# 课程介绍

课程名称：Storm是什么，Storm集群部署及单词技术

课程目标：

通过该课程的学习能够了解离线计算与流式计算的区别、掌握Storm框架的基础知识、了解流式计算的一般架构图。通过本课程能够掌握Strom集群搭建、Storm配置文件、Storm源码管理、Storm编程模型。

课程大纲：

1. 离线计算是什么？
2. 流式计算是什么？
3. 流式计算与离线计算的区别？
4. Storm是什么？
5. Storm与Hadoop的区别？
6. Storm的应用场景及行业案例
7. Storm的核心组件（重点掌握）
8. Storm的编程模型（重点掌握）
9. 流式计算的一般架构图(重点掌握)
10. 集群部署的基本流程
11. 集群部署的基础环境准备
12. Storm集群部署
13. Storm集群的常用操作命令
14. Storm集群的进程及日志熟悉
15. Storm源码下载及目录熟悉
16. Storm 单词计数案列

# 背景介绍

Storm背景介绍

# 课程内容

## 1、离线计算是什么？

离线计算：批量获取数据、批量传输数据、**周期性**批量计算数据、数据展示

代表技术：Sqoop批量导入数据、HDFS批量存储数据、MapReduce批量计算数据、Hive批量计算数据、\*\*\*任务调度

1，hivesql

2、调度平台

3、Hadoop集群运维

4、数据清洗（脚本语言）

5、元数据管理

6、数据稽查

7、数据仓库模型架构

## 2、流式计算是什么

流式计算：数据实时产生、数据实时传输、数据实时计算、实时展示

代表技术：Flume实时获取数据、Kafka/**metaq**实时数据存储、**Storm/JStorm**实时数据计算、Redis实时**结果**缓存、持久化存储(mysql)。

一句话总结：将源源不断产生的数据实时收集并实时计算，尽可能快的得到计算结果

## 3、离线计算与实时计算的区别

最大的区别：实时收集、实时计算、实时展示

## 4、Storm是什么？

Flume实时采集，低延迟

Kafka消息队列，低延迟

Storm实时计算，低延迟

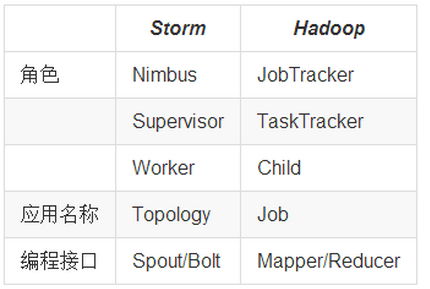
Redis实时存储，低延迟

Storm用来**实时处理数据**，特点：低延迟、高可用、分布式、可扩展、**数据不丢失**。提供简单容易理解的接口，便于开发。

海量数据？数据类型很多，产生数据的终端很多，处理数据能力增强

## 5、Storm与Hadoop的区别

* Storm用于实时计算，Hadoop用于离线计算。
* Storm处理的数据保存在内存中，源源不断；Hadoop处理的数据保存在文件系统中，一批一批。
* Storm的数据通过网络传输进来；Hadoop的数据保存在磁盘中。
* Storm与Hadoop的编程模型相似



Job：任务名称

JobTracker：项目经理

TaskTracker：开发组长、产品经理

Child:负责开发的人员

Mapper/Reduce:开发人员中的两种角色，一种是服务器开发、一种是客户端开发

Topology:任务名称

Nimbus:项目经理

Supervisor:开组长、产品经理

Worker:开人员

Spout/Bolt：开人员中的两种角色，一种是服务器开发、一种是客户端开发

## 6、Storm应用场景及行业案例

Storm用来**实时计算**源源不断产生的数据，如同流水线生产。

### 6.1、运用场景

* **日志分析**

从**海量**日志中分析出特定的**数据**，并将**分析**的结果**存入外部存储器**用来辅佐决策。

* **管道系统**

将一个数据从一个系统传输到另外一个系统，比如将数据库同步到Hadoop

* **消息转化器**

将接受到的消息按照某种格式进行转化，存储到另外一个系统如消息中间件

### 6.2、典型案列

* **一淘-实时分析系统：实时分析用户的属性，并反馈给搜索引擎**

最初，用户属性分析是通过每天在云梯上定时运行的MR job来完成的。为了满足实时性的要求，**希望能够实时分析用户的行为日志**，将最新的用户属性反馈给搜索引擎，能够为用户展现最贴近其当前需求的结果。

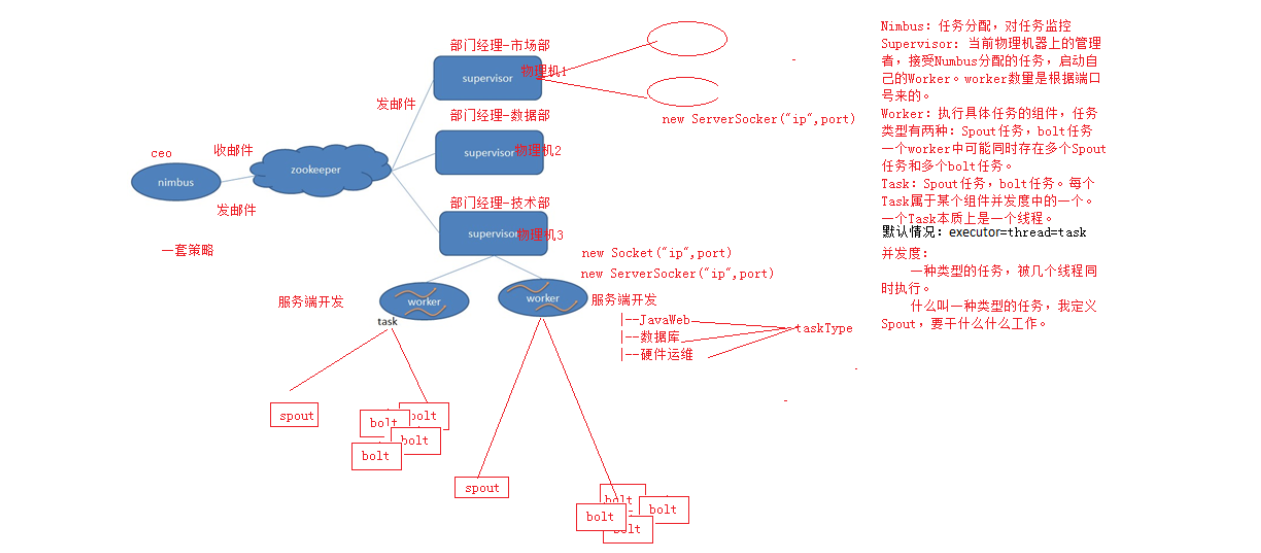
* **携程-网站性能监控：实时分析系统监控携程网的网站性能**

利用HTML5提供的performance标准获得可用的指标，并记录日志。Storm集群实时分析日志和入库。使用DRPC聚合成报表，通过历史数据对比等判断规则，触发预警事件。

* **阿里妈妈-用户画像：实时计算用户的兴趣数据**

为了更加精准投放广告，阿里妈妈后台计算引擎需要维护每个用户的兴趣点（理想状态是，你对什么感兴趣，就向你投放哪类广告）。用户兴趣主要基于用户的历史行为、用户的实时查询、用户的实时点击、用户的地理信息而得，其中实时查询、实时点击等用户行为都是实时数据。**考虑到系统的实时性**，阿里妈妈使用Storm维护用户兴趣数据，并在此基础上进行受众定向的广告投放。

## 7、Storm核心组件（重要）



**架构**

Nimbus：任务分配

Supervisor：接受任务，并启动worker。worker的数量根据端口号来的。

Worker:执行任务的具体组件（其实就是一个JVM）,可以执行两种类型的任务，Spout任务或者bolt任务。

Task：Task=线程=executor。 一个Task属于一个Spout或者Bolt并发任务。

Zookeeper：保存任务分配的信息、心跳信息、元数据信息。

* Nimbus：负责资源分配和任务调度。
* Supervisor：负责接受nimbus分配的任务，启动和停止属于自己管理的worker进程。**---通过配置文件设置当前supervisor上启动多少个worker。**
* Worker：运行具体处理组件逻辑的进程。Worker运行的任务类型只有两种，一种是Spout任务，一种是Bolt任务。
* Task：worker中每一个spout/bolt的线程称为一个task. 在storm0.8之后，task不再与物理线程对应，不同spout/bolt的task可能会共享一个物理线程，该线程称为executor。

**并发度**

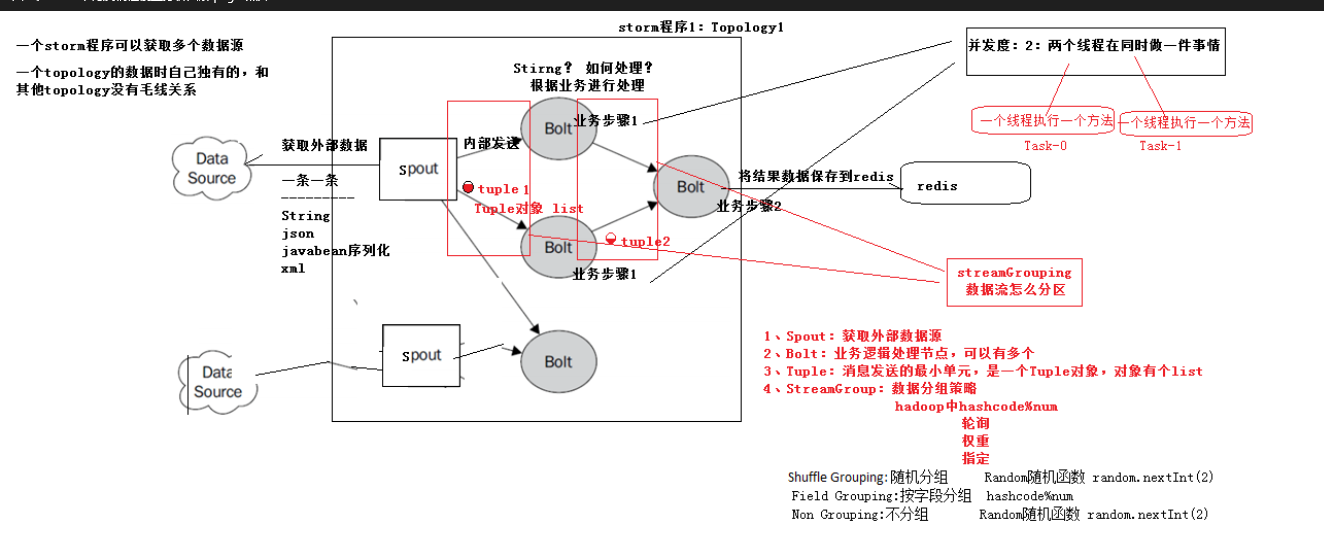
用户指定的一个任务，可以被多个线程执行，**并发度的数量等于线程的数量**。一个任务的多个线程，会被运行在多个Worker（JVM）上，有一种类似于平均算法的负载均衡策略。尽可能减少网络IO，和Hadoop中的MapReduce中的本地计算的道理一样。

**Worker与topology**

一个worker只属于一个topology,每个worker中运行的task只能属于这个topology。 反之，一个topology包含多个worker，其实就是这个topology运行在多个worker上。

一个topology要求的worker数量如果不被满足，集群在任务分配时，根据现有的worker先运行topology。如果当前集群中worker数量为0，那么最新提交的topology将只会被标识active，不会运行，只有当集群有了空闲资源之后，才会被运行。

## 8、Storm编程模型（重要）



DataSource：外部数据源

Spout：接受外部数据源的组件，将外部数据源转化成Storm内部的数据，以Tuple为基本的传输单元下发给Bolt

Bolt:接受Spout发送的数据，或上游的bolt的发送的数据。根据业务逻辑进行处理。发送给下一个Bolt或者是存储到某种介质上。介质可以是Redis可以是mysql，或者其他。

Tuple：Storm内部中数据传输的基本单元，里面封装了一个List对象，用来保存数据。

StreamGrouping:数据分组策略

7种：shuffleGrouping(Random函数),Non Grouping(Random函数),FieldGrouping(Hash取模)、Local or ShuffleGrouping 本地或随机，优先本地。

* Topology：Storm中运行的一个实时应用程序的名称。（拓扑）
* Spout：在一个topology中获取**源数据流**的组件。

通常情况下spout会从外部数据源中读取数据，然后转换为topology内部的源数据。

* Bolt：接受数据然后执行处理的组件,用户可以在其中执行自己想要的操作。
* Tuple：一次消息传递的基本单元，理解为一组消息就是一个Tuple。
* Stream：表示数据的流向。

## 9、流式计算一般架构图（重要）



* 其中flume用来获取数据。
* Kafka用来临时保存数据。
* Strom用来计算数据。
* Redis是个内存数据库，用来保存数据。

# 课程内容

## 10、集群部署的基本流程

集群部署的流程：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

注意：

    所有的集群上都需要配置hosts

    vi  /etc/hosts

192.168.239.128 storm01 zk01 hadoop01

      192.168.239.129 storm02 zk02 hadoop02

* + - 1. storm03 zk03 hadoop03

## 11、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

## 12、Storm集群部署

### 12.1、下载安装包

wget    <http://124.202.164.6/files/1139000006794ECA/apache.fayea.com/storm/apache-storm-0.9.5/apache-storm-0.9.5.tar.gz>

### 12.2、解压安装包

tar -zxvf apache-storm-0.9.5.tar.gz -C /export/servers/

cd /export/servers/

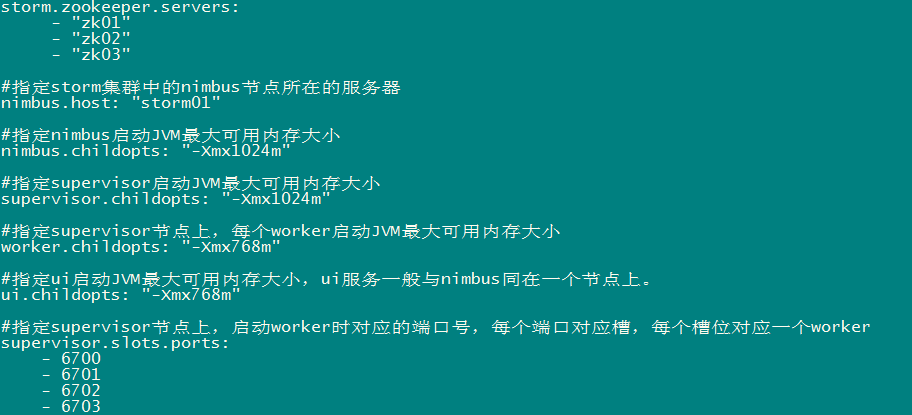
ln -s apache-storm-0.9.5 storm

### 12.3、修改配置文件

mv /export/servers/storm/conf/storm.yaml /export/servers/storm/conf/storm.yaml.bak

vi /export/servers/storm/conf/storm.yaml

输入以下内容：



### 13.4、分发安装包

scp -r /export/servers/apache-storm-0.9.5 storm02:/export/servers

然后分别在各机器上创建软连接

cd /export/servers/

ln -s apache-storm-0.9.5 storm

### 13.5、启动集群

* 在nimbus.host所属的机器上启动 nimbus服务

cd /export/servers/storm/bin/

**nohup ./storm nimbus &**

* 在nimbus.host所属的机器上启动ui服务

cd /export/servers/storm/bin/

**nohup ./storm ui &**

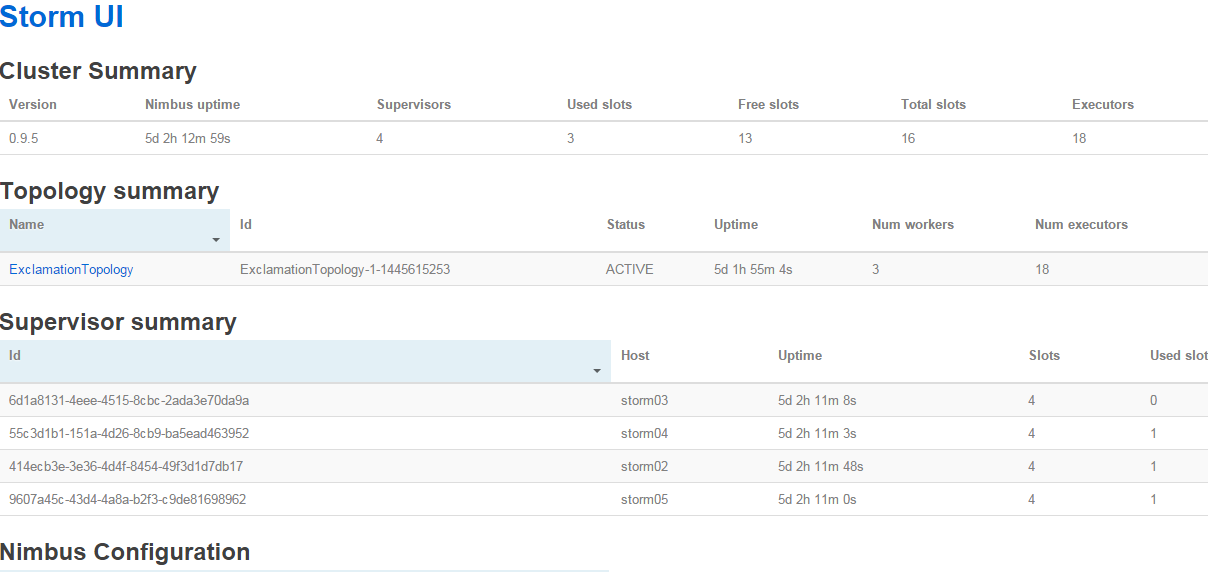
* 在其它个点击上启动supervisor服务

cd /export/servers/storm/bin/

**nohup ./storm supervisor &**

### 13.6、查看集群

访问nimbus.host:/8080，即可看到storm的ui界面。



# 14、Storm常用操作命令

有许多简单且有用的命令可以用来管理拓扑，它们可以提交、杀死、禁用、再平衡拓扑。

* 提交任务命令格式：storm jar 【jar路径】 【拓扑包名.拓扑类名】 【拓扑名称】

bin/storm jar examples/storm-starter/storm-starter-topologies-0.9.6.jar storm.starter.WordCountTopology wordcount

* 杀死任务命令格式：storm kill 【拓扑名称】 -w 10（执行kill命令时可以通过-w [等待秒数]指定拓扑停用以后的等待时间）

storm kill topology-name -w 10

* 停用任务命令格式：storm deactivte 【拓扑名称】

storm deactivte topology-name

我们能够挂起或停用运行中的拓扑。当停用拓扑时，所有已分发的元组都会得到处理，但是spouts的nextTuple方法不会被调用。销毁一个拓扑，可以使用kill命令。它会以一种安全的方式销毁一个拓扑，首先停用拓扑，在等待拓扑消息的时间段内允许拓扑完成当前的数据流。

* 启用任务命令格式：storm activate【拓扑名称】

storm activate topology-name

* **重新部署任务命令格式：storm rebalance 【拓扑名称】**

storm rebalance topology-name

再平衡使你重分配集群任务。这是个很强大的命令。比如，你向一个运行中的集群增加了节点。再平衡命令将会停用拓扑，然后在相应超时时间之后重分配工人，并重启拓扑。

# 15、Storm集群的进程及日志熟悉

## 15.1、部署成功之后，启动storm集群。

依次启动集群的各种角色

## 15.2、查看nimbus的日志信息

在nimbus的服务器上

cd /export/servers/storm/logs

tail -100f /export/servers/storm/logs/nimbus.log

## 15.3、查看ui运行日志信息

在ui的服务器上，一般和nimbus一个服务器

cd /export/servers/storm/logs

tail -100f /export/servers/storm/logs/ui.log

## 15.4、查看supervisor运行日志信息

在supervisor服务上

cd /export/servers/storm/logs

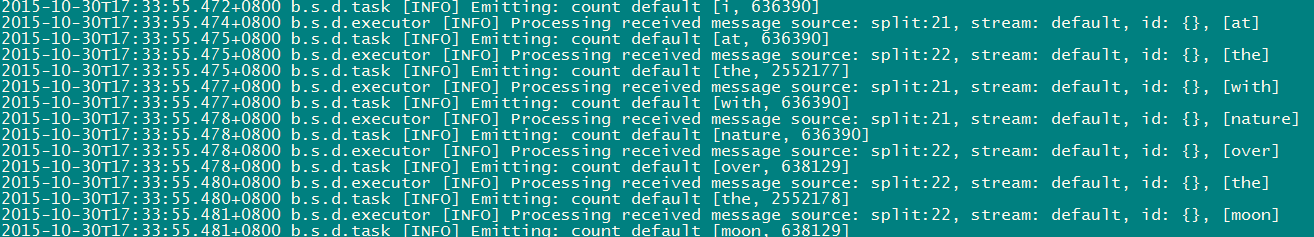
tail -100f /export/servers/storm/logs/supervisor.log

## 15.5、查看supervisor上worker运行日志信息

在supervisor服务上

cd /export/servers/storm/logs

tail -100f /export/servers/storm/logs/worker-6702.log

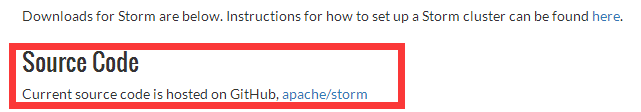


(该worker正在运行wordcount程序)

# 16、Storm源码下载及目录熟悉

## 16.1、在Storm官方网站上寻找源码地址

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



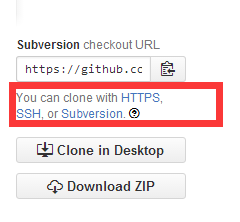
## 16.2、点击文字标签进入github

点击Apache/storm文字标签，进入github

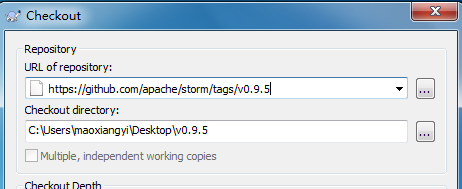
<https://github.com/apache/storm>

## 16.3、拷贝storm源码地址

在网页右侧，拷贝storm源码地址



## 16.4、使用Subversion客户端下载

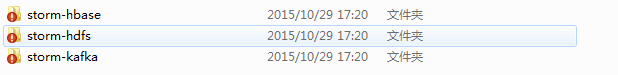


<https://github.com/apache/storm/tags/v0.9.5>

## 16.5、Storm源码目录分析（重要）



扩展包中的三个项目，使storm能与hbase、hdfs、kafka交互



# 17、Storm单词技术案例（重点掌握）

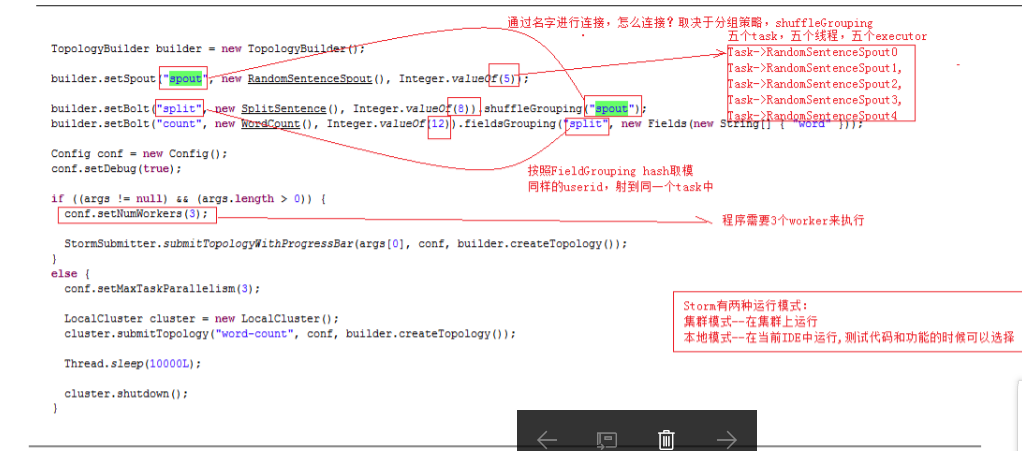
## 17.1、功能说明

<https://www.cnblogs.com/xuwujing/p/8584684.html>

设计一个topology，来实现对文档里面的单词出现的频率进行统计。

整个topology分为三个部分：

* RandomSentenceSpout：数据源，在已知的英文句子中，随机发送一条句子出去。
* SplitSentenceBolt：负责将单行文本记录（句子）切分成单词
* WordCountBolt：负责对单词的频率进行累加



## 17.2、项目主要流程



## 17.3、RandomSentenceSpout的实现及生命周期



## 17.4、SplitSentenceBolt的实现及生命周期

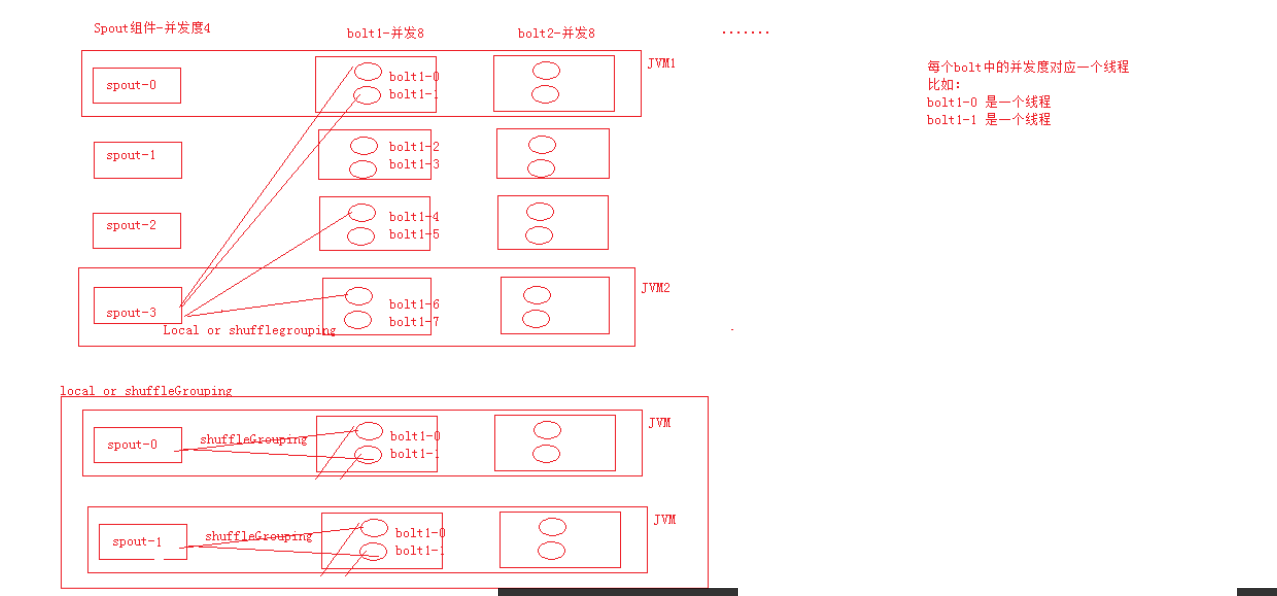


## 17.5、WordCountBolt的实现及生命周期



## 17.6、Stream Grouping详解

Storm里面有7种类型的stream grouping

* Shuffle Grouping: 随机分组， 随机派发stream里面的tuple，保证每个bolt接收到的tuple数目大致相同。
* Fields Grouping：按字段分组，比如按userid来分组，具有同样userid的tuple会被分到相同的Bolts里的一个task，而不同的userid则会被分配到不同的bolts里的task。
* All Grouping：广播发送，对于每一个tuple，所有的bolts都会收到。
* Global Grouping：全局分组， 这个tuple被分配到storm中的一个bolt的其中一个task。再具体一点就是分配给id值最低的那个task。
* Non Grouping：不分组，这stream grouping个分组的意思是说stream不关心到底谁会收到它的tuple。**目前这种分组和Shuffle grouping是一样的效果，** 有一点不同的是storm会把这个bolt放到这个bolt的订阅者同一个线程里面去执行。
* Direct Grouping： 直接分组， 这是一种比较特别的分组方法，用这种分组意味着消息的发送者指定由消息接收者的哪个task处理这个消息。只有被声明为Direct Stream的消息流可以声明这种分组方法。而且这种消息tuple必须使用emitDirect方法来发射。消息处理者可以通过TopologyContext来获取处理它的消息的task的id （OutputCollector.emit方法也会返回task的id）。
* **Local or shuffle grouping：如果目标bolt有一个或者多个task在同一个工作进程中，tuple将会被随机发生给这些tasks。否则，和普通的Shuffle Grouping行为一致。**
* 

总结：

启动流程：

<https://www.cnblogs.com/sunfie/p/7268157.html>

# [Storm启动流程分析](https://www.cnblogs.com/sunfie/p/7268157.html)

1. 客户端运行storm nimbus时，会调用storm的python脚本，该脚本中为每个命令编写一个方法，每个方法都可以生成一条相应的java命令。  
命令格式如下：java -server xxxx.ClassName -args

nimbus---> Running: /export/servers/jdk/bin/java -server backtype.storm.daemon.nimbus

supervisor---> Running: /export/servers/jdk/bin/java -server backtype.storm.daemon.supervisor

2. nibums启动之后，接受客户端提交任务

命令格式：storm jar xxx.jar xxx驱动类 参数

Running: /export/servers/jdk/bin/java -client -Dstorm.jar=/export/servers/storm/examples/storm-starter/storm-starter-topologies-0.9.6.jar storm.starter.WordCountTopology wordcount-28

　　该命令会执行 storm-starter-topologies-0.9.6.jar 中的storm-starter-topologies-0.9.6.jar的main方法，main方法中会执行以下代码：

StormSubmitter.submitTopology("mywordcount",config,topologyBuilder.createTopology());

　　topologyBuilder.createTopology()，会将程序员编写的spout对象和bolt对象进行序列化，并将用户的jar上传到 nimbus物理节点的 /export/data/storm/workdir/nimbus/inbox目录下。同时修改名称，改名的规则是**添加了一个UUID字符串**。  
在nimbus物理节点的 /export/data/storm/workdir/nimbus/stormdist目录下,有当前正在运行的topology的jar包和配置文件，序列化对象文件。

3. nimbus接受到任务之后，会将任务进行分配，分配会产生一个**assignment对象，该对象会保存到zk中，目录是/storm/assignments ，**该目录只保存正在运行的topology任务。

4. **supervisor通过watch机制**，感知到nimbus在zk上的任务分配信息，从**zk上拉取任务信息，分辨出属于自己任务。**

ResourceWorkerSlot[hostname=192.168.1.106,memSize=0,cpu=0,tasks=[1, 2, 3, 4, 5, 6, 7, 8],jvm=<null>,nodeId=61ce10a7-1e78-4c47-9fb3-c21f43a331ba,port=6900]

5. supervisor 根据自己的任务信息，启动自己的worker，并分配一个端口。

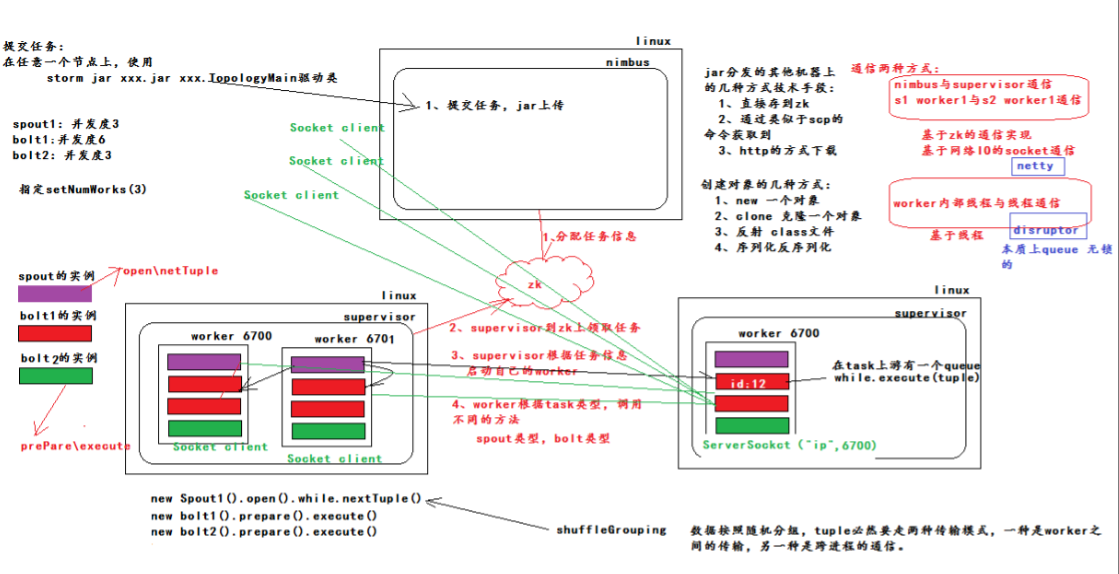
'/export/servers/jdk/bin/java' '-server' '-Xmx768m' export/data/storm/workdir/supervisor/stormdist/wordcount1-3-1461683066/stormjar.jar' 'backtype.storm.daemon.worker' 'wordcount1-3-1461683066' 'a69bb8fc-e08e-4d55-b51f-e539b066f90b' '6701' '9fac2805-7d2b-4e40-aabc-1c85c9856d64'

6. worker启动之后，连接zk，拉取任务

ResourceWorkerSlot[hostname=192.168.1.106,memSize=0,cpu=0,tasks=[1, 2, 3, 4, 5, 6, 7, 8],jvm=<null>,nodeId=61ce10a7-1e78-4c47-9fb3-c21f43a331ba,port=6900]

假设任务信息：  
1--->spout---type:spout  
2--->bolt ---type:bolt  
3--->acker---type:bolt  
  
得到对象有几种方式？ new ClassName 创建对象、class.forName 反射对象、clone 克隆对象、序列化反序列化  
worker通过反序列化，得到程序员自己定义的spout和bolt对象。  
  
7. worker根据任务类型，分别执行spout任务或者bolt任务。  
spout的声明周期是：open、nextTuple、outPutFiled  
bolt的生命周期是：prepare、execute(tuple)、outPutFiled

集群架构中的各个模块如何启动？



任务如何分配，如何执行？

