# 1.消息系统概述

# 一、模式

1.Peer-to-Peer模式 - 点对点

（1）往往消费者是串行消费的

（2）数据往往只能被消费一次

（3）解耦能力相对较弱

2发布/订阅模式

（1）允许消费者进行并行消费

（2）往往会提供数据的持久化机制，从而保证数据可以被消费多次

（3）解耦能力更强

# 二、消息队列的优势

1.解耦：各个系统之间通过消息系统这个统一接口交换数据，无须了解彼此的存在

2.数据的可靠性：部分消息系统拥有持久化的能力，可以规避消息处理前丢失的风险

3.消峰限流：消息系统可顶住峰值流量，业务系统可根据处理能力从消息系统中获取数据

4.异步通信：在不需要立即处理请求的场景下，可以将请求放入消息系统在需要的时候再取出处理

# 2.简介

1.Kafka是由LinkedIn(领英)开发的消息发布系统，后来贡献给了Apache

2.Kafka是发布订阅模式的消息队列

3.Kafka的特征

（1）发布和订阅流式数据

（2）提供了容错机制来存储流式数据

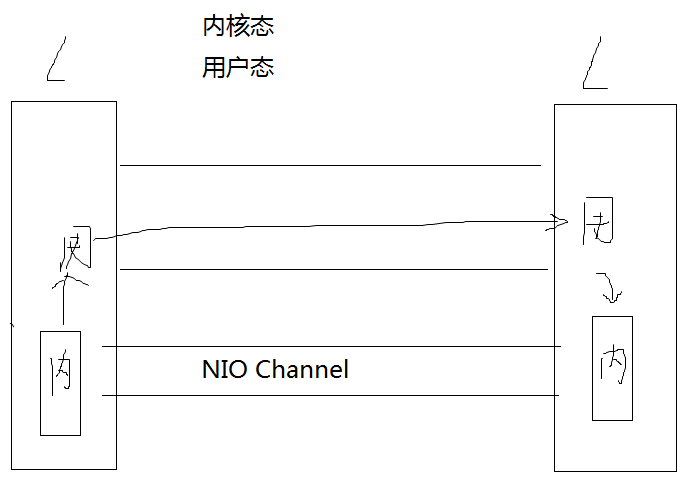
（3）在流式数据出现的时候能够实时处理

4.应用场景：

（1）能够在系统或者应用之间构建一个可靠的实时流传输管道

（2）能够构建一个用于流式数据的传输和转化的实时流应用

5.实际开发中，KafKa节点个数一般是3-5个。Kafka之所以能实现大量数据的传输，是因为底层采用了Scala语言开发，并且引入了零拷贝技术



# 3.特点

1.高吞吐量

Kafka 每秒可以生产约 100 万消息（1000 MB），每秒处理 200 万消息（2000 MB）

2.持久化数据存储

可进行持久化操作。将消息持久化到磁盘，因此可用于批量消费，例如 ETL，以及实时应用程序。通过将数据持久化到硬盘以及 replication 防止数据丢失。

3.分布式系统易于扩展

所有的 producer、broker（kafka节点） 和 consumer 都会有多个，均为分布式的。无需停机即可扩展机器。

4.客户端状态维护

消息被处理的状态是在 consumer 端维护，而不是由 server 端维护。减轻服务器端的压力，为客户端会话管理提供了更好的灵活性。

# 4.Kafka详解

1.Producer：生产者。产生数据放入Kafka

2.Consumer：消费者。从Kafka中拿取数据

3.broker：经纪人。在Kafka集群中，每一个Kafka节点都称之为是一个broker

4.topic：主题/话题。

（1）在Kafka中，要求每一条数据必须放在某个主题中

（2）主题在Kafka中的作用是用于进行数据的分类

（3）所谓的主题可以认为是在Kafka中维系了一个子队列

（4）每一个主题可以对应1个到多个分区

（5）主题在执行删除命令之后，不会立即删除而是被标记为删除状态，等30s才会被删除。如果需要立即删除，那么修改server.properties中的delete.topic.enable，将这一项的值改为true

5.partition：分区

（1）为了做到数据的分布式存储，设计了分区

（2）如果设置了多个分区，那么每一个分区都会对应一个目录

（3）实际过程中，如果配置了多个Kafka节点，那么分区会平均分配到每一个节点上

6.replication：副本。对数据进行备份，保证数据不会丢失

# 5.Kafka安装配置

**伪分布式：**

1.下载Kafka安装包

<http://kafka.apache.org>

2.上传Linux并解压

tar -zxvf kafka\_2.11-1.0.0.tgz

3.修改server.properties文件

配置Kafka存储位置，要配置为非/tmp目录

log.dirs=非/tmp目录

4.修改zookeeper.properties文件

Kafka内置了一个zookeeper，在伪分布式模式下，可以使用这个内置zk作为集群协调工具，但是这个内置的zookeeper只有一个节点，不能用在生产环境下。如果使用这个内置的zk，需要在zookeeper.properties文件中修改zk存储路径，改为非/tmp目录

dataDir=非/tmp目录

5.启动kafka

（1）启动zookeeper：

bin/zookeeper-server-start.sh config/zookeeper.properties &

（2）启动kafka：

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

**完全分布式：（搭建）**

1.下载Kafka安装包

<http://kafka.apache.org>

2.上传Linux并解压

tar -zxvf kafka\_2.11-1.0.0.tgz

3.修改server.properties文件（config目录下）

（1）配置当前broker的编号，每个节点要求唯一

broker.id=0 #当前server编号，21行

（2）配置Kafka日志存储位置，要配置为非/tmp目录

log.dirs=/home/software/kafka\_2.11-1.0.0/kafka-logs #非/tmp目录 60行

（3）配置zookeeper集群的地址

zookeeper.connect=hadoop01:2181,hadoop02:2181,hadoop03:2181 #123行

5.将配置完成的kafka复制到其它broker中，并修改broker的编号

scp -r kafka\_2.11-1.0.0/ root@hadoop02:/home/software/

6.启动kafka

（1）启动zk集群

sh zkServer.sh start#在每个zk服务器中都要执行

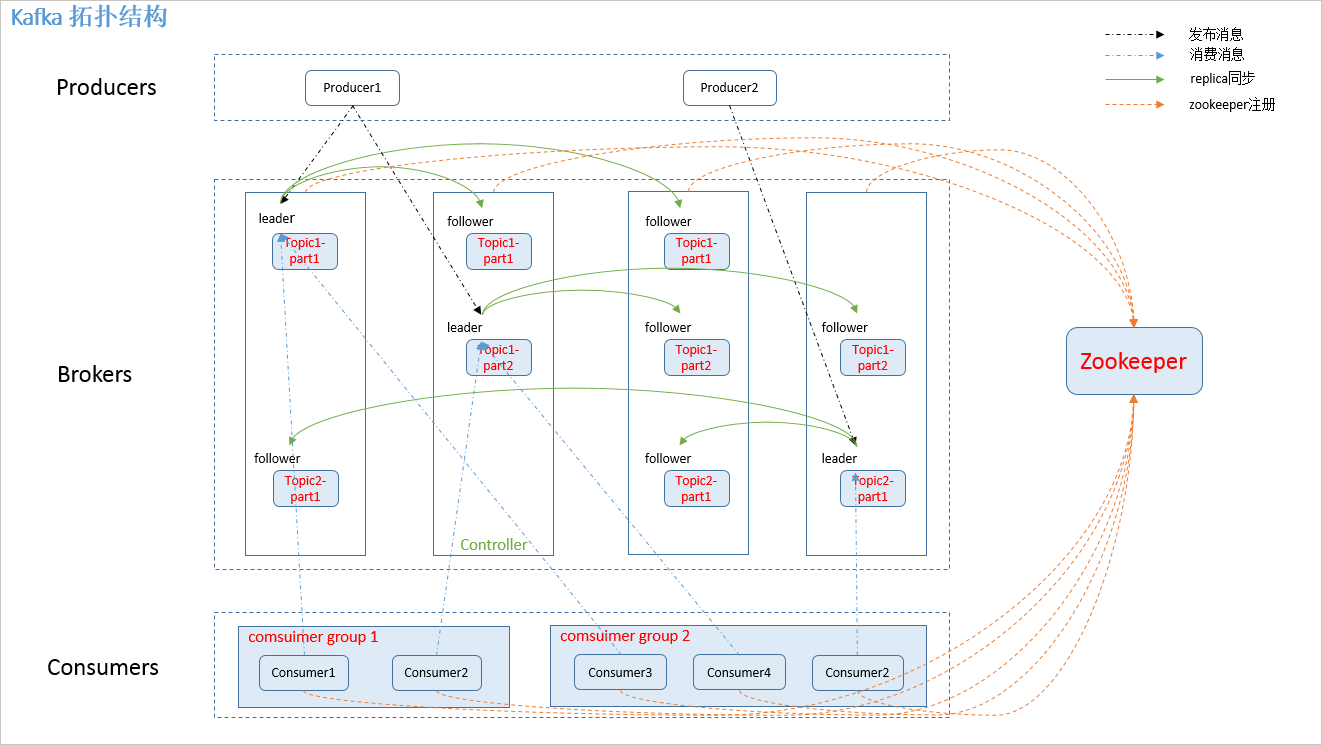
（2）启动kafka集群,进入Kafka安装目录的bin目录下

sh kafka-server-start.sh ../config/server.properties #在每个broker中都要执行

# 6.基本指令

|  |  |
| --- | --- |
| 命令 | 作用 |
| sh kafka-topics.sh --create --zookeeper hadoop01:2181,hadoop02:2181,hadoop03:2181 --replication-factor 1 --partitions 1 --topic englishbook | 创建主题  在创建的时候，副本数量要小于等于节点数量 |
| sh kafka-topics.sh --list --zookeeper hadoop01:2181 | 查看所有的topic |
| sh kafka-console-producer.sh --broker-list hadoop01:9092 --topic englishbook | 启动生产者 |
| sh kafka-console-consumer.sh --zookeeper hadoop01:2181 --topic englishbook --from-beginning | 启动消费者 |
| sh kafka-topics.sh --delete --zookeeper hadoop01:2181 --topic englishbook | 删除topic |
| sh kafka-topics.sh --describe --zookeeper hadoop01:2181 --topic englishbook | 描述topic的信息 |

# 7.Kafka架构



1.leader与follower：

（1）Kafka中leader和follower指的并不是节点，而是副本

（2）在Kafka中可以设置多个副本，当出现多个副本的时候，这些副本之间会选择一个副本成为leader副本，其他副本成为follower副本

（3）producer和consumer在和Kafka进行交互的时候，只和leader副本来交换数据

（4）leader副本和follower来进行实时的更新，并且更新之后，follower副本会给leader副本返回ack信号。同时leader副本还会保存一个返回信号的节点队列。这个队列准确的说是维系在了Zookeeper节点/isr\_change\_notification上。如果leader副本挂掉，那么优先从队列中来选leader，这个过程称之为ISR

（5）只要有副本存活，那么Kafka就可以继续对外服务

（6）如果所有副本都挂掉，那么先复活的副本就会成为leader

2.controller：控制器，监控leader副本的状态。如果leader副本挂掉，那么controller负责选择一个新的leader副本

3.Consumer Group：消费者组。将一个或者多个消费者绑定在一块形成一个消费者组。消息在**组间共享**（即不同的消费者组可以获取相同的消息）在**组内竞争**（即消息只能被这个消费者组中的某一个消费者消费不能被这个组内的多个消费者进行消费）

# 8.Kafka细节与机制

1.数据放入Kafka中之后，即使被多次消费，也不会从队列中移除

2.Kafka会将数据以日志形式记录到本地磁盘上

3.数据清理条件：

（1）默认情况下，每隔7天清理一次

（2）如果单个日志文件达到1G的时候，也会清理一次（一般不启用）

4.leader和follower之间进行备份的时候，是follower主动到leader来拉取数据

5.offset机制：

（1）偏移量的作用是用于记录消费者组上一次的消费位置防止数据的重复消费

（2）Kafka在启动之后，会在log.dirs目录下自动生成50个子目录用于存储消费者组的偏移量（用户组哈希取余）

6.索引机制：

（1）在分区目录下会存在index文件和log文件。每一个log文件都会对应一个index文件。一个log+一个index=一个segment

（2）数据是记录到log文件中，index是针对log的索引

（3）在Kafka中，建立的是稀疏索引 - 并不是针对每条数据都建立索引而是隔几条建立一个索引

（4）在读取索引文件的时候，也不是从头读取，而是采用了二分查找

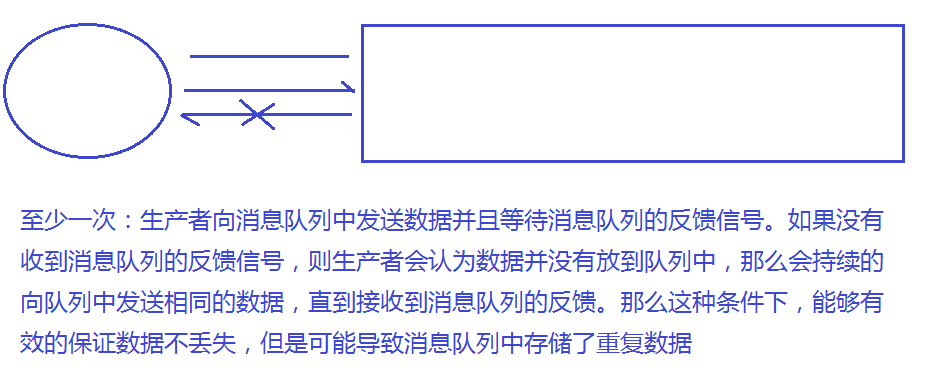
（5）当log文件达到1G的时候，会自动生成一个新的log文件，同时生成一个新的index文件。如果生成了新的log文件和index文件，则文件名是这个log文件所记录数据的起始位置

（6）timeindex为时间戳，记录log文件生成的时间，可以根据此时间计算log的清理时间

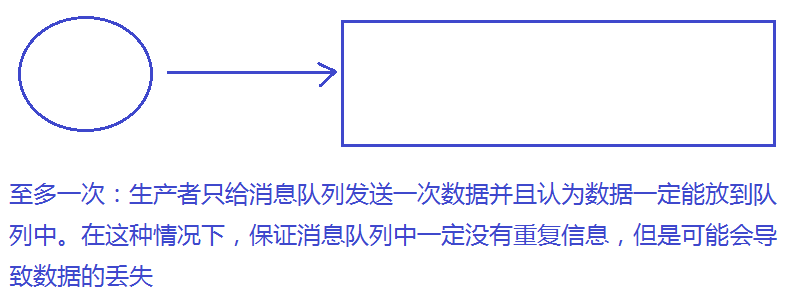
（7）leader-epoch-checkpoint：记录follower从leader中上次拿取的数据（热备）的位置，下次从此位置开始拿

# 三、语义

1.至少一次：



2.至多一次：



3.精确一次：

