# 27-storm

**Storm**

– Basic Concepts

– Quick Start

– Other Issues

**Objectives**

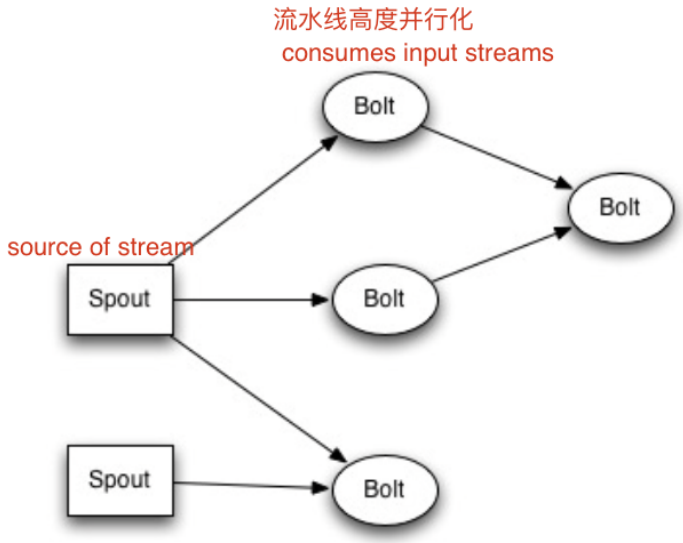
– 能够针对高性能计算需求，设计并实现基于Storm的的流数据处理方案

storm从来不停，zookeeper来管理整个集群，支撑可拓展性、容错性

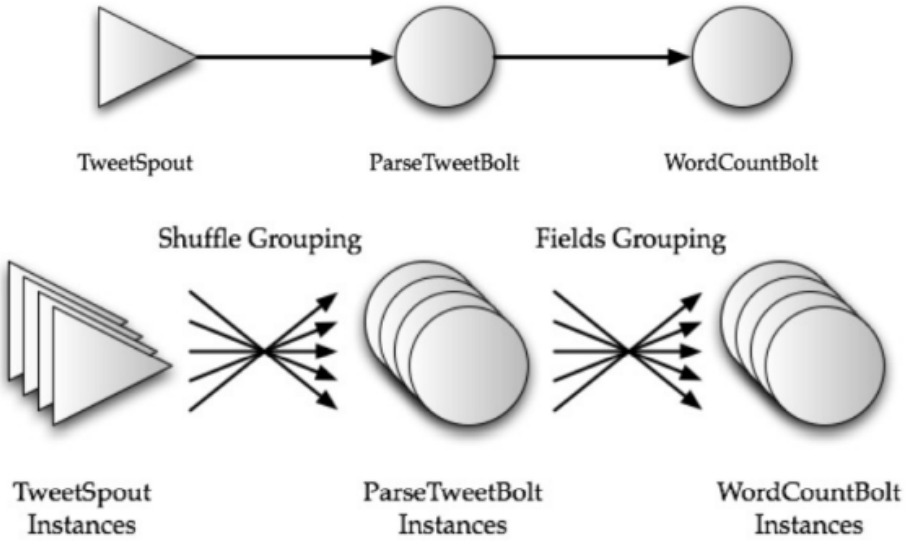
## Why use Apache Storm ?

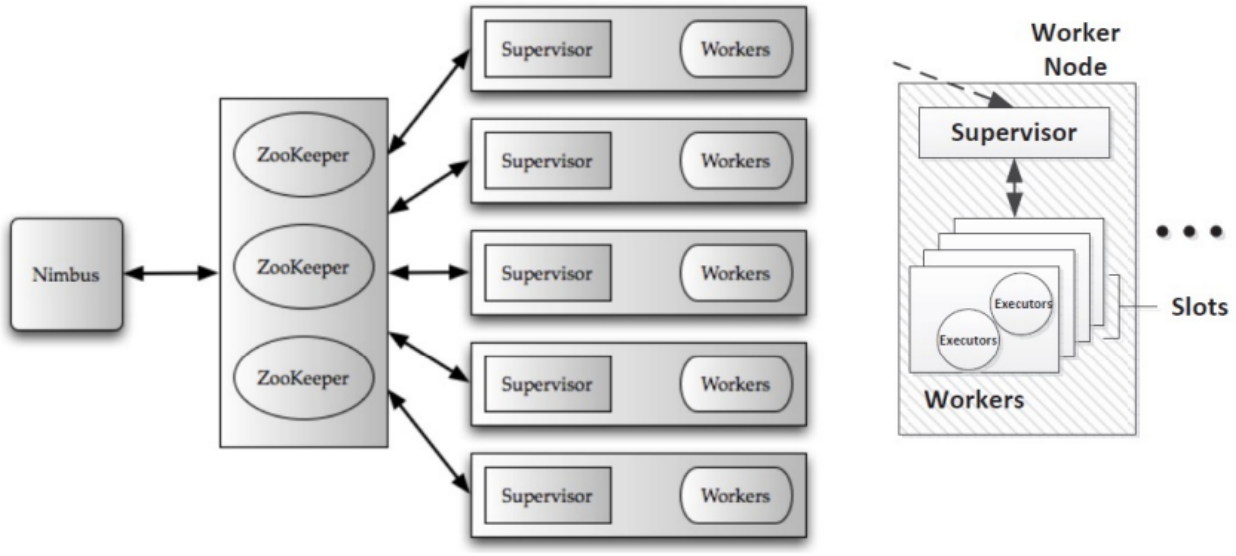
REin
REliable, INtelligent & Scalable Systems

- Apache Storm is a free and open source distributed realtime computation system.
  - Apache Storm makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing.
  - Apache Storm is simple, can be used with any programming language, and is a lot of fun to use!
- Apache Storm has many use cases:
  - realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Apache Storm is fast: a benchmark clocked it at over **a million tuples processed per second per node**. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate. zookeeper 集群管理
- Apache Storm integrates with the queueing and database technologies you already use. 数据来源：队列，图数据库…，因此可以和他们做集成
  - An Apache Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed. Read more in the tutorial


流水线高度并行化
consumes input streams
Bolt
Bolt
source of stream
Spout
Bolt
Spout
Bolt

WordCount


TweetSpout
ParseTweetBolt
WordCountBolt
Shuffle Grouping
Fields Grouping
TweetSpout Instances
ParseTweetBolt Instances
WordCountBolt Instances




## Storm Cluster的组成部分

Supervisor向Zookeeper报告自己状态和任务进度，Zookeeper告诉Nimbus可以调用什么机器，Zookeeper负责资源管理，Nimbus负责任务调度

总结：一个分布式系统最好有master，并将资源管理和任务调度分开来管！

Hadoop/Spark/Storm其实在结构上都是类似的思想，但是侧重的方面不一样

没有master的好处是任何一台机器挂了都没有关系，但是难以维护统一的结果

The basic primitives Storm provides for doing stream transformations are “spouts” and “bolts”.

Bolt计算，并行处理

- Let's look at the ExclamationTopology definition from storm-starter:

```
TopologyBuilder builder = new TopologyBuilder();
builder.setSpout("words", new TestWordSpout(), 10);
builder.setBolt("exclaim1", new ExclamationBolt(), 3)
        .shuffleGrouping("words");
builder.setBolt("exclaim2", new ExclamationBolt(), 2)
        .shuffleGrouping("exclaim1");
```

- This topology contains a spout and two bolts.
  - The spout emits words, and each bolt appends the string "!!!" to its input.
  - The nodes are arranged in a line: the spout emits to the first bolt which then emits to the second bolt.
  - If the spout emits the tuples ["bob"] and ["john"], then the second bolt will emit the words ["bob!!!!!!"] and ["john!!!!!!"].

10表示实例的数量

- ExclamationBolt appends the string "!!!" to its input.

```
public static class ExclamationBolt implements IRichBolt {
 OutputCollector _collector;

 @Override public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
   _collector = collector;
 }

 @Override public void execute(Tuple tuple) {
   _collector.emit(tuple, new Values(tuple.getString(0) + "!!!"));
   _collector.ack(tuple);
 }

 @Override public void cleanup() { }

 @Override public void declareOutputFields(OutputFieldsDeclarer declarer) {
  declarer.declare(new Fields("word"));
 }
 @Override public Map<String, Object> getComponentConfiguration() {
  return null;
 }
}
```

ack确认收到，cleanup和prepare功能相反

## Storm与Spark、Hadoop三种框架对比

1.Storm是最佳的**流式计算框架**，Storm由Java和Clojure写成，Storm的优点是**全内存计算**，所以它的定位是**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。

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）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。

2.Spark是一个**基于内存计算的开源集群计算系统**，目的是**更快速的进行数据分析**。
Spark使用Scala开发，类似于Hadoop MapReduce的通用并行计算框架，Spark基于Map Reduce算法实现的分布式计算，拥有Hadoop MapReduce所具有的优点，但不同于MapReduce的是Job中间输出和结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的Map Reduce的算法。
Spark的适用场景：
1）多次操作特定数据集的应用场合
Spark是基于内存的迭代计算框架，适用于需要多次操作特定数据集的应用场合。需要反复操作的次数越多，所需读取的数据量越大，受益越大，数据量小但是计算密集度较大的场合，受益就相对较小。
2）粗粒度更新状态的应用
由于RDD的特性，Spark不适用那种异步细粒度更新状态的应用，例如Web服务的存储或者是增量的Web爬虫和索引。就是对于那种增量修改的应用模型不适合。
总的来说Spark的适用面比较广泛且比较通用。

3.Hadoop是实现了MapReduce的思想，将数据**切片**计算来处理大量的**离线数据**。
Hadoop处理的数据必须是已经存放在HDFS上或者类似HBase的数据库中，所以Hadoop实现的时候是通过移动计算到这些存放数据的机器上来提高效率。
Hadoop的适用场景：
1）海量数据的离线分析处理
2）大规模Web信息搜索
3）数据密集型并行计算

顺提一下ch30: **hive是基于Hadoop的一个数据仓库工具**，可以将结构化的数据文件映射为一张数据库表，并提供完整的sql查询功能，可以将sql语句转换为MapReduce任务进行运行，这套SQL 简称HQL。

简单来说：
Hadoop适合于离线的批量数据处理适用于对实时性要求极低的场景
Storm适合于实时流数据处理，实时性方面做得极好
Spark是内存分布式计算框架，试图吞并Hadoop的Map-Reduce批处理框架和Storm的流处理框架，但是Spark已经做得很不错了，批处理方面性能优于Map-Reduce，但是流处理目前还是弱于Storm，产品仍在改进之中