10 | 生产者压缩算法面面观

2019-06-25 胡夕

你好，我是胡夕。今天我要和你分享的内容是：生产者压缩算法面面观。

说起压缩（compression），我相信你一定不会感到陌生。它秉承了用时间去换空间的经典trade-off思想，具体来说就是用CPU时间去换磁盘空间或网络I/O传输量，希望以较小的CPU开销带来更少的磁盘占用或更少的网络I/O传输。在Kafka中，压缩也是用来做这件事的。今天我就来跟你分享一下Kafka中压缩的那些事儿。

怎么压缩？

Kafka是如何压缩消息的呢？要弄清楚这个问题，就要从Kafka的消息格式说起了。目前Kafka共有两大类消息格式，社区分别称之为V1版本和V2版本。V2版本是Kafka 0.11.0.0中正式引入的。

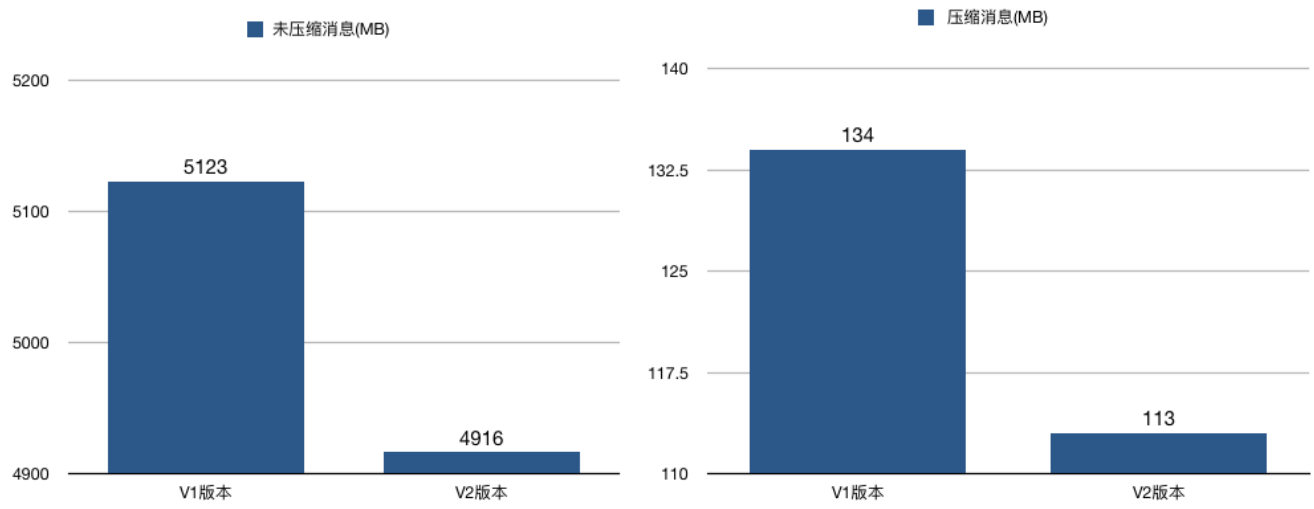
不论是哪个版本，Kafka的消息层次都分为两层：消息集合（message set）以及消息（message）。一个消息集合中包含若干条日志项（record item），而日志项才是真正封装消息的地方。Kafka底层的消息日志由一系列消息集合日志项组成。Kafka通常不会直接操作具体的一条条消息，它总是在消息集合这个层面上进行写入操作。

那么社区引入V2版本的目的是什么呢？V2版本主要是针对V1版本的一些弊端做了修正，和我们今天讨论的主题相关的修正有哪些呢？先介绍一个，就是把消息的公共部分抽取出来放到外层消息集合里面，这样就不用每条消息都保存这些信息了。

我来举个例子。原来在V1版本中，每条消息都需要执行CRC校验，但有些情况下消息的CRC值是会发生变化的。比如在Broker端可能会对消息时间戳字段进行更新，那么重新计算之后的CRC值也会相应更新；再比如Broker端在执行消息格式转换时（主要是为了兼容老版本客户端程序），也会带来CRC值的变化。鉴于这些情况，再对每条消息都执行CRC校验就有点没必要了，不仅浪费空间还耽误CPU时间，因此在V2版本中，消息的CRC校验工作就被移到了消息集合这一层。

V2版本还有一个和压缩息息相关的改进，就是保存压缩消息的方法发生了变化。之前V1版本中保存压缩消息的方法是把多条消息进行压缩然后保存到外层消息的消息体字段中；而V2版本的做法是对整个消息集合进行压缩。显然后者应该比前者有更好的压缩效果。

我对两个版本分别做了一个简单的测试，结果显示，在相同条件下，不论是否启用压缩，V2版本都比V1版本节省磁盘空间。当启用压缩时，这种节省空间的效果更加明显，就像下面这两张图展示的那样：



何时压缩？

在Kafka中，压缩可能发生在两个地方：生产者端和Broker端。

生产者程序中配置compression.type参数即表示启用指定类型的压缩算法。比如下面这段程序代码展示了如何构建一个开启GZIP的Producer对象：

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("acks", "all");
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
// 开启GZIP压缩
props.put("compression.type", "gzip");

Producer<String, String> producer = new KafkaProducer<>(props);
```

这里比较关键的代码行是props.put(“compression.type”, “gzip”)，它表明该Producer的压缩算法使用的是GZIP。这样Producer启动后生产的每个消息集合都是经GZIP压缩过的，故而能很好地节省网络传输带宽以及Kafka Broker端的磁盘占用。

在生产者端启用压缩是很自然的想法，那为什么我说在Broker端也可能进行压缩呢？其实大部分情况下Broker从Producer端接收到消息后仅仅是原封不动地保存而不会对其进行任何修改，但这里的“大部分情况”也是要满足一定条件的。有两种例外情况就可能让Broker重新压缩消息。

情况一：Broker端指定了和Producer端不同的压缩算法。

先看一个例子。想象这样一个对话。

Producer说：“我要使用GZIP进行压缩。”

Broker说：“不好意思，我这边接收的消息必须使用Snappy算法进行压缩。”

你看，这种情况下Broker接收到GZIP压缩消息后，只能解压缩然后使用Snappy重新压缩一遍。如果你翻开Kafka官网，你会发现Broker端也有一个参数叫compression.type，和上面那个例子中的同名。但是这个参数的默认值是producer，这表示Broker端会“尊重”Producer端使用的压缩算法。可一旦你在Broker端设置了不同的compression.type值，就一定要小心了，因为可能会发生预料之外的压缩/解压缩操作，通常表现为Broker端CPU使用率飙升。

情况二：Broker端发生了消息格式转换。

所谓的消息格式转换主要是为了兼容老版本的消费者程序。还记得之前说过的V1、V2版本吧？在一个生产环境中，Kafka集群中同时保存多种版本的消息格式非常常见。为了兼容老版本的格式，Broker端会对新版本消息执行向老版本格式的转换。这个过程中会涉及消息的解压缩和重新压缩。一般情况下这种消息格式转换对性能是有很大影响的，除了这里的压缩之外，它还让Kafka丧失了引以为豪的Zero Copy特性。

所谓“Zero Copy”就是“零拷贝”，我在专栏第6期提到过，说的是当数据在磁盘和网络进行传输时避免昂贵的内核态数据拷贝，从而实现快速的数据传输。因此如果Kafka享受不到这个特性的话，性能必然有所损失，所以尽量保证消息格式的统一吧，这样不仅可以避免不必要的解压缩/重新压缩，对提升其他方面的性能也大有裨益。如果有兴趣你可以深入地了解下Zero Copy的原理。

何时解压缩？

有压缩必有解压缩！通常来说解压缩发生在消费者程序中，也就是说Producer发送压缩消息到Broker后，Broker照单全收并原样保存起来。当Consumer程序请求这部分消息时，Broker依然原样发送出去，当消息到达Consumer端后，由Consumer自行解压缩还原成之前的消息。

那么现在问题来了，Consumer怎么知道这些消息是用何种压缩算法压缩的呢？其实答案就在消息中。Kafka会将启用了哪种压缩算法封装进消息集合中，这样当Consumer读取到消息集合时，它自然就知道了这些消息使用的是哪种压缩算法。如果用一句话总结一下压缩和解压缩，那么我希望你记住这句话：**Producer端压缩、Broker端保持、Consumer端解压缩。**

除了在Consumer端解压缩，Broker端也会进行解压缩。注意了，这和前面提到消息格式转换时发生的解压缩是不同的场景。每个压缩过的消息集合在Broker端写入时都要发生解压缩操作，目的就是为了对消息执行各种验证。我们必须承认这种解压缩对Broker端性能是有一定影响的，特别是对CPU的使用率而言。

事实上，最近国内京东的小伙伴们刚刚向社区提出了一个bugfix，建议去掉因为做消息校验而引入的解压缩。据他们称，去掉了解压缩之后，Broker端的CPU使用率至少降低了50%。不过有些遗憾的是，目前社区并未采纳这个建议，原因就是这种消息校验是非常重要的，不可盲目去之。毕竟先把事情做对是最重要的，在做对的基础上，再考虑把事情做好做快。针对这个使用场景，你也可以思考一下，是否有一个两全其美的方案，既能避免消息解压缩也能对消息执行校验。

各种压缩算法对比

那么我们来谈谈压缩算法。这可是重头戏！之前说了这么多，我们还是要比较一下各个压缩算法的优劣，这样我们才能有针对性地配置适合我们业务的压缩策略。

在Kafka 2.1.0版本之前，Kafka支持3种压缩算法：GZIP、Snappy和LZ4。从2.1.0开始，Kafka正式支持Zstandard算法（简写为zstd）。它是Facebook开源的一个压缩算法，能够提供超高的压缩比（compression ratio）。

对了，看一个压缩算法的优劣，有两个重要的指标：一个指标是压缩比，原先占100份空间的东西经压缩之后变成了占20份空间，那么压缩比就是5，显然压缩比越高越好；另一个指标就是压缩/解压缩吞吐量，比如每秒能压缩或解压缩多少MB的数据。同样地，吞吐量也是越高越好。

下面这张表是Facebook Zstandard官网提供的一份压缩算法benchmark比较结果：

Compressor name	Ratio	Compression	Decompress.
zstd 1.3.4 -1	2.877	470 MB/s	1380 MB/s
zlib 1.2.11 -1	2.743	110 MB/s	400 MB/s
brotli 1.0.2 -0	2.701	410 MB/s	430 MB/s
quicklz 1.5.0 -1	2.238	550 MB/s	710 MB/s
lzo1x 2.09 -1	2.108	650 MB/s	830 MB/s
lz4 1.8.1	2.101	750 MB/s	3700 MB/s
snappy 1.1.4	2.091	530 MB/s	1800 MB/s
lzf 3.6 -1	2.077	400 MB/s	860 MB/s

从表中我们可以发现zstd算法有着最高的压缩比，而在吞吐量上的表现只能说中规中矩。反观LZ4算法，它在吞吐量方面则是毫无疑问的执牛耳者。当然对于表格中数据的权威性我不做过多解读，只想用它来说明一下当前各种压缩算法的大致表现。

在实际使用中，GZIP、Snappy、LZ4甚至是zstd的表现各有千秋。但对于Kafka而言，它们的性能测试结果却出奇得一致，即在吞吐量方面：LZ4 > Snappy > zstd和GZIP；而在压缩比方面，zstd > LZ4 > GZIP > Snappy。具体到物理资源，使用Snappy算法占用的网络带宽最多，zstd最少，这是合理的，毕竟zstd就是要提供超高的压缩比；在CPU使用率方面，各个算法表现得差不多，只是在压缩时Snappy算法使用的CPU较多一些，而在解压缩时GZIP算法则可能使用更多的CPU。

最佳实践

了解了这些算法对比，我们就能根据自身的实际情况有针对性地启用合适的压缩算法。

首先来说压缩。何时启用压缩是比较合适的时机呢？

你现在已经知道Producer端完成的压缩，那么启用压缩的一个条件就是Producer程序运行机器上的CPU资源要很充足。如果Producer运行机器本身CPU已经消耗殆尽了，那么启用消息压缩无疑是雪上加霜，只会适得其反。

除了CPU资源充足这一条件，如果你的环境中带宽资源有限，那么我也建议你开启压缩。事实上我见过的很多Kafka生产环境都遭遇过带宽被打满的情况。这年头，带宽可是比CPU和内存还要珍贵的稀缺资源，毕竟万兆网络还不是普通公司的标配，因此千兆网络中Kafka集群带宽资源耗尽这件事情就特别容易出现。如果你的客户端机器CPU资源有很多富余，我强烈建议你开启zstd压缩，这样能极大地节省网络资源消耗。

其次说说解压缩。其实也没什么可说的。一旦启用压缩，解压缩是不可避免的事情。这里只想强调一点：我们对不可抗拒的解压缩无能为力，但至少能规避掉那些意料之外的解压缩。就像我前面说的，因为要兼容老版本而引入的解压缩操作就属于这类。有条件的话尽量保证不要出现消息格式转换的情况。

小结

总结一下今天分享的内容：我们主要讨论了Kafka压缩的各个方面，包括Kafka是如何对消息进行压缩的、何时进行压缩及解压缩，还对比了目前Kafka支持的几个压缩算法，最后我给出了工程化的最佳实践。分享这么多内容，我就只有一个目的：就是希望你能根据自身的实际情况恰当地选择合适的Kafka压缩算法，以求实现最大的资源利用率。

开放讨论

最后给出一道作业题，请花时间思考一下：前面我们提到了Broker要对压缩消息集合执行解压缩操作，然后逐条对消息进行校验，有人提出了一个方案：把这种消息校验移到Producer端来做，Broker直接读取校验结果即可，这样就可以避免在Broker端执行解压缩操作。你认同这种方案吗？

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。

huxi_2b

3

刚刚看到4天前京东提的那个jira已经修复了，看来规避了broker端为执行校验而做的解压缩操作，代码也merge进了2.4版本。有兴趣的同学可以看一下：https://issues.apache.org/jira/browse/KAFKA-8106

2019-06-26

趙衍

21

不行吧，校验操作应该也是为了防止在网络传输的过程中出现数据丢失的情况，在Producer端做完校验之后如果在传输的时候出现了错误，那这个校验就没有意义了。
我有一个问题想请教老师，如果每次传到Broker的消息都要做一次校验，那是不是都要把消息从内核态拷贝到用户态做校验？如果是这样的话那零拷贝机制不是就没有用武之地了？

2019-06-25

dream

5

老师，对于消息层次、消息集合、消息这三者的概念与关系我有点懵，能不能详细说一下？谢谢！

2019-06-25

常超

3

文中对于消息结构的描述，确实引起了一些混乱，下面试图整理一下，希望对大家有帮助。
消息（v1叫message，v2叫record）是分批次（batch）读写的，batch是kafka读写（网络传输和文件读写）的基本单位，不同版本，对相同（或者叫相似）的概念，叫法不一样。
v1（kafka 0.11.0之前）:message set, message
v2（kafka 0.11.0以后）:record batch,record
其中record batch对英语message set，record对应于message。
一个record batch（message set）可以包含多个record（message）。

对于每个版本的消息结构的细节，可以参考kafka官方文档的5.3 Message Format 章，里面对消息结构列得非常清楚。

2019-06-28

Hello world

3

做一个笔记
怎么压缩：
1、新版本改进将每个消息公共部分取出放在外层消息集合，例如消息的 CRC 值
2、新老版本的保存压缩消息的方法变化，新版本是对整个消息集合进行压缩
何时压缩：
1、正常情况下都是producer压缩，节省带宽，磁盘存储
2、例外情况 a、broker端和producer端使用的压缩方法不同 b、broker与client交互，消息版本不同
何时解压缩：
1、consumer端解压缩
2、broker端解压缩，用来对消息执行验证

优化：选择适合自己的压缩算法，是更看重吞吐量还是压缩率。其次尽量server和client保持一致，这样不会损失kafka的zero copy优势

2019-06-25

秋 3

我看了三遍老师的课，得到了我要的答案：
1.如果生产者使用了压缩，broker为了crc校验，会启动解压，这个解压过程不可避免；
2.v2的broker为了低版本的消费者，会把消息再次解压并进行协议转换。
所以消费者的兼容成本较大，需要避免这个情况。

2019-06-25

pain 2

怎么样才能保持消息格式统一呢，只要集群中的 kafka 版本一致吗？

2019-06-27

作者回复

嗯嗯，版本一致肯定是能保证的，不过通常比较难做到。

2019-06-28

南辕北辙 1

老师有一点有点迷惑，broker为了多版本消息兼容，意思是一份消息有多个版本存在吗，是这个意思吗？

2019-07-01

作者回复

同一台broker上可能存在多个版本的消息，但每条消息只会以1个版本的形式保存。

2019-07-01

南辕北辙 1

老师有一点不是很明白，在正常情况下broker端会原样保存起来，但是为了检验需要解压缩。该怎么去理解这个过程呢，broker端解压缩以后还会压缩还原吗？
这个过程是在用户态执行的吗，总感觉怪怪的

2019-06-27

作者回复

它只是解压缩读取而已，不会将解压缩之后的数据回写到磁盘。另外就像我置顶的留言那样，目前社区已经接纳了京东小伙伴的修改，貌似可以绕过这部分解压缩了，。

2019-06-28

giantbroom 1

我觉得有2个方案可以考虑：
1. 在Producer和Broker建立连接是，生成一个token，Producer每次发送消息是都带着token，Broker只需验证token的有效性，而不必在解压缩；
2. Producer在压缩之后，根据压缩后的数据生成jwt token，Broker同样只需验证jwt即可。

2019-06-26

cricket1981 1

校验的目的是防止因为网络传输出现问题导致broker端接收了受损的消息，所以应该放在作为serverr broker端进行，而不是在作为client端的producer。改进的方案可以是针对一次传输的整个message set进行CRC检验，而不是针对一条条消息，这能够大大提高校验效率，因为避免了解压缩。

2019-06-25

电光火石 0

老师好，问个问题，

1. 如果producer设置了compress.type，而broker没有设置，是否broker在接收到压缩的数据后，解压然后存储未压缩的数据到disk上面？

2. 如果producer没有设置compress.type，而broker设置了compress.type，是否broker在接收到原数据后，进行压缩然后存储？

谢谢了！

2019-06-28

作者回复

1. 会解压，然后保存在disk上

2. 会压缩

2019-07-01

Jowin 0

是不是每个RecordBatch只能包含同一个主题的消息？否则就需要解压，保存到不同主题的日志文件中？

2019-06-28

作者回复

是的。不过这和解压不解压没有关系。RecordBatch中设置了压缩类型

2019-07-01

Jowin 0

@胡老师，请假一个问题，记得Kafka协议升级到2.0之后，一个MessageSet中会包含多条消息，那么在broker端保存日志文件并更新索引的时候，不需要把MessageSet拆开，查看每个消息在分区中的的偏移，从而更新索引文件么？

2019-06-28

作者回复

索引机制没有变化，本来也是稀疏索引，保存的是消息位移到物理文件位置的映射。不论哪个消息格式，消息位移数据都是准确且不变的。

2019-07-01

风中花 0

胡老师您好！ 我们已经学历10多节课了！ 针对我们得留言和反馈，不知道您有没有给我们一些后续得课程得学习建议和方法？我目前得学习就是您告诉我们得，我必须学会记住。但是看同学们得评论和反馈，我觉得貌似还有很多很多知识啊且不知也不懂，故有此一问！希望老师能给与一点一点学习建议？ 感谢老师

2019-06-26

作者回复

个人觉得学一个东西最重要的还是要用，如果只是参加一些培训课程很难全面的理解。您这么多的留言我一直坚持回复。我也一直是这个观点：用起来，自然问题就来了。

我学机器学习的经历和您现在学Kafka很像。没有实际使用场景怎么学都觉得深入不了。

我给您的建议是：把Kafka官网通读几遍然后再实现一个实时日志收集系统（比如把服务器日志实时放入Kafka）。事实上，能把官网全面理解的话已经比很多Kafka使用者要强了。

2019-06-27

爱学习的猪 0

生产者端根据原生消息压缩后的压缩内容，给出检验规则，服务端直接根据检验规则对压缩后的内容做检验？

2019-06-26

作者回复

依然可能要解压缩

2019-06-26

Geek_8441fd 0

broker端校验可以分两步走。
第1步，message set 层面，增加一个 crc，这样可以不用解压缩，直接校验压缩后的数据。
如果校验不成功，说明message set 中有损坏的message；
这时，再做解压操作，挨个校验message，找出损坏的那一个。

这样的话，绝大部分情况下，是不用做解压操作的；只有在确实发生错误时，才需要解压。
请指正。

2019-06-25

作者回复

嗯嗯，挺好的。我自己也学到了一些。另外校验不仅仅是CRC校验，还有消息级别的检查。

2019-06-26

WL 0

请问一下老师怎样在消费者端配置让消费者不用兼容老版本V1版本的消息?

2019-06-25

作者回复

消费者都是自动向后兼容的，这也是好事吧。。。。

2019-06-26

刘朋 0

问题: 压缩/解压缩格式类型保存位置表述不清晰
文内在解压缩时说,Kafka会将启用了哪些压缩算法封装进消息集合中,这样当Consumer读取到消息集合时,

它自然就知道了这些消息使用的是哪种压缩算法.但在文内压缩时说, V2版本是对整个消息集合进行压缩.

从字面意思来理解,Producer在对消息集合压缩前,已经将压缩格式封装到了消息集合中.Consumer是怎么获取的压缩格式类型?要获取压缩格式得先进行解压,然后才能获取压缩类型. 这前后表述有矛盾了,有待解惑.

2019-06-25

作者回复

消息batch的元数据不用解压缩就能获取

2019-06-26

dream

0

老师，我对消息层次、消息集合、消息、日志项这些概念与它们之间的关系感觉很懵，
消息层次都分消息集合以及消息，消息集合中包含日志项，日志项中封装消息，
那么日志项中封装的是producer发送的消息吗？
一个日志项中会包含多条消息吗？
消息集合中消息项封装的的消息与消息层次包含的消息有什么关系呢？
这两个消息与producer发送的消息有什么关系呢？
一个消息集合对应是producer发送的一条消息还是多条消息呢？
最后，老师能不能详细说一下CRC校验，谢谢！

2019-06-25

作者回复

消息批次RecordBatch里面包含若干条消息（record)。

你可以认为消息批次和消息集合是等价的，消息和日志项是等价的。

这样消息层次有两层：外层是消息批次（或消息集合）；里层是消息（或日志项）。

Producer以recordbatch为单位发送消息，对于V2版本一个batch中通常包含多条消息。

在V2版本中，在batch层面计算CRC值；在V1版本中，每条消息都要计算CRC值。

2019-06-26