如何选择合适的分区数

如何选择合适的分区数？这是很多 Kafka 的使用者经常面临的问题，不过对这个问题而言，似乎并没有非常权威的答案。而且这个问题显然也没有固定的答案，只能从某些角度来做具体的分析，最终还是要根据实际的业务场景、软件条件、硬件条件、负载情况等来做具体的考量。本节主要介绍与本问题相关的一些重要决策因素，使读者在遇到类似问题时能够有参考依据。

性能测试工具

在 Kafka 中，性能与分区数有着必然的关系，在设定分区数时一般也需要考虑性能的因素。对不同的硬件而言，其对应的性能也会不太一样。在实际生产环境中，我们需要了解一套硬件所对应的性能指标之后才能分配其合适的应用和负荷，所以性能测试工具必不可少。

本节要讨论的性能测试工具是 Kafka 本身提供的用于生产者性能测试的 kafka-producer- perf-test.sh 和用于消费者性能测试的 kafka-consumer-perf-test.sh。

首先我们通过一个示例来了解一下 kafka-producer-perf-test.sh 脚本的使用。我们向一个只有1个分区和1个副本的主题topic-1中发送100万条消息，并且每条消息大小为1024B，生产者对应的 acks 参数为1。详细内容参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
producer-perf-test.sh --topic topic-1 --num-
records 1000000 --record-size 1024 --throughput
-1 --producer-props bootstrap.
servers=localhost:9092 acks=1
273616 records sent, 54723.2 records/sec (53.44
MB/sec), 468.6 ms avg latency, 544.0 max latency.
337410 records sent, 67482.0 records/sec (65.90
MB/sec), 454.4 ms avg latency, 521.0 max latency.
341910 records sent, 68382.0 records/sec (66.78
MB/sec), 449.4 ms avg latency, 478.0 max latency.
1000000 records sent, 63690.210815 records/sec
(62.20 MB/sec), 456.17 ms avg latency, 544.00 ms
max latency, 458 ms 50th, 517 ms 95th, 525 ms
99th, 543 ms 99.9th.
```

示例中在使用 kafka-producer-perf-test.sh 脚本时用了多一个参数，其中 topic 用来指定生产者发送消息的目标主题；num-records 用来指定发送消息的总条数；record-size 用来设置每条消息的字节数；producer-props 参数用来指定生产者的配置，可同时指定多组配置，各组配置之间以空格分隔，与 producer-props 参数对应的还有一个 producer.config 参数，它用来指定生产者的配置文件；throughput 用来进行限流控制，当设定的值小于0时不限流，当设定的值大于0时，当发送的吞吐量大于该值时就会被阻塞一段时间。下面的示例中设置了 throughout 的值为100字节，我们来看一下实际的效果：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
producer-perf-test.sh --topic topic-1 --num-
records 1000000 --record-size 1024 --throughput
100 --producer-props bootstrap.
servers=localhost:9092 acks=1
502 records sent, 100.3 records/sec (0.10
MB/sec), 2.5 ms avg latency, 266.0 max latency.
501 records sent, 100.0 records/sec (0.10
MB/sec), 0.9 ms avg latency, 11.0 max latency.
500 records sent, 99.9 records/sec (0.10 MB/sec),
0.8 ms avg latency, 3.0 max latency.
501 records sent, 100.2 records/sec (0.10
MB/sec), 0.7 ms avg latency, 3.0 max latency.
500 records sent, 100.0 records/sec (0.10
MB/sec), 0.7 ms avg latency, 5.0 max latency.
(…省略若干)
```

kafka-producer-perf-test.sh 脚本中还有一个有意思的参数 print-metrics，指定了这个参数时会在测试完成之后打印很多指标信息，对很多测试任务而言具有一定的参考价值。示例参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
producer-perf-test.sh --topic topic-1 --num-
records 1000000 --record-size 1024 --throughput
-1 --print-metrics --producer-props
bootstrap.servers=localhost:9092 acks=1
272926 records sent, 54585.2 records/sec (53.31
MB/sec), 469.6 ms avg latency, 638.0 max latency.
331020 records sent, 66204.0 records/sec (64.65
MB/sec), 463.8 ms avg latency, 507.0 max latency.
345960 records sent, 69192.0 records/sec (67.57
MB/sec), 443.8 ms avg latency, 477.0 max latency.
1000000 records sent, 63552.589768 records/sec
```

```
(62.06 MB/sec), 457.73 ms avg latency, 638.00 ms
max latency, 457 ms 50th, 532 ms 95th, 592 ms
99th, 633 ms 99.9th.

Metric Name
Value
app-info:commit-id:{client-id=producer-1}
: 3402a8361b734732
app-info:version:{client-id=producer-1}
: 2.0.0
kafka-metrics-count:count:{client-id=producer-1}
: 94.000
producer-metrics:batch-size-avg:{client-
id=producer-1}  : 15555.923
producer-metrics:batch-size-max:{client-
id=producer-1}  : 15556.000
producer-metrics:batch-split-rate:{client-
id=producer-1}    : 0.000
producer-metrics:batch-split-total:{client-
id=producer-1}   : 0.000
producer-metrics:buffer-available-bytes:{client-
id=producer-1}        : 33554432.000
producer-metrics:buffer-exhausted-rate:{client-
id=producer-1}     : 0.000
producer-metrics:buffer-exhausted-total:{client-
id=producer-1}  : 0.000
producer-metrics:buffer-total-bytes:{client-
id=producer-1}    : 33554432.000
producer-metrics:bufferpool-wait-ratio:{client-
id=producer-1}     : 0.278
producer-metrics:bufferpool-wait-time-total:
{client-id=producer-1}    : 12481086207.000
(…省略若干)
```

kafka-producer-perf-test.sh 脚本中还有一些其他的参数，比如 payload-delimiter、transactional-id 等，读者可以自行探索一下此脚本的更多细节。

我们再来关注 kafka-producer-perf-test.sh 脚本的输出信息，以下面的一行内容为例：

```
1000000 records sent, 63690.210815 records/sec
(62.20 MB/sec), 456.17 ms avg latency, 544.00 ms
max latency, 458 ms 50th, 517 ms 95th, 525 ms
99th, 543 ms 99.9th.
```

records sent 表示测试时发送的消息总数；records/sec 表示以每秒发送的消息数来统计吞吐量，括号中的 MB/sec 表示以每秒发送的消息大小来统计吞吐量，注意这两者的维度；avg latency 表示消息处理的平均耗时；max latency 表示消息处理的最大耗时；50th、95th、99th 和 99.9th 分别表示 50%、95%、99% 和 99.9% 的消息处理耗时。
kafka-consumer-perf-test.sh 脚本的使用也比较简单，下面的示例简单地演示了其使用方式：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-perf-test.sh --topic topic-1 --messages
1000000 --broker-list localhost:9092
start.time, end.time, data.consumed.in.MB,
MB.sec, data.consumed.in.nMsg, nMsg.sec,
rebalance.time.ms, fetch.time.ms, fetch.MB.sec,
fetch.nMsg.sec
2018-09-22 12:27:49:827, 2018-09-22 12:27:57:068,
976.5625, 134.8657, 1000000, 138102.4720, 105,
7136, 136.8501, 140134.5291
```

示例中只是简单地消费主题 topic-1 中的100万条消息。脚本中还包含了许多其他的参数，比如 from-latest、group、print-metrics、threads 等，篇幅限制，读者可以自行了解这些参数的使用细节。

输出结果中包含了多项信息，分别对应起始运行时间（start.time）、结束运行时间（end.time）、消费的消息总量（data.consumed.in.MB，单位为MB）、按字节大小计算的消费吞吐量（MB.sec，单位为MB/s）、消费的消息总数（data.consumed.in.nMsg）、按消息个数计算的吞吐量（nMsg.sec）、再平衡的时间（rebalance.time.ms，单位为ms）、拉取消息的持续时间（fetch.time.ms，单位为ms）、每秒拉取消息的字节大小（fetch.MB.sec，单位为MB/s）、每秒拉取消息的个数（fetch.nMsg.sec）。其中 fetch.time.ms = end.time - start.time - rebalance.time.ms。

这里只是简单地了解两个脚本的基本用法，读者还可以通过设置不同的参数来调节测试场景以获得针对当前硬件资源的一份相对比较完善的测试报告。

分区数越多吞吐量就越高吗

分区是 Kafka 中最小的并行操作单元，对生产者而言，每一个分区的数据写入是完全可以并行化的；对消费者而言，Kafka 只允许单个分区中的消息被一个消费者线程消费，一个消费组的消费并行度完全依赖于所消费的分区数。如此看来，如果一个主题中的分区数越多，理论上所能达到的吞吐量就越大，那么事实真的如预想的一样吗?

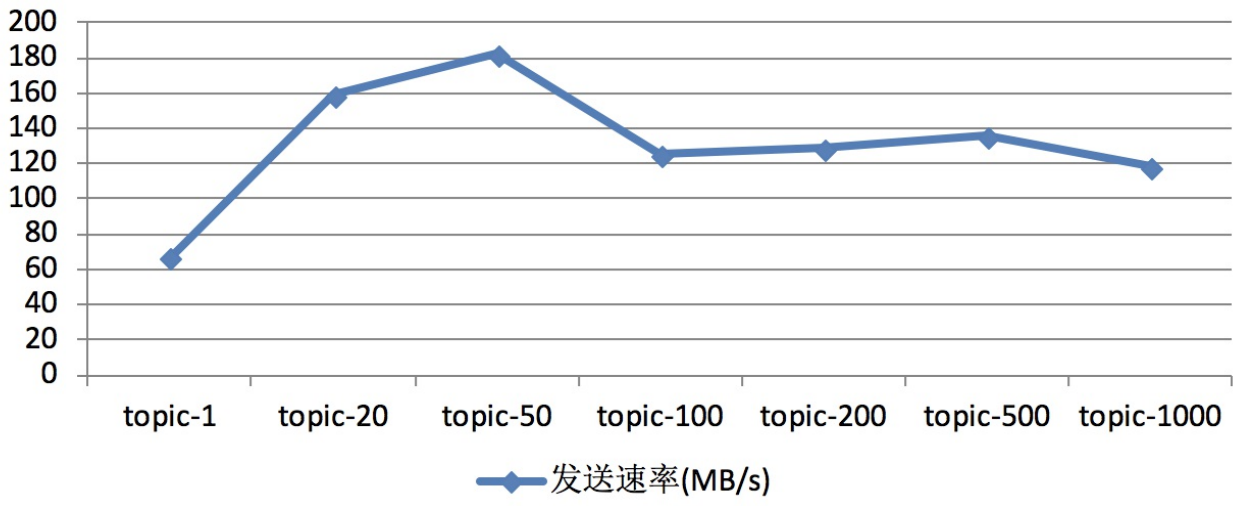
我们使用上面介绍的性能测试工具来实际测试一下。首先分别创建分区数为1、20、50、100、200、500、1000的主题，对应的主题名称分别为topic-1、topic-20、topic-50、topic-100、topic-200、topic-500、topic-1000，所有主题的副本因子都设置为1。

消息中间件的性能一般是指吞吐量（广义来说还包括延迟）。抛开硬件资源的影响，消息写入的吞吐量还会受到消息大小、消息压缩方式、消息发送方式（同步/异步）、消息确认类型（acks）、副本因子等参数的影响，消息消费的吞吐量还会受到应用逻辑处理速度的影响。本案例中暂不考虑这些因素的影响，所有的测试除了主题的分区数不同，其余的因素都保持相同。

本次案例中使用的测试环境为一个由3台普通云主机组成的3节点的Kafka 集群，每台云主机的内存大小为8GB、磁盘大小为40GB、4核CPU的主频为2600MHz。JVM 版本为1.8.0_112，Linux系统版本为2.6.32-504.23.4.el6.x86_64。
使用 kafka-producer-perf-test.sh 脚本分别向这些主题中发送100万条消息体大小为1KB的消息，对应的测试命令如下：

```
bin/kafka-producer-perf-test.sh --topic topic-xxx
--num-records 1000000 --record-size 1024 --
throughput -1 --producer-props
bootstrap.servers=localhost: 9092 acks=1
```

对应的生产者性能测试结果如下图所示。不同的硬件环境，甚至不同批次的测试得到的测试结果也不会完全相同，但总体趋势还是会保持和下图中的一样。

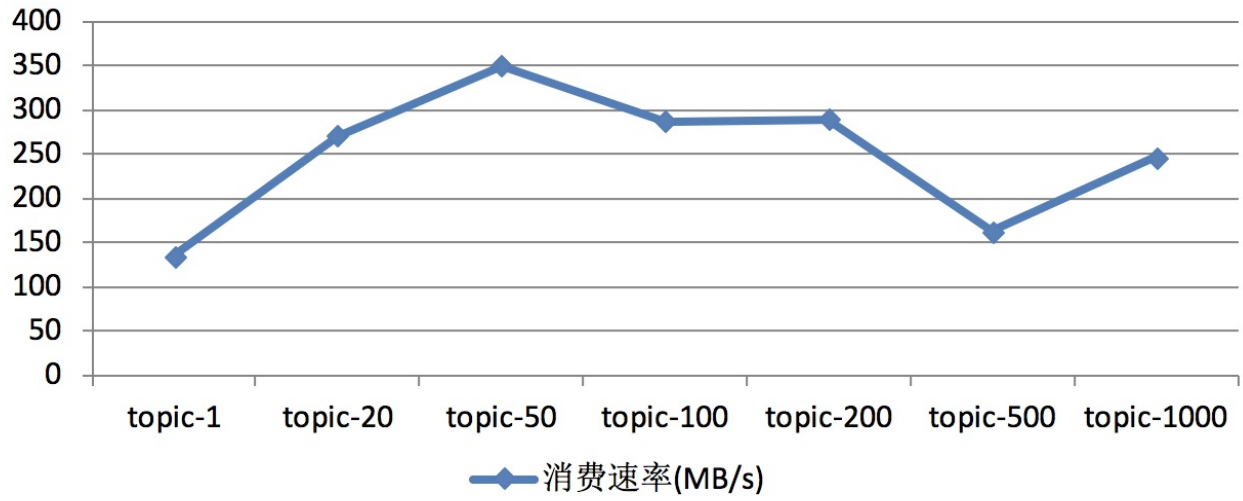


在上图中，我们可以看到分区数为1时吞吐量最低，随着分区数的增长，相应的吞吐量也跟着上涨。一旦分区数超过了某个阈值之后，整体的吞吐量是不升反降的。也就是说，并不是分区数越多吞吐量也越大。这里的分区数临界阈值针对不同的测试环境也会表现出不同的结果，实际应用中可以通过类似的测试案例（比如复制生产流量以便进行测试回放）来找到一个合理的临界值区间。

上面针对的是消息生产者的测试，对消息消费者而言同样有吞吐量方面的考量。使用 kafka-consumer-perf-test.sh 脚本分别消费这些主题中的100万条消息，对应的测试命令如下：

```
bin/kafka-consumer-perf-test.sh --topic topic-xxx
--messages 1000000 --broker-list localhost:9092
```

消费者性能测试的结果如下图所示。与生产者性能测试相同的是，不同的测试环境或不同的测试批次所得到的测试结果也不尽相同，但总体趋势还是会保持和下图中的一样。



在上图中，随着分区数的增加，相应的吞吐量也会有所增长。一旦分区数超过了某个阈值之后，整体的吞吐量也是不升反降的，同样说明了分区数越多并不会使吞吐量一直增长。

在同一套环境下，我们还可以测试一下同时往两个分区数为200的主题中发送消息的性能，假设测试结果中两个主题所对应的吞吐量分别为A和B，再测试一下只往一个分区数为200的主题中发送消息的性能，假设此次测试结果中得到的吞吐量为C，会发现A<C、B<C且A+B>C。可以发现由于共享系统资源的因素，A和B之间会彼此影响。通过A+B>C的结果，可知第一张图中 topic-200 的那个点位也并没有触及系统资源的瓶颈，发生吞吐量有所下降的结果也并非是系统资源瓶颈造成的。

本节针对分区数越多吞吐量越高这个命题进行反证，其实要证明一个观点是错误的，只需要举个反例即可，本节的内容亦是如此。不过本节并没有指明分区数越多吞吐量就越低这个观点，并且具体吞吐量的数值和走势还会和磁盘、文件系统、I/O调度策略相关。分区数越多吞吐量也就越高？网络上很多资料都认可这一观点，但实际上很多事情都会有一个临界值，当超过这个临界值之后，很多原本符合既定逻辑的走向又会变得不同。读者需要对此有清晰的认知，懂得去伪求真，实地测试验证不失为一座通向真知的桥梁。