厦门大学计算机科学系研究生课程

《大数据技术基础》

第8章 流计算

（2013年新版）

林子雨

厦门大学计算机科学系

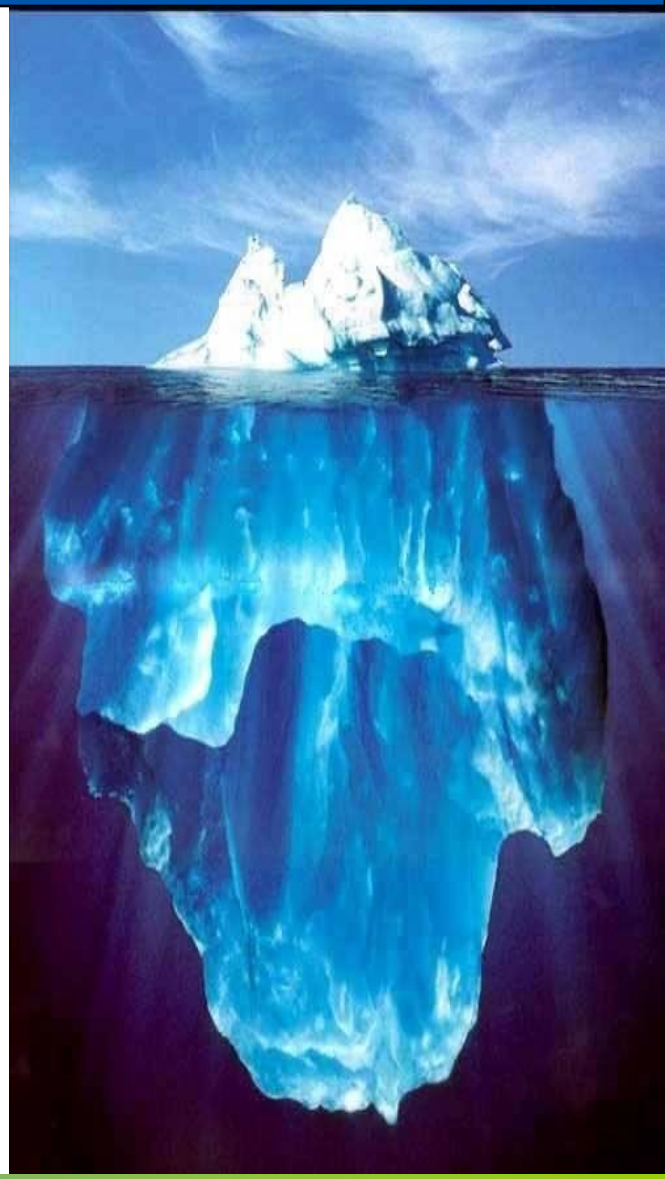
E-mail: ziyulin@xmu.edu.cn

主页：http://www.cs.xmu.edu.cn/linziyu

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料

本讲义PPT存在配套教材，由林子雨通过大量阅读、收集、整理各种资料后编写而成
下载配套教材请访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423



流计算产生的背景

大数据时代数据处理及业务的变化

- 初期：数据量小，业务简单
  - 少量人力、服务器就可以满足需求
- 过渡期：数据量有所膨胀，业务较复杂
  - 需要增加大量服务器以支撑业务
- 大数据时期：数据量急剧膨胀，业务很复杂
  - 传统方案扛不住，简单的增加服务器已不能满足需求

挑战

- 数据量膨胀所带来的质变
- 个性化、实时化的需求
- …

什么是流计算

流计算来自于一个信念：

- 数据的价值随着时间的流逝而降低，所以事件出现后必须尽快地对它们进行处理，最好数据出现时便立刻对其进行处理，发生一个事件进行一次处理，而不是缓存起来成一批再处理。

流计算的概念：

- 流计算是针对**流式数据**的**实时计算**。
- 流式数据（流数据）：是指将数据看作数据流的形式来处理。数据流是在时间分布和数量上无限的一系列动态数据集合体；数据记录是数据流的最小组成单元。
- 流数据具有数据实时持续不断到达、到达次序独立、数据来源众多格式复杂、数据规模大且不十分关注存储、注重数据的整体价值而不关注个别数据等特点。

什么是流计算

流计算应用场景

- 流计算是针对流数据的实时计算，其主要应用在于产生大量流数据、同时对实时性要求高的领域。
- 流计算一方面可应用于处理金融服务如股票交易、银行交易等产生的大量实时数据。
- 另一方面流计算主要应用于各种实时Web服务中，如搜索引擎、购物网站的实时广告推荐，SNS社交类网站的实时个性化内容推荐，大型网站、网店的实时用户访问情况分析等。

什么是流计算

流计算：对流数据实时分析，从而获取有价值的实时信息

流计算与关系存储模型的区别

主要区别有如下几个方面：

- 流中的数据元素在线到达；
- 系统无法控制将要处理的新到达的数据元素的顺序；
- 数据流的潜在大小也许是无穷无尽的；
- 一旦数据流中的某个元素经过处理，要么被丢弃，要么被归档存储。因此，除非该数据被直接存储在内存中，否则将不容易被检索。相对于数据流的大小，这是一种典型的极小相关。

流计算需求

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：必须保证一个较低的延迟时间，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。
- 针对不同的应用场景，相应的流计算系统会有不同的需求，但是，针对海量数据的流计算，无论在数据采集、数据处理中都应达到秒级别的要求。

流计算与Hadoop

- Hadoop的批量化处理是人们喜爱它的地方，但这在某些领域仍显不足，尤其是在例如移动、Web客户端或金融、网页广告等需要实时计算的领域。这些领域产生的数据量极大，没有足够的存储空间来存储每个业务收到的数据。而流计算则可以实时对数据进行分析，并决定是否抛弃无用的数据，而这无需经过Map/Reduce的环节。
- MapReduce框架为批处理做了高度优化，系统典型地通过调度批量任务来操作静态数据，任务不是常驻服务，数据也不是实时流入；而数据流计算的典型范式之一是不确定数据速率的事件流流入系统，系统处理能力必须与事件流量匹配。数据流实时处理的模式决定了要和批处理使用非常不同的架构，试图搭建一个既适合流式计算又适合批处理的通用平台，结果可能会是一个高度复杂的系统，并且最终系统可能对两种计算都不理想。

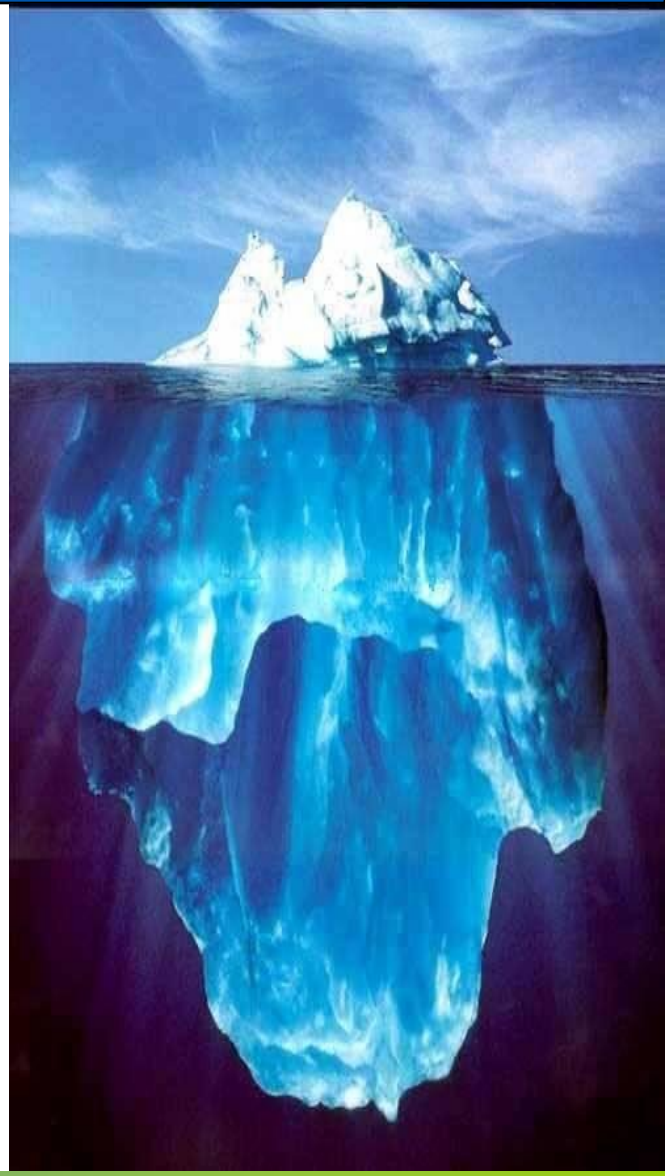
流计算与Hadoop

基于MapReduce的业务不得不面对处理延迟的问题。有一种想法是将基于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，这种实现需要减少每个片段的延迟，并且需要考虑系统的复杂度：

- 将输入数据分隔成固定大小的片段，再由MapReduce平台处理，缺点在于处理延迟与数据片段的长度、初始化处理任务的开销成正比。小的分段是会降低延迟，但是，也增加附加开销，并且分段之间的依赖管理更加复杂（例如一个分段可能会需要前一个分段的信息）；反之，大的分段会增加延迟。最优化的分段大小取决于具体应用。
- 为了支持流式处理，MapReduce需要被改造成Pipeline的模式，而不是reduce直接输出；考虑到效率，中间结果最好只保存在内存中等等。这些改动使得原有的MapReduce框架的复杂度大大增加，不利于系统的维护和扩展。
- 用户被迫使用MapReduce的接口来定义流式作业，这使得用户程序的可伸缩性降低。

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料



传统数据处理流程

- 传统的数据操作，首先将数据采集并存储在DBMS中，然后通过query和DBMS进行交互，得到用户想要的结果。这样的一个流程隐含了两个前提：
  - **Data is old**。当对数据做查询的时候，里面数据其实是过去某一个时刻数据的一个snapshot，数据可能已经过期了；
  - 这样的流程需要人们主动发出query。也就是说**用户是主动的，而DBMS系统是被动的**。

传统数据处理流程示意图

流计算处理流程

- 流计算一般有三个处理流程：数据实时采集、数据实时计算、实时查询服务。

实时计算三个阶段

流计算处理流程

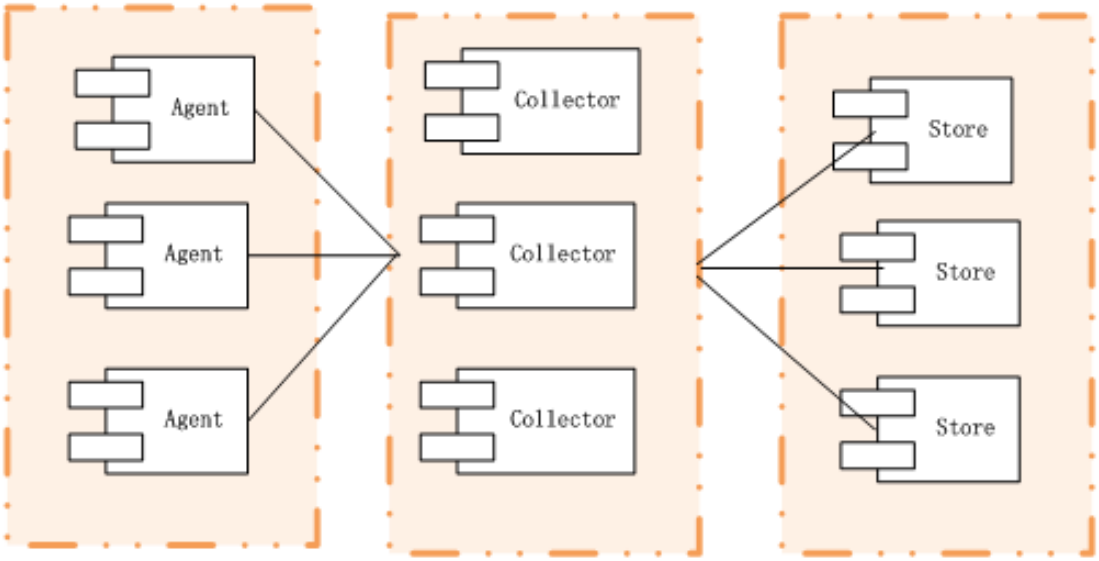
阶段一：数据实时采集

- 为流计算提供实时数据，要保证实时性、低延迟、稳定可靠。
- 许多开源分布式日志收集系统均可满足每秒数百MB的数据采集和传输需求。
  - Hadoop的 **Chukwa**
  - Facebook的 **Scribe**
  - LinkedIn的 **Kafka**
  - Cloudera的 **Flume**
  - 淘宝的 **TimeTunnel**

流计算的阶段

阶段一：数据实时采集

- 数据采集系统基本架构一般由三部分组成
  - Agent: 主动采集数据，并把数据推送到collector
  - 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的数据（对于流计算来说，这边接收的数据一般直接用于计算）

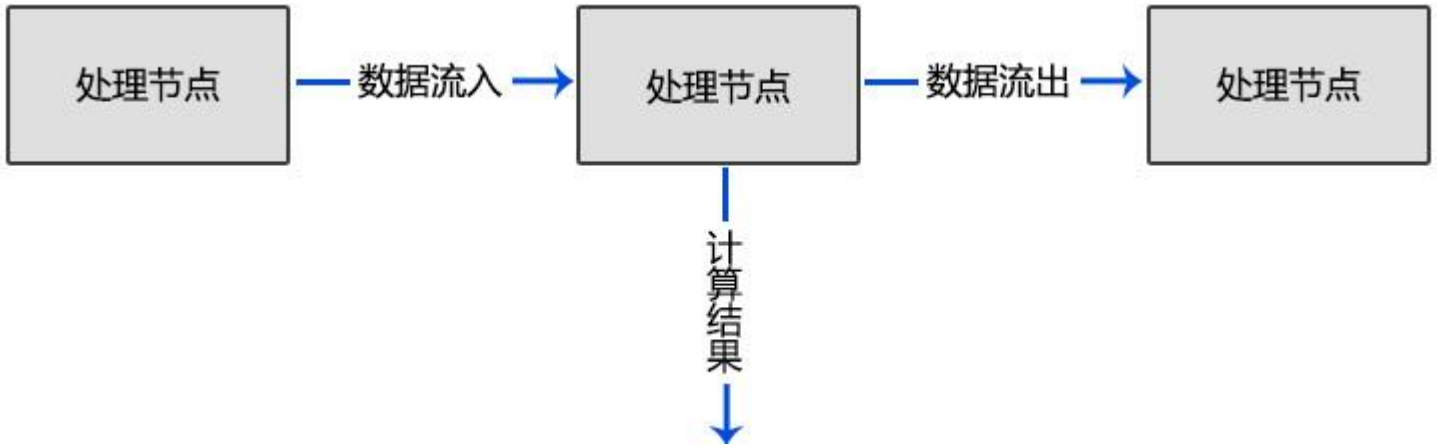


实时采集系统基本架构

流计算的阶段

阶段二：数据实时计算

- 现在大量存在的实时数据，人们需要根据当前的数据实时的作出判断。流计算在流数据不断变化的运动过程中实时地进行分析，捕捉到可能对用户有用的信息，并把结果发送出去，在这种情况下：
  - 能对流数据做出实时回应；
  - **用户是被动的而DBMS是主动的。**



数据实时计算示意图

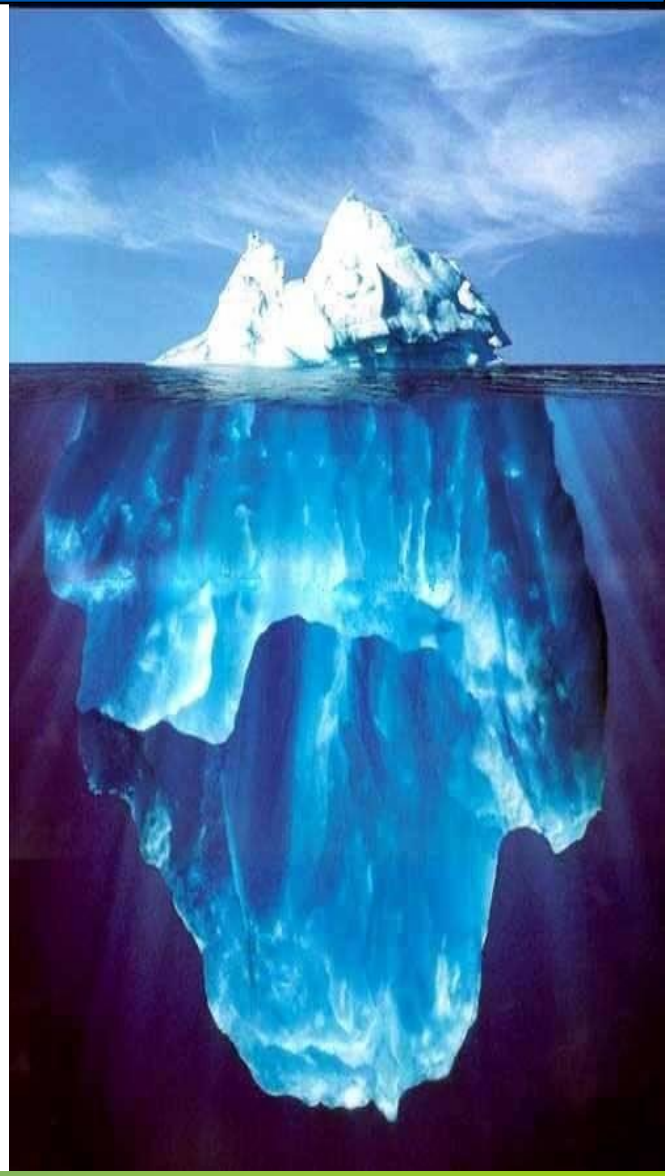
流计算的阶段

阶段三：实时查询服务

- 经由流计算框架得出的结果可供实时查询、展示或存储。

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料



流计算的应用

分析系统

- 传统的分析系统都是离线计算，即将数据全部保存下来，然后每隔一定时间进行离线分析，再将结果保存。但这样会有一定的延时，这取决于离线计算的间隔时间和计算时长。
- 而通过流计算，能在秒级别内得到实时分析结果，有利于根据实时分析结果及时做出决策、调整。

基于分析系统的应用场景

- 广告系统：如搜索引擎和购物网站，实时分析用户信息，展示更佳的相关广告。
- 个性化推荐：如社交网站，实时统计和分析用户行为，精确推荐，增加用户粘性。
- …

流计算的应用 – 量子恒道

挑战

- 实时计算处理数据3T/日
- 离线分布式计算处理数据超过20T/日
- 服务超过百万的淘宝卖家
- …

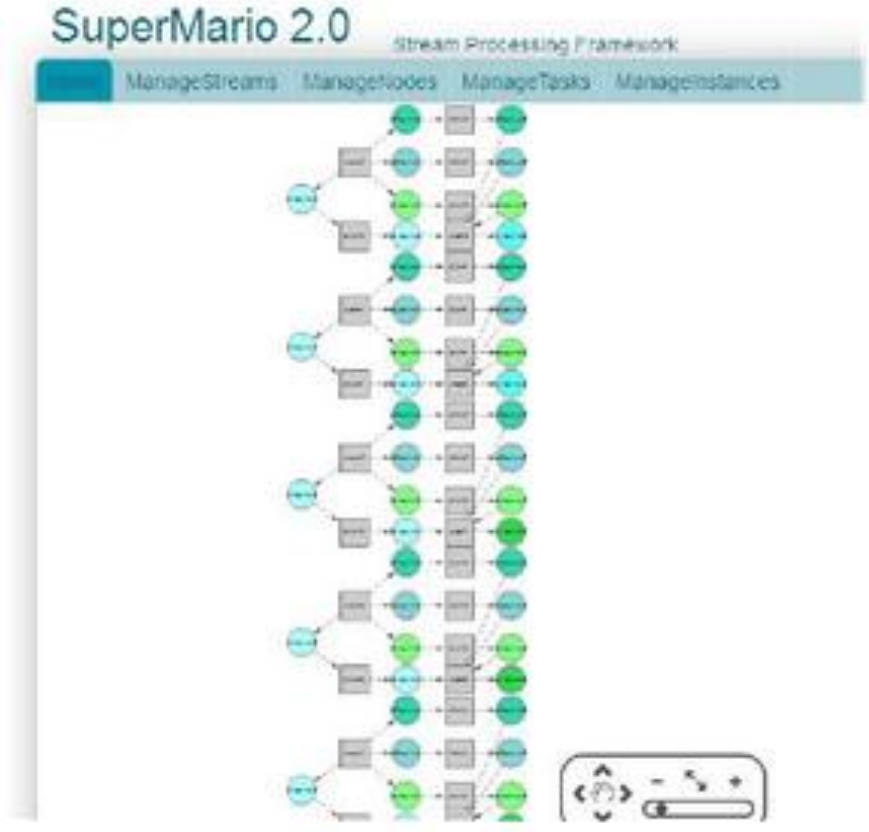
问题

- 离线计算分析延时太大，对于需要实时分析数据的应用场景（如双11，双12，一年就一次，需要实时数据来帮助调整决策），如何实现秒级别的实时分析？

流计算的应用 – 量子恒道

Super Mario 2.0流计算框架

- 海量数据实时计算引擎、实时流传输框架
- 基于Erlang+Zookeeper开发
- 低延迟、高可靠性



Super Mario 2.0(监控界面)

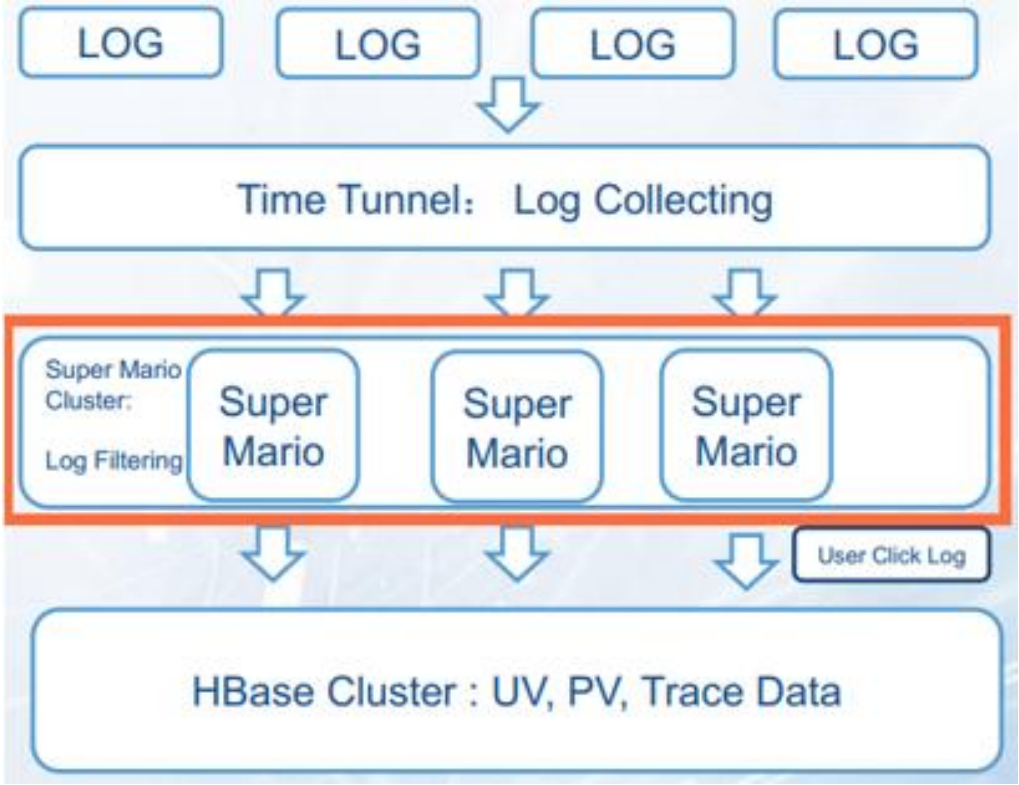
流计算的应用 – 量子恒道

实时数据处理流程

- Log数据由TimeTunnel在毫秒级别内实时送达。
- 实时数据经由Super Mario流计算框架进行处理。
- HBase输出、存储结果

实现效果

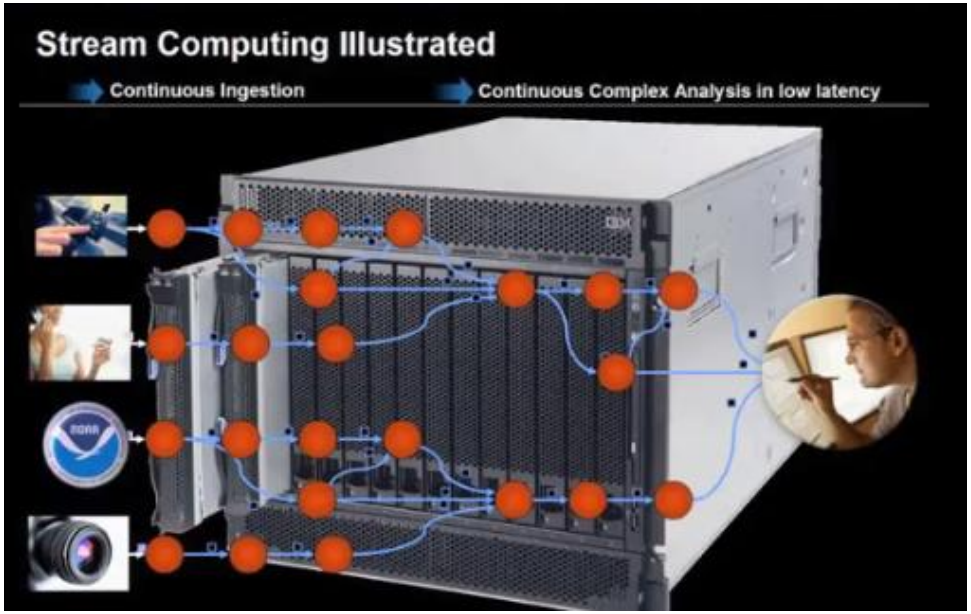
- 可处理每天TB级的实时流数据。
- 从用户发起请求到数据展示，延时控制在2-3秒内。



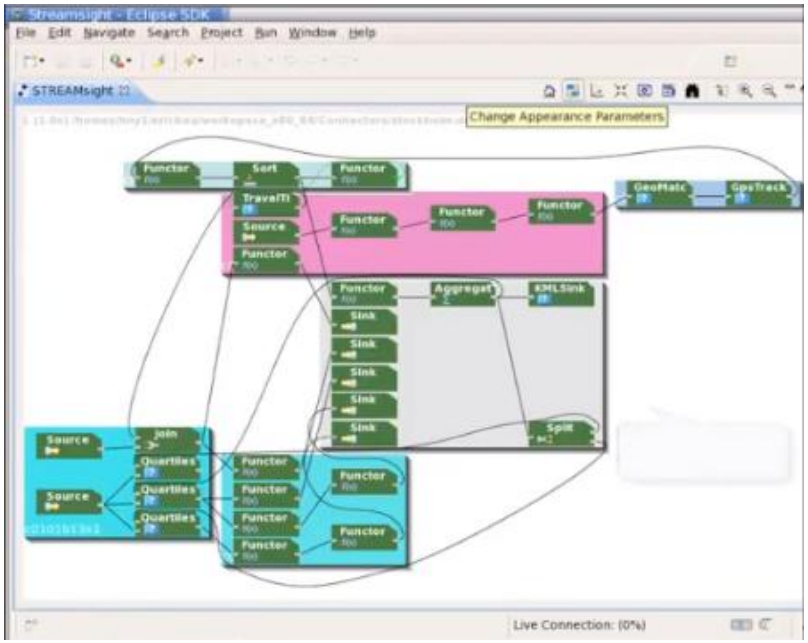
量子恒道实时数据处理示意图

流计算的应用 – 实时交通信息管理

- IBM的流计算平台InfoSphere Streams能够广泛应用于制造、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。



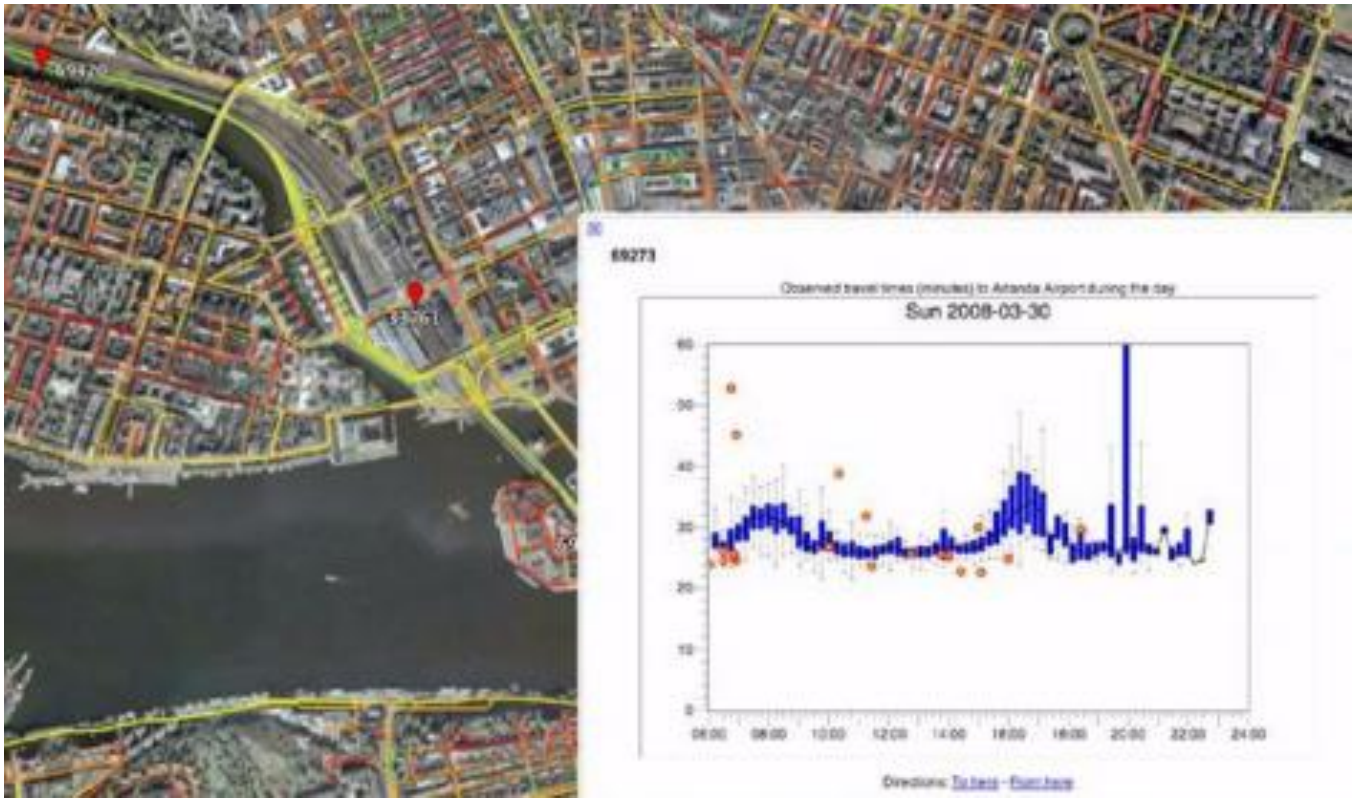
汇总来自不同源的实时数据



InfoSphere Stream界面

流计算的应用 – 实时交通信息管理

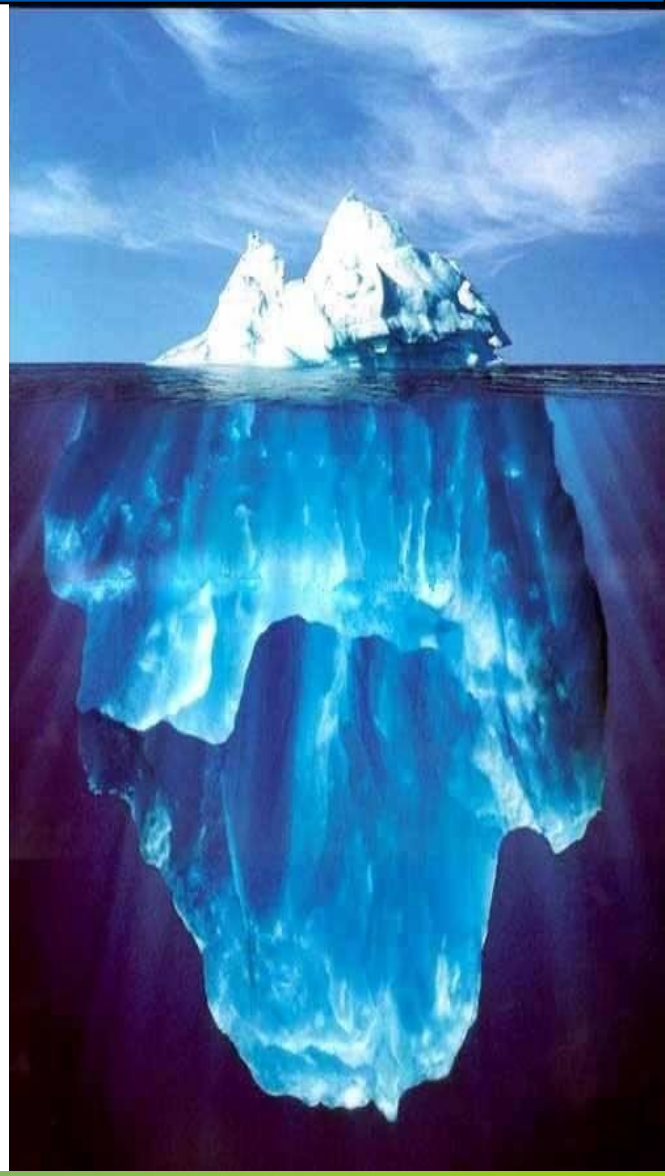
- Streams应用于斯德哥尔摩的交通信息管理，通过结合来自不同源的实时数据，Streams可以生成动态的、多方位的看待交通流量的方式，为城市规划者和乘客提供实时交通状况查看。



通过InfoSphere Streams分析实时交通信息

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料



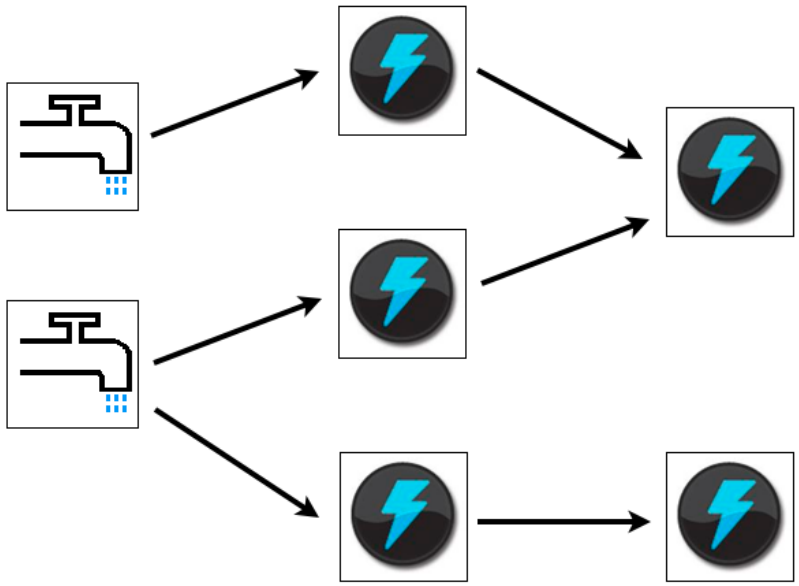
流计算框架要求

流计算框架要求

- 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级数据，甚至是PB级
- 实时性
  - 保证较低延迟事件，达到秒级，最好是毫秒级
- 分布式
  - 支持大数据的基本架构，必须能平滑扩展
- 易用性
- 可靠性
- …

Twitter Storm简介

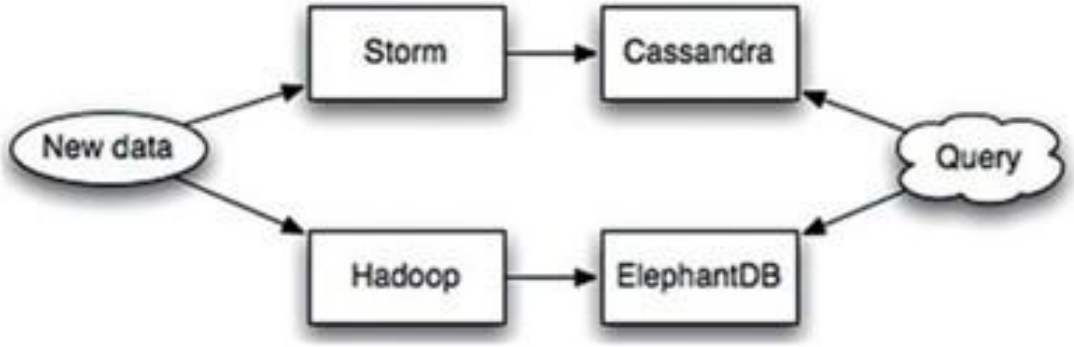
- 免费、开源的分布式实时计算系统
- 简单、高效、可靠地处理大量的流数据
- Storm对于实时计算的意义类似于Hadoop对于批处理的意义
- 基于Clojure和Java开发



Storm 流式计算

Twitter Storm简介

- 为了处理最近的数据，需要一个实时系统和批处理系统同时运行。要计算一个查询函数，需要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数据。
- Twitter中进行实时计算的系统就是Storm，它在数据流上进行持续计算，并且对这种流式数据处理提供了有力保障。
- Twitter分层的数据处理架构由Hadoop和ElephantDB组成批处理系统，Storm和Cassandra组成实时系统，实时系统处理的结果最终会由批处理系统来修正，正是这个观点使得Storm的设计与众不同。



Twitter数据系统分层处理架构

Storm应用领域

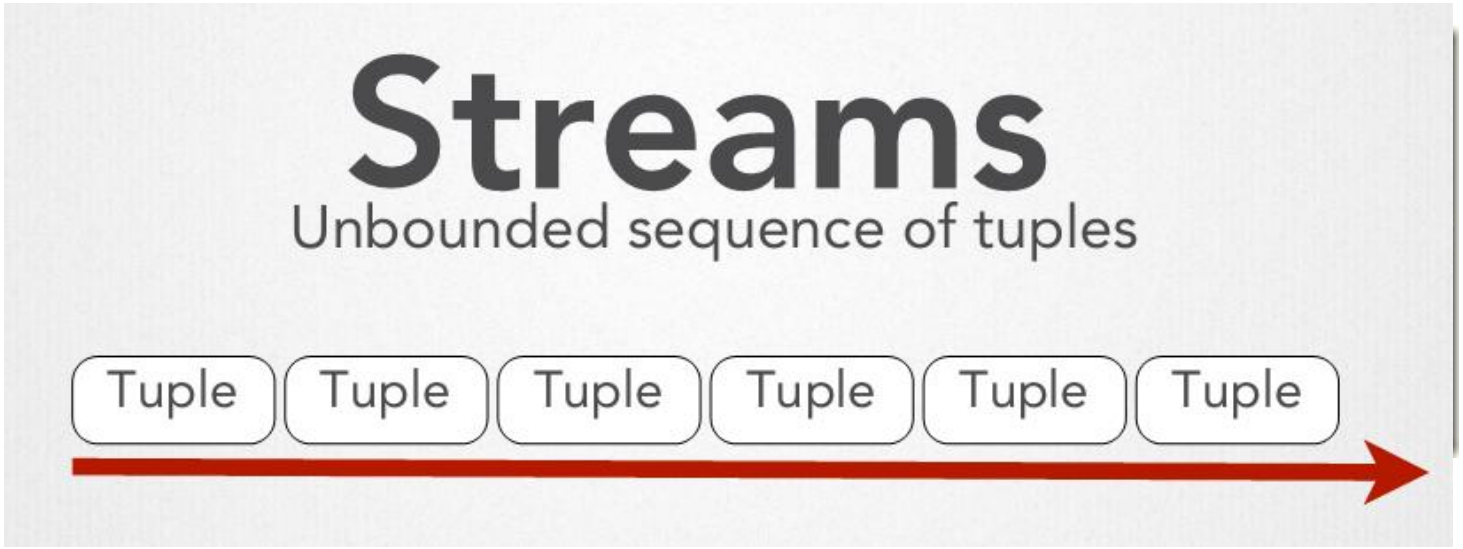
- 流计算（Stream processing）
- 实时分析（Real-time analytics）
- 连续计算（Continuous computation）
- 分布式远程过程调用（Distributed RPC）
- 在线机器学习（Online machine learning）
- 更多…

Storm主要特点

- **简单的编程模型**：Storm降低了进行实时处理的复杂性。
- **支持各种编程语言**：默认支持Clojure、Java、Ruby和Python。要增加对其他语言的支持，只需实现一个简单的Storm通信协议即可。
- **容错性**：Storm会自动管理工作进程和节点的故障。
- **水平扩展**：计算是在多个线程、进程和服务器之间并行进行的。
- **可靠的消息处理**：Storm保证每个消息至少能得到一次完整处理。
- **快速**：系统的设计保证了消息能得到快速的处理。
- **本地模式**：Storm有一个“本地模式”，可以在处理过程中完全模拟Storm集群，这样可以快速进行开发和单元测试。
- **容易部署**：Storm集群易于部署，只需少量的安装和配置就可运行。

Storm设计思想

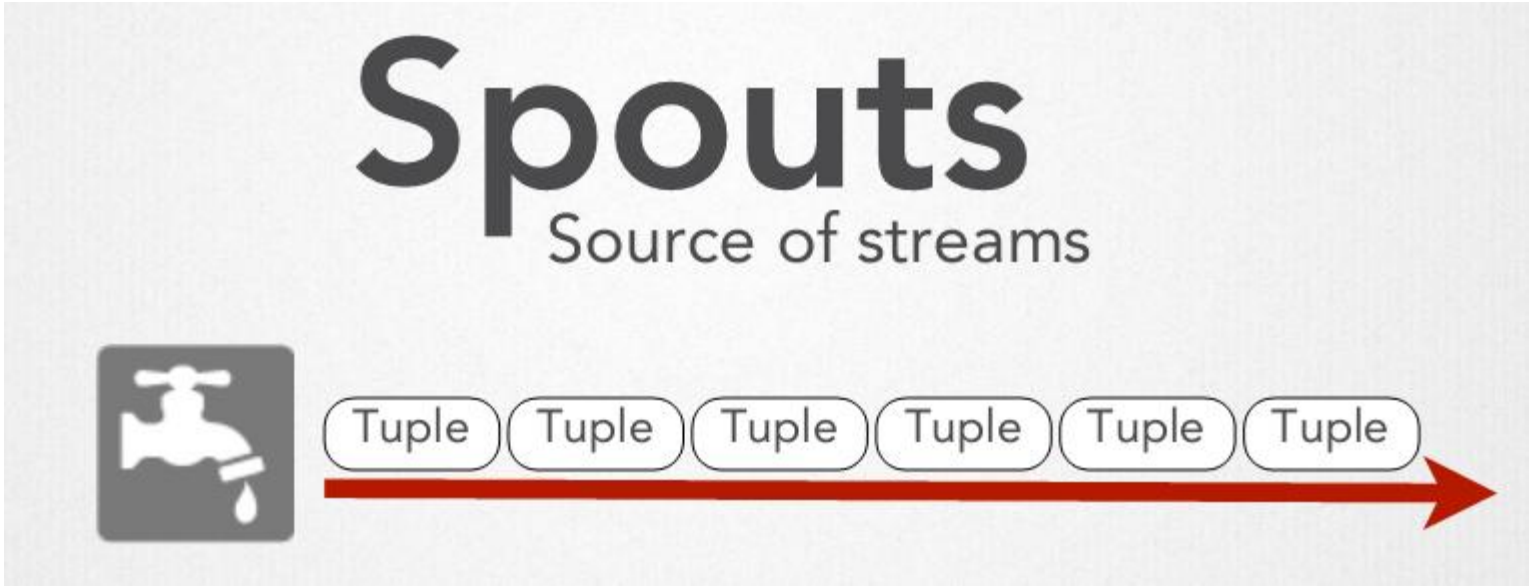
- Storm对于流Stream的抽象：流是一个不间断的无界的连续Tuple（元组，是元素有序列表）。



Stream消息流，是一个没有边界的Tuple序列，这些Tuples会被以一种分布式的方式并行地创建和处理。

Storm设计思想

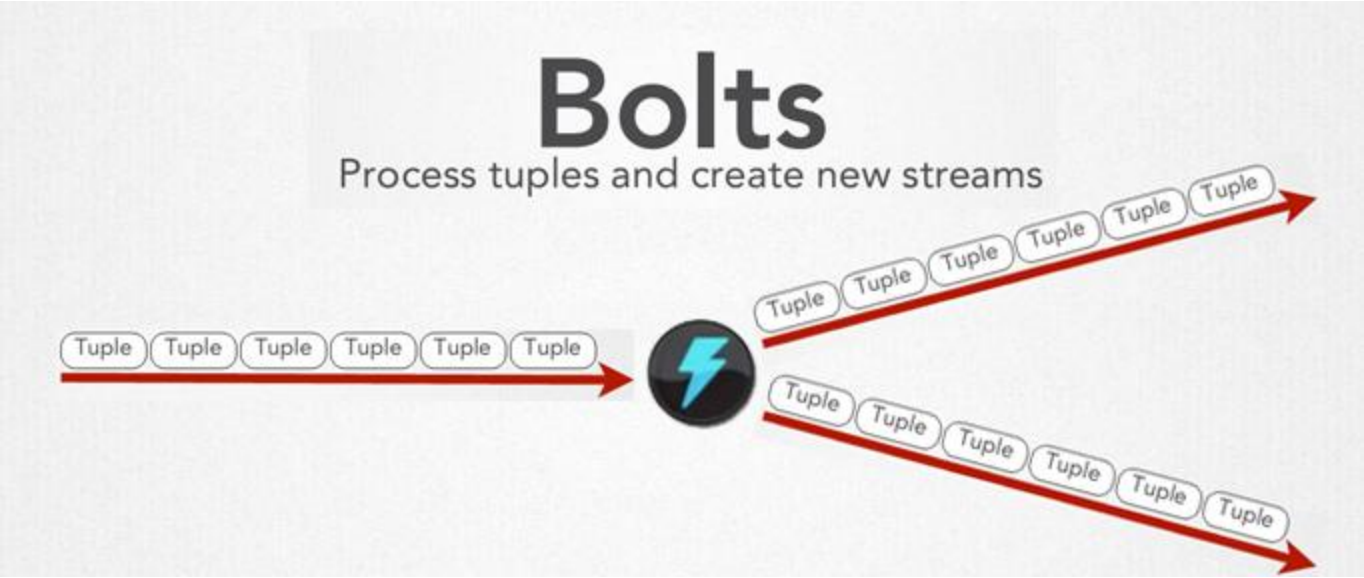
- Storm认为每个Stream都有一个源头，它将这个源头抽象为Spouts。



Spouts流数据源，它会从外部读取流数据并发出Tuple。

Storm设计思想

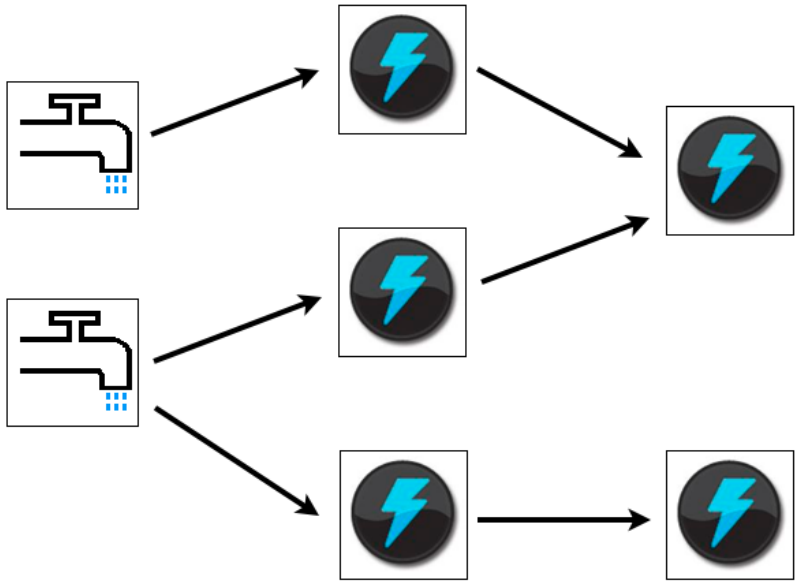
- Storm将流的中间状态转换抽象为Bolts，Bolts可以处理Tuples，同时它也可以发送新的流给其他Bolts使用。



Bolts消息处理者，所有的消息处理逻辑被封装在Bolts里面，处理输入的数据流并产生输出的新数据流，可执行过滤，聚合，查询数据库等操作。

Storm设计思想

- 为了提高效率，在Spout源接上多个Bolts处理器。Storm将这样的无向环图抽象为Topology（拓扑）。



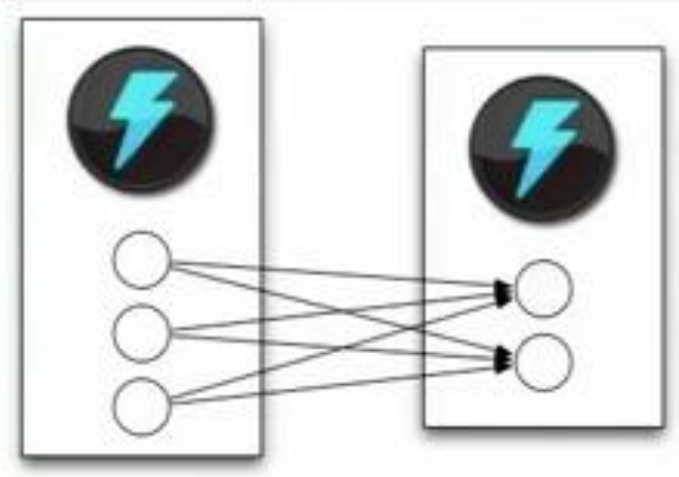
Topology是Storm中最高层次的抽象概念，它可以被提交到Storm集群执行，一个拓扑就是一个流转换图。图中的边表示Bolt订阅了哪些流。当Spout或者Bolt发送元组到流时，它就发送元组到每个订阅了该流的Bolt上进行处理。

Storm设计思想

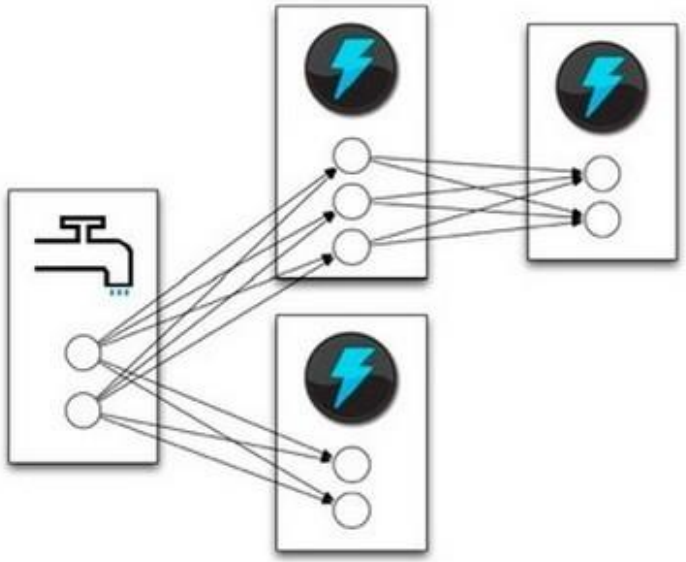
- Topology实现：Storm中拓扑定义仅仅是一些Thrift结构体（Thrift是基于二进制的高性能的通讯中间件），这样一来就可以使用其他语言来创建和提交拓扑。
- Tuple实现：一个Tuple就是一个值列表。列表中的每个value都有一个name，并且该value可以是基本类型，字符类型，字节数组等，也可以是其他可序列化的类型。
- 拓扑的每个节点都要说明它所发射出的元组字段的name，其他节点只需要订阅该name就可以接收数据。

Storm设计思想

- Stream groupings（消息分发策略）：定义一个Stream应该如何分配给Bolts，解决两个组件（Spout和Bolt）之间发送tuple元组的问题。
- Task（任务）：每一个Spout和Bolt会被当作很多task在整个集群里面执行,每一个task对应到一个线程。

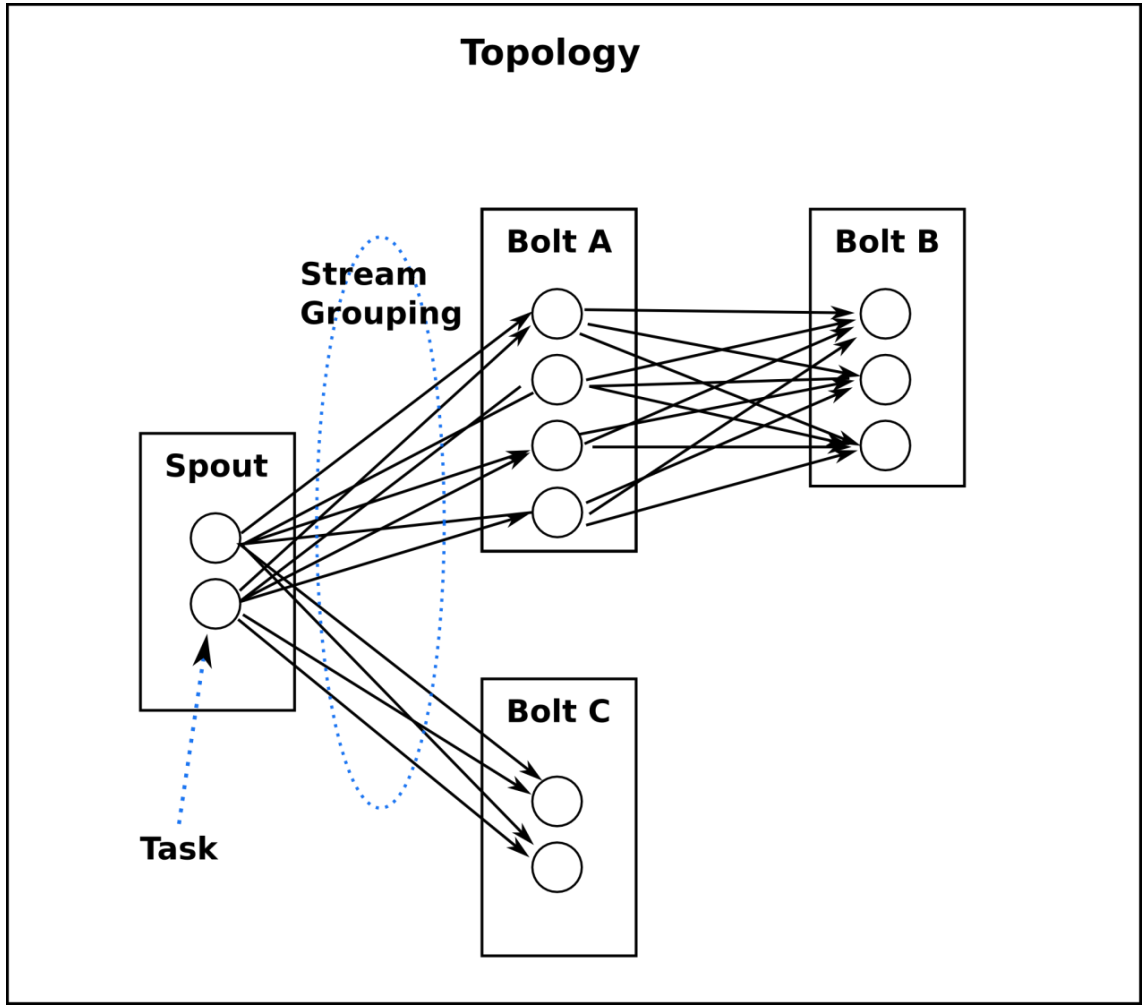


Stream groupings示意图



Task示意图

Storm设计思想



一个Topology的完整示意图

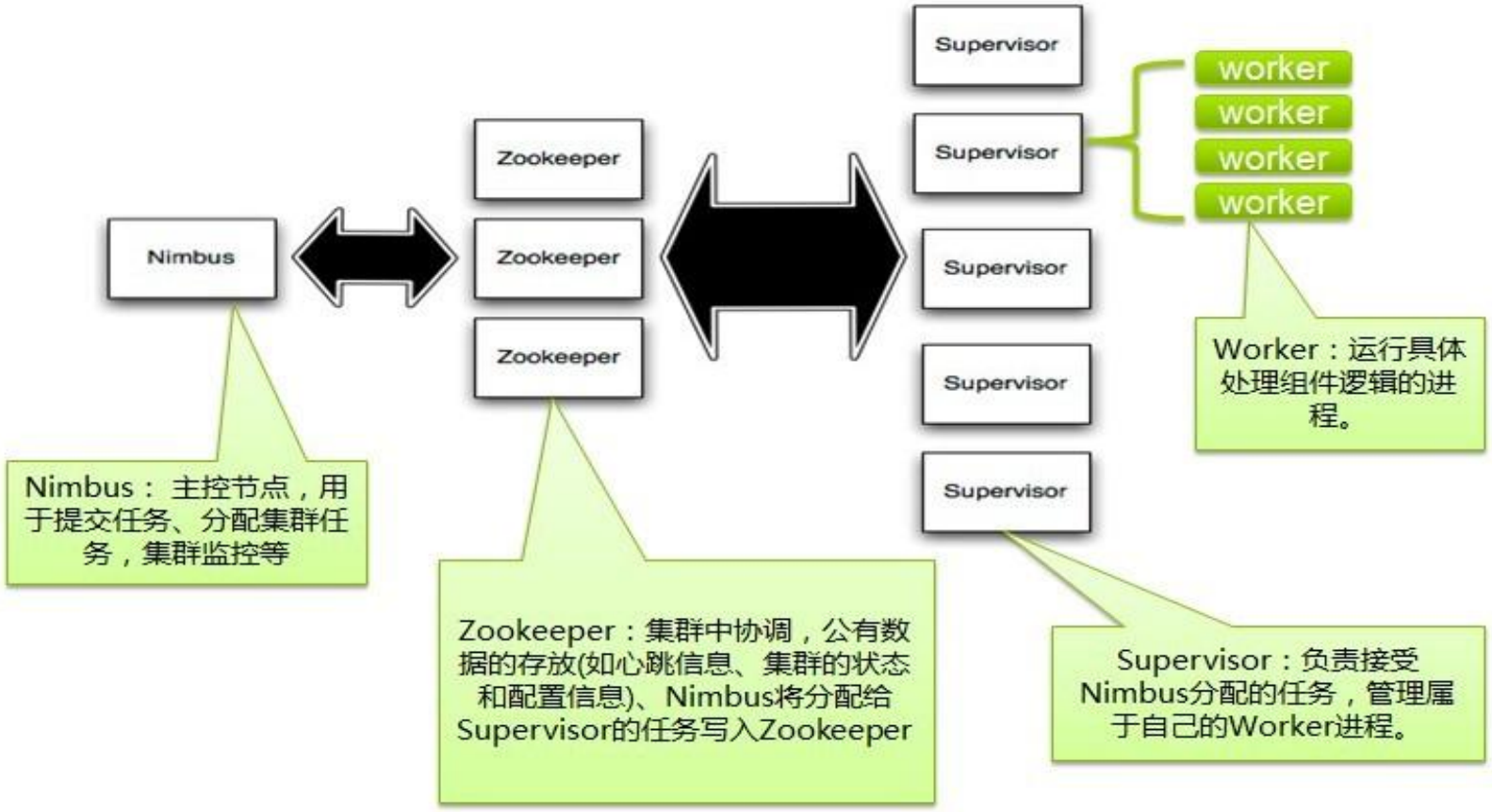
Storm框架设计

- Storm集群表面类似Hadoop集群。
- 在Hadoop上运行的是“MapReduce jobs”，在Storm上运行的是“Topologies”。两者大不相同，一个关键不同是一个MapReduce的Job最终会结束，而一个Topology永远处理消息（或直到kill它）。

- Storm集群有两种节点：控制（Master）节点和工作者（Worker）节点。
- 控制节点运行一个称之为“Nimbus”的后台程序，负责在集群范围内分发代码、为worker分配任务和故障监测。
- 每个工作者节点运行一个称之“Supervisor”的后台程序，监听分配给它所在机器的工作，基于Nimbus分配给它的事情来决定启动或停止工作者进程。

Storm框架设计

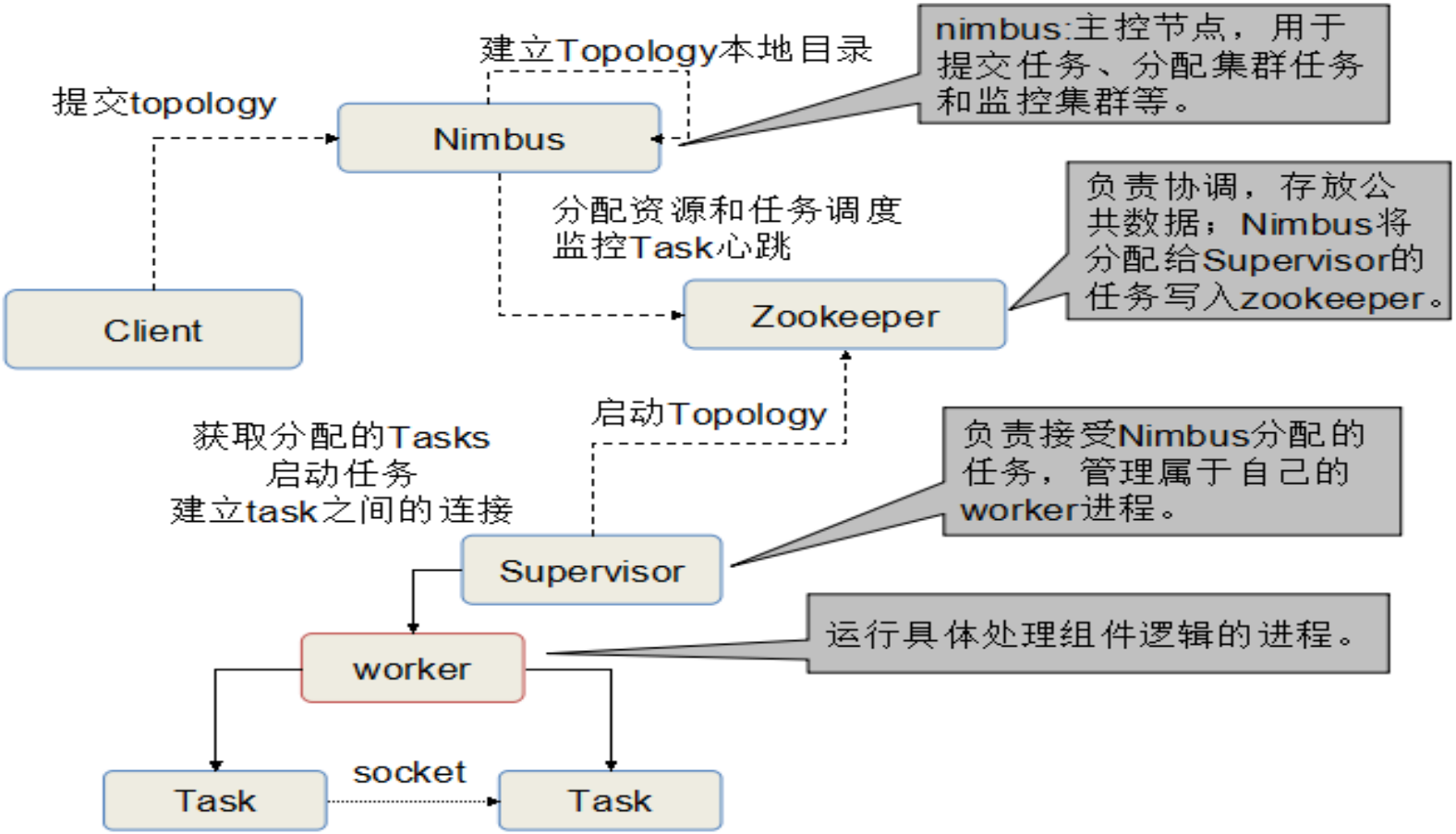
- 一个Zookeeper集群负责Nimbus和多个Supervisor之间的所有协调工作（一个完整的拓扑可能被分为多个子拓扑并由多个supervisor完成）。



Storm框架设计

- Nimbus后台程序和Supervisor后台程序都是快速失败（fail-fast）和无状态的，所有状态维持在Zookeeper或本地磁盘。
- 这种设计中master并没有直接和worker通信，而是借助中介Zookeeper，这样一来可以分离master和worker的依赖，将状态信息存放在zookeeper集群内以快速回复任何失败的一方。
- 这意味着你可以kill杀掉nimbus进程和supervisor进程，然后重启，它们将恢复状态并继续工作，这种设计使Storm极其稳定。

Storm框架设计



Storm工作流程示意图

Storm实例

单词统计

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("sentences", new RandomSentenceSpout(), 5);
builder.setBolt("split", new SplitSentence(), 8)
        .shuffleGrouping("sentences");
builder.setBolt("count", new WordCount(), 12)
        .fieldsGrouping("split", new Fields("word"));
```

Bolt通过订阅Tuple的name值来接收数据

- 编程模型非常简单，通过Topology定义整个处理逻辑。
- Topology中定义了一个Spout和两个处理消息的Bolt。

Storm实例

单词统计

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("sentences", new RandomSentenceSpout(), 5);
builder.setBolt("split", new SplitSentence(), 8)
        .shuffleGrouping("sentences");
builder.setBolt("count", new WordCount(), 12)
        .fieldsGrouping("split", new Fields("word"));
```

- Shuffle Grouping是随机分组，表示Tuple会被随机的分发给Bolt。
- Fields Grouping是按字段分组，保证具有相同field值的Tuple会分发给同一个Task进行统计，保证统计的准确性。

Storm实例

SplitSentence

```
public static class SplitSentence extends ShellBolt implements IRichBolt {

    public SplitSentence() {
        super("python", "splitsentence.py");
    }

    @Overrid
    public v
        decl
    }

    @Overrid
    public M
        retu
    }
}
```

```
import storm

class SplitSentenceBolt(storm.BasicBolt):
    def process(self, tup):
        words = tup.values[0].split(" ")
        for word in words:
            storm.emit([word])
```

Storm实例

WordCount

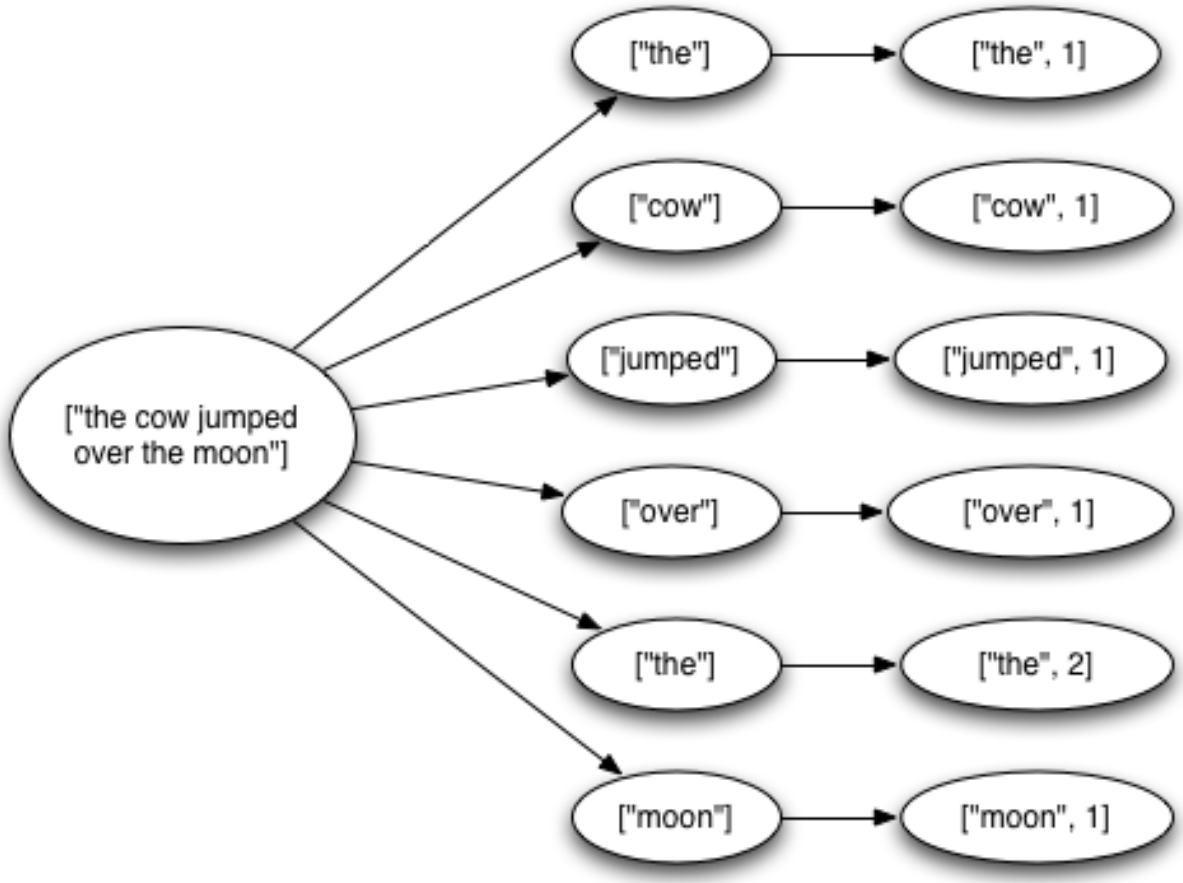
```
public static class WordCount extends BaseBasicBolt {
    Map<String, Integer> counts = new HashMap<String, Integer>();

    @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"));
    }
}
```

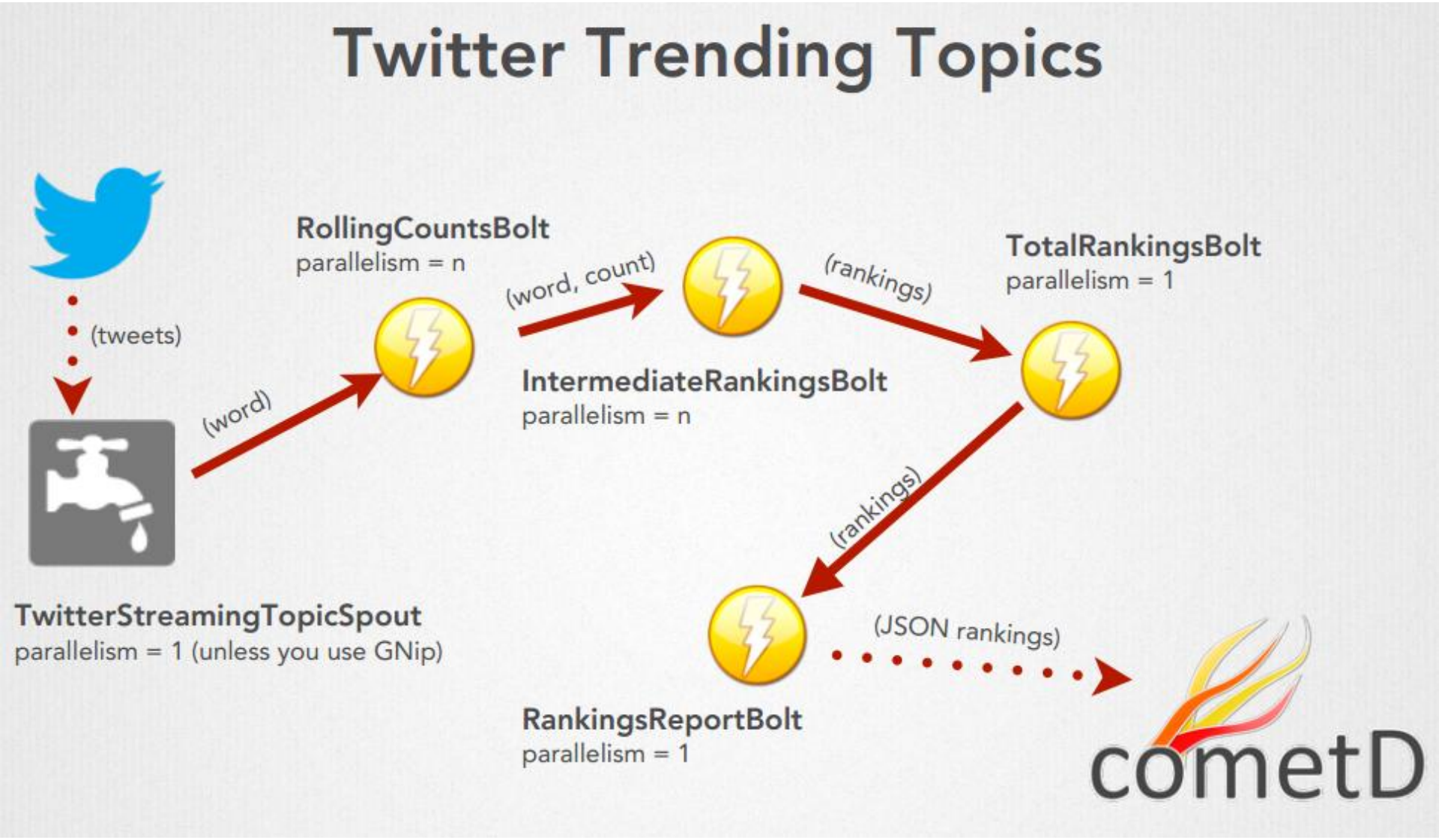
Storm实例

- 每个从spout发送出来的消息（英文句子）都会触发很多的task被创建。
- Bolts将句子分解为独立的单词， 然后发射这些单词。
- 最后，实时的输出每个单词以及它出现过的次数。



一个句子经单词统计后的统计结果示意图

Storm实例


Twitter Trending Topics
(tweets)
RollingCountsBolt
parallelism = n
(word)
(word, count)
IntermediateRankingsBolt
parallelism = n
(rankings)
TotalRankingsBolt
parallelism = 1
(rankings)
TwitterStreamingTopicSpout
parallelism = 1 (unless you use GNip)
(JSON rankings)
RankingsReportBolt
parallelism = 1
cometD

Storm应用

使用Storm的公司和项目

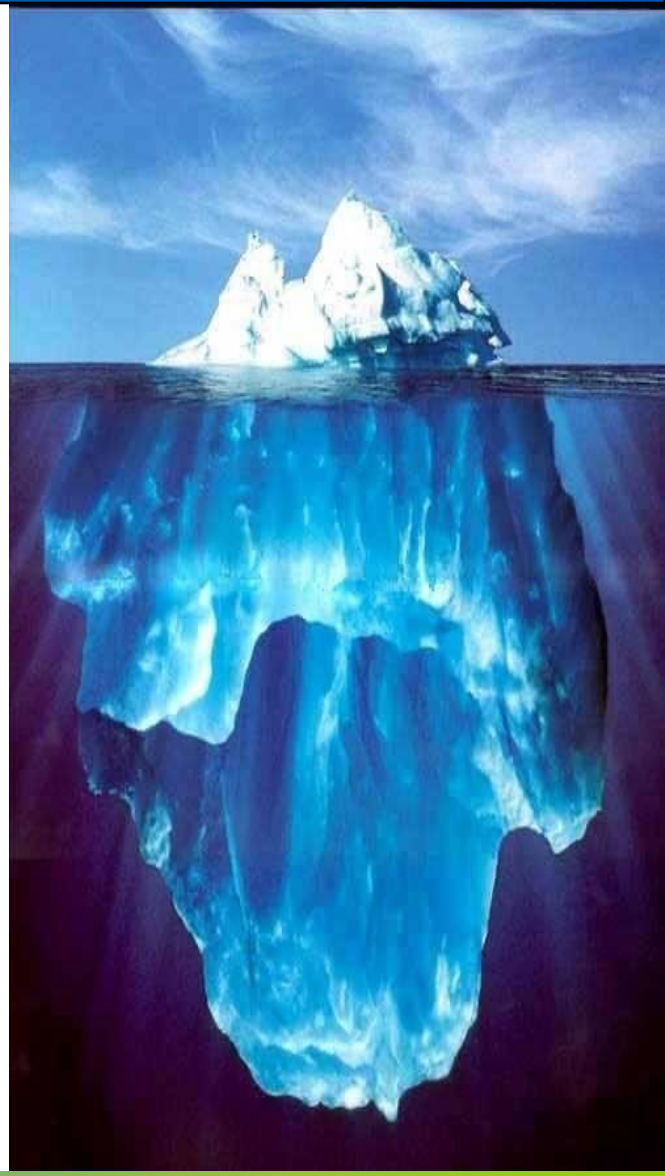
Storm应用

- 淘宝、阿里巴巴将流计算广泛应用于业务监控、广告推荐、买家实时数据分析等场景。

淘宝数据部新架构

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料



流计算框架汇总

- **IBM InfoSphere Streams**：商业级高级计算平台，帮助用户开发的应用程序快速摄取、分析和关联来自数千个实时源的信息。http://www-03.ibm.com/software/products/cn/zh/infosphere-streams/
- **IBM StreamBase**：IBM开发的另一款商业流计算系统，在金融部门和政府部门使用。http://www.streambase.com

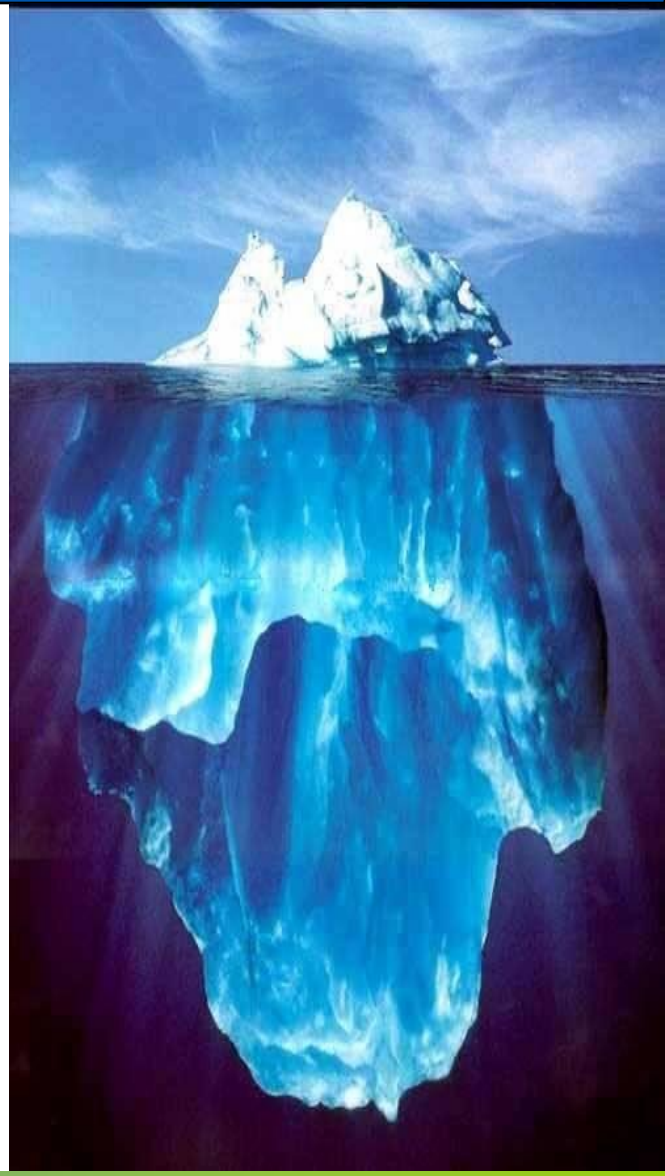
- **Twitter Storm**：免费、开源的分布式实时计算系统，可简单、高效、可靠地处理大量的流数据 http://storm-project.net/
- **Yahoo! S4（Simple Scalable Streaming System）**：开源流计算平台，是通用的、分布式的、可扩展的、分区容错的、可插拔的流式系统。http://incubator.apache.org/s4/

- **Facebook Puma**：Facebook使用Puma和Hbase相结合来处理实时数据。
- **DStream**：百度正在开发的属于百度的通用实时数据流计算系统。
- **银河流数据处理平台**：淘宝开发的通用流数据实时计算系统。
- **Super Mario**：基于erlang语言和zookeeper模块开发的高性能数据流处理框架。

- **Hstream**、**Esper**、**SQLstream**等…

课程提要

- 什么是流计算
- 流计算处理流程
- 流计算应用实例
- 流计算框架 – Twitter Storm
- 流计算框架汇总
- 参考资料



网上资料

- 关于流计算的文章
  - 对互联网海量数据实时计算的理解
    http://www.cnblogs.com/panfeng412/archive/2011/10/28/2227195.html
  - Beyond MapReduce：谈2011年风靡的数据流计算系统
    http://www.programmer.com.cn/9642/

- Twitter Storm
  - http://storm-project.net/ （Storm官方网站）
  - https://github.com/nathanmarz/storm （Storm的GitHub主页，有完善的Wiki）
  - http://xumingming.sinaapp.com （徐明明，GitHub上Storm的核心贡献者，博客中提供了不少关于Storm的文章）
  - http://blog.linezing.com （量子恒道提供的Storm入门教程）

主讲教师和助教

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

助教：赖明星

单位：厦门大学计算机科学系数据库实验室2011级硕士研究生（导师：林子雨）
E-mail: mingxinglai@gmail.com
个人主页：http://mingxinglai.com

欢迎访问《大数据技术基础》2013班级网站：http://dblab.xmu.edu.cn/node/423
本讲义PPT存在配套教材《大数据技术基础》，请到上面网站下载。

Department of Computer Science, Xiamen University, September, 2013