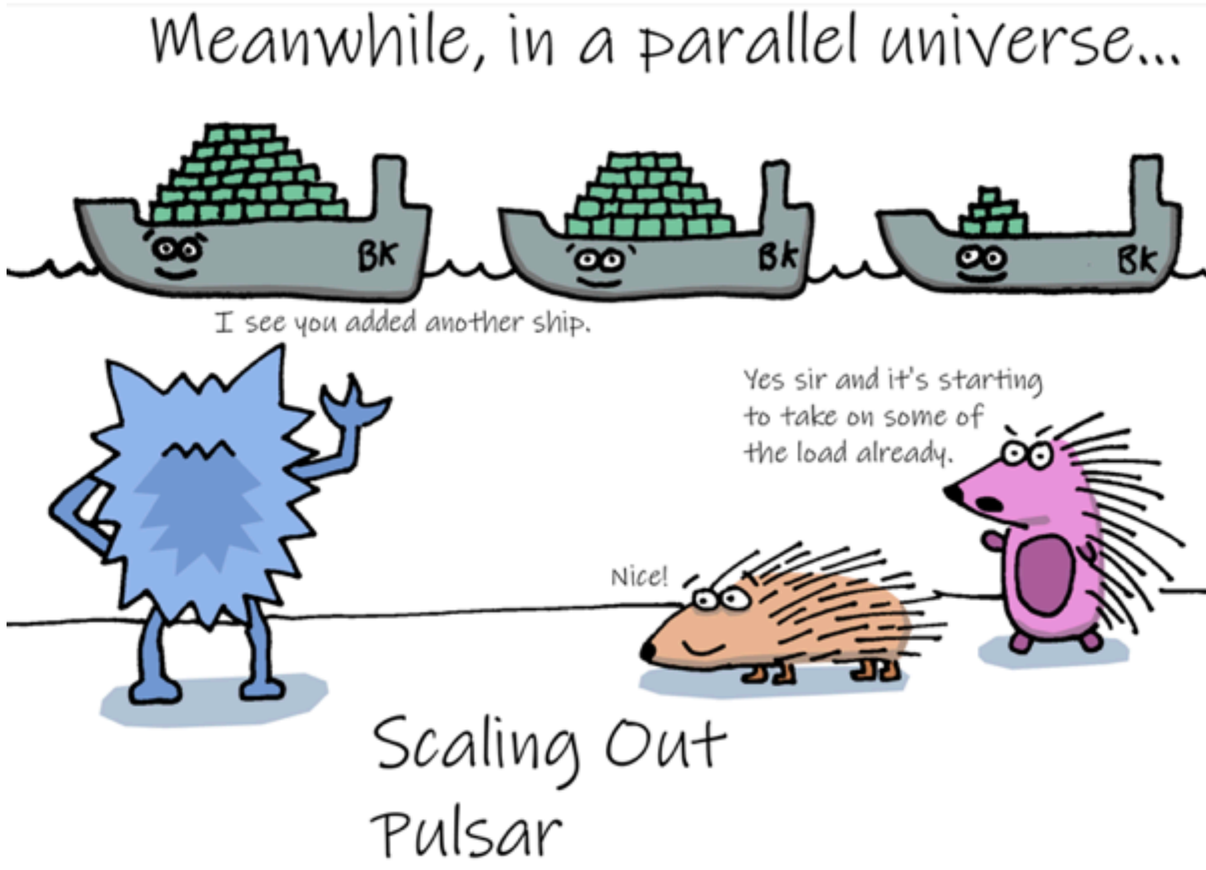


分片架构

- 逻辑分区，“物理”分片
- 存储和计算分离
- 失效处理相互分离，快速、无痛点
- 弹性扩容

分区 vs 分片





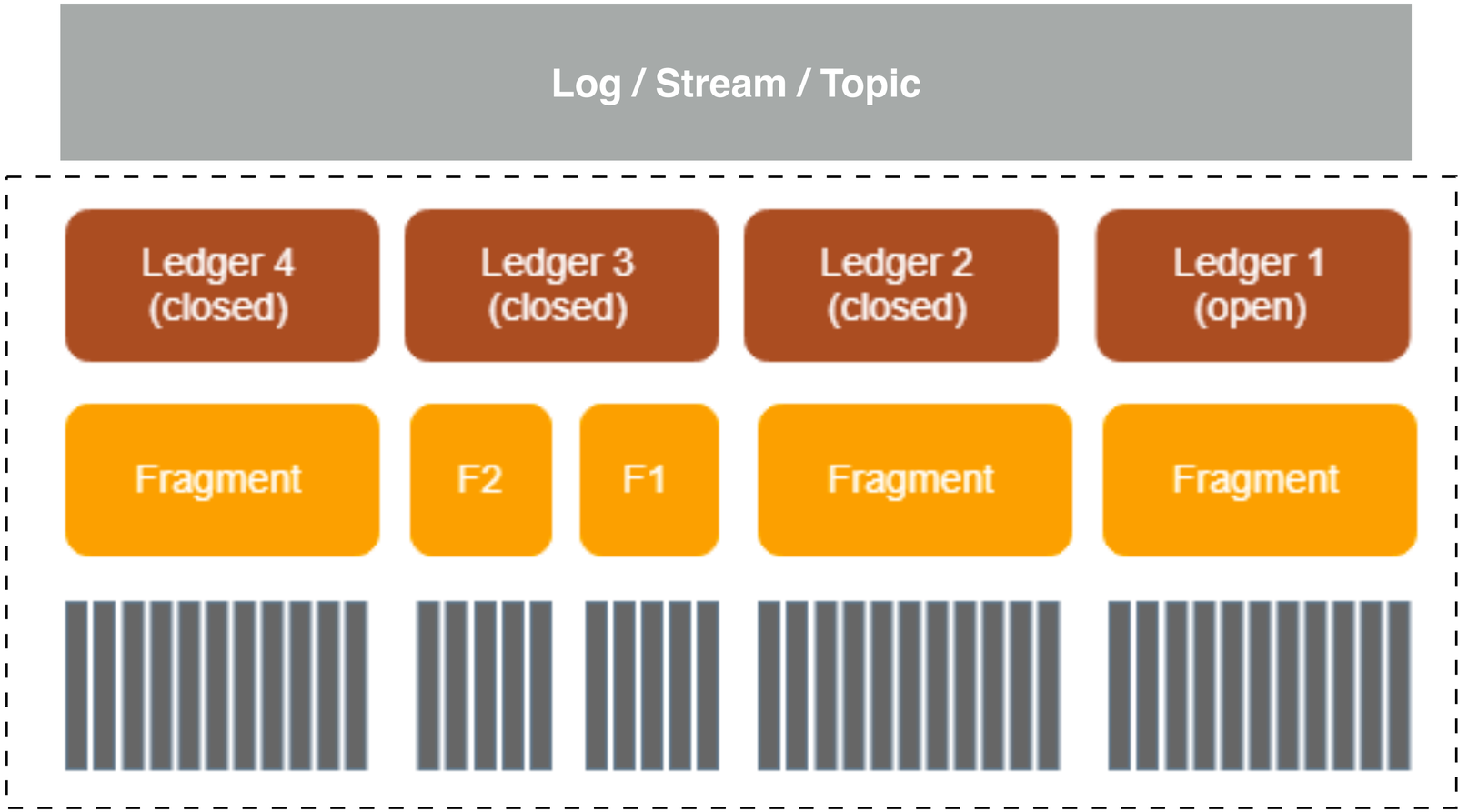
https://jack-vanlightly.com/sketches/2018/10/2/kafka-vs-pulsar-rebalancing-sketch

Apache BookKeeper

分布式日志/流存储

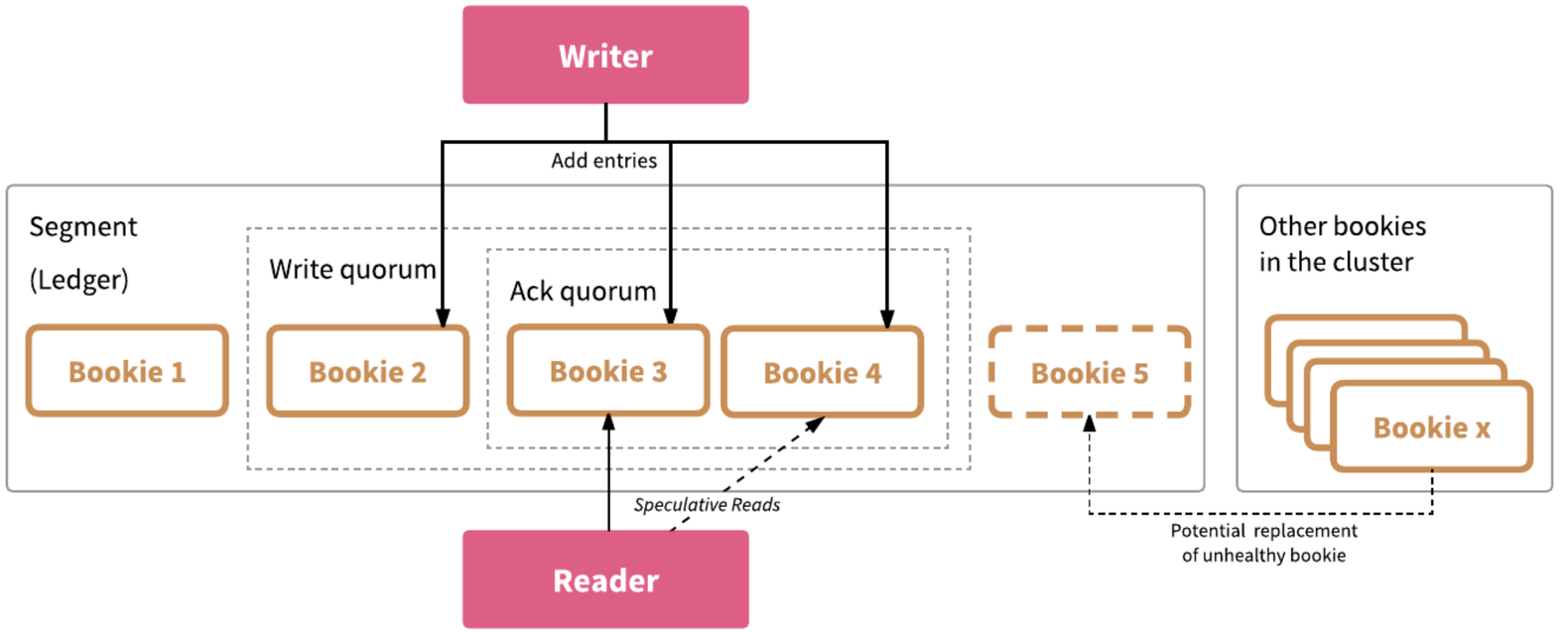
- 低延时多复本复制:
  - Quorum Parallel Replication
- 持久化: 所有操作保证刷盘后才进行ACK
- 强一致性:
  - 可重复读的一致性 (Repeatable Read Consistency)
- 读写高可用
- 存储节点的读写隔离

概念： Entry, Ledger & Log

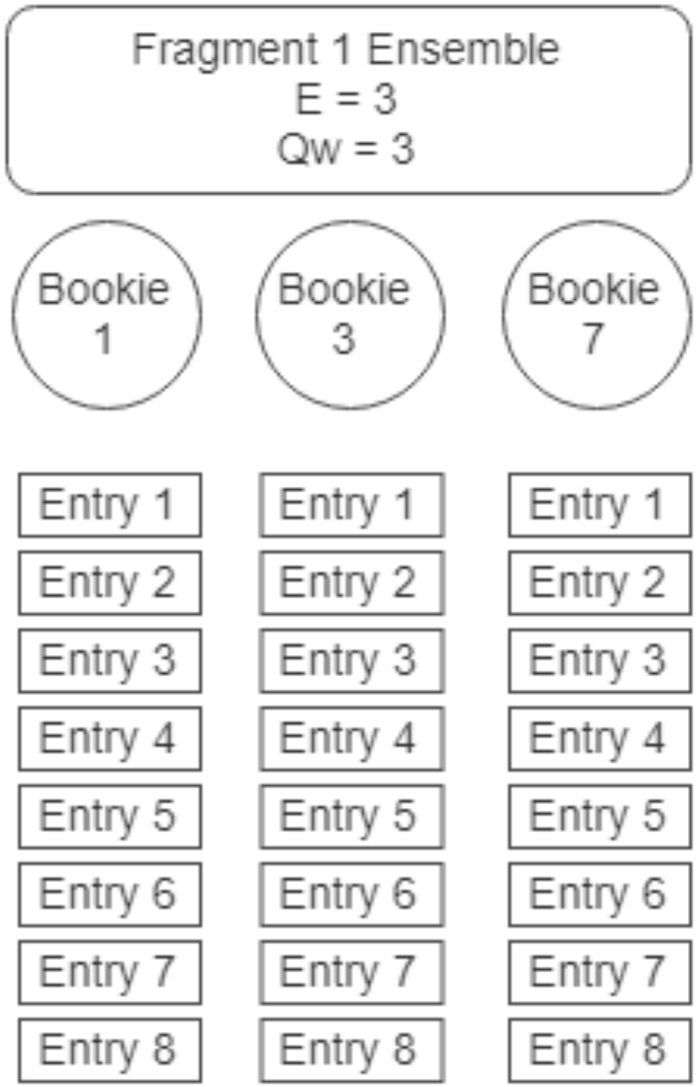


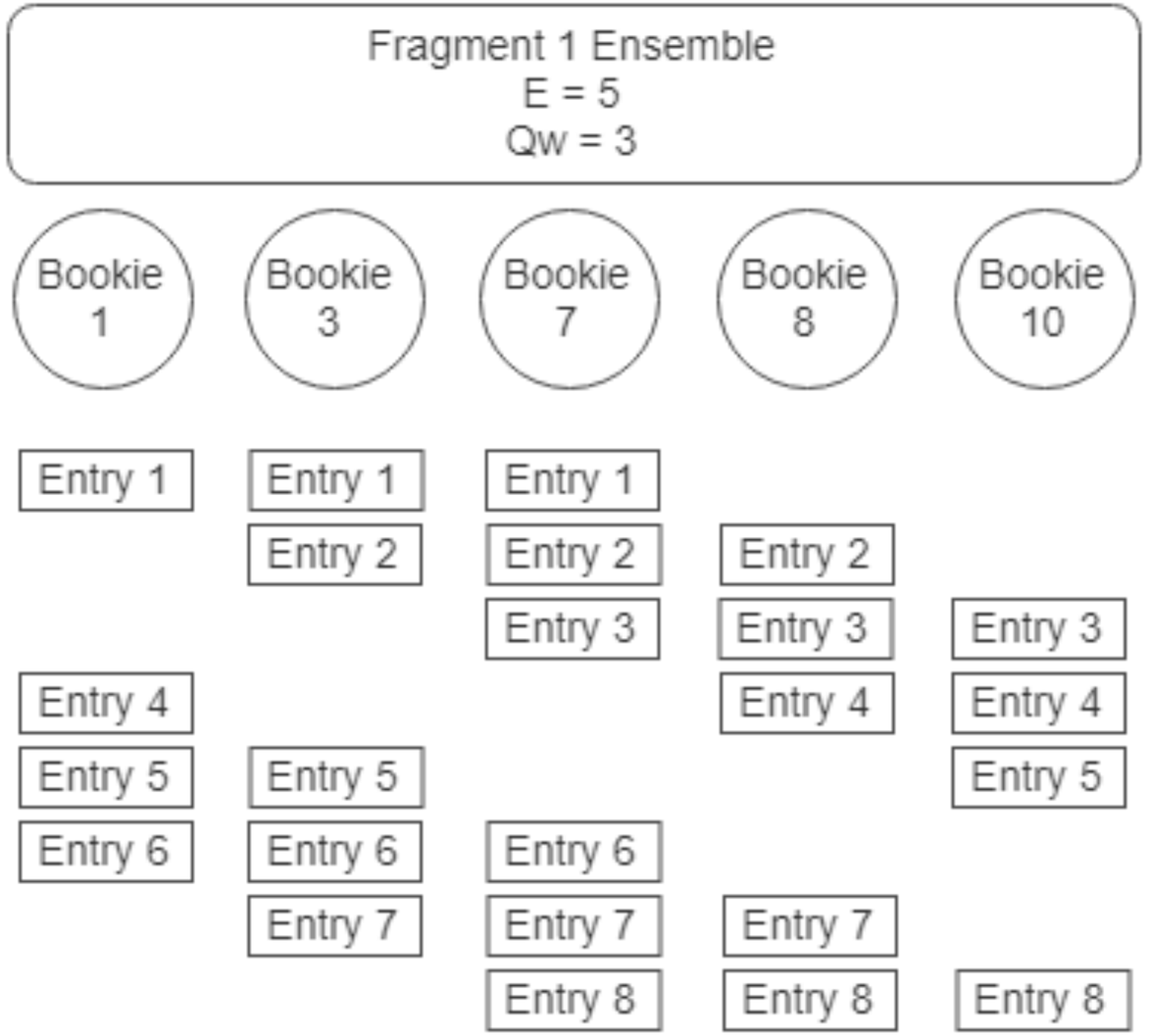
多副本复制(1)

- Parallel Quorum Replication
- Ensemble
- Write Quorum
- Ack Quorum



多副本复制(2)


Fragment 1 Ensemble
E = 3
Qw = 3
Bookie 1
Bookie 3
Bookie 7
Entry 1
Entry 2
Entry 3
Entry 4
Entry 5
Entry 6
Entry 7
Entry 8
Entry 1
Entry 2
Entry 3
Entry 4
Entry 5
Entry 6
Entry 7
Entry 8
Entry 1
Entry 2
Entry 3
Entry 4
Entry 5
Entry 6
Entry 7
Entry 8

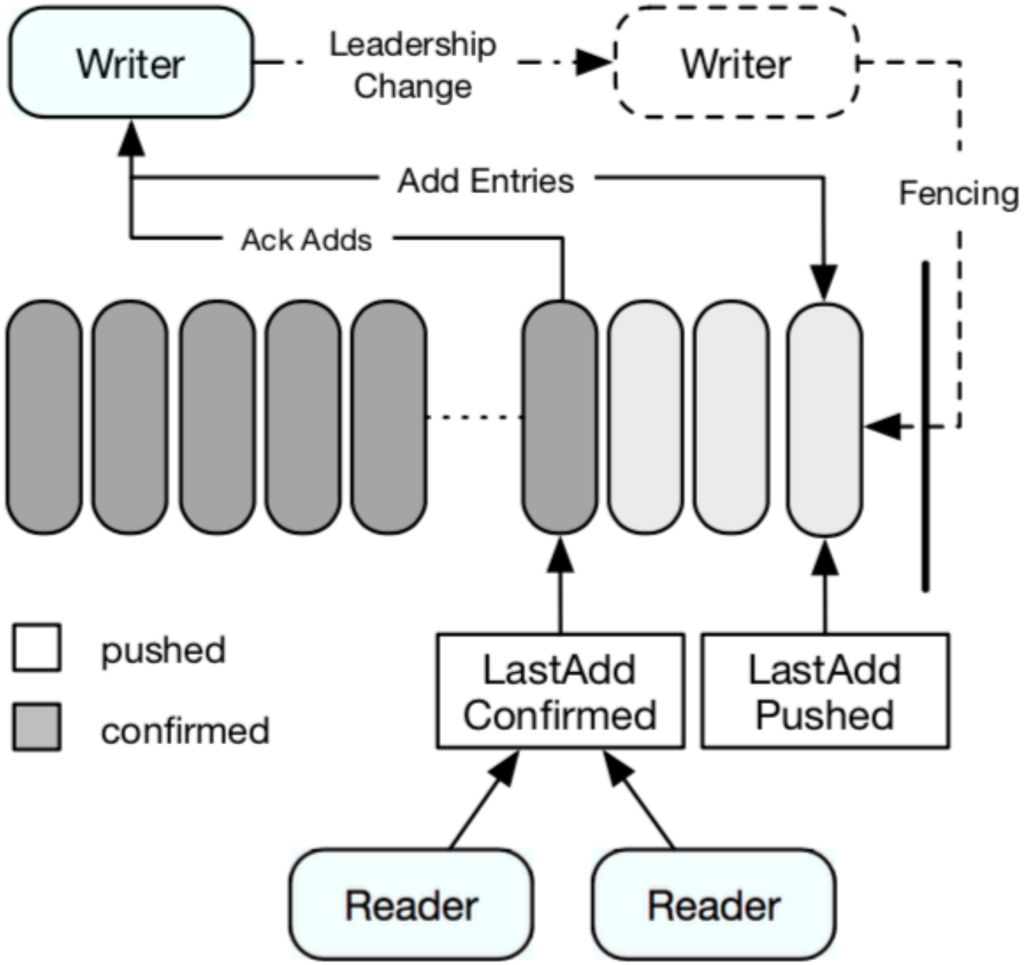

Fragment 1 Ensemble
E = 5
Qw = 3
Bookie 1
Bookie 3
Bookie 7
Bookie 8
Bookie 10
Entry 1
Entry 4
Entry 5
Entry 6
Entry 1
Entry 2
Entry 5
Entry 6
Entry 7
Entry 1
Entry 2
Entry 3
Entry 6
Entry 7
Entry 8
Entry 2
Entry 3
Entry 4
Entry 7
Entry 8
Entry 3
Entry 4
Entry 5
Entry 8

多副本复制(3)

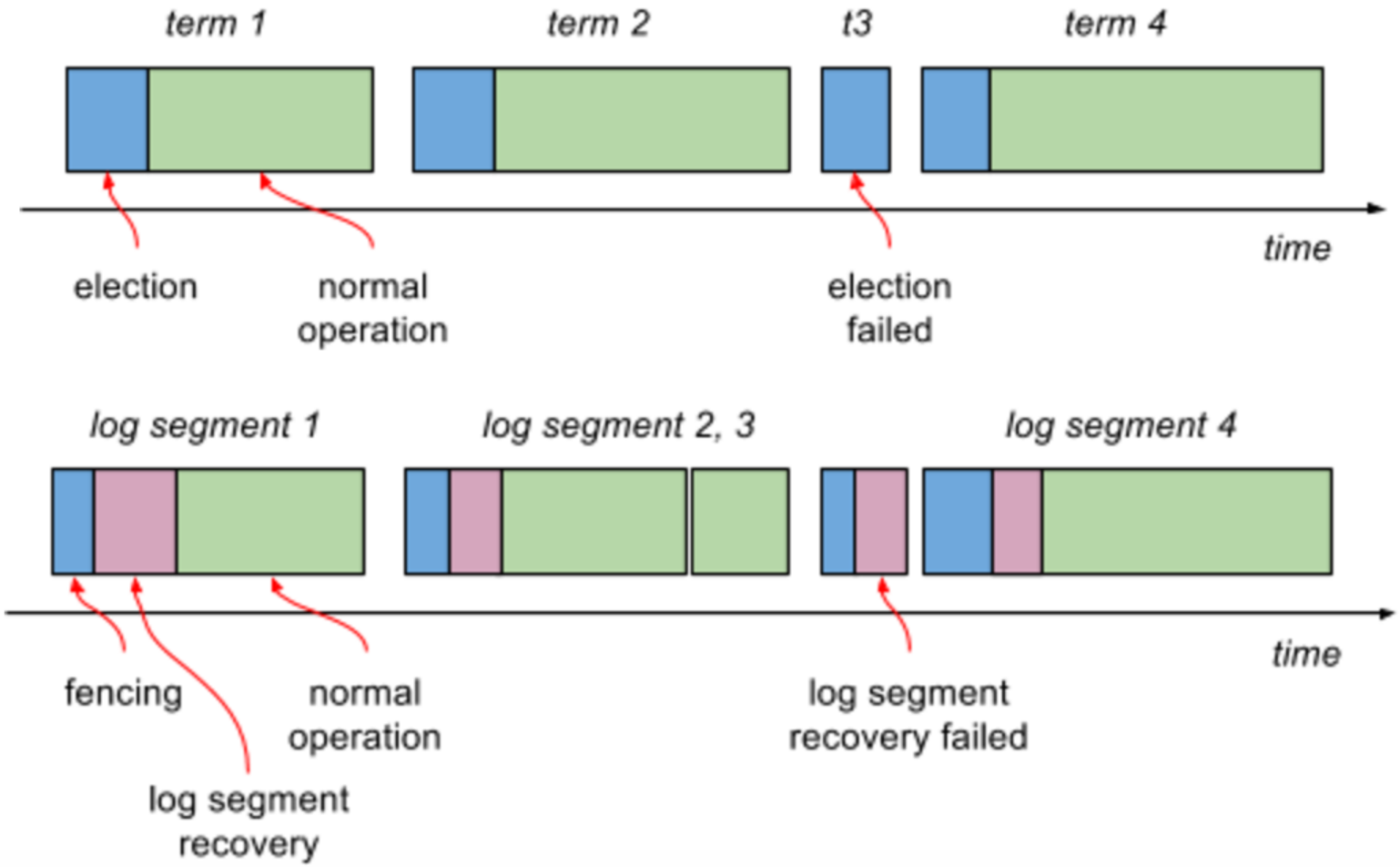
- Ensemble: 控制一个Ledger的读写带宽
- Write Quorum：控制一条记录的复本数量
- Ack Quorum：写每条记录需要等待的Ack数量，控制时延
- 灵活性：
  - 增加Ensemble，可以增加读写带宽
  - 减少Ack Quorum，可以减少长尾时延

一致性(1)–可重复读

- LastAddPushed
- LastAddConfirmed
- Fencing避免脑裂



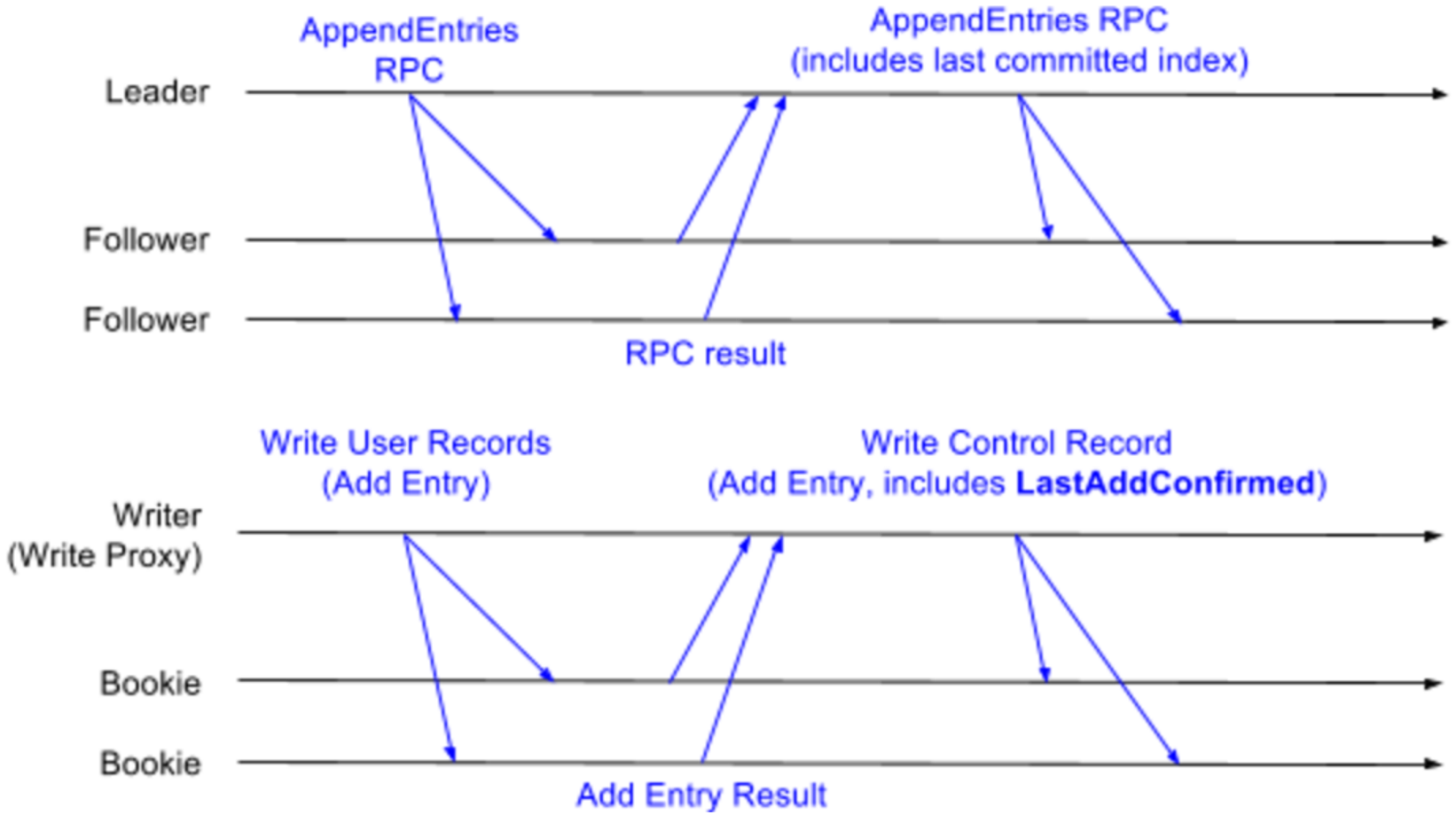
一致性(2)


term 1
term 2
t3
term 4
time
election
normal operation
election failed
log segment 1
log segment 2, 3
log segment 4
time
fencing
normal operation
log segment recovery
log segment recovery failed

Raft

BK

一致性(3)

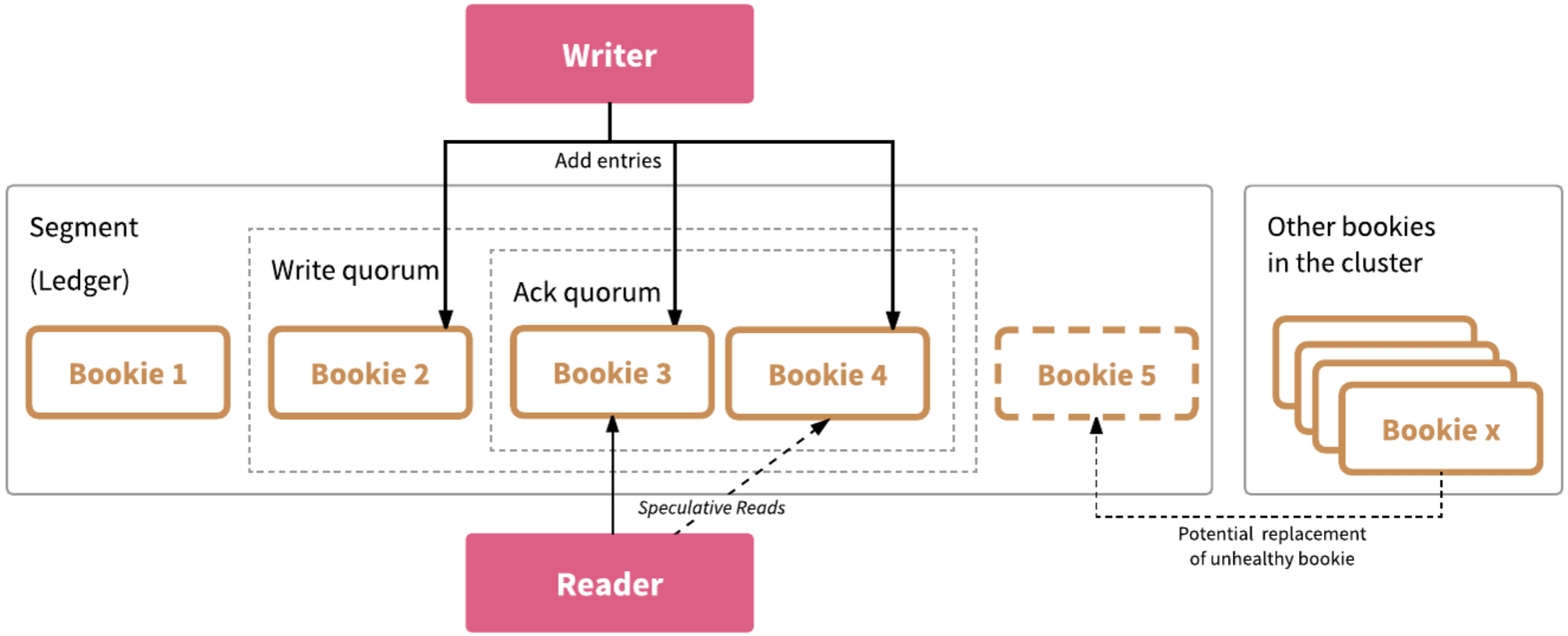

AppendEntries
RPC
AppendEntries RPC
(includes last committed index)
Leader
Follower
Follower
RPC result
Write User Records
(Add Entry)
Write Control Record
(Add Entry, includes **LastAddConfirmed**)
Writer
(Write Proxy)
Bookie
Bookie
Add Entry Result

Raft

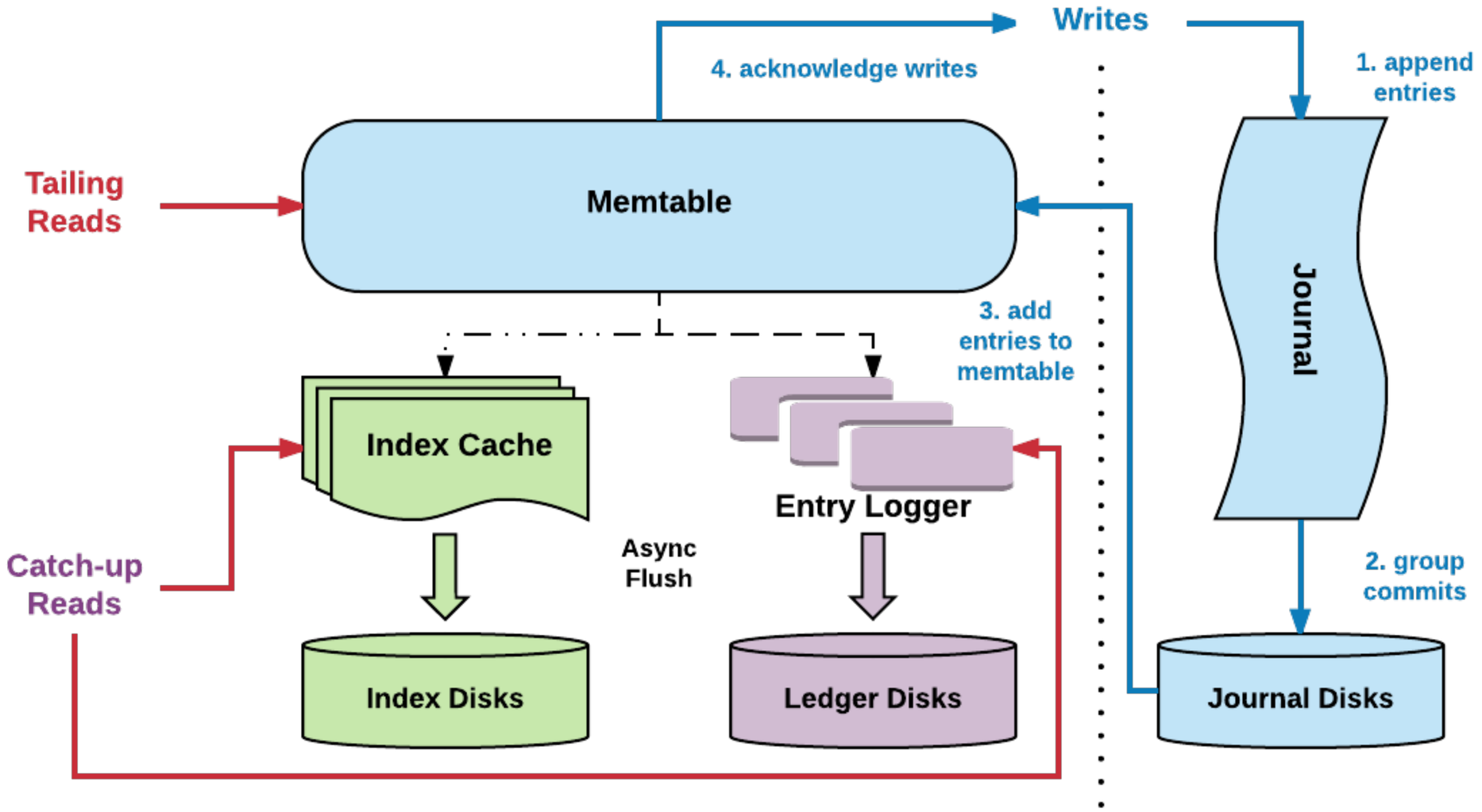
BK

读写高可用

- 写高可用 - Ensemble Change
  - 最大化数据放置可能性
- 读高可用 - Speculative Reads
  - 没有主节点
  - 每个副本都可以提供读
  - 通过Speculative减少长尾时延



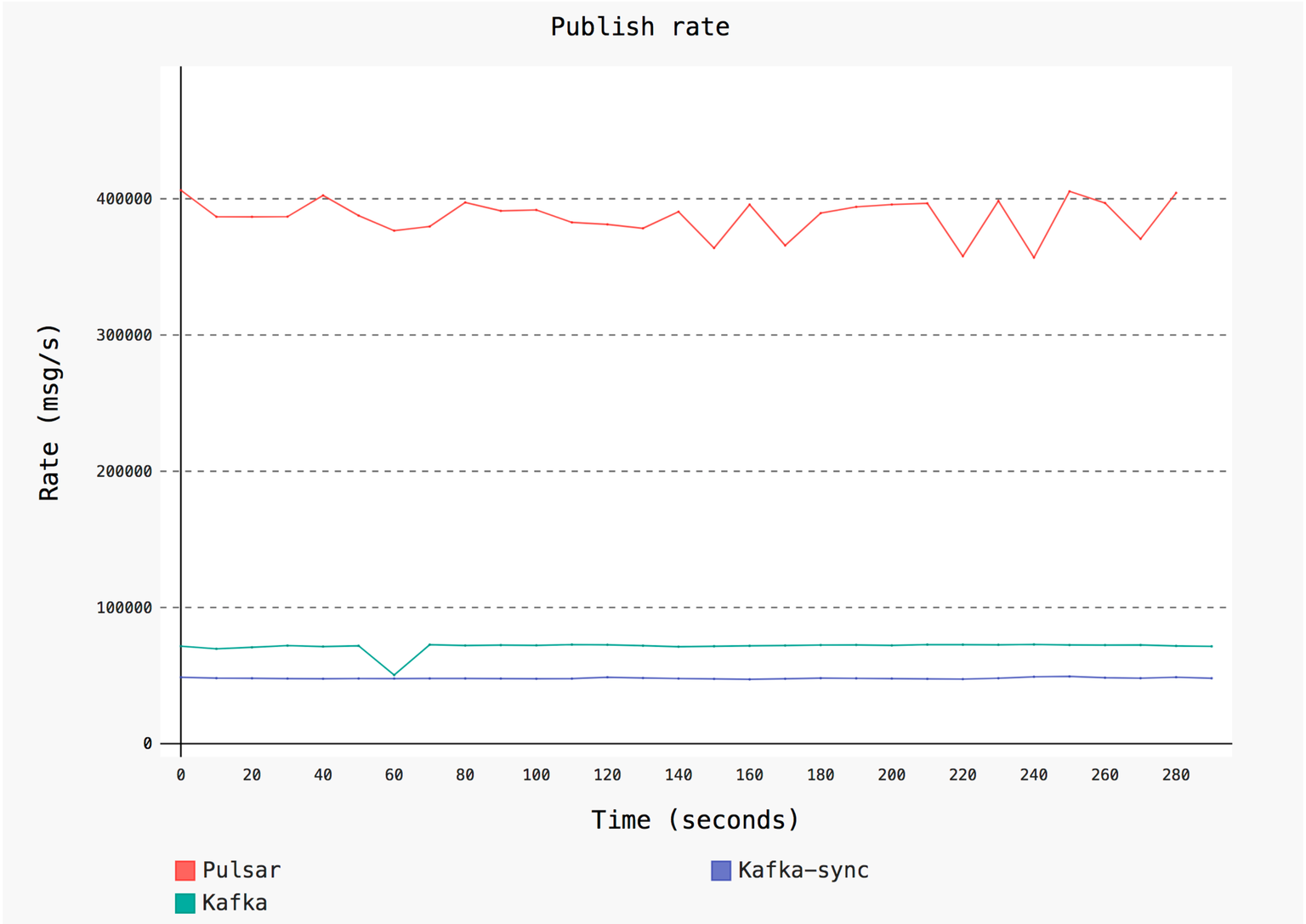
存储节点的读写隔离


Writes
4. acknowledge writes
1. append entries
Tailing Reads
Memtable
Journal
3. add entries to memtable
Index Cache
Entry Logger
Async Flush
Catch-up Reads
2. group commits
Index Disks
Ledger Disks
Journal Disks

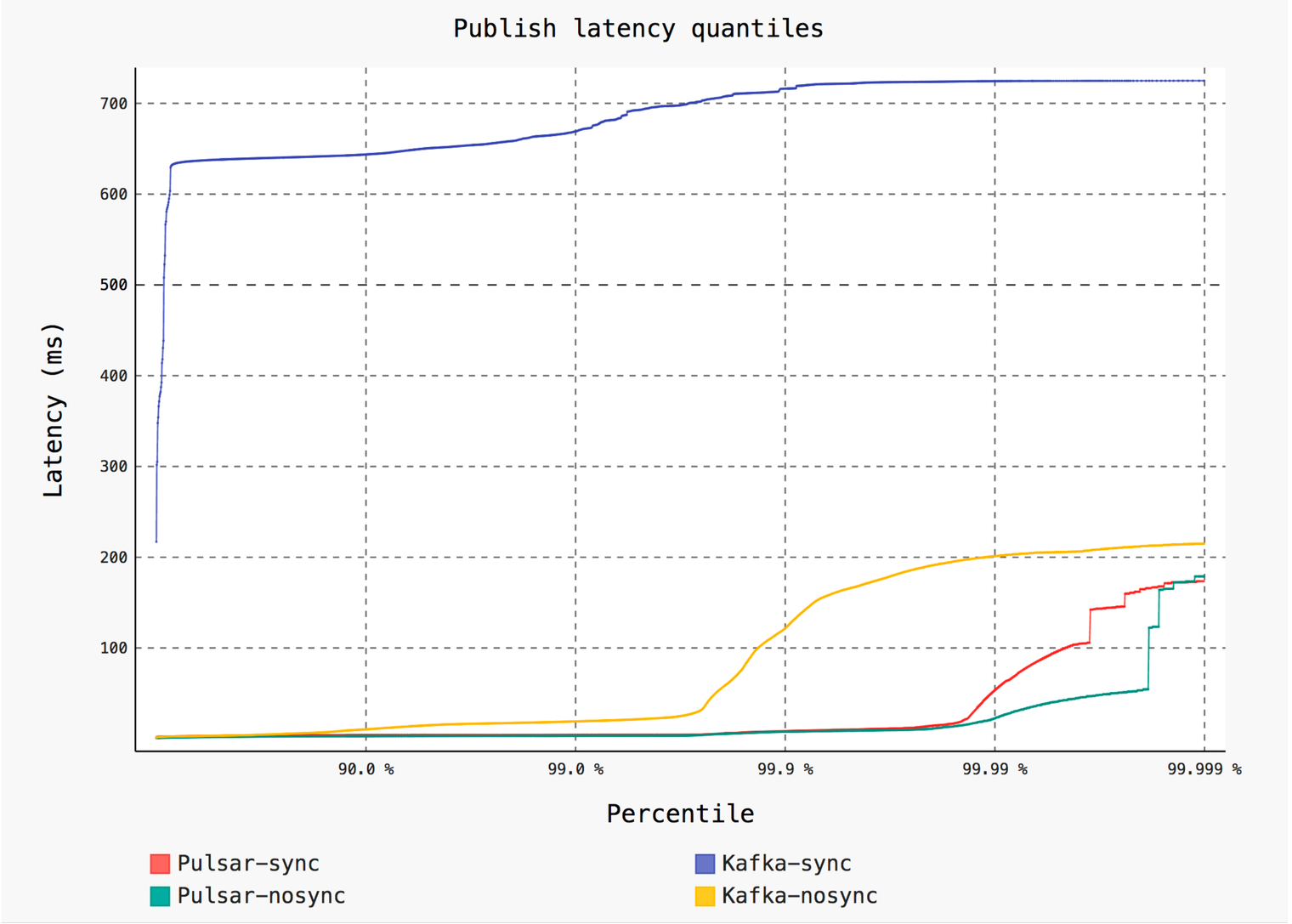
Benchmark

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

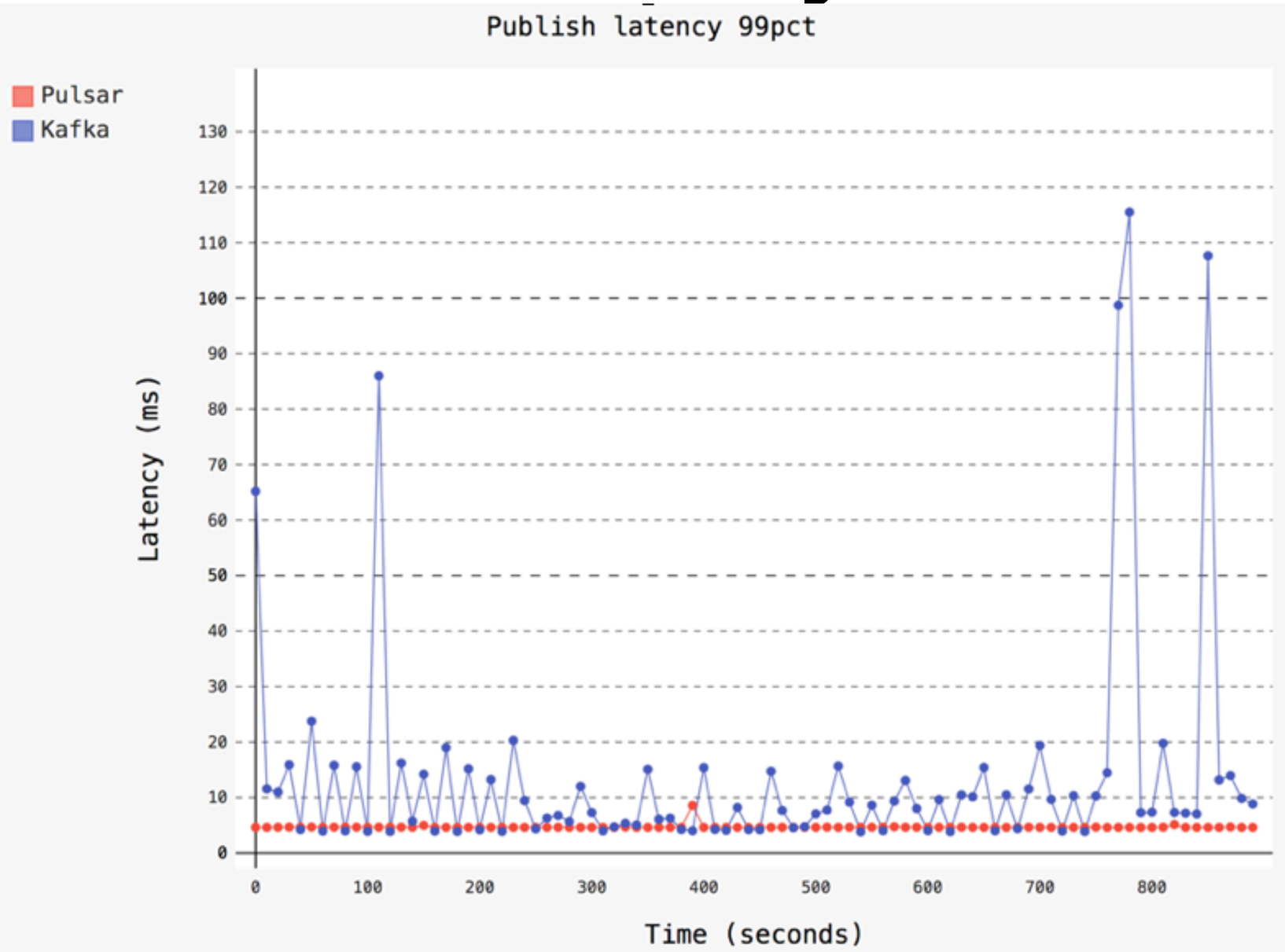
Throughput


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

Latency


Publish latency 99pct
Pulsar
Kafka
Latency (ms)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
0
100
200
300
400
500
600
700
800
Time (seconds)

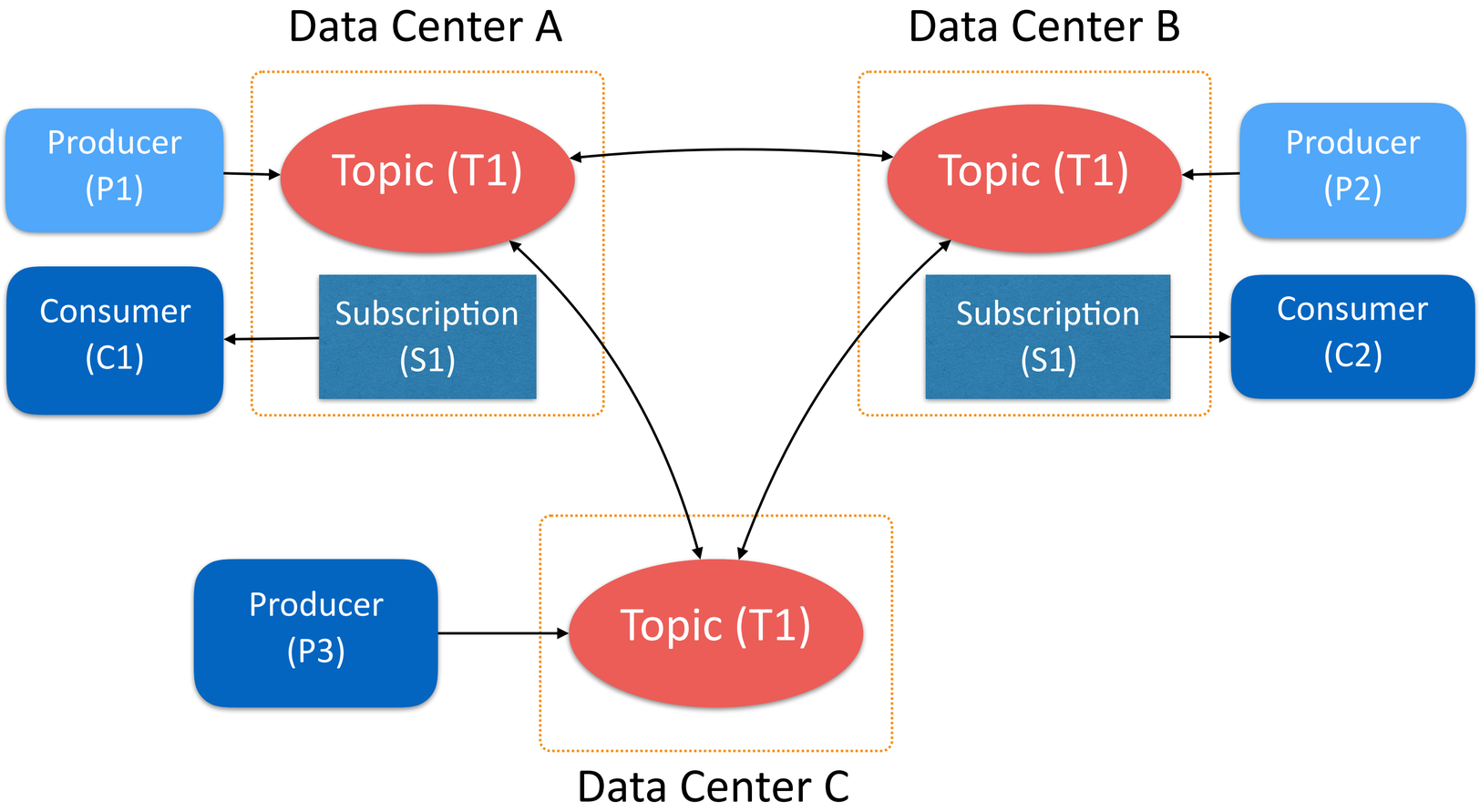
谢谢

FYI

- 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

多机房互备



- Scalable asynchronous replication
- Integrated in the broker message flow
- Simple configuration to add/remove regions

多租户

- Authentication / Authorization / Namespaces / Admin APIs
- I/O Isolations between writes and reads
  - Provided by BookKeeper - Ensure readers draining backlog won't affect publishers
- ***Soft isolation***
  - Storage quotas – flow control – rate limiting – back pressure
- ***Hardware isolation***
  - Constrain some tenants on a subset of brokers or bookies