读书笔记 3 Broker 消息队列的核心机制

消息的存储和发送

消息的存储结构图：

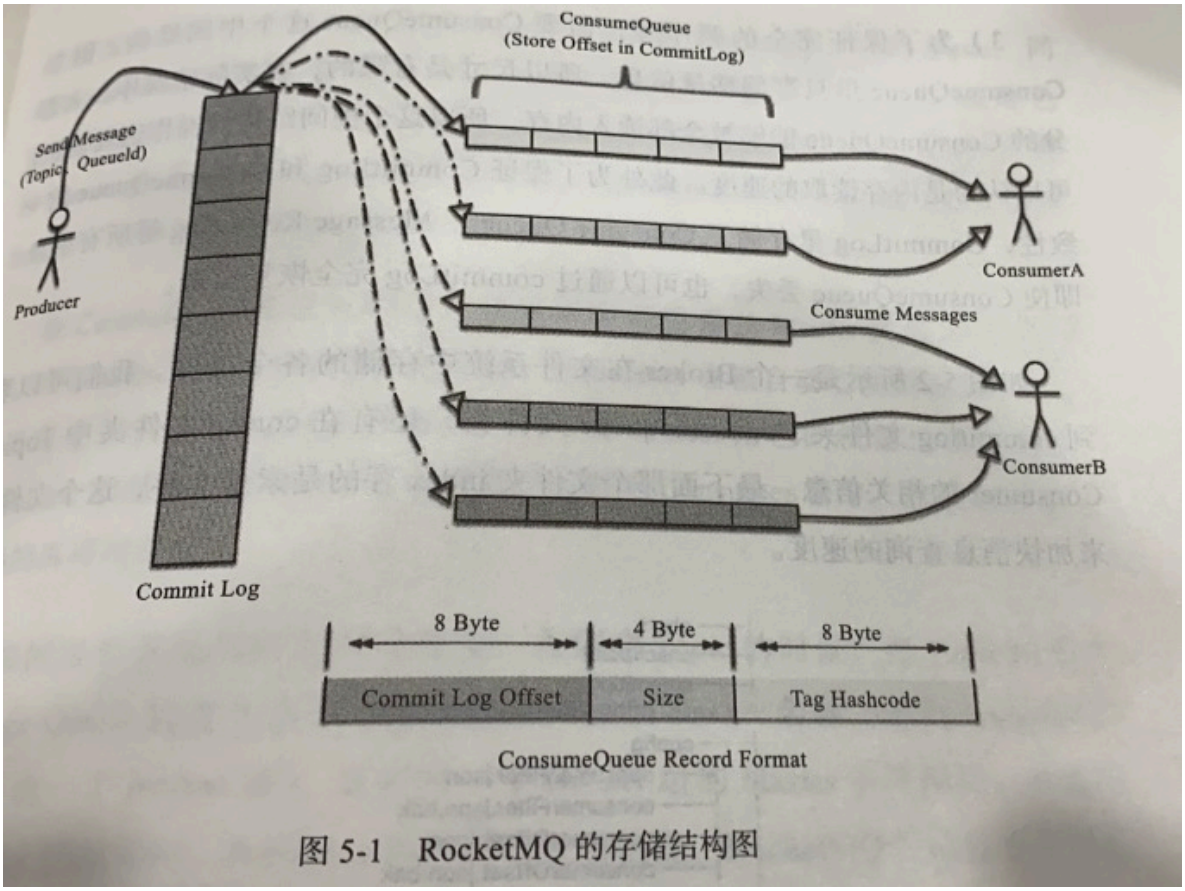

图 5-1　RocketMQ 的存储结构图

rocketMq的存储 通过 ConsumeQueue 和 CommitLog配合完成
消息的真正物理存储文件是CommitLog
ConsumeQueue是消息的逻辑队列，类似索引，存储的是指向物理存储的地址

每个Topic下 每个Message queue 都有一个对应的 ConsumeQueue文件
${$storeRoot}/ consumequeue/${topicName}/ ${queueID}/ ${fileName}

CommitLog 以物理文件方式存放，每台Broker 上 commitLog被本机所有ConsumeQueue共享

commitLog的消息存储长度不固定， 但是 rocketMq采取机制 尽量向 CommitLog中 顺序写、随机读。
consumeQueue内容也会持久化到磁盘中

- 顺序写==》提升写入效率

- 随机读==〉 通过pageCache机制， 批量读取，cache到内存，加速后续读取速度
- 保证完全的顺序写，需要ConsumeQueue这个中间结构
  consumeQueue只存偏移量信息，存量有限，而且大部都直接读到内存中
  为了保证consumeQueue和commitLog一致性， commitLog中存储了所有信息，可以用来恢复

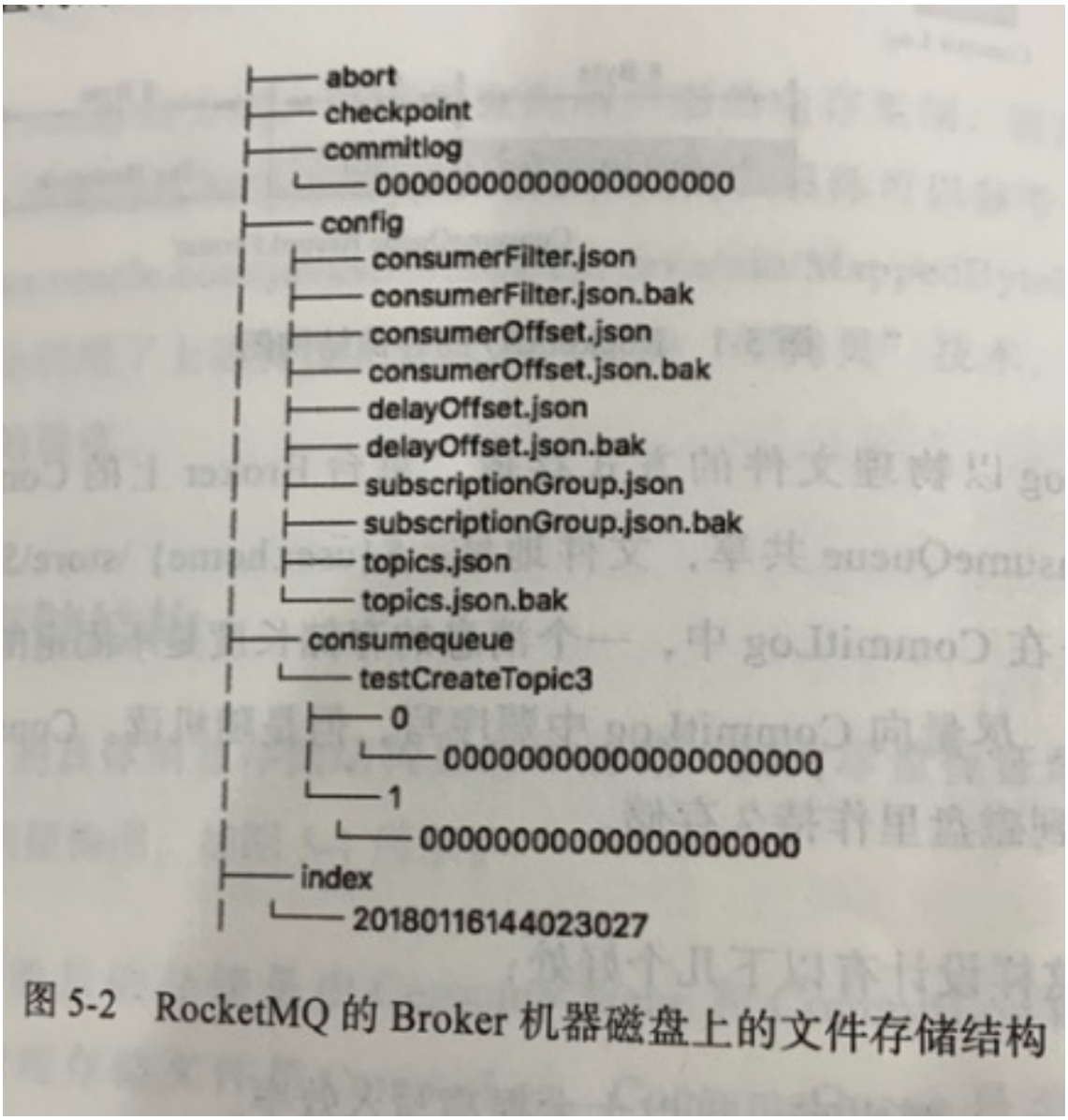

图 5-2 RocketMQ 的 Broker 机器磁盘上的文件存储结构

RocketMq 高可用机制

rocketMq集群通过Master 和 slave 的配合达到高可用。

broker 配置文件中， brokerId = 0 代表master。 > 0 slave 还有brokerRole也是同样作用

master支持 读、写 ， slave 只能读===》 producer只能连接master写，consumer可以两个读

消费端的高可用===〉 Master 不可用时， consumer 会自动切换到 slave 读

生产端的高可用===》 创建 Topic 时，topic 的多个 message queue 创建在多个 broker 组上（相同 broker 名称，不同 brokerId 的机器组成 broker 组）
当一个 broker 组的 master 不可用，其他组的 master 仍然可用，producer 还是可以发消息
（rocketmq 暂时不支持 slave 自动转成 master，需要时要手动停止 slave 的 broker，新配置文件启动）

同步刷盘、 异步刷盘

Rocketmq 消息存储在磁盘上（保证断电恢复 & 防止内存不足）， 为了性能，尽可能保证顺序写。

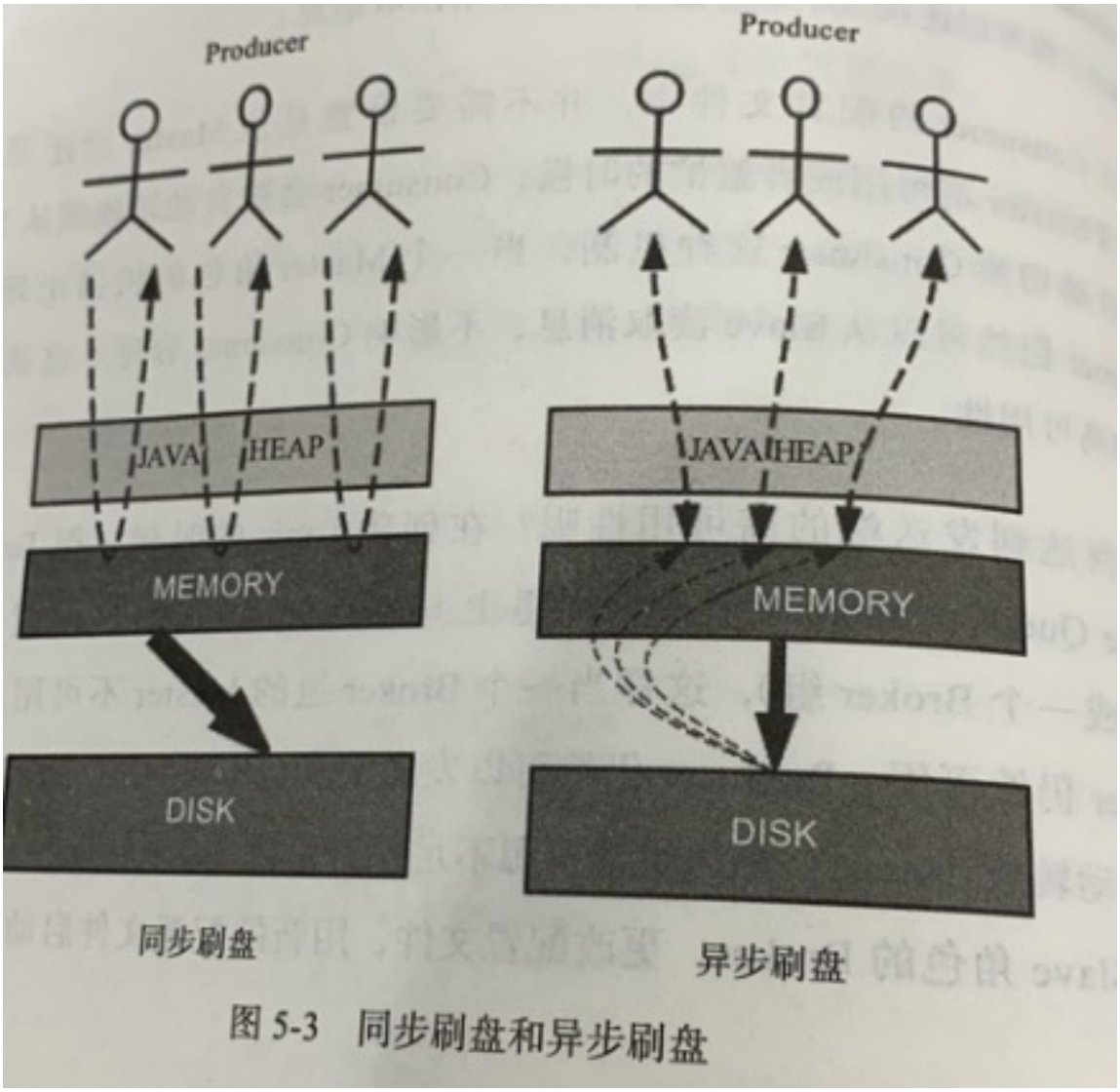

图 5-3 同步刷盘和异步刷盘

Producer 写入磁盘有两种方式：

- 异步刷盘：写成功时，消息可能只是写入内存 的 PageCache，返回快，吞肚量大；内存里消息累积到一定程度，触发写入；
- 同步刷盘：返回写成功时，已经写入磁盘。 流程：写入 pageCache 的消息，立刻 通知磁盘线程刷盘，等待刷盘完成==》唤醒等待线程，返回写成功

broker配置中， flushDiskType 参数设置： SYNC_FLUSH 、 ASYNC_FLUSH

同步复制、 异步复制

broker 祖上有 master、 slave ，消息从 master 复制到 slave

- 同步复制： master、slave 都写入成功时
  优点：master 故障，slave 有全部备份，容易恢复
  但是增大数据写入延迟，降低吞吐量

- 异步复制： 只要 master 写成功就返回客户端 success
  延迟低、吞吐量高
  未写入 slave 的可能丢失

brokerRole 设置 ASYNC_MASTER. SYNC_MASTER. SALVE
要结合业务场景调整

总结：
rocketMQ 基于 “顺序写”，“随机读”的原则设计，利用“零拷贝”基数，克服磁盘操作限制

为了高可用的主从机制，数据被及时复制到多个机器
Rocketmq 将选择留给用户根据业务场景选择