《大数据技术原理与应用》

http://dblab.xmu.edu.cn/post/bigdata

温馨提示：编辑幻灯片母版，可以修改每页PPT的厦大校徽和底部文字

第八章 流计算

（PPT版本号：2016年1月29日版本）

林子雨

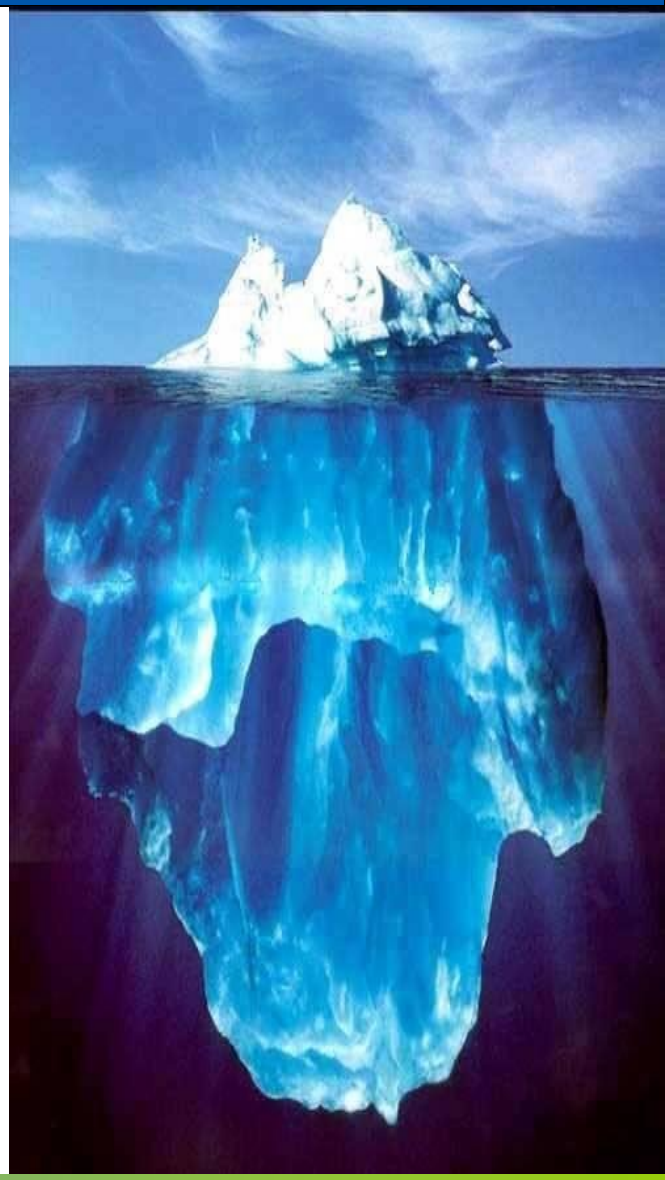
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

主页：**http://www.cs.xmu.edu.cn/linziyu**

提纲

8.1 流计算概述
8.2 流计算处理流程
8.3 流计算应用
8.4 流计算开源框架 – Storm
8.5 Storm安装和运行实例



本PPT是如下教材的配套讲义：
21世纪高等教育计算机规划教材
《大数据技术原理与应用
——概念、存储、处理、分析与应用》
（2015年6月第1版）
厦门大学 林子雨 编著，人民邮电出版社
ISBN:978-7-115-39287-9

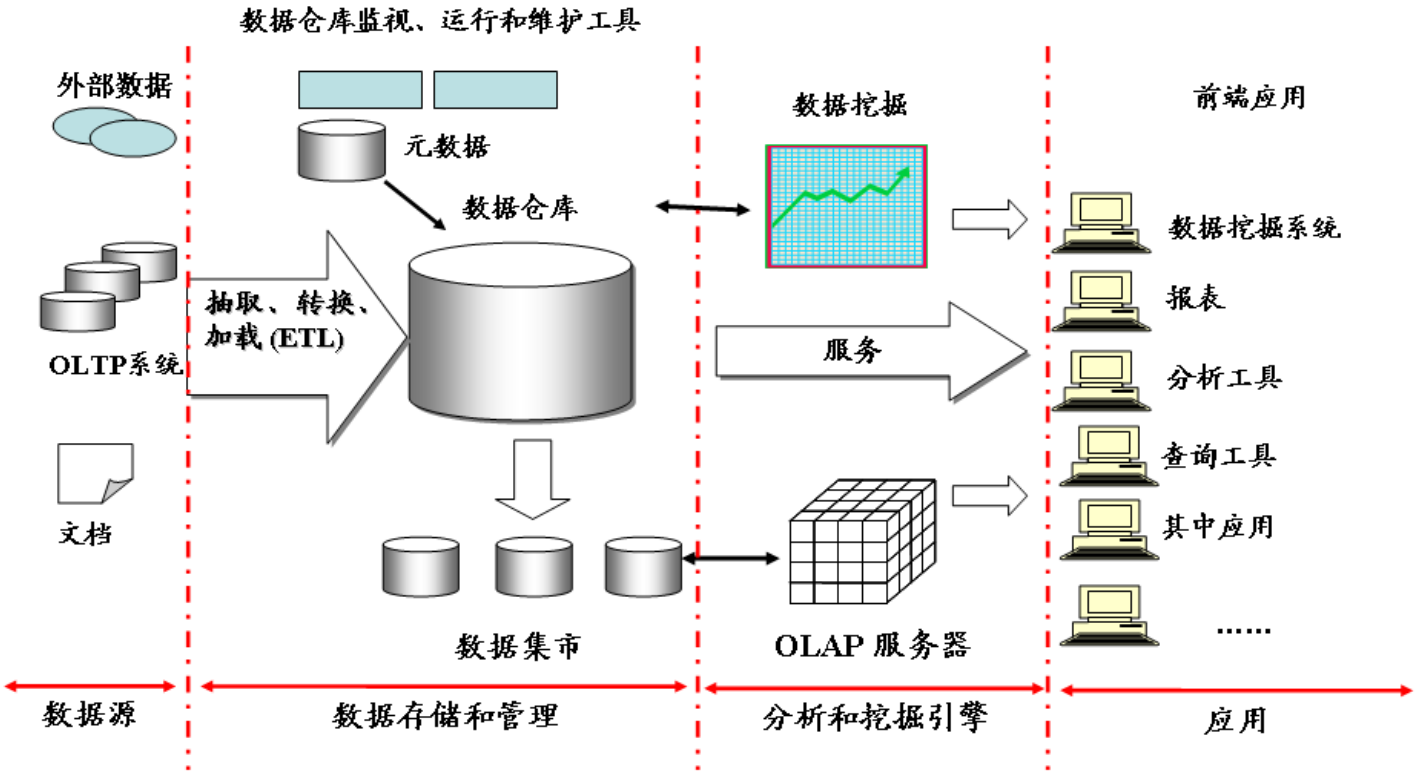
欢迎访问《大数据技术原理与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

8.1 流计算概述

- 8.1.1 静态数据和流数据
- 8.1.2 批量计算和实时计算
- 8.1.3 流计算概念
- 8.1.4 流计算与Hadoop
- 8.1.5 流计算框架

8.1.1 静态数据和流数据

- 很多企业为了支持决策分析而构建的数据仓库系统，其中存放的大量历史数据就是静态数据。技术人员可以利用数据挖掘和OLAP（On-Line Analytical Processing）分析工具从静态数据中找到对企业有价值的信息

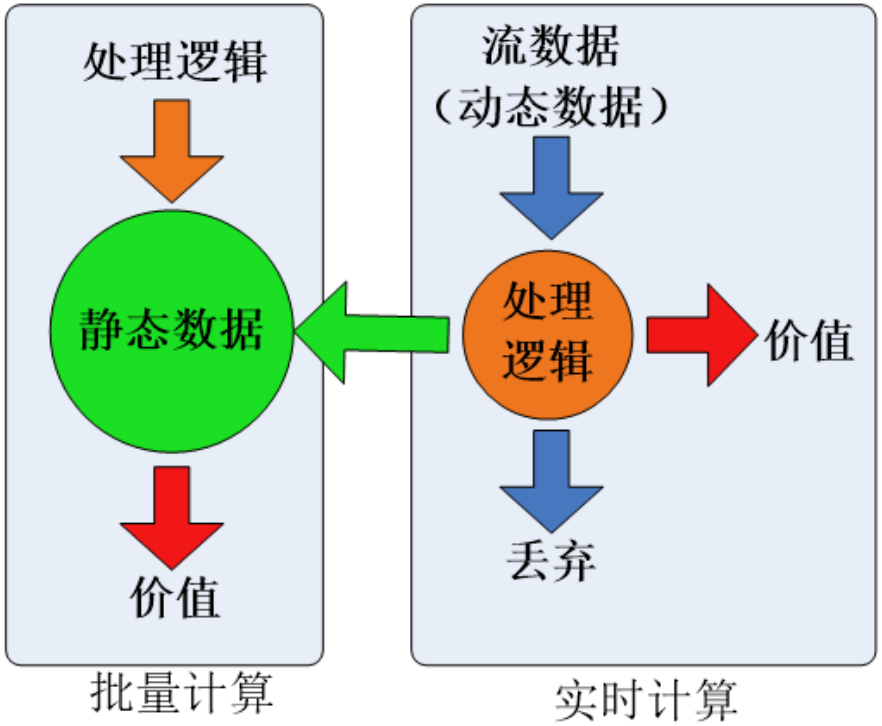


8.1.1 静态数据和流数据

- 近年来，在Web应用、网络监控、传感监测等领域，兴起了一种新的数据密集型应用——流数据，即数据以大量、快速、时变的流形式持续到达
- 流数据具有如下特征：
  - 数据快速持续到达，潜在大小也许是无穷无尽的
  - 数据来源众多，格式复杂
  - 数据量大，但是不十分关注存储，一旦经过处理，要么被丢弃，要么被归档存储
  - 注重数据的整体价值，不过分关注个别数据
  - 数据顺序颠倒，或者不完整，系统无法控制将要处理的新到达的数据元素的顺序

8.1.2 批量计算和实时计算

- 对静态数据和流数据的处理，对应着两种截然不同的计算模式：批量计算和实时计算



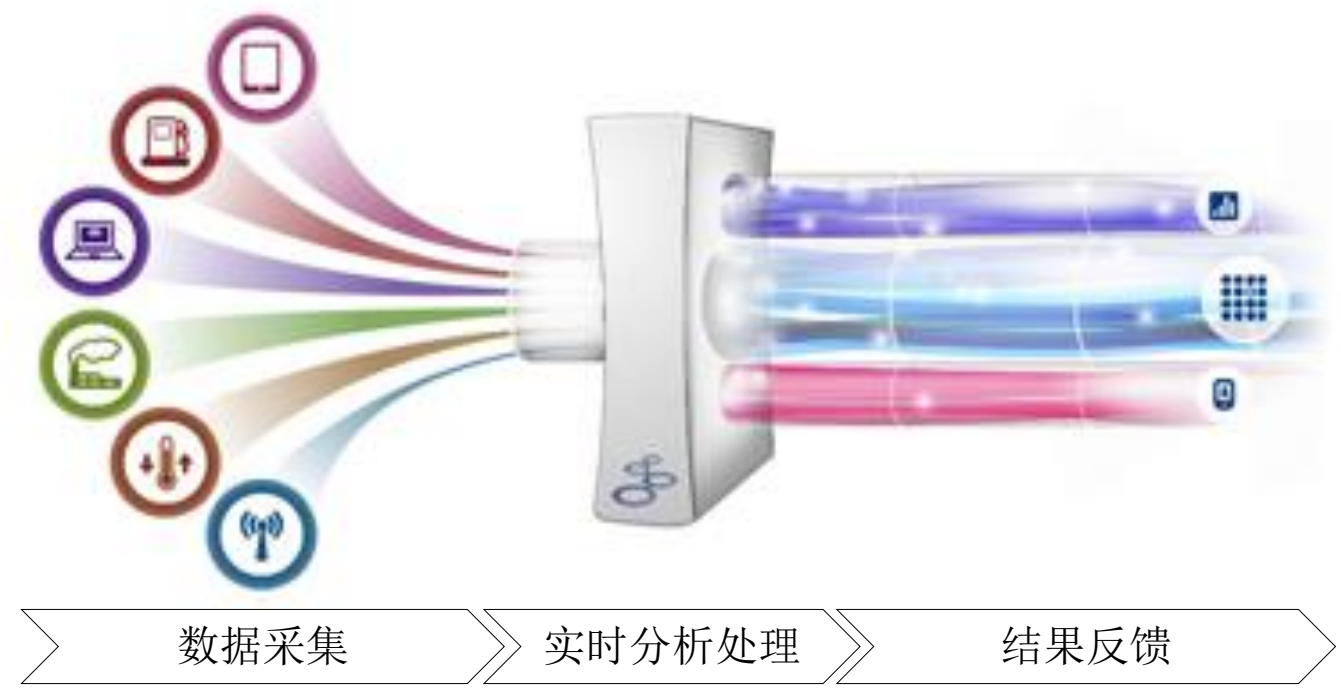
数据的两种处理模型

8.1.2 批量计算和实时计算

- 批量计算以“静态数据”为对象，可在充裕的时间内对海量数据进行批量处理，计算得到有价值的信息。Hadoop是典型的批处理模型，由HDFS和HBase存放大量的静态数据，由MapReduce负责对海量数据执行批量计算
- 流数据须采用实时计算。实时计算最重要的一个需求是能够实时得到计算结果，一般要求响应时间为秒级。当只需要处理少量数据时，实时计算并不是问题；但是，在大数据时代，数据格式复杂、来源众多、数据量巨大，对实时计算提出了很大的挑战。因此，针对流数据的实时计算——流计算，应运而生

8.1.3 流计算概念

- 流计算：实时获取来自不同数据源的海量数据，经过实时分析处理，获得有价值的信息



流计算示意图

8.1.3 流计算概念

- 流计算秉承一个基本理念，即**数据的价值随着时间的流逝而降低**。因此，当事件出现时就应该立即进行处理，而不是缓存起来进行批量处理。为了及时处理流数据，就需要一个低延迟、可扩展、高可靠的处理引擎
- 对于一个流计算系统来说，它应达到如下需求：
  - 高性能：处理大数据的基本要求，如每秒处理几十万条数据
  - 海量式：支持TB级甚至是PB级的数据规模
  - 实时性：保证较低的延迟时间，达到秒级别，甚至是毫秒级别
  - 分布式：支持大数据的基本架构，必须能够平滑扩展
  - 易用性：能够快速进行开发和部署
  - 可靠性：能可靠地处理流数据

8.1.4 流计算与Hadoop

- Hadoop设计的初衷是面向大规模数据的批量处理，每台机器并行运行MapReduce任务，最后对结果进行汇总输出
- MapReduce是专门面向静态数据的批量处理的，内部各种实现机制都为批处理做了高度优化，不适合用于处理持续到达的动态数据
- 我们可能会想到一种“变通”的方案来降低批处理的时间延迟——将基于MapReduce的批量处理转为小批量处理，将输入数据切成小的片段，每隔一个周期就启动一次MapReduce作业。但这种方式也无法有效处理流数据

8.1.4 流计算框架

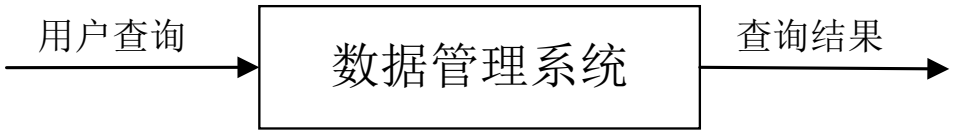
- 当前业界诞生了许多专门的流数据实时计算系统来满足各自需求
- 目前有三类常见的流计算框架和平台：商业级的流计算平台、开源流计算框架、公司为支持自身业务开发的流计算框架
- 较为常见的是开源流计算框架，代表如下：
  - Twitter Storm：免费、开源的分布式实时计算系统，可简单、高效、可靠地处理大量的流数据
  - Yahoo! S4（Simple Scalable Streaming System）：开源流计算平台，是通用的、分布式的、可扩展的、分区容错的、可插拔的流式系统

8.2 流计算处理流程

- 8.2.1 概述
- 8.2.2 数据实时采集
- 8.2.3 数据实时计算
- 8.2.4 实时查询服务

8.2.1 数据处理流程

- 传统的数据处理流程，需要先采集数据并存储在关系数据库等数据管理系统中，之后由用户通过查询操作和数据管理系统进行交互

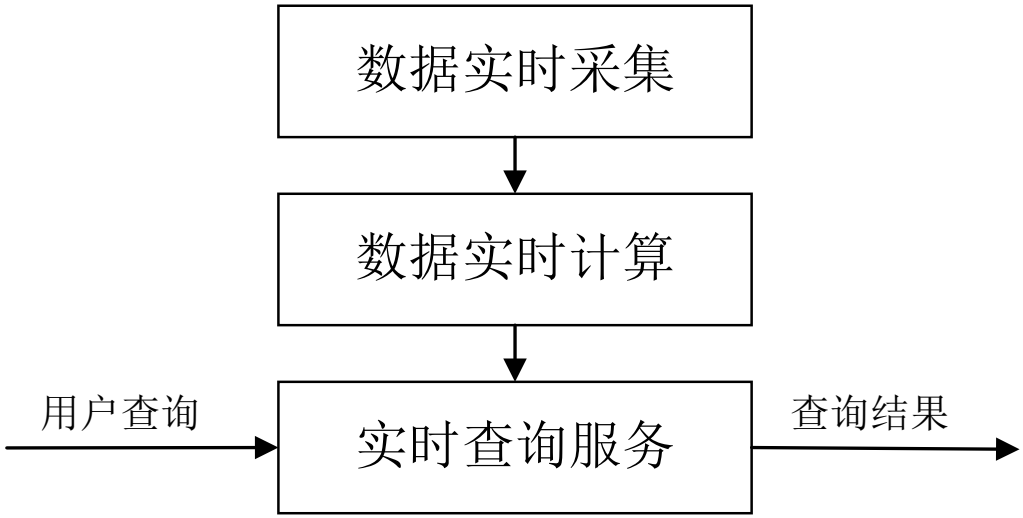


传统的数据处理流程示意图

- 传统的数据处理流程隐含了两个前提：
  - **存储的数据是旧的**。存储的静态数据是过去某一时刻的快照，这些数据在查询时可能已不具备时效性了
  - **需要用户主动发出查询来获取结果**

8.2.1 数据处理流程

- 流计算的处理流程一般包含三个阶段：数据实时采集、数据实时计算、实时查询服务



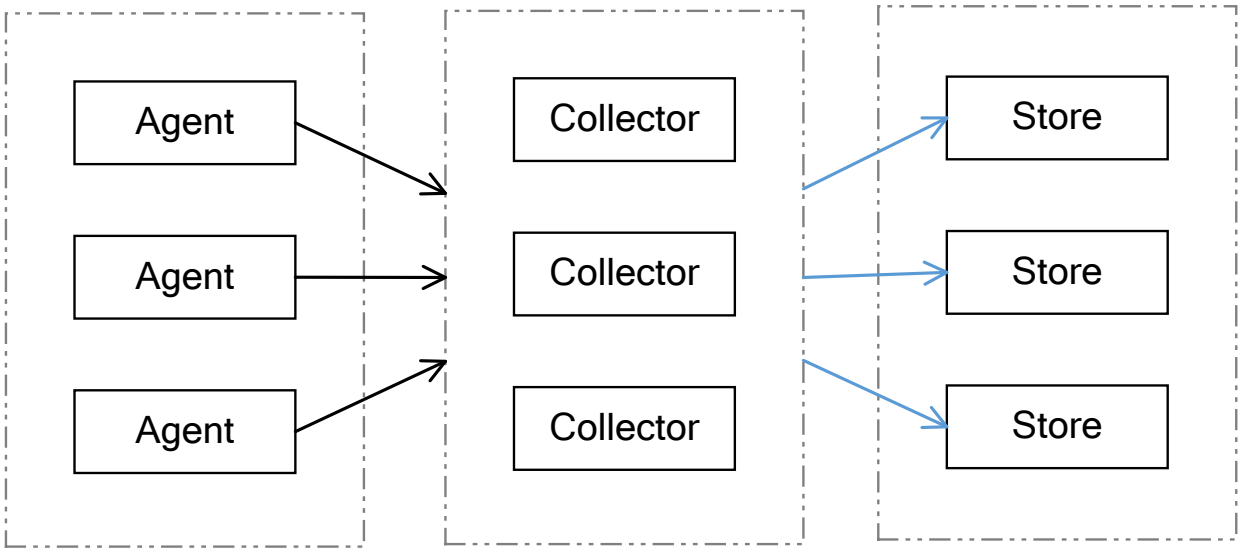
流计算处理流程示意图

8.2.2 数据实时采集

- 数据实时采集阶段通常采集多个数据源的海量数据，需要保证实时性、低延迟与稳定可靠
- 以日志数据为例，由于分布式集群的广泛应用，数据分散存储在不同的机器上，因此需要实时汇总来自不同机器上的日志数据
- 目前有许多互联网公司发布的开源分布式日志采集系统均可满足每秒数百MB的数据采集和传输需求，如：
  - Facebook的Scribe
  - LinkedIn的Kafka
  - 淘宝的Time Tunnel
  - 基于Hadoop的Chukwa和Flume

8.2.2 数据实时采集

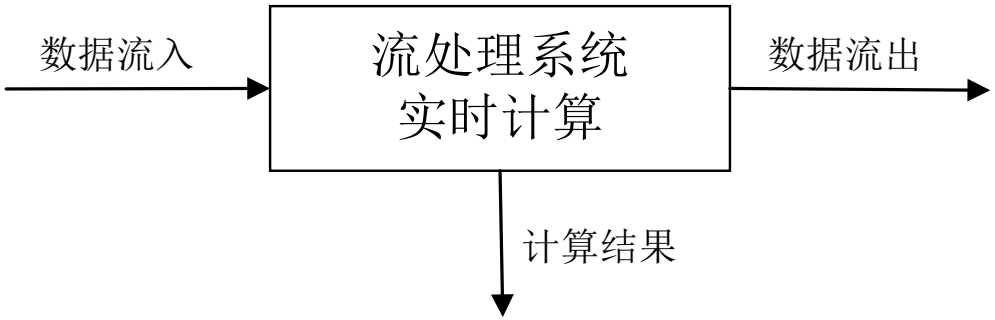
- 数据采集系统的基本架构一般有以下三个部分：
  - Agent：主动采集数据，并把数据推送到Collector部分
  - Collector：接收多个Agent的数据，并实现有序、可靠、高性能的转发
  - Store：存储Collector转发过来的数据



数据采集系统基本架构

8.2.3 数据实时计算

- 数据实时计算阶段对采集的数据进行实时的分析和计算，并反馈实时结果
- 经流处理系统处理后的数据，可视情况进行存储，以便之后再进行分析计算。在时效性要求较高的场景中，处理之后的数据也可以直接丢弃



数据实时计算流程

8.2.3 实时查询服务

- 实时查询服务：经由流计算框架得出的结果可供用户进行实时查询、展示或储存
- 传统的数据处理流程，用户需要主动发出查询才能获得想要的结果。而在流处理流程中，实时查询服务可以不断更新结果，并将用户所需的结果实时推送给用户
- 虽然通过对传统的数据处理系统进行定时查询，也可以实现不断地更新结果和结果推送，但通过这样的方式获取的结果，仍然是根据过去某一时刻的数据得到的结果，与实时结果有着本质的区别

8.2.3 实时查询服务

- 可见，流处理系统与传统的数据处理系统有如下不同：
  - 流处理系统处理的是实时的数据，而传统的数据处理系统处理的是预先存储好的静态数据
  - 用户通过流处理系统获取的是实时结果，而通过传统的数据处理系统，获取的是过去某一时刻的结果
  - 流处理系统无需用户主动发出查询，实时查询服务可以主动将实时结果推送给用户

8.3 流计算的应用

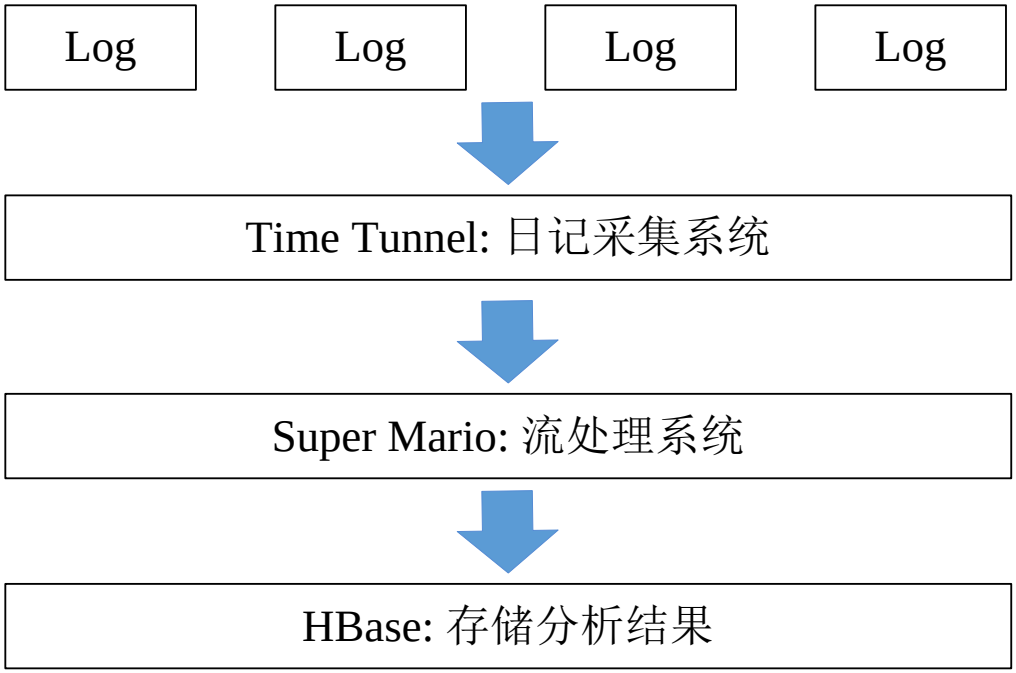
- 流计算是针对流数据的实时计算，可以应用在多种场景中，如Web服务、机器翻译、广告投放、自然语言处理、气候模拟预测等
- 如百度、淘宝等大型网站中，每天都会产生大量流数据，包括用户的搜索内容、用户的浏览记录等数据。采用流计算进行实时数据分析，可以了解每个时刻的流量变化情况，甚至可以分析用户的实时浏览轨迹，从而进行实时个性化内容推荐
- 但是，并不是每个应用场景都需要用到流计算的。流计算适合于需要处理持续到达的流数据、对数据处理有较高实时性要求的场景

8.3.1 应用场景1: 实时分析

- 传统的业务分析一般采用分布式离线计算的方式，即将数据全部保存起来，然后每隔一定的时间进行离线分析来得到结果。但这样会导致一定的延时，难以保证结果的实时性
- 如淘宝网“双十一”、“双十二”的促销活动，商家需要根据广告效果来即使调整广告，这就需要对广告的受访情况进行分析。但以往采用分布式离线分析，需要几小时甚至一天的延时才能得到分析结果。而促销活动只持续一天，因此，隔天才能得到的分析结果便失去了价值
- 虽然分布式离线分析带来的小时级的分析延时可以满足大部分商家的需求，但随着实时性要求越来越高，如何实现秒级别的实时分析响应成为业务分析的一大挑战

8.3.1 应用场景1: 实时分析

- 针对流数据，“量子恒道”开发了海量数据实时流计算框架Super Mario。通过该框架，量子恒道可处理每天TB级的实时流数据，并且从用户发出请求到数据展示，整个延时控制在2-3秒内，达到了实时性的要求



Super Mario处理流程

8.3.1 应用场景2: 实时交通

- 流计算不仅为互联网带来改变，也能改变我们的生活
- 如提供导航路线，一般的导航路线并没有考虑实时的交通状况，即便在计算路线时有考虑交通状况，往往也只是使用了以往的交通状况数据。要达到根据实时交通状态进行导航的效果，就需要获取海量的实时交通数据并进行实时分析
- 借助于流计算的实时特性，不仅可以根据交通情况制定路线，而且在行驶过程中，也可以根据交通情况的变化实时更新路线，始终为用户提供最佳的行驶路线

8.3.1 应用场景2: 实时交通

- IBM的流计算平台InfoSphere Streams，广泛应用于制造、零售、交通运输、金融证券以及监管各行各业的解决方案之中，使得实时快速做出决策的理念得以实现
- 以上述的实时交通为例，InfoSphere Streams应用于斯德哥尔摩的交通信息管理，通过结合来自不同源的实时数据，可以生成动态的、多方位的观察交通流量的方式，为城市规划者和乘客提供实时交通状况查询

8.4 开源流计算框架Storm

- 8.4.1 Storm简介
- 8.4.2 Storm的特点
- 8.4.3 Storm设计思想
- 8.4.4 Storm框架设计
- 8.4.5 Storm实例
- 8.4.6 哪些公司在使用Storm

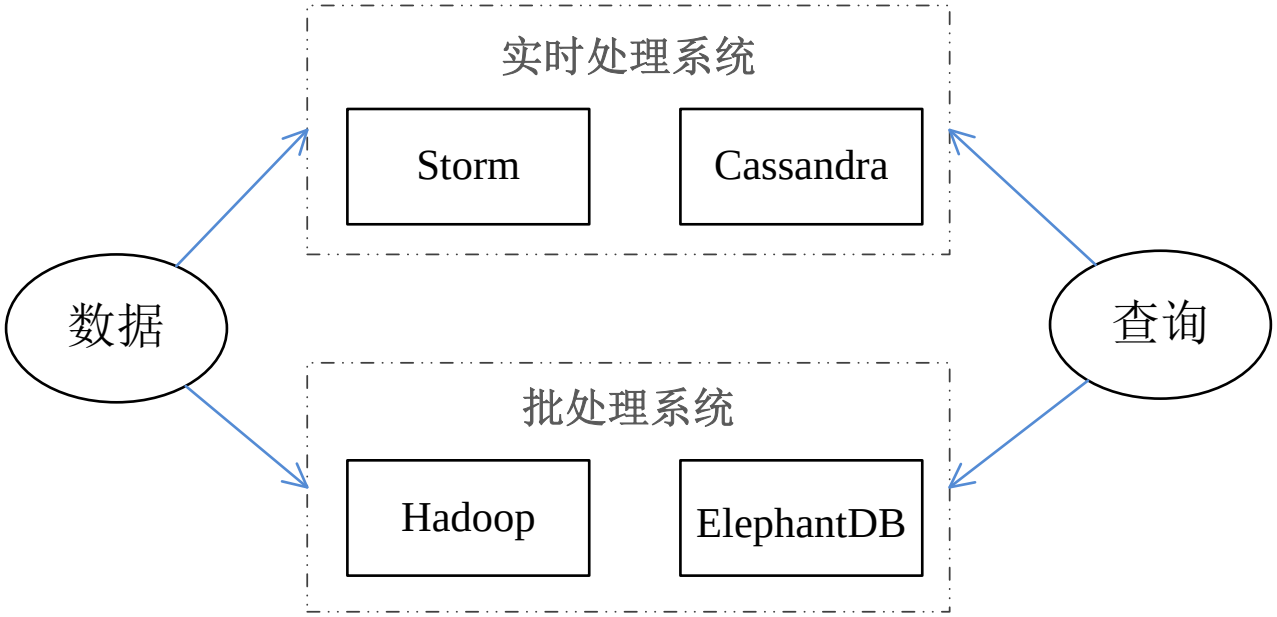
8.4.1 Storm简介

- Twitter Storm是一个免费、开源的分布式实时计算系统，Storm对于实时计算的意义类似于Hadoop对于批处理的意义，Storm可以简单、高效、可靠地处理流数据，并支持多种编程语言
- Storm框架可以方便地与数据库系统进行整合，从而开发出强大的实时计算系统

- Twitter是全球访问量最大的社交网站之一，Twitter开发Storm流处理框架也是为了应对其不断增长的流数据实时处理需求

8.4.1 Storm简介

- Twitter采用了由实时系统和批处理系统组成的分层数据处理架构，一方面由Hadoop和ElephantDB组成批处理系统，另一方面由Storm和Cassandra组成实时系统
- 在计算查询时，该系统会同时查询批处理视图和实时视图，并把它们合并起来以得到最终的结果



Twitter的分层数据处理架构

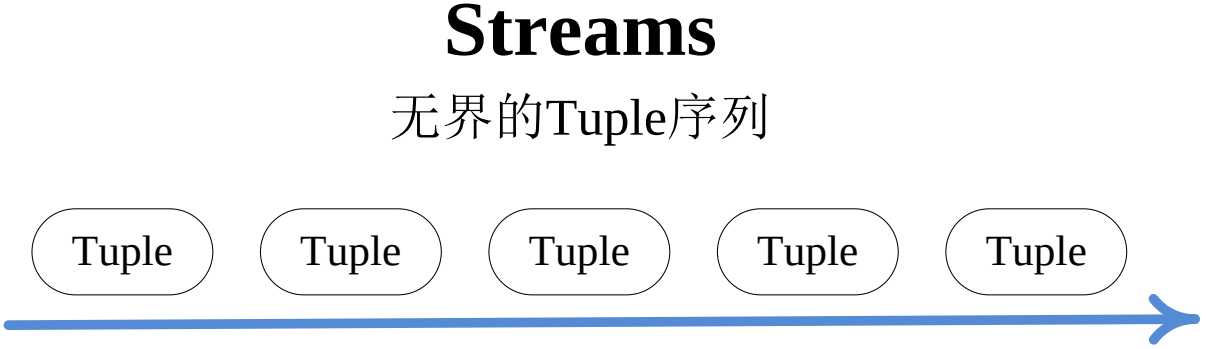
8.4.2 Storm的特点

- Storm可用于许多领域中，如实时分析、在线机器学习、持续计算、远程RPC、数据提取加载转换等
- Storm具有以下主要特点：
  - 整合性：Storm可方便地与队列系统和数据库系统进行整合
  - 简易的API：Storm的API在使用上即简单又方便
  - 可扩展性：Storm的并行特性使其可以运行在分布式集群中
  - 容错性：Storm可自动进行故障节点的重启、任务的重新分配
  - 可靠的消息处理：Storm保证每个消息都能完整处理
  - 支持各种编程语言：Storm支持使用各种编程语言来定义任务
  - 快速部署：Storm可以快速进行部署和使用
  - 免费、开源：Storm是一款开源框架，可以免费使用

8.4.3 Storm设计思想

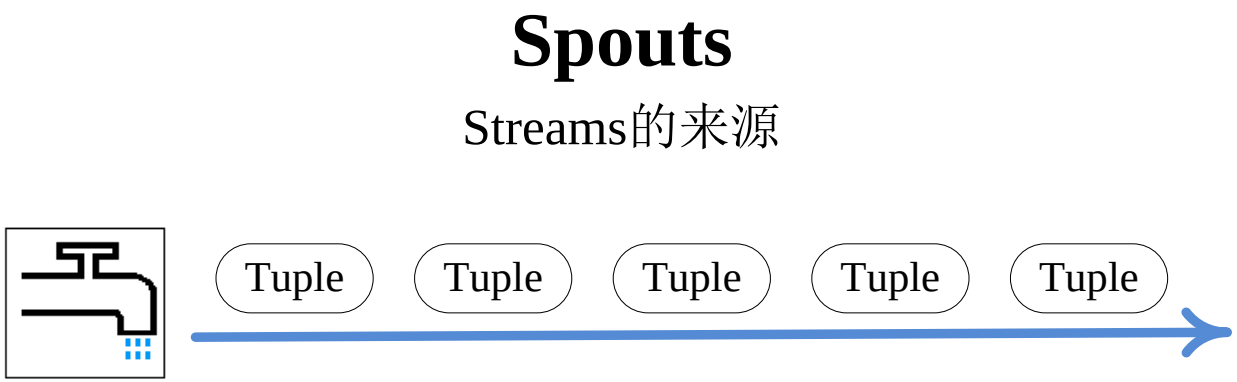
- 要了解Storm，首先需要了解Storm的设计思想。Storm对一些设计思想进行了抽象化，其主要术语包括Streams、Spouts、Bolts、Topology和Stream Groupings

- **Streams**：Storm将流数据Stream描述成一个无限的Tuple序列，这些Tuple序列会以分布式的方式并行地创建和处理



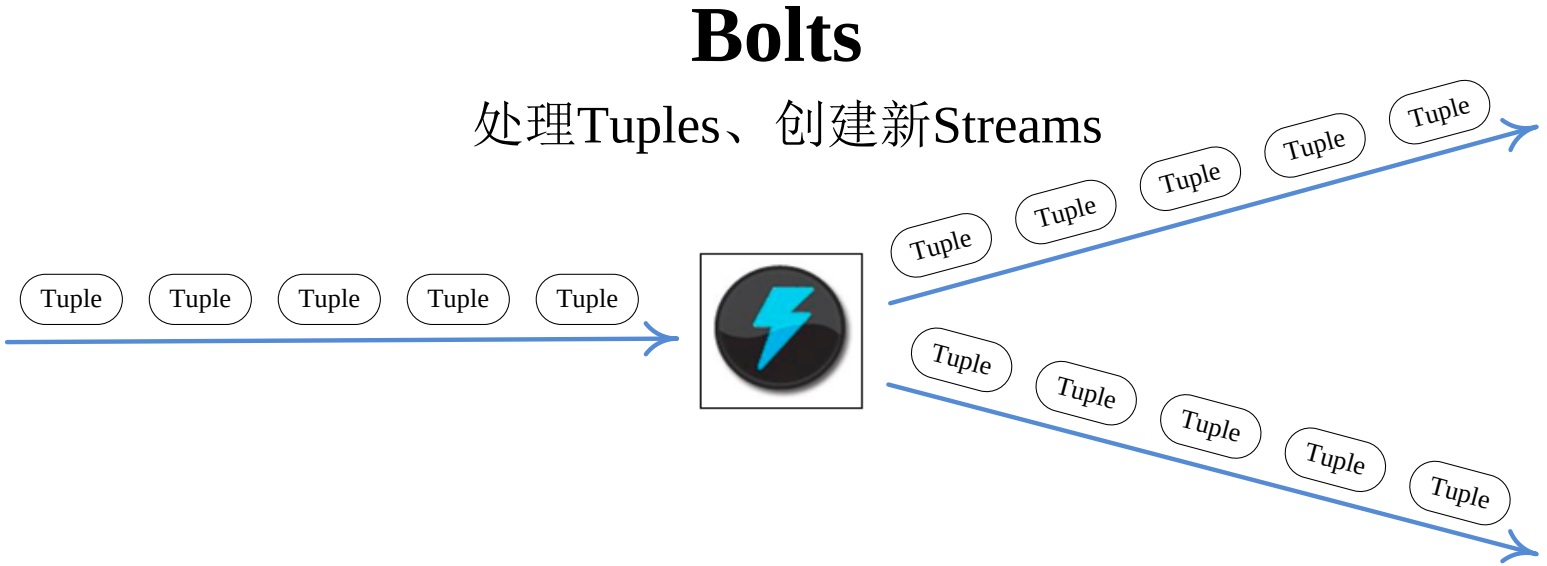
8.4.3 Storm设计思想

- **Spouts**：Storm认为每个Stream都有一个源头，并把这个源头抽象为Spouts。Spouts会从外部读取流数据并持续发出Tuple



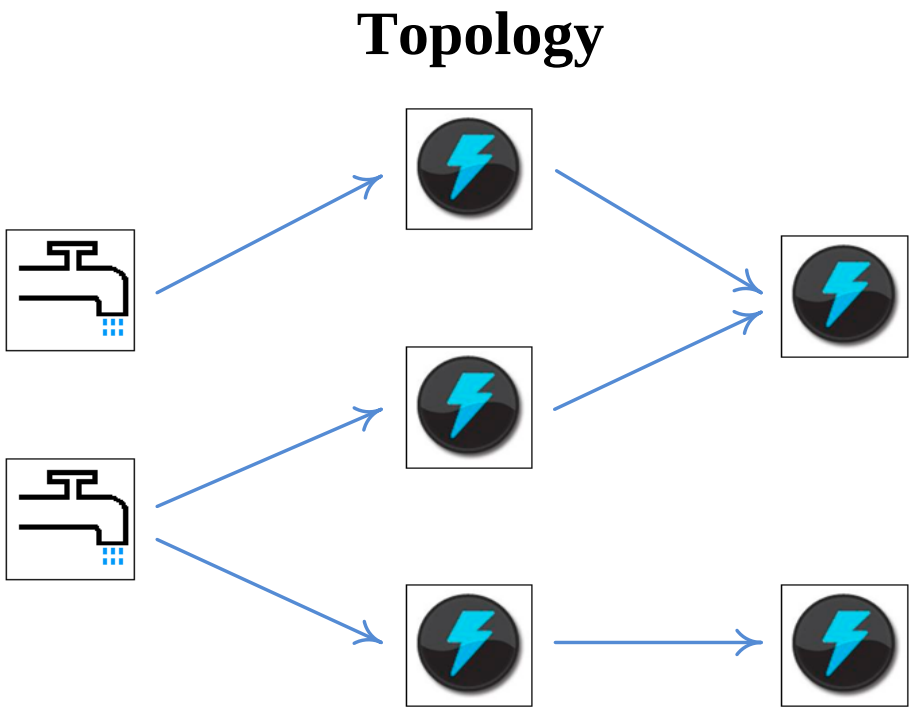
8.4.3 Storm设计思想

- **Bolts**：Storm将Streams的状态转换过程抽象为Bolts。Bolts即可以处理Tuple，也可以将处理后的Tuple作为新的Streams发送给其他Bolts。对Tuple的处理逻辑都被封装在Bolts中，可执行过滤、聚合、查询等操作



8.4.3 Storm设计思想

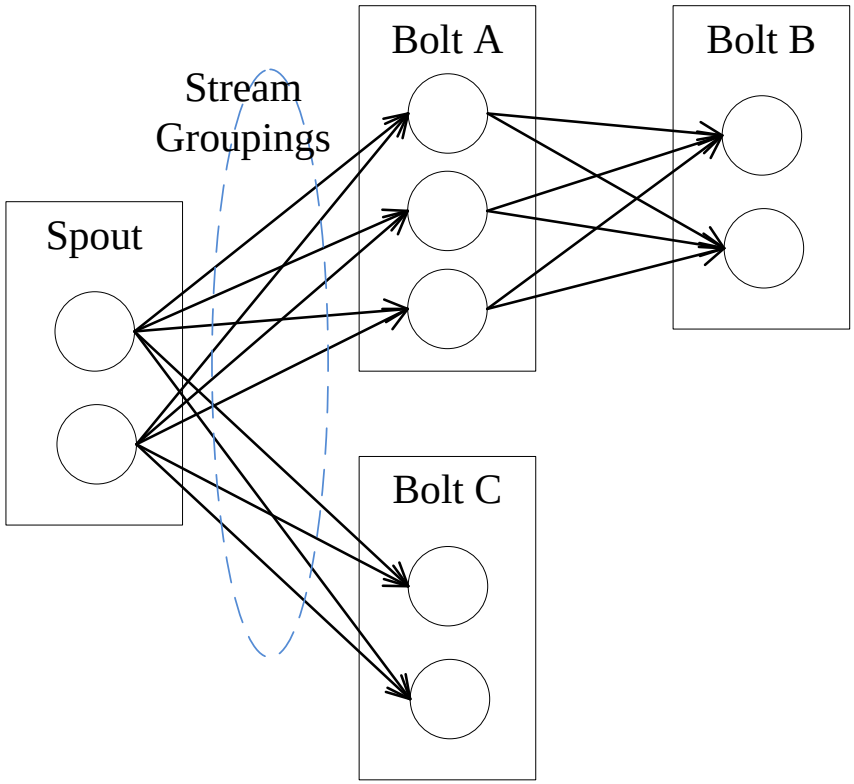
- **Topology**：Storm将Spouts和Bolts组成的网络抽象成Topology，它可以被提交到Storm集群执行。Topology可视为流转换图，图中节点是一个Spout或Bolt，边则表示Bolt订阅了哪个Stream。当Spout或者Bolt发送元组时，它会把元组发送到每个订阅了该Stream的Bolt上进行处理



8.4.3 Storm设计思想

- **Stream Groupings：S**torm中的Stream Groupings用于告知Topology如何在两个组件间（如Spout和Bolt之间，或者不同的Bolt之间）进行Tuple的传送。每一个Spout和Bolt都可以有多个分布式任务，一个任务在什么时候、以什么方式发送Tuple就是由Stream Groupings来决定的

8.4.3 Storm设计思想



Stream Groupings示意图：箭头表示Tuple的流向，而圆圈则表示任务

8.4.3 Storm设计思想

- 目前，Storm中的Stream Groupings有如下几种方式：
  - Shuffle Grouping：随机分组，随机分发Tuple
  - Fields Grouping：按字段分组，具有相同值的Tuple会被分发到对应的Bolt
  - All Grouping：广播分发，每个Tuple都会被分发到所有Bolt中
  - Global Grouping：全局分组，Tuple只会分发给一个Bolt
  - Non Grouping：不分组，与随机分组效果类似
  - Direct Grouping：直接分组，由Tuple的生产者来定义接收者

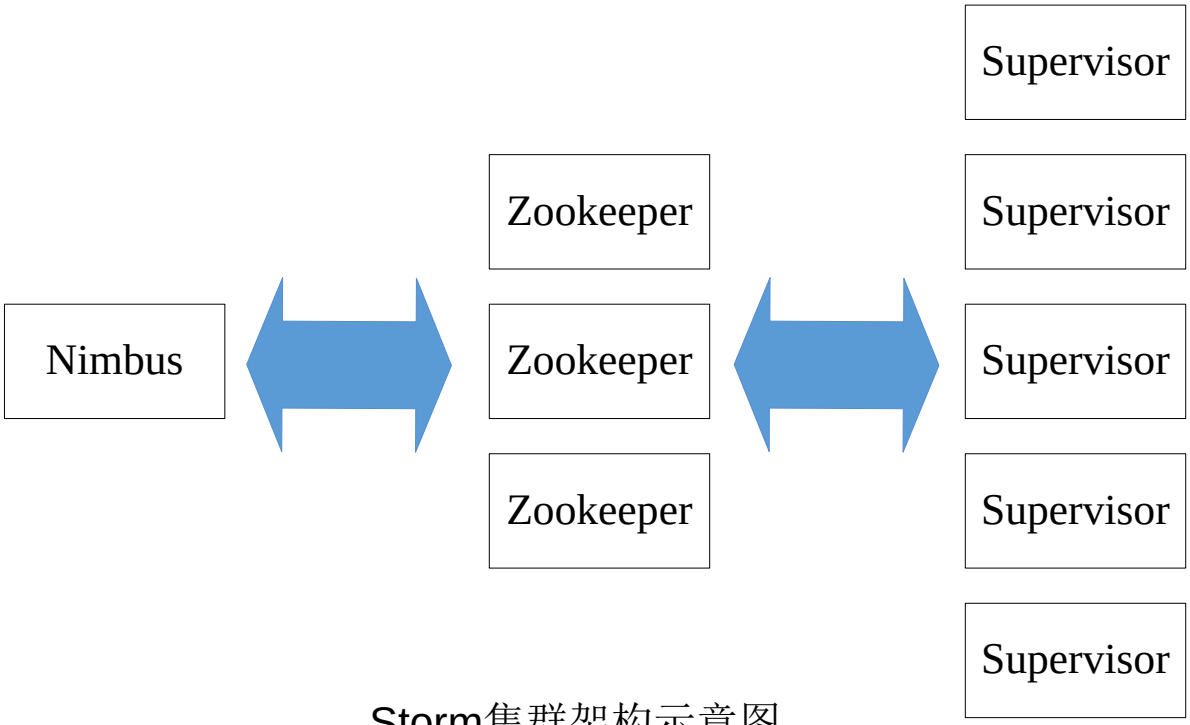
8.4.4 Storm框架设计

- Storm运行任务的方式与Hadoop类似：Hadoop运行的是MapReduce作业，而Storm运行的是“Topology”
- 但两者的任务大不相同，主要的不同是：MapReduce作业最终会完成计算并结束运行，而Topology将持续处理消息（直到人为终止）

- Storm集群采用“Master—Worker”的节点方式：
  - Master节点运行名为“Nimbus”的后台程序（类似Hadoop中的“JobTracker”），负责在集群范围内分发代码、为Worker分配任务和监测故障
  - Worker节点运行名为“Supervisor”的后台程序，负责监听分配给它所在机器的工作，即根据Nimbus分配的任务来决定启动或停止Worker进程

8.4.4 Storm框架设计

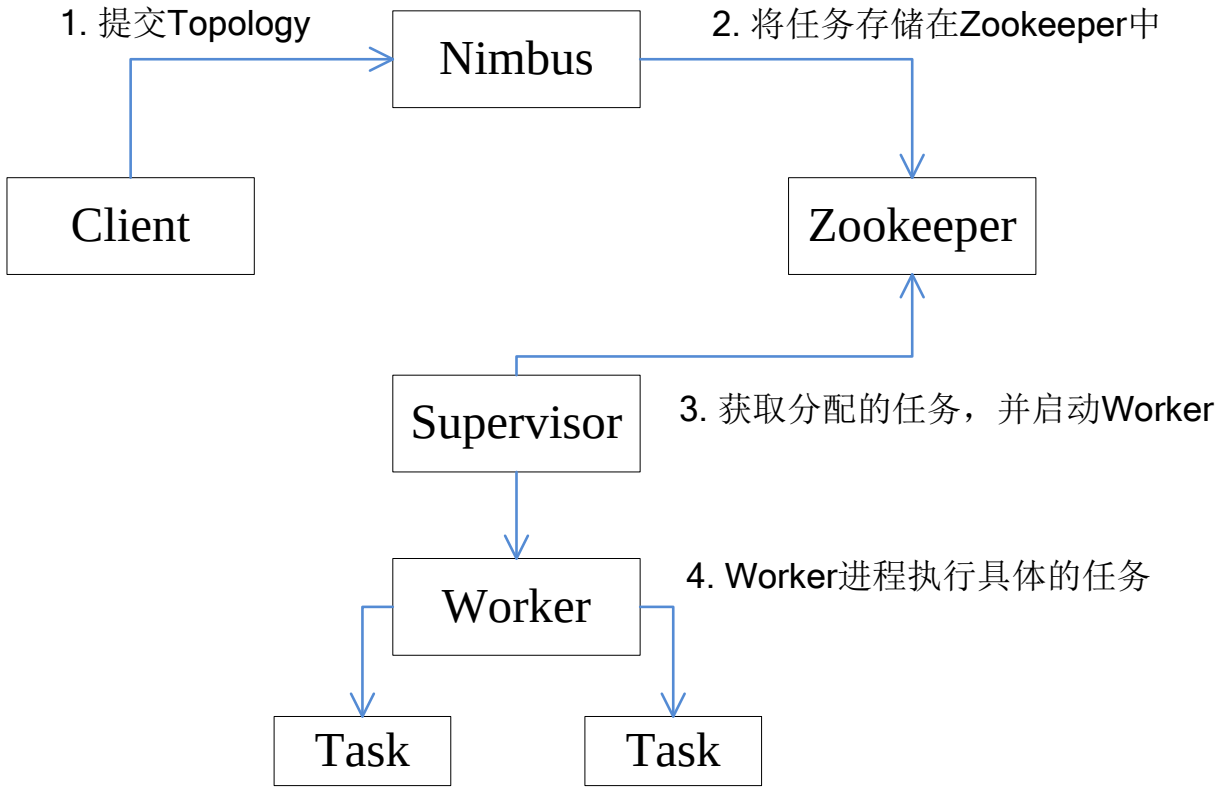
- Storm使用Zookeeper来作为分布式协调组件，负责Nimbus和多个Supervisor之间的所有协调工作。借助于Zookeeper，若Nimbus进程或Supervisor进程意外终止，重启时也能读取、恢复之前的状态并继续工作，使得Storm极其稳定



Storm集群架构示意图

8.4.4 Storm框架设计

- 基于这样的架构设计，Storm的工作流程如下图所示：



Storm工作流程示意图

8.4.5 Storm实例

- 我们以单词统计的实例来加深对Storm的认识
- Storm的编程模型非常简单，如下代码即定义了整个单词统计Topology的整体逻辑

```
// 创建一个Topology builder
TopologyBuilder builder = new TopologyBuilder();
// 使用builder.setSpout()方法对Spout数据源进行定义
builder.setSpout("sentences", new RandomSentenceSpout(), 5);
// 使用builder.setBolt()定义Bolt处理任务
builder.setBolt("split", new SplitSentence(), 8)
        .shuffleGrouping("sentences");
// Groupings()系列方法定义了Tuple发送方式
builder.setBolt("count", new WordCount(), 12)
        .fieldsGrouping("split", new Fields("word"));
```

8.4.5 Storm实例

- Topology中仅定义了整体的计算逻辑，还需要定义具体的处理函数。具体的处理函数可以使用任一编程语言来定义，甚至也可以结合多种编程语言来实现
- 如SplitSentence()方法虽然是通过Java语言定义的，但具体的操作可通过Python脚本来完成

```
public static class SplitSentence extends ShellBolt implements IRichBolt {
    public SplitSentence() {
        // 单词分割的具体实现由Python脚本实现
        super("python", "splitsentence.py");
    }
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word"));
    }
}
```

8.4.5 Storm实例

- Python脚本splitsentence.py定义了一个简单的单词分割方法，即通过空格来分割单词。分割后的单词通过emit()方法以Tuple的形式发送给订阅了该Stream的Bolt进行接收和处理

```
# 简单的单词分割实现，以空格作为分割点
import storm
class SplitSentenceBolt(storm.BasicBolt):
    def process(self, tup):
        words = tup.values[0].split(" ")
        for word in words:
            storm.emit([word])
SplitSentenceBolt().run()
```

8.4.5 Storm实例

- 单词统计的具体逻辑：首先判断单词是否统计过，若未统计过，需先将count值置为0。若单词已统计过，则每出现一次该单词，count值就加1

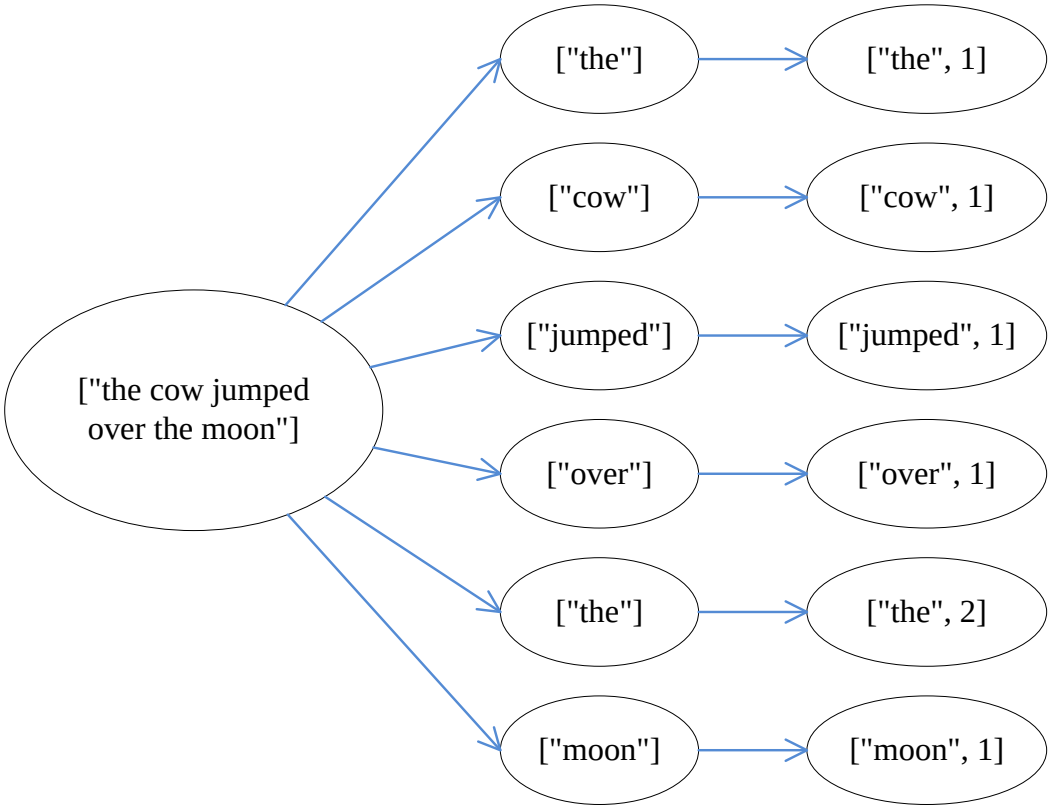
```
// 对单词进行计数
public static class WordCount extends BaseBasicBolt {
    Map<String, Integer> counts = new HashMap<String, Integer>();
    @Override
    public void execute(Tuple tuple, BasicOutputCollector collector) {
        String word = tuple.getString(0);
        Integer count = counts.get(word);
        if (count == null)
        count = 0;
        count++;
        counts.put(word, count);
        collector.emit(new Values(word, count));
    }
    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word", "count"));
    }
}
```

8.4.5 Storm实例

- 基于Storm的单词统计在形式上与基于MapReduce的单词统计是类似的，MapReduce使用的是Map和Reduce的抽象，而Storm使用的是Soput和Bolt的抽象

- 总结一下Storm进行单词统计的整个流程：
  - 从Spout中发送Stream（每个英文句子为一个Tuple）
  - 用于分割单词的Bolt将接收的句子分解为独立的单词，将单词作为Tuple的字段名发送出去
  - 用于计数的Bolt接收表示单词的Tuple，并对其进行统计
  - 输出每个单词以及单词出现过的次数

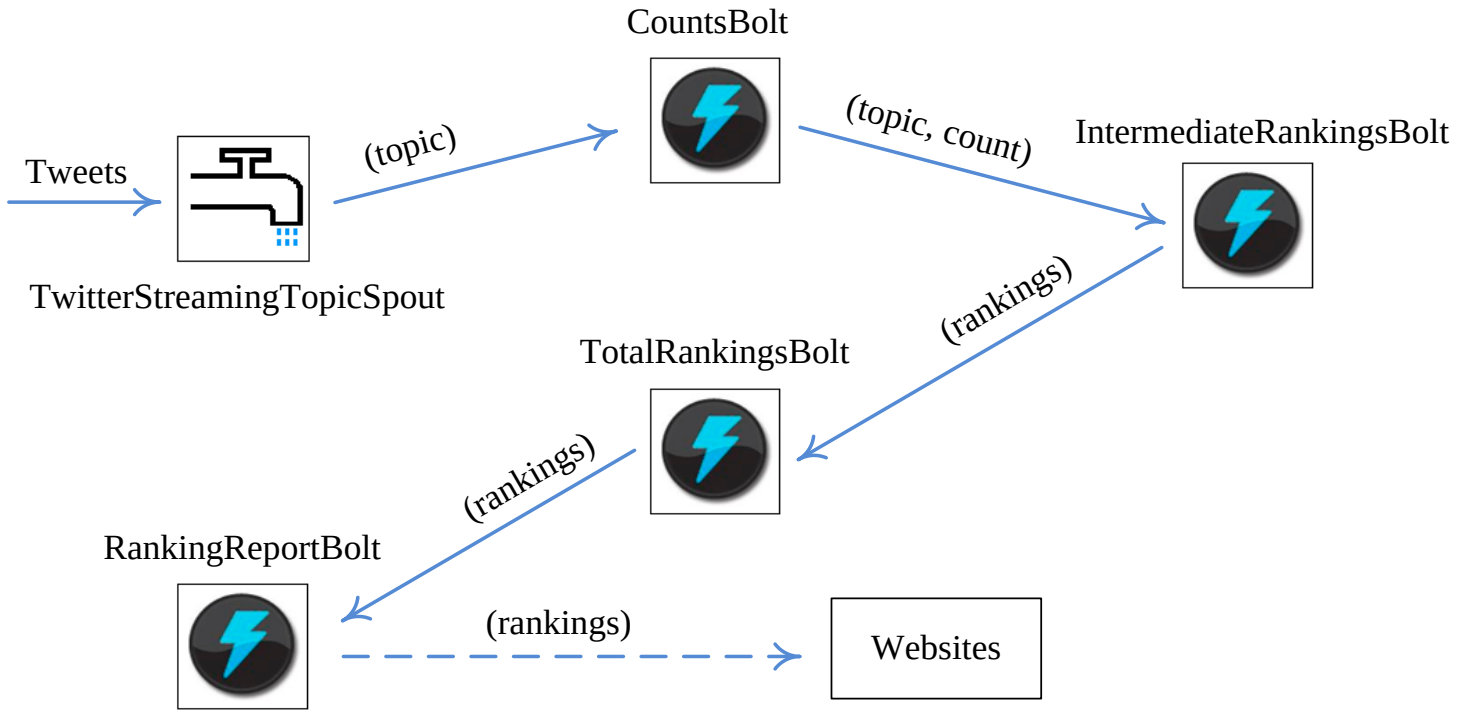
8.4.5 Storm实例



一个句子经Storm的单词统计得出的结果

8.4.5 Storm实例

- 上述虽然是一个简单的单词统计，但对其进行扩展，便可应用到许多场景中，如微博中的实时热门话题。Twitter也正是使用了Storm框架实现了实时热门话题



Twitter实时热门话题处理流程示意图

8.4.6 哪些公司在使用Storm

- Storm自2011年发布以来，凭借其优良的框架设计及开源特性，在流计算领域获得了广泛认可，已应用到许多大型互联网公司的实际项目中

使用Storm的公司和项目

8.5 Storm安装和运行实例

8.5.1 安装Storm的基本过程
8.5.2 运行Storm实例

具体过程请参考厦门大学数据库实验室制作的教程：
《**Storm安装教程_CentOS6.4/Storm0.9.6**》
http://dblab.xmu.edu.cn/blog/install-storm/

8.5.1 安装Storm的基本过程

本实例中Storm具体运行环境如下：
- CentOS 6.4
- Storm 0.9.6
- Java JDK 1.7
- ZooKeeper 3.4.6
- Python 2.6

备注：CentOS中已默认安装了Python 2.6，我们还需要安装 JDK 环境以及分布式应用程序协调服务 Zookeeper

安装Storm的基本过程如下：
- 第一步：安装Java环境
- 第二步：安装 Zookeeper
- 第三步：安装Storm（单机）
- 第四步：关闭Storm

8.5.1 安装Storm的基本过程

第一步：安装Java环境

•Storm 运行需要 Java 环境，可选择 Oracle 的 JDK，或是 OpenJDK，现在一般 Linux 系统默认安装的基本是 OpenJDK，如 CentOS 6.4 就默认安装了 OpenJDK 1.7。但需要注意的是，CentOS 6.4 中默认安装的只是 Java JRE，而不是 JDK，为了开发方便，我们还是需要通过 yum 进行安装 JDK

```
$ sudo yum install java-1.7.0-openjdk java-1.7.0-openjdk-devel
```

•接着需要配置一下 JAVA_HOME 环境变量，为方便，可以在 ~/.bashrc 中进行设置

8.5.1 安装Storm的基本过程

第二步：安装Zookeeper

到官网下载Zookeeper，比如下载 “zookeeper-3.4.6.tar.gz”

下载后执行如下命令进行安装 zookeeper（将命令中 3.4.6 改为你下载的版本）：

```
$ sudo tar -zxf ~/下载/zookeeper-3.4.6.tar.gz -C /usr/local
$ cd /usr/local
$ sudo mv zookeeper-* zookeeper #修改目录名称方便使用
$ sudo chown -R hadoop:hadoop ./zookeeper    # 此处的hadoop为你的用户名
```

chown命令让hadoop用户拥有zookeeper目录下的所有文件的权限

8.5.1 安装Storm的基本过程

第二步：安装Zookeeper（续）

接着执行如下命令进行zookeeper配置：

```
$ cd /usr/local/zookeeper
$ mkdir tmp
$ cp ./conf/zoo_sample.cfg ./conf/zoo.cfg
$ vim ./conf/zoo.cfg
```

将当中的 dataDir=/tmp/zookeeper 更改为 dataDir=/usr/local/zookeeper/tmp 。接着执行：

```
$ ./bin/zkServer.sh start
```

8.5.1 安装Storm的基本过程

第三步：安装Storm（单机）

到官网下载Storm，比如Storm0.9.6
下载后执行如下命令进行安装Storm：

```
$ sudo tar -zxf ~/下载/apache-storm-0.9.6.tar.gz -C /usr/local
$ cd /usr/local
$ sudo mv apache-storm-0.9.6 storm
$ sudo chown -R hadoop:hadoop ./storm    # 此处的hadoop为你的用户名
```

接着执行如下命令进行Storm配置:

```
$ cd /usr/local/storm
$ vim ./conf/storm.yaml
```

8.5.1 安装Storm的基本过程

第三步：安装Storm（单机）（续）

修改其中的 storm.zookeeper.servers 和 nimbus.host 两个配置项，即取消掉注释且都修改值为 127.0.0.1（我们只需要在单机上运行），如下图所示。

然后就可以启动 Storm 了。执行如下命令启动 nimbus 后台进程：

```
$ ./bin/storm nimbus
```

8.5.1 安装Storm的基本过程

第三步：安装Storm（单机）（续）

启动 nimbus 后，终端被该进程占用了，不能再继续执行其他命令了。因此我们需要另外开启一个终端，然后执行如下命令启动 supervisor 后台进程：

```
$ # 需要另外开启一个终端
$ /usr/local/storm/bin/storm supervisor
```

同样的，启动 supervisor 后，我们还需要开启另外的终端才能执行其他命令。另外，我们可以使用 jps 命令 检查是否成功启动，若成功启动会显示 nimbus、supervisor、QuorumPeeMain （QuorumPeeMain 是 zookeeper 的后台进程，若显示 config_value 表明 nimbus 或 supervisor 还在启动中），如下图所示。

8.5.1 安装Storm的基本过程

第四步：关闭Storm

之前启动的 nimbus 和 supervisor 占用了两个终端窗口，切换到这两个终端窗口，按键盘的 Ctrl+C 可以终止进程，终止后，也就相当于关闭了 Storm。

8.5.2 运行Storm实例

Storm中自带了一些例子，我们可以执行一下 WordCount 例子来感受一下 Storm 的执行流程。执行如下命令：

```
$ /usr/local/storm/bin/storm jar /usr/local/storm/examples/storm-starter/storm-starter-topologies-0.9.6.jar storm.starter.WordCountTopology
```

该程序是不断地取如下四句英文句子中的一句作为数据源，然后发送给 bolt 来统计单词出现的次数。

```
{
"the cow jumped over the moon",
 "an apple a day keeps the doctor away",
 "four score and seven years ago",
 "snow white and the seven dwarfs",
 "i am at two with nature"
}
```

本章小结

- 本章首先介绍了流计算的基本概念和需求。流数据即持续到达的大量数据，对流数据的处理强调实时性，一般要求为秒级。MapReduce框架虽然广泛应用于大数据处理中，但其面向的是海量数据的离线处理，并不适合用于处理持续到达的流数据
- 本章阐述了流计算的处理流程，一般包括数据实时采集、数据实时计算和实时查询服务三个部分，并比较其与传统的数据处理流程的不同。流计算处理的是实时数据，而传统的批处理则处理的是预先存储好的静态数据
- 流计算可应用在多个场景中，如实时业务分析，流计算带来的实时性特点，可以大大增加实时数据的价值，为业务分析带来质的提升
- 本章接着介绍了流计算框架Storm的设计思想和架构设计。Storm流处理框架具有可扩展性、高容错性、能可靠地处理消息的特点，使用简单，学习和开发成本较低。Storm框架对设计概念进行了抽象化，其主要术语包括Streams、Spouts、Bolts、Topology和Stream Groupings，在Topology中定义整体任务的处理逻辑，再通过Bolt具体执行，Stream Groupings则定义了Tuple如何在不同组件间进行传输
- 文章最后通过一个单词统计的实例来加深对Storm框架的了解

附录：主讲教师

主讲教师：林子雨

单位：厦门大学计算机科学系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://www.cs.xmu.edu.cn/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），现为厦门大学计算机科学系助理教授（讲师），曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度厦门大学奖教金获得者。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，编著出版中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》并成为畅销书籍，编著并免费网络发布40余万字中国高校第一本闪存数据库研究专著《闪存数据库概念与技术》；主讲厦门大学计算机系本科生课程《数据库系统原理》和研究生课程《分布式数据库》《大数据技术基础》。具有丰富的政府和企业信息化培训经验，曾先后给中国移动通信集团公司、福州马尾区政府、福建省物联网科学研究院、石狮市物流协会、厦门市物流协会、福建龙岩卷烟厂等多家单位和企业开展信息化培训，累计培训人数达2000人以上。

附录：大数据学习教材推荐

扫一扫访问教材官网

《大数据技术原理与应用——概念、存储、处理、分析与应用》，由厦门大学计算机科学系林子雨博士编著，是中国高校第一本系统介绍大数据知识的专业教材。

全书共有13章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、流计算、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase和MapReduce等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。

欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata

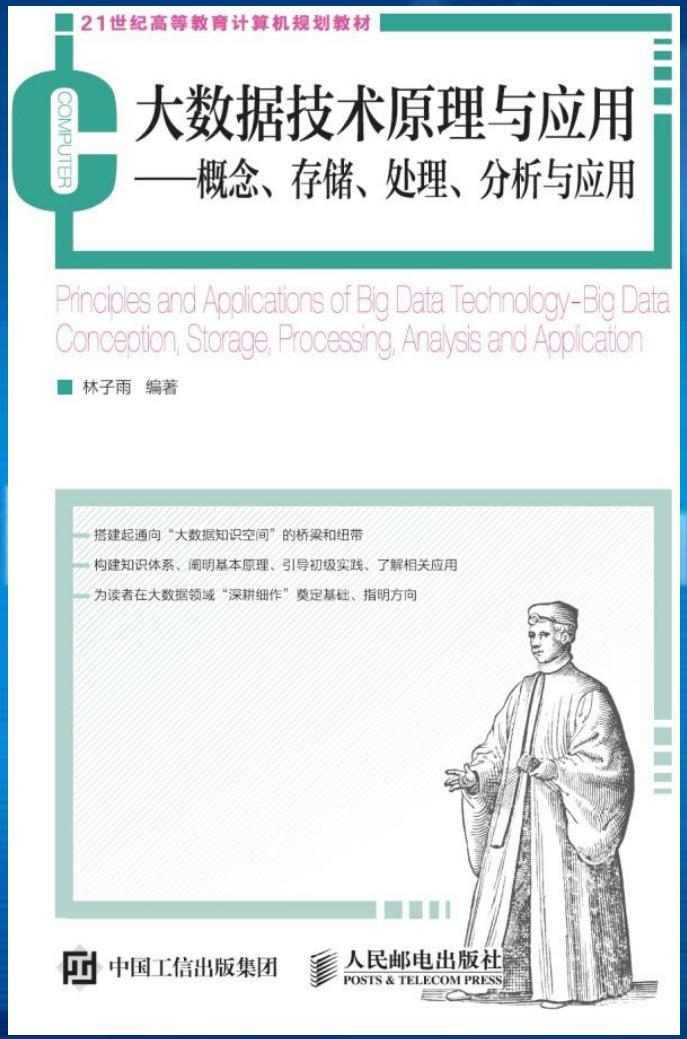
附录：中国高校大数据课程公共服务平台

中国高校大数据课程
公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片



Department of Computer Science, Xiamen University, 2016