kafka_partitions_problem

vintagewang edited this page 9 days ago · 4 revisions

Kafka模型产生自日志记录场景，受到场景所限，Kafka不需要太高的并发度。而在阿里这样的大规模应用中，我们经过实践发现，原有模型已经不能满足阿里的实际需要。ONS(RocketMQ)则比较好的解决了并发数问题，已经是内部非常广泛使用的一套产品。

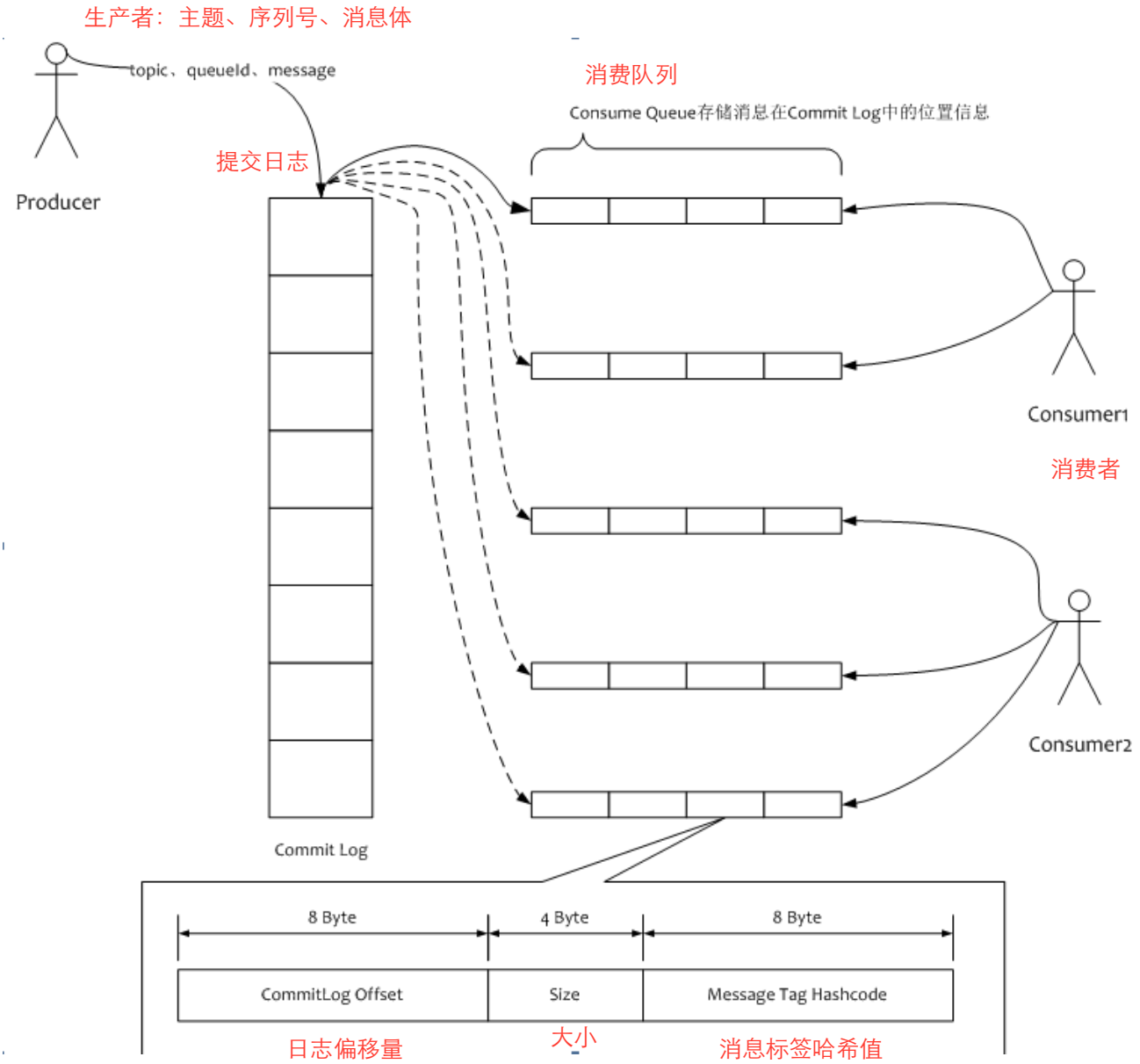
分区在Kafka中有什么作用?

1. Producer在Broker并发写入与分区数成正比。
2. Consumer消费某个Topic的并行度与分区数保持一致，假设分区数是20，那么Consumer的消费并行度最大为20。
3. 每个Topic由固定数量的分区数组成，分区数的多少决定了单台Broker能支持的Topic数量，Topic数量又决定了支持的业务数量。

为什么Kafka不能支持更多的分区数?

1. 每个分区存储了完整的消息数据，虽然每个分区写入是磁盘顺序写，但是多个分区同时顺序写入在操作系统层面变为了随机写入。
2. 由于数据分散为多个文件，很难利用IO层面的Group Commit机制，网络传输也会用到类似优化算法。

Alibaba RocketMQ（Also Aliyun ONS）如何支持更多分区数?



1. 所有数据单独存储到一个Commit Log，完全顺序写，随机读。
2. 对最终用户展现的队列实际只存储消息在Commit Log的位置信息，并且串行方式刷盘。

> 这样做的好处如下：

1. 队列轻量化，单个队列数据量非常少。
2. 对磁盘的访问串行化，避免磁盘竞争，不会因为队列增加导致IOWAIT增高。

> 每个方案都有缺点，它的缺点如下：

1. 写虽然完全是顺序写，但是读却变成了完全的随机读。 走向两个极端
2. 读一条消息，会先读Consume Queue，再读Commit Log，增加了开销。
3. 要保证Commit Log与Consume Queue完全的一致，增加了编程的复杂度。

> 以上缺点如何克服：

1. 随机读，尽可能让读命中PAGECACHE，减少IO读操作，所以内存越大越好。如果系统中堆积的消息过多，读数据要访问磁盘会不会由于随机读导致系统性能急剧下降，答案是否定的。
    - 访问PAGECACHE时，即使只访问1k的消息，系统也会提前预读出更多数据，在下次读时，就可能命中内存。
    - 随机访问Commit Log磁盘数据，系统IO调度算法设置为NOOP方式，会在一定程度上将完全的随机读变成顺序跳跃方式，而顺序跳跃方式读较完全的随机读性能会高5倍以上，可参见以下针对各种IO方式的性能数据。 http://stblog.baidu-tech.com/?p=851 另外4k的消息在完全随机访问情况下，仍然可以达到8K次每秒以上的读性能。

2. 由于Consume Queue存储数据量极少，而且是顺序读，在PAGECACHE预读作用下，Consume Queue的读性能几乎与内存一致，即使堆积情况下。所以可认为Consume Queue完全不会阻碍读性能。

3. Commit Log中存储了所有的元信息，包含消息体，类似于Mysql、Oracle的redolog，所以只要有Commit Log在，Consume Queue即使数据丢失，仍然可以恢复出来。

可靠性、容错性不错

联系本文作者

- 新浪微博
- vintage.wang@gmail.com
- 加入RocketMQ开源群, 群号：364685175