## Kafka测试报告

### 进程失效测试

目的：Broker失效，看是否会丢失消息，看业务停顿时长。

准备：5台hicloud虚机（5台ubuntu 16.04 64位，均为4核16G内存），安装Kafka 0.10.2，ZooKeeper 3.4.9。创建主题my-topic，复制数=3，分区数=1，消息约20字节，100万条，batch.size=16384，linger.ms=1。

方法：

HiProducer用来发消息。HiConsumer用来收消息。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-FO-0001 | 正常发消息 | 每秒约8万笔 |
| KFK-FO-0002 | 杀leader进程，ack=0 | 停顿时间6秒，丢失7000条消息 |
| KFK-FO-0003 | 杀leader进程，ack=1，retries=0 | 停顿时间6秒，丢失800条消息 |
| KFK-FO-0004 | 杀leader进程，ack=-1，retries=0 | 停顿时间7秒，丢失1092条消息 |
| KFK-FO-0005 | 杀leader进程，ack=-1，retries=3 | 停顿时间6秒，消息无丢失 |
| KFK-FO-0006 | 杀zk leader，ack=-1，retries=3 | 无影响 |
| KFK-FO-0007 | 杀controller，ack=-1，retries=3 | 无影响 |
| KFK-FO-0008 | 杀coordinator（同时也是follower），ack=-1，retries=3 | （偶尔）停顿时间6秒，消息重复200条 |
| KFK-FO-0009 | 杀coordinator（同时也是leader），ack=-1，retries=3 | （偶尔）停顿时间7秒，消息重复230条 |

分析：

Broker失效切换会有几秒钟停顿。如果设置ack=-1，retries不为0，则可保证不丢失订单。偶尔会出现重复消息，触发条件尚未明确。

### 主题规模测试

目的：主题、分区达到一定规模后，观察对性能的影响，观察占用zookeeper空间情况。

准备：broker数=5，消息约1000字节，复制数=3，20万条，batch.size=16384，linger.ms=1；ack=-1，retries=3。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-MT-0000 | 正常发消息，主题数=1，分区数=1 | 每秒约8000笔，平均延迟3.8秒 |
| n/a | 主题数=1，分区数=10 |  |
| n/a | 主题数=10，分区数=1 |  |
| n/a | 主题数=10，分区数=10 |  |
| n/a | 主题数=1，分区数=10，消费者数=10 |  |
| n/a | 主题数=100，分区数=10 |  |
| n/a | 主题数=1000，分区数=10 |  |
| n/a | 主题数=100，分区数=10，zk leader切换 |  |
| KFK-MT-0001 | 当前主题 | Znode=3, watch=0/3/25, data\_size=12816 |
| KFK-MT-0002 | 2000+主题\*10分区：TT1~TT2000 | Znode=47292, watch=0/2078, data\_size=3573306 |

分析：

批量用kafka-topics.sh --create创建主题，速度很慢，数小时后空间耗尽，操作未成功。

### Zk bypass测试

目的：观察zookeeper失效对kafka的影响，评估能否bypass zookeeper的影响。

准备：5个zk节点；broker数=5，消息约1000字节，复制数=3，20万条；ack=-1；my-topic：s2、s3、s4。

方法：

HiProducer用来发消息。HiConsumer用来收消息。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-BP-0000 | 正常发消息 | 20秒（也有过39秒），平均延迟3.1秒，最大延迟4秒，最大接收gap 0.5秒 |
| KFK-BP-0001 | 杀leader zk（s4） | 27秒，平均延迟3.6秒，最大延迟5.3秒，最大接收gap 0.65秒 |
| KFK-BP-0002 | 杀第2个zk节点 | 26秒，平均延迟3.5秒，最大延迟5秒，最大接收gap 0.75秒 |
| KFK-BP-0003 | 杀第3个zk节点 | 26秒，平均延迟3.5秒，最大延迟4.7秒，最大接收gap 0.7秒；消息继续收发，但kafka出现大量报错 |
| n/a | 杀连接zk节点（s2）（hifi：是配置错误） | S2持续报错，Consumer挂起；杀掉s2后才恢复；原因是kafka只配了本地zk，改为配多个zk |
| KFK-BP-0004 | 连续杀3个zk节点（s1, s2, s3） | 36秒，平均延迟5秒，最大延迟9秒，最大接收gap 0.5秒；消息继续收发，但kafka出现大量报错 |
| KFK-BP-0005 | 断开3个zk节点，发送消息 | Kafka正常收发；28秒，平均延迟3.9秒，最大延迟5.4秒，最大接收gap 0.55秒 |
| n/a | 断开3个zk节点（s2, s3, s4） | Kafka报错；此时zkCli也不可用 |
| KFK-BP-0006 | 断开所有zk节点 | 仍可消息收发 |
| KFK-BP-0007 | 断开所有zk节点，杀s3 | 无法消息收发 |
|  |  |  |

分析：

确认在Kafka新版本中，分区leader信息在Broker中缓存，因此可以一定程度上与zookeeper解耦，在zookeeper不可用的情况下，Kafka一方面持续报错，另一方面可以正常转发消息。

### Zk影响面测试

目的：进一步分析zookeeper失效影响。

准备：创建指定到s1~s5的主题test1~test5。

分析：

Broker失效，即便其上的主题与其它节点没有关联，但也使得Kafka所有节点均不可用。python客户端在zk失效后无法使用；java客户端在zk失效后仍可使用，但在杀s3之后即不可用。

### Zk亚健康测试

目的：模拟zookeeper亚健康，观察对Kafka的影响

准备：将Zk进程kill -STOP，此时本机zk不可用，可用kill -CONT恢复。如果zk杀掉3个，则无法使用kill -CONT恢复，杀掉其中一个zk后重起，全部恢复。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-SH-0001 | 暂停1-2个zk | 无影响 |
| KFK-SH-0002 | 暂停3个zk | 可正常收发消息，kafka持续报警 |
|  |  |  |

分析：

暂停zookeeper与杀进程效果相似。

### 发送批量试验

目的：观察producer在不同批量参数下的消息处理吞吐量。

准备：20万条消息，消息大小1k字节，3副本全确认（s2,s3,s4），s1发，s1收。

修改Producer参数可直接在HiProducer.java中进行，重新编译运行即可，不需要重启Kafka。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-BA-0001 | batch.size=16384 | 13秒发送完成，15秒接收完成，轮询11363次，2.2秒延迟 |
| KFK-BA-0002 | batch.size=8192 | 24秒发送完成，28秒接收完成，轮询26562次，3.7秒延迟 |
| KFK-BA-0003 | batch.size=163840 | 4.5秒发送完成，5.6秒接收完成，轮询911次，0.7秒延迟 |
| KFK-BA-0004 | batch.size=1638400 | 5.3秒发送完成，6.2秒接收完成，轮询486次，0.8秒延迟 |
|  |  |  |

分析：

消息打包大小对于producer发送性能有很大的影响，同时也减少了consumer轮询次数。

### 轮询效率试验

目的：观察consumer在不同场景下的处理速度。

准备：20万条消息，消息大小1k字节，3副本全确认（s2,s3,s4），22秒发送完成，s1发，s1收。Consumer缺省参数设置：max.poll.records=500，fetch.min.bytes=1，fetch.max.wait.ms=500，max.partition.fetch.bytes=1048576，receive.buffer.bytes=65536，fetch.max.bytes=52428800。

修改Consumer参数可直接在HiConsumer.java中进行，重新编译运行即可，不需要重启Kafka。

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-PO-0001 | 实时接收 | 26秒接收完成，轮询9936次，3.6秒延迟 |
| KFK-PO-0002 | 离线接收 | 5秒接收完成，轮询405次 |
| KFK-PO-0003 | 开始发送约10秒后开始接收 | 12秒接收完成，轮询3989次 |
|  |  |  |
| 再次试验 | 发送 | 15秒发送完成 |
| KFK-PO-0001 | 实时接收 | 17秒接收完成，轮询12498次，2.4秒延迟 |
| KFK-PO-0002 | 离线接收 | 2.6秒接收完成，轮询405次 |
| KFK-PO-0003 | 开始发送约10秒后开始接收 | 6.5秒接收完成，轮询6543次 |
| KFK-PO-0004 | 实时接收，将max.poll.records放大到5000 | 16秒接收完成，轮询11756次，2.2秒延迟 |
| KFK-PO-0005 | 实时接收，将max.poll.records放大到5000，fetch.min.bytes设为2000000，max.partition.fetch.bytes设为2048000 | 14秒接收完成，轮询106次，2秒延迟 |
| KFK-PO-0006 | 实时接收，将max.poll.records放大到5000，fetch.min.bytes设为5000000，max.partition.fetch.bytes设为5000000 | 14秒接收完成，轮询43次，2.2秒延迟 |
|  |  |  |

分析：

从离线接收场景看，消息打包大小对于consumer消费性能有很大的影响。但尚未能构造出消息积压场景来做验证。

### Rebalance测试

目的：对Leader Rebalance机制进行试验，观察分区leader不均衡情况下Kafka如何自动处置，分析评估rebalance对性能和可用性的影响。

准备：5台hicloud虚机（2台centos 7.2 64位，3台ubuntu 16.04 64位，均为4核16G内存），安装Kafka 0.10.2，ZooKeeper 3.4.9。创建主题testb，复制数=3，分区数=10。

方法：

RbProducer用来发消息。RbConsumer用来收消息。

创建分区工具：

* kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 3 --partitions 10 --topic testb

显示分区工具：

* kafka-topics.sh --describe --zookeeper localhost:2181 --topic testb

手工杀掉S4上的kafka，然后再把S4上的kafka拉起来

调整leader工具：

* kafka-preferred-replica-election.sh --zookeeper localhost:2181

调整分区工具：

* kafka-reassign-partitions.sh --zookeeper localhost:2181 --broker-list "1,2,3,4,5" --topics-to-move pmove1.json --generate
* kafka-reassign-partitions.sh --zookeeper localhost:2181 --reassignment-json-file pmove1.json --execute
* kafka-reassign-partitions.sh --zookeeper localhost:2181 --reassignment-json-file pmove1.json --verify

缩放分区工具：

* kafka-topics.sh --zookeeper localhost:2181 --alter --partitions 15 --topic testb

查看消费者组：

* kafka-consumer-groups.sh --bootstrap-server S2:9092 --group mtest --describe

日志内容：

* controller.log：在S3（controller节点）上看，每5分钟会检查一次分区是否均衡；分区leader选举
* server.log：每10分钟会检查一次有无过期offset；记录broker动作：监听线程、元数据加载、ISR调整、消费组调整（restablize）
* state-change.log：记录broker上的交互消息（如果是controller，还记录controller交互消息），如元数据同步、leader/follower角色调整、replica集合、ISR同步等
* zookeeper日志：可以看到session termination信息

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-RB-0001 | 正常情况 | 10个分区leader轮流由1-5担任 |
| KFK-RB-0002 | Broker S4失效 | 有2个分区leader因4失效而调整，各个分区ISR均去掉4 |
| KFK-RB-0003 | Broker S4失效后再恢复 | 首先是Replicas增加4，过一会ISR中增加4，过一段时间后有2个分区leader调整回4，形成均衡状态 |
| KFK-RB-0004 | 启用两个consumer | 可以看到每个consumer消费5个分区 |
| KFK-RB-0005 | 杀掉其中1个consumer | 可以看到剩下一个consumer消费10个分区 |
| KFK-RB-0006 | 增至15个分区 | 首先是创建分区，过3分钟后consumer重新分配分区 |
| KFK-RB-0007 | 减至5个分区 | 报错：不允许减少分区 |
| KFK-RB-0008 | 持续发送消息过程中，Broker S4失效 | 接收消息未见停顿 |
| KFK-RB-0009 | 持续发送消息过程中，Broker S4失效后再恢复 | 过一会分区leader调整为均衡状态，未见明显影响 |
| KFK-RB-0010 | 持续发送消息过程中，杀掉两个consumer中的一个 | 剩下一个consumer接收完原分区消息后，过一会又接收到其它分区消息 |
| KFK-RB-0011 | 持续发送消息过程中，启用第2个consumer | 瞬间重新分配分区，两个consumer正常接收消息 |
|  |  |  |

分析：

Server.log日志中的关键字：

* Broker节点失效：Loading offsets；Finished；Shutting down；completed
* Broker节点恢复：Expanding ISR；Error for partition … not the leader
* Consumer重分配：restabilize group；Stabilized group；Assignment
* 扩大分区：created log for partition

观察工具：

* kafka-topics.sh --describe，可以看分区leader和ISR
* kafka-consumer-groups.sh --describe，可以看consumer所分配的分区变化

### 内存盘测试

目的：使用tmpfs内存盘存储数据文件，观察消息处理性能，试验存储bypass效果。

准备：5台hicloud虚机（2台centos 7.2 64位，3台ubuntu 16.04 64位，均为4核16G内存），安装Kafka 0.10.2，ZooKeeper 3.4.9。创建主题my-topic，复制数=3，分区数=1，broker数=5，消息约1000字节，复制数=3，100万条，batch.size=16384，linger.ms=1；ack=-1，retries=3。

创建内存盘mdisk，大小够存放一个批次的消息，再打一批即会溢出：

* mkdir ~/mdisk
* sudo mount -t tmpfs -o size=300M mdisk ~/mdisk

在配置文件server.properties中设置数据目录：

* sed -i "s:log.dirs.\*$:log.dirs=/home/hifi/mdisk/kafka-logs:" /usr/local/kafka/config/server.properties

配置多条数据目录：

* mkdir ~/sdisk
* sudo mount -t tmpfs -o size=300M sdisk ~/sdisk
* sed -i "s:log.dirs.\*$:log.dirs=/home/hifi/mdisk/kafka-logs,/home/hifi/sdisk/kafka-logs:" /usr/local/kafka/config/server.properties

设置日志清理策略：

* sed -i "s:#log.retention.bytes.\*$:log.retention.bytes=150000000:" /usr/local/kafka/config/server.properties
* sed -i "s:log.retention.hours.\*$:log.retention.minutes=2:" /usr/local/kafka/config/server.properties

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-MD-0001 | 正常打一批消息，观察内存盘性能 | 5秒发完，7秒收完，轮询4113次，每秒14万条消息，延迟1.1秒；磁盘占用138M，46% |
| KFK-MD-0002 | 发送多批消息，看内存盘是否溢出报错 | 第3批导致磁盘满，S2、S3、S4退出，S1、S5因连接中断报警；清空内存盘后重启节点，Kafka可正常工作 |
| KFK-MD-0003 | 设置多个数据目录，第一个目录满了是否会访问其它目录 | 第一个目录满后Kafka退出，重启后亦退出 |
| KFK-MD-0004 | 设置日志清理策略，限制log.retention.bytes大小为150000000，连续打多批消息 | 第3批导致磁盘满，S2、S3、S4退出 |
| KFK-MD-0005 | 设置日志清理策略，限制log.retention.bytes大小为150000000，打两批消息后等待一段时间 | 日志大小275M，始终没有清理（应该5分钟检查一次） |
| KFK-MD-0006 | 设置日志清理策略，限制log.retention.minutes大小为2，打两批消息后等待一段时间 | 日志生命周期为2分钟，等待约5分钟后，日志文件先被标注delete，随后被删除，空间被释放 |
|  |  |  |

分析：

内存盘使得消息转发效率提升17倍。多数据目录尚不能自动切换（Kafka1.0提供该功能？）。日志清理策略不能立刻触发，需要定时触发。

### 跨集群复制MirrorMaker测试

目的：试验MirrorMaker效果。

准备：5台hicloud虚机（2台centos 7.2 64位，3台ubuntu 16.04 64位，均为4核16G内存）构成两个集群（S1+S2，S3+S4+S5）。安装Kafka 0.10.2，ZooKeeper 3.4.9。创建主题my-topic，复制数=2，分区数=1。batch.size=16384。

清除过往Kafka日志、数据，清除过往ZooKeeper元数据

* rm -rf /usr/local/kafka/logs/\*
* rm -rf /tmp/kafka\*/\*
* rm -rf /usr/local/zookeeper/version-2/\*

修改配置文件，重新设置两个zookeeper和kafka集群

在两边集群同步创建同样的主题

设置consumer.properties（消费组input\_cluster）和producer.properties，运行MirrorMaker

* kafka-run-class.sh kafka.tools.MirrorMaker --consumer.config consumer.properties --num.streams 2 --producer.config producer.properties --whitelist=".\*"

查看复制进度：

* kafka-consumer-groups.sh --bootstrap-server s1:9092 --describe --group input\_cluster
* kafka-consumer-groups.sh --bootstrap-server s3:9092 --describe --group test

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-MM-0001 | 向集群1发送消息，从集群2接收 | 16秒发完，27秒收完，13203次轮询。 |
|  |  |  |

分析：

MirrorMaker可用。接收消息耗时较多，参数还可以优化。

### 日志文件测试

目的：测试kafka重建日志索引的场景，观察有何标志性信息

准备：

日志位置：/tmp/kafka-logs/my-topic-0

Offset日志位置：/tmp/kafka-logs/\_\_consumer\_offsets-48

清除过往ZooKeeper元数据

* rm -rf /usr/local/zookeeper/version-2/\*

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-LG-0001 | 删除my-topic的index和timeindex文件，持续打单；创建空文件，持续打单 | 消息收发正常，index文件未重建 |
| KFK-LG-0002 | 等待日志滚动 | Server.log报kafka.log.\*：\_\_consumer\_offsets-48 删除 |
| KFK-LG-0003 | 删除index文件后，重启broker | 开机期间检测到index文件损坏，自动重建，Server.log报kafka.log.\*：corrupted index，完成报Completed load of log，最后完成有个Logs loading complete，完全启动完成有个kafka.server.KafkaServer started |
| KFK-LG-0004 | 关闭kafka和zk，删除所有zk数据，再重启kafka，尝试打单 | S4发现若干index文件损坏（之前S4没启用），自动重建；打单过程中S3和S5（两个follower，初始值为0）\_\_consumer\_offsets-48报out of range和reset offset |
| KFK-LG-0005 | 关闭kafka和zk，在S3、S4和S5上删除所有zk数据，删除my-topic、\_\_consumer\_offsets-48主题的index和timeindex文件再重启kafka，尝试打单 | S4发现若干index文件损坏，但S3、S5却没告警；S5（leader）告警reset offset，但S3、S4却没告警 |

分析：

日志Index rebuild仅在Kafka启动时实施。删除index文件会引发index rebuild。删除zk数据后，leader会从日志中取出最新HW，consumer会通过reset重置offset。

正常重启有时也会遇到index rebuild。

### 日志老化测试

目的：测试kafka的日志老化机制，尤其是内部日志的老化机制

准备：

持续打单：每秒1笔1K大小的消息，一小时数据量是3.6MB，一天数据量是86MB；每秒1笔32字节的offset，一小时数据量是115KB，一天数据量是2.8MB

断续消费：每次继续之前的消费位置，消费指定条数后退出，打印出消费的起止位置

Broker参数：

* 日志老化类
* log.retention.bytes: 不设置（缺省：-1），不起作用
* log.retention.hours：1，每一小时做一次老化
* log.roll.ms：1200000，每20分钟归档一次
* log.segment.bytes: 不设置（缺省：1073741824），不起作用
* log.segment.delete.delay.ms，不设置（缺省60000），滞后1分钟再删除
* log.index.interval.bytes，不设置（缺省4096），每4096个记录做一个索引
* 内部日志（offset）类
* offsets.retention.minutes，30，每30分钟做一次offset老化
* offsets.topic.segment.bytes，1048576，内部日志到1M就做归档
* 内部日志压缩（cleaner）类
* log.cleaner.delete.retention.ms，不设置（缺省86400000），1天后删除tombstone？
* log.cleaner.min.cleanable.dirty.ratio，不设置（缺省0.5），超过50%自动做压缩
* log.cleaner.min.compaction.lag.ms，不设置（缺省0），意义不明
* log.retention.check.interval.ms，不设置（缺省300000），5分钟做一次日志压缩检查

Consumer参数：

* auto.commit.interval.ms，1000，每隔1秒钟提交offset到内部日志

使用S3-S5主机，主题为my-topic，消费组为test（对应内部日志第48分区，S3为主），副本数为1，分区数为1，S5为主

发送端用CtProducer，持续低速报单，直接ctrl-c或kill退出；支持断点续发

接收端用CtConsumer，功能：1）延续之前进度继续消费，每10秒钟输出当前消费进度和速度，接收退出指令；2）消费指定条数后退出；3）从最新位置开始消费，每10秒钟输出当前消费进度和速度，接收退出指令；直接ctrl-c或kill退出，实现优雅退出能力

日志位置：/tmp/kafka-logs/my-topic-0

Offset日志位置：/tmp/kafka-logs/\_\_consumer\_offsets-48

工具：

kafka-run-class.sh kafka.tools.DumpLogSegments --files /tmp/kafka-logs/\_\_consumer\_offsets-48/00000000000000024940.log --print-data-log

输出：

offset: 34906 position: 839315 CreateTime: 1516700245862 isvalid: true payloadsize: 28 magic: 1 compresscodec: NONE crc: 3118446024 keysize: 22 key: tesmy-topic payload:

工具：

kafka-run-class.sh kafka.tools.DumpLogSegments --files /tmp/kafka-logs/\_\_consumer\_offsets-48/00000000000000024940.log --offsets-decoder

输出：

offset: 25699 position: 65927 CreateTime: 1516691037556 isvalid: true payloadsize: 28 magic: 1 compresscodec: NONE crc: 698516620 keysize: 22 key: offset::test:my-topic:0 payload: 1200012

清除S3-S5所有zk和kafka数据

* rm -rf /usr/local/kafka/logs/\*
* rm -rf /tmp/kafka\*/\*
* rm -rf /usr/local/zookeeper/version-2/\*

创建my-topic主题

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-RT-0101 | 持续打单和消费，观察业务日志和内部日志增长速度 |  |
| KFK-RT-0102 | 持续打单，验证消费端重连后可以断点续传 |  |
| KFK-RT-0103 | 持续打单和消费，观察30分钟和1MB大小时内部日志是否会归档和老化 |  |
| KFK-RT-0104 | 持续打单，观察业务日志是否每20分钟做归档、每1小时做老化 |  |
| KFK-RT-0105 | 持续打单，待内部日志老化后，验证消费端重连后可以断点续传 |  |
| KFK-RT-0106 | 持续打单，待业务日志老化后，验证消费端重连后offset越界 |  |
| KFK-RT-0107 | 持续打单，回滚消费进度 |  |
| KFK-RT-0108 | 持续打单，待内部日志老化后，回滚消费进度 |  |

分析：

内部日志似乎没做压缩，每条记录84字节（key 22字节，消息体28字节，overhead 34字节）

### 副本切换测试

目的：测试副本数和主备切换逻辑

准备：

调整主题副本数（不被允许，无法执行）：

~~kafka-topics.sh --zookeeper localhost:2181 --alter --replication-factor 2 --topic test2~~

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
|  | 仅1节点，创建2副本主题 | 主题创建失败，副本数超出节点数 |
|  | 2节点，创建2副本主题 | 主题创建成功 |
|  | 杀备节点 | 主题显示正常，副本数=2，ISR只有单节点 |
|  | 恢复备节点，杀主节点 | 主题显示正常，leader发生切换，副本数=2，ISR只有单节点 |
|  | 创建1副本主题，杀主节点 | 主题显示不可用，leader=-1；节点重启后恢复正常 |
|  | 调整主题副本数为2 | 不成立 |
|  | 初次自动创建\_\_consumer\_offsets主题，只有单副本 | 单节点时自动创建，之后增加节点 |

分析：

### 端口复制测试

目的：测试端口复制对性能的影响

准备：

原始发送接收客户端：HiProducer/HiConsumer

代理发送接收客户端：HxProducer/HxConsumer

代理工具：kproxy.py

统计工具：maxgap.py

结果：（20万消息，单副本，单分区，接入节点S4，主节点S5，2018.1.31，21:49开始）

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
|  |  | 9342ms，1137ms |
|  |  | 9291ms，1552ms |
|  |  | 10439ms，1560ms |
|  | 代理 | 时长10400ms，时延1365ms |
|  |  | 10080ms，2007 |
|  |  | 11026，1576ms |
|  |  |  |

分析：

### 磁盘切换测试

目的：验证kafka1.1.1引入的磁盘切换能力

准备：3台hicloud虚机（3台ubuntu 16.04 64位，均为4核16G内存）构成集群（S3+S4+S5）。安装Kafka 1.1.1，ZooKeeper 3.4.9。创建主题my-topic，复制数=3，分区数=1。使用HiProducer、HiConsumer，一批打100万条消息。

清除Kafka日志、数据，清除过往ZooKeeper元数据

* rm -rf /usr/local/kafka/logs/\*
* rm -rf /usr/local/kafka111/logs/\*
* rm -rf /tmp/kafka\*/\*
* rm -rf /usr/local/zookeeper/version-2/\*

创建内存盘mdisk，大小够存放一个批次（100万条）的消息，再打一批即会溢出：

* mkdir ~/mdisk
* mkdir ~/sdisk
* sudo mount -t tmpfs -o size=300M mdisk ~/mdisk
* sudo mount -t tmpfs -o size=300M sdisk ~/sdisk
* sed -i "s:log.dirs.\*$:log.dirs=/home/hifi/mdisk/kafka-logs,/home/hifi/sdisk/kafka-logs:" /usr/local/kafka111/config/server.properties

恢复：

* sed -i "s:log.dirs.\*$:log.dirs=/tmp/kafka-logs:" /usr/local/kafka111/config/server.properties

调整分区：

* kafka-reassign-partitions.sh --zookeeper localhost:2181 --broker-list "3,4,5" --topics-to-move-json-file topics.json --generate

结果：

|  |  |  |
| --- | --- | --- |
| 编号 | 用例 | 结果 |
| KFK-DK-0001 | 创建topic（分区数=2） | 在多个目录下平均分配分区 |
| KFK-DK-0002 | 100万订单打满磁盘（分区数=1） | 磁盘满，各个副本都不能工作，但是broker还存活 |
| KFK-DK-0002 | 再创建一个topic，继续打单100万 | 另一个内存盘也满了，broker退出 |
|  |  |  |
|  |  |  |
|  |  |  |

分析：

### 用例索引

* KFK-FO-nnnn：进程失效用例
* KFK-MT-nnnn：主题规模用例
* KFK-BP-nnnn：Zk Bypass用例
* KFK-SH-nnnn：Zk亚健康用例
* KFK-BA-nnnn：发送批量用例
* KFK-PO-nnnn：轮询效率用例
* KFK-RB-nnnn：再均衡（rebalance）用例
* KFK-MD-nnnn：内存盘用例
* KFK-MM-nnnn：跨集群复制（MirrorMaker）用例
* KFK-LG-nnnn：日志文件用例
* KFK-RT-nnnn：日志老化用例
* KFK-DK-nnnn：磁盘切换用例