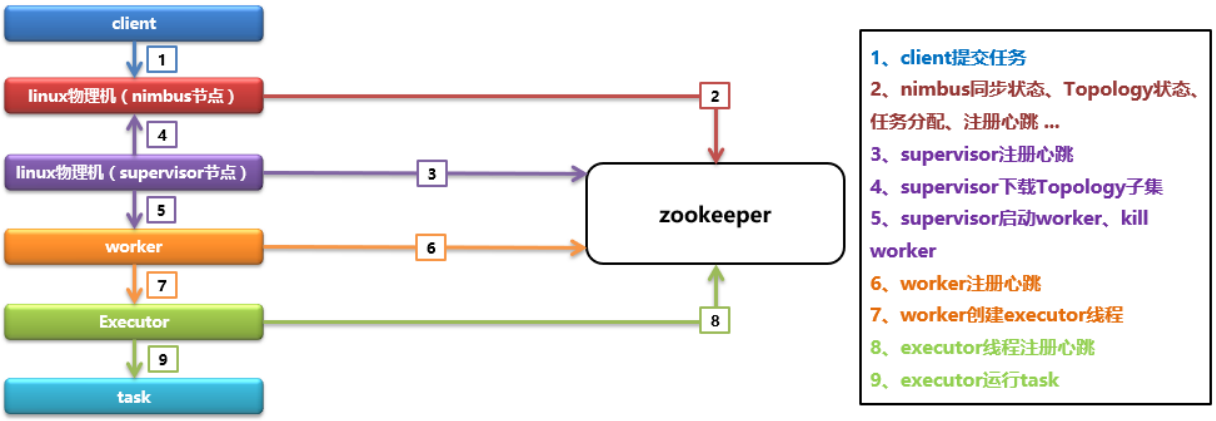
1 运行架构



storm集群由一个主节点(nimbus)和一个或者多个工作节点（supervisor）组成。

nimbus：storm 的主节点，类似于Hadoop中的jobtracker，管理、协调和监控在集群上运行的Topology。包括Topology的发布，事件处理失败时重新指派任务

supervisor：每个工作节点运行Supervisor守护进程，负责监听工作节点上已经分配的主机作业，启动和停止Nimbus已经分配的工作进程。supervisor会定时从zookeeper获取拓补信息topologies、任务分配信息assignments及各类心跳信息，以此为依据进行任务分配。在supervisor同步时，会根据新的任务分配情况来启动新的worker或者关闭旧的worker并进行负载均衡。

zookeeper：storm主要使用zookeeper来协调集群中的状态信息，比如任务的分配情况、
worker的状态、supervisor之间的nimbus的拓扑质量。nimbus和supervisor节点的通信主
要是结合zookeeper的状态变更通知和监控通知来处理的。

Worker：具体处理Spout/Bolt逻辑的进程，根据提交的拓扑中conf.setNumWorkers()方法定义分配每个拓扑对应的worker数量，Storm会在每个Worker上均匀分配任务，**一个Worker只能执行一个Topology的任务子集**。worker 进程会占用固定的可由配置进行修改的内存空间（默认768M）。

2 任务分配

1.client：提交Topology

2.nimbus：这个角色所做的操作相对较多，具体如下：

a.会把提交的jar包放到nimbus所在服务器的storm.local.dir/nimbus/inbox目录下

b.submitTopology方法会负责Topology的处理；包括检查集群是否有active节点、配置文件是否正确、是否有重复的Topology名称、各个bolt/spout名是否使用相同的id等。

c.nimbus任务分配，根据Topology中的定义，给spout/bolt设置task的数目，并分配对应的task-id。

d.nimbus在zookeeper上创建workerbeats目录，要求每个worker定时向nimbus汇报

e.将分配好的任务写入到zookeeper的storm/assignments目录下,此时任务提交完毕。

f.将Topology信息写入到zookeeper的/storm/storms目录。

3.supervisor

a.建立Topology的本地目录，storm.local.dir/supervisor/stormdist/topology-uuid

该目录包括三个文件：

stormjar.jar --从nimbus/inbox目录拷贝

stormcode.ser --此Topology对象的序列化

stormconf.ser --此Topology的配置文件序列化

b.定期扫描zookeeper上的storms目录，看看是否有新的任务，有就下载。

c.删除本地不需要的Topology

d.根据nimbus指定的任务信息启动worker

4.worker

a.查看需要执行的任务，根据任务id分辨出spout/bolt任务

b.计算出所代表的spout/bolt会给哪些task发送信息

c.worker根据分配的tasks信息，启动多个executor线程，同时实例化spout、bolt、acker等组件，此时，等待所有connections（worker和其它机器通讯的网络连接）启动完毕执行spout任务或者blot任务。

注：

1、 在slot充沛的情况下，能够保证所有topology的task被均匀的分配到整个机器的所有机器上

2、 在slot不足的情况下，它会把topology的所有的task分配到仅有的slot上去，这时候其实不是理想状态，所以在nimbus发现有多余slot的时候，它会重新分配topology的task分配到空余的slot上去以达到理想状态。

3、 在没有slot的时候，它什么也不做