08 | 最最最重要的集群参数配置（下）

2019-06-20 胡夕

今天我们继续来聊那些重要的Kafka集群配置，下半部分主要是Topic级别参数、JVM参数以及操作系统参数的设置。

在上一期中，我们讨论了Broker端参数设置的一些法则，但其实Kafka也支持为不同的Topic设置不同的参数值。当前最新的2.2版本总共提供了大约25个Topic级别的参数，当然我们也不必全部了解它们的作用，这里我挑出了一些最关键的参数，你一定要把它们掌握清楚。除了Topic级别的参数，我今天还会给出一些重要的JVM参数和操作系统参数，正确设置这些参数是搭建高性能Kafka集群的关键因素。

Topic级别参数

说起Topic级别的参数，你可能会有这样的疑问：如果同时设置了Topic级别参数和全局Broker参数，到底听谁的呢？哪个说了算呢？答案就是Topic级别参数会覆盖全局Broker参数的值，而每个Topic都能设置自己的参数值，这就是所谓的Topic级别参数。

举个例子说明一下，上一期我提到了消息数据的留存时间参数，在实际生产环境中，如果为所有Topic的数据都保存相当长的时间，这样做既不高效也无必要。更适当的做法是允许不同部门的Topic根据自身业务需要，设置自己的留存时间。如果只能设置全局Broker参数，那么势必要提取所有业务留存时间的最大值作为全局参数值，此时设置Topic级别参数把它覆盖，就是一个不错的选择。

下面我们依然按照用途分组的方式引出重要的Topic级别参数。从保存消息方面来考量的话，下面这组参数是非常重要的：

- retention.ms：规定了该Topic消息被保存的时长。默认是7天，即该Topic只保存最近7天的消息。一旦设置了这个值，它会覆盖掉Broker端的全局参数值。
- retention.bytes：规定了要为该Topic预留多大的磁盘空间。和全局参数作用相似，这个值通常在多租户的Kafka集群中会有用武之地。当前默认值是-1，表示可以无限使用磁盘空间。

上面这些是从保存消息的维度来说的。如果从能处理的消息大小这个角度来看的话，有一个参数是必须要设置的，即max.message.bytes。它决定了Kafka Broker能够正常接收该Topic的最大消息大小。我知道目前在很多公司都把Kafka作为一个基础架构组件来运行，上面跑了很多的业务数据。如果在全局层面上，我们不好给出一个合适的最大消息值，那么不同业务部门能够自行设定这个Topic级别参数就显得非常必要了。在实际场景中，这种用法也确实是非常常见的。

好了，你要掌握的Topic级别的参数就这么几个。下面我来说说怎么设置Topic级别参数吧。其实说到这个事情，我是有点个人看法的：我本人不太赞同那种做一件事情开放给你很多种选择的设计方式，看上去好似给用户多种选择，但实际上只会增加用户的学习成本。特别是系统配置，如果你告诉我只能用一种办法来做，我会很努力地把它学会；反之，如果你告诉我说有两种方法甚至是多种方法都可以实现，那么我可能连学习任何一种方法的兴趣都没有了。Topic级别参数的设置就是这种情况，我们有两种方式可以设置：

- 创建Topic时进行设置
- 修改Topic时设置

我们先来看看如何在创建Topic时设置这些参数。我用上面提到的retention.ms和max.message.bytes举例。设想你的部门需要将交易数据发送到Kafka进行处理，需要保存最近半年的交易数据，同时这些数据很大，通常都有几MB，但一般不会超过5MB。现在让我们用以下命令来创建Topic：

```
bin/kafka-topics.sh--bootstrap-serverlocalhost:9092--create--topictransaction--partitions1--replication-factor1--config
```

我们只需要知道Kafka开放了kafka-topics命令供我们来创建Topic即可。对于上面这样一条命令，请注意结尾处的--config设置，我们就是在config后面指定了想要设置的Topic级别参数。

下面看看使用另一个自带的命令kafka-configs来修改Topic级别参数。假设我们现在要发送最大值是10MB的消息，该如何修改呢？命令如下：

```
bin/kafka-configs.sh--zookeeperlocalhost:2181--entity-typetopics--entity-nametransaction--alter--add-configmax.m
```

总体来说，你只能使用这么两种方式来设置Topic级别参数。我个人的建议是，你最好始终坚持使用第二种方式来设置，并且在未来，Kafka社区很有可能统一使用kafka-configs脚本来调整Topic级别参数。

JVM参数

我在专栏前面提到过，Kafka服务器端代码是用Scala语言编写的，但终归还是编译成Class文件在JVM上运行，因此JVM参数设置对于Kafka集群的重要性不言而喻。

首先我先说说Java版本，我个人极其不推荐将Kafka运行在Java 6或7的环境上。Java 6实在是太过陈旧了，没有理由不升级到更新版本。另外Kafka自2.0.0版本开始，已经正式摒弃对Java 7的支持了，所以有条件的话至少使用Java 8吧。

说到JVM端设置，堆大小这个参数至关重要。虽然在后面我们还会讨论如何调优Kafka性能的问题，但现在我想无脑给出一个通用的建议：将你的JVM堆大小设置成6GB吧，这是目前业界比较公认的一个合理值。我见过很多人就是使用默认的Heap Size来跑Kafka，说实话默认的1GB有点小，毕竟Kafka Broker在与客户端进行交互时会在JVM堆上创建大量的ByteBuffer实例，Heap Size不能太小。

JVM端配置的另一个重要参数就是垃圾回收器的设置，也就是平时常说的GC设置。如果你依然在使用Java 7，那么可以根据以下法则选择合适的垃圾回收器：

- 如果Broker所在机器的CPU资源非常充裕，建议使用CMS收集器。启用方法是指定-XX:+UseCurrentMarkSweepGC。
- 否则，使用吞吐量收集器。开启方法是指定-XX:+UseParallelGC。

当然了，如果你已经在使用Java 8了，那么就用默认的G1收集器就好了。在没有任何调优的情况下，G1表现得要比CMS出色，主要体现在更少的Full GC，需要调整的参数更少等，所以使用G1就好了。

现在我们确定好了要设置的JVM参数，我们该如何为Kafka进行设置呢？有些奇怪的是，这个问题居然在Kafka官网没有被提及。其实设置的方法也很简单，你只需要设置下面这两个环境变量即可：

- KAFKA_HEAP_OPTS：指定堆大小。
- KAFKA_JVM_PERFORMANCE_OPTS：指定GC参数。

比如你可以这样启动Kafka Broker，即在启动Kafka Broker之前，先设置上这两个环境变量：

```
$> export KAFKA_HEAP_OPTS=--Xms6g  --Xmx6g
$> export  KAFKA_JVM_PERFORMANCE_OPTS= -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatiI
$> bin/kafka-server-start.sh config/server.properties
```

操作系统参数

最后我们来聊聊Kafka集群通常都需要设置哪些操作系统参数。通常情况下，Kafka并不需要设置太多的OS参数，但有些因素最好还是关注一下，比如下面这几个：

- 文件描述符限制
- 文件系统类型
- Swappiness
- 提交时间

首先是ulimit -n。我觉得任何一个Java项目最好都调整下这个值。实际上，文件描述符系统资源并不像我们想象的那样昂贵，你不用太担心调大此值会有什么不利的影响。通常情况下将它设置成一个超大的值是合理的做法，比如ulimit -n 1000000。还记得电影《让子弹飞》里的对话吗：“你和钱，谁对我更重要？都不重要，没有你对我很重要！”。这个参数也有点这么个意思。其实设置这个参数一点都不重要，但不设置的话后果很严重，比如你会经常看到“Too many open files”的错误。

其次是文件系统类型的选择。这里所说的文件系统指的是如ext3、ext4或XFS这样的日志型文件系统。根据官网的测试报告，XFS的性能要强于ext4，所以生产环境最好还是使用XFS。对了，最近有个Kafka使用ZFS的数据报告，貌似性能更加强劲，有条件的话不妨一试。

第三是swap的调优。网上很多文章都提到设置其为0，将swap完全禁掉以防止Kafka进程使用swap空间。我个人反倒觉得还是不要设置成0比较好，我们可以设置成一个较小的值。为什么呢？因为一旦设置成0，当物理内存耗尽时，操作系统会触发OOM killer这个组件，它会随机挑选一个进程然后kill掉，即根本不给用户任何的预警。但如果设置成一个比较小的值，当开始使用swap空间时，你至少能够观测到Broker性能开始出现急剧下降，从而给你进一步调优和诊断问题的时间。基于这个考虑，我个人建议将swappniess配置成一个接近0但不为0的值，比如1。

最后是提交时间或者说是Flush落盘时间。向Kafka发送数据并不是真要等数据被写入磁盘才会认为成功，而是只要数据被写入到操作系统的页缓存（Page Cache）上就可以了，随后操作系统根据LRU算法会定期将页缓存上的“脏”数据落盘到物理磁盘上。这个定期就是由提交时间来确定的，默认是5秒。一般情况下我们会认为这个时间太频繁了，可以适当地增加提交间隔来降低物理磁盘的写操作。当然你可能会有这样的疑问：如果在页缓存中的数据在写入到磁盘前机器宕机了，那岂不是数据就丢失了。的确，这种情况数据确实就丢失了，但鉴于Kafka在软件层面已经提供了多副本的冗余机制，因此这里稍微拉大提交间隔去换取性能还是一个合理的做法。

小结

今天我和你分享了关于Kafka集群设置的各类配置，包括Topic级别参数、JVM参数以及操作系统参数，连同上一篇一起构成了完整的Kafka参数配置列表。我希望这些最佳实践能够在你搭建Kafka集群时助你一臂之力，但切记配置因环境而异，一定要结合自身业务需要以及具体的测试来验证它们的有效性。

开放讨论

很多人争论Kafka不需要为Broker设置太大的堆内存，而应该尽可能地把内存留给页缓存使用。对此你是怎么看的？在你的实际使用中有哪些好的法则来评估Kafka对内存的使用呢？

欢迎写下你的思考和答案，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

丰富

13

G1是jdk9中默认的，jdk8还是需要显式指定的

2019-06-20

作者回复

嗯嗯，笔误了。多谢纠正：）

2019-06-20

aoe

8

ulimit -n这个参数说的太好了！如果不设置，单机在Centos7上几百的并发就报“Too many open files”了。网上搜索后设置成65535，用JMater压测单机也只能支撑到1000左右的并发，原来这个值可以设置到1000000！《Kafka权威指南》上说Kafka单机可以轻松处理300万并发；《响应式架构:消息模式Actor实现与Scala、Akka应用集成》上说Scala用Actor单机可以处理5000万并发。请问胡老师有没有推荐的Linux方面的书籍，可以详细解答ulimit -n参数、如何知道单台Linux机器可以处理的连接数上线？

我在mac笔记本上用Go开启了10万个goroutine，压测服务器，结果得到异常“Too many open files”，后来也修改了ulimit -65535，但也只能保证1万左右的请求正常，请问Mac上也是只要设置ulimit -n参数就可以将请求的连接数提升到上限吗？

2019-06-20

Xiao

4

帅气的胡老师，后边是否会将Kafka数据丢失和消息重复的场景以以及解决思路！

2019-06-20

作者回复

会有的，后面有防止消息丢失和重复消费，到时候一起讨论哈

2019-06-21

吃饭饭

2

Java8默认的新生代垃圾回收器是：UseParallelGC，可以用-XX:+PrintCommandLineFlags -version查看，还有如果显示指定 -XX:+UseCurrentMarkSweepGC 的话，会默认开启 -XX:+UseParallelGC

2019-06-20

作者回复

嗯嗯，这点笔误了。Java 9默认的GC收集器才是G1。Java 8应该还是吞吐量收集器。

2019-06-20

黎

1

老师的美式英语发音真标准

2019-06-29

赌神很低调

1

胡老师，kafka认为写入成功是指写入页缓存成功还是数据刷到磁盘成功算成功呢？还是上次刷盘宕机失败的问题，页缓存的数据如果刷盘失败，是不是就丢了？这个异常会不会响应给生产者让其重发呢？

2019-06-24

作者回复

写入到页缓存即认为成功。如果在flush之前机器就宕机了，的确这条数据在broker上就算丢失了。producer端表现如何取决于acks的设定。如果是acks=1而恰恰是leader broker在flush前宕机，那么的确有可能消息就丢失了，而且producer端不会重发—因为它认为是成功了。

2019-06-26

saup007

1

修改 Topic 级 max.message.bytes，还要考虑以下两个吧？
还要修改 Broker的 replica.fetch.max.bytes 保证复制正常
消费还要修改配置 fetch.message.max.bytes

2019-06-21

作者回复

是的，您考虑得很全面：）

2019-06-21

Hello world

1

老师说的无脑配置给jvm heap 6G大小，这应该也看机器的吧，现在机器的内存也越来越大，我们这的机器都是64G 内存，配了16G的heap，老师觉得可以优化吗

2019-06-20

作者回复

虽然无脑推荐6GB，但绝不是无脑推荐>6GB。一个16GB的堆Full GC一次要花多长时间啊，所以我觉得6GB可以是一个初始值，你可以实时监控堆上的live data大小，根据这个值调整heap size。只是因为大内存就直接调整到16GB，个人觉得不可取。

另外堆越小留给页缓存的空间也就越大，这对Kafka是好事啊。

2019-06-20

陈华应

1

是不是要根据服务器性能来设置呢？比如机器的总内存是多少，单运行kafka的话，再针对性的按比例设置jvm内存大小呢？无脑设置6g也是在一定规格的服务器配置的情况下吧

2019-06-20

作者回复

有道理。这里给出的6GB一般对那些很多的生产服务器而言的。如果只有8GB的服务器，自然不建议分配这么大的heap size。

另外监控堆占用也是sizing的一个好办法，特别是监控Full GC之后的live data大小。

2019-06-20

theivanxu

1

最近环境中有一台3G堆内存的节点在某个topic handle request的时候一直OOM，调整到5G重启后恢复正常，很想知道如何评判堆内存大小设置的标准。

2019-06-20

作者回复

没有通用的标准，只有一个最佳实践值：6GB。最好还是监控一下实时的堆大小，特别是GC之后的live data大小，通常将heapsize设置成其1.5~2倍就足以了

2019-06-20

刘朋

1

系统会根据LRU算法定期将页缓存上的 脏 数据落盘到物理磁盘上. 这个定期就是由提交时间来确定的,默认是5秒.

这个时间如何设置？ 是内核参数吗？

2019-06-20

作者回复

不算内核参数，是文件系统的参数。你可以查询一下文件系统手册。比如ext4就是commit=Nseconds这样设置

2019-06-20

风中花 1

给老师点个赞！按时发布！辛苦

2019-06-20

yswang 0

老师您好，Kafka 下的 topic 有没有数量限制？公司一个使用Kafka的团队说，他们查询网络资料了解到 Kafka topic 超过 64 个时，会影响读写性能。

请问老师，这是真的吗？

2019-07-01

其实我很屌 0

老师，我用的kafka0.11.0，在新建topic的时候，命令上也是支持自定义topic价格参数的，但我发现我设置的log.segment.bytes参数并没有生效，还是用的集群配置，想知道topic的个性化配置，是从哪个版本提供的？是不是一开始还有缺陷？麻烦老师帮忙解答，谢谢

2019-06-28

作者回复

topic级别参数是segment.bytes。应该没有缺陷。主要是它和你想的原理不太一样。看看我写的这篇：https://www.cnblogs.com/huxi2b/p/8042099.html

2019-07-01

有钱的包子 0

G1 收集器的开启码

2019-06-26

小头针 0

胡老师，在本课程最后留的问题，又成功的引起了我的注意，我曾经因为kafka假死，不知原因为何，而尝试过设置Broker的内存为（32G/256G），然而进程假死更加频繁（后面检测是那个版本存在线程死锁）。后来还是设置为16G了。当然我这真真的是无脑设置。我也看到了评论了胡老师的建议，很值得参考。
另外，胡老师在这节课里，讲到了页缓存，我想请问一下这个页缓存它存在的意义和作用，以及它在整个过程中的机制又是怎样的呢？

2019-06-26

作者回复

页缓存属于磁盘缓存（Disk cache）的一种，主要是为了改善系统性能。重复访问磁盘上的磁盘块是常见的操作，把它们保存在内存中可以避免昂贵的磁盘IO操作。

既然叫页缓存，它是根据页（page）来组织的内存结构。每一页包含了很多磁盘上的块数据。Linux使用Radix树实现页缓存，主要是加速特定页的查找速度。另外一般使用LRU策略来淘汰过期页数据。总之它是一个完全由内核来管理的磁盘缓存，用户应用程序通常是无感知的。

如果要详细了解page cache，可以参见《Understanding the Linux Kernel》一书的第15章

2019-06-27

xishuai 0

老师，之前有消息jdk闭源，有可能要使用openjdk，这个版本问题有相关配置吗？

2019-06-26

作者回复

目前没有Kafka + OpenJDK的案例分享。不过大致应该是类似的吧。

2019-06-26

wykkx 0

老师我的kafka的配置文件server.properties 里没有 message.max.bytes这个参数，是不是我要手工的加上去

2019-06-23

作者回复

这些参数都是有默认值的，如果没加就是官网中的默认值。

2019-06-24

Geek_986289 0

老师请问，
retention.ms
retention.bytes
这两个参数是不是只要满足一个，Kafka就会开始清消息了？还是需要两个同时满足才会清消息？

2019-06-23

作者回复

满足任何一个就会开始删除消息

2019-06-24

赌神很低调 0

“如果在页缓存中的数据在写入到磁盘前机器宕机了，那岂不是数据就丢失了。的确，这种情况数据确实就丢失了，但鉴于 Kafka 在软件层面已经提供了多副本的冗余机制，因此这里稍微拉大提交间隔去换取性能还是一个合理的做法。“即使提供了副本，这种情况数据也会丢吧？还是说这部分数据会重发？

2019-06-22

作者回复

我的意思是至少还有其他正常的副本可以使用。。。这个副本重启回来后会重新加载日志段，获取到当前末端位移，因此也能感知刚才为成功写入的消息并重新拉取之~~

2019-06-24