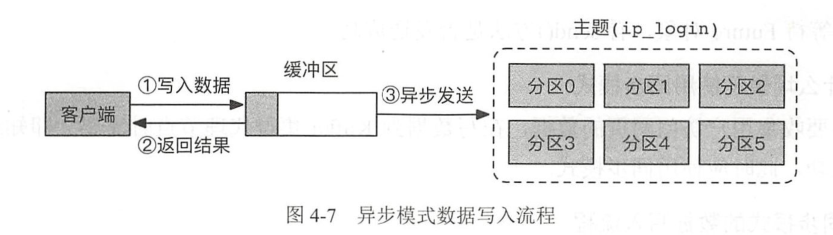
### Kafka生产者

#### 发送消息到kafka主题---异步模式

Kafka0.10.00后客户端调用send方法默认采用异步模式[send(topic,message,callback)],消息发送到kafka代理节点的响应后会触发回调函数。

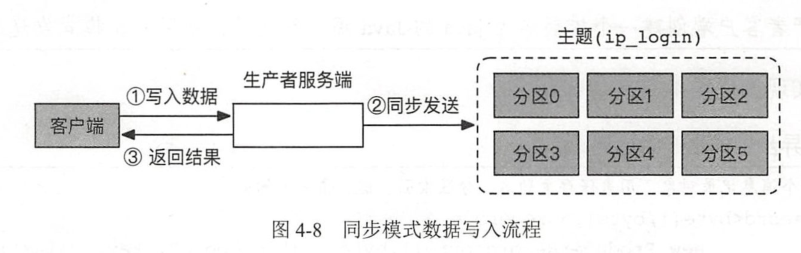
例如一个业务有6个分区，生产者客户端写入一条消息记录后，消息记录会先写入某个缓冲区，生产者直接得到结果(这时，缓冲区的数据并没有写到kafka代理节点中主题的某个分区)。之后，缓冲区中的数据会通过异步模式发送到kafka代理节点中主题的某个分区中。



同步模式---实例代码

|  |
| --- |
| *kafkaProducer*.send(record, new Callback() {  @Override  public void onCompletion(RecordMetadata recordMetadata, Exception e) {  if(null != e){  System.*out*.println(e);  }  if(null != recordMetadata){  System.*out*.println(String.*format*("offset:%s,partition:%s", recordMetadata.offset() , recordMetadata.partition()));  }  } }); |

#### 发送消息到kafka主题---同步模式

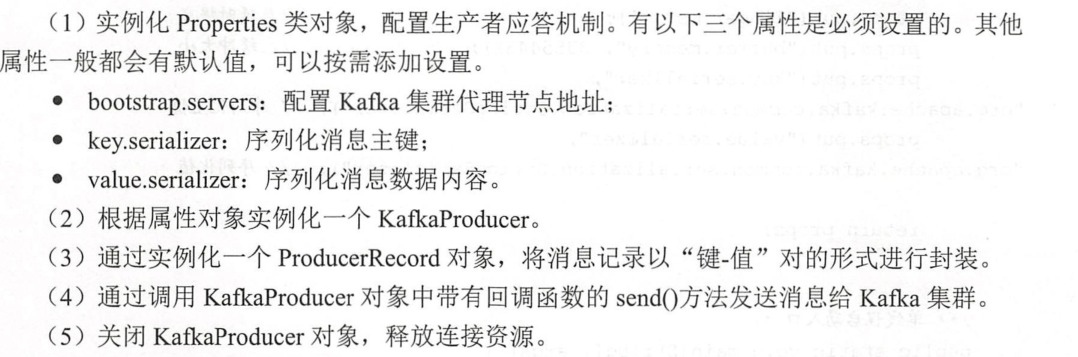


同步模式---代码实例

|  |
| --- |
| Future<RecordMetadata> future = *kafkaProducer*.send(record); //返回一个future，并调用get等待发送成功返回 future.get(); |

#### 发送消息到kafka主题---多线程发送

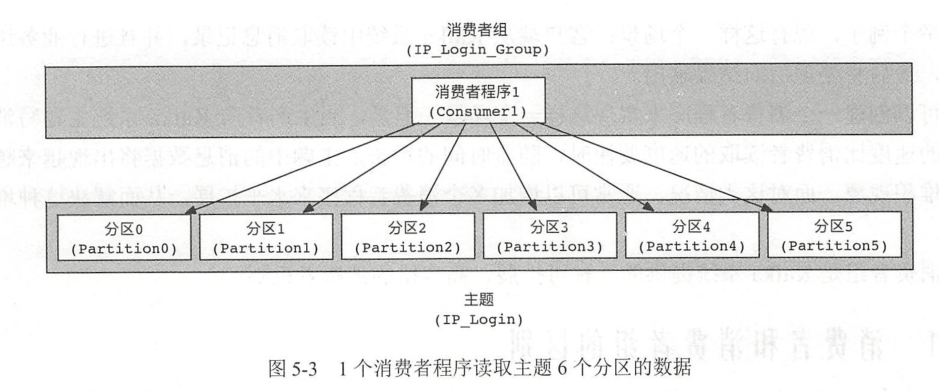
发送消息流程如下:

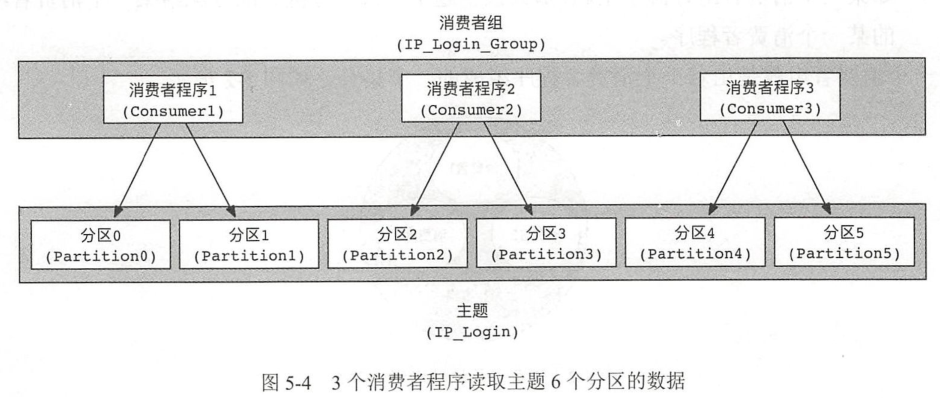


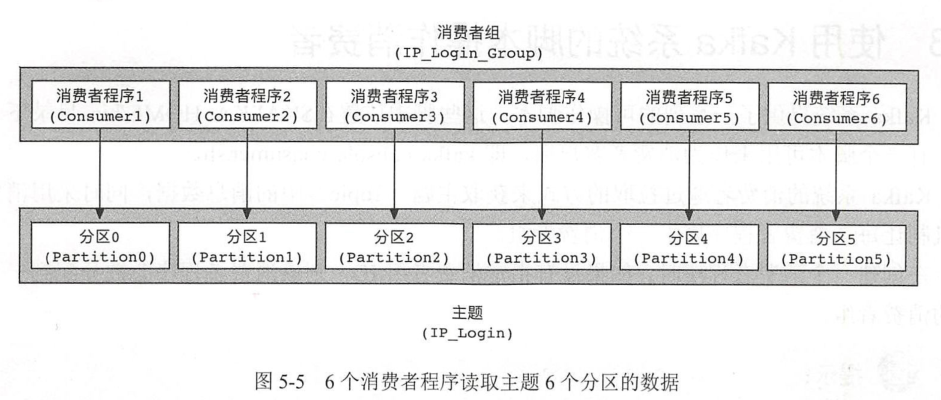
kafkaProducer是线程安全的，所以可以多并发的发送消息

### Kakfa消费者

#### 消费者组消费分区



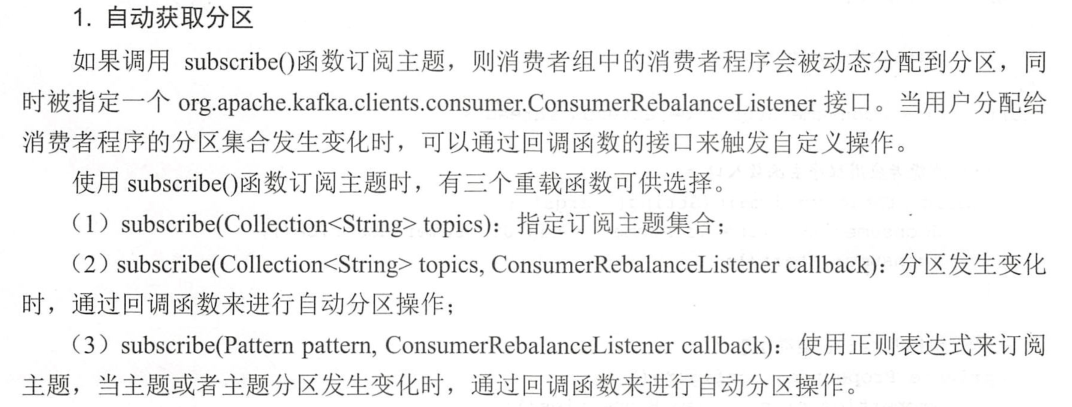




消费者程序的数量尽量不要超过主题的最大分区数，因为，多出来的消费者程序是空闲的，不仅没有任何帮助，反而浪费系统资源

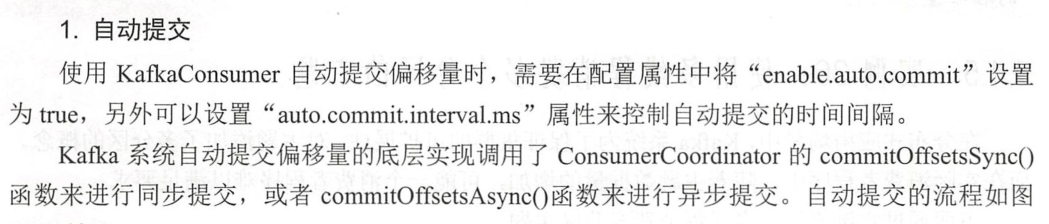
KafkaConsumer是非线程安全的，如果多个线程公用一个KafkaConsumer实例则会抛出异常错误信息

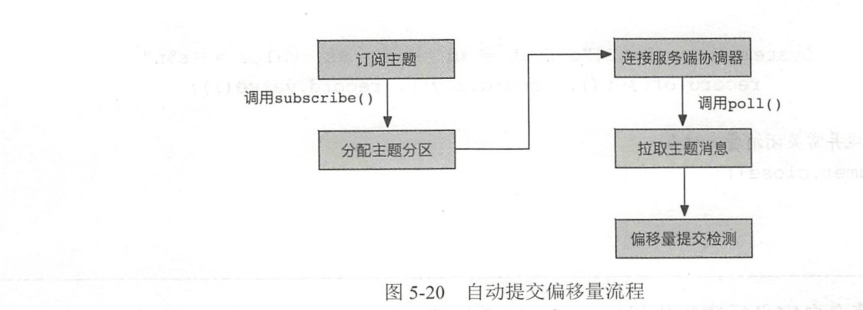
#### 消费者消费--自动获取分区消费



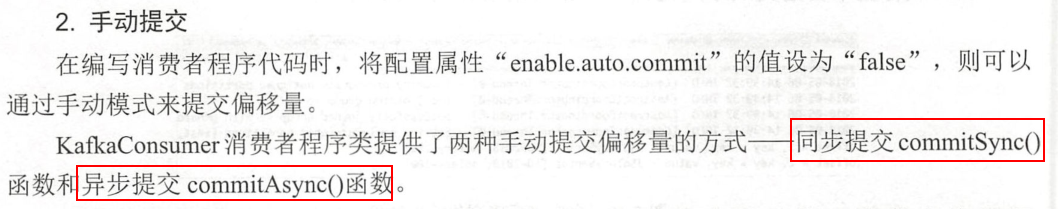
#### 如何提交消息的偏移量

自动提交偏移量





手动提交偏移量



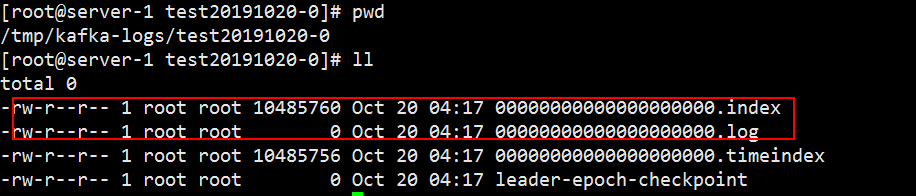
同步提交和异步提交的区别在于:同步提交需要等待相应结果,会造成阻塞现象;异步提交不会阻塞

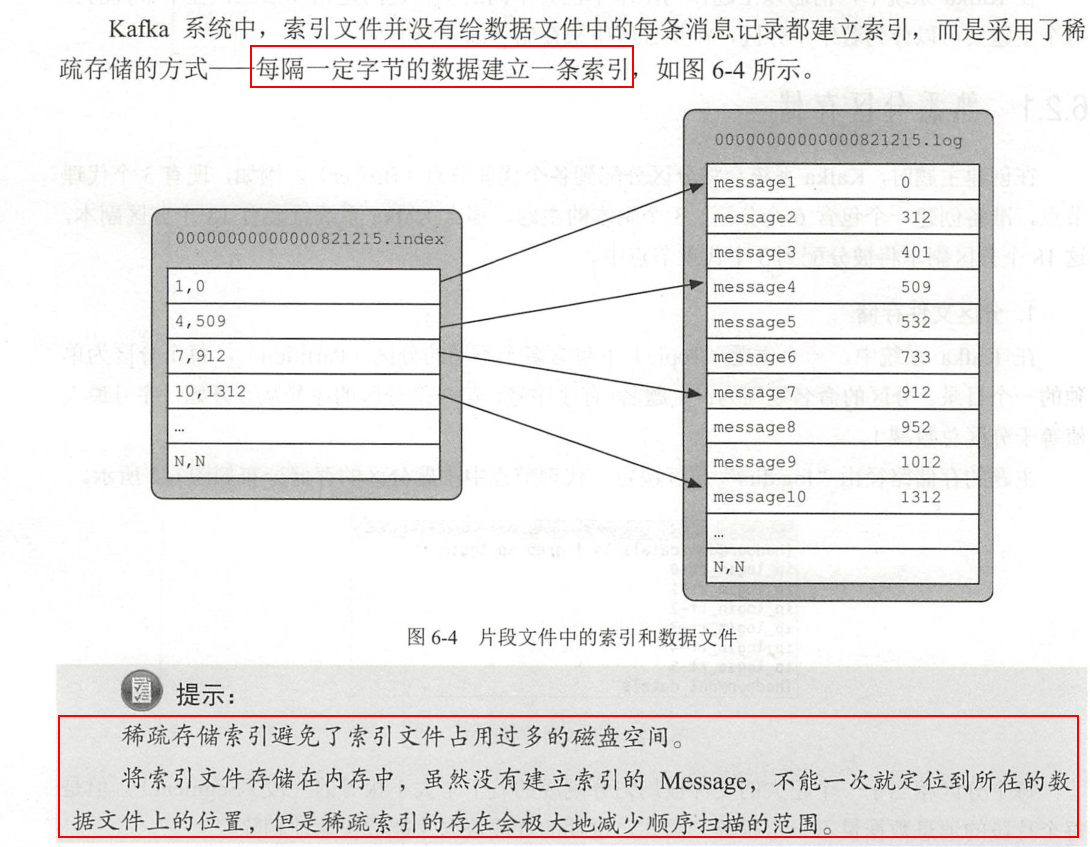
### Kafka消息存储

#### Kafka消息存放路径和结构

Kafka消息存放的路径是由server.properties配置文件里面的log.dirs配置决定的

Kafka的消息以段的形式存储，如图





#### Kafka消息的清理

Kafka的消息是会被清理了，主要是两种方式:基于时间和大小的删除策略和压缩清理策略

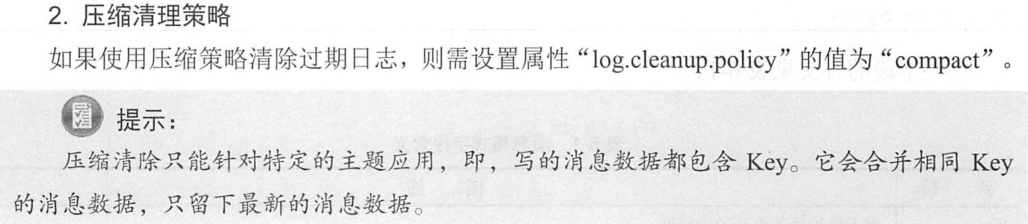
##### 基于时间和大小的删除策略

这种方式主要通过配置

log.retention.hours=168 #系统在168个小时后删除消息

log.retention.bytes=-1 #消息大于多少的时候删除

##### 压缩清理策略



### Kafka连接器

#### 连接器简介

Kafka的连接器的作用主要是跟其他系统对接，例如将mysql的数据收集到kafka主题中





主要分成两种

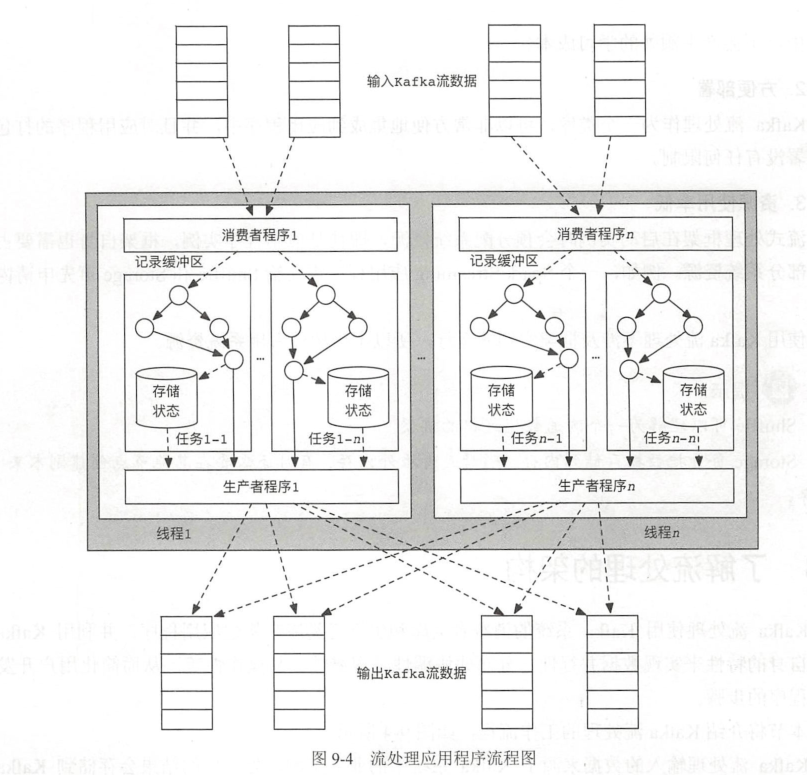
1. Source连接器：负责将数据导入到Kafka系统
2. Sink连接器：负责将数据从Kafka系统中导出

#### 实例:单机模式下，将数据导入到Kafka主题中

先建一个主题

### Kafka流处理

#### 流程图

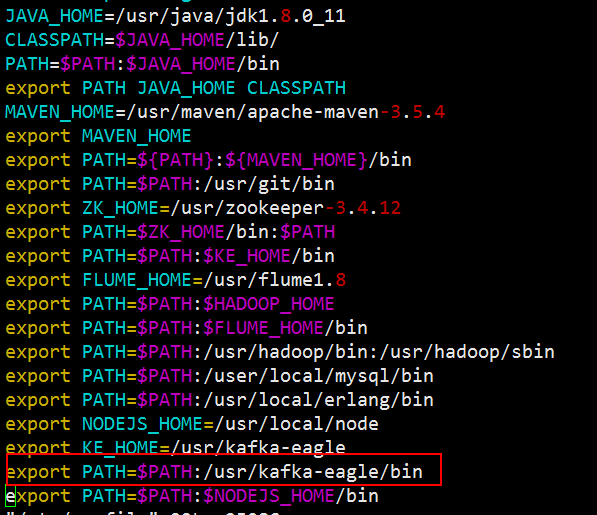


### Kafka eagle监控工具搭建

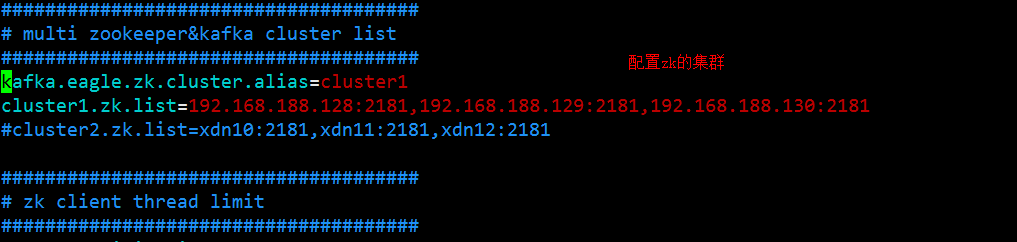
#### 下载

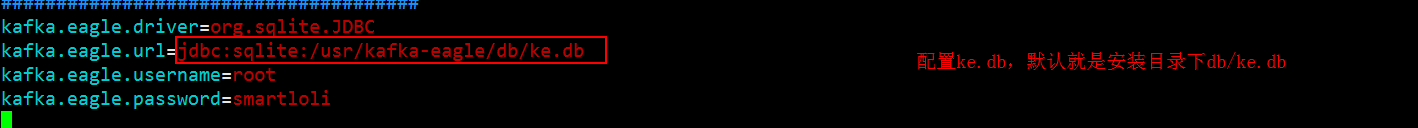
<http://download.kafka-eagle.org/>

#### 配置 /etc/profile



#### 配置kafka eagle配置文件system-config.properties





#### 启动

sh ke.sh start