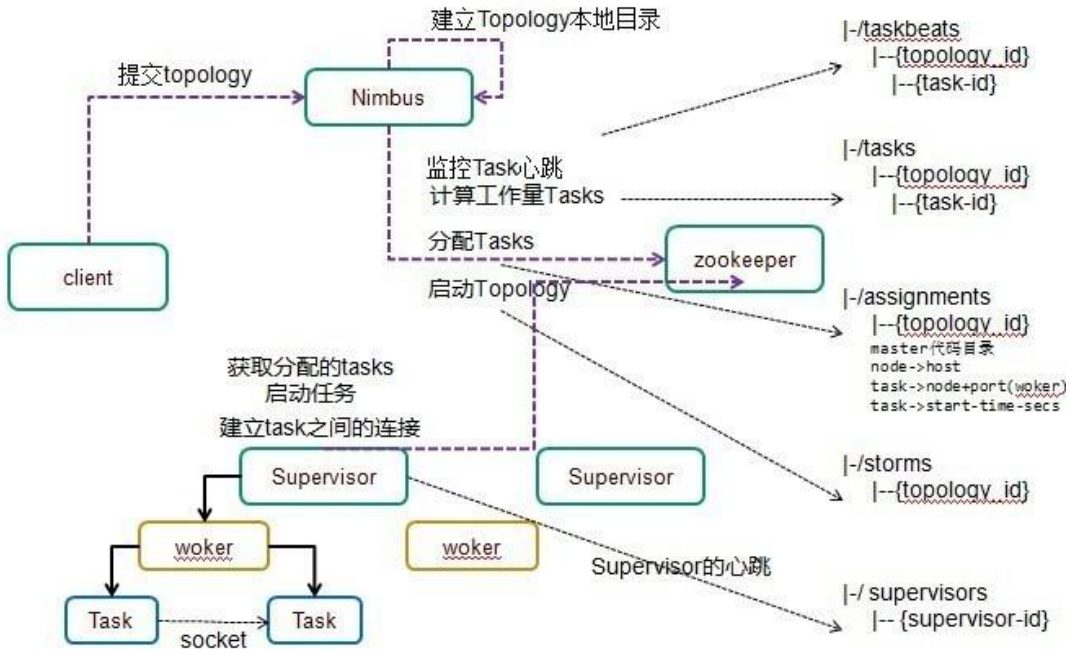
1. 说说Storm集群架构?

Apache Storm的主要亮点是，它是一个容错，快速，无单点故障的分布式系统。我们可以根据需要在多个节点上安装Apache Storm，以增加集群的容量。下图描述了Apache Storm集群设计：



Apache Storm有两种类型的节点，Nimbus(主节点)和Supervisor(工作节点)。Nimbus是Apache Storm的核心组件。Nimbus的主要工作是运行Storm拓扑。Nimbus分析拓扑并收集要执行的任务。然后，它将任务分配给可用的Supervisor。

Supervisor将有一个或多个工作进程。Supervisor将任务委派给工作进程。工作进程将根据需要产生尽可能多的执行器并运行任务。Storm使用内部分布式消息传递系统来进行Nimbus和管理程序之间的通信。

Nimbus(主节点): Nimbus是Storm集群的主节点，负责在所有工作节点之间分发数据，向工作节点分配任务和监视故障。

Supervisor(工作节点):执行指令的节点被称为Supervisors。Supervisor有多个工作进程，它管理工作进程以完成由nimbus分配的任务。

Worker process(工作进程)：工作进程将执行与特定拓扑相关的任务。工作进程不会自己运行任务，而是创建执行器并要求他们执行特定的任务。工作进程将有多个执行器。

Executor(执行器)：执行器是工作进程产生的单个线程。执行器运行一个或多个任务，但仅用于特定的Spout或Bolt。

Task(任务)：任务执行实际的数据处理。所以，它是一个Spout或Bolt。

ZooKeeper：Apache ZooKeeper是一种分布式一致性服务。Nimbus是无状态的，它依赖于ZooKeeper来监视工作节点的状态。ZooKeeper帮助 Supervisor与Nimbus交互。它负责维持Nimbus，Supervisor的状态。

2. Storm工作原理是什么?

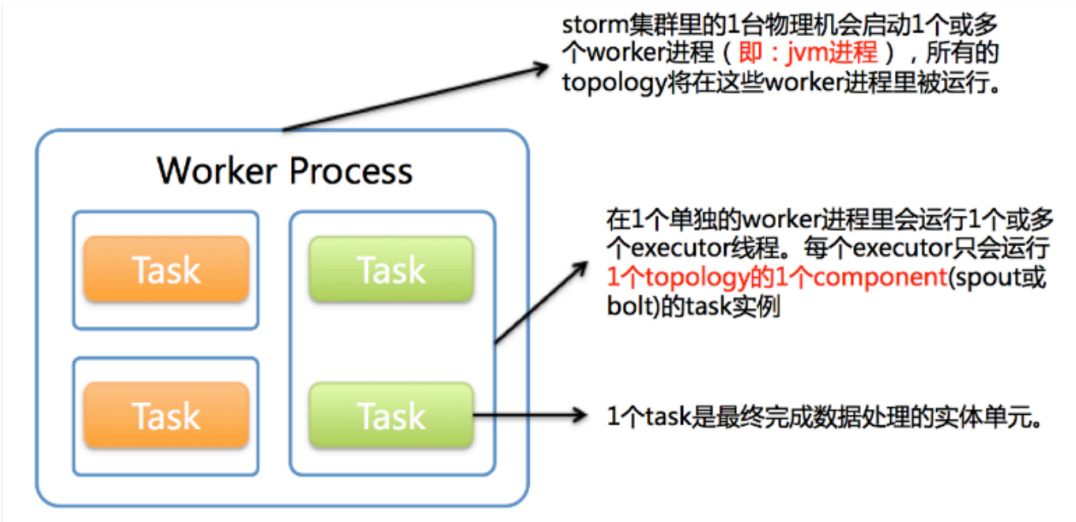
在Storm中,一个实时应用的计算任务被打包作为Topology发布，这同Hadoop的MapReduce任务相似。但是有一点不同的是:在Hadoop中，MapReduce任务最终会执行完成后结束；而在Storm中，Topology任务一旦提交后永远不会结束，除非你显示去停止任务。计算任务Topology是由不同的Spouts和Bolts，通过数据流（Stream）连接起来的图。一个Storm在集群上运行一个Topology时，主要通过以下3个实体来完成Topology的执行工作：

(1). Worker（进程）

(2). Executor（线程）

(3). Task

下图简要描述了这3者之间的关系：



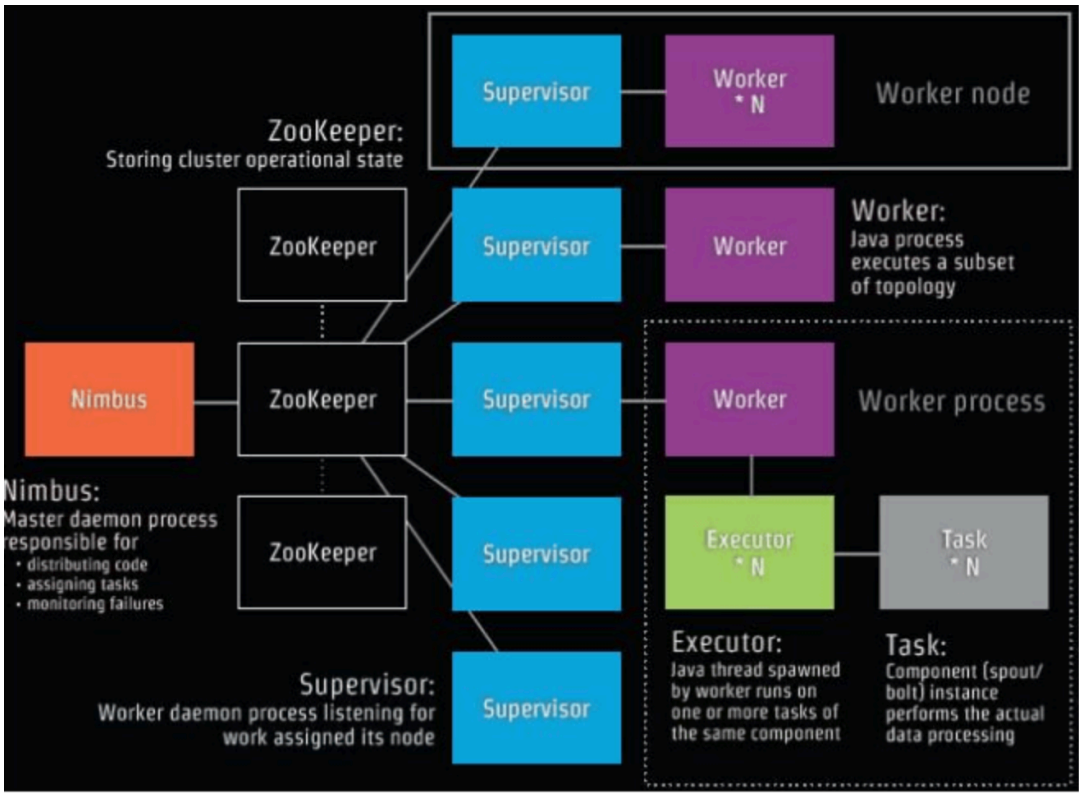
1个worker进程执行的是1个topology的子集（注：不会出现1个worker为多个topology服务）。1个worker进程会启动1个或多个executor线程来执行1个topology的component(spout或bolt)。因此，1个运行中的topology就是由集

群中多台物理机上的多个worker进程组成的。

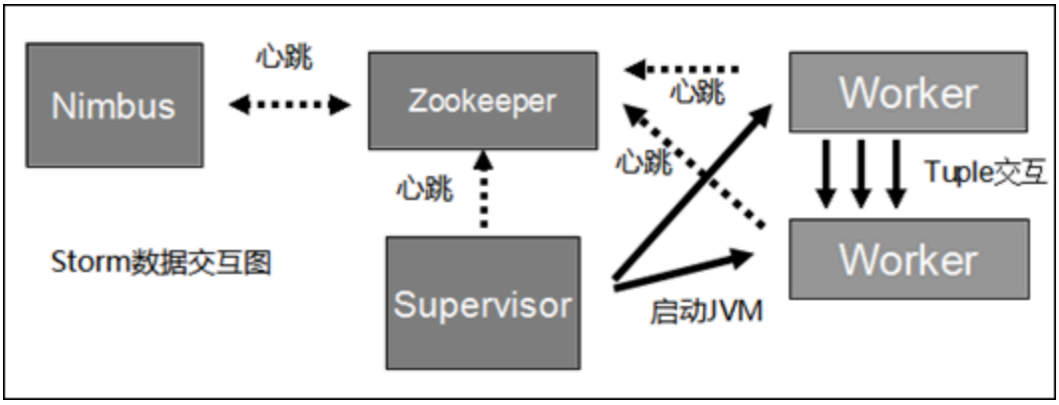
executor是1个被worker进程启动的单独线程。每个executor只会运行1个topology的1个component(spout或bolt)的task（注：task可以是1个或多个，storm默认是1个component只生成1个task，executor线程里会在每次循环里顺序调用所有task实例）。

task是最终运行spout或bolt中代码的单元（注：1个task即为spout或bolt的1个实例，executor线程在执行期间会调用该task的nextTuple或execute方法）。topology启动后，1个component(spout或bolt)的task数目是固定不变的，但该component使用的executor线程数可以动态调整（例如：1个executor线程可以执行该component的1个或多个task实例）。这意味着，对于1个component存在这样的条件：#threads<=#tasks（即：线程数小于等于task数目）。默认情况下task的数目等于executor线程数目，即1个executor线程只运行1个task。

总体的Topology处理流程图为：



下图是Storm的数据交互图，可以看出两个模块Nimbus和Supervisor之间没有直接交互。状态都是保存在Zookeeper上，Worker之间通过Netty传送数据。Storm与Zookeeper之间的交互过程，暂时不细说了。重要的一点:storm所有的元数据信息保存在Zookeeper中！



3. 流的模式是什么？默认是什么？

流是Storm中的核心抽象。一个流由无限的元组序列组成，这些元组会被分布式并行地创建和处理。通过流中元组包含的字段名称来定义这个流。

每个流声明时都被赋予了一个ID。只有一个流的Spout和Bolt非常常见，所以`OutputFieldsDeclarer`提供了不需要指定ID来声明一个流的函数(Spout和Bolt都需要声明输出的流)。这种情况下，流的ID是默认的“default”。

4. 对于mapreduce如何理解？

MapReduce的编程模型的原理是：利用一个输入key/value对集合来产生一个输出的key/value对集合。MapReduce库的用户用两个函数表达这个计算：Map和Reduce。用户自定义的Map函数接受一个输入的key/value对值，然后产生一个中间key/value对值的集合。MapReduce库把所有具有相同中间key值的中间value值集合在一起后传递给Reduce函数。用户自定义的Reduce函数接受一个中间key的值和相关的一个value值的集合。Reduce函数合并这些value值，形成一个较小的value值的集合。

5. Storm的特点和特性是什么？

1. 编程简单：开发人员只需要关注应用逻辑，而且跟Hadoop类似，Storm提供的编程原语也很简单
2. 高性能，低延迟：可以应用于广告搜索引擎这种要求对广告主的操作进行实时响应的场景。
3. 分布式：可以轻松应对数据量大，单机搞不定的场景
4. 可扩展： 随着业务发展，数据量和计算量越来越大，系统可水平扩展
5. 容错：单个节点挂了不影响应用
6. 消息不丢失：保证消息处理

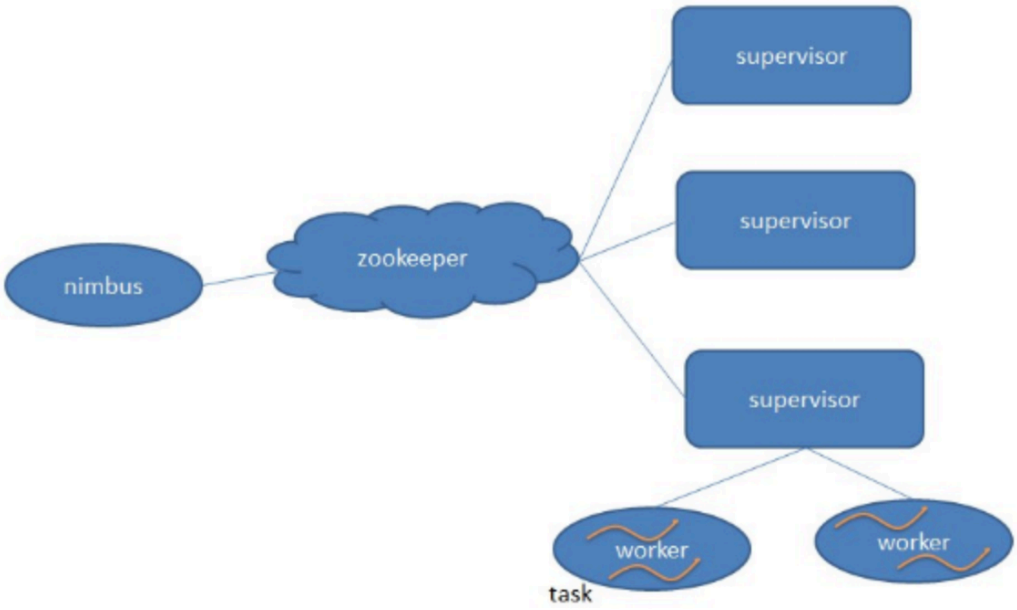
不过Storm不是一个完整的解决方案。使用Storm时你需要关注以下几点：

i. 如果使用的是自己的消息队列，需要加入消息队列做数据的来源和产出的代码

ii. 需要考虑如何做故障处理：如何记录消息队列处理的进度，应对Storm重启，挂掉的场景

iii. 需要考虑如何做消息的回退：如果某些消息处理一直失败怎么办?

6. Storm组件有哪些?



1. Nimbus（主节点）：负责资源分配和任务调度。

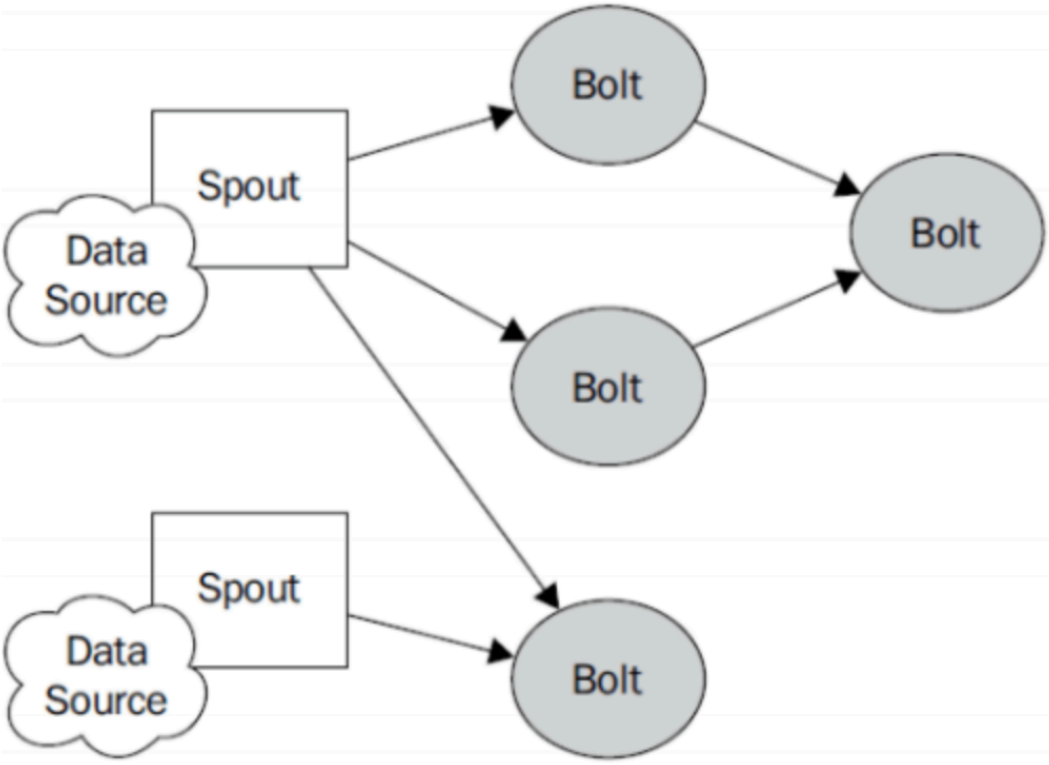
2. Supervisor（从节点）：负责接受nimbus分配的任务，启动和停止属于自己管理的worker进程。---通过配置文件设置当前 supervisor上启动多少个worker。worker的数量根据端口号来的!

3. Worker（进程）：运行具体处理组件逻辑的进程（其实就是一个JVM）。Worker运行的任务类型只有两种，一种是Spout任务，一种是Bolt任务。

4. Task（线程）：worker中每一个spout/bolt的线程称为一个task. 在storm0.8之后，task不再与物理线程对应，不同spout/bolt的task可能会共享一个物理线程，该线程称为executor。task=线程=executor

5. Zookeeper（分布式协调服务）:保存任务分配的信息、心跳信息、元数据信息。

7. 说说Storm编程模型?



1. Topology：Storm中运行的一个实时应用程序的名称。（拓扑）
2. Spout：在一个topology中获取源数据流的组件。 通常情况下spout会从外部数据源中读取数据，然后转换为topology内部的源数据。
3. Bolt：接受数据然后执行处理的组件,用户可以在其中执行自己想要的操作。
4. Tuple：一次消息传递的基本单元，理解为一组消息就是一个Tuple。
5. Stream：表示数据的流向。

8. Storm怎么保证一致性?

Storm是一个分布式的流处理系统，利用anchor和ack机制保证所有tuple都被成功处理。如果tuple出错，则可以被重传，但是如何保证出错的tuple只被处理一次呢？Storm提供了一套事务性组件Transaction Topology，用来解决这个问题。

Transactional Topology目前已经不再维护，由Trident来实现事务性topology，但是原理相同。

9.sparkStreaming和Storm比较

对比点	Storm	Spark Streaming
实时计算模型	纯实时，来一条数据，处理一条数据	准实时，对一个时间段内的数据收集起来，作为一个RDD，再处理
实时计算延迟度	毫秒级	秒级
吞吐量	低	高
事务机制	支持完善	支持，但不够完善
健壮性 / 容错性	ZooKeeper，Acker，非常强	Checkpoint，WAL，一般
动态调整并行度	支持	不支持

10.SparkStreaming与Storm的应用场景?

对于Storm来说：

1、建议在那种需要纯实时，不能忍受1秒以上延迟的场景下使用，比如实时金融系统，要求纯实时进行金融交易和分析

2、此外，如果对于实时计算的功能中，要求可靠的事务机制和可靠性机制，即数据的处理完全精准，一条也不能多，一条也不能少，也可以考虑使用Storm

3、如果还需要针对高峰低峰时间段，动态调整实时计算程序的并行度，以最大限度利用集群资源（通常是在小型公司，集群资源紧张的情况），也可以考虑用Storm

4、如果一个大数据应用系统，它就是纯粹的实时计算，不需要在中间执行SQL交互式查询、复杂的transformation算子等，那么用Storm是比较好的选择

对于Spark Streaming来说：

1、如果对上述适用于Storm的三点，一条都不满足的实时场景，即，不要求纯实时，不要求强大可靠的事务机制，不要求动态调整并行度，那

么可以考虑使用Spark Streaming

2、考虑使用Spark Streaming最主要的一个因素，应该是针对整个项目进行宏观的考虑，即，如果一个项目除了实时计算之外，还包括了离线批处理、交互式查询等业务功能，而且实时计算中，可能还会牵扯到高延迟批处理、交互式查询等功能，那么就应该首选Spark生态，用Spark Core开发离线批处理，用Spark SQL开发交互式查询，用Spark Streaming开发实时计算，三者可以无缝整合，给系统提供非常高的可扩展性。

11.storm如何完成单词计数?

- 构造一个 Spout，为下游 Bolt 作业提供数据源
- 构造一个 Bolt，处理上游流向数据，进行单词切分
- 构造一个 Bolt，处理上游 Bolt ，进行单词计数
- 将 Spout 、Bolt 组装起来，构建成一个拓扑（Topology）
- 将 Topology 提交到 storm 集群，等待结果

参考：https://www.jianshu.com/p/2c205ccd6210

12.storm有哪些优化方案?

1. 硬件配置的优化
2. 代码层面的优化
3. topology 并行度的优化
4. Storm 集群配置参数和 topology 运行参数的优化