



Storm实时数据分析平台 第1周

DATAGURU专业数据分析社区

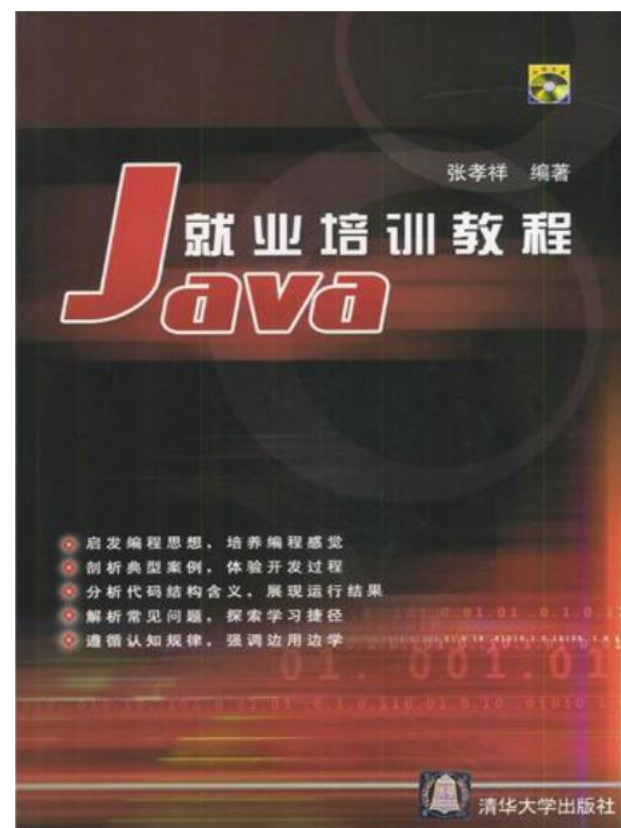
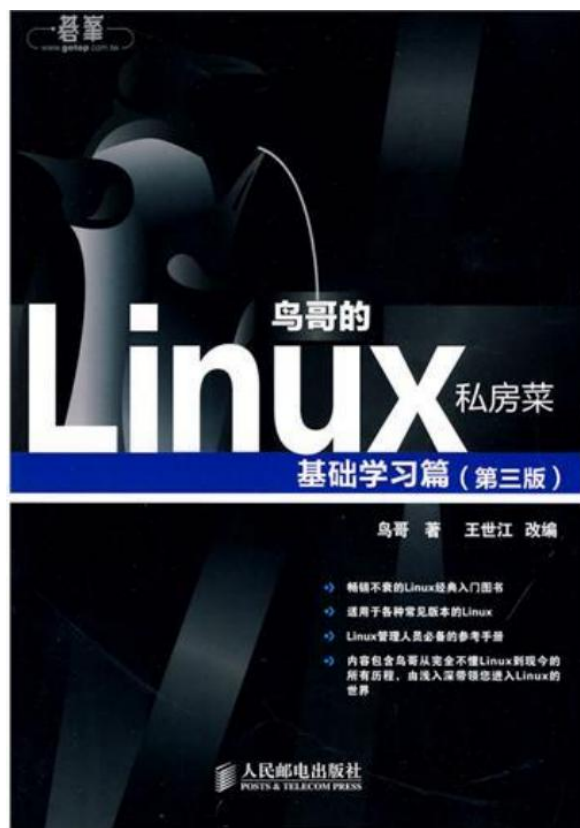
【声明】 本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

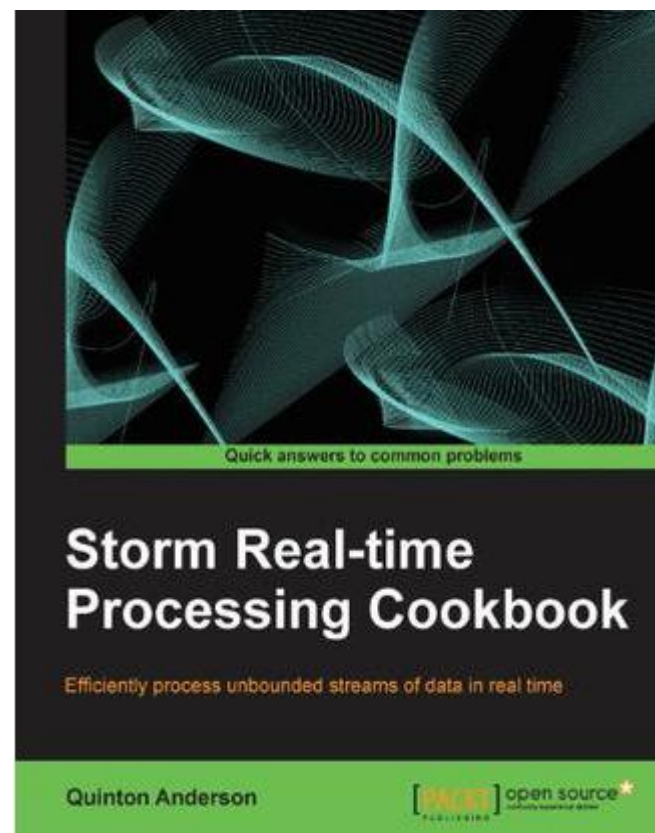
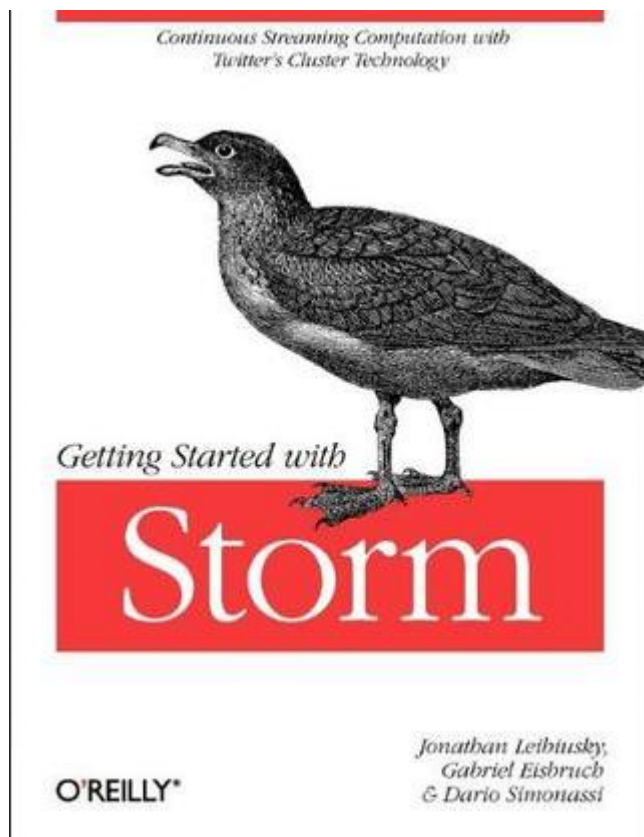
课程详情访问炼数成金培训网站

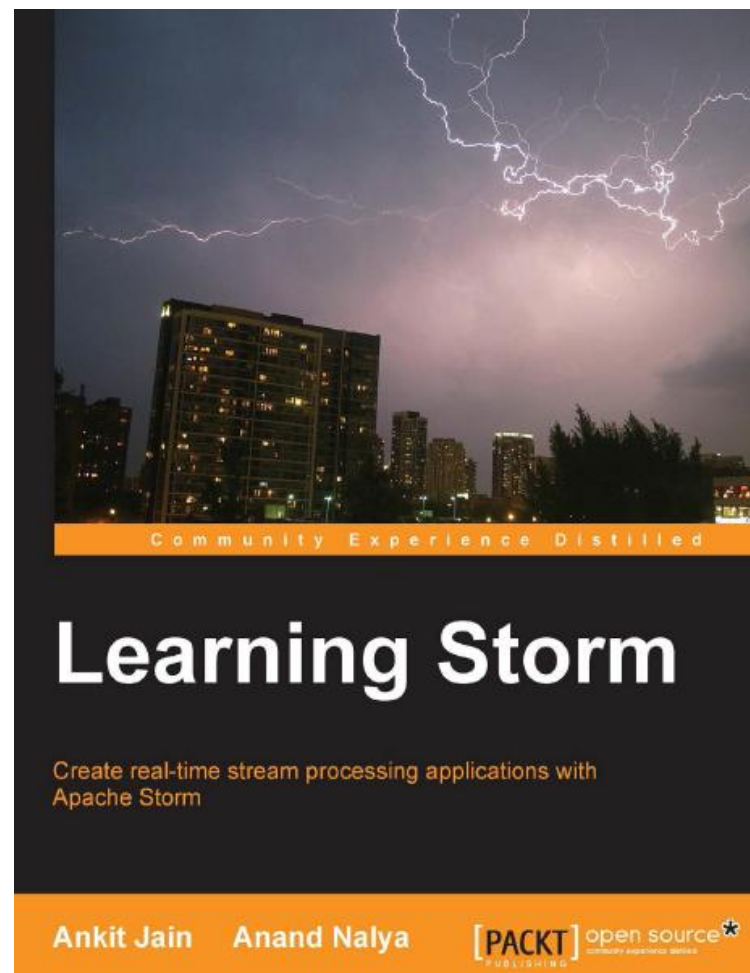
<http://edu.dataguru.cn>

关于本课程的预备知识

- ◆ Linux：懂基本操作
- ◆ Java：能看懂Java程序

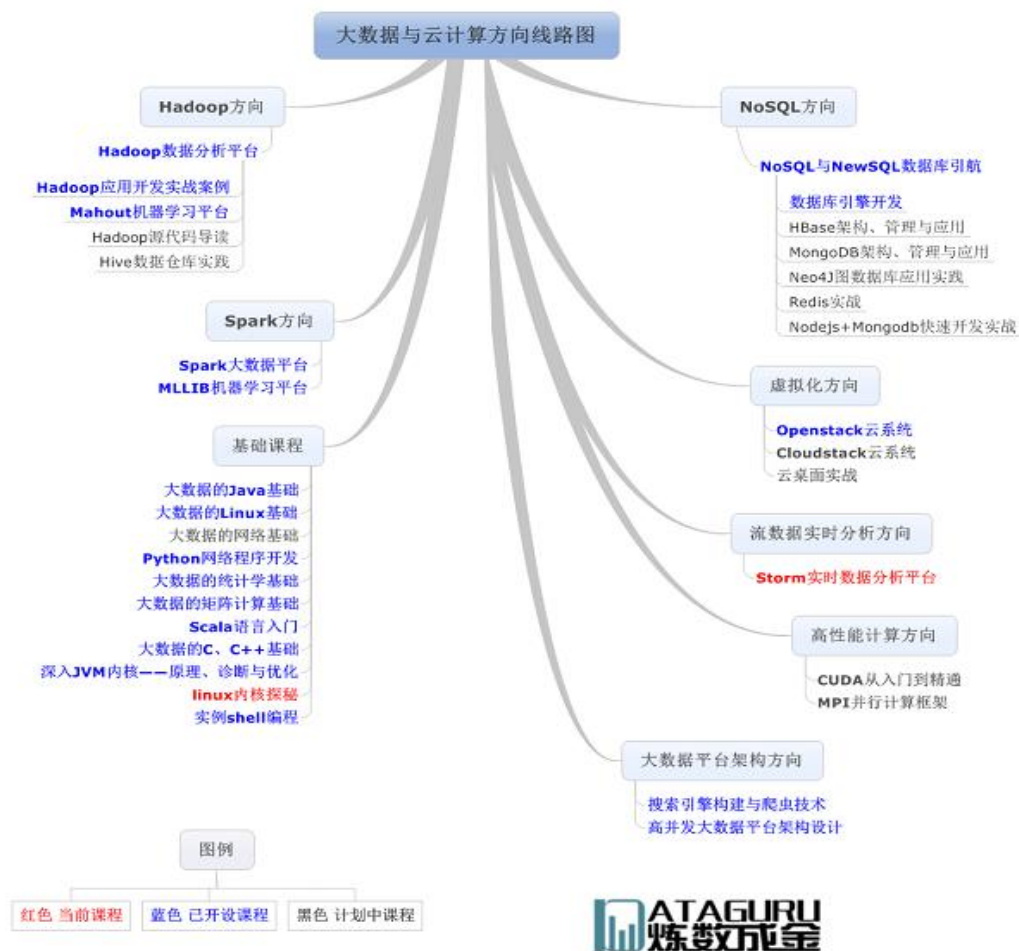






- ◆ 与Hadoop课程类似
- ◆ 三节点，可以使用虚拟机
- ◆ Linux , JVM

大数据平台知识路线图



- ◆ 部署：storm和zookeeper集群
- ◆ 了解storm的架构和基本工作原理
- ◆ 掌握storm的spout,bolts,topology等组件的组成和原理
- ◆ 能书写基本的java程序运行storm的实例，懂得在本地模式下调试，开发topology和提交topology到storm集群
- ◆ 应用Storm进行实时数据分析

- ◆ 互联网从诞生的第一时间起，对世界的最大的改变就是让信息能够实时交互，从而大大加速了各个环节的效率。正因为大家对信息实时响应、实时交互的需求，软件行业除了个人操作系统之外，数据库（更精确的说是关系型数据库）应该是软件行业发展最快、收益最为丰厚的产品了。记得十年前，很多银行别说实时转账，连实时查询都做不到，但是数据库和高速网络改变了这个情况。
- ◆ 随着互联网的更进一步发展，从Portal信息浏览型到Search信息搜索型到SNS关系交互传递型，以及电子商务、互联网旅游生活产品等将生活中的流通环节在线化。对效率的要求让大家对于实时性的要求进一步提升，而信息的交互和沟通正在从点对点往信息链甚至信息网的方向发展，这样必然带来数据在各个维度的交叉关联，数据爆炸已不可避免。因此流式处理加NoSQL产品应运而生，分别解决实时框架和数据大规模存储计算的问题。

流式处理场景：算法交易

Advanced Micro Devices, Inc. (Public, NYSE:AMD) [Watch this stock](#)

6.43 -0.10 (-1.53%)

After Hours: **6.40 -0.03 (-0.47%)**

Nov 16, 7:54PM EST

NYSE real-time data - [Disclaimer](#)

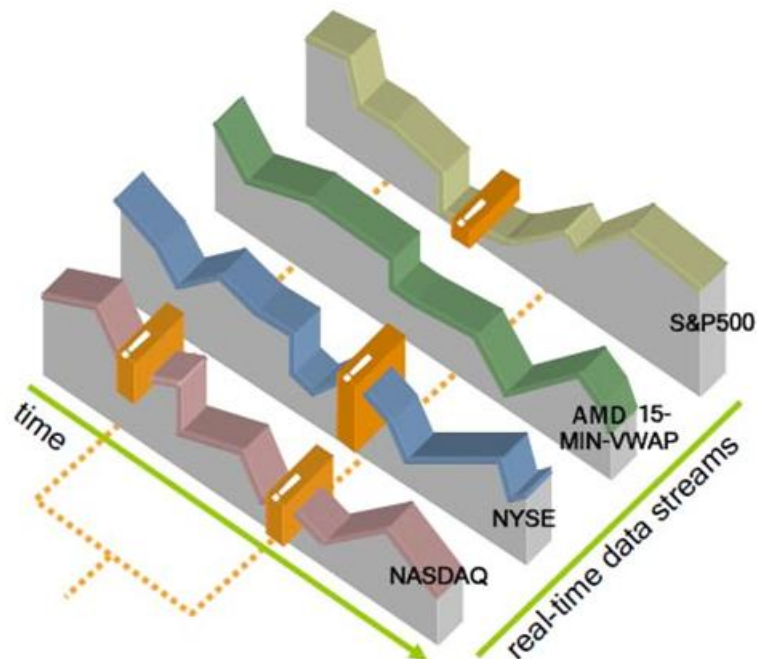
Range	6.32 - 6.70	Mkt cap	4.31B	Shares	670.30M
52 week	1.62 - 6.73	P/E	-	Beta	2.14
Open	6.56	Div/yield	-	Inst. own	46%
Vol / Avg	48.80M/33.67M	EPS	-3.70		



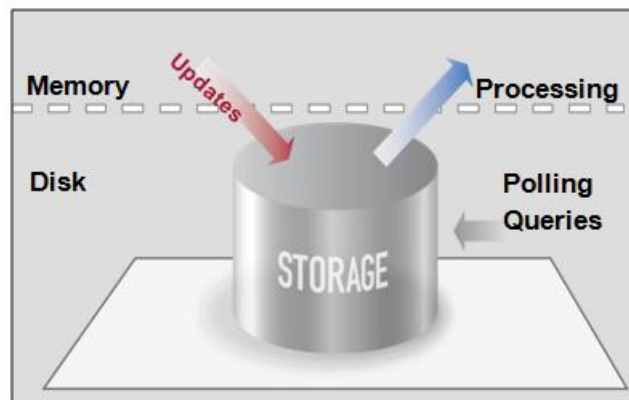
交易规则

```
IF  
  AMD price moves outside 2% of AMD-15-minute-VWAP  
FOLLOWED-BY (  
  S&P moving by 0.5%  
  AND (  
    AMD's price moves up by 5%  
    OR  
    INTEL's price moves down by 2%  
  )  
)  
ALL WITHIN  
  any 2 minute time period  
THEN  
  BUY INTEL  
  SELL AMD
```

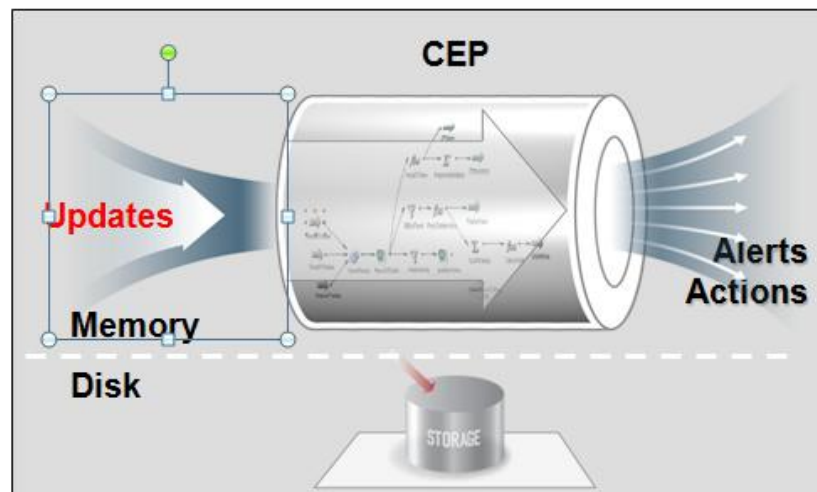
海量数据分析
实时事件处理



关系型数据库不适合流式处理



- 先存储数据，然后查询、处理
- 为业务数据处理而优化



- 随着数据的流动获取、分析数据
- 全新的方法论
 - 把数据送到查询中
 - 只加载极少量数据

优点: 超短延时

- 没有等待
- 实时提交结果

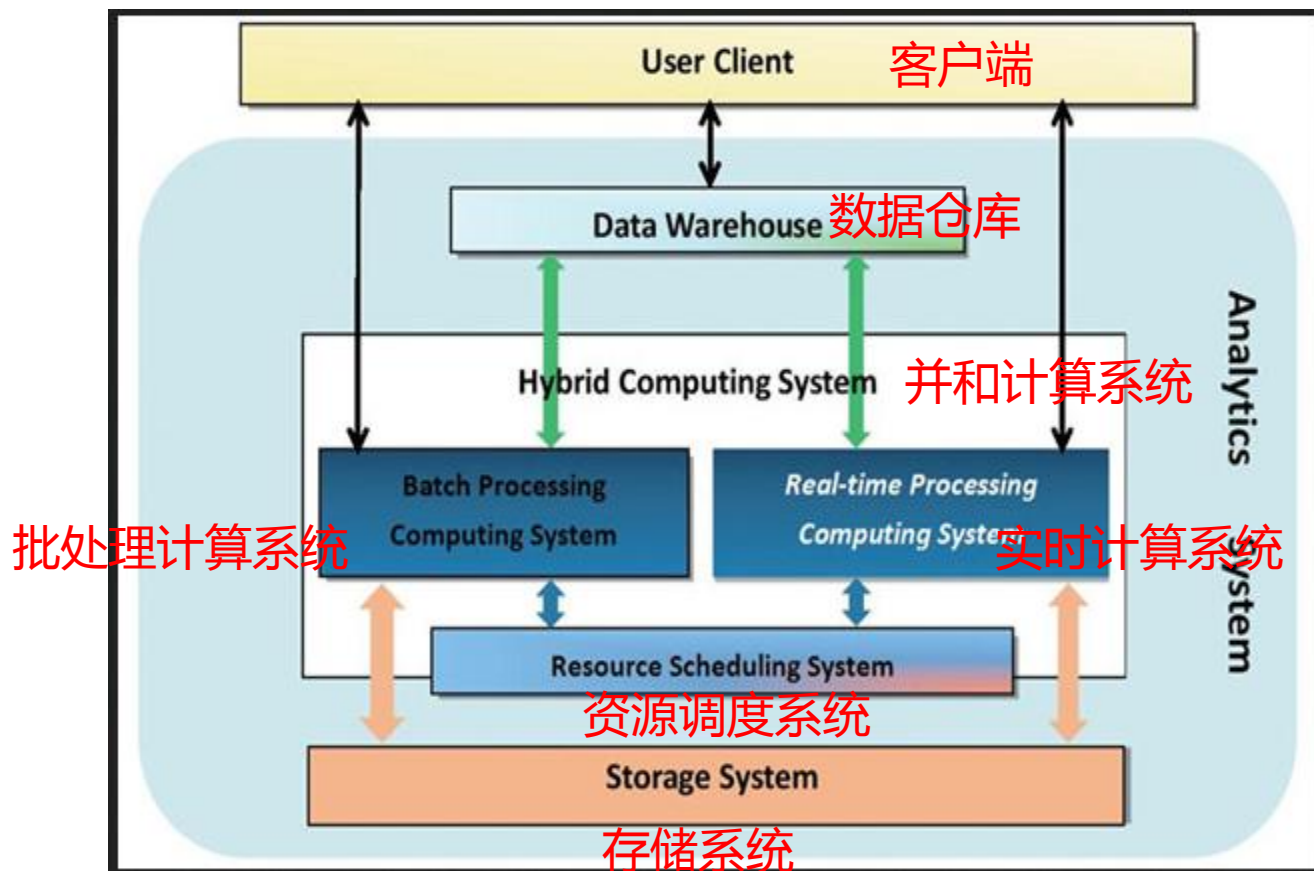
- ◆ 早在7、8年前诸如UC伯克利、斯坦福等大学就开始了对流式数据处理的研究，但是由于更多的关注于金融行业的业务场景或者互联网流量监控的业务场景，以及当时互联网数据场景的限制，造成了研究多是基于对传统数据库处理的流式化，对流式框架本身的研究偏少。目前这样的研究逐渐没有了声音，工业界更多的精力转向了实时数据库。
- ◆ 2010年Yahoo！对S4的开源，2011年twitter对Storm的开源，改变了这个情况。以前互联网的开发人员在做一个实时应用的时候，除了要关注应用逻辑计算处理本身，还要为了数据的实时流转、交互、分布大伤脑筋。但是现在情况却大为不同，以Storm为例，开发人员可以快速的搭建一套健壮、易用的实时流处理框架，配合SQL产品或者NoSQL产品或者MapReduce计算平台，就可以低成本的做出很多以前很难想象的实时产品：比如一淘数据部的量子恒道品牌旗下的多个产品就是构建在实时流处理平台上的

如果只用一句话来描述storm的话，可能会是这样：分布式实时计算系统。按照storm作者的说法，storm对于实时计算的意义类似于hadoop对于批处理的意义。我们都知道，根据google mapreduce来实现的hadoop为我们提供了map, reduce原语，使我们的批处理程序变得非常地简单和优美。同样，storm也为实时计算提供了一些简单优美的原语。我们会在第三节中详细介绍。

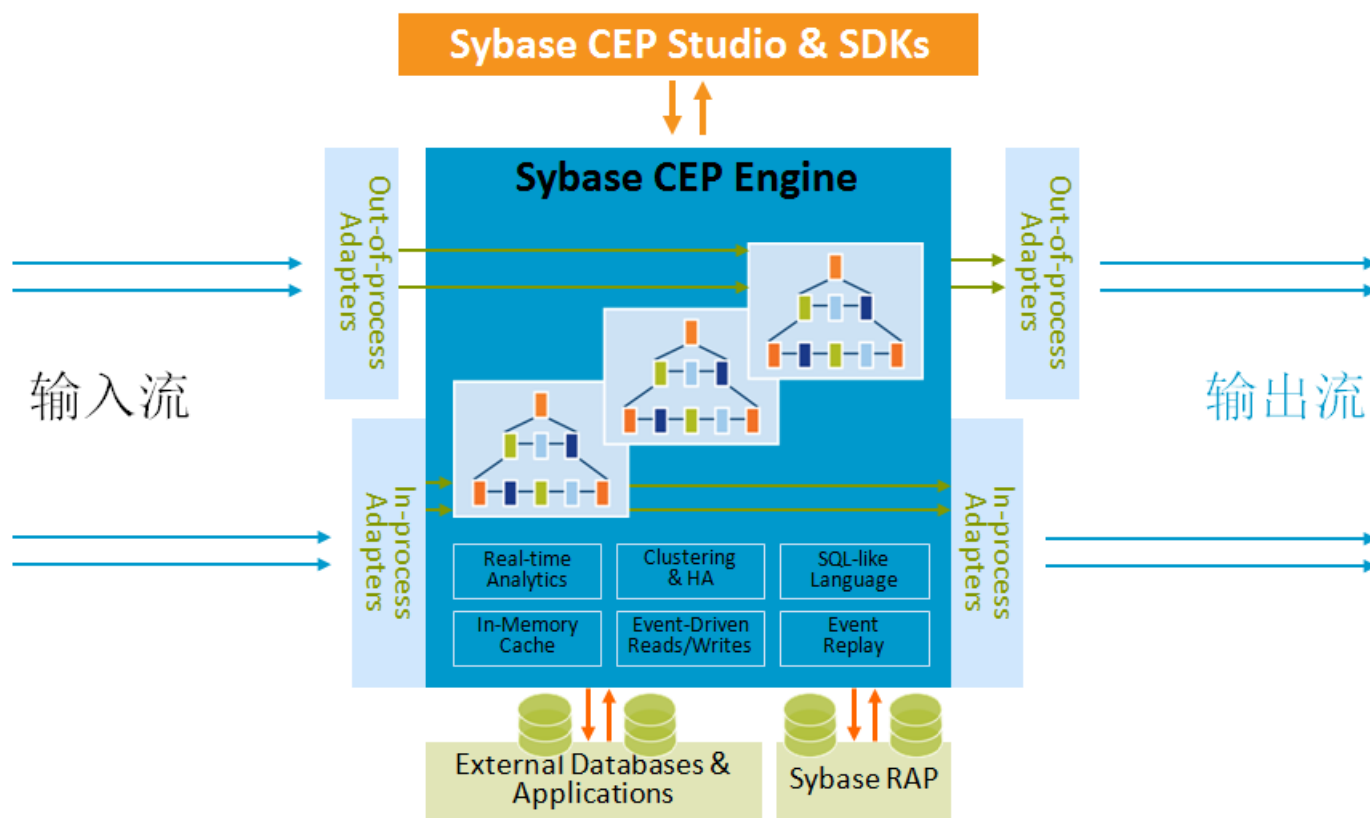
我们来看一下storm的适用场景。

- 1、流数据处理。Storm可以用来处理源源不断流进来的消息，处理之后将结果写入到某个存储中去。
- 2、持续计算。连续发送数据到客户端，使它们能够实时更新并显示结果，如网站指标。
- 3、分布式rpc。由于storm的处理组件是分布式的，而且处理延迟极低，所以可以作为一个通用的分布式rpc框架来使用。当然，其实我们的搜索引擎本身也是一个分布式rpc系统。

数据分析系统的组成



一种流式处理软件架构图



Storm关注的是数据多次处理一次写入，而Hadoop关注的是数据一次写入，多次查询、使用。Storm系统运行起来后是持续不断的，而Hadoop往往只是在业务需要时调用数据

Storm和Hadoop对比

	Hadoop	Storm
系统角色	JobTracker	Nimbus
	TaskTracker	Supervisor
	Child	Worker
应用名称	Job	Topology
组件接口	Mapper/Reducer	Spout/Bolt

- ◆ Storm是一个开源的分布式实时计算系统，可以简单、可靠的处理大量的数据流。
Storm有很多使用场景：如实时分析，在线机器学习，持续计算，分布式RPC，ETL等等。Storm支持水平扩展，具有高容错性，保证每个消息都会得到处理，而且处理速度很快（在一个小集群中，每个结点每秒可以处理数以百万计的消息）。Storm的部署和运维都很便捷，而且更为重要的是可以使用任意编程语言来开发应用。

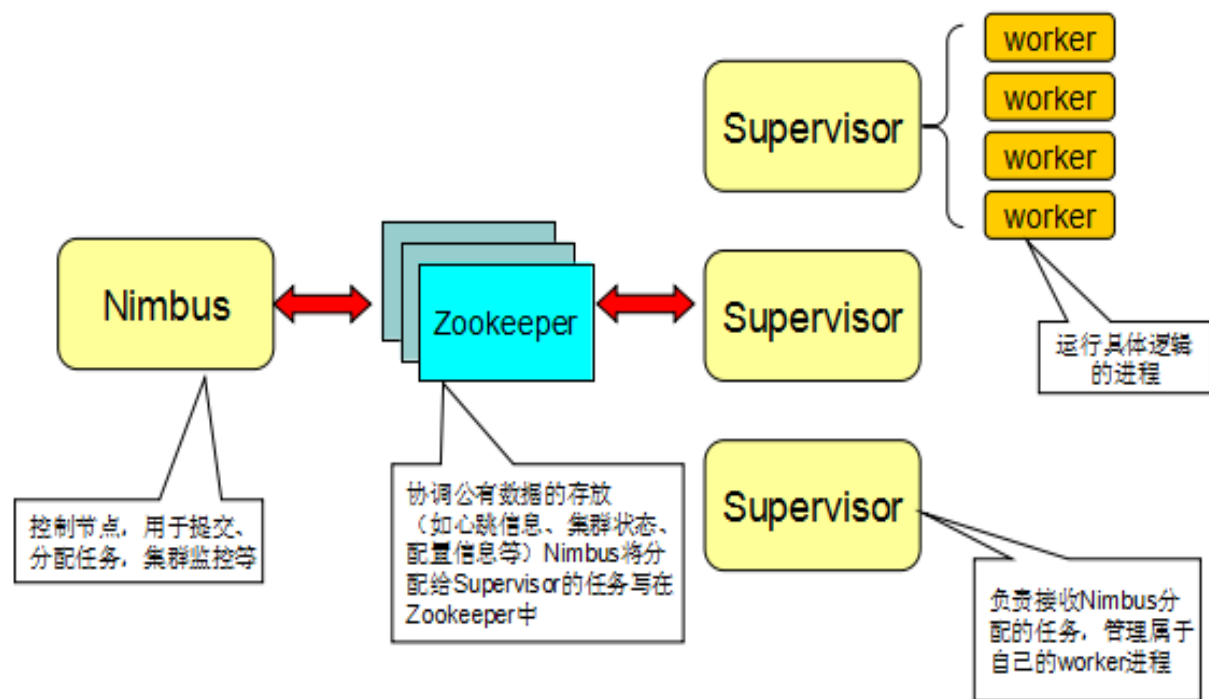
Storm有如下特点：

- ◆ 编程模型简单
- ◆ 可扩展
- ◆ 高可靠性
- ◆ 高容错性
- ◆ 支持多种编程语言

本地模式

远程模式

Storm集群架构:



◆ Nimbus主节点：

主节点通常运行一个后台程序 —— Nimbus，用于响应分布在集群中的节点，分配任务和监测故障。这个很类似于Hadoop中的Job Tracker。

◆ Supervisor工作节点：

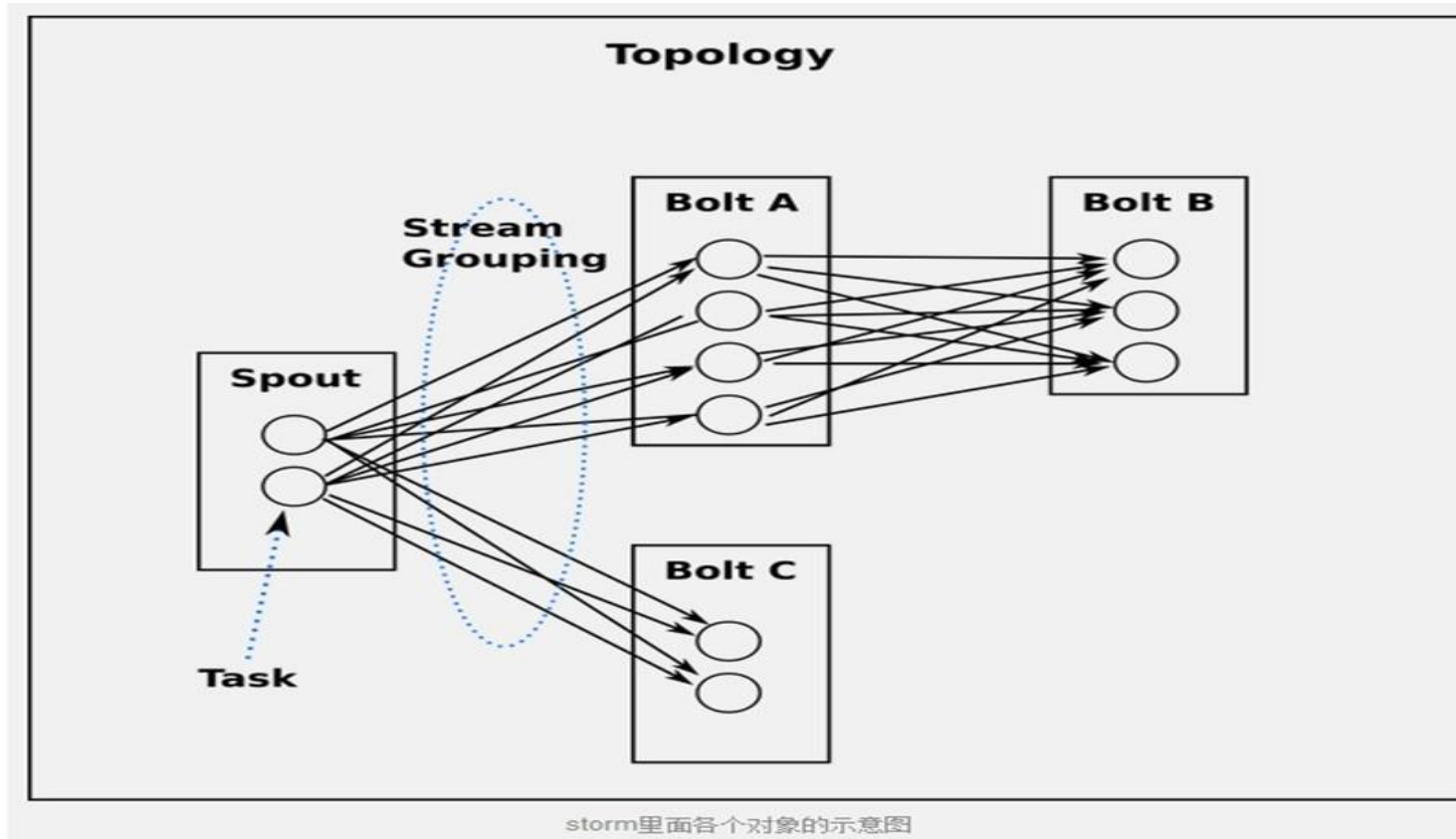
工作节点同样会运行一个后台程序 —— Supervisor，用于收听工作指派并基于要求运行工作进程。每个工作节点都是topology中一个子集的实现。而Nimbus和Supervisor之间的协调则通过Zookeeper系统或者集群。

◆ Zookeeper

Zookeeper是完成Supervisor和Nimbus之间协调的服务。而应用程序实现实时的逻辑则被封装进Storm中的“topology”。topology则是一组由Spouts（数据源）和Bolts（数据操作）通过Stream Groupings进行连接的图。下面对出现的术语进行更深刻的解析。

Topology (拓扑)

storm中运行的一个实时应用程序，因为各个组件间的消息流动形成逻辑上的一个拓扑结构。一个topology是spouts和bolts组成的图，通过stream groupings将图中的spouts和bolts连接起来，如下图：



- ◆ 一个topology会一直运行直到你手动kill掉，Storm自动重新分配执行失败的任务，并且Storm可以保证你不会有数据丢失（如果开启了高可靠性的话）。如果一些机器意外停机它上面的所有任务会被转移到其他机器上。
- ◆ 运行一个topology很简单。首先，把你所有的代码以及所依赖的jar打进一个jar包。然后运行类似下面的这个命令：
- ◆ `storm jar all-my-code.jar backtype.storm.MyTopology arg1 arg2`
- ◆ 这个命令会运行主类: `backtype.storm.MyTopology`, 参数是`arg1`, `arg2`。这个类的`main`函数定义这个topology并且把它提交给Nimbus。storm jar负责连接到Nimbus并且上传jar包。
- ◆ Topology的定义是一个Thrift结构，并且Nimbus就是一个Thrift服务，你可以提交由任何语言创建的topology。上面的方面是用JVM-based语言提交的最简单的方法。

- ◆ storm使用tuple来作为它的数据模型。每个tuple是一堆值，每个值有一个名字，并且每个值可以是任何类型，在我的理解里面一个tuple可以看作一个没有方法的java对象。总体来看，storm支持所有的基本类型、字符串以及字节数组作为tuple的值类型。你也可以使用你自己定义的类型来作为值类型，只要你实现对应的序列化器(serializer)。
- ◆ 一个Tuple代表数据流中的一个基本的处理单元，例如一条cookie日志，它可以包含多个Field，每个Field表示一个属性。



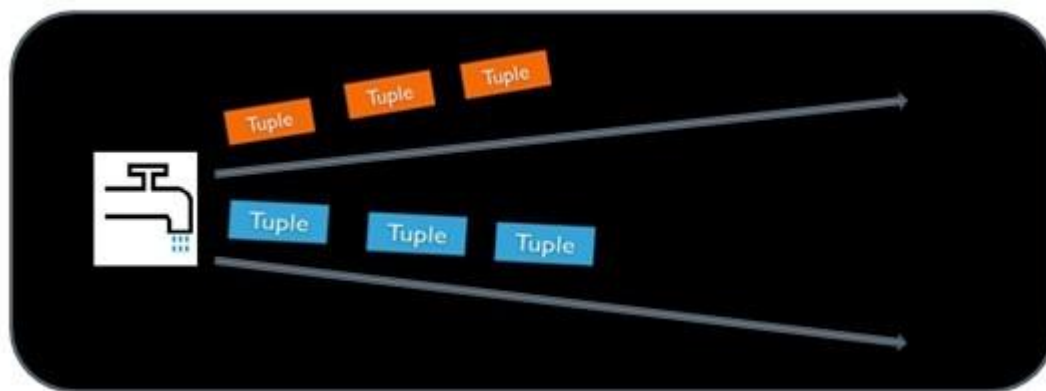
- ◆ Tuple本来应该是一个Key-Value的Map，由于各个组件间传递的tuple的字段名称已经事先定义好了，所以Tuple只需要按序填入各个Value，所以就是一个Value List。



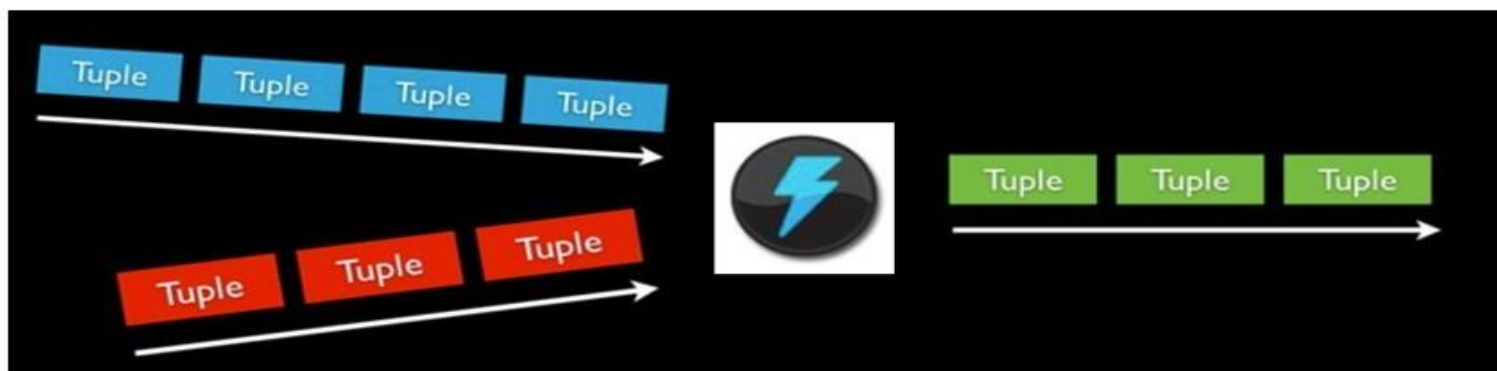
- ◆ 一个没有边界的、源源不断的、连续的Tuple序列就组成了Stream。

- ◆ Stream是storm里面的关键抽象。一个stream是一个没有边界的tuple序列。storm提供一些原语来分布式地、可靠地把一个stream传输进一个新的stream。比如：你可以把一个tweets流传输到热门话题的流。
- ◆ storm提供的最基本的处理stream的原语是spout和bolt。你可以实现Spout和Bolt对应的接口以处理你的应用的逻辑

- ◆ 消息源spout是Storm里面一个topology里面的消息生产者,简而言之, Spout从来源处读取数据并放入topology。spout是流的源头。比如一个spout可能从Kestrel队列里面读取消息并且把这些消息发射成一个流。又比如一个spout可以调用twitter的一个api并且把返回的tweets发射成一个流。
- ◆ 通常Spout会从外部数据源（队列、数据库等）读取数据，然后封装成Tuple形式，之后发送到Stream中。Spout是一个主动的角色，在接口内部有个nextTuple函数，Storm框架会不停的调用该函数



- ◆ Topology中所有的处理都由Bolt完成。即所有的消息处理逻辑被封装在bolts里面。Bolt可以完成任何事，比如：连接的过滤、聚合、访问文件/数据库、等等。
- ◆ Bolt处理输入的Stream，并产生新的输出Stream。Bolt可以执行过滤、函数操作、Join、操作数据库等任何操作。Bolt是一个被动的角色，其接口中有一个execute(Tuple input)方法，在接收到消息之后会调用此函数，用户可以在此方法中执行自己的处理逻辑。
- ◆ Bolt从Spout中接收数据并进行处理，如果遇到复杂流的处理也可能将tuple发送给另一个Bolt进行处理。即需要经过很多bolts。比如算出一堆图片里面被转发最多的图片就至少需要两步：第一步算出每个图片的转发数量。第二步找出转发最多的前10个图片。（如果要把这个过程做得更具有扩展性那么可能需要更多的步骤）。



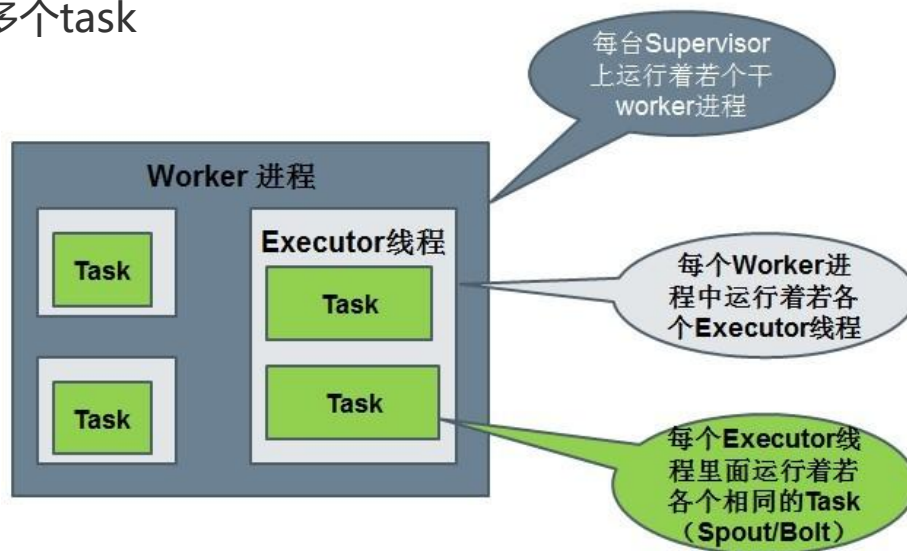
- ◆ Stream Grouping定义了一个流在Bolt任务间该如何被切分。这里有storm提供的7个Stream Grouping类型：
- ◆ 1) 随机分组 (Shuffle grouping) : 随机分发tuple到Bolt的任务，保证每个任务获得相等数量的tuple。
- ◆ 2) 字段分组 (Fields grouping) : 根据指定字段分割数据流，并分组。例如，根据 “user-id” 字段，相同 “user-id” 的元组总是分发到同一个任务，不同 “user-id” 的元组可能分发到不同的任务。
- ◆ 3) 全部分组 (All grouping) : tuple被复制到bolt的所有任务。这种类型需要谨慎使用。
- ◆ 4) 全局分组 (Global grouping) : 全部流都分配到bolt的同一个任务。明确地说，是分配给ID最小的那个task。
- ◆ 5) 无分组 (None grouping) : 你不需要关心流是如何分组。目前，无分组等效于随机分组。但最终，Storm将把无分组的Bolts放到Bolts或Spouts订阅它们的同一线程去执行（如果可能）。
- ◆ 6) 直接分组 (Direct grouping) : 这是一个特别的分组类型。元组生产者决定tuple由哪个元组处理者任务接收。
- ◆ 7) 本地或随机分组 (Local or shuffle grouping) : 如果目标bolt有一个或多个任务在同一个的worker进程中，tuples会随机发送给这些任务。否则，就和普通的随机分组一样。
- ◆ 当然还可以实现CustomStreamGrouping接口来定制自己需要的分组。

运行中的Topology的组件

- ◆ 运行中的Topology主要由以下三个组件组成的：
- ◆ Worker processes (进程)
- ◆ Executors (threads) (线程)
- ◆ Tasks

运行中的Topology的组件

- ◆ Worker：运行具体处理组件逻辑的进程。
- ◆ 一个Topology可能会在一个或者多个Worker(工作进程)里面执行，每个worker是一个物理JVM并且执行整个Topology的一部分。storm会尽量均匀地将工作分配给所有的worker
- ◆ executor:每个execuator对应一个线程。1个executor是1个worker进程生成的1个线程。它可能运行着1个相同的组件(spout或bolt)的1个或多个task
- ◆ Task:每个Spout或者Bolt 会被当做很多task在整个集群中运行，在executor这个线程中运行一个或多个task



<http://www.ibm.com/developerworks/cn/opensource/os-cn-zookeeper/>

<http://agapple.iteye.com/blog/1111377>

<http://zookeeper.apache.org/doc/r3.3.2/zookeeperOver.html>

- ◆ 用来解决分布式应用中经常遇到的一些数据管理问题，如：

统一命名空间(Name Service)

配置推送 (Watch)

集群管理(Group membership)

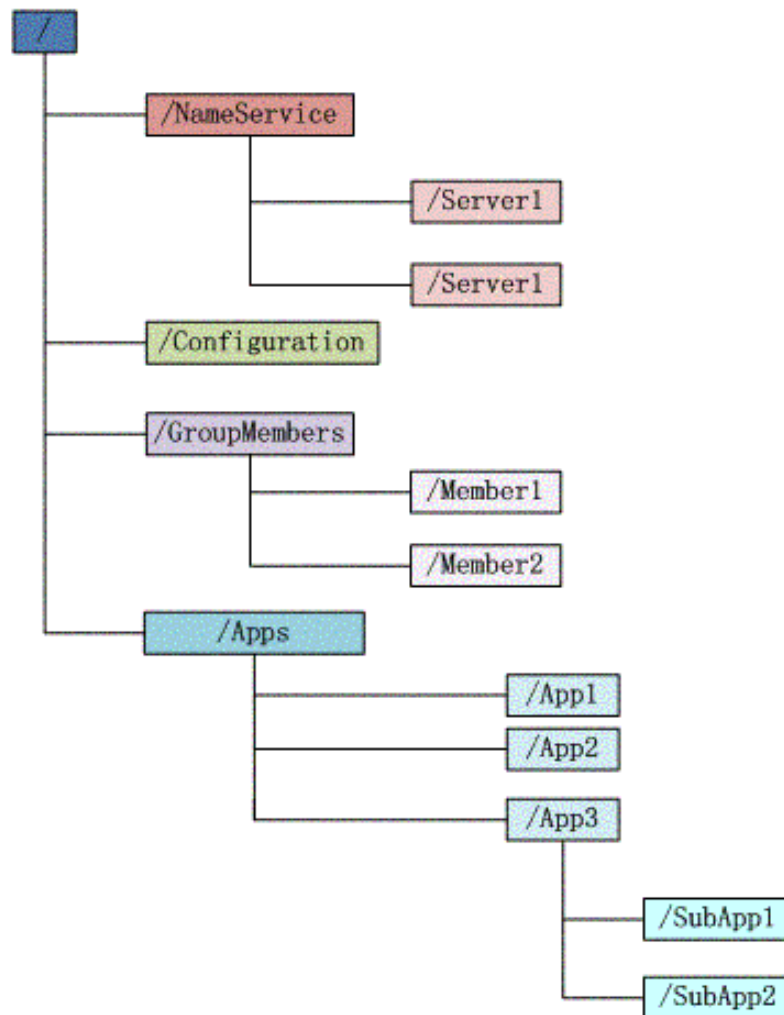
- ◆ 使用zookeeper提供分布式锁机制，从而实现分布式的一致性处理。典型的几个场景：

Barrier

Queue

Lock

2PC



- ◆ 每个子目录项如 NameService 都被称作为 znode，这个 znode 是被它所在的路径唯一标识，如 Server1 这个 znode 的标识为 /NameService/Server1
- ◆ znode 可以有子节点目录，并且每个 znode 可以存储数据，注意 EPHEMERAL 类型的目录节点不能有子节点目录
- ◆ znode 是有版本的，每个 znode 中存储的数据可以有多个版本，也就是一个访问路径中可以存储多份数据
- ◆ znode 可以是临时节点，一旦创建这个 znode 的客户端与服务器失去联系，这个 znode 也将自动删除，Zookeeper 的客户端和服务端通信采用长连接方式，每个客户端和服务端通过心跳来保持连接，这个连接状态称为 session，如果 znode 是临时节点，这个 session 失效，znode 也就删除了
- ◆ znode 的目录名可以自动编号，如 App1 已经存在，再创建的话，将会自动命名为 App2
- ◆ znode 可以被监控，包括这个目录节点中存储的数据的修改，子节点目录的变化等，一旦变化可以通知设置监控的客户端，这个是 Zookeeper 的核心特性，Zookeeper 的很多功能都是基于这个特性实现的

ZooKeeper 基本的操作示例

// 创建一个与服务器的连接

```
ZooKeeper zk = new ZooKeeper("localhost:" + CLIENT_PORT,  
    ClientBase.CONNECTION_TIMEOUT, new Watcher() {  
        // 监控所有被触发的事件  
        public void process(WatchedEvent event) {  
            System.out.println("已经触发了" + event.getType() + "事件！");  
        }  
    });
```

// 创建一个目录节点

```
zk.create("/testRootPath", "testRootData".getBytes(), Ids.OPEN_ACL_UNSAFE,  
    CreateMode.PERSISTENT);
```

ZooKeeper 基本的操作示例

// 创建一个子目录节点

```
zk.create("/testRootPath/testChildPathOne", "testChildDataOne".getBytes(),  
    Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
```

```
System.out.println(new String(zk.getData("/testRootPath", false, null)));
```

// 取出子目录节点列表

```
System.out.println(zk.getChildren("/testRootPath", true));
```

// 修改子目录节点数据

```
zk.setData("/testRootPath/testChildPathOne", "modifyChildDataOne".getBytes(),  
    1);
```

```
System.out.println("目录节点状态 : [" + zk.exists("/testRootPath", true) + "]);
```

ZooKeeper 基本的操作示例

// 创建另外一个子目录节点

```
zk.create("/testRootPath/testChildPathTwo", "testChildDataTwo".getBytes(),  
    Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
```

```
System.out.println(new  
    String(zk.getData("/testRootPath/testChildPathTwo", true, null)));
```

// 删除子目录节点

```
zk.delete("/testRootPath/testChildPathTwo", -1);
```

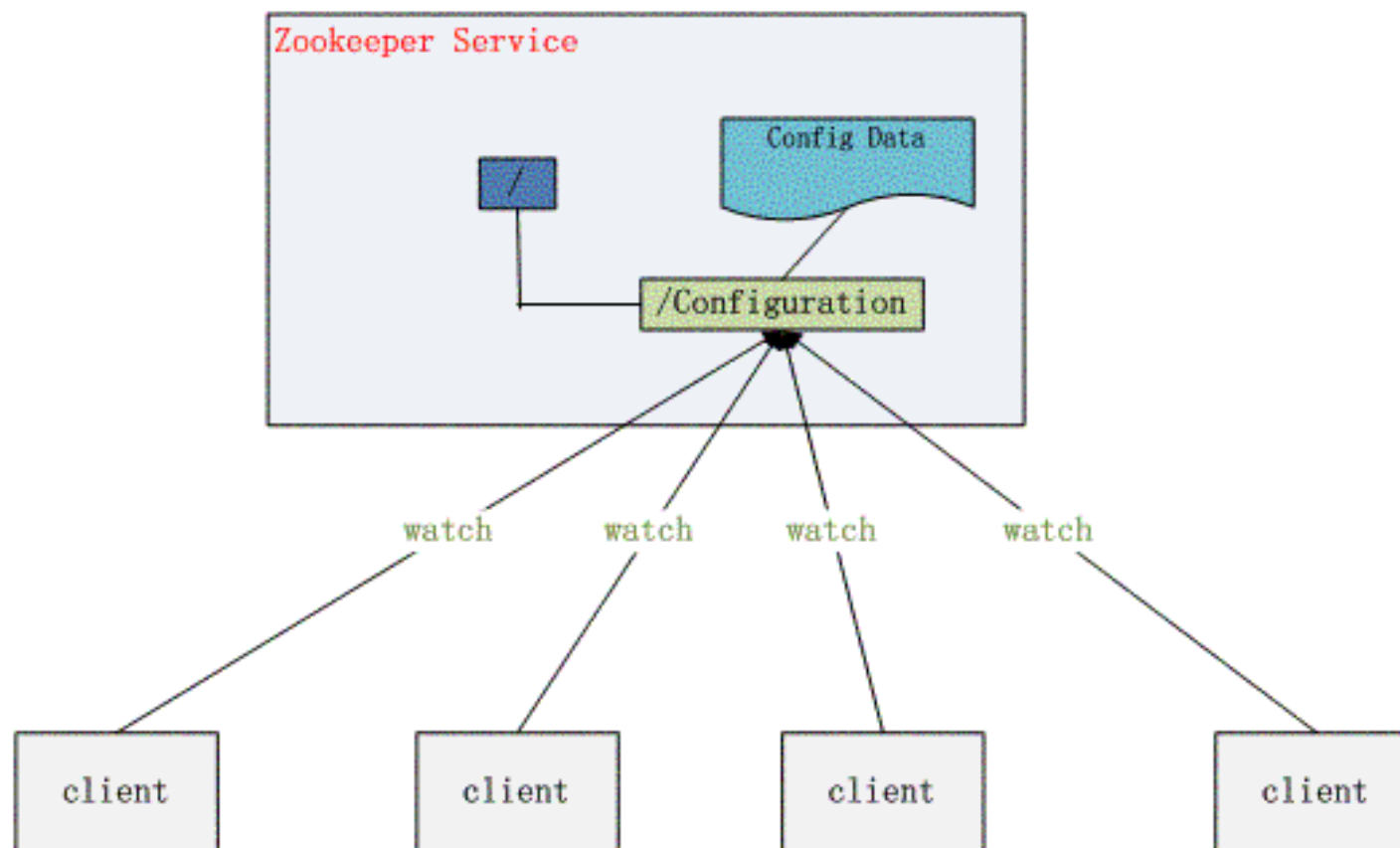
```
zk.delete("/testRootPath/testChildPathOne", -1);
```

// 删除父目录节点

```
zk.delete("/testRootPath", -1);
```

// 关闭连接

```
zk.close();
```

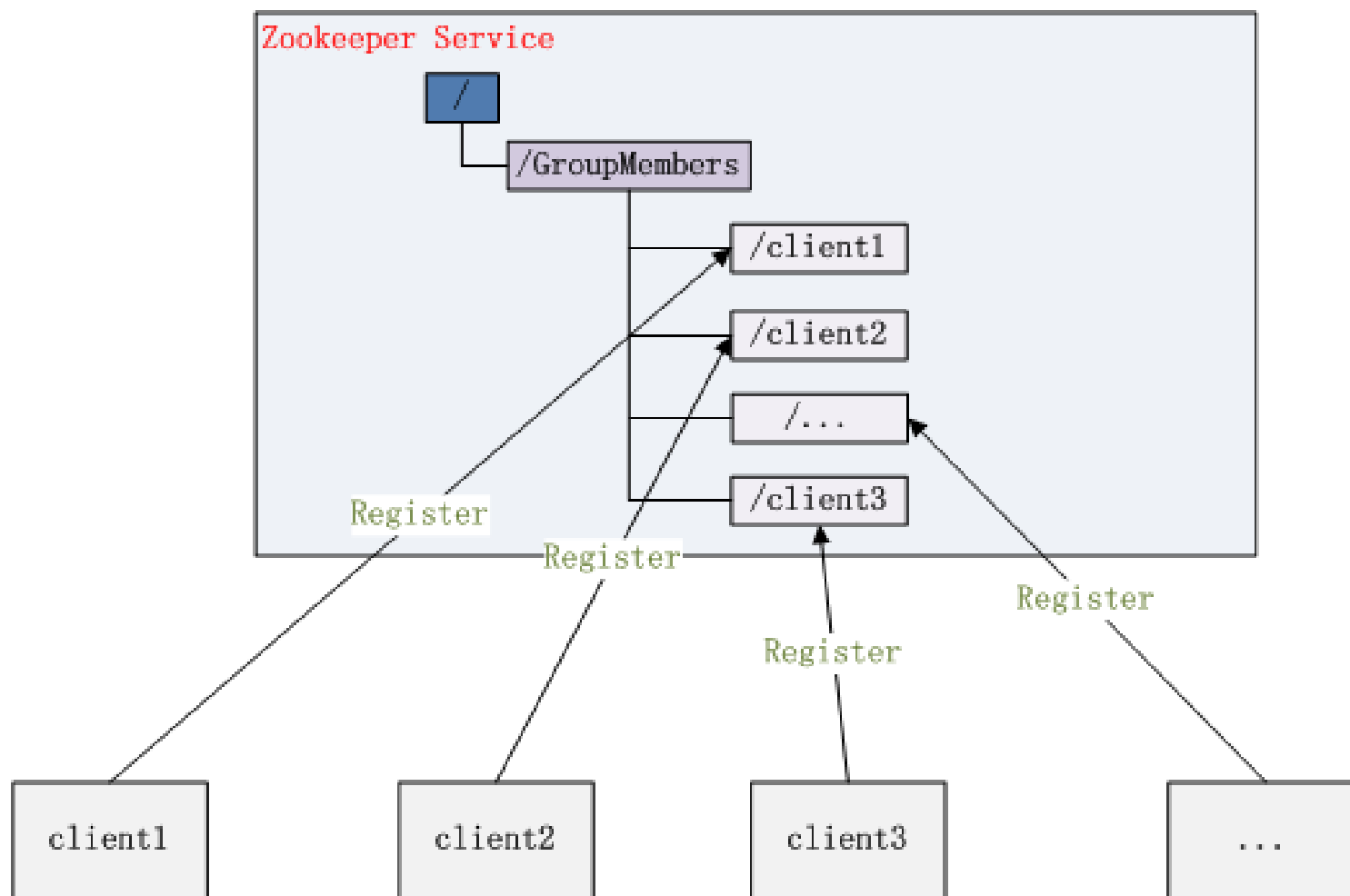


Zookeeper 能够很容易的实现集群管理的功能，如有多台 Server 组成一个服务集群，那么必须要一个“总管”知道当前集群中每台机器的服务状态，一旦有机器不能提供服务，集群中其它集群必须知道，从而做出调整重新分配服务策略。同样当增加集群的服务能力时，就会增加一台或多台 Server，同样也必须让“总管”知道。

Zookeeper 不仅能够帮你维护当前的集群中机器的服务状态，而且能够帮你选出一个“总管”，让这个总管来管理集群，这就是 Zookeeper 的另一个功能 Leader Election。

它们的实现方式都是在 Zookeeper 上创建一个 EPHEMERAL 类型的目录节点，然后每个 Server 在它们创建目录节点的父目录节点上调用 `getChildren(String path, boolean watch)` 方法并设置 `watch` 为 `true`，由于是 EPHEMERAL 目录节点，当创建它的 Server 死去，这个目录节点也随之被删除，所以 Children 将会变化，这时 `getChildren` 上的 Watch 将会被调用，所以其它 Server 就知道已经有某台 Server 死去了。新增 Server 也是同样的原理。

Zookeeper 如何实现 Leader Election，也就是选出一个 Master Server。和前面的一样每台 Server 创建一个 EPHEMERAL 目录节点，不同的是它还是一个 SEQUENTIAL 目录节点，所以它是个 EPHEMERAL_SEQUENTIAL 目录节点。之所以它是 EPHEMERAL_SEQUENTIAL 目录节点，是因为我们可以给每台 Server 编号，我们可以选择当前是最小编号的 Server 为 Master，假如这个最小编号的 Server 死去，由于是 EPHEMERAL 节点，死去的 Server 对应的节点也被删除，所以当前的节点列表中又出现一个最小编号的节点，我们就选择这个节点为当前 Master。这样就实现了动态选择 Master，避免了传统意义上单 Master 容易出现单点故障的问题。



Zookeeper在分布式系统中的应用

- ◆ 实现共享锁
- ◆ 实现FIFO队列
- ◆ 参考：<http://www.ibm.com/developerworks/cn/opensource/os-cn-zookeeper/>

- ◆ 服务器：ESXi，可以在上面部署多台虚拟机，能同时启动3台
- ◆ PC：要求linux环境，linux可以是standalone或者使用虚拟机
- ◆ SSH：windows下可以使用SecureCRT或putty等ssh client程序，作用是用来远程连接linux服务器，linux下可以直接使用ssh命令
- ◆ Vmware client：用于管理ESXi

- ◆ 至少4G内存，最好运行64位windows系统，因为32位xp只能支持3G多的内存
- ◆ 安装vmware workstation或virtual box
- ◆ 部署3台虚拟机，能同时运行，虚拟网络配置为网桥方式
- ◆ 安装linux和java,python

- ◆ 安装虚拟机和linux，虚拟机推荐使用vbox或vmware，PC可以使用workstation，服务器可以使用ESXi，在管理上比较方便。可以使用复制虚拟机功能简化准备流程。如果只是实验用途，内存分配可以在1G左右，硬盘大约预留20-30G空间即可。
- ◆ 以Centos为例，分区可以选择默认，安装选项选择Desktop Gnome，以及Server、Server GUI即可。其它Linux，注意选项里应包括ssh，vi（用于编辑配置文件）等
- ◆ 到Oracle官网下载java jdk安装包
- ◆ 安装Linux后一定要确认iptables,selinux等防火墙访问控制机制已经关闭，否则实验很可能受影响

- ◆ 1. 搭建Zookeeper集群；
- ◆ 2. 安装Storm依赖库；
- ◆ 3. 下载并解压Storm发布版本；
- ◆ 4. 修改storm.yaml配置文件；
- ◆ 5. 启动Storm各个后台进程。

- ◆ 关闭防火墙
- ◆ 修改/etc/hosts 文件
- ◆ 添加修改集群中主机和IP的映射关系

```
[root@user7 src]# vim /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.0.110 user5.hadoop.com
192.168.0.111 user6.hadoop.com
192.168.0.112 user7.hadoop.com
```

```
~
~
~
~
```


- ◆ 下载并安装jdk 6(或以上版本)
- ◆ 配置JAVA_HOME , CLASSPATH环境变量
- ◆ 运行java -version 查看java 版本

```
[root@user7 src]# java -version
java version "1.7.0_60-ea"
Java(TM) SE Runtime Environment (build 1.7.0_60-ea-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.60-b09, mixed mode)
[root@user7 src]#
```

- ◆ 每台机器上都要安装java

搭建Zookeeper集群

- ◆ 每台机器安装完java后，下载zookeeper:
- ◆ wget <http://mirror.bit.edu.cn/apache/zookeeper/stable/zookeeper-3.4.6.tar.gz>
- ◆ 解压：tar -zxf zookeeper-3.4.6.tar.gz

```
[grid@user5 ~]$ ls
ddd  hadoop-2.3.0  hadoop-2.3.0.tar.gz  test  test.txt  zookeeper-3.4.6.tar.gz
[grid@user5 ~]$ tar zxf zookeeper-3.4.6.tar.gz
[grid@user5 ~]$ ls
ddd  hadoop-2.3.0  hadoop-2.3.0.tar.gz  test  test.txt  zookeeper-3.4.6  zookeeper-3.4.6.tar.gz
[grid@user5 ~]$ cd zookeeper-3.4.6
[grid@user5 zookeeper-3.4.6]$ ls
bin          CHANGES.txt  contrib      docs          ivy.xml      LICENSE.txt  README_packaging.txt  recipes  zookeeper-3.4.6.jar  zookeeper-3.4.6.jar.md5
build.xml    conf          dist-maven   ivysettings.xml  lib          NOTICE.txt  README.txt            src      zookeeper-3.4.6.jar.asc  zookeeper-3.4.6.jar.sha1
```

- ◆ 配置zookeeper-3.4.6/conf/zoo.cfg文件
- ◆ cp -p zoo_sample.cfg zoo.cfg

```
[grid@user5 zookeeper-3.4.6]$ cd conf/
[grid@user5 conf]$ ls
configuration.xml  log4j.properties  zoo_sample.cfg
[grid@user5 conf]$ cp -p zoo_sample.cfg zoo.cfg
[grid@user5 conf]$ ls
configuration.xml  log4j.properties  zoo.cfg  zoo_sample.cfg
[grid@user5 conf]$
```

配置zoo.cfg

```
[grid@user5 conf]$ vim zoo.cfg
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sake.
dataDir=/home/grid/zookeeper-3.4.6/data
# the port at which the clients will connect
clientPort=2181
server.1=user5.hadoop.com:2888:3888
server.2=user6.hadoop.com:2888:3888
server.3=user7.hadoop.com:2888:3888

# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
```

分发到其余两个节点

- ◆ 要搭建一个3（奇数）个节点的zookeeper集群,将zookeeper 复制发送到其他两个节点
- ◆ `scp -rp zookeeper-3.4.6 grid@user6.hadoop.com:/home/grid`
- ◆ `scp -rp zookeeper-3.4.6 grid@user7.hadoop.com:/home/grid`

- ◆ 在dataDir (/home/grid/zookeeper-3.4.6/data) 中创建一个文件myid
- ◆ 因为server.1= user5.hadoop.com:2888:3888 server指定的是1,所以在user5.hadoop.com的机器上 : echo "1" > myid
- ◆ 其余两台机器配置 , user6.hadoop.com下面的myid是2 , user7.hadoop.com下面myid是3

```
[grid@user5 data]$ ls  
myid  
[grid@user5 data]$ pwd  
/home/grid/zookeeper-3.4.6/data  
[grid@user5 data]$ cat myid  
1
```

```
[grid@user6 data]$ pwd  
/home/grid/zookeeper-3.4.6/data  
[grid@user6 data]$ cat myid  
2
```

```
[grid@user7 data]$ pwd  
/home/grid/zookeeper-3.4.6/data  
[grid@user7 data]$ cat myid  
3
```

启动zookeeper

- ◆ 启动zookeeper:\$ZOOKEEPER_HOME/bin/zkServer.sh start

```
[grid@user5 zookeeper-3.4.6]$ bin/zkServer.sh start
JMX enabled by default
Using config: /home/grid/zookeeper-3.4.6/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[grid@user5 zookeeper-3.4.6]$ bin/zkServer.sh status
JMX enabled by default
Using config: /home/grid/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: follower
[grid@user5 zookeeper-3.4.6]$
```

```
[grid@user5 zookeeper-3.4.6]$ jps
3007 QuorumPeerMain
3060 Jps
[grid@user5 zookeeper-3.4.6]$
```

- ◆ bin/zkServer.sh status 在不同的机器上使用该命令，其中二台显示follower，一台显示leader

```
[grid@user7 zookeeper-3.4.6]$ bin/zkServer.sh status
JMX enabled by default
Using config: /home/grid/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: leader
[grid@user7 zookeeper-3.4.6]$
```

```
[grid@user6 zookeeper-3.4.6]$ bin/zkServer.sh status
JMX enabled by default
Using config: /home/grid/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: follower
[grid@user6 zookeeper-3.4.6]$
```

- ◆ （先确定你系统自带的Python版本，如果是2.6.6或者之上的不需要安装）
- ◆ 查看python版本：python -V（推荐安装2.6.6版本的python）
- ◆ wget <https://www.python.org/ftp/python/2.6.6/Python-2.6.6.tgz>
- ◆ ./configure
- ◆ make
- ◆ sudo make install

```
[root@user7 src]# pwd
/usr/src
[root@user7 src]# wget https://www.python.org/ftp/python/2.6.6/Python-2.6.6.tgz
--2014-09-18 21:58:17-- https://www.python.org/ftp/python/2.6.6/Python-2.6.6.tgz
Resolving www.python.org... 103.245.222.223
Connecting to www.python.org|103.245.222.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13318547 (13M) [application/octet-stream]
Saving to: 'Python-2.6.6.tgz'

100%[=====>] 13,318,547 111K/s in 74s

2014-09-18 21:59:32 (176 KB/s) - 'Python-2.6.6.tgz' saved [13318547/13318547]

[root@user7 src]# tar -zxf Python-2.6.6.tgz
[root@user7 src]# ls
cmake-3.0.1  cmake-3.0.1.tar.gz  debug  kernels  mysql-5.5.8  Python-2.6.6  Python-2.6.6.tgz
[root@user7 src]#
```

下载并解压storm

- ◆ 下载storm:
- ◆ wget <http://mirror.bit.edu.cn/apache/incubator/storm/apache-storm-0.9.1-incubating/apache-storm-0.9.1-incubating.tar.gz>
- ◆ 解压 : tar -zxf apache-storm-0.9.1-incubating.tar.gz
- ◆ 修改conf/storm.yaml 文件

```
##### These MUST be filled in for a storm configuration
storm.zookeeper.servers:
- "192.168.0.110"
- "192.168.0.111"
- "192.168.0.112"
#
nimbus.host: "192.168.0.110"
storm.zookeeper.port: 2181
storm.local.dir: "/home/grid/storm-0.9.1/data"
supervisor.slots.ports:
- 6700
- 6701
- 6702
- 6703
#
#
```

- ◆ 有关的其他配置项可以参看 :
<https://github.com/nathanmarz/storm/blob/master/conf/defaults.yaml>

storm.yaml配置项的简单说明

- ◆ storm.zookeeper.servers:Storm 集群使用的Zookeeper集群地址
- ◆ nimbus.host:Storm 集群的Nimbus机器的地址，各个Supervisor工作节点需要知道哪个机器是Nimbus，以便下载Topologies的jars、confs等文件
- ◆ storm.local.dir：Nimbus和Supervisor进程用于存储少量状态，如jars、confs等的本地磁盘目录，需要提前创建该目录并给以足够的访问权限
- ◆ supervisor.slots.ports: 对于每个Supervisor工作节点，需要配置该工作节点可以运行的worker数量。每个worker占用一个单独的端口用于接收消息，该配置选项 即用于定义哪些端口是可被worker使用的。默认情况下，每个节点上可运行4个workers，分别在6700、6701、6702和6703端口

复制分发到集群中的其他节点

- ◆ 集群中的其他机器要安装JDK,Python
- ◆ 复制storm的安装目录到其他节点：
- ◆ `scp -rp storm-0.9.1/ grid@user6.hadoop.com:/home/grid/`
- ◆ `scp -rp storm-0.9.1/ grid@user7.hadoop.com:/home/grid/`

- ◆ 在此之前先要启动zookeeper
- ◆ 启动storm :
- ◆ Nimbus: 在主节点上运行 `bin/storm nimbus &` 启动Nimbus后台程序, 并放到后台执行
- ◆ Supervisor:在工作节点上运行 `bin/storm supervisor &` 启动Supervisor后台程序, 并放到后台执行;
- ◆ UI:在主节点上运行 `bin/storm ui &`
- ◆ UI启动后可以在浏览器上输入主节点的ip:port (默认8080号端口)
如:`http://192.168.0.110:8080`
- ◆ logviewer:logviewer 在Storm UI通过点击相应的Worker来查看对应的工作日志,在主节点上运行`bin/storm logviewer &`

启动storm

```
[grid@user5 ~]$ storm nimbus &
[1] 1796
[grid@user5 ~]$ Running: java -server -Dstorm.options= -Dstorm.home=/home/grid/storm-0.9.1 -Djava.library.path=/usr/local/lib:/opt/local
/lib:/usr/lib -Dstorm.conf.file= -cp /home/grid/storm-0.9.1/lib/compojure-1.1.3.jar:/home/grid/storm-0.9.1/lib/netty-3.6.3.Final.jar:/ho
me/grid/storm-0.9.1/lib/zookeeper-3.3.3.jar:/home/grid/storm-0.9.1/lib/commons-io-1.4.jar:/home/grid/storm-0.9.1/lib/guava-13.0.jar:/ho
me/grid/storm-0.9.1/lib/servlet-api-2.5-20081211.jar:/home/grid/storm-0.9.1/lib/objenesis-1.2.jar:/home/grid/storm-0.9.1/lib/tools.loggin
g-0.2.3.jar:/home/grid/storm-0.9.1/lib/slf4j-api-1.6.5.jar:/home/grid/storm-0.9.1/lib/commons-codec-1.4.jar:/home/grid/storm-0.9.1/lib/a
sm-4.0.jar:/home/grid/storm-0.9.1/lib/commons-fileupload-1.2.1.jar:/home/grid/storm-0.9.1/lib/commons-exec-1.1.jar:/home/grid/storm-0.9.
1/lib/httpclient-4.1.1.jar:/home/grid/storm-0.9.1/lib/minlog-1.2.jar:/home/grid/storm-0.9.1/lib/clj-time-0.4.1.jar:/home/grid/storm-0.9.
1/lib/ring-core-1.1.5.jar:/home/grid/storm-0.9.1/lib/tools.cli-0.2.2.jar:/home/grid/storm-0.9.1/lib/carbonite-1.3.2.jar:/home/grid/storm
-0.9.1/lib/meat-locker-0.3.1.jar:/home/grid/storm-0.9.1/lib/ring-jetty-adapter-0.3.11.jar:/home/grid/storm-0.9.1/lib/jline-2.11.jar:/hom
e/grid/storm-0.9.1/lib/storm-core-0.9.1-incubating.jar:/home/grid/storm-0.9.1/lib/clout-1.0.1.jar:/home/grid/storm-0.9.1/lib/jetty-util-
6.1.26.jar:/home/grid/storm-0.9.1/lib/json-simple-1.1.jar:/home/grid/storm-0.9.1/lib/logback-core-1.0.6.jar:/home/grid/storm-0.9.1/lib/m
ath.numeric-tower-0.0.1.jar:/home/grid/storm-0.9.1/lib/log4j-over-slf4j-1.6.6.jar:/home/grid/storm-0.9.1/lib/clj-stacktrace-0.2.4.jar:/h
ome/grid/storm-0.9.1/lib/hiccup-0.3.6.jar:/home/grid/storm-0.9.1/lib/ring-devel-0.3.11.jar:/home/grid/storm-0.9.1/lib/commons-logging-1.
1.1.jar:/home/grid/storm-0.9.1/lib/snakeyaml-1.11.jar:/home/grid/storm-0.9.1/lib/tools.macro-0.1.0.jar:/home/grid/storm-0.9.1/lib/junit-
3.8.1.jar:/home/grid/storm-0.9.1/lib/curator-client-1.0.1.jar:/home/grid/storm-0.9.1/lib/kryo-2.17.jar:/home/grid/storm-0.9.1/lib/common
s-lang-2.5.jar:/home/grid/storm-0.9.1/lib/closure-1.4.0.jar:/home/grid/storm-0.9.1/lib/servlet-api-2.5.jar:/home/grid/storm-0.9.1/lib/lo
gback-classic-1.0.6.jar:/home/grid/storm-0.9.1/lib/httpcore-4.1.jar:/home/grid/storm-0.9.1/lib/ring-servlet-0.3.11.jar:/home/grid/storm-
0.9.1/lib/reflectasm-1.07-shaded.jar:/home/grid/storm-0.9.1/lib/jgrapht-core-0.9.0.jar:/home/grid/storm-0.9.1/lib/jetty-6.1.26.jar:/home
/grid/storm-0.9.1/lib/joda-time-2.0.jar:/home/grid/storm-0.9.1/lib/core.incubator-0.1.0.jar:/home/grid/storm-0.9.1/lib/curator-framework
-1.0.1.jar:/home/grid/storm-0.9.1/lib/disruptor-2.10.1.jar:/home/grid/storm-0.9.1/conf -Xmx1024m -Dlogfile.name=nimbus.log -Dlogback.con
figurationFile=/home/grid/storm-0.9.1/logback/cluster.xml backtype.storm.daemon.nimbus
```

```
[grid@user6 ~]$ storm supervisor &
[1] 14881
[grid@user6 ~]$ Running: java -server -Dstorm.options= -Dstorm.home=/home/grid/storm-0.9.1 -Djava.library.path=/usr/local/lib:/opt/local
/lib:/usr/lib -Dstorm.conf.file= -cp /home/grid/storm-0.9.1/lib/commons-io-1.4.jar:/home/grid/storm-0.9.1/lib/jetty-util-6.1.26.jar:/hom
e/grid/storm-0.9.1/lib/carbonite-1.3.2.jar:/home/grid/storm-0.9.1/lib/reflectasm-1.07-shaded.jar:/home/grid/storm-0.9.1/lib/clj-time-0.4
.1.jar:/home/grid/storm-0.9.1/lib/netty-3.6.3.Final.jar:/home/grid/storm-0.9.1/lib/logback-classic-1.0.6.jar:/home/grid/storm-0.9.1/lib/
tools.macro-0.1.0.jar:/home/grid/storm-0.9.1/lib/json-simple-1.1.jar:/home/grid/storm-0.9.1/lib/disruptor-2.10.1.jar:/home/grid/storm-0.
9.1/lib/kryo-2.17.jar:/home/grid/storm-0.9.1/lib/jgrapht-core-0.9.0.jar:/home/grid/storm-0.9.1/lib/curator-client-1.0.1.jar:/home/grid/s
torm-0.9.1/lib/ring-core-1.1.5.jar:/home/grid/storm-0.9.1/lib/commons-lang-2.5.jar:/home/grid/storm-0.9.1/lib/ring-devel-0.3.11.jar:/hom
e/grid/storm-0.9.1/lib/clout-1.0.1.jar:/home/grid/storm-0.9.1/lib/joda-time-2.0.jar:/home/grid/storm-0.9.1/lib/junit-3.8.1.jar:/home/gri
d/storm-0.9.1/lib/closure-1.4.0.jar:/home/grid/storm-0.9.1/lib/hiccup-0.3.6.jar:/home/grid/storm-0.9.1/lib/clj-stacktrace-0.2.4.jar:/hom
e/grid/storm-0.9.1/lib/httpcore-4.1.jar:/home/grid/storm-0.9.1/lib/snakeyaml-1.11.jar:/home/grid/storm-0.9.1/lib/tools.logging-0.2.3.jar
:/home/grid/storm-0.9.1/lib/core.incubator-0.1.0.jar:/home/grid/storm-0.9.1/lib/curator-framework-1.0.1.jar:/home/grid/storm-0.9.1/lib/c
ompojure-1.1.3.jar:/home/grid/storm-0.9.1/lib/minlog-1.2.jar:/home/grid/storm-0.9.1/lib/meat-locker-0.3.1.jar:/home/grid/storm-0.9.1/lib
/ring-jetty-adapter-0.3.11.jar:/home/grid/storm-0.9.1/lib/commons-exec-1.1.jar:/home/grid/storm-0.9.1/lib/storm-core-0.9.1-incubating.ja
r:/home/grid/storm-0.9.1/lib/logback-core-1.0.6.jar:/home/grid/storm-0.9.1/lib/asm-4.0.jar:/home/grid/storm-0.9.1/lib/commons-fileupload
-1.2.1.jar:/home/grid/storm-0.9.1/lib/jline-2.11.jar:/home/grid/storm-0.9.1/lib/objenesis-1.2.jar:/home/grid/storm-0.9.1/lib/jetty-6.1.2
6.jar:/home/grid/storm-0.9.1/lib/math.numeric-tower-0.0.1.jar:/home/grid/storm-0.9.1/lib/httpclient-4.1.1.jar:/home/grid/storm-0.9.1/lib
/commons-codec-1.4.jar:/home/grid/storm-0.9.1/lib/guava-13.0.jar:/home/grid/storm-0.9.1/lib/servlet-api-2.5.jar:/home/grid/storm-0.9.1/l
ib/commons-logging-1.1.1.jar:/home/grid/storm-0.9.1/lib/zookeeper-3.3.3.jar:/home/grid/storm-0.9.1/lib/ring-servlet-0.3.11.jar:/home/gri
d/storm-0.9.1/lib/tools.cli-0.2.2.jar:/home/grid/storm-0.9.1/lib/log4j-over-slf4j-1.6.6.jar:/home/grid/storm-0.9.1/lib/servlet-api-2.5-2
0081211.jar:/home/grid/storm-0.9.1/lib/slf4j-api-1.6.5.jar:/home/grid/storm-0.9.1/conf -Xmx256m -Dlogfile.name=supervisor.log -Dlogback.
configurationFile=/home/grid/storm-0.9.1/logback/cluster.xml backtype.storm.daemon.supervisor
```

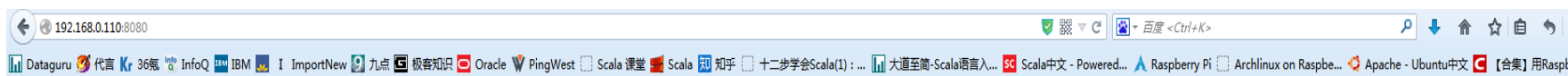
◆ 主节点user5:

```
[grid@user5 ~]$ jps
1942 Jps
1912 supervisor
1838 core
1870 logviewer
1796 nimbus
1678 QuorumPeerMain
[grid@user5 ~]$
```

◆ 工作节点

```
[grid@user7 ~]$ jps
14656 supervisor
14686 Jps
12769 QuorumPeerMain
[grid@user7 ~]$
```

```
[grid@user6 ~]$ jps
4810 QuorumPeerMain
14946 Jps
14881 supervisor
[grid@user6 ~]$
```



Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.9.1-incubating	22m 41s	3	0	12	12	0	0

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
------	----	--------	--------	-------------	---------------	-----------

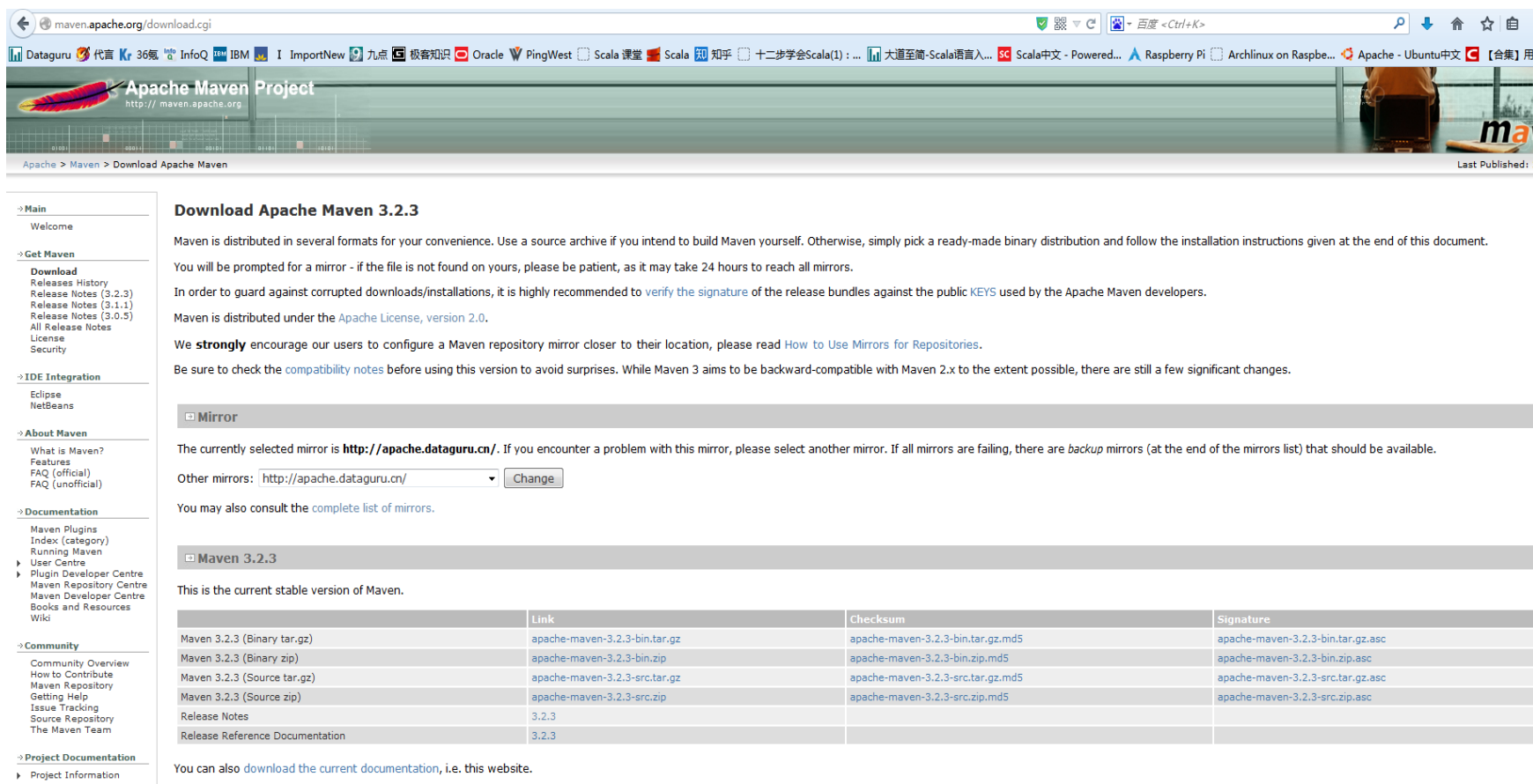
Supervisor summary

Id	Host	Uptime	Slots	Used slots
49ab2175-e6e0-4f3e-a4e7-3d3613614726	user7.hadoop.com	11s	4	0
54a115e9-3478-4f55-9d5a-5c4e92a1372e	user6.hadoop.com	6s	4	0
fb1d7530-7b45-43f0-8e46-7f1fc9450082	user5.hadoop.com	11m 14s	4	0

Nimbus Configuration

测试前的准备：安装maven

◆ 安装maven（最好使用3.x版本）：



Download Apache Maven 3.2.3

Maven is distributed in several formats for your convenience. Use a source archive if you intend to build Maven yourself. Otherwise, simply pick a ready-made binary distribution and follow the installation instructions given at the end of this document.

You will be prompted for a mirror - if the file is not found on yours, please be patient, as it may take 24 hours to reach all mirrors.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

Maven is distributed under the [Apache License, version 2.0](#).

We **strongly** encourage our users to configure a Maven repository mirror closer to their location, please read [How to Use Mirrors for Repositories](#).

Be sure to check the [compatibility notes](#) before using this version to avoid surprises. While Maven 3 aims to be backward-compatible with Maven 2.x to the extent possible, there are still a few significant changes.

Mirror

The currently selected mirror is <http://apache.dataguru.cn/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are *backup* mirrors (at the end of the mirrors list) that should be available.

Other mirrors:

You may also consult the [complete list of mirrors](#).

Maven 3.2.3

This is the current stable version of Maven.

	Link	Checksum	Signature
Maven 3.2.3 (Binary tar.gz)	apache-maven-3.2.3-bin.tar.gz	apache-maven-3.2.3-bin.tar.gz.md5	apache-maven-3.2.3-bin.tar.gz.asc
Maven 3.2.3 (Binary zip)	apache-maven-3.2.3-bin.zip	apache-maven-3.2.3-bin.zip.md5	apache-maven-3.2.3-bin.zip.asc
Maven 3.2.3 (Source tar.gz)	apache-maven-3.2.3-src.tar.gz	apache-maven-3.2.3-src.tar.gz.md5	apache-maven-3.2.3-src.tar.gz.asc
Maven 3.2.3 (Source zip)	apache-maven-3.2.3-src.zip	apache-maven-3.2.3-src.zip.md5	apache-maven-3.2.3-src.zip.asc
Release Notes	3.2.3		
Release Reference Documentation	3.2.3		

You can also [download the current documentation](#), i.e. this website.

测试前的准备：安装maven

◆ mvn -v

```
[grid@user5 ~]$ mvn -v
Apache Maven 3.2.2 (45f7c06d68e745d05611f7fd14efb6594181933e; 2014-06-17T09:51:42-04:00)
Maven home: /usr/local/apache-maven-3.2.2
Java version: 1.7.0_60-ea, vendor: Oracle Corporation
Java home: /usr/jdk1.7.0_60/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "2.6.32-431.23.3.el6.x86_64", arch: "amd64", family: "unix"
[grid@user5 ~]$
```

◆ maven 安装成功

- ◆ yum -y install git
- ◆ 下载测试代码：
- ◆ git clone git://github.com/nathanmarz/storm-starter.git

- ◆ 参考例子网址：
- ◆ <https://github.com/apache/storm/tree/master/examples/storm-starter>

◆ 运行测试例子：在本地模式下编译并运行WordCountTopology：

```
mvn -f m2-pom.xml compile exec:java -Dstorm.topology=storm.starter.WordCountTopology
```

```
[Thread-36] INFO backtype.storm.daemon.task - Emitting: split default ["score"]
[Thread-21-count] INFO backtype.storm.daemon.executor - Processing received message source: split:7, stream: default, id: {core"}
[Thread-21-count] INFO backtype.storm.daemon.task - Emitting: count default [score, 50]
[Thread-36] INFO backtype.storm.daemon.task - Emitting: split default ["and"]
[Thread-23-count] INFO backtype.storm.daemon.executor - Processing received message source: split:7, stream: default, id: {nd"}
[Thread-23-count] INFO backtype.storm.daemon.task - Emitting: count default [and, 108]
[Thread-36] INFO backtype.storm.daemon.task - Emitting: split default ["seven"]
[Thread-19-count] INFO backtype.storm.util - Async loop interrupted!
[Thread-18] INFO backtype.storm.util - Async loop interrupted!
[Thread-12] INFO backtype.storm.event - Event manager interrupted
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Shutting down in process zookeeper
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Done shutting down in process zookee
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Deleting temporary path /tmp/8a58880ecf076
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Deleting temporary path /tmp/df34b09f6278a
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Deleting temporary path /tmp/5050cadfa59dee
[storm.starter.wordCountTopology.main()] INFO backtype.storm.testing - Deleting temporary path /tmp/0531935a75aea
INFO] -----
INFO] BUILD SUCCESS
INFO] -----
INFO] Total time: 47.583 s
INFO] Finished at: 2014-09-19T02:39:38-04:00
INFO] Final Memory: 27M/65M
INFO] -----
```

◆ 打包上传到storm集群：

◆ 打包：mvn -f m2-pom.xml package

```
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.1/plexus-utils-2.0.1.jar (217 KB)
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/maven/ already added, skipping
[INFO] Building jar: /home/grid/storm-starter/target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar
[INFO] META-INF/MANIFEST.MF already added, skipping
[INFO] META-INF/ already added, skipping
[INFO] META-INF/maven/ already added, skipping
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:10 min
[INFO] Finished at: 2014-09-19T02:53:54-04:00
[INFO] Final Memory: 20M/49M
[INFO] -----
grid@user6:storm-starter$
```

◆ 在本地模式下运行RollingTopWords：

◆ storm jar target/storm-starter-*-jar-with-dependencies.jar
storm.starter.RollingTopWords

```
35973 [Thread-42-wordGenerator] INFO backtype.storm.daemon.task - Emitting: wordGenerator default [bertels]
35974 [Thread-20-counter] INFO backtype.storm.daemon.executor - Processing received message source: wordGenerator:15, stream: default,
id: {}, [bertels]
36006 [Thread-34-wordGenerator] INFO backtype.storm.daemon.task - Emitting: wordGenerator default [bertels]
36007 [Thread-20-counter] INFO backtype.storm.daemon.executor - Processing received message source: wordGenerator:11, stream: default,
id: {}, [bertels]
36044 [Thread-36-wordGenerator] INFO backtype.storm.daemon.task - Emitting: wordGenerator default [jackson]
36044 [Thread-20-counter] INFO backtype.storm.daemon.executor - Processing received message source: wordGenerator:12, stream: default,
id: {}, [jackson]
36060 [Thread-38-wordGenerator] INFO backtype.storm.daemon.task - Emitting: wordGenerator default [golda]
36060 [Thread-16-counter] INFO backtype.storm.daemon.executor - Processing received message source: wordGenerator:13, stream: default,
id: {}, [golda]
36060 [Thread-40-wordGenerator] INFO backtype.storm.daemon.task - Emitting: wordGenerator default [mike]
36061 [Thread-18-counter] INFO backtype.storm.daemon.executor - Processing received message source: wordGenerator:14, stream: default,
id: {}, [mike]
```

- ◆ 远程模式下提交topology到集群：
- ◆ storm jar target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.ExclamationTopology ExclamationTopology

```
[grid@user6 storm-starter]$ storm jar target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.ExclamationTopology ExclamationTopology
Running: java -client -Dstorm.options=-Dstorm.home=/home/grid/storm-0.9.1 -Djava.library.path=/usr/local/lib:/opt/local/lib:/usr/lib -Dstorm.conf.file=-cp /home/grid/storm-0.9.1/lib/commons-io-1.4.jar:/home/grid/storm-0.9.1/lib/jetty-util-6.1.26.jar:/home/grid/storm-0.9.1/lib/carbonite-1.3.2.jar:/home/grid/storm-0.9.1/lib/reflectasm-1.07-shaded.jar:/home/grid/storm-0.9.1/lib/clj-time-0.4.1.jar:/home/grid/storm-0.9.1/lib/netty-3.6.3.Final.jar:/home/grid/storm-0.9.1/lib/logback-classic-1.0.6.jar:/home/grid/storm-0.9.1/lib/tools.macro-0.1.0.jar:/home/grid/storm-0.9.1/lib/json-simple-1.1.jar:/home/grid/storm-0.9.1/lib/disruptor-2.10.1.jar:/home/grid/storm-0.9.1/lib/kryo-2.17.jar:/home/grid/storm-0.9.1/lib/jgrapht-core-0.9.0.jar:/home/grid/storm-0.9.1/lib/curator-client-1.0.1.jar:/home/grid/storm-0.9.1/lib/ring-core-1.1.5.jar:/home/grid/storm-0.9.1/lib/commons-lang-2.5.jar:/home/grid/storm-0.9.1/lib/ring-devel-0.3.11.jar:/home/grid/storm-0.9.1/lib/clout-1.0.1.jar:/home/grid/storm-0.9.1/lib/joda-time-2.0.jar:/home/grid/storm-0.9.1/lib/junit-3.8.1.jar:/home/grid/storm-0.9.1/lib/b/closure-1.4.0.jar:/home/grid/storm-0.9.1/lib/hiccup-0.3.6.jar:/home/grid/storm-0.9.1/lib/clj-stacktrace-0.2.4.jar:/home/grid/storm-0.9.1/lib/httpcore-4.1.jar:/home/grid/storm-0.9.1/lib/snakeyaml-1.11.jar:/home/grid/storm-0.9.1/lib/tools.logging-0.2.3.jar:/home/grid/storm-0.9.1/lib/core.incubator-0.1.0.jar:/home/grid/storm-0.9.1/lib/curator-framework-1.0.1.jar:/home/grid/storm-0.9.1/lib/compojure-1.1.3.jar:/home/grid/storm-0.9.1/lib/minlog-1.2.jar:/home/grid/storm-0.9.1/lib/meat-locker-0.3.1.jar:/home/grid/storm-0.9.1/lib/ring-jetty-adapter-0.3.11.jar:/home/grid/storm-0.9.1/lib/commons-exec-1.1.jar:/home/grid/storm-0.9.1/lib/storm-core-0.9.1-incubating.jar:/home/grid/storm-0.9.1/lib/logback-core-1.0.6.jar:/home/grid/storm-0.9.1/lib/asm-4.0.jar:/home/grid/storm-0.9.1/lib/commons-fileupload-1.2.1.jar:/home/grid/storm-0.9.1/lib/jline-2.11.jar:/home/grid/storm-0.9.1/lib/objenesis-1.2.jar:/home/grid/storm-0.9.1/lib/jetty-6.1.26.jar:/home/grid/storm-0.9.1/lib/math.numeric-tower-0.0.1.jar:/home/grid/storm-0.9.1/lib/httpclient-4.1.1.jar:/home/grid/storm-0.9.1/lib/commons-codec-1.4.jar:/home/grid/storm-0.9.1/lib/guava-13.0.jar:/home/grid/storm-0.9.1/lib/servlet-api-2.5.jar:/home/grid/storm-0.9.1/lib/commons-logging-1.1.1.jar:/home/grid/storm-0.9.1/lib/zookeeper-3.3.3.jar:/home/grid/storm-0.9.1/lib/ring-servlet-0.3.11.jar:/home/grid/storm-0.9.1/lib/tools.cli-0.2.2.jar:/home/grid/storm-0.9.1/lib/log4j-over-slf4j-1.6.6.jar:/home/grid/storm-0.9.1/lib/servlet-api-2.5-20081211.jar:/home/grid/storm-0.9.1/lib/slf4j-api-1.6.5.jar:target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar:/home/grid/storm-0.9.1/conf:/home/grid/storm-0.9.1/bin -Dstorm.jar=target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar storm.starter.ExclamationTopology ExclamationTopology
1031 [main] INFO  backtype.storm.StormSubmitter - Jar not uploaded to master yet. Submitting jar...
1191 [main] INFO  backtype.storm.StormSubmitter - Uploading topology jar target/storm-starter-0.0.1-SNAPSHOT-jar-with-dependencies.jar to assigned location: /home/grid/storm-0.9.1/data/nimbus/inbox/stormjar-c96e24a1-595e-471f-a7ce-539ce92a6e80.jar
1457 [main] INFO  backtype.storm.StormSubmitter - Successfully uploaded topology jar to assigned location: /home/grid/storm-0.9.1/data/nimbus/inbox/stormjar-c96e24a1-595e-471f-a7ce-539ce92a6e80.jar
1457 [main] INFO  backtype.storm.StormSubmitter - Submitting topology ExclamationTopology in distributed mode with conf {"topology.workers":3,"topology.debug":true}
2435 [main] INFO  backtype.storm.StormSubmitter - Finished submitting topology: ExclamationTopology
```

查看topology

Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.9.1-incubating	2h 10m 45s	3	3	9	12	18	18

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
ExclamationTopology	ExclamationTopology-1-1411112015	ACTIVE	16s	3	18	18

Supervisor summary

Id	Host	Uptime	Slots	Used slots
49ab2175-e6e0-4f3e-a4e7-3d3613614726	user7.hadoop.com	1h 48m 14s	4	1
54a115e9-3478-4f55-9d5a-5c4e92a1372e	user6.hadoop.com	1h 48m 11s	4	1
fb1d7530-7b45-43f0-8e46-7f1fc9450082	user5.hadoop.com	1h 59m 16s	4	1

Nimbus Configuration

查看topology

Storm UI

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
ExclamationTopology	ExclamationTopology-1-1411112015	ACTIVE	3m 0s	3	18	18

Topology actions

[Activate](#) [Deactivate](#) [Rebalance](#) [Kill](#)

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	44880	29920	0.000	0	0
3h 0m 0s	44880	29920	0.000	0	0
1d 0h 0m 0s	44880	29920	0.000	0	0
All time	44880	29920	0.000	0	0

Spouts (All time)

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Last error
word	10	10	14960	14960	0.000	0	0	

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Last error
exclaim1	3	3	14960	14960	0.024	0.636	14960	0.658	14980	0	
exclaim2	2	2	14960	0	0.013	0.256	14940	0.242	14960	0	

Topology Configuration

0.9.0之前的版本的安装步骤

- ◆ 同样要安装JDK和Python,zookeeper集群
- ◆ 跟前面的配置大致一致，只是多了要安装storm的两个依赖库：ZeroMQ和JZMQ
- ◆ 1. 搭建Zookeeper集群；
- ◆ 2. 安装Storm依赖库（JDK,Python,ZeroMQ,JZMQ）；
- ◆ 3. 下载并解压Storm发布版本；
- ◆ 4. 修改storm.yaml配置文件；
- ◆ 5. 启动Storm各个后台进程。

安装ZeroMQ 2.1.7

- ◆ 安装依赖包 : `yum install gcc-c++ libuuid-devel`
- ◆ 下载 : `wget http://download.zeromq.org/zeromq-2.1.7.tar.gz`
- ◆ 解压 : `tar -zxf zeromq-2.1.7.tar.gz`

```
[root@user7 src]# wget http://download.zeromq.org/zeromq-2.1.7.tar.gz
--2014-09-18 22:54:49-- http://download.zeromq.org/zeromq-2.1.7.tar.gz
Resolving download.zeromq.org... 95.142.169.98
Connecting to download.zeromq.org|95.142.169.98|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1877380 (1.8M) [application/x-gzip]
Saving to: 'zeromq-2.1.7.tar.gz'

100%[=====>] 1,877,380 24.0K/s in 68s

2014-09-18 22:55:58 (27.1 KB/s) - 'zeromq-2.1.7.tar.gz' saved [1877380/1877380]

[root@user7 src]# ls
cmake-3.0.1  cmake-3.0.1.tar.gz  debug  kernels  mysql-5.5.8  Python-2.7.2  Python-2.7.2.tgz  zeromq-2.1.7.tar.gz
[root@user7 src]# tar -zxf zeromq-2.1.7.tar.gz
[root@user7 src]# ls
cmake-3.0.1  cmake-3.0.1.tar.gz  debug  kernels  mysql-5.5.8  Python-2.7.2  Python-2.7.2.tgz  zeromq-2.1.7  zeromq-2.1.7.tar.gz
[root@user7 src]#
```


安装ZeroMQ 2.1.7

◆ ./configure

```
[root@user7 zeromq-2.1.7]# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking how to create a ustar tar archive... gnutar
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
```

◆ make

◆ make install

```
[root@user7 zeromq-2.1.7]# make install
Making install in src
make[1]: Entering directory `/usr/src/zeromq-2.1.7/src'
make[2]: Entering directory `/usr/src/zeromq-2.1.7/src'
test -z "/usr/local/lib" || /bin/mkdir -p "/usr/local/lib"
/bin/sh ../libtool --mode=install /usr/bin/install -c libzmq.la '/usr/local/lib'
libtool: install: /usr/bin/install -c .libs/libzmq.so.1.0.0 /usr/local/lib/libzmq.so.1.0.0
libtool: install: (cd /usr/local/lib && { ln -s -f libzmq.so.1.0.0 libzmq.so.1 || { rm -f libzmq.so.
.1; }; })
libtool: install: (cd /usr/local/lib && { ln -s -f libzmq.so.1.0.0 libzmq.so || { rm -f libzmq.so &
})
libtool: install: /usr/bin/install -c .libs/libzmq.lai /usr/local/lib/libzmq.la
libtool: install: /usr/bin/install -c .libs/libzmq.a /usr/local/lib/libzmq.a
```

编译安装JZMQ

- ◆ `yum -y install git`
- ◆ `git clone git://github.com/nathanmarz/jzmq.git`
- ◆ `cd jzmq`
- ◆ `./autogen.sh`
- ◆ `./configure`
- ◆ `make`
- ◆ `make install`

编译安装JZMQ

```
[root@user7 jzmq]# ./autogen.sh
autoreconf: Entering directory `.'
autoreconf: configure.in: not using Gettext
autoreconf: running: aclocal --force -I config
autoreconf: configure.in: tracing
autoreconf: running: libtoolize --copy --force
libtoolize: putting auxiliary files in AC_CONFIG_AUX_DIR, `config'.
libtoolize: copying file `config/ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIR, `config'.
libtoolize: copying file `config/libtool.m4'
libtoolize: copying file `config/ltoptions.m4'
libtoolize: copying file `config/ltsugar.m4'
libtoolize: copying file `config/ltversion.m4'
libtoolize: copying file `config/lt~obsolete.m4'
autoreconf: running: /usr/bin/autoconf --include=config --force
autoreconf: running: /usr/bin/autoheader --include=config --force
autoreconf: running: automake --add-missing --copy --force-missing
configure.in:31: installing `config/compile'
configure.in:28: installing `config/config.guess'
configure.in:28: installing `config/config.sub'
configure.in:14: installing `config/install-sh'
configure.in:14: installing `config/missing'
src/Makefile.am: installing `config/depcomp'
Makefile.am: installing `./INSTALL'
autoreconf: Leaving directory `.'
[root@user7 jzmq]# ls
aclocal.m4      autom4te.cache  config          COPYING        INSTALL        Makefile.in    pom.xml        src
AUTHORS        builds          configure       COPYING.LESSER jzmq.spec      NEWS           README         test
autogen.sh     ChangeLog      configure.in    debian         Makefile.am    perf           README-PERF
[root@user7 jzmq]# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
```

```
[root@user7 jzmq]# make install
Making install in src
make[1]: Entering directory `/usr/src/jzmq/src'
make install-am
make[2]: Entering directory `/usr/src/jzmq/src'
make[3]: Entering directory `/usr/src/jzmq/src'
test -z "/usr/local/lib" || /bin/mkdir -p "/usr/local/lib"
/bin/sh ./libtool --mode=install /usr/bin/install -c libjzmq.la '/usr/local/lib'
libtool: install: /usr/bin/install -c libs/libjzmq.so.0.0.0 /usr/local/lib/libjzmq.so.0.0.0
libtool: install: (cd /usr/local/lib && { ln -s -f libjzmq.so.0.0.0 libjzmq.so.0 || { rm -f libjzmq.so.0 && ln -s libjzmq.so.0.0.0 libjzmq.so.0; }; })
libtool: install: (cd /usr/local/lib && { ln -s -f libjzmq.so.0.0.0 libjzmq.so || { rm -f libjzmq.so && ln -s libjzmq.so.0.0.0 libjzmq.so; }; })
```

下载并解压Storm

- ◆ `yum -y install unzip`
- ◆ 下载 : `wget https://github.com/downloads/nathanmarz/storm/storm-0.8.1.zip`
- ◆ `unzip storm-0.8.1.zip`
- ◆ `cd storm-0.8.1`
- ◆ 修改`conf/storm.yaml` 文件 (配置时一定要注意在每一项的开始时要加空格, 冒号后也必须要加空格)

```
##### These MUST be filled in for a storm configuration
storm.zookeeper.servers:
  - "192.168.0.110"
  - "192.168.0.111"
  - "192.168.0.112"
#
nimbus.host: "192.168.0.112"
storm.local.dir: "/home/grid/storm-0.8.1/data"
supervisor.slots.ports:
  - 6700
  - 6701
  - 6702
  - 6703
#
##### These may optionally be filled in:
```

- ◆ 在此之前先要启动zookeeper
- ◆ 启动storm :
- ◆ Nimbus: 在Storm主控节点上运行“ bin/storm nimbus >/dev/null 2>&1 &” 启动Nimbus后台程序，并放到后台执行；
- ◆ Supervisor: 在Storm各个工作节点上运行“ bin/storm supervisor>/dev/null 2>&1 &” 启动Supervisor后台程序，并放到后台执行；
- ◆ UI: 在Storm主控节点上运行“ bin/storm ui >/dev/null 2>&1 &” 启动UI后台程序，并放到后台执行

```
[grid@user7 ~]$ jps
13019 supervisor
13088 nimbus
13119 core
13253 Jps
12769 QuorumPeerMain
```

- ◆ **Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。**
- ◆ **关于逆向收费式网络的详情，请看我们的培训网站 <http://edu.dataguru.cn>**



Thanks

FAQ时间