Storm的配置文件一般存放在$STORM_HOME/conf下，通常名为storm.yaml，它符合yaml格式要求。Storm的配置参数对任务的稳定运行以及吞吐率至关重要，这里介绍一下storm常见的配置项参数。

一. storm基本配置

- **storm.local.dir**: nimbus 和supervisor进程存储一些状态信息（conf或者jars）的本地路径，需要每台storm node单独创建该目录并保证该目录有正确的读写权限;
- **storm.log4j2.conf.dir**： log4j的配置目录;
- **storm.zookeeper.servers**: storm严重依赖zookeeper存储状态信息，要保证zookeeper的高可用，最好设置多个zk地址;
- **storm.zookeeper.port**: 默认2181;
- **storm.zookeeper.root**: 在zookeeper中存储的根目录，如果多个storm集群公用一个zk集群，需要修改其根目录名称即可;
- **storm.session.timeout**: 默认20s，nimbus和supervisor和zk的session超时时间，如果log中常用sessiontimeout错误，考虑增加其值或者修改gc参数。该值并不能无限设置，zk有自己的最大session时间（默认20 ticktime）；
- **storm.zookeeper.connection.timeout**：连接超时时间；
- **storm.zookeeper.retry.times**: 默认5，执行zk操作重试次数；
- **storm.zookeeper.retry.interval**: 默认1000，即间隔1s；
- **storm.zookeeper.retry.intervalceiling.millis**: 300000 （5分钟）执行重试的时间间隔最长时间；
- **storm.messaging.transport**：”backtype.storm.messaging.netty.Context” task之间的消息传输协议，默认使用netty传输；
- **storm.cluster.mode**: “distributed” storm集群模式；
- **storm.id**：运行中拓扑的id,由storm name和一个唯一随机数组成；
- **storm.local.mode.zmq**：Local模式下是否使用ZeroMQ作消息系统，如果设置为false则使用java消息系统。默认为false；
  注：Storm严重依赖zookeeper，而且zk在分布式使用中扮演了非常重要的角色。

二. nimbus相关设置

- **storm.nimbus.retry.times**： 5 nimbus操作的重试次数
- **storm.nimbus.retry.interval.millis**： 2s 重试间隔
- **storm.nimbus.retry.intervalceiling.millis**: 60000 最大重试时间 10分钟
- **nimbus.seeds**: [] 用于leader nimbus发现的nimbus hosts 列表，解决nimbus的单点故障问题，代替了原来的nimbus.host 配置
- **nimbus.thrift.port**: 6627 nimbus工作的thrift端口，客户端上传jar和提交拓扑的端口（nimbus的thrift监听端口）
- **nimbus.thrift.threads**: 64 nimbus thrift 线程数目
- **nimbus.thrift.max_buffer_size**: 1048576 1m
- **nimbus.childopts**: “-Xmx1024m” nimbus java 进程jvm设置
- **nimbus.task.timeout.secs**：30 与task没有心跳时多久nimbus可以认为该task已经死掉并且可以重新分配该task
- **nimbus.supervisor.timeout.secs**: 60 一分钟没有心跳 nimbus可以认为该supervisor已经dead，不会分配新的work
- **nimbus.monitor.freq.secs**: 10 nimbus多久查询下supervisor心跳信息并且重新分配工作。注意当一台机子曾经挂掉，nimbus会立即采取一些操作
- **nimbus.reassign**：当发现task失败时nimbus是否重新分配执行。默认为真，不建议修改。
- **nimbus.cleanup.inbox.freq.secs**: 600 多久时间启动清理inbox文件的线程
- **nimbus.inbox.jar.expiration.secs**: 3600 一个小时 jar过期时间
- **nimbus.code.sync.freq.secs**: 300 5分钟同步一次未执行的拓扑的代码
- **nimbus.task.launch.secs**: 120 用于task 第一次启动时的超时时间
- **nimbus.file.copy.expiration.secs**: 600 上传下载文件超时时间
- **nimbus.topology.validator**: “backtype.storm.nimbus.DefaultTopologyValidator” 拓扑验证，控制该拓扑是否可以执行
- **topology.min.replication.count**: 1 当nimbus seeds中该拓扑代码的备份达到最小数目时leader nimbus才可以执行拓扑动作。
- **topology.max.replication.wait.time.sec**: 60 当代码备份在nimbus list中达到topology.min.replication.count设置的最大等待时间，如果超时，不管有没有最小备份个数，都要执行该拓扑

三. supervisor相关配置

- **supervisor.slots.ports**: 设置当台机子上可跑的worker数目，每个worker对应一个port，通常情况下多少个cpu core就设置多少个worker，类似与hadoop中nodemanager中slot的设置
- **supervisor.childopts**: “-Xmx256m” supervisor jvm参数设置
- **supervisor.worker.start.timeout.secs**: 120 supervisor等待worker启动的最长时间
- **supervisor.worker.timeout.secs**: 30 worker的最长超时时间
- **supervisor.worker.shutdown.sleep.secs**: 1秒 supervisor shutdown worker需要等待的时间
- **supervisor.monitor.frequency.secs**: 3s检查一次worker的心跳确保是否要重启这些worker
- **supervisor.heartbeat.frequency.secs**: 5s supervisor和nimbus心跳的频率
- **supervisor.enable**: true supervisor是否要启动分配它的worker

四. worker 配置

- **worker.childopts**: “-Xmx768m”
- **worker.gc.childopts**: “” worker gc set 可以被topology.worker.gc.childopts.覆盖
- **worker.heartbeat.frequency.secs**: 1 worker 和supervisor的heartbeat时间
- **topology.worker.receiver.thread.count**: 1 每个worker设置的receiver 线程个数
- **task.heartbeat.frequency.secs**: 3s task向nimbus发送心跳的频率
- **task.refresh.poll.secs**: 10 多久和其他task同步连接（如果task重新分配，发往该task信息的那些task需要重练他们之间的连接）