Kafka，主讲：汤小洋

一、消息系统

1. 消息队列

消息Message：网络中的两台计算机或者两个通讯设备之间传递的数据，如：文本、音频、视频等。

队列Queue：一种特殊的线性表，特殊之处在于它只允许在首部删除元素（队首），在尾部添加元素（队尾）。

消息队列MQ：保存消息的队列，是消息在传输过程中的容器。主要提供生产和消费接口供外部调用，进行数据的存储和获取。

2. MQ分类

主要分为两类：点对点（Peer-to-Peer）、发布/订阅（Publish/Subscribe）

共同点：消息生产者(Producer)生产消息发送到队列中，然后消息消费者(Consumer)从队列中读取并消费消息。

不同点：

- 点对点

  组成：消息队列(Queue)、发送者(Sender)、接收者(Receiver)

  一个生产者生产的消息只能有一个消费者，消息一旦被消费，消息就不在消息队列中了，如：打电话

  即发送到消息队列的消息能且只能被一个接收者接收

- 发布/订阅

  组成：消息队列(Queue)、发布者(Publisher)、订阅者(Subscriber)、主题(Topic)

  每个消息可以有多个消费者，彼此互不影响，如：我发布一个微博，关注我的人都能看到

  即发布到消息队列的消息能被多个接收者（订阅者）接收

3. 常见消息系统

- ActiveMQ：历史悠久，实现了JMS（Java Message Service）规范，支持性较好，性能相对不高
- RabbitMQ：可靠性高、安全
- Kafka：分布式、高性能、跨语言
- RocketMQ：阿里开源的消息中间件，纯Java实现

二、Kafka简介

1. 介绍

Kafka是一个分布式的发布/订阅消息系统，最初由LinkedIn(领英)公司发布，使用Scala语言编写，后成为Apache的顶级项目。

主要用于处理活跃的数据，如登录、浏览、点击、分享、喜欢等用户行为产生的数据。

特点：

- 高吞吐量

  可以满足每秒百万级别消息的生产和消费。

- 持久性

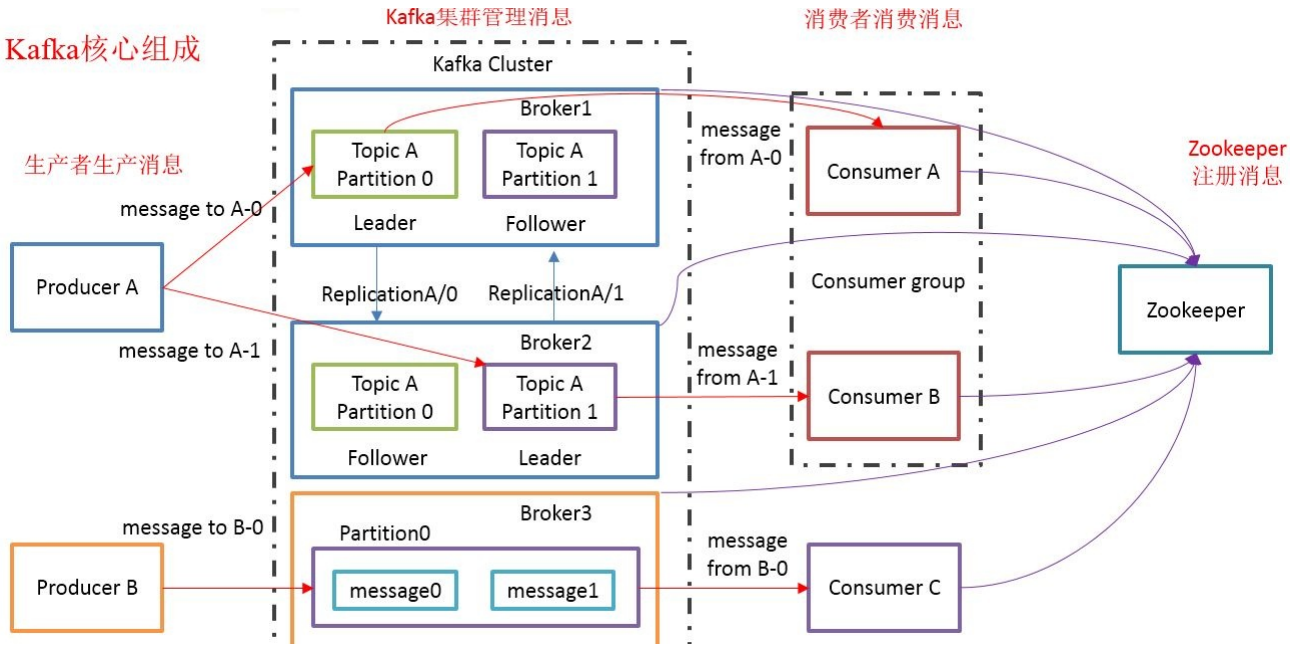
  有一套完善的消息存储机制，确保数据的高效安全的持久化。

- 分布式

  基于分布式的扩展和容错机制；

  Kafka的数据会复制到多台服务器上，当某一台发生故障失效时，生产者和消费者转而使用其它的机器。

2. 架构



3. 组成

Broker：kafka集群中包含多个kafka服务节点，每一个kafka服务节点就称为一个broker

Topic：主题，用来存储不同类别的消息（Kafka消息数据是存储在硬盘上的）

Partition：分区，每个Topic包含一个或多个Partition，在创建Topic时指定包含的Partition数量（目的是为了进行分布式存储）

Replication：副本，每个分区可以有多个副本，分布在不同的Broker上，会选出一个副本作为Leader，所有的读写请求都会通过Leader完成，Follower只负责备份数据

所有Follower会自动的从Leader中复制数据，当Leader宕机后， 会从Follower中选出一个新的Leader继续提供服务，实现故障自动转移

Message：消息，是通信的基本单位，每个消息都属于一个Partition

Producer：消息的生产者，向Kafka的一个topic发布消息

Consumer：消息的消费者，订阅topic并读取其发布的消息

Consumer Group：每个Consumer属于一个特定的Consumer Group，多个Consumer可以属于同一个Consumer Group中

Zookeeper：协调kafka的正常运行，Kafka将元数据信息保存在Zookeeper中，但发送给Topic本身的消息数据并不存储在ZK中，而在存储在磁盘文件中

三、Kafka安装

1. 安装

步骤：

1. 解压kafka_2.12-2.3.0.tgz

```
cd ~/software
tar -zxf kafka_2.12-2.3.0.tgz
```

2. 配置

```
# 创建存放数据的文件夹
cd kafka_2.12-2.3.0
mkdir data
# 修改kafka配置文件
cd config
vi server.properties
    #listeners=PLAINTEXT://:9092  # kafka默认监听的端口号为9092
    log.dirs=../data  # 指定数据的存放目录
    zookeeper.connect=localhost:2181  # zookeeper的连接信息
```

3. 启动zookeeper

- 使用kafka内置zk

```
cd ~/software/kafka_2.12-2.3.0/bin/
./zookeeper-server-start.sh ../config/zookeeper.properties
```

- 使用外部zk（推荐）

```
cd ~/software/zookeeper-3.4.13/bin/
./zkServer.sh start
```

4. 启动kafka

```
./kafka-server-start.sh ../config/server.properties &  # &表示后台运行，也可使用-daemon选项
jps  # 查看进程，jps是jdk提供的用来查看所有java进程的命令
```

5. 创建Topic（主题）

```
./kafka-topics.sh \
    --create \
    --zookeeper localhost:2181 \
    --replication-factor 1 \
    --partitions 3 \
    --topic hello
```

```
# 查看Topic列表
./kafka-topics.sh --list --zookeeper localhost:2181  # __consumer_offsets是kafka的内部Topic
# 查看某一个具体的Topic
./kafka-topics.sh --describe --zookeeper localhost:2181 --topic hello
# 修改Topic：只能增加partition个数，不能减少，且不能修改replication-factor
./kafka-topics.sh --alter --zookeeper localhost:2181 --topic hello --partitions 5
# 删除Topic （需要启用topic删除功能）
./kafka-topics.sh --delete --zookeeper localhost:2181 --topic hello
```

6. 启动kafka的Producer（生产者 ）

```
./kafka-console-producer.sh --broker-list localhost:9092 --topic hello
```

7. 启动kafka的Consumer（消费者）

```
./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic hello --from-beginning
```

8. 验证

查看data数据存放目录：Topic的每个Partition对应一个目录，数据存储在目录下的00000000000000000000.log文件中

查看zookeeper中的内容：get /brokers/topics/hello/partitions/0/state

2. 配置文件

```
############################# Server Basics #############################
# broker的id，值为整数，且必须唯一，在一个集群中不能重复
broker.id=0

############################# Socket Server Settings #############################
# kafka默认监听的端口为9092
#listeners=PLAINTEXT://:9092
# 处理网络请求的线程数量，默认为3个
num.network.threads=3
# 执行磁盘IO操作的线程数量，默认为8个
num.io.threads=8
# socket服务发送数据的缓冲区大小，默认100KB
socket.send.buffer.bytes=102400
# socket服务接受数据的缓冲区大小，默认100KB
socket.receive.buffer.bytes=102400
# socket服务所能接受的一个请求的最大大小，默认为100M
socket.request.max.bytes=104857600

############################# Log Basics #############################
# kafka存储消息数据的目录
log.dirs=../data
# 每个topic默认的partition数量
num.partitions=1
# 在启动时恢复数据和关闭时刷新数据时每个数据目录的线程数量
num.recovery.threads.per.data.dir=1

############################# Log Flush Policy #############################
# 消息刷新到磁盘中的消息条数阈值
#log.flush.interval.messages=10000
# 消息刷新到磁盘中的最大时间间隔
#log.flush.interval.ms=1000

############################# Log Retention Policy #############################
# 日志保留小时数，超时会自动删除，默认为7天
log.retention.hours=168
# 日志保留大小，超出大小会自动删除，默认为1G
#log.retention.bytes=1073741824
# 日志分片策略，单个日志文件的大小最大为1G，超出后则创建一个新的日志文件
log.segment.bytes=1073741824
# 每隔多长时间检测数据是否达到删除条件
log.retention.check.interval.ms=300000

############################# Zookeeper #############################
# Zookeeper连接信息，如果是zookeeper集群，则以逗号隔开
zookeeper.connect=localhost:2181
# 连接zookeeper的超时时间
zookeeper.connection.timeout.ms=6000
```

```
# 是否可以删除topic，默认为false
delete.topic.enable=true
```

四、Kafka集群搭建

1. 搭建zk集群

可以在一台主机上启动多个zk服务，配置使用不同的端口即可

步骤：

1. 拷贝多个zk目录

   zookeeper1、zookeeper2、zookeeper3

2. 分别配置每个zk

```
vi zookeeper1/conf/zoo.cfg
    clientPort=2181
  server.1=192.168.2.153:6661:7771
  server.2=192.168.2.153:6662:7772
  server.3=192.168.2.153:6663:7773
echo 1 > zookeeper1/data/myid

vi zookeeper2/conf/zoo.cfg
    clientPort=2182
  server.1=192.168.2.153:6661:7771
  server.2=192.168.2.153:6662:7772
  server.3=192.168.2.153:6663:7773
echo 2 > zookeeper2/data/myid

vi zookeeper3/conf/zoo.cfg
    clientPort=2183
  server.1=192.168.2.153:6661:7771
  server.2=192.168.2.153:6662:7772
  server.3=192.168.2.153:6663:7773
echo 3 > zookeeper3/data/myid
```

3. 启动zk集群

2. 搭建Kafka集群

步骤

1. 拷贝多个kafka目录

   kafka1、kafka2、kafka3

2. 分别配置每个kafka

```
vi kafka1/config/server.properties
    broker.id=1
    listeners=PLAINTEXT://192.168.2.153:9091
    zookeeper.connect=192.168.2.153:2181,192.168.2.153:2182,192.168.2.153:2183

vi kafka2/config/server.properties
    broker.id=2
    listeners=PLAINTEXT://192.168.2.153:9092
    zookeeper.connect=192.168.2.153:2181,192.168.2.153:2182,192.168.2.153:2183

vi kafka3/config/server.properties
    broker.id=3
    listeners=PLAINTEXT://192.168.2.153:9093
    zookeeper.connect=192.168.2.153:2181,192.168.2.153:2182,192.168.2.153:2183
```

3. 启动kafka集群
4. 创建Topic

```
./kafka-topics.sh \
    --create \
    --zookeeper 192.168.7.40:2181,192.168.7.40:2182,192.168.7.40:2183 \
    --replication-factor 3 \
    --partitions 5 \
    --topic aaa
```

5. 生成数据/发布消息

```
./kafka-console-producer.sh --broker-list
192.168.7.40:9091,192.168.7.40:9092,192.168.7.40:9093 --topic aaa
```

6. 消费数据/订阅消息

```
./kafka-console-consumer.sh --bootstrap-server
192.168.7.40:9091,192.168.7.40:9092,192.168.7.40:9093 --topic aaa --from-
beginning
```

五、SpringBoot集成Kafka

1. 简介

SpringBoot提供了一个名为 `spring-kafka` 的starter，用于在Spring项目里快速集成kafka

2. 用法

步骤：

1. 创建SpringBoot项目

   勾选Spring Web Starter和Spring for Apache Kafka

2. 配置kafka，编辑application.yml文件

```yaml
spring:
  kafka:
    # kafka服务器地址(可以多个)
    bootstrap-servers: 192.168.7.40:9091,192.168.7.40:9092,192.168.7.40:9093
    producer:
        # 每次批量发送消息的数量
      batch-size: 65536
      buffer-memory: 524288
      # key/value的序列化
      key-serializer: org.apache.kafka.common.serialization.StringSerializer
      value-serializer: org.apache.kafka.common.serialization.StringSerializer
    consumer:
      # 指定一个默认的组名
      group-id: test
      # key/value的反序列化
      key-deserializer:
org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer:
org.apache.kafka.common.serialization.StringDeserializer
```

3. 创建生产者

```java
@RestController
public class KafkaProducer {

    @Autowired
    private KafkaTemplate template;

    /**
     * 发送消息到Kafka
     * @param topic   主题
     * @param message 消息
     */
    @RequestMapping("/sendMsg")
    public String sendMsg(String topic, String message) {
        template.send(topic, message);
        return "success";
    }
}
```

4. 创建消费者

```
@Component
public class KafkaConsumer {

    /**
     * 订阅指定主题的消息
     * @param record 消息记录
     */
    @KafkaListener(topics = {"hello","world"})
    public void listen(ConsumerRecord record) {
        // System.out.println(record);
        System.out.println(record.topic()+","+record.value());
    }

}
```

5. 测试

   访问http://localhost:8080/sendMsg?topic=hello&message=aaaa