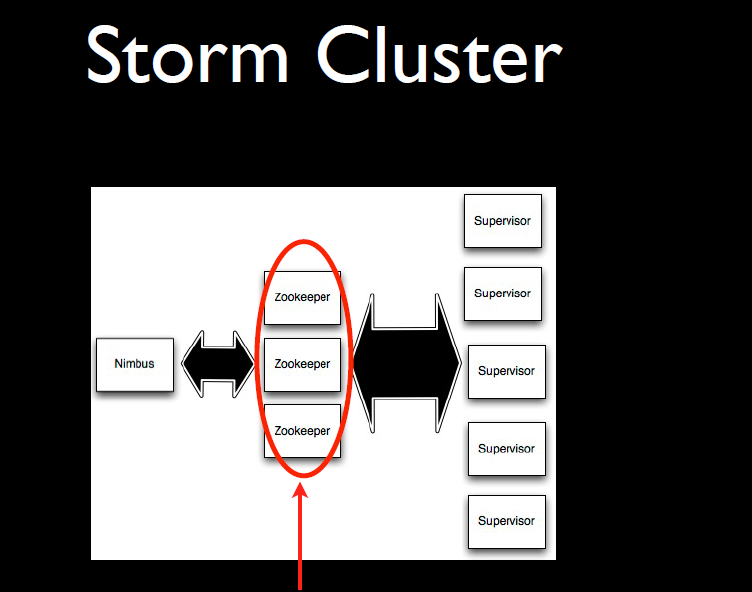
# Storm介绍—专注实时计算（stream流处理）

## 一、基本概念

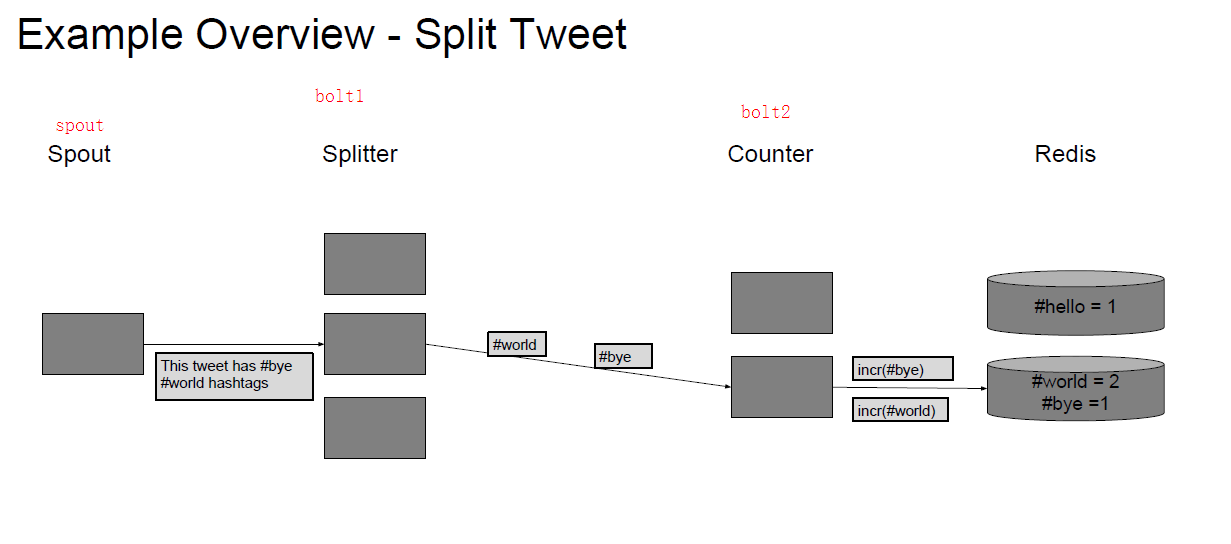
### 1.1 集群角色介绍

|  |  |  |
| --- | --- | --- |
| **名称** | **描述** | **数量** |
| **Nimbus** | **管理节点** | **1** |
| **SuperVisor** | **计算节点** | **N** |
| **Zookeeper** | **消息集群**  **注意：storm里面的zookeper并不是用来传递计算数据，而是传递节点的运行状态，所以负载很低，大多数情况下zookeeper只要一个节点就满足要求，为了宕机安全考虑，建议设置2个几点。** | **集群** |



### 1.2 计算概念介绍

|  |  |  |
| --- | --- | --- |
| **名称** | **描述** | **数量** |
| **Stream** | **数据流** | **N** |
| **Spouts** | **数据流的源头，产生处理数据** | **N** |
| **Bolt（博尔特）** | **处理数据** | **N** |
| **Topology(拓扑) storm计算任务，类似与一个Mapreduce任务 1** | | |

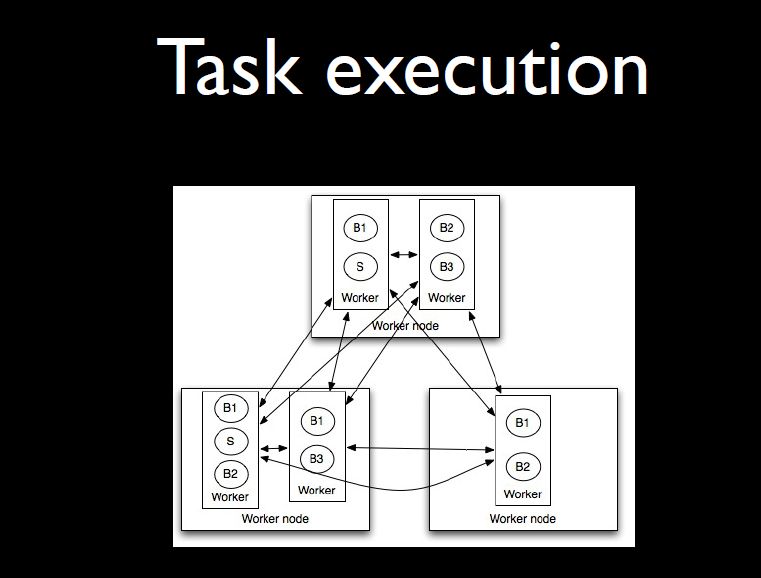


### 1.3storm任务启动过程

**以上图为例：我们要启动一个storm计算任务，从tweet接收消息，然后对文本切分（split），再对单词进行统计（count），最后入redis库**

|  |
| --- |
| ToPolpgyBuilder builder = new TopologyBuilder();  Builder.setSpout(“spout”,new KestrelSpout(),5) ;  Builder.setBolt(“split”, new Split(), 8).shuffleGrouping(“spout”);  Builder.setBolt(“count”, new Count(), 8).fieldsGrouping(“split” , new Fields(“word”) );  StormSubmitter.submitTopology(“word-count”,conf,builder.createTopology() ); |





## 二、storm特性

### 2.1、分布式

### 2.2、fault-tolerance（保证数据不会丢失）

Spout生成一个消息后，会为这个消息生成一棵处理树，一旦超过一定时间这个消息还没有正确处理，那么spout就认为数据处理失败，会重新发送这个消息。

1. 效率很低？

00 2个字节

### 2.3、数据分发规则

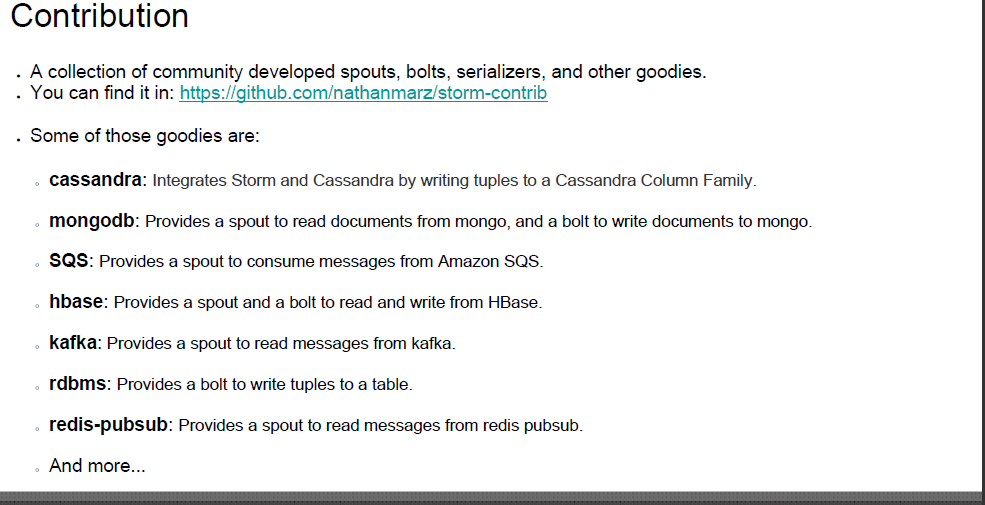
优先选择本地节点分发最好

|  |  |  |
| --- | --- | --- |
| **Shuffle grouping:** | 随机分发 |  |
| **Fields grouping** | 类似mapreduce，map的默认分发，对字段计算hash值，进行分发  **注：这个很有用，可以用来做聚合** |  |
| **All grouping** | 所有的都分发 |  |
| **Global grouping** | 发给taskid最小的 |  |
| **Direct grouping** | **类似流平台的分发，可以根据条件，分发给指定的流** |  |

### 2.4、事务

**以1条轨迹入库为例，轨迹数据入hdfs文件、轨迹索引如索引表（多张索引表）、提取联系统计，这是一个完整的事务过程，理论上：三个操作都必须同步。Storm提供了一个特殊的事务机制，来保证数据一致提交。**

### 2.5、与各大平台的对接



Hadoop 2.0 storm kafka

## 三、storm vs流平台

|  |  |  |
| --- | --- | --- |
| **特性** | Storm |  |
| **分布式** | YES |  |
| **数据格式** | 类json：  支持int、float、double、string、bool  支持：map、list、set |  |
| **容错机制** | 有（重启、挂死，不会造成数据丢失） |  |
| **全分发规则** | 支持多种分发规则 |  |
| **实例定位** | 1. 无法定位实例运行在哪台机器 |  |
| **碰撞处理** | 1、由于无法定位实例，一台计算节点上，可能会有多个实运行，对于大内存比对这种需要加载大量碰撞数据的处理，支持不够方便 |  |
| **语言支持** | 几乎所有语言都支持   1. Java 2. 提供thrift接口 3. 提供标准输入输出接口 |  |
| **平台支持** | 支持从众多数据平台读取、写数据  Kafka、Hbase、mongodb、redis、rdbm、hadoop（oracle、mysql）等 |  |
| **处理性能** | 未测试 |  |

## 四、kafka + storm

### 4.1、kafka介绍—专注消息队列

#### 4.1.1 设计理念

Linkln

5

1. 分布式

每秒400M吞吐一台普通节点

2、关注大吞吐量，而不是别的特性

3、针对实时场景

#### 4.1.2 角色介绍

|  |  |  |
| --- | --- | --- |
| **Broker** | Kafka里面的基本节点 | N |
| **Zookeeper** | 传递集群管理消息、配置信息 | 集群 |



#### 4.1.3 概念介绍

|  |  |  |
| --- | --- | --- |
| **Topic** | Kafka里面消息队列的名字  ：我们可以想像成表名 | 1 |
| **Producer** | 数据生产者：像kafka的topic传递数据 | N |
| **Consumer** | 数据消费这：从kafka的topic获取数据 | N |

#### 4.1.4 特性介绍

|  |  |  |
| --- | --- | --- |
| **平台支持** | 与hadoop完美融合 |  |
| **吞吐量** | 一台普通的boker节点可以达到数百兆数据的吞吐 |  |
| **持久化** | 磁盘持久化 |  |

### 4.2、kafka编写consumer