22 | Kafka和RocketMQ的消息复制实现的差异点在哪?

李玥 · 消息队列高手课

你好，我是李玥。

之前我在《05 | 如何确保消息不会丢失?》那节课中讲过，消息队列在收发两端，主要是依靠业务代码，配合请求确认的机制，来保证消息不会丢失的。而在服务端，一般采用持久化和复制的方式来保证不丢消息。

把消息复制到多个节点上，不仅可以解决丢消息的问题，还可以保证消息服务的高可用。即使某一个节点宕机了，还可以继续使用其他节点来收发消息。所以大部分生产系统，都会把消息队列配置成集群模式，并开启消息复制，来保证系统的高可用和数据可靠性。

这节课我们来讲一下，消息复制需要解决的一些问题，以及 RocketMQ 和 Kafka 都是如何应对这些问题来实现复制的。

消息复制面临什么问题?

我们希望消息队列最好能兼具高性能、高可用并且还能提供数据一致性的保证。虽然很多消息队列产品宣称三个特性全都支持，但你需要知道，这都是有前置条件的。

首先来说性能。任何的复制实现方式，数据的写入性能一定是不如单节点的。这个很好理解，因为无论采用哪种复制实现方式，都需要数据被写入到多个节点之后再返回，性能一定是不如只写入一个节点的。

需要写入的节点数量越多，可用性和数据可靠性就越好，但是写入性能就越低，这是一个天然的矛盾。不过，复制对消费的性能影响不大，不管采用哪种复制方式，消费消息的时候，都只是选择多副本中一个节点去读数据而已，这和单节点消费并没有差别。

再来说一致性，消息队列对数据一致性的要求，既包括了“不丢消息”这个要求，也包括“严格顺序”的要求。如果要确保数据一致性，必须采用“主 – 从”的复制方式，这个结论是有严格的数学论证的，大家只要记住就可以了。

在“主 – 从”模式下，数据先写入到主节点上，从节点只从主节点上复制数据，如果出现主从数据不一致的情况，必须以主节点上的数据为准。这里面需要注意一下，这里面的主节点它并不是不可变的，在很多的复制实现中，当主节点出现问题的时候，其他节点可以通过选举的方式，变成主节点。只要保证，在任何一个时刻，集群的主节点数不能超过 1 个，就可以确保数据一致性。

最后说一下高可用。既然必须要采用主从的复制方式，高可用需要解决的就是，当某个主节点宕机的时候，尽快再选出一个主节点来接替宕机的主节点。

比较快速的实现方式是，使用一个第三方的管理服务来管理这些节点，发现某个主节点宕机的时候，由管理服务来指定一个新的主节点。但引入管理服务会带来一系列问题，比如管理服务本身的高可用、数据一致性如何保证?

有的消息队列选择自选举的方式，由还存活的这些节点通过投票，来选出一个新的主节点，这种投票的实现方式，它的优点是没有外部依赖，可以实现自我管理。缺点就是投票的实现都比较复杂，并且选举的过程是比较慢的，几秒至几十秒都有可能，在选出新的主节点前，服务一直是不可用的。

大部分复制的实现，都不会选择把消息写入全部副本再返回确认，因为这样虽然可以保证数据一致性，但是，一旦这些副本中有任何一个副本宕机，写入就会卡死了。如果只把消息写入到一部分副本就认为写入成功并返回确认，就可以解决卡死的问题，并且性能也会比写全部副本好很多。

到底写入多少个副本算写入成功呢？这又是一个非常难抉择的问题。

假设我们的集群采用“一主二从三副本”的模式，如果只要消息写入到两个副本就算是写入成功了，那这三个节点最多允许宕机一个节点，否则就没法提供服务了。如果说我们把要求写入的副本数量降到 1，只要消息写入到主节点就算成功了，那三个节点中，可以允许宕机两个节点，系统依然可以提供服务，这个可用性就更好一些。但是，有可能出现一种情况：主节点有一部分消息还没来得复制到任何一个从节点上，主节点就宕机了，这时候就会丢消息，数据一致性又没有办法保证了。

以上我讲的这些内容，还没有涉及到任何复制或者选举的方法和算法，都是最朴素，最基本的原理。你可以看出，这里面是有很多天然的矛盾，所以，**目前并没有一种完美的实现方案能够兼顾高性能、高可用和一致性。**

不同的消息队列选择了不同的复制实现方式，这些实现方式都有各自的优缺点，在高性能、高可用和一致性方面提供的能力也是各有高低。接下来我们一起来看一下 RocketMQ 和 Kafka 分别是如何来实现复制的。

RocketMQ 如何实现复制?

RocketMQ 在 2018 年底迎来了一次重大的更新，引入 Deldger，增加了一种全新的复制方式。我们先来说一下传统的复制方式。

在 RocketMQ 中，复制的基本单位是 Broker，也就是服务端的进程。复制采用的也是主从方式，通常情况下配置成一主一从，也可以支持一主多从。

RocketMQ 提供了两种复制方式，一种是异步复制，消息先发送到主节点上，就返回“写入成功”，然后消息再异步复制到从节点上。另外一种方式是同步双写，消息同步双写到主从节点

上，主从都写成功，才返回“写入成功”。这两种方式本质上的区别是，写入多少个副本再返回“写入成功”的问题，异步复制需要的副本数是 1，同步双写需要的副本数是 2。

我刚刚讲过，如果在返回“写入成功”前，需要写入的副本数不够多，那就会丢消息。对 RocketMQ 来说，如果采用异步复制的方式会不会丢消息呢？答案是，并不会丢消息。

我来跟你说一下为什么不会丢消息。

在 RocketMQ 中，Broker 的主从关系是通过配置固定的，不支持动态切换。如果主节点宕机，生产者就不能再生产消息了，消费者可以自动切换到从节点继续进行消费。这时候，即使有一些消息没有来得及复制到从节点上，这些消息依然躺在主节点的磁盘上，除非是主节点的磁盘坏了，否则等主节点重新恢复服务的时候，这些消息依然可以继续复制到从节点上，也可以继续消费，不会丢消息，消息的顺序也是没有问题的。

从设计上来讲，**RocketMQ 的这种主从复制方式，牺牲了可用性，换取了比较好的性能和数据一致性。**

那 RocketMQ 又是如何解决可用性的问题的呢？一对儿主从节点可用性不行，多来几对儿主从节点不就解决了？RocketMQ 支持把一个主题分布到多对主从节点上去，每对主从节点中承担主题中的一部分队列，如果某个主节点宕机了，会自动切换到其他主节点上继续发消息，这样既解决了可用性的问题，还可以通过水平扩容来提升 Topic 总体的性能。

这种复制方式在大多数场景下都可以很好的工作，但也面临一些问题。

比如，在需要保证消息严格顺序的场景下，由于在主题层面无法保证严格顺序，所以必须指定队列来发送消息，对于任何一个队列，它一定是落在一组特定的主从节点上，如果这个主节点宕机，其他的主节点是无法替代这个主节点的，否则就无法保证严格顺序。在这种复制模式下，严格顺序和高可用只能选择一个。

RocketMQ 引入 Dledger，使用新的复制方式，可以很好地解决这个问题。我们来看一下 Dledger 是怎么来复制的。

Dledger 在写入消息的时候，要求至少消息复制到半数以上的节点之后，才给客户端返回写入成功，并且它是支持通过选举来动态切换主节点的。

同样拿 3 个节点举例说明一下。当主节点宕机的时候，2 个从节点会通过投票选出一个新的主节点来继续提供服务，相比主从的复制模式，解决了可用性的问题。由于消息要至少复制到 2 个节点上才会返回写入成功，即使主节点宕机了，也至少有一个节点上的消息是和主节点一样的。Dledger 在选举时，总会把数据和主节点一样的从节点选为新的主节点，这样就保证了数据的一致性，既不会丢消息，还可以保证严格顺序。

当然，Dledger 的复制方式也不是完美的，依然存在一些不足：比如，选举过程中不能提供服务。最少需要 3 个节点才能保证数据一致性，3 节点时，只能保证 1 个节点宕机时可用，如果 2 个节点同时宕机，即使还有 1 个节点存活也无法提供服务，资源的利用率比较低。另外，由于至少要复制到半数以上的节点才返回写入成功，性能上也不如主从异步复制的方式快。

讲完了 RocketMQ，我们再来看看 Kafka 是怎么来实现复制的。

Kafka 是如何实现复制的?

Kafka 中，复制的基本单位是分区。每个分区的几个副本之间，构成一个小的复制集群，Broker 只是这些分区副本的容器，所以 Kafka 的 Broker 是不分主从的。

分区的多个副本中也是采用一主多从的方式。Kafka 在写入消息的时候，采用的也是异步复制的方式。消息在写入到主节点之后，并不会马上返回写入成功，而是等待足够多的节点都复制成功后再返回。在 Kafka 中这个“足够多”是多少呢？Kafka 的设计哲学是，让用户自己来决定。

Kafka 为这个“足够多”创造了一个专有名词：ISR（In Sync Replicas），翻译过来就是“保持数据同步的副本”。ISR 的数量是可配的，但需要注意的是，这个 ISR 中是包含主节点的。

Kafka 使用 ZooKeeper 来监控每个分区的多个节点，如果发现某个分区的主节点宕机了，Kafka 会利用 ZooKeeper 来选出一个新的主节点，这样解决了可用性的问题。ZooKeeper

是一个分布式协调服务，后面，我会专门用一节课来介绍 ZooKeeper。选举的时候，会从所有 ISR 节点中来选新的主节点，这样可以保证数据一致性。

默认情况下，如果所有的 ISR 节点都宕机了，分区就无法提供服务了。你也可以选择配置成让分区继续提供服务，这样只要有一个节点还活着，就可以提供服务，代价是无法保证数据一致性，会丢消息。

Kafka 的这种高度可配置的复制方式，优点是非常灵活，你可以通过配置这些复制参数，在可用性、性能和一致性这几方面做灵活的取舍，缺点就是学习成本比较高。

总结

这节课我们主要来讲了一下，消息复制需要面临的问题以及 RocketMQ 和 Kafka 都是如何应对这些问题来实现复制的。

RocketMQ 提供新、老两种复制方式：传统的主从模式和新的基于 Dledger 的复制方式。传统的主从模式性能更好，但灵活性和可用性稍差，而基于 Dledger 的复制方式，在 Broker 故障的时候可以自动选举出新节点，可用性更好，性能稍差，并且资源利用率更低一些。Kafka 提供了基于 ISR 的更加灵活可配置的复制方式，用户可以自行配置，在可用性、性能和一致性这几方面根据系统的情况来做取舍。但是，这种灵活的配置方式学习成本较高。

并没有一种完美的复制方案，可以同时能够兼顾高性能、高可用和一致性。你需要根据你实际的业务需求，先做出取舍，然后再去配置消息队列的复制方式。

思考题

假设我们有一个 5 节点的 RocketMQ 集群，采用 Dledger5 副本的复制方式，集群中只有一个主题，50 个队列均匀地分布到 5 个 Broker 上。

如果需要你来配置一套 Kafka 集群，要求达到和这个 RocketMQ 集群一样的性能（不考虑 Kafka 和 RocketMQ 本身的性能差异）、可用性和数据一致性，该如何配置？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有帮助，也欢迎把它分享给你的朋友。

精选留言 (34)

a、
2019-09-26

5副本，10个分区，至少保持isr集合中有三个broker

bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 5 --partitions 10 --topic test

min.insync.replicas=3

作者回复: 👍👍👍

共 5 条评论>

35

kiddkidd
2020-01-11

老师，对于文中“由于消息要至少复制到 2 个节点上才会返回写入成功，即使主节点宕机了，也至少有一个节点上的消息是和主节点一样的”我有个疑问：
假如1主2从3副本条件下，主收到msg1，并复制到从1，之后主又收到msg2，并复制到从2.然后主节点宕机了，此时从1和从2都跟主不一样啊。请问如何理解?

作者回复: 消息的复制是按照顺序来复制的，因为msg2在msg1后面，如果说，msg2已经复制到从2上，那从2上一定有msg1，所以不存在你说这种情况。

共 2 条评论>

22

WL
2019-10-13

请问一下老师，kafka消息复制中，“Broker只是副本分区的容器，Broker不分主从”这句话具体怎么理解？ 是指一个Broker上可能有Topic1的partition2的从副本和Topic2的partition1的主副本，所以在Broker上不分主从，是这样吗?

作者回复: 没错，就是这样的。

15

郑勺子
2019-09-17

请问老师～可用于消息中间性能测试的工具有哪些～

作者回复: 你可以参考一下这个：https://github.com/openmessaging/openmessaging-benchmark/

11

丁小明
2020-05-05

看起来好像新的rocktmq复制方式，就是内置了类似zk一样的一致性协调器。

作者回复: 是的，RocketMQ新的复制方式采用的也是Raft一致性协议。

9

不能扮演天使
2019-09-16

采用RocketMQ Dledger 5副本复制方式，那就是最多允许宕机两台，因为要求半数以上；broker的设置我觉得一样就行，所以：kafka 配置5台broker,一个主题50个分区，ISR算上leader副本至少是3；

5

业余草
2019-09-16

没有完美的方案，我相信JMQ也是这样。高性能，高可用本身和一致性就是一个矛盾体。

4

极客雷
2020-03-21

RocketMQ不满足京东的使用场景吗?

作者回复: RocketMQ是很优秀的开源MQ，京东的JMQ“年纪”和RocketMQ是差不多的。在当年还没有RocketMQ的时候，其它的MQ又满足不了需求，所以就诞生了JMQ。

共 2 条评论>

3

二货

2020-05-02

所有的 ISR 节点都宕机了，分区就无法提供服务了。你也可以选择配置成让分区继续提供服务，这样只要有一个节点还活着，就可以提供服务
老师，这里ISR节点都宕机了，分区为啥还是正常的，ISR不就是分区副本吗

作者回复: ISR只是分区的一部分副本，不是全部。

比如，一个分区3副本，ISR可以配置为2。

共 2 条评论>

2

夏目

2020-01-09

老师，这种不是同步方式吗？-- Kafka 在写入消息的时候，采用的也是异步复制的方式。消息在写入到主节点之后，并不会马上返回写入成功，而是等待足够多的节点都复制成功后再返回。

作者回复: 从请求响应角度来说，是同步请求。
实现方式上，采用的是异步方式来实现的。

2

饭粒

2019-11-16

请教下老师，kafka 使用时是一个节点对应一个 broker 吗?然后 broker 作为分区副本容器存放不同主题-分区的主/从副本。

作者回复: 是这样的。

2

再见理想

2022-04-08

集群模式下，需要通过节点间的消息复制保证消息的一致性。
生产者将消息发送到主节点后，主节点需要将消息复制到从节点后，返回生产者发送成功。极端情况下，主节点将消息复制到所有从节点后，才返回，这样保证了集群的可用性和数据一致性，但是却大大的降低了性能。而主节点不复制消息自己将消息保存成功后就返回，那么就会得到高性能，降低了数据一致性和可用性。
rocketmq提供了两种方式，1，异步复制，主节点保存消息后，返回成功，异步将消息复制到从节点，如果主节点宕机，集群将无法收到消息，但消费者可以自动转移消费从节点的消息。这样保证了系统的高性能和数据一致性，但丢失了一部分可用性。 这个方案的改进版本是，部署多对主从节点，主题队列均匀分布在多个主节点上，当一个主节点宕机时，生产者可以更换主节点继续发送消息，但是消息顺序无法保证。Dleger方案集群可以自动选举主节点，消息复制半数以上节点时，返回成功。当集群节点不足半数时，无法提供服务。


1

Tesla

2020-03-08

老师好，请问旧的复制模式中，无法保证消息的严格顺序，是因为用了异步的方式同步消息，从节点接收到的顺序会收到网络影响不一定是哪个消息先到达吗？那可不可以给每个消息一个序号，从节点用一个队列来保存严格顺序的消息，用一个排序的数组来保存乱序的消息，只有当sortArrary的一部分能与队列合并为严格顺序的消息时，再把sortArrary的消息取出push到队列呢？

> 作者回复: 这个方法很难实现，需要考虑任何一个节点都有可能故障的情况，这个sortArray和序号都不好处理。


1

凡

2020-02-25

RockerMQ传统方式的的异步复制会出现数据丢失啊，比如当消息写入到主节点，返回成功，但是这时候还未复制到从节点，主机挂掉，消费者切换到从节点后读不到这些消息

> 作者回复: 是这样的。 但是数据也没丢，可以等主节点恢复后继续消费。
>
> 除非主节点磁盘坏了，那就真的丢数据了。

共 3 条评论>

1

成立-Charlie
2019-11-05

老师, 关于Kafka的高可用性我有一点迷惑。因为kafka是使用zookeeper作为其集群服务的协调服务器，Zookeeper采用超过半数可用的原则，3台Zookeeper集群只允许一台宕机。那Kafka却可以实现只有一台存活，仍然可以提供服务，这是如何实现的呢，Kafka可以脱离Zookeeper工作吗。谢谢!

作者回复: 你这些问题在第24节课中，会有详细的说明。

1

WL
2019-10-13

请问一下老师，我看RocketMQ官微上说DLedger只做一件事就是Commitlog，上面用Etcd对Raft协议的实现做对比，说Etcd对于Raft的实现时StateMachine+CommitLog的方式，CommitLog记录日志和操作记录，StateMachine通过操作记录构建出来的系统状态，请问一下在DLedger中的系统状态是怎样判定的?

作者回复: Dledger中只做Log复制，它没有状态，或者说它的状态就是Log，所以不需要状态机。

1

Ric
2023-03-27 来自广东

可以画几张图的，一图胜千言

Geek_66be20
2021-07-07

玥哥，我理解rocketMQ的老版本消息复制类似于JMQ2的方式，kafka的消息复制更像是JMQ4的方式。思考题，是brokerA上面有队列1的主、队列2的从、队列3的从。brokerB上面有队列1的从、队列2的主、队列3的从，brokerC上面有队列1的从、队列2的从、队列3的主。这种架构对吗?

语法糖不甜

2021-03-25

老师您好，我想问一下消息复制是把队列复制过去还是把队列里的消息复制过去。如果是把消息从队列a复制到队列b，队列a所在的节点宕机了，顺序消息是怎么去队列b找消息的那？


Warder

2021-02-01

"在需要保证消息严格顺序的场景下，由于在主题层面无法保证严格顺序，所以必须指定队列来发送消息，对于任何一个队列，它一定是落在一组特定的主从节点上，如果这个主节点宕机，其他的主节点是无法替代这个主节点的，否则就无法保证严格顺序"，这里是指当写入消息的时候只能往指定的一个主节点写入，因为还没写完也就没有开始复制，所以其他主节点的内容和当前主节点内容不一样？所以不能替换？