卷皮
JUANPI.COM

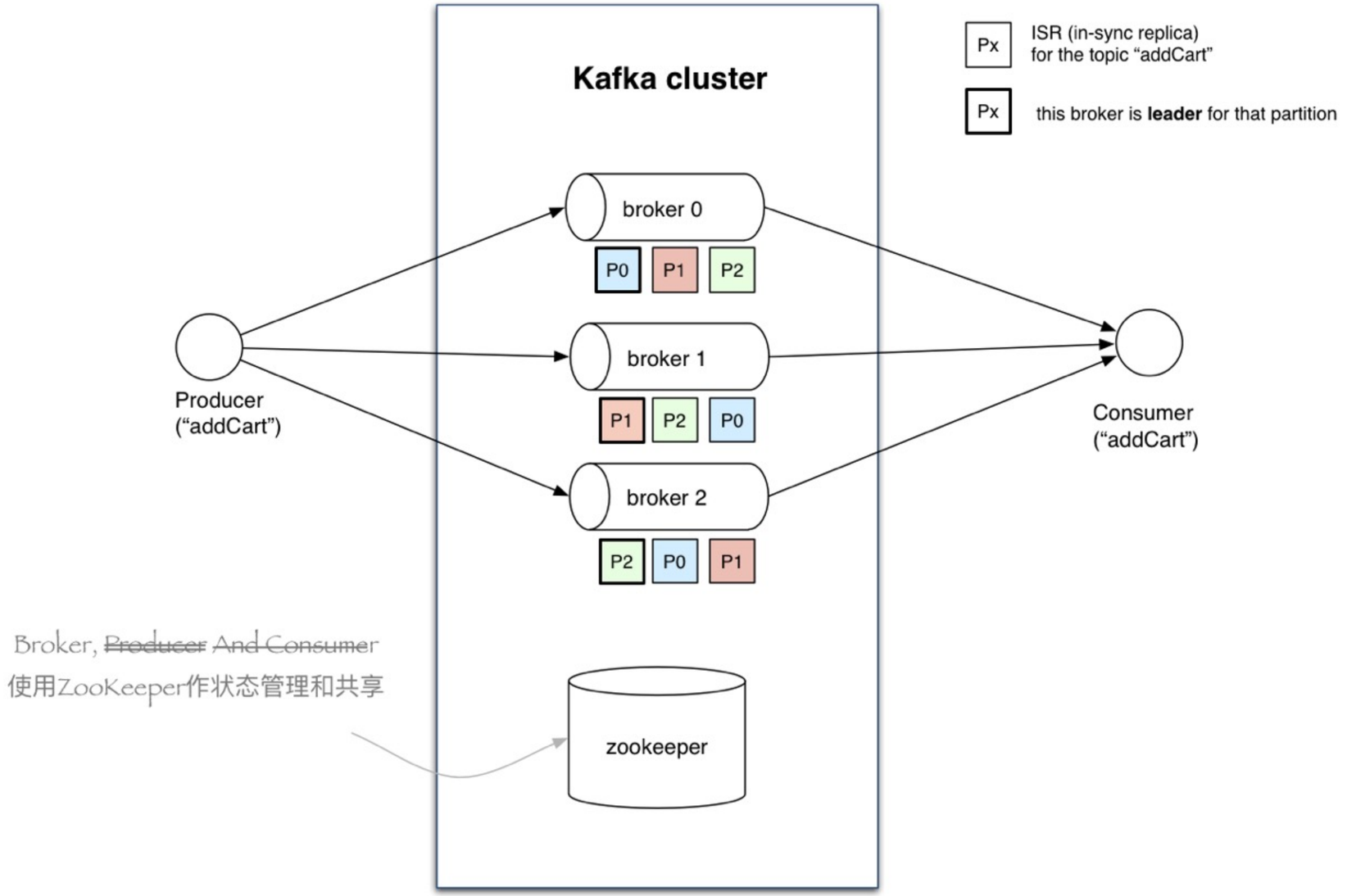
Kafka进阶-分布式存储

阿昌.何伟昌
2016.03.26

“这是最好的时代，这是最坏的时代”

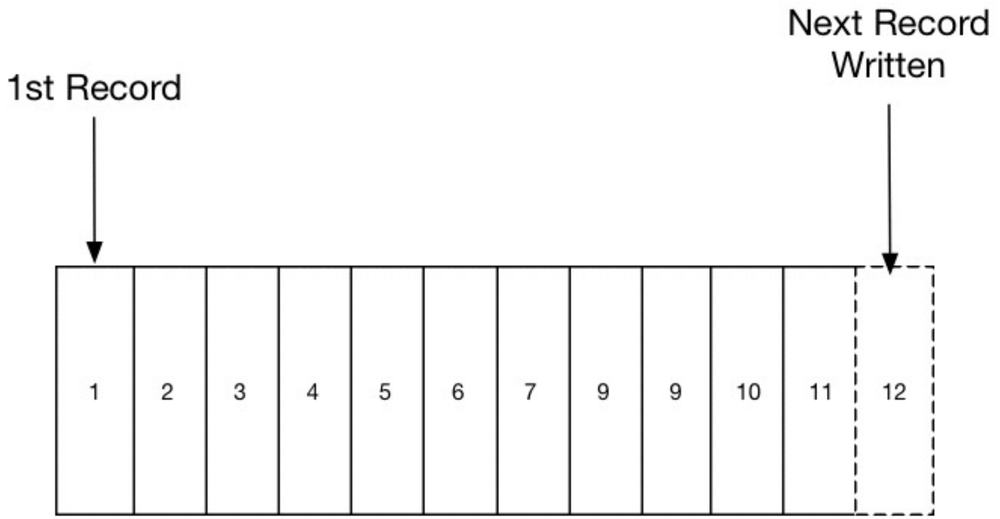
–双城记

回顾： Kafka


Kafka cluster
Px
ISR (in-sync replica)
for the topic “addCart”
Px
this broker is leader for that partition
broker 0
P0
P1
P2
broker 1
P1
P2
P0
broker 2
P2
P0
P1
Producer
(“addCart”)
Consumer
(“addCart”)
zookeeper
Broker, ~~Producer~~ ~~And Consumer~~
使用ZooKeeper作状态管理和共享

回顾：The Log

- 日志：EventLog、CommitLog
- 日志：**有序(ordered) + 不可变(immutable）**
- 《日志：有关实时数据的统一抽象》



日志：分布式系统的基石?

- 日志解决了两个问题：更改动作的排序和数据的分发。
- 日志 + Column Family = NOSQL(eg. Amama Dynamo、Apache Cassandra)
- 日志 + Lucence = Apache SolrCloud、ElasticSearch
- 日志 + SamzaJob&Yarn = Apache Samza
- 日志 + CEP(eg. Esper, Siddhi) = ?
- 日志 + DDD = ?

Replicated Log Service?

- **Apache Kafka:** Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design

- **Apache BookKeeper:** a replicated log service which can be used to build replicated state machines

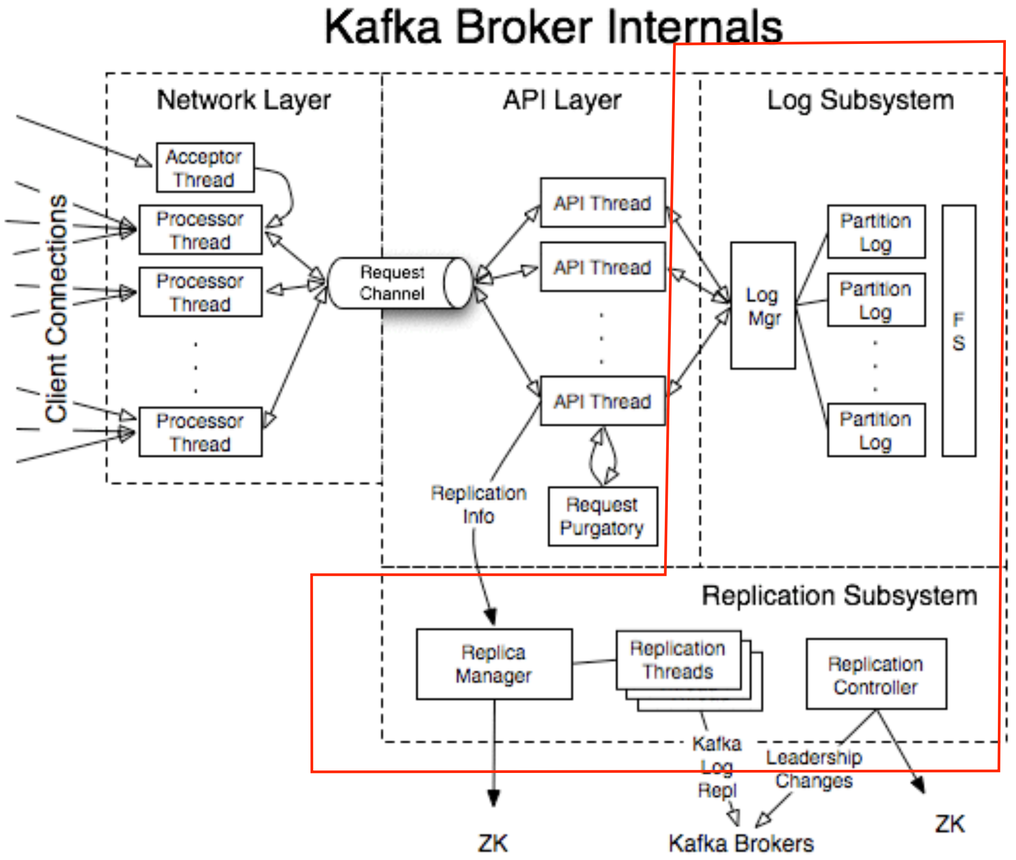
- **Apache Helix:** a cluster management framework for partition and replicated distributed resources

- **Twitter DistruteLog:** Through our experience building distributed systems at Twitter, we've identified the following requirements for any replicated log service capable of underpinning our most critical infrastructure and applications

议程

- **Kafka存储设计** — NIO，MMAP，Zero-Copy，Sparse Index，Segment，CheckPoint

- **Kafka副本设计** — Partition、State machine replication、Paxos（Controller&ISR）、ZooKeeper



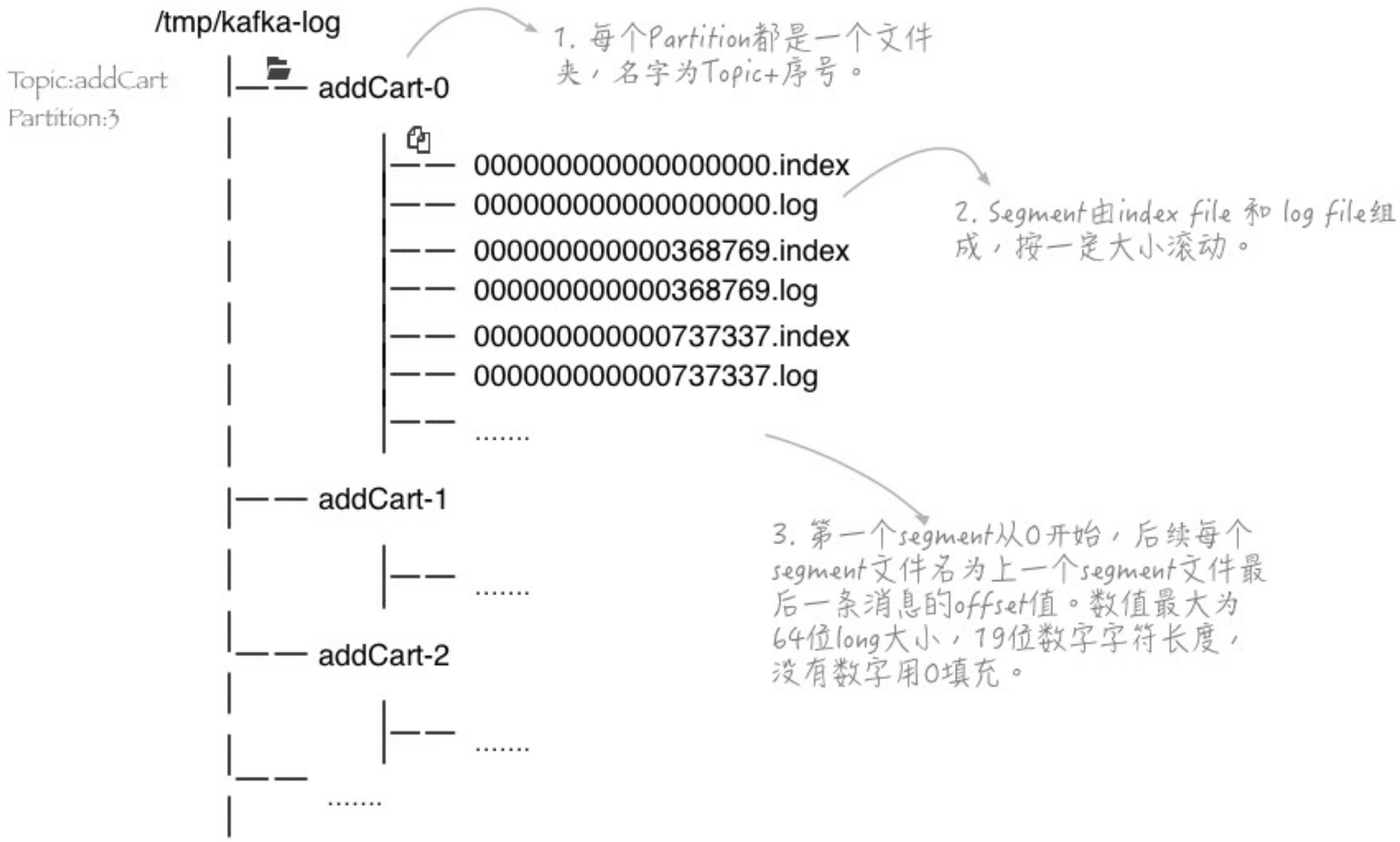
* 基于Kafka 0.9.x.x

Kafka存储设计

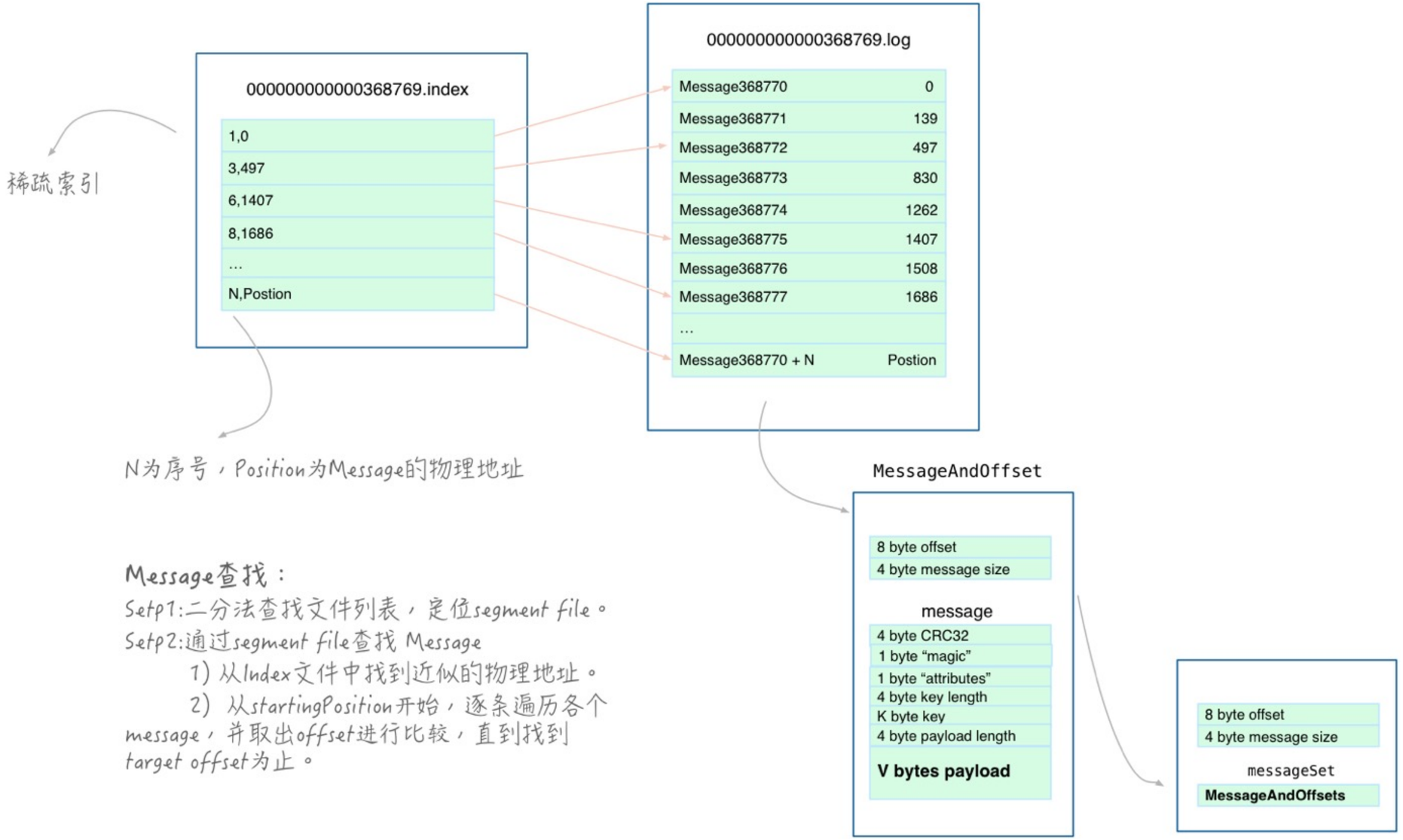
不要害怕文件系统!

- 顺序读写某些情况比随机内存读写还快
- PageCache
- MMAP
- SendFile(Zero-Copy)

日志文件


/tmp/kafka-log
Topic:addCart
Partition:3
addCart-0
00000000000000000000.index
00000000000000000000.log
00000000000368769.index
00000000000368769.log
00000000000737337.index
00000000000737337.log
.....
addCart-1
.....
addCart-2
.....
.....
1. 每个Partition都是一个文件夹，名字为Topic+序号。
2. Segment由index file 和 log file组成，按一定大小滚动。
3. 第一个segment从0开始，后续每个segment文件名为上一个segment文件最后一条消息的offset值。数值最大为64位long大小，19位数字字符长度，没有数字用0填充。

日志读写



Message查找：

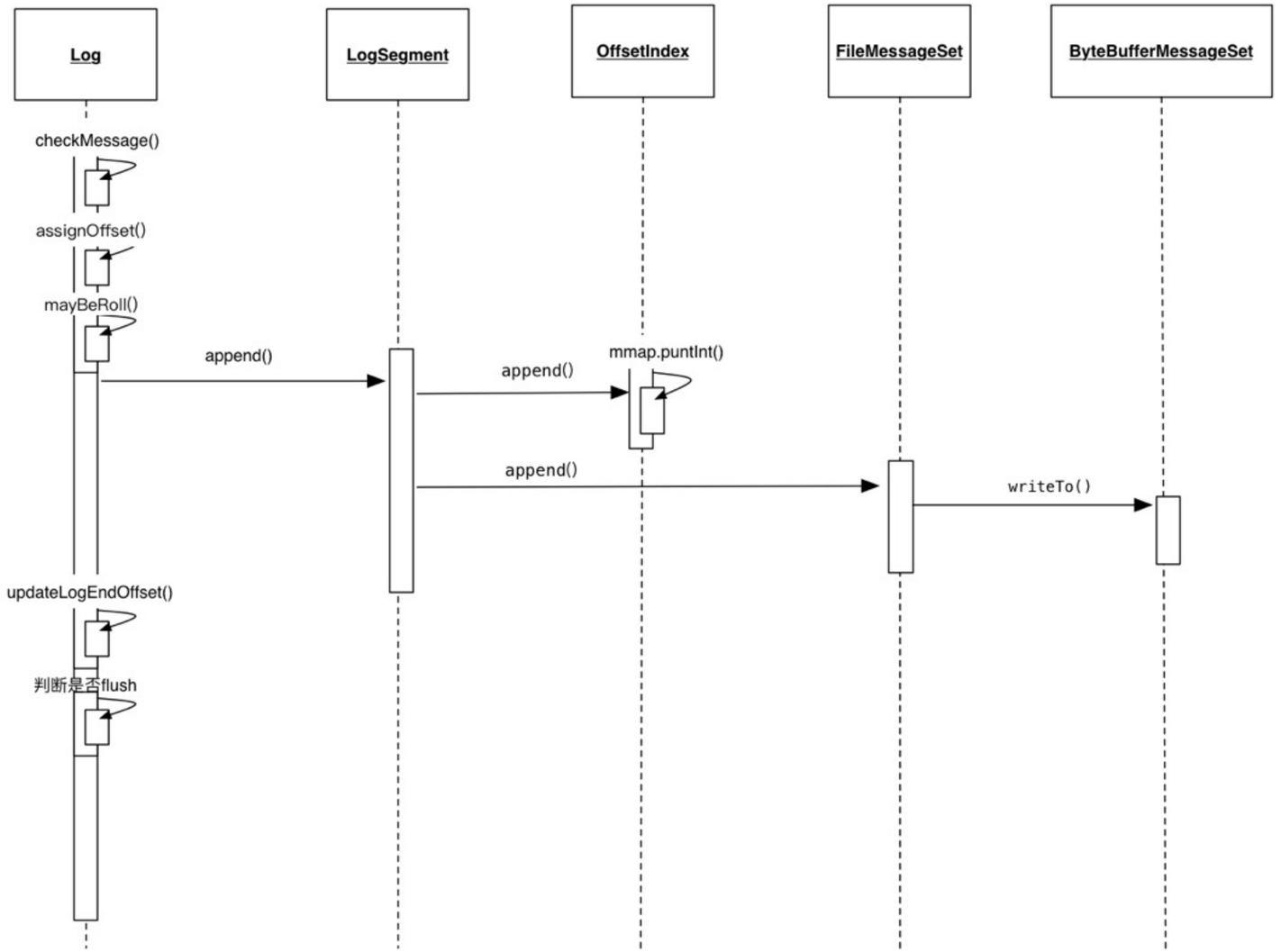
Setp1:二分法查找文件列表，定位segment file。

Setp2:通过segment file查找 Message

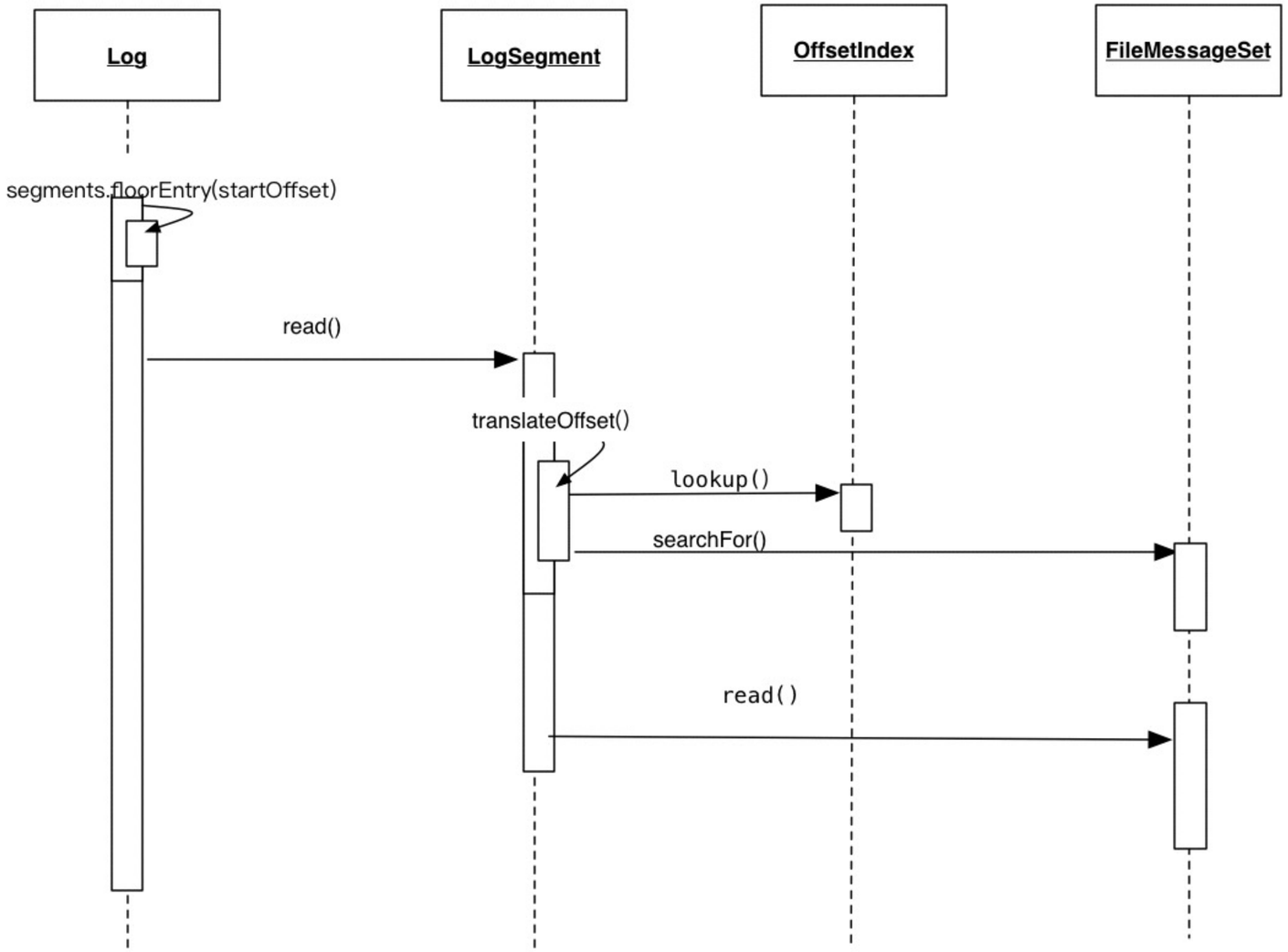
1) 从Index文件中找到近似的物理地址。

2) 从startingPosition开始，逐条遍历各个message，并取出offset进行比较，直到找到target offset为止。

日志写顺序图


Log
LogSegment
OffsetIndex
FileMessageSet
ByteBufferMessageSet
checkMessage()
assignOffset()
mayBeRoll()
append()
append()
mmap.puntInt()
append()
writeTo()
updateLogEndOffset()
判断是否flush

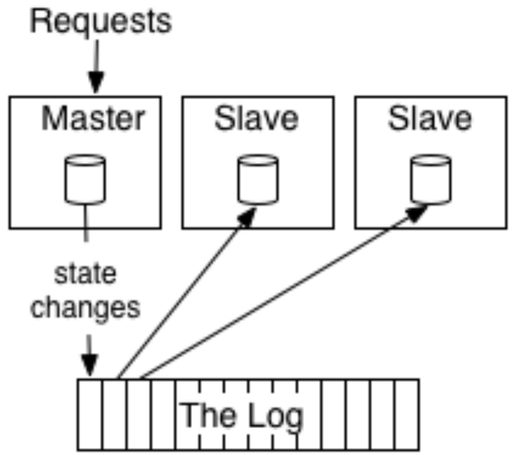
日志查找顺序图


Log
LogSegment
OffsetIndex
FileMessageSet
segments.floorEntry(startOffset)
read()
translateOffset()
lookup()
searchFor()
read()

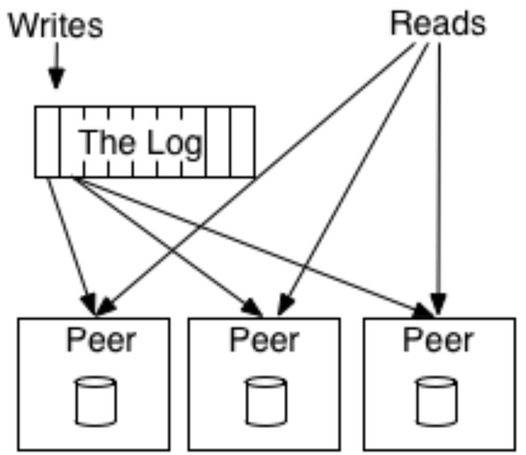
Kafka副本设计

副本(Replica)?

- Why?
  - 更强持久性（Durability）
  - 高可用（HA）
- How?
  - 副本分配：如何将一个Partition的副本均匀分配到Broker?
  - 副本传播：对于一个给定的分区，如何将每个消息传播给所有副本?

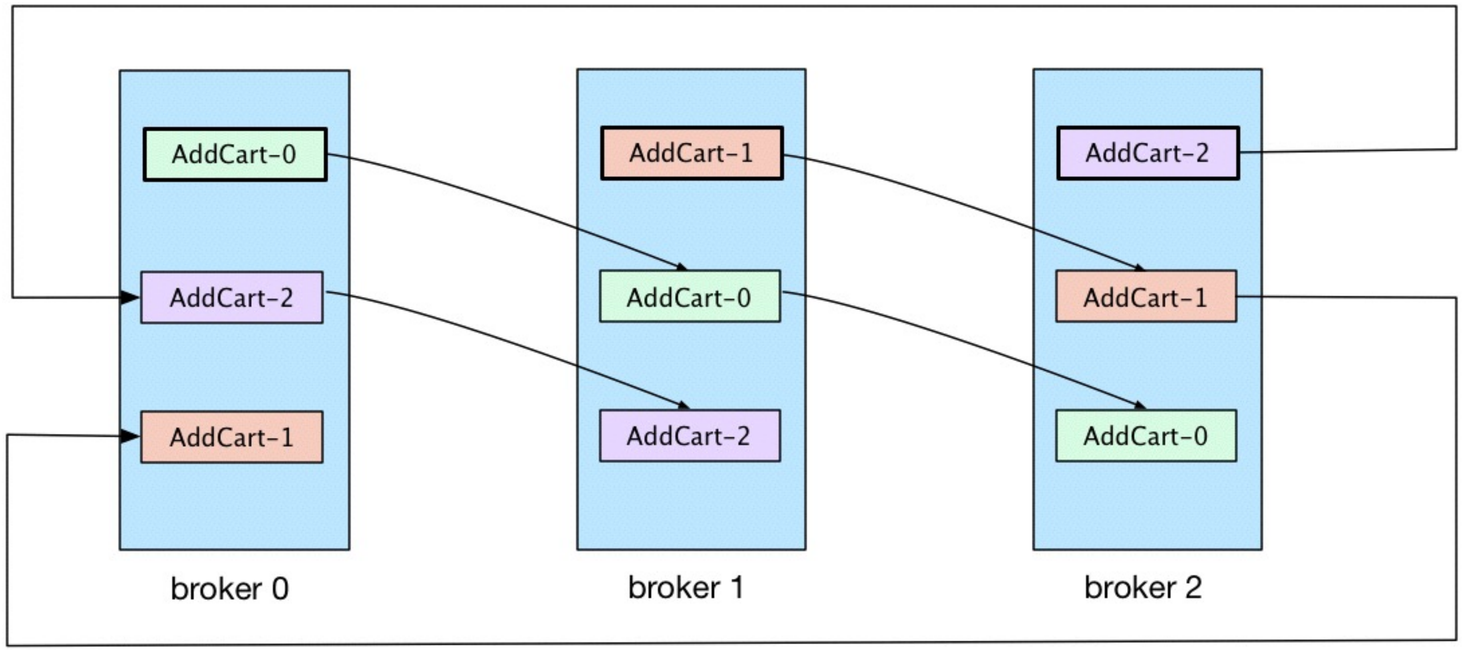


Primary-Backup



State-Machine Replication

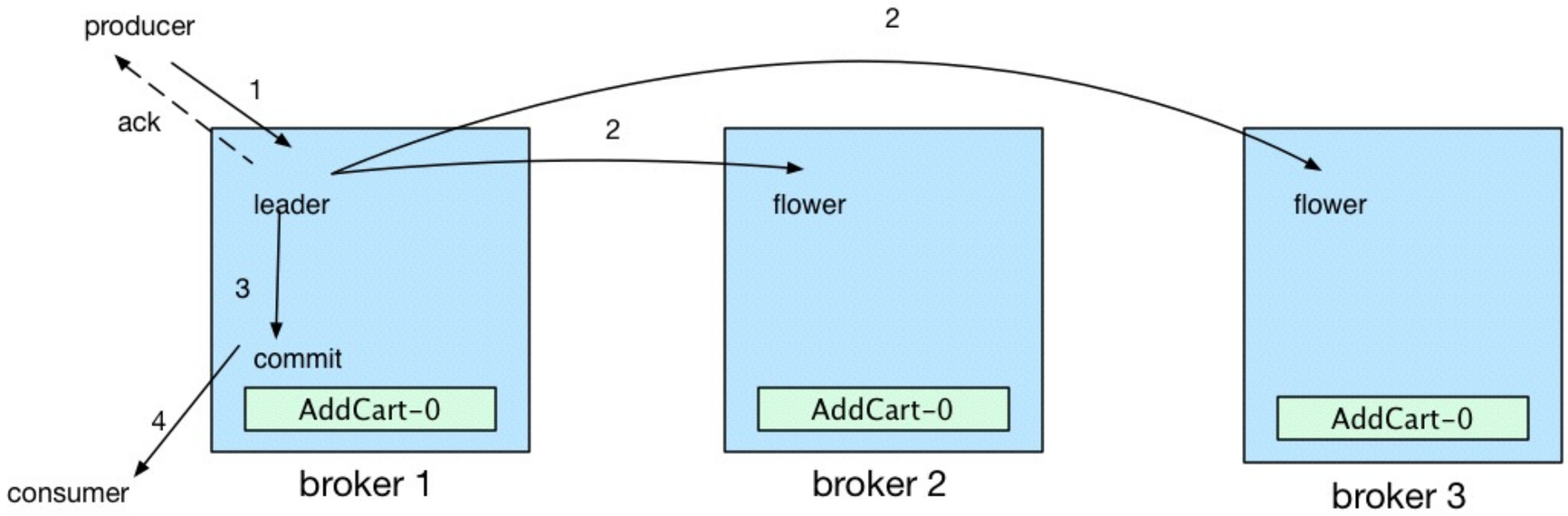
副本分配



副本分配算法
* 将n个broker和待分配的i个partition排序
* 将第i个partition分配到第(i mod n)个Broker
* 将第i个partition的第j个副本分配到第((i + j) mod n) 个broker上
注：j又叫复制因子，replication-factor

思考：增加2个broker,partition增加到5个？然后replication-factor由3变为5?
某broker0宕机了？partition能减小吗？如何解决Leader分配不均的问题?

副本传播



When producer receives ack	Latency	Durability on failures
no ack	no network delay	some data loss
wait for leader	1 network roundtrip	a few data loss
wait for committed	2 network roundtrips	no data loss

Controller

- Brokers中选举一个内嵌Controller
  - Broker抢注ZooKeeper临时节点（使用ZooKeeper leader elect)
  - 增加Epoch减小网络分区的问题。
- Controller的职责
  - Replicas Leader选举，确定ISR
  - 删除Topic及Replica的重新分配

Leader

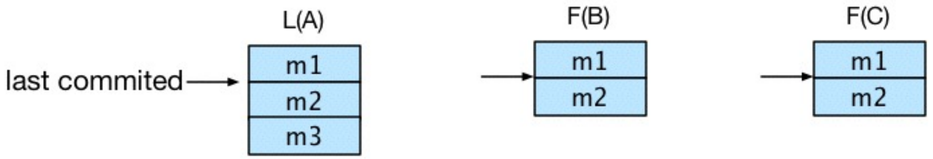
- Leader 选举
  - Controller在ISR中选举Replica Leader
- Leader 职责
  - 处理Client的读写请求，处理Follwer拉取消息的请求
  - 维护ISR列表
  - 维护HW（high watemark）

Follower

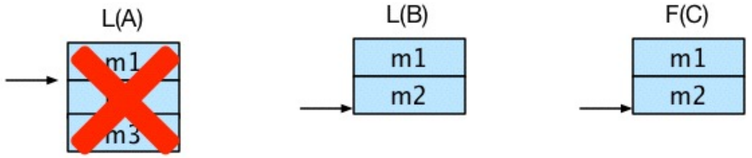
- 不停地去Leader拉取消息
- 当 Follower 重新启动
  - 从leader拉取消息
  - 当完成跟上Leader，加入ISR

副本恢复例子

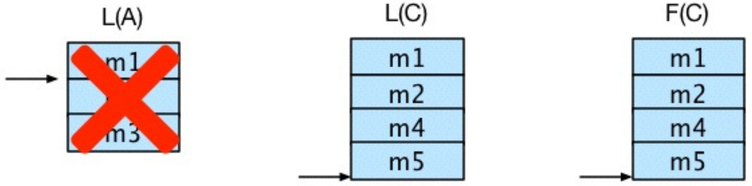
1. ISR = {A,B,C}; LeaderA 提交消息m1



2. A失败然后B是新的Leader， ISR = {B,C}， B提交消息m2而不是m3

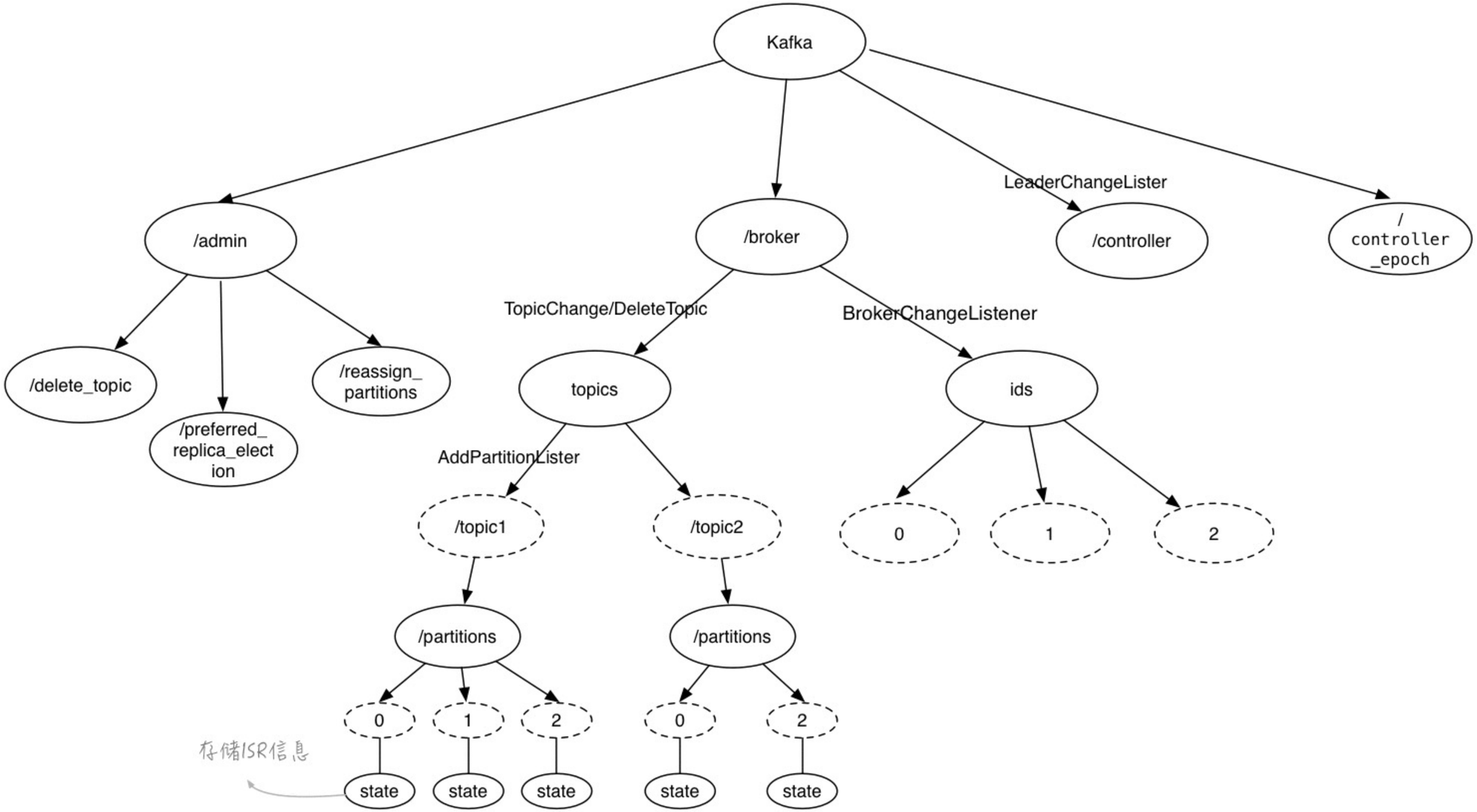


3. B 提交消息m4, m5



4. A 恢复， A从m1开始同步，直到跟上Leader然后加入ISR，ISR={A, B, C}

从ZNode看复制


Kafka
/admin
/broker
LeaderChangeLister
/controller
/controller_epoch
/delete_topic
/preferred_replica_election
/reassign_partitions
TopicChange/DeleteTopic
BrokerChangeListener
topics
ids
AddPartitionLister
/topic1
/topic2
0
1
2
/partitions
/partitions
0
1
2
0
2
state
state
state
state
state
存储ISR信息

综合讨论：Topic&Partition的成本

后续培训计划

- Kafka 网络通信
- 0.9后的NEW Consumer设计

2016.03.26