**1、kafka环境搭建总结**

http://blog.csdn.net/shymi1991/article/details/52971994

http://blog.csdn.net/tangdong3415/article/details/53432166

1启动kafka(启动之前先启动zookeeper)

sh bin/kafka-server-start.sh config/server.properties

2创建一个test的topic

topic 名字为test，2个副本和1个分块

sh bin/kafka-topics.sh --create --zookeeper 192.168.40.46:2181 --replication-factor 2 --partitions 1 --topic test

提示：Created topic "test".

可以通过list命令查看创建的topic:

sh bin/kafka-topics.sh --list --zookeeper 192.168.40.46:2181

显示：test

3运行producer并在控制台中输一些消息，这些消息将被发送到服务端：

sh bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test

4Kafka也有一个命令行consumer可以读取消息并输出到标准输出：

sh bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

describe topics命令来观察哪一个代理在处理该topic。

sh bin/kafka-topics.sh --describe --zookeeper 192.168.40.46:2181 --topic my-replicated-topic

第一行是topic概观，描述该topic的信息。接下来每一行显示该topic一个分块的信息。因为我们这个topic只有一个分块，所以只有一行。

Leader：负责该分块的读写，随机选举出来的节点；

replica：复制分块的节点；

isr：是replica的子集，leader的候选集，即slave。

在该例子中，node 1是mytest2唯一一个分块的leader。

Topic:mytest2 PartitionCount:1 ReplicationFactor:2 Configs:

Topic: mytest2 Partition: 0 Leader: 1 Replicas: 1,2 Isr: 1,2

容错测试。我们kill作为leader的Broker 1：

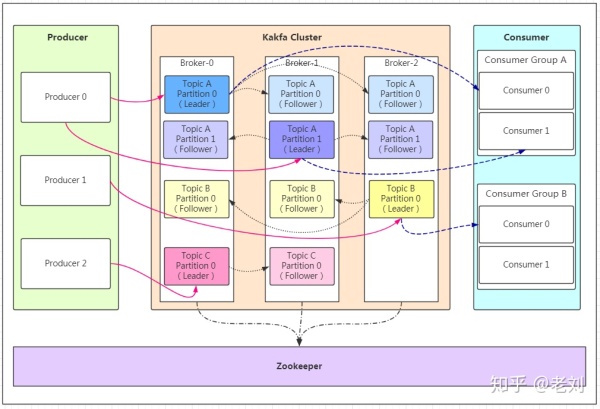
ps aux | grep server2.properties

**2、kafka原理总结**

作者：老刘  
链接：https://zhuanlan.zhihu.com/p/68052232  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**基础架构及术语**

话不多说，先看图，通过这张图我们来捋一捋相关的概念及之间的关系：



如果看到这张图你很懵逼，木有关系！我们先来分析相关概念

**Producer**：Producer即生产者，消息的产生者，是消息的入口。

**kafka cluster**：

**Broker**：Broker是kafka实例，每个服务器上有一个或多个kafka的实例，我们姑且认为每个broker对应一台服务器。每个kafka集群内的broker都有一个**不重复**的编号，如图中的broker-0、broker-1等……

**Topic**：消息的主题，可以理解为消息的分类，kafka的数据就保存在topic。在每个broker上都可以创建多个topic。

**Partition**：Topic的分区，每个topic可以有多个分区，分区的作用是做负载，提高kafka的吞吐量。同一个topic在不同的分区的数据是不重复的，partition的表现形式就是一个一个的文件夹！

**Replication**:每一个分区都有多个副本，副本的作用是做备胎。当主分区（Leader）故障的时候会选择一个备胎（Follower）上位，成为Leader。在kafka中默认副本的最大数量是10个，且副本的数量不能大于Broker的数量，follower和leader绝对是在不同的机器，同一机器对同一个分区也只可能存放一个副本（包括自己）。

**Message**：每一条发送的消息主体。

**Consumer**：消费者，即消息的消费方，是消息的出口。

**Consumer Group**：我们可以将多个消费组组成一个消费者组，在kafka的设计中同一个分区的数据只能被消费者组中的某一个消费者消费。同一个消费者组的消费者可以消费同一个topic的不同分区的数据，这也是为了提高kafka的吞吐量！

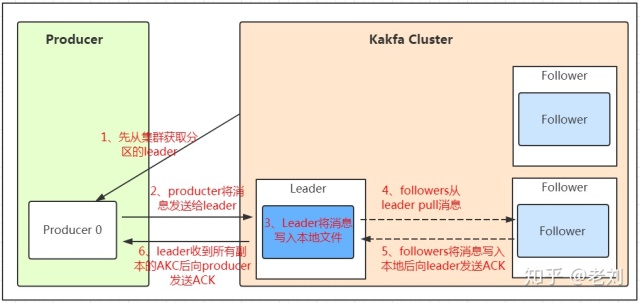
**Zookeeper**：kafka集群依赖zookeeper来保存集群的的元信息，来保证系统的可用性。

**工作流程分析**

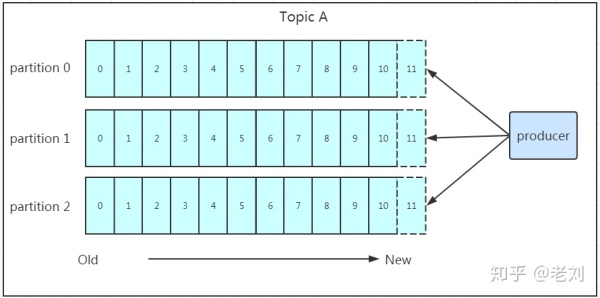
上面介绍了kafka的基础架构及基本概念，不知道大家看完有没有对kafka有个大致印象，如果对还比较懵也没关系！我们接下来再结合上面的结构图分析kafka的工作流程，最后再回来整个梳理一遍我相信你会更有收获！

**发送数据**

我们看上面的架构图中，producer就是生产者，是数据的入口。注意看图中的红色箭头，Producer在写入数据的时候**永远的找leader**，不会直接将数据写入follower！那leader怎么找呢？写入的流程又是什么样的呢？我们看下图：



发送的流程就在图中已经说明了，就不单独在文字列出来了！需要注意的一点是，消息写入leader后，follower是主动的去leader进行同步的！producer采用push模式将数据发布到broker，每条消息追加到分区中，顺序写入磁盘，所以保证**同一分区**内的数据是有序的！写入示意图如下：



上面说到数据会写入到不同的分区，那kafka为什么要做分区呢？相信大家应该也能猜到，分区的主要目的是：

**1、 方便扩展**。因为一个topic可以有多个partition，所以我们可以通过扩展机器去轻松的应对日益增长的数据量。

**2、 提高并发**。以partition为读写单位，可以多个消费者同时消费数据，提高了消息的处理效率。

熟悉负载均衡的朋友应该知道，当我们向某个服务器发送请求的时候，服务端可能会对请求做一个负载，将流量分发到不同的服务器，那在kafka中，如果某个topic有多个partition，producer又怎么知道该将数据发往哪个partition呢？kafka中有几个原则：

1、 partition在写入的时候可以指定需要写入的partition，如果有指定，则写入对应的partition。

2、 如果没有指定partition，但是设置了数据的key，则会根据key的值hash出一个partition。

3、 如果既没指定partition，又没有设置key，则会轮询选出一个partition。

保证消息不丢失是一个消息队列中间件的基本保证，那producer在向kafka写入消息的时候，怎么保证消息不丢失呢？其实上面的写入流程图中有描述出来，那就是通过ACK应答机制！在生产者向队列写入数据的时候可以设置参数来确定是否确认kafka接收到数据，这个参数可设置的值为**0**、**1**、**all**。

0代表producer往集群发送数据不需要等到集群的返回，不确保消息发送成功。安全性最低但是效率最高。

1代表producer往集群发送数据只要leader应答就可以发送下一条，只确保leader发送成功。

all代表producer往集群发送数据需要所有的follower都完成从leader的同步才会发送下一条，确保leader发送成功和所有的副本都完成备份。安全性最高，但是效率最低。

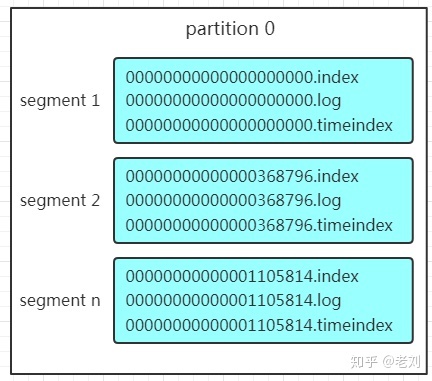
最后要注意的是，如果往不存在的topic写数据，能不能写入成功呢？kafka会自动创建topic，分区和副本的数量根据默认配置都是1。

**保存数据**

Producer将数据写入kafka后，集群就需要对数据进行保存了！kafka将数据保存在磁盘，可能在我们的一般的认知里，写入磁盘是比较耗时的操作，不适合这种高并发的组件。Kafka初始会单独开辟一块磁盘空间，顺序写入数据（效率比随机写入高）。

**Partition 结构**

前面说过了每个topic都可以分为一个或多个partition，如果你觉得topic比较抽象，那partition就是比较具体的东西了！Partition在服务器上的表现形式就是一个一个的文件夹，每个partition的文件夹下面会有多组segment文件，每组segment文件又包含.index文件、.log文件、.timeindex文件（早期版本中没有）三个文件， log文件就实际是存储message的地方，而index和timeindex文件为索引文件，用于检索消息。



如上图，这个partition有三组segment文件，每个log文件的大小是一样的，但是存储的message数量是不一定相等的（每条的message大小不一致）。文件的命名是以该segment最小offset来命名的，如000.index存储offset为0~368795的消息，kafka就是利用分段+索引的方式来解决查找效率的问题。

**Message结构**

上面说到log文件就实际是存储message的地方，我们在producer往kafka写入的也是一条一条的message，那存储在log中的message是什么样子的呢？消息主要包含消息体、消息大小、offset、压缩类型……等等！我们重点需要知道的是下面三个：

1、 offset：offset是一个占8byte的有序id号，它可以唯一确定每条消息在parition内的位置！

2、 消息大小：消息大小占用4byte，用于描述消息的大小。

3、 消息体：消息体存放的是实际的消息数据（被压缩过），占用的空间根据具体的消息而不一样。

**存储策略**

无论消息是否被消费，kafka都会保存所有的消息。那对于旧数据有什么删除策略呢？

1、 基于时间，默认配置是168小时（7天）。

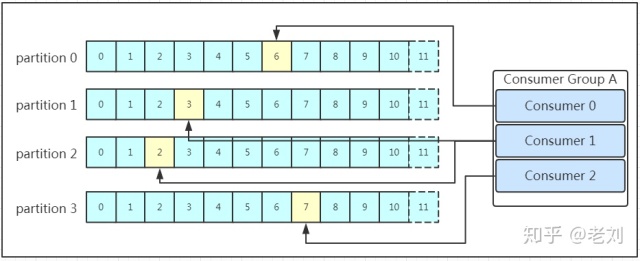
2、 基于大小，默认配置是1073741824。

需要注意的是，kafka读取特定消息的时间复杂度是O(1)，所以这里删除过期的文件并不会提高kafka的性能！

**消费数据**

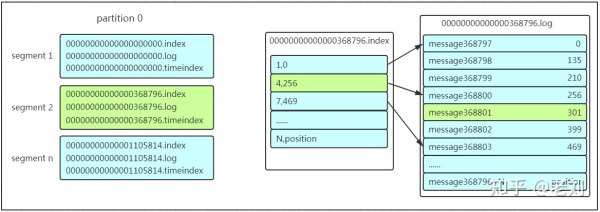
消息存储在log文件后，消费者就可以进行消费了。在讲消息队列通信的两种模式的时候讲到过点对点模式和发布订阅模式。Kafka采用的是点对点的模式，消费者主动的去kafka集群拉取消息，与producer相同的是，消费者在拉取消息的时候也是**找leader**去拉取。

多个消费者可以组成一个消费者组（consumer group），每个消费者组都有一个组id！同一个消费组者的消费者可以消费同一topic下不同分区的数据，但是不会组内多个消费者消费同一分区的数据！！！是不是有点绕。我们看下图：



图示是消费者组内的消费者小于partition数量的情况，所以会出现某个消费者消费多个partition数据的情况，消费的速度也就不及只处理一个partition的消费者的处理速度！如果是消费者组的消费者多于partition的数量，那会不会出现多个消费者消费同一个partition的数据呢？上面已经提到过不会出现这种情况！多出来的消费者不消费任何partition的数据。所以在实际的应用中，建议**消费者组的consumer的数量与partition的数量一致**！

在保存数据的小节里面，我们聊到了partition划分为多组segment，每个segment又包含.log、.index、.timeindex文件，存放的每条message包含offset、消息大小、消息体……我们多次提到segment和offset，查找消息的时候是怎么利用segment+offset配合查找的呢？假如现在需要查找一个offset为368801的message是什么样的过程呢？我们先看看下面的图：



1、 先找到offset的368801message所在的segment文件（利用二分法查找），这里找到的就是在第二个segment文件。

2、 打开找到的segment中的.index文件（也就是368796.index文件，该文件起始偏移量为368796+1，我们要查找的offset为368801的message在该index内的偏移量为368796+5=368801，所以这里要查找的**相对offset**为5）。由于该文件采用的是稀疏索引的方式存储着相对offset及对应message物理偏移量的关系，所以直接找相对offset为5的索引找不到，这里同样利用二分法查找相对offset小于或者等于指定的相对offset的索引条目中最大的那个相对offset，所以找到的是相对offset为4的这个索引。

3、 根据找到的相对offset为4的索引确定message存储的物理偏移位置为256。打开数据文件，从位置为256的那个地方开始顺序扫描直到找到offset为368801的那条Message。

这套机制是建立在offset为有序的基础上，利用**segment**+**有序offset**+**稀疏索引**+**二分查找**+**顺序查找**等多种手段来高效的查找数据！至此，消费者就能拿到需要处理的数据进行处理了。那每个消费者又是怎么记录自己消费的位置呢？在早期的版本中，消费者将消费到的offset维护zookeeper中，consumer每间隔一段时间上报一次，这里容易导致重复消费，且性能不好！在新的版本中消费者消费到的offset已经直接维护在kafk集群的\_\_consumer\_offsets这个topic中！