kafka性能基准测试

转载请注明出处：http://www.cnblogs.com/xiaodf/

1、测试环境

该benchmark用到了六台机器，机器配置如下

l IntelXeon 2.5 GHz processor with six cores

l Six7200 RPM SATA drives

l 32GB ofRAM

l 1GbEthernet

这6台机器其中3台用来搭建Kafka broker集群，另外3台用来安装Zookeeper及生成测试数据。6个drive都直接以非RAID方式挂载。实际上kafka对机器的需求与Hadoop的类似。

2、producer吞吐率

该项测试只测producer的吞吐率，也就是数据只被持久化，没有consumer读数据。总数据条数50 million。

测试过程中可以改变不同参数的取值，来测试特定参数对性能的影响。

使用官方提供的测试工具kafka-producer-perf-test.sh来测试。

kafka-producer-perf-test.sh中参数说明：

参数	说明
messages	生产者发送总的消息数量
message-size	每条消息大小
batch-size	每次批量发送消息的数量
topics	生产者发送的topic
threads	生产者使用几个线程同时发送
broker-list	安装kafka服务的机器ip:port列表
producer-num-retries	一个消息失败发送重试次数
request-timeout-ms	一个消息请求发送超时时间

2.1 线程数

创建一个6个分区，没有备份的topic，设置不同的线程数生产相同量的数据，查看性能变化。运行结果如下表所示，(可测多组用折线图表示)。

操作	线程	partition	replication	Records/second	MB/second
1	*1*	*6*	*1*	99837.8633	9.5213
2	*3*	*6*	*1*	261730.7628	24.9606
3	*6*	*6*	*1*	306865.1757	29.2649

执行命令：

1、创建topic

bin/kafka-topics.sh--create --zookeeper localhost:2181/kafka --topic test-rep-one --partitions 6--replication-factor 1

2、生产数据

(1) 一个进程

bin/kafka-producer-perf-test.sh--messages 50000000 --topicstest-rep-one --threads 1 --broker-list m103:9092,m105:9092

结果：

start.time,end.time, compression, message.size, batch.size, total.data.sent.in.MB, MB.sec,total.data.sent.in.nMsg, nMsg.sec

2015-05-2611:44:12:728, 2015-05-26 11:52:33:540, 0, 100, 200, 4768.37, 9.5213, 50000000,99837.8633

(2) 3个进程

bin/kafka-producer-perf-test.sh--messages 50000000 --topicstest-rep-one --threads 3 --broker-list m103:9092,m105:9092

(2) 6个进程

bin/kafka-producer-perf-test.sh--messages 50000000 --topicstest-rep-one --threads 6 --broker-list m103:9092,m105:9092

2.2 分区数

新建topic，改变分区数，生产等量数据，查看性能变化。一般情况下，对于大数据量，partition的数量大于等于broker的数据。(可测多组用折线图表示)

操作	进程	partition	replication	Records/second	MB/second
1	*1*	*6*	*1*	99837.8633	9.5213
2	*1*	*12*	*1*	84516.7081	8.0601

执行命令：

1、创建topic

bin/kafka-topics.sh--create --zookeeper localhost:2181/kafka --topic test-rep-one-part --partitions 12 --replication-factor 1

2、生产数据

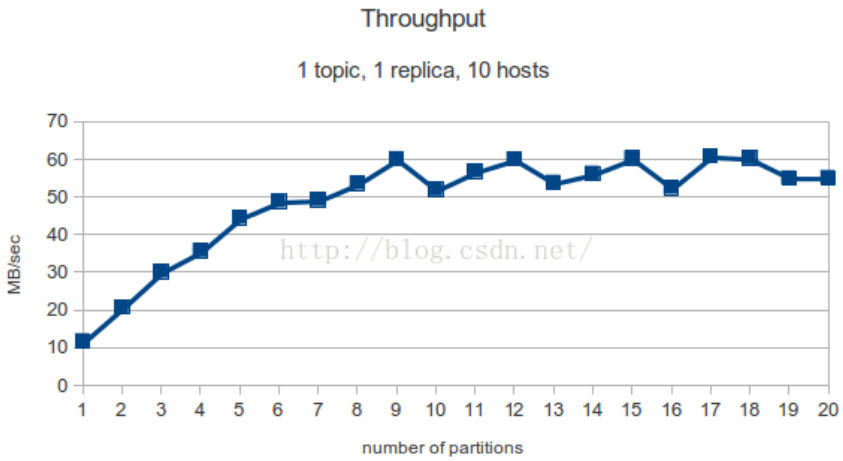
(1) 一个进程

bin/kafka-producer-perf-test.sh--messages 50000000 --topicstest-rep-one-part --threads 1 --broker-list m103:9092,m105:9092 --request-timeout-ms 10000

结果：

2015-06-0115:36:58:224, 2015-06-01 15:46:49:823, 0, 100, 200, 4768.37, 8.0601, 50000000,84516.7081

初步结论：分区数越多，单线程生产者吞吐率越小。



Throughputincreases very markedly at first as more brokers and disks on them starthosting different partitions. Once all brokers and disks are used though,adding additional partitions does not seem to have any effect.

2.3 备份数

新建topic，改变分区数，生产等量数据，查看性能变化。可以推断吞吐性能会随备份数增多而减少。(可测多组用折线图表示)

操作	线程	partition	replication	Records/second	MB/second
1	*1*	*6*	*0*	99837.8633	9.5213
2	*2*	*6*	*2*	86491.7227	8.2485

1、创建topic

bin/kafka-topics.sh--create --zookeeper localhost:2181/kafka --topic test-rep-two --partitions 6 --replication-factor 2

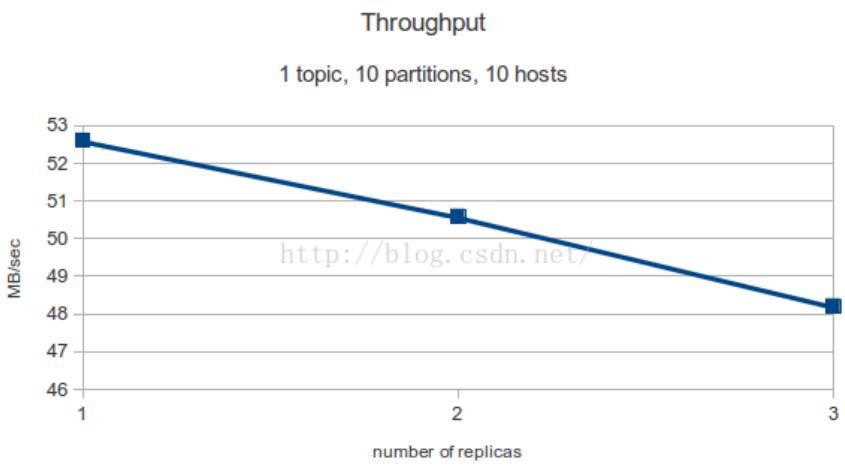
2、生成数据

bin/kafka-producer-perf-test.sh--messages50000000 --topics test-rep-two --threads 2 --broker-list m103:9092,m105:9092

结果：

2015-05-2615:26:39:899, 2015-05-26 15:36:17:989, 0, 100, 200, 4768.37, 8.2485, 50000000,86491.7227

初步结论：备份数越多，吞吐率越低。



2.4 broker数

增加broker的个数，查看性能变化。

操作	线程	Broker数	partition	replication	Records/second	MB/second
1	*2*	*1*	*6*	*2异步*	248172.2117	23.6675
2	*2*	*2*	*6*	*2异步*	251764.8718	24.0102

1、异步产生数据

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --batch-size 3000 --broker-list m103:9092,m105:9092 --request-timeout-ms 100000

结论：

增加broker的个数，吞吐量增加。

2.5 同步异步

分别以同步异步方式，生产等量数据，查看性能变化。

操作	线程	partition	replication	Records/second	MB/second
1	*2*	*6*	*2同步*	68860.1849	6.5670
2	*2*	*6*	*2异步*	172405.4701	16.4419

运行命令：

1、创建topic

bin/kafka-topics.sh--create --zookeeper localhost:2181/kafka --topic test-rep-two_2 --partitions 6 --replication-factor 2

2、同步产生数据

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --broker-listm103:9092,m105:9092 --sync

结果：

2015-06-0210:38:16:846, 2015-06-02 10:50:22:955, 0, 100, 200, 4768.37, 6.5670, 50000000,68860.1849

3、异步产生数据

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --batch-size 5000 --broker-list m103:9092,m105:9092 --request-timeout-ms 100000

结果：

2015-06-0210:36:12:492, 2015-06-02 10:41:02:506, 0, 100, 5000, 4768.37, 16.4419,50000000, 172405.4701

结论：异常产生数据比同步产生数据吞吐率高近3倍。

2.6 批处理大小

异步生产等量数据，改变批处理量大小，查看性能变化。

操作	线程	partition	replication	批大小	Records/second	MB/second
1	*2*	*6*	*2*	*200*	86491.7227	8.2485
2	*2*	*6*	*2*	*1000*	187594.7353	17.8904
3	*2*	*6*	*2*	*2000*	244479.6495	23.3154

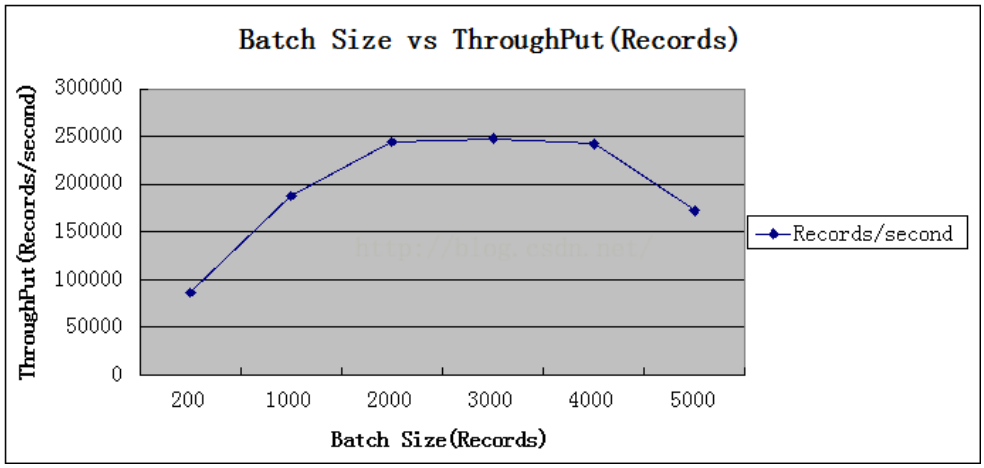
4	*2*	*6*	*2*	*3000*	248172.2117	23.6675
5	*2*	*6*	*2*	*4000*	242217.5501	23.0997
6	*2*	*6*	*2*	*5000*	172405.4701	16.4419

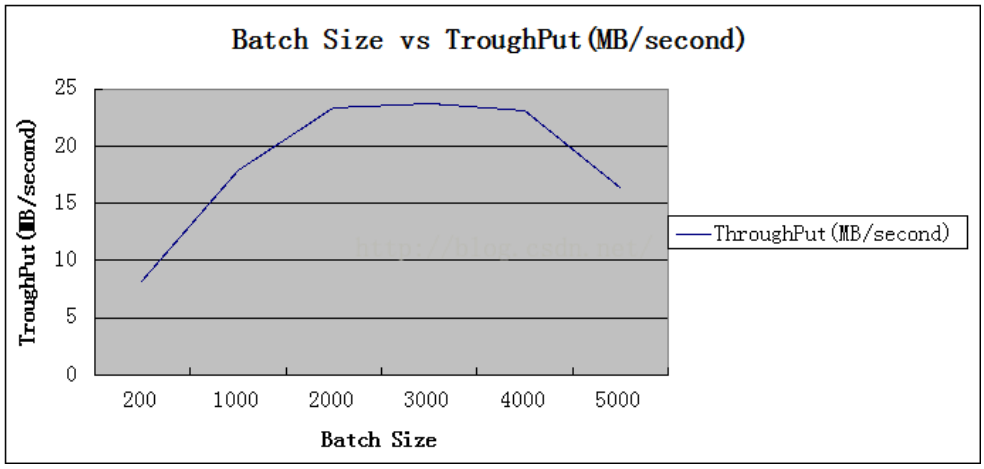
运行命令：

1、生成数据

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --batch-size 1000 --broker-list m103:9092,m105:9092--request-timeout-ms 100000

结果：





2.7 消息长度

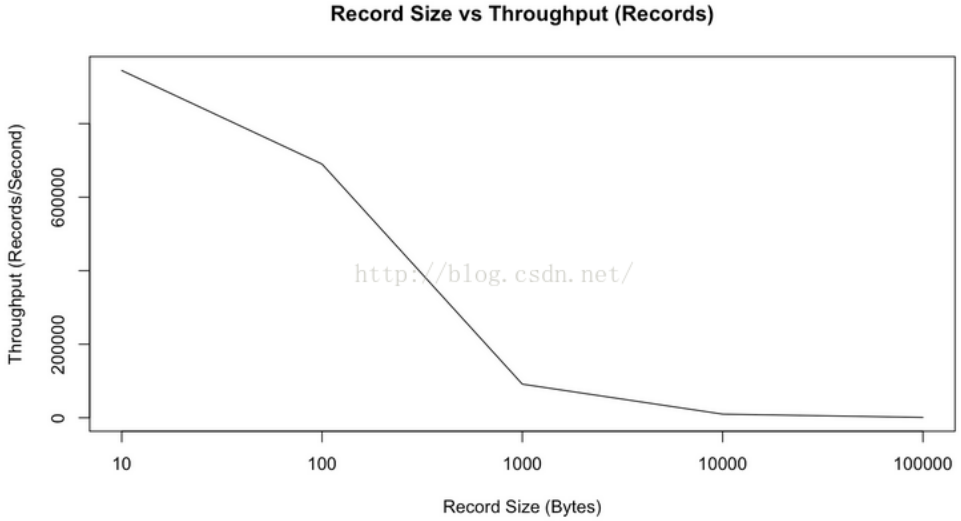
操作	线程	消息长度（bytes）	Records/second	MB/second
1	*2*	*100*	248172.2117	23.6675
2	*2*	*200*	132873.0610	25.3435
3	*2*	*500*	79277.1195	37.8023
4	*2*	*1000*	39015.5290	37.2081

异步产生数据：

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --batch-size 3000 --broker-list m103:9092,m105:9092 --request-timeout-ms 100000 --message-size200

结论：

上面的所有测试都基于短消息（payload100字节），而正如上文所说，短消息对Kafka来说是更难处理的使用方式，可以预期，随着消息长度的增大，records/second会减小，但MB/second会有所提高。下图是records/second与消息长度的关系图。



正如我们所预期的那样，随着消息长度的增加，每秒钟所能发送的消息的数量逐渐减小。但是如果看每秒钟发送的消息的总大小，它会随着消息长度的增加而增加，如下图所示。

从上图可以看出，当消息长度为10字节时，因为要频繁入队，花了太多时间获取锁，CPU成了瓶颈，并不能充分利用带宽。但从100字节开始，我们可以看到带宽的使用逐渐趋于饱和（虽然MB/second还是会随着消息长度的增加而增加，但增加的幅度也越来越小）。

2.8 数据压缩

分别以同步异步方式，生产等量数据，查看性能变化。

操作	进程	partition	压缩	Records/second	MB/second
1	*2*	*6*	*无(0)*	87677.3054	8.3616
2	*2*	*6*	*GZIP(1)*	84312.6245	8.0407
3	*2*	*6*	*Snappy(2)*	140113.7163	13.3623

运行命令：

1、生成数据

bin/kafka-producer-perf-test.sh--messages 50000000 --topics test-rep-two_2 --threads 2 --compression-codec 1 --batch-size3000 --broker-list m103:9092,m105:9092 --request-timeout-ms 100000

结果：

start.time,end.time, compression, message.size, batch.size, total.data.sent.in.MB, MB.sec,total.data.sent.in.nMsg, nMsg.sec

2015-06-0214:17:29:393, 2015-06-02 14:23:26:246, 2, 100, 3000, 4768.37, 13.3623,50000000, 140113.7163

3、consumer吞吐率

需要注意的是，replicationfactor并不会影响consumer的吞吐率测试，因为consumer只会从每个partition的leader读数据，而与replicaiton factor无关。同样，consumer吞吐率也与同步复制还是异步复制无关。

该测试从有6个partition，3个replication的topic消费50 million的消息。使用官方提供的测试工具kafka-consumer-perf-test.sh来测试。

kafka-consumer-perf-test.sh中参数说明：

参数	说明
zookeeper	zookeeper端口配置
messages	消费者消费消息总数量
topic	消费者需要消费的topic
threads	消费者使用几个线程同时消费
group	消费者组名称
socket-buffer-sizesocket	缓冲大小
fetch-size	每次向kafka broker请求消费大小
consumer.timeout.ms	消费者去kafka broker拿去一条消息超时时间

可以看到，Kafka的consumer是非常高效的。它直接从broker的文件系统里读取文件块。Kafka使用sendfile API来直接通过操作系统直接传输，而不用把数据拷贝到用户空间。该项测试实际上从log的起始处开始读数据，所以它做了真实的I/O。

在生产环境下，consumer可以直接读取producer刚刚写下的数据（它可能还在缓存中）。实际上，如果在生产环境下跑I/O stat，你可以看到基本上没有物理"读"。也就是说生产环境下consumer的吞吐率会比该项测试中的要高。

3.1 Consumer数

将上面的consumer复制到3台不同的机器上，并且并行运行它们（从同一个topic上消费数据）。consumer的吞吐率几乎线性增涨。

Consumer	Records/second	MB/second
1		
3		

运行命令：

bin/kafka-consumer-perf-test.sh--zookeeper localhost:2181/kafka --messages 50000000 --topic test-rep-two_2--threads 1

结果：

start.time,end.time, compression, message.size, batch.size, total.data.sent.in.MB, MB.sec,total.data.sent.in.nMsg, nMsg.sec

3.2 分区数

消费等量不同topic的数据，消费者数相同，不同topic分区数不同，查看性能变化。

操作	线程	partition	replication	Records/second	MB/second

1	*1*	*6*	*3*		
2	*1*	*12*	*3*		

3.3 线程数

消费同一个topic的数据，消费者数相同，改变线程数，查看性能变化。一般线程数大于等于分区数。

操作	线程	partition	replication	Records/second	MB/second
1	*1*	*6*	*3*		
2	*3*	*6*	*3*		

4、Producer and Consumer

上面的测试只是把producer和consumer分开测试，而该项测试同时运行producer和consumer，这更接近使用场景。实际上目前的replication系统中follower就相当于consumer在工作。

该项测试，在具有6个partition和3个replica的topic上同时使用1个producer和1个consumer，并且使用异步复制。

Producer	Consumer	Records/second	MB/second
1	*1*		

可以看到，该项测试结果与单独测试1个producer时的结果几乎一致。所以说consumer非常轻量级。

6、负载均衡

6.1 Producer

(1) 创建具有多个partition的topic，生产数据，观察partition在broker集群的分布及数据量大小状况。若数据均匀分布，producer负载均衡功能良好。

(2) 开发自己的分区规则，查看分区状况

6.2 Consumer

(1) 开启多个Consumer属于同一个Consumer Group，消费同一个topic数据，查看各Consumer消费数据量是否均衡。

(2) 消费过程中挂掉其中一个Consumer，查看数据消费量变化。

7、端到端延迟(待定)

上文中讨论了吞吐率，那消息传输的latency如何呢？也就是说消息从producer到consumer需要多少时间呢？该项测试创建1个producer和1个consumer并反复计时。结果是，2 ms (median), 3ms (99th percentile,14ms (99.9th percentile)。（这里并没有说明topic有多少个partition，也没有说明有多少个replica，replication是同步还是异步。实际上这会极大影响producer发送的消息被commit的latency，而只有committed的消息才能被consumer所消费，所以它会最终影响端到端的latency）

posted on 2016-11-02 16:50 XIAO_DF 阅读(5289) 评论(1) 编辑 收藏