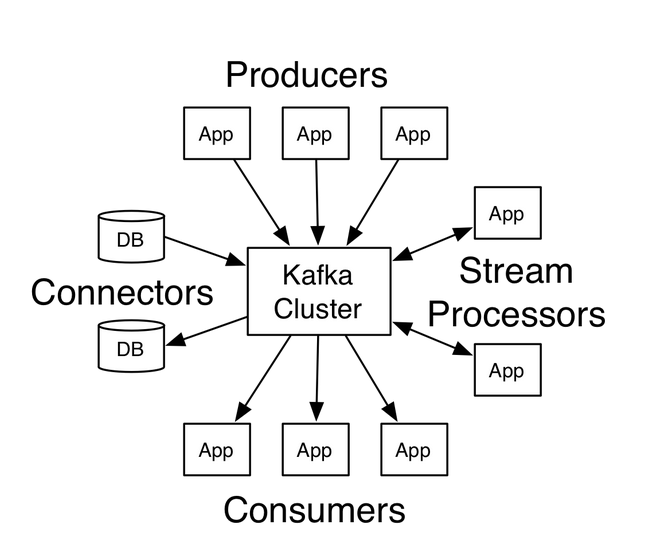
# Kafka guide



## 1 原理

kafka的[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128" \t "_blank)初衷是希望作为一个统一的信息收集平台,能够实时的收集反馈信息,并需要能够支撑较大的数据量,且具备良好的容错能力.

    1、持久性

    kafka使用文件存储消息,这就直接决定kafka在性能上严重依赖文件系统的本身特性.且无论任何OS下,对文件系统本身的优化几乎没有可能.文件缓存/直接内存映射等是常用的手段.因为kafka是对日志文件进行append操作,因此磁盘检索的开支是较小的;同时为了减少磁盘写入的次数,broker会将消息暂时buffer起来,当消息的个数(或尺寸)达到一定阀值时,再flush到磁盘,这样减少了磁盘IO调用的次数.

2、性能

    需要考虑的影响性能点很多,除磁盘IO之外,我们还需要考虑网络IO,这直接关系到kafka的吞吐量问题.kafka并没有提供太多高超的技巧;对于producer端,可以将消息buffer起来,当消息的条数达到一定阀值时,批量发送给broker;对于consumer端也是一样,批量fetch多条消息.不过消息量的大小可以通过配置文件来指定.对于kafka broker端,似乎有个sendfile系统调用可以潜在的提升网络IO的性能:将文件的数据映射到系统内存中,socket直接读取相应的内存区域即可,而无需进程再次copy和交换. 其实对于producer/consumer/broker三者而言,CPU的开支应该都不大,因此启用消息压缩机制是一个良好的策略;压缩需要消耗少量的CPU资源,不过对于kafka而言,网络IO更应该需要考虑.可以将任何在网络上传输的消息都经过压缩.kafka支持gzip/snappy等多种压缩方式.

    3、生产者

    负载均衡: producer将会和Topic下所有partition leader保持socket连接;消息由producer直接通过socket发送到broker,中间不会经过任何"路由层".事实上,消息被路由到哪个partition上,有producer[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)决定.比如可以采用"random""key-hash""轮询"等,如果一个topic中有多个partitions,那么在producer端实现"消息均衡分发"是必要的.

    其中partition leader的位置(host:port)注册在zookeeper中,producer作为zookeeper client,已经注册了watch用来监听partition leader的变更事件.

    异步发送：将多条消息暂且在客户端buffer起来，并将他们批量的发送到broker，小数据IO太多，会拖慢整体的网络延迟，批量延迟发送事实上提升了网络效率。不过这也有一定的隐患，比如说当producer失效时，那些尚未发送的消息将会丢失。

    4、消费者

    consumer端向broker发送"fetch"请求,并告知其获取消息的offset;此后consumer将会获得一定条数的消息;consumer端也可以重置offset来重新消费消息.

    在JMS实现中,Topic模型基于push方式,即broker将消息推送给consumer端.不过在kafka中,采用了pull方式,即consumer在和broker建立连接之后,主动去pull(或者说fetch)消息;这中模式有些优点,首先consumer端可以根据自己的消费能力适时的去fetch消息并处理,且可以控制消息消费的进度(offset);此外,消费者可以良好的控制消息消费的数量,batch fetch.

    其他JMS实现,消息消费的位置是有prodiver保留,以便避免重复发送消息或者将没有消费成功的消息重发等,同时还要控制消息的状态.这就要求JMS broker需要太多额外的工作.在kafka中,partition中的消息只有一个consumer在消费,且不存在消息状态的控制,也没有复杂的消息确认机制,可见kafka broker端是相当轻量级的.当消息被consumer接收之后,consumer可以在本地保存最后消息的offset,并间歇性的向zookeeper注册offset.由此可见,consumer[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)也很轻量级.

## 2 业务场景使用

1、Messaging

    对于一些常规的消息系统,kafka是个不错的选择;partitons/replication和容错,可以使kafka具有良好的扩展性和性能优势.不过到目前为止,我们应该很清楚认识到,kafka并没有提供JMS中的"事务性""消息传输担保(消息确认机制)""消息分组"等企业级特性;kafka只能使用作为"常规"的消息系统,在一定程度上,尚未确保消息的发送与接收绝对可靠(比如,消息重发,消息发送丢失等)

    2、Websit activity tracking

    kafka可以作为"网站活性跟踪"的最佳工具;可以将网页/用户操作等信息发送到kafka中.并实时监控,或者离线统计分析等

    3、Log Aggregation

    kafka的特性决定它非常适合作为"日志收集中心";application可以将操作日志"批量""异步"的发送到kafka集群中,而不是保存在本地或者DB中;kafka可以批量提交消息/压缩消息等,这对producer端而言,几乎感觉不到性能的开支.此时consumer端可以使hadoop等其他系统化的存储和分析系统.

## 3 运行环境与开发环境搭建

### 基于星环平台，对kafka组件的相关测试

### 1 kafka常用操作命令：

#### 1.1启动kafka：

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

#### 1.2 创建主题：

kafka-topics.sh --zookeeper localhost:2181 --replica 1 --partition 1 --topic topic

#### 1.3 添加认证方式后消息发送：

sh kafka-console-producer.sh --broker-list 172.16.19.152:9092 --topic test0524 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH

#### 1.4添加认证方式后消息消费

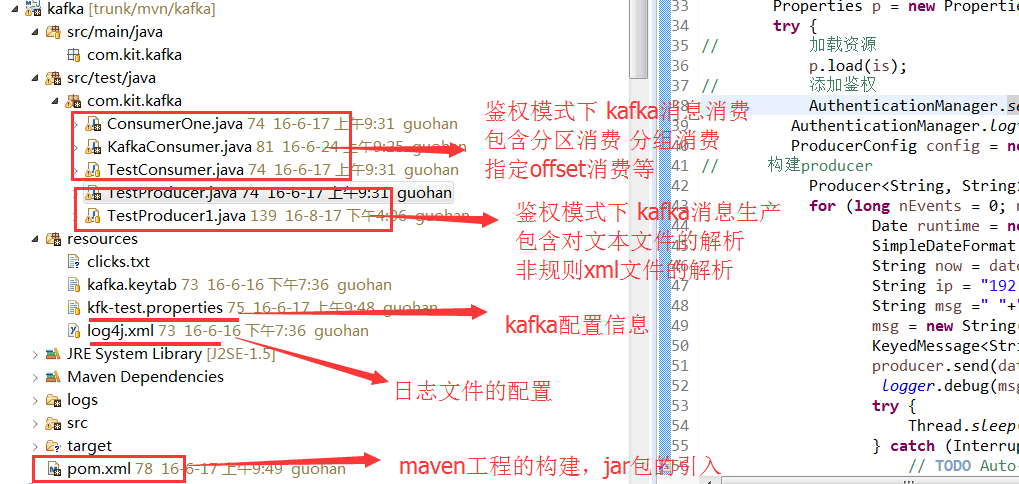
sh kafka-console-consumer.sh -zookeeper 172.16.19.152:2181 --topic test0524 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH

#### 1.5 添加认证后 删除主题操作：

sh kafka-run-class.sh kafka.admin.DeleteTopicCommand --topic test05 --zookeeper 172.16.19.152:2181 --keytab /etc/kafka1/kafka.keytab --principal kafka/kit-b2@TDH --broker 172.16.19.152:9092

### 2构建kafka，maven工程步骤：

#### 2.1 创建java工程



#### 2.2 配置pom.xml文件，引用相关jar包

注意事项：需引用当前服务器下适用的kafka版本jar包



Pom.xml文件配置：



#### 2.3 配置kafka资源文件与日志文件，设定相关配置



#### 2.4 添加鉴权文件配置

注意事项：因引用的是第三方组件，需引用对应的jar包，在工程初始化的时，需进行鉴权配置



#### 2.5 编码实现示列



##### 2.5.1生产消息：

a 加载配置（读取配置文件）

b 添加鉴权

c 构建Producer

d 发送消息

e 关闭producer，或让线程休眠

##### 2.5.2消费消息：

a加载配置（读取配置文件）

b 添加鉴权

c 构建consumer消费器

d 消费消息



## 4 代码演示

## 5日志查看与监听管理



/opt/soft/kafka-manager/kafka-manager-1.3.1.6# ./bin/kafka-manager -Dhttp.port=9002

