05 | 聊聊Kafka的版本号

2019-06-13 胡夕

你好，我是胡夕。今天我想和你聊聊如何选择Kafka版本号这个话题。今天要讨论的内容实在是太重要了，我觉得它甚至是你日后能否用好Kafka的关键。

上一期我介绍了目前流行的几种Kafka发行版，其实不论是哪种Kafka，本质上都内嵌了最核心的Apache Kafka，也就是社区版Kafka，那今天我们就来说说Apache Kafka版本号的问题。在开始之前，我想强调一下后面出现的所有“版本”这个词均表示Kafka具体的版本号，而非上一篇中的Kafka种类，这一点切记切记！

那么现在你可能会有这样的疑问：我为什么需要关心版本号的问题呢？直接使用最新版本不就好了吗？当然了，这的确是一种有效的选择版本的策略，但我想强调的是这种策略并非在任何场景下都适用。如果你不了解各个版本之间的差异和功能变化，你怎么能够准确地评判某Kafka版本是不是满足你的业务需求呢？因此在深入学习Kafka之前，花些时间搞明白版本演进，实际上是非常划算的一件事。

Kafka版本命名

当前Apache Kafka已经迭代到2.2版本，社区正在为2.3.0发版日期进行投票，相信2.3.0也会马上发布。但是稍微有些令人吃惊的是，很多人对于Kafka的版本命名理解存在歧义。比如我们在官网上下载Kafka时，会看到这样的版本：

- Binary downloads:
  - Scala 2.11 - kafka_2.11-2.2.1.tgz (asc, sha512)
  - Scala 2.12 - kafka_2.12-2.2.1.tgz (asc, sha512)

于是有些同学就会纳闷，难道Kafka版本号不是2.11或2.12吗？其实不然，前面的版本号是编译Kafka源代码的Scala编译器版本。Kafka服务器端的代码完全由Scala语言编写，Scala同时支持面向对象编程和函数式编程，用Scala写成的源代码编译之后也是普通的“.class”文件，因此我们说Scala是JVM系的语言，它的很多设计思想都是为人称道的。

事实上目前Java新推出的很多功能都是在不断向Scala语言靠近罢了，比如Lambda表达式、函数式接口、val变量等。一个有意思的事情是，Kafka新版客户端代码完全由Java语言编写，于是有些人展开了“Java VS Scala”的大讨论，并从语言特性的角度尝试分析Kafka社区为什么放弃Scala转而使用Java重写客户端代码。其实事情远没有那么复杂，仅仅是因为社区来了一批Java程序员而已，而以前老的Scala程序员隐退罢了。可能有点跑题了，但不管怎样我依然建议你有空去学学Scala语言。

回到刚才的版本号讨论。现在你应该知道了对于kafka-2.11-2.1.1的提法，真正的Kafka版本号实际上是2.1.1。那么这个2.1.1又表示什么呢？前面的2表示大版本号，即Major Version；中间的1表示小版本号或次版本号，即Minor Version；最后的1表示修订版本号，也就是Patch号。Kafka社区在发布1.0.0版本后特意写过一篇文章，宣布Kafka版本命名规则正式从4位演进到3位，比如0.11.0.0版本就是4位版本号。

坦率说，这里我和社区的意见是有点不同的。在我看来像0.11.0.0这样的版本虽然有4位版本号，但其实它的大版本是0.11，而不是0，所以如果这样来看的话Kafka版本号从来都是由3个部分构成，即“大版本号 - 小版本号 - Patch号”。这种视角可以统一所有的Kafka版本命名，也方便我们日后的讨论。我们来复习一下，假设碰到的Kafka版本是0.10.2.2，你现在就知道了它的大版本是0.10，小版本是2，总共打了两个大的补丁，Patch号是2。

Kafka版本演进

Kafka目前总共演进了7个大版本，分别是0.7、0.8、0.9、0.10、0.11、1.0和2.0，其中的小版本和Patch版本很多。哪些版本引入了哪些重大的功能改进？关于这个问题，我建议你最好能做到如数家珍，因为这样不仅令你在和别人交谈Kafka时显得很酷，而且如果你要向架构师转型或者已然是架构师，那么这些都是能够帮助你进行技术选型、架构评估的重要依据。

我们先从0.7版本说起，实际上也没什么可说的，这是最早开源时的“上古”版本了，以至于我也从来都没有接触过。这个版本只提供了最基础的消息队列功能，甚至连副本机制都没有，我实在

想不出有什么理由你要使用这个版本，因此一旦有人向你推荐这个版本，果断走开就好了。

Kafka从0.7时代演进到0.8之后正式引入了**副本机制**，至此Kafka成为了一个真正意义上完备的分布式高可靠消息队列解决方案。有了副本备份机制，Kafka就能够比较好地做到消息无丢失。那时候生产和消费消息使用的还是老版本的客户端API，所谓的老版本是指当你用它们的API开发生产者和消费者应用时，你需要指定ZooKeeper的地址而非Broker的地址。

如果你现在尚不能理解这两者的区别也没关系，我会在专栏的后续文章中详细介绍它们。老版本客户端有很多的问题，特别是生产者API，它默认使用同步方式发送消息，可以想见其吞吐量一定不会太高。虽然它也支持异步的方式，但实际场景中可能会造成消息的丢失，因此0.8.2.0版本社区引入了**新版本Producer API**，即需要指定Broker地址的Producer。

据我所知，国内依然有少部分用户在使用0.8.1.1、0.8.2版本。**我的建议是尽量使用比较新的版本。如果你不能升级大版本，我也建议你至少要升级到0.8.2.2这个版本，因为该版本中老版本消费者API是比较稳定的。另外即使你升到了0.8.2.2，也不要使用新版本Producer API，此时它的Bug还非常多。**

时间来到了2015年11月，社区正式发布了0.9.0.0版本。在我看来这是一个重量级的大版本更迭，0.9大版本增加了基础的安全认证/权限功能，同时使用Java重写了新版本消费者API，另外还引入了Kafka Connect组件用于实现高性能的数据抽取。如果这么多眼花缭乱的功能你一时无暇顾及，那么我希望你记住这个版本的另一个好处，那就是**新版本Producer API在这个版本中算比较稳定了**。如果你使用0.9作为线上环境不妨切换到新版本Producer，这是此版本一个不太为人所知的优势。但和0.8.2引入新API问题类似，不要使用新版本Consumer API，因为Bug超多的，绝对用到你崩溃。即使你反馈问题到社区，社区也不会管的，它会无脑地推荐你升级到新版本再试试，因此千万别用0.9的新版本Consumer API。对于国内一些使用比较老的CDH的创业公司，鉴于其内嵌的就是0.9版本，所以要格外注意这些问题。

0.10.0.0是里程碑式的大版本，因为该版本**引入了Kafka Streams**。从这个版本起，Kafka正式升级成分布式流处理平台，虽然此时的Kafka Streams还基本不能线上部署使用。0.10大版本包含两个小版本：0.10.1和0.10.2，它们的主要功能变更都是在Kafka Streams组件上。如果你把Kafka用作消息引擎，实际上该版本并没有太多的功能提升。不过在我的印象中自0.10.2.2版本起，新版本Consumer API算是比较稳定了。**如果你依然在使用0.10大版本，我强烈建议你至少升级到0.10.2.2然后使用新版本Consumer API。还有个事情不得不提，0.10.2.2修复了一个可能导致Producer性能降低的Bug。基于性能的缘故你也应该升级到0.10.2.2。**

在2017年6月，社区发布了0.11.0.0版本，引入了两个重量级的功能变更：一个是提供幂等性Producer API以及事务（Transaction） API；另一个是对Kafka消息格式做了重构。

前一个好像更加吸引眼球一些，毕竟Producer实现幂等性以及支持事务都是Kafka实现流处理结果正确性的基石。没有它们，Kafka Streams在做流处理时无法向批处理那样保证结果的正确

性。当然同样是由于刚推出，此时的事务API有一些Bug，不算十分稳定。另外事务API主要是为Kafka Streams应用服务的，实际使用场景中用户利用事务API自行编写程序的成功案例并不多见。

第二个重磅改进是消息格式的变化。虽然它对用户是透明的，但是它带来的深远影响将一直持续。因为格式变更引起消息格式转换而导致的性能问题在生产环境中屡见不鲜，所以你一定要谨慎对待0.11版本的这个变化。不得不说的是，这个版本中各个大功能组件都变得非常稳定了，国内该版本的用户也很多，应该算是目前最主流的版本之一了。也正是因为这个缘故，社区为0.11大版本特意推出了3个Patch版本，足见它的受欢迎程度。我的建议是，如果你对1.0版本是否适用于线上环境依然感到困惑，那么至少将你的环境升级到0.11.0.3，因为这个版本的消息引擎功能已经非常完善了。

最后我合并说下1.0和2.0版本吧，因为在我看来这两个大版本主要还是Kafka Streams的各种改进，在消息引擎方面并未引入太多的重大功能特性。Kafka Streams的确在这两个版本有着非常大的变化，也必须承认Kafka Streams目前依然还在积极地发展着。如果你是Kafka Streams的用户，至少选择2.0.0版本吧。

去年8月国外出了一本书叫Kafka Streams in Action（中文版：《Kafka Streams实战》），它是基于Kafka Streams 1.0版本撰写的。最近我用2.0版本去运行书中的例子，居然很多都已经无法编译了，足见两个版本变化之大。不过如果你在意的依然是消息引擎，那么这两个大版本都是适合于生产环境的。

最后还有个建议，不论你用的是哪个版本，都请尽量保持服务器端版本和客户端版本一致，否则你将损失很多Kafka为你提供的性能优化收益。

小结

我希望现在你对如何选择合适的Kafka版本能做到心中有数了。每个Kafka版本都有它恰当的使用场景和独特的优缺点，切记不要一味追求最新版本。事实上我周围的很多工程师都秉承这样的观念：不要成为最新版本的“小白鼠”。了解了各个版本的差异之后，我相信你一定能够根据自己的实际情况作出最正确的选择。

开放讨论

如何评估Kafka版本升级这件事呢？你和你所在的团队有什么独特的见解？

欢迎你写下自己的思考或疑问，我们一起讨论 。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

精选留言

少林寺三毛

5

[]

2019-06-13

小头针

3

胡老师讲的这个版本，一下戳到我的痛处。讲一下我在生产环境中遇到的Kafka版本带来的坑。我参与到项目中一年，运行的版本是0.10.0.1，前半年还算稳定，偶尔出现进程假死问题。但是慢慢的生产环境数据量增加，假死频发，导致客户数据丢失，问题很严重。但是一直又没有证据证明这个版本确实存在问题，虽然官网上有提到，但是我们老大的意思还是要找到根本原因。妥所以开始对生产环境的进程进程监控，确实监控出此版本存在线程死锁问题。（妥从一个开发转变为现场运维人员）然后研究官网bug列表，选出0.11.0.3这个版本，至今稳定运行。

再顺便讲一下选择这门课的原因，虽然项目已经高一段落，但是在整个项目过程中，一直处于哪里不会点哪里的状态，感觉一直还是没有真正的掀开Kafka的面纱。所以想系统的学习一下，听了几节课，之前有些知其然而不知其所以然的内容，似乎开始有点茅塞顿开了。

2019-06-13

南辕北辙

2

记得在刚开始学kafka写demo时，找到了kafka.producer.Producer,以及apache.kafka...KafkaProducer，还以为只是2种不同的实现方式，后来在老师的书上才得知这完全是2个版本。针对今天讨论的版本差异，书上也做了很好了总结。

现在看来比较难理解的就是客户端的版本与服务端版本的兼容问题，与之前的各种技术还是有点差异的，并不是一味的较新客户端就完事，kafka中还有一种请求版本号的存在。
图片为书中版本对比
https://raw.githubusercontent.com/DarkerPWQ/picgo/master/img/Kafka%E7%89%88%E6%9C%AC%E5%8F%98%E8%BF%81.png

2019-06-13

作者回复

嗯嗯，请求版本号偏底层的设计了，一般用户用不到。其实客户端和服务器端版本的差异很大一部分也是请求版本号的差异

2019-06-13

开水 2

目前用的是hdp2.4.2内嵌版本。应该是apache版本的0.8.2.0。遇到很多问题都很难找到解决方法。比如前几天遇到了replicaFetcherThread oom的问题，网上根本找不到什么正经的解释。但又不能一味的调高jvm参数。老大说现在生产稳定就行了，暂时不要升级了，改代码耗时切面对的问题未知。期待老大能看到这篇文章，升级到0.11.0.3。

2019-06-13

作者回复

嗯嗯，可以查一下ZooKeeper中是否存在大量session超时的情况。不过还是建议升级吧，听着很像是一个已知的bug。如果暂时不能升级，可以尝试调低replica.fetch.max.bytes的值试试。

2019-06-13

QQ怪 2

的确在工作中遇到了kafka版本不同导致消息格式不兼容问题，后来服务端和客户端统一版本号才解决，

2019-06-13

King Yao 1

我是一名运维，在维护工作环境维护几十台Kafka。最近打算扩容。我有三个问题请教：
1.kafka如何做压力测试，它的参考主要指标是什么，比如QPS,最大连接数，延迟等等。
2.扩容如何做到平滑扩容，不影响原业务
3.kafka有什么好的监控软件。

2019-06-13

作者回复

1. Kafka提供了命令行脚本可以执行producer和consumer的性能测试，主要指标还是TPS，延时

2. 增加broker很简单，也不会对现有业务有影响。关键是做好迁移计划—比如避开业务高峰时刻，如果迁移对业务影响最小

2019-06-15

Geek_89bbab 1

原来kafka在演进过程中这么坑啊

2019-06-13

疯琴

1

美音最正的老师没有之一。有两个问题：1 文章前面说某些旧版本的服务端不适用新版本的客户端，文末又说二者应该保持一致，感觉是矛盾的，应该是我没理解清楚，还请说明 2 服务端版本靠编号就可以识别，而客户端是怎么定义新旧版本的呢？谢谢。

2019-06-13

作者回复

1. 其实我也没太明白您的意思：）“旧版本的服务端不适用新版本的客户端 所以建议保持一致”，不是挺自然的结论吗。。。。

2. 客户端也有版本号，和broker端是一样的。比如Java客户端，如果我们使用Gradle的话，就类似于这样：

compile group: 'org.apache.kafka', name: 'kafka-clients', version: '2.2.1'

2019-06-13

mayunyong

0

棒

2019-06-15

focusOn

0

未开启Kafka Security， 全局一个 Producer实例， OOM 是业务代码导致非 Kafka 插入消息导致

2019-06-15

作者回复

哦哦，那可以从优化业务代码入手：）

2019-06-15

focusOn

0

应用中只存在一个 Kafka Producer

2019-06-15

focusOn

0

使用 kafka 2.0.0版本 在整个应用中只创建了一个 Kafka Producer , 每次调用 包装过的同步静态方法插入消息，其中插入静态方法调用了 Kafka Producer 的 send 方法，应用 OOM 情况下，Kafka 的 Producer 发送消息的 IoThread 线程被关闭，导致消息不能正常写入，这个是一个 bug 吗？老师？

2019-06-14

作者回复

最好多给一些信息，比如OOM是哪部分内存溢出了。另外是否启用了Kafka Security等，还有就是是否频繁地创建KafkaProducer实例？

2019-06-15

摇山樵客™

0

版本号：
大 + 小 + patch

0.7版本:
只有基础消息队列功能，无副本；打死也不使用

0.8版本:
增加了副本机制，新的producer API；建议使用0.8.2.2版本；不建议使用0.8.2.0之后的producer API

0.9版本:
增加权限和认证，新的consumer API，Kafka Connect功能；不建议使用consumer API;

0.10版本:
引入Kafka Streams功能，bug修复；建议版本0.10.2.2；建议使用新版consumer API

0.11版本:
producer API幂等，事物API，消息格式重构；建议版本0.11.0.3；谨慎对待消息格式变化

1.0和2.0版本:
Kafka Streams改进；建议版本2.0;

江湖经验：不要成为最新版本的小白鼠

谢谢老师的讲解，收工。

2019-06-14

清晨吼于林 0

老师您好，打扰您了，不过还是想问一下您：
kafka的时间轮里面，第一层的时间轮，默认tick一次是1ms，轮长度为20，一个轮子转下来也就20ms，那为什么ExpiredOperationReaper的时间间隔是200ms啊？

2019-06-14

作者回复

嗯嗯，这就是一个经验值。目前Kafka延时任务的超时多是以秒级，或至少是压秒级别的。如果按照第一层时间轮的间隔驱动会有一点低效。

2019-06-15

Chloe 0

只想简单点赞[]，看了标题觉得版本号没有什么好聊了，看到才知道Kaka一路走来充满荆棘啊。感谢感谢！

2019-06-14

燕子上 0

一般的，版本都会选择比最新低一个minor version的版本。除非团队有人精通最新版(解决了旧版本的哪些痛点，没有增加其他新问题)，才会使用。当然我说的是所有的版本选择。

2019-06-14

闭门造车 0

升级的话考虑的无非就是新版本的稳定性比线上的版本好或者有需要的新功能，在升级前需要进行测试，评估风险，保证线上稳定这是关键

2019-06-13

bee 0

请问胡老师， RESTful的幂等性是无论调用多少次都不会有不同结果的HTTP 方法。也就是说不会去改变资源。这里的幂等性虽然是确保了单个partition的数据不会重复，但是更改了资源。这样说会有冲突吗？

2019-06-13

作者回复

这里的幂等性也没有你所谓的更改资源。当producer发送了一条重复消息时Broker端会拒绝接收，而不是在后面自己做去重

2019-06-15

永光 0

我理解，对于学习或者新环境使用直接用最新就好了（用于生产就用最新的稳定版本），至于生产上已经再用的，升级到对应大版本的最新Patch版本就好了。貌似对于所有软件使用都是这样，哈哈哈

2019-06-13

黑色 0

我们目前用的是kafka1.0x, 用的spark版本号是2.1.3,spark-kafka依赖jar目前最新的是0.10.x,这样是不是服务端与客户端的版本不一样了？

2019-06-13

作者回复

是的，而且是相差很大的版本。建议关注broker端CPU使用率：）

2019-06-13