

07-最最最重要的集群参数配置（上）

你好，我是胡夕。今天我想和你聊聊最最最重要的Kafka集群配置。我这里用了3个“最”字并非哗众取宠，而是因为有些配置的重要性并未体现在官方文档中，并且从实际表现看，很多参数对系统的影响要比从文档上看更加明显，因此很有必要集中讨论一下。

我希望通过两期内容把这些重要的配置讲清楚。严格来说这些配置并不单单指Kafka服务器端的配置，其中既有Broker端参数，也有主题（后面我用我们更熟悉的Topic表示）级别的参数、JVM端参数和操作系统级别的参数。下面我先从Broker端参数说起。

Broker端参数

目前Kafka Broker提供了近200个参数，这其中绝大部分参数都不用你亲自过问。当谈及这些参数的用法时，网上的文章多是罗列出一些常见的参数然后一个一个地给出它们的定义，事实上我以前写文章时也是这么做的。不过今天我打算换个方法，按照大的用途类别一组一组地介绍它们，希望可以更有针对性，也更方便你记忆。

首先Broker是需要配置存储信息的，即Broker使用哪些磁盘。那么针对存储信息的重要参数有以下这么几个：

- `log.dirs`：这是非常重要的参数，指定了Broker需要使用的若干个文件目录路径。要知道这个参数是没有默认值的，这说明什么？这说明它必须由你亲自指定。
- `log.dir`：注意这是`dir`，结尾没有`s`，说明它只能表示单个路径，它是补充上一个参数用的。

这两个参数应该怎么设置呢？很简单，你只要设置`log.dirs`，即第一个参数就好了，不要设置`log.dir`。而且更重要的是，在线上生产环境中一定要为`log.dirs`配置多个路径，具体格式是一个CSV格式，也就是用逗号分隔的多个路径，比如`/home/kafka1,/home/kafka2,/home/kafka3`这样。如果有条件的话你最好保证这些目录挂载到不同的物理磁盘上。这样做有两个好处：

- 提升读写性能：比起单块磁盘，多块物理磁盘同时读写数据有更高的吞吐量。
- 能够实现故障转移：即Failover。这是Kafka 1.1版本新引入的强大功能。要知道在以前，只要Kafka Broker使用的任何一块磁盘挂掉了，整个Broker进程都会关闭。但是自1.1开始，这种情况被修正了，坏掉的磁盘上的数据会自动地转移到其他正常的磁盘上，而且Broker还能正常工作。还记得上一期我们关于Kafka是否需要使用RAID的讨论吗？这个改进正是我们舍弃RAID方案的基础：没有这种Failover的话，我们只能依靠RAID来提供保障。

下面说说与ZooKeeper相关的设置。首先ZooKeeper是做什么的呢？它是一个分布式协调框架，负责协调管理并保存Kafka集群的所有元数据信息，比如集群都有哪些Broker在运行、创建了哪些Topic，每个Topic都有多少分区以及这些分区的Leader副本都在哪些机器上等信息。

Kafka与ZooKeeper相关的最重要的参数当属`zookeeper.connect`。这也是一个CSV格式的参数，比如我可以指定它的值为`zk1:2181,zk2:2181,zk3:2181`。2181是ZooKeeper的默认端口。

现在问题来了，如果我让多个Kafka集群使用同一套ZooKeeper集群，那么这个参数应该怎么设置呢？这时候`chroot`就派上用场了。这个`chroot`是ZooKeeper的概念，类似于别名。

如果你有两套Kafka集群，假设分别叫它们kafka1和kafka2，那么两套集群的`zookeeper.connect`参数可以这样指定：`zk1:2181,zk2:2181,zk3:2181/kafka1`和`zk1:2181,zk2:2181,zk3:2181/kafka2`。切记`chroot`只需要写一次，而且是加到最后的。我经常碰到有人这样指定：`zk1:2181/kafka1,zk2:2181/kafka2,zk3:2181/kafka3`，这样的格式是不对的。

第三组参数是与Broker连接相关的，即客户端程序或其他Broker如何与该Broker进行通信的设置。有以下三个参数：

- `listeners`：学名叫监听器，其实就是告诉外部连接者要通过什么协议访问指定主机名和端口开放的Kafka服务。
- `advertised.listeners`：和`listeners`相比多了个`advertised`。Advertised的含义表示宣称的、公布的，就是说这组监听器是Broker用于对外发布的。
- `host.name/port`：列出这两个参数就是想说你把它们忘掉吧，压根不要为它们指定值，毕竟都是过期的参数了。

我们具体说说监听器的概念，从构成上来说，它是若干个逗号分隔的三元组，每个三元组的格式为<协议名称, 主机名, 端口号>。这里的协议名称可能是标准的名字，比如PLAINTEXT表示明文传输、SSL表示使用SSL或TLS加密传输等；也可能是你自己定义的协议名字，比如CONTROLLER: //localhost:9092。

一旦你自己定义了协议名称，你必须还要指定`listener.security.protocol.map`参数告诉这个协议底层使用了哪种安全协议，比如指定`listener.security.protocol.map=CONTROLLER:PLAINTEXT`表示CONTROLLER这个自定义协议底层使用明文不加密传输数据。

至于三元组中的主机名和端口号则比较直观，不需要做过多解释。不过有个事情你还是要注意一下，经常有人问主机名这个设置中我到底使用IP地址还是主机名。**这里我给出统一的建议：最好全部使用主机名，即Broker端和Client端应用配置中全部填写主机名。** Broker源代码中也使用的是主机名，如果你在有些地方使用了IP地址进行连接，可能会发生无法连接的问题。

第四组参数是关于Topic管理的。我来讲讲下面这三个参数：

- `auto.create.topics.enable`：是否允许自动创建Topic。
- `unclean.leader.election.enable`：是否允许Unclean Leader选举。
- `auto.leader.rebalance.enable`：是否允许定期进行Leader选举。

我还是一个个说。

`auto.create.topics.enable`参数我建议最好设置成`false`，即不允许自动创建Topic。在我们的线上环境里面有很多名字稀奇古怪的Topic，我想大概都是因为该参数被设置成了`true`的缘故。

你可能有这样的经历，要为名为test的Topic发送事件，但是不小心拼写错误了，把test写成了tst，之后启动了生产者程序。恭喜你，一个名为tst的Topic就被自动创建了。

所以我一直相信好的运维应该防止这种情形的发生，特别是对于那些大公司而言，每个部门被分配的Topic应该由运维严格把控，决不能允许自行创建任何Topic。

第二个参数`unclean.leader.election.enable`是关闭Unclean Leader选举的。何谓Unclean？还记得Kafka有多个副本这件事吗？每个分区都有多个副本来提供高可用。在这些副本中只能有一个副本对外提供服务，即所谓的Leader副本。

那么问题来了，这些副本都有资格竞争Leader吗？显然不是，只有保存数据比较多的那些副本才有资格竞选，那些落后进度太多的副本没资格做这件事。

好了，现在出现这种情况了：假设那些保存数据比较多的副本都挂了怎么办？我们还要不要进行Leader选举了？此时这个参数就派上用场了。

如果设置成false，那么就坚持之前的原则，坚决不能让那些落后太多的副本竞选Leader。这样做的后果是这个分区就不可用了，因为没有Leader了。反之如果是true，那么Kafka允许你从那些“跑得慢”的副本中选一个出来当Leader。这样做的后果是数据有可能就丢失了，因为这些副本保存的数据本来就不全，当了Leader之后它本人就变得膨胀了，认为自己的数据才是权威的。

这个参数在最新版的Kafka中默认就是false，本来不需要我特意提的，但是比较搞笑的是社区对这个参数的默认值来来回回改了好几版了，鉴于我不知道你用的是哪个版本的Kafka，所以建议你还是显式地把它设置成false吧。

第三个参数`auto.leader.rebalance.enable`的影响貌似没什么人提，但其实对生产环境影响非常大。设置它的值为true表示允许Kafka定期地对一些Topic分区进行Leader重选举，当然这个重选举不是无脑进行的，它要满足一定的条件才会发生。严格来说它与上一个参数中Leader选举的最大不同在于，它不是选Leader，而是换Leader！比如Leader A一直表现得很好，但若`auto.leader.rebalance.enable=true`，那么有可能一段时间后Leader A就要被强行卸任换成Leader B。

你要知道换一次Leader代价很高的，原本向A发送请求的所有客户端都要切换成向B发送请求，而且这种换Leader本质上没有任何性能收益，因此我建议你在生产环境中把这个参数设置成false。

最后一组参数是数据留存方面的，即：

- `log.retention.{hour|minutes|ms}`：这是个“三兄弟”，都是控制一条消息数据被保存多长时间。从优先级上来说ms设置最高、minutes次之、hour最低。
- `log.retention.bytes`：这是指定Broker为消息保存的总磁盘容量大小。
- `message.max.bytes`：控制Broker能够接收的最大消息大小。

先说这个“三兄弟”，虽然ms设置有最高的优先级，但是通常情况下我们还是设置hour级别的多一些，比如`log.retention.hour=168`表示默认保存7天的数据，自动删除7天前的数据。很多公司把Kafka当做存储来使用，那么这个值就要相应地调大。

其次是这个`log.retention.bytes`。这个值默认是-1，表明你想在这台Broker上保存多少数据都可以，至少在容量方面Broker绝对为你开绿灯，不会做任何阻拦。这个参数真正发挥作用的场景其实是在云上构建多租户的Kafka集群：设想你要做一个云上的Kafka服务，每个租户只能使用100GB的磁盘空间，为了避免有个“恶意”租户使用过多的磁盘空间，设置这个参数就显得至关重要了。

最后说说`message.max.bytes`。实际上今天我和你说过的重要参数都是指那些不能使用默认值的参数，这

个参数也是一样，默认的1000012太少了，还不到1KB。实际场景中突破1MB的消息都是屡见不鲜的，因此在线上环境中设置一个比较大的值还是比较保险的做法。毕竟它只是一个标尺而已，仅仅衡量Broker能够处理的最大消息大小，即使设置大一点也不会耗费什么磁盘空间的。

小结

再次强调一下，今天我和你分享的所有参数都是那些要修改默认值的参数，因为它们的默认值不适合一般的生产环境。当然，我并不是说其他100多个参数就不重要。事实上，在专栏的后面我们还会陆续提到其他的一些参数，特别是那些和性能息息相关的参数。所以今天我提到的所有参数，我希望作为一个最佳实践给你，可以有的放矢地帮助你规划和调整你的Kafka生产环境。

开放讨论

除了今天我分享的这些参数，还有哪些参数是你认为比较重要而文档中没有提及的？你曾踩过哪些关于参数配置的“坑”？欢迎提出来与我和大家一起讨论。

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。



Kafka 核心技术与实战

全面提升你的 Kafka 实战能力

胡夕
人人贷计算平台部总监
Apache Kafka Contributor



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言：

- 趙衍 2019-06-18 01:37:07
老师好！关于Unclean这个参数，将其设置为false之后，就意味着如果ISR内的所有broker都宕机，那么这个分区就不可用了。
刚好我前几天看到饶军在2013年的一次报告上讲到Kafka在CAP问题上的取舍，他说，因为Kafka是部署在一个DataCenter中的，而一个DataCenter很少会出现Partitioning的情况，所以Kafka放弃了分区容忍性。我想问的是，Kafka舍弃了分区容忍性这一点是否可以体现在社区默认将Unclean设置为false上呢？
附上报告的地址：<https://www.youtube.com/watch?v=XcvHmqmh16g>
关于CAP的取舍出现在21:50左右的地方。谢谢老师！
- 烟波 2019-06-18 01:23:39
生产者的ack配置

- 草帽路飞 2019-06-18 00:32:48

老师 advertised.listeners 这个配置能否再解释一下。感觉配置了 listeners之后就不用配置这个了呀？