各种消息队列对比

解耦、异步、削峰

各种消息中间件

kafka教程（scala编写）

kafka学习资料

kafka基础介绍简书

B站旧版视频

B站2019版视频

github基础代码、Springboot整合案例

具体实战参考busmessage工程是配置、注解、磁盘日志---kafka-agent是使用案例

pulsar替代kafka

http://pulsar.apache.org/docs/en/client-libraries-java/ https://www.jianshu.com/p/50e8f1276999 理由1topic支持不多2，扩展性不强

学习提纲

- 集群搭建、架构理解
- 工作流程
- 各种概念 分区生产消费
- 高级API低级API 新旧对比
- 拦截器
- Kafka Stream流式处理

1、应用场景：

天然分布式，顺序写磁盘、零拷贝，文件系统存储以及页缓存

> 日志收集：一个公司可以用Kafka可以收集各种服务的log，通过kafka以统一接口服务的方式开放给各种consumer，例如hadoop、Hbase、Solr等。
> 消息系统：解耦和生产者和消费者、缓存消息等。
> 用户活动跟踪：Kafka经常被用来记录web用户或者app用户的各种活动，如浏览网页、搜索、点击等活动，这些活动信息被各个服务器发布到kafka的topic中，然后订阅者通过订阅这些topic来做实时的监控分析，或者装载到hadoop、数据仓库中做离线分析和挖掘。
> 运营指标：Kafka也经常用来记录运营监控数据。包括收集各种分布式应用的数据，生产各种操作的集中反馈，比如报警和报告。
> 流式处理：比如spark streaming和storm

2、集群部署

zk集群作配置中心，kafka集群

- **普通版搭建：**单机版略，集群版参考文档，修改5处位置

```
broker.id=1
delete.topic.enable=true
listeners=PLAINTEXT://192.168.25.104:9092
log.dirs=/root/kafka/logs
zookeeper.connect=192.168.25.130:2182,192.168.25.130:2183,192.168.25.130:2184
目前三台kafka，一台zk伪分布
全伪分布参考https://blog.csdn.net/qq_34898847/article/details/83377179
```

jms命令、xsync命令 jms -l 进程主类全类名 之前ps可能进程名一致的很多个 xsync kafka/ 分发

- **docker版搭建:**

```
---单机
docker run --name zk01 -p 2181:2181 --restart always -d zookeeper
docker run -d --name kafka01 -e HOST_IP=localhost -e KAFKA_ADVERTISED_PORT=9092 -e
KAFKA_ADVERTISED_HOST_NAME=kafka01 -e KAFKA_ZOOKEEPER_CONNECT="localhost:2181" -e
KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://localhost:9092  -e KAFKA_BROKER_ID=1 -e ZK=zk -p
9092:9092 --link zk01:zk -t wurstmeister/kafka

---集群
参考以下博客和ncore
https://www.cnblogs.com/jay763190097/p/10292227.html 参考ncore
https://yq.aliyun.com/articles/716134
```

3、CLI操作

以三个结点为例

bin/kafka-server-start.sh config/server.properties & bin/kafka-server-stop.sh stop 1)查看当前服务器中的所有topic --zookeeper指定一台即可 集群会分发

```
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --list
```

2）查看某个 Topic 的详情

```
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --describe --topic test1
```

3)创建 topic 副本数不能大于节点数区别于HDFS

```
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --create --replication-factor 2 --partitions
2 --topic test1
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --create --replication-factor 3 --partitions
1 --topic test2
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --create --replication-factor 3 --partitions
2 --topic test3
```

4)删除topic

```
bin/kafka-topics.sh --zookeeper 192.168.25.130:2182 --delete --topic first
```

5)发送消息 生产者连kafka集群 如果没创建topic默认会根据consumer.prop1个副本1个分区创建topic --broker-list指定一个就好因为集群配置好了

```
bin/kafka-console-producer.sh --broker-list 192.168.25.102:9092 --topic test1
>hello world
>atguigu atguigu
```

6）消费消息 指定groupId一个组，默认随机创建 --from-beginning：会把 first 主题中以往所有的数据都读取出来。根据业务场景选择是否增加该配置。

```
消费者连zk，老版本zk管理offset
新版本本地--bootstrap-server hadoop102:9092  同时创建一个topic保存offset __consumer_offsets
bin/kafka-console-consumer.sh --zookeeper 192.168.25.130:2182 --from-beginning --topic test1
bin/kafka-console-consumer.sh --bootstrap-server  192.168.25.102:9092  --from-beginning --topic
test1
```

4、概念

为什么新版本offset不保存在zk，保存在本地?

```
提高效率。本身就需要和kafka通信，又要开线程和zk通信，效率低下。其实本地创建一个topic保存offset
__consumer_offsets。
```

Zookeeper 在 Kafka 中的作用

```
Kafka 集群中有一个 broker 会被选举为 Controller，负责管理集群 broker 的上下线，所有 topic 的分区副本分
配和 leader 选举等工作。Controller 的管理工作都是依赖于Zookeeper
```

几个重点

1. 分区partition用于负载均衡，Topic有分区，分区有副本 同时不同消费者可以消费不同分区topic，提高并发。
2. kafka集群和consumer依赖于zk，zk集群，读从结点直接返回，写操作主结点处理转发给从再响应，而kafka集群从结点只作备份或容灾。
3. 同一个组的消费者不能消费同一分区的数据，不同分区的数据可以，不同组的消费者可以消费同一分区的数据
4. 一个消费者可消费多个topic数据，可以消费不同区的数据
5. kafka 只保证按一个 partition 中的顺序将消息发给consumer，不保证一个 topic 的整体（多个 partition 间）的顺序

分区的原因

1. 方便在集群中扩展，每个 Partition 可以通过调整以适应它所在的机器，而一个 topic又可以有多个 Partition 组成，因此整个集群就可以适应任意大小的数据了；
2. 可以提高并发，因为可以以 Partition 为单位读写了。

分区的原则

1. 指定了 patition，则直接使用；
2. 未指定 patition 但指定 key，通过对 key 的 value 进行 hash 出一个 patition；
3. patition 和 key 都未指定，使用轮询选出一个 patition。

kafka事务

事务可以保证 Kafka 在 Exactly Once 语义的基础上，生产和消费可以跨分区和会话，要么全部成功，要么全部失败。

5、流程分析

5.1 写入的流程

1）producer先从Zookeeper的 “/brokers/../state”节点找到该partition的Leader 2）producer将消息发送给该leader 3）leader将消息写入本地log 4）followers从Leader pull消 息 5）写入本地log后向Leader发送ack 6）leader收到所有replication的ACK后，并向producer发送ACK

应答级别

```
应答级别ack{0, 1, all或-1}

0效率最高不确保，未写入磁盘就返回

1只确保leader

all所有确保 防止丢失数据（两个副本的时候0的效率大概是all的10倍）

为防止all，follower故障迟迟不同步，有ISR即与kafka同步的集合，长时间不同步，会被踢出局

LEO：指的是每个副本最大的 offset；

HW：指的是消费者能见到的最大的 offset，ISR 队列中最小的 LEO。
```

5.2 broker保存

存储方式

```
物理上把 topic 分成一个或多个 patition（对应 server.properties 中的 num.partitions=3 配置），每个 patition 物理上对应一个文件夹（该文件夹中log存储该 patition 的所有消息和索引文件），为防止 log 文件过大导致数据定位，效率低下，Kafka 采取了分片和索引机制，将每个 partition 分为多个 segment，每个 segment 对应两个文件—“.index”文件和“.log”文件


集群配置信息保存在zk中。
brokers:zkCli get /brokers/ids/0     get /brokers/topics/first/partitions/0/state

consumers: ls /consumers/console-consumer8888/offsets/first/0   仅限--zk启动消费的，--bootsrap维护在__topic中
```

存储策略　类比redis的六种。

无论消息是否被消费，kafka 都会保留所有消息。有两种策略可以删除旧数据：

1. 基于时间：log.retention.hours=168
2. 基于大小：log.retention.bytes=1073741824

需要注意的是，因为 Kafka 读取特定消息的时间复杂度为 O(1)，即与文件大小无关，所

以这里删除过期文件与提高 Kafka 性能无关。

5.3 消费过程分析

高级API：

【优】不需要自行去管理 offset，系统通过 zookeeper 自行管理。不需要管理分区，副本等情况，.系统自动管理

【缺】不能自行控制 offset（对于某些特殊需求来说）不能细化控制如分区、副本、zk 等

低级API：

【优】能够让开发者自己控制 offset，想从哪里读取就从哪里读取。对zk依赖降低（offset不一定非要靠zk存储，自行存储offset 即可，比如存在文件或者内存中）

【缺】太过复杂，需要自行控制 offset，连接哪个分区，找到分区 leader 等

消费者组

一般来说生产时 分区数n>=同一个消费者组消费者数m。

消费者是以 consumer group 消费者组的方式工作，由一个或者多个消费者组成一个组，共同消费一个 topic。 每个分区在同一时间只能由 group 中的一个消费者读取，但是多个 group可以同时消费这个 partition。

消费方式

```
P2P拉消费者不会丢失 广播推消费者网速不一致丢失

kafka采取pull模式，为避免可能的循环，拉请求设置参数允许长轮询阻塞

同一个消费者组中的消费者，同一时刻只能有一个消费者消费。 不同消费组不作限制
bin/kafka-console-consumer.sh --bootstrap-server  192.168.25.102:9092  --from-beginning --topic
test1  --consumer.config config/consumer.properties

如果不指定消费者组，默认创建消费者会自动分配随机的组
bin/kafka-console-producer.sh --broker-list 192.168.25.102:9092 --topic test1
```

6、API实战

6.1 测试生产者

CLI打开一个消费者bin/kafka-console-consumer.sh --bootstrap-server 192.168.25.102:9092 --from-beginning --topic first 客户端发送 *老API produce message 不指定分区 随机选择分区 *新API kafkaproduce producerecord 带回调函数 自定义指定分区的生产者 消费完一个分区再消费下一个分区 同一分区内消息有序 ProducerConfig 序列化到leader分区的log文件

6.2 测试消费者（只有消费者有高级和低级API，生产者只有新旧API）

CLI打开一个生产者bin/kafka-console-producer.sh --broker-list 192.168.25.102:9092 --topic first2

1. 高级API不可以指定分区，offset等 客户端启动消费 一个消费者可以消费多个topic

   反序化 autocommit 或者设定时间

   读数据----业务处理---提交offset 第三步宕机，可能出现重复消费 （生产环境中解决问题：采用低级API。业务数据完全处理完毕后再提交保持offset一致） 消费多个ArraysasList 单个推荐Collections.singlonList

2. 低级API 指定分区自行控制offset 用到nio包 如读取Topic second的Partition0从12offset开始读

   实现使用低级 API 读取指定 topic，指定 partition,指定 offset 的数据。 1）消费者使用低级 API 的主要步骤： 步骤 主要工作 1 根据指定的分区从主题元数据中找到主副本 如3副本100个机器第一次寻找从100台找leader副本 2 获取分区最新的消费进度 3 从主副本拉取分区的消息 4 识别主副本的变化，重试 保存leader和follower 若leader挂了下次直接找 2）方法描述： findLeader() 客户端向种子节点发送主题元数据，将副本集加入备用节点 getLastOffset() 消费者客户端发送偏移量请求，获取分区最近的偏移量 run() 消费者低级 API 拉取消息的主要方法 findNewLeader() 当分区的主副本节点发生故障，客户将要找出新的主副本leader

如果想重复消费一个topic两个办法

```
1一是新建一个组earliest， 类比--begining
2二是利用低级API控制offset，
3三高级API提供的assgin和seek方法指定offset但是无法维护offset只能每次都重置。
```

单播都在一个组，广播一个消费者一个组

如何查看__consumeroffsets里保存的偏移量数据

参考博文

组+topic+分区号唯一确定 一个组只维护一个offset 再次验证同一消费组不同消费者不能同时消费同一分区的topic数据 一秒提交一次 zk下是多级路径查看

```
先将consumer.properties中设置exclude.internal.topics=false
bin/kafka-console-consumer.sh --topic __consumer_offsets --zookeeper localhost:2181 --formatter
"kafka.coordinator.group.GroupMetadataManager$OffsetsMessageFormatter" --consumer.config
config/consumer.properties --from-beginning
```

7.扩展部分

1，拦截器

对于 producer 而言，interceptor 使得用户在消息发送前以及 producer 回调逻辑前有机会 对消息做一些定制化需求，比如修改消息等 configure()、send（）、onAcknowledgement(RecordMetadata, Exception)：、close（） 时间拦截器、计数拦截器等

2，监控 manager

修改kafka启动服务脚本、修改监控中间件的配置

3，Kafka Stream流式处理

类似的Spark微批处理（主流吞吐高） Storm一条条处理（扛不住大压力） Flink（上升趋势）

```
1）功能强大
高扩展性，弹性，容错
2）轻量级
无需专门的集群
一个库，而不是框架
3）完全集成
100%的 Kafka 0.10.0 版本兼容
易于集成到现有的应用程序
4）实时性
毫秒级延迟
并非微批处理
窗口允许乱序数据
允许迟到数据
```

选用kafka流式处理的原因:

```
1,Kafka Stream 提供的是一个基于 Kafka 的流式处理类库,直接使用，相对其他简单
2,打包部署没复杂要求，方便嵌入，其他的则很复杂
3,Kafka Stream 可以在线动态调整并行度
4,由于 Kafka 本身提供数据持久化，因此 Kafka Stream 提供滚动部署和滚动升级以及重新计算的能力
5,使用 Storm 或 Spark Streaming 时，需要为框架本身的进程预留资源，但是 Kafka 作为类库不占用系统资源
6,大部分流式系统中都已部署了 Kafka，此时使用 Kafka Stream 的成本非常低。
```

处理handle、processor 数据清洗案例 bin/kafka-console-producer.sh --broker-list 192.168.25.102:9092 --topic test1 bin/kafka-console-consumer.sh --zookeeper 192.168.25.130:2182 --from-beginning --topic stream888

和Flume对比

```
在企业中必须要清楚流式数据采集框架 flume 和 kafka 的定位是什么：
flume：cloudera 公司研发：
适合多个生产者；
适合下游数据消费者不多的情况；
适合数据安全性要求不高的操作；
适合与 Hadoop 生态圈对接的操作。
kafka：linkedin 公司研发：
适合数据下游消费众多的情况；
适合数据安全性要求较高的操作，支持 replication。
因此我们常用的一种模型是：
线上数据 --> flume --> kafka --> flume(根据情景增删该流程) --> HDFS
后台服务器---汇总agent---对接kafka---     离线（HDFS）、实时（对接SparkStreaming）
```

8、Springboot整合kafka

- 利用kafkatemplate
- 自己客户端封装

参考博客一

参考博客二

其他自定义配置，组合注解参考ncore的@QuarkEnableKafka消息总线