# 课程介绍

课程名称：Storm上游数据源之Kakfa

课程目标：

通过本课程理解Storm消费的数据来源、理解JMS规范、理解Kafka核心组件、掌握Kakfa生产者API、掌握Kafka消费者API。对流式计算的生态环境有深入的了解，具备流式计算项目架构的能力。

课程大纲：

1. kafka是什么？
2. JMS规范是什么？
3. 为什么需要消息队列？
4. Kafka核心组件
5. Kafka安装部署
6. Kafka生产者Java API
7. Kafka消费者Java API

# 课程内容

## 1、Kafka是什么

在流式计算中，Kafka一般用来缓存数据，Storm通过消费Kafka的数据进行计算。

KAFKA + STORM +REDIS

* Apache Kafka是一个开源**消息**系统，由Scala写成。是由Apache软件基金会开发的一个开源消息系统项目。
* Kafka最初是由LinkedIn开发，并于2011年初开源。2012年10月从Apache Incubator毕业。该项目的目标是为处理实时数据提供一个统一、高通量、低等待的平台。
* **Kafka是一个分布式消息队列：生产者、消费者的功能。它提供了类似于JMS的特性，但是在设计实现上完全不同，此外它并不是JMS规范的实现。**
* Kafka对消息保存时根据Topic进行归类，发送消息者称为Producer,消息接受者称为Consumer,此外kafka集群有多个kafka实例组成，每个实例(server)成为broker。
* 无论是kafka集群，还是producer和consumer都依赖于**zookeeper**集群保存一些meta信息，来保证系统可用性

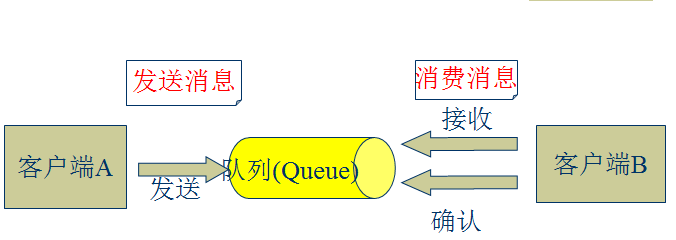
## 2、JMS是什么

### 2.1、JMS的基础

JMS是什么：JMS是Java提供的一套技术规范

JMS干什么用：用来异构系统 集成通信，缓解系统瓶颈，提高系统的伸缩性增强系统用户体验，使得系统模块化和组件化变得可行并更加灵活

通过什么方式：生产消费者模式（生产者、服务器、消费者）



jdk，kafka，activemq……

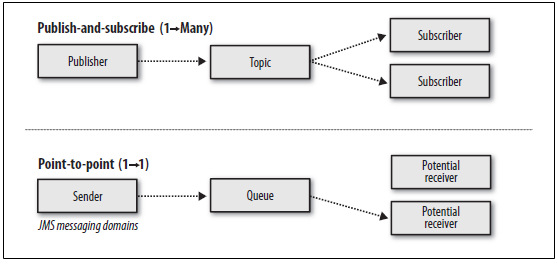
### 2.2、JMS消息传输模型

* 点对点模式**（一对一，消费者主动拉取数据，消息收到后消息清除）**

点对点模型通常是一个基于拉取或者轮询的消息传送模型，这种模型从队列中请求信息，而不是将消息推送到客户端。这个模型的特点是发送到队列的消息被**一个且只有一个接收者接收处理**，即使有多个消息监听者也是如此。

* 发布/订阅模式**（一对多，数据生产后，推送给所有订阅者）**

发布订阅模型则是一个基于推送的消息传送模型。发布订阅模型可以有多种不同的订阅者，临时订阅者只在主动监听主题时才接收消息，而持久订阅者则监听主题的所有消息，**即时当前订阅者不可用，处于离线状态**。

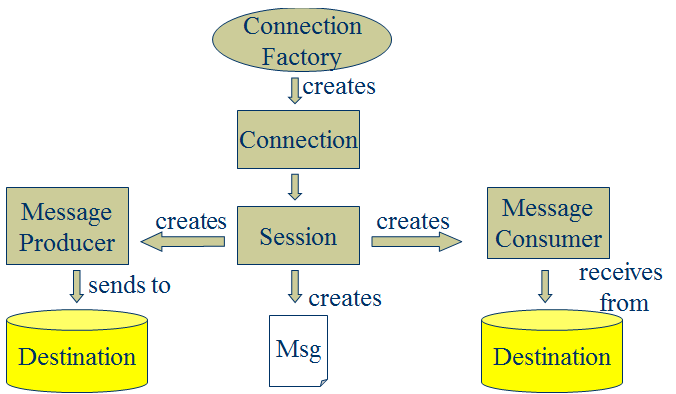


queue.put（object） 数据生产

queue.take(object) 数据消费

### 2.3、JMS核心组件

* Destination：消息发送的目的地，也就是前面说的Queue和Topic。
* Message ：从字面上就可以看出是被发送的消息。
* Producer： 消息的生产者，要发送一个消息，必须通过这个生产者来发送。
* MessageConsumer： 与生产者相对应，这是消息的消费者或接收者，通过它来接收一个消息。



通过与ConnectionFactory可以获得一个connection

通过connection可以获得一个session会话。

### 2.4、常见的类JMS消息服务器

#### 2.4.1、JMS消息服务器 ActiveMQ

ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的。

主要特点：

* 多种语言和协议编写客户端。语言: Java, C, C++, C#, Ruby, Perl, Python, PHP。应用协议: OpenWire,Stomp REST,WS Notification,XMPP,AMQP
* 完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)
* 对Spring的支持,ActiveMQ可以很容易内嵌到使用Spring的系统里面去,而且也支持Spring2.0的特性
* 通过了常见J2EE服务器(如 Geronimo,JBoss 4, GlassFish,WebLogic)的测试,其中通过JCA 1.5 resource adaptors的配置,可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上
* 支持多种传送协议:in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA
* 支持通过JDBC和journal提供高速的消息持久化
* 从设计上保证了高性能的集群,客户端-服务器,点对点
* 支持Ajax
* 支持与Axis的整合
* 可以很容易得调用内嵌JMS provider,进行测试

#### 2.4.2、分布式消息中间件 Metamorphosis

Metamorphosis (**MetaQ**) 是一个高性能、高可用、可扩展的分布式消息中间件，类似于LinkedIn的Kafka，具有消息存储顺序写、吞吐量大和支持本地和XA事务等特性，适用于大吞吐量、顺序消息、广播和日志数据传输等场景，在淘宝和支付宝有着广泛的应用，现已开源。

主要特点：

* 生产者、服务器和消费者都可分布
* 消息存储顺序写
* 性能极高,吞吐量大
* 支持消息顺序
* 支持本地和XA事务
* 客户端pull，随机读,利用sendfile系统调用，zero-copy ,批量拉数据
* 支持消费端事务
* 支持消息广播模式
* 支持异步发送消息
* 支持http协议
* 支持消息重试和recover
* 数据迁移、扩容对用户透明
* 消费状态保存在客户端
* 支持同步和异步复制两种HA
* 支持group commit

#### 2.4.3、分布式消息中间件 RocketMQ

RocketMQ 是一款分布式、队列模型的消息中间件，具有以下特点：

* 能够保证严格的消息顺序
* 提供丰富的消息拉取模式
* 高效的订阅者水平扩展能力
* 实时的消息订阅机制
* 亿级消息堆积能力
* **Metaq3.0 版本改名，产品名称改为RocketMQ**

#### 2.4.4、其他MQ

* .NET消息中间件 DotNetMQ
* 基于HBase的消息队列 HQueue
* Go 的 MQ 框架 KiteQ
* AMQP消息服务器 RabbitMQ
* MemcacheQ 是一个基于 MemcacheDB 的消息队列服务器。

## 3、为什么需要消息队列（重要）

消息系统的核心作用就是三点：解耦，异步和并行

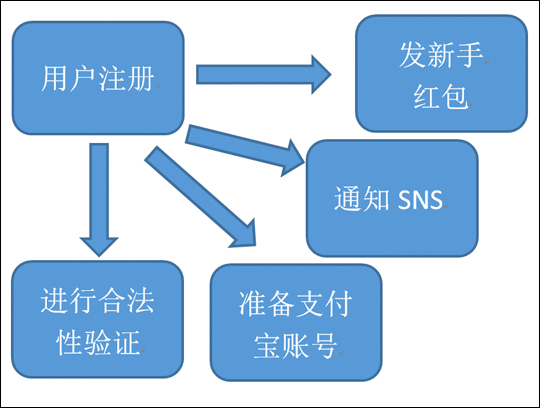
以用户注册的案列来说明消息系统的作用

### 3.1、用户注册的一般流程

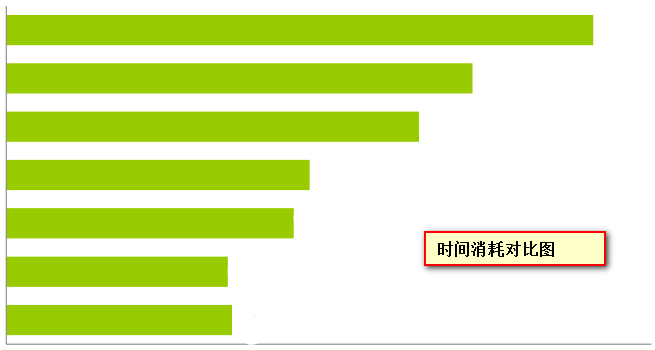


**问题**：随着后端流程越来越多，每步流程都需要额外的耗费很多时间，从而会导致用户更长的等待延迟。

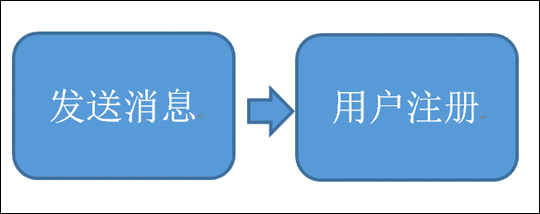
### 3.2、用户注册的并行执行



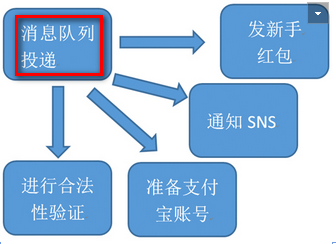
**问题**：系统并行的发起了4个请求，4个请求中，如果某一个环节执行1分钟，其他环节再快，用户也需要等待1分钟。如果其中一个环节异常之后，整个服务挂掉了。



### 3.3、用户注册的最终一致



1. **保证主流程的正常执行**、执行成功之后，发送MQ消息出去。
2. 需要这个destination的其他系统通过消费数据再执行，最终一致。



## 4、Kafka核心组件

* Topic ：消息根据Topic进行归类
* Producer：发送消息者
* Consumer：消息接受者
* broker：每个kafka实例(server)
* Zookeeper：依赖集群保存meta信息。



## 5、Kafka集群部署

### 5.1集群部署的基本流程

下载安装包、解压安装包、修改配置文件、分发安装包、启动集群

### 5.2集群部署的基础环境准备

安装前的准备工作（zk集群已经部署完毕）

* 关闭防火墙

chkconfig iptables off && setenforce 0

* 创建用户

groupadd realtime &&　useradd realtime　&& usermod -a -G realtime realtime

* 创建工作目录并赋权

mkdir /export

mkdir /export/servers

chmod 755 -R /export

* 切换到realtime用户下

su realtime

### 5.3 Kafka集群部署

#### 5.3.1、下载安装包

<http://kafka.apache.org/downloads.html>

在linux中使用wget命令下载安装包

wget http://mirrors.hust.edu.cn/apache/kafka/0.8.2.2/kafka\_2.11-0.8.2.2.tgz

#### 5.3.2、解压安装包

tar -zxvf /export/software/kafka\_2.11-0.8.2.2.tgz -C /export/servers/

cd /export/servers/

ln -s kafka\_2.11-0.8.2.2 kafka

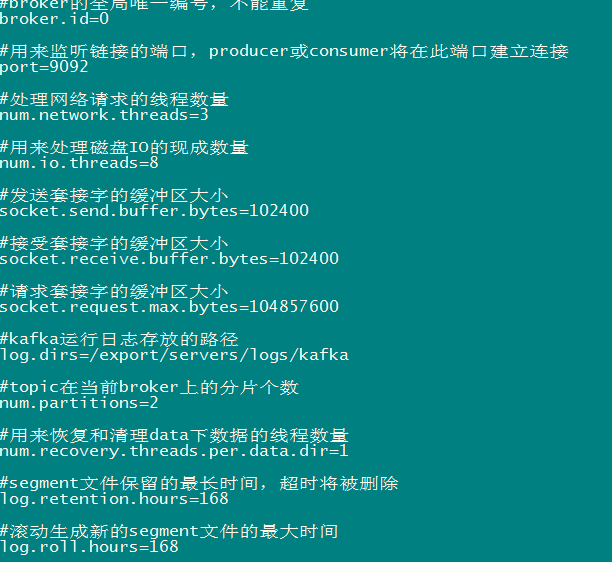
#### 5.3.3、修改配置文件

cp /export/servers/kafka/config/server.properties

/export/servers/kafka/config/server.properties.bak

vi /export/servers/kafka/config/server.properties

输入以下内容：



#### 5.3.4、分发安装包

scp -r /export/servers/kafka\_2.11-0.8.2.2 kafka02:/export/servers

然后分别在各机器上创建软连

cd /export/servers/

ln -s kafka\_2.11-0.8.2.2 kafka

#### 5.3.5、再次修改配置文件（重要）

**依次修改各服务器上配置文件的的broker.id，分别是0,1,2不得重复。**

#### 5.3.6、启动集群

依次在各节点上启动kafka

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

### 5.4、Kafka常用操作命令

* 查看当前服务器中的所有topic

bin/kafka-topics.sh --list --zookeeper zk01:2181

* 创建topic

./kafka-topics.sh --create --zookeeper mini1:2181 --replication-factor 1 --partitions 3 --topic first

* 删除topic

sh bin/kafka-topics.sh --delete --zookeeper zk01:2181 --topic test

需要server.properties中设置**delete.topic.enable=true**否则只是标记删除或者直接重启。

* 通过shell命令发送消息

kafka-console-producer.sh --broker-list kafka01:9092 --topic itheima

* 通过shell消费消息

sh bin/kafka-console-consumer.sh --zookeeper zk01:2181 --from-beginning --topic test1

**不加--from-beginning表示从从头消费，offset为最大值**

* 查看消费位置

sh kafka-run-class.sh kafka.tools.ConsumerOffsetChecker --zookeeper zk01:2181 --group testGroup

* 查看某个Topic的详情

sh kafka-topics.sh --topic test --describe --zookeeper zk01:2181

## 6、Kafka生产者Java API



## 7、Kafka消费者Java API

