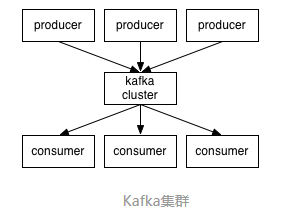
# ****1 基础理论****

这块是整个kafka的核心无论你是先操作在来看还是先看在操作都需要多看几遍。

首先来了解一下Kafka所使用的基本术语

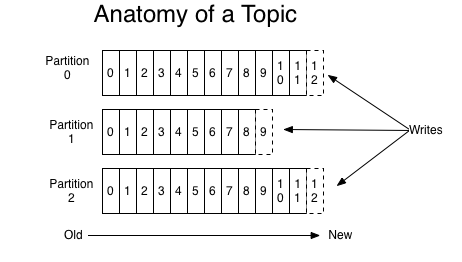
**Topic**  
Kafka将消息种子(Feed)分门别类 每一类的消息称之为话题(Topic).  
**Producer**  
发布消息的对象称之为话题生产者(Kafka topic producer)  
**Consumer**  
订阅消息并处理发布的消息的种子的对象称之为话题消费者(consumers)  
**Broker**  
已发布的消息保存在一组服务器中称之为Kafka集群。集群中的每一个服务器都是一个代理(Broker). 消费者可以订阅一个或多个话题并从Broker拉数据从而消费这些已发布的消息。  
  
让我们站的高一点从高的角度来看Kafka集群的业务处理就像这样子  
  
[](http://s4.51cto.com/wyfs02/M00/87/C8/wKiom1fhP-vRkftcAAAfOoQwTtk506.png)

Client和Server之间的通讯是通过一条简单、高性能并且和开发语言无关的TCP协议。除了Java Client外还有非常多的其它编程语言的Client。

### 话题和日志  (Topic和Log)

让我们更深入的了解Kafka中的Topic。

Topic是发布的消息的类别或者种子Feed名。对于每一个TopicKafka集群维护这一个分区的log就像下图中的示例

[](http://s4.51cto.com/wyfs02/M02/87/C4/wKioL1fhQHSQG62WAAA-QB83hq8979.png)

每一个分区都是一个顺序的、不可变的消息队列 并且可以持续的添加。分区中的消息都被分配了一个序列号称之为偏移量(offset)在每个分区中此偏移量都是唯一的。 Kafka集群保持所有的消息直到它们过期 无论消息是否被消费了。 实际上消费者所持有的仅有的元数据就是这个偏移量也就是消费者在这个log中的位置。 这个偏移量由消费者控制正常情况当消费者消费消息的时候偏移量也线性的的增加。但是实际偏移量由消费者控制消费者可以将偏移量重置为更老的一个偏移量重新读取消息。 可以看到这种设计对消费者来说操作自如 一个消费者的操作不会影响其它消费者对此log的处理。 再说说分区。Kafka中采用分区的设计有几个目的。一是可以处理更多的消息不受单台服务器的限制。Topic拥有多个分区意味着它可以不受限的处理更多的数据。第二分区可以作为并行处理的单元。

### 分布式(Distribution)

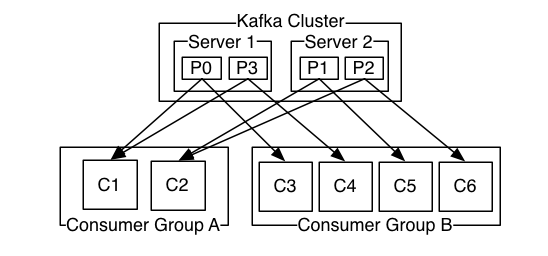
Log的分区被分布到集群中的多个服务器上。每个服务器处理它分到的分区。 根据配置每个分区还可以复制到其它服务器作为备份容错。 每个分区有一个leader零或多个follower。Leader处理此分区的所有的读写请求而follower被动的复制数据。如果leader宕机其它的一个follower会被推举为新的leader。 一台服务器可能同时是一个分区的leader另一个分区的follower。 这样可以平衡负载避免所有的请求都只让一台或者某几台服务器处理。

### 生产者(Producers)

生产者往某个Topic上发布消息。生产者也负责选择发布到Topic上的哪一个分区。最简单的方式从分区列表中轮流选择。也可以根据某种算法依照权重选择分区。开发者负责如何选择分区的算法。

### 消费者(Consumers)

通常来讲消息模型可以分为两种 队列和发布-订阅式。 队列的处理方式是 一组消费者从服务器读取消息一条消息只有其中的一个消费者来处理。在发布-订阅模型中消息被广播给所有的消费者接收到消息的消费者都可以处理此消息。Kafka为这两种模型提供了单一的消费者抽象模型 消费者组 consumer group。 消费者用一个消费者组名标记自己。 一个发布在Topic上消息被分发给此消费者组中的一个消费者。 假如所有的消费者都在一个组中那么这就变成了queue模型。 假如所有的消费者都在不同的组中那么就完全变成了发布-订阅模型。 更通用的 我们可以创建一些消费者组作为逻辑上的订阅者。每个组包含数目不等的消费者 一个组内多个消费者可以用来扩展性能和容错。正如下图所示

[](http://s1.51cto.com/wyfs02/M00/87/C8/wKiom1fhQUrTOmDcAABf0AAq7-s668.png)

  2个kafka集群托管4个分区P0-P32个消费者组消费组A有2个消费者实例消费组B有4个。

正像传统的消息系统一样Kafka保证消息的顺序不变。 再详细扯几句。传统的队列模型保持消息并且保证它们的先后顺序不变。但是 尽管服务器保证了消息的顺序消息还是异步的发送给各个消费者消费者收到消息的先后顺序不能保证了。这也意味着并行消费将不能保证消息的先后顺序。用过传统的消息系统的同学肯定清楚消息的顺序处理很让人头痛。如果只让一个消费者处理消息又违背了并行处理的初衷。 在这一点上Kafka做的更好尽管并没有完全解决上述问题。 Kafka采用了一种分而治之的策略分区。 因为Topic分区中消息只能由消费者组中的唯一一个消费者处理所以消息肯定是按照先后顺序进行处理的。但是它也仅仅是保证Topic的一个分区顺序处理不能保证跨分区的消息先后处理顺序。 所以如果你想要顺序的处理Topic的所有消息那就只提供一个分区。

### Kafka的保证(Guarantees)

生产者发送到一个特定的Topic的分区上的消息将会按照它们发送的顺序依次加入

消费者收到的消息也是此顺序

如果一个Topic配置了复制因子( replication facto)为N 那么可以允许N-1服务器宕机而不丢失任何已经增加的消息

Kafka官网

http://kafka.apache.org/

作者半兽人  
链接http://orchome.com/5  
来源OrcHome  
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。

# ****2 安装和启动****

**1、下载二进制安装包直接解压**

|  |  |
| --- | --- |
| 1  2 | tar xf kafka\_2.11-0.10.0.1.tgz  cd kafka\_2.11-0.10.0.1 |

**2、启动服务**

Kafka需要用到ZooKeepr所以需要先启动一个ZooKeepr服务端如果没有单独的ZooKeeper服务端可以使用Kafka自带的脚本快速启动一个单节点ZooKeepr实例

|  |  |
| --- | --- |
| 1  2  3 | bin/zookeeper-server-start.sh config/zookeeper.properties  # 启动zookeeper服务端实例    bin/kafka-server-start.sh config/server.properties  # 启动kafka服务端实例 |

# ****3 基本操作指令****

**0、启动**

nohup bin/kafka-server-start.sh config/server.properties > /dev/null 2>&1 &

**1、新建一个主题topic**

创建一个名为“test”的Topic只有一个分区和一个备份

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

**2、创建好之后可以通过运行以下命令查看已创建的topic信息**

|  |  |
| --- | --- |
| 1 | bin/kafka-topics.sh --list  --zookeeper localhost:2181 |

**3、发送消息**

Kafka提供了一个命令行的工具可以从输入文件或者命令行中读取消息并发送给Kafka集群。每一行是一条消息。

运行producer生产者,然后在控制台输入几条消息到服务器。

|  |  |
| --- | --- |
| 1  2  3 | bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test  This is a message  This is another message |

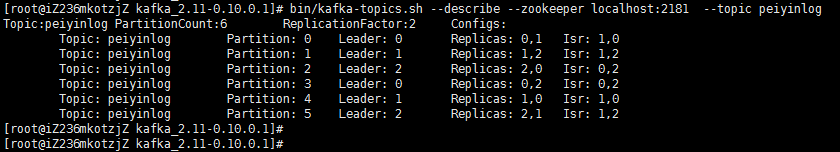
**4、消费消息**

Kafka也提供了一个消费消息的命令行工具，将存储的信息输出出来。

|  |  |
| --- | --- |
| 1  2  3 | bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning  This is a message  This is another message |

**5、查看topic详细情况**

|  |  |
| --- | --- |
| 1 | bin/kafka-topics.sh --describe --zookeeper localhost:2181  --topic peiyinlog |

[](http://s5.51cto.com/wyfs02/M00/87/CB/wKiom1fh5DiCDyFHAABNw6sr0ok754.png)

Topic: 主题名称

Partition: 分片编号

Leader: 该分区的leader节点

Replicas: 该副本存在于哪个broker节点

Isr: 活跃状态的broker

**6、给Topic添加分区**

|  |  |
| --- | --- |
| 1 | bin/kafka-topics.sh --zookeeper 192.168.90.201:2181 --alter --topic test2 --partitions 20 |

**7、删除Topic**

|  |  |
| --- | --- |
| 1 | bin/kafka-topics.sh --zookeeper zk\_host:port/chroot --delete --topic my\_topic\_name |

**8 kafka查看队列的消费情况**

kafka-run-class.sh kafka.tools.ConsumerOffsetChecker --broker-info --group $group --topic $topic --zookeeper $zk\_host:2181

主题（Topic）删除选项默认是关闭的，需要服务器配置开启它。

|  |  |
| --- | --- |
| 1 | delete.topic.enable=true |

**注：如果需要在其他节点作为客户端使用指令连接kafka broker，则需要注意以下两点（二选一即可）**

**另 : ( 使用logstash input 连接kafka也需要注意 )**

1、设置kafka broker 配置文件中 host.name 参数为监听的IP地址

2、给broker设置一个唯一的主机名，然后在本机/etc/hosts文件配置解析到自己的IP（当然如果有内网的DNS服务器也行），同时还需要在zk server 和 客户端的 /etc/hosts 添加broker主机名的解析。

**原因详解:**

**场景假设**

|  |  |  |  |
| --- | --- | --- | --- |
| broker\_server ip | 主机名 | zookeeper ip | 客户端 ip |
| 192.168.1.2 | 默认 localhost | 192.168.1.4 | 192.168.1.5 |

|  |  |
| --- | --- |
| 1  2  3 | # 此时客户端向broker发起一些消费:    bin/kafka-console-consumer.sh --zookeeper 192.168.1.4:2181 --topic test2 --from-beginning |

这时客户端连接到zookeeper要求消费数据，zk则返回broker的ip地址和端口给客户端，但是如果broker没有设置host.name 和 advertised.host.name  broker默认返回的是自己的主机名，默认就是localhost和端口9092，这时客户端拿到这个主机名解析到自己，操作失败。

所以，需要配置broker 的host.name参数为监听的IP，这时broker就会返回IP。 客户端就能正常连接了。

或者也可以设置好broker的主机名，然后分别给双方配置好解析。

# 4 broker基本配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | #  server.properties    broker.id=0  # broker节点的唯一标识 ID 不能重复。  host.name=10.10.4.1  # 监听的地址，如果不设置默认返回主机名给zk\_server  log.dirs=/u01/kafka/kafka\_2.11-0.10.0.1/data  # 消息数据存放路径  num.partitions=6  # 默认主题（Topic）分片数  log.retention.hours=24  # 消息数据的最大保留时长  zookeeper.connect=10.160.4.225:2181  # zookeeper server 连接地址和端口 |

# ****5 Logstash + Kafka 实战应用****

Logstash-1.51才开始内置Kafka插件，也就是说用之前的logstash版本是需要手动编译Kafka插件的，相信也很少人用了。建议使用2.3以上的logstash版本。

**1、使用logstash向kafka写入一些数据**

软件版本:

logstash 2.3.2

kafka\_2.11-0.10.0.1

logstash output 部分配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | output {    kafka {      workers => 2      bootstrap\_servers => "10.160.4.25:9092,10.160.4.26:9092,10.160.4.27:9092"      topic\_id => "xuexilog"    }    } |

参数解释 :

workers：用于写入时的工作线程

bootstrap\_servers：指定可用的kafka broker实例列表

topic\_id：指定topic名称，可以在写入前手动在broker创建定义好分片数和副本数，也可以不提前创建，那么在logstash写入时会自动创建topic，分片数和副本数则默认为broker配置文件中设置的。

**2、使用logstash消费一些数据，并写入到elasticsearch**

软件版本:

logstash 2.3.2

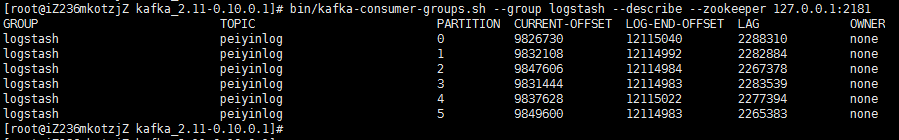
elasticsearch-2.3.4

logstash 配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  2 | input{      kafka {          zk\_connect => "112.100.6.1:2181,112.100.6.2:2181,112.100.6.3:2181"          group\_id => "logstash"          topic\_id => "xuexilog"          reset\_beginning => false          consumer\_threads => 5          decorate\_events => true    }    }    # 这里group\_id 需要解释一下，在Kafka中，相同group的Consumer可以同时消费一个topic，不同group的Consumer工作则互不干扰。  # 补充: 在同一个topic中的同一个partition同时只能由一个Consumer消费，当同一个topic同时需要有多个Consumer消费时，则可以创建更多的partition。    output {  **if** [type] == "nginxacclog" {          elasticsearch {              hosts => ["10.10.1.90:9200"]              index => "logstash-nginxacclog-%{+YYYY.MM.dd}"              manage\_template => true              flush\_size => 50000              idle\_flush\_time => 10              workers => 2  }    }    } |

**3、通过group\_id 查看当前详细的消费情况**

|  |  |
| --- | --- |
| 1 | bin/kafka-consumer-groups.sh --group logstash --describe --zookeeper 127.0.0.1:2181 |

[](http://s3.51cto.com/wyfs02/M00/87/D4/wKiom1fiTL-xhYo5AABDZmsbids038.png)

输出解释:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GROUP | TOPIC | PARTITION | CURRENT-OFFSET | LOG-END-OFFSET | LAG |
| 消费者组 | 话题id | 分区id | 当前已消费的条数 | 总条数 | 未消费的条数 |