**Storm配置项详解:**

|  |  |
| --- | --- |
| **配置项** | **配置说明** |
| storm.zookeeper.servers | ZooKeeper服务器列表 |
| storm.zookeeper.port | ZooKeeper连接端口 |
| storm.local.dir | storm使用的本地文件系统目录(必须存在并且storm进程可读写) |
| storm.cluster.mode | Storm集群运行模式([distributed|local]) |
| 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 |
| storm.zookeeper.root | ZooKeeper中Storm的根目录位置 |
| storm.zookeeper.session.timeout | 客户端连接ZooKeeper超时时间 |
| storm.id | 运行中拓扑的id,由storm name和一个唯一随机数组成。 |
| nimbus.host | nimbus服务器地址 |
| nimbus.thrift.port | nimbus的thrift监听端口 |
| nimbus.childopts | 通过storm-deploy项目部署时指定给nimbus进程的jvm选项 |
| nimbus.task.timeout.secs | 心跳超时时间，超时后nimbus会认为task死掉并重分配给另一个地址。 |
| nimbus.monitor.freq.secs | nimbus检查心跳和重分配任务的时间间隔.注意如果是机器宕掉nimbus会立即接管并处理。 |
| nimbus.supervisor.timeout.secs | supervisor的心跳超时时间,一旦超过nimbus会认为该supervisor已死并停止为它分发新任务. |
| nimbus.task.launch.secs | task启动时的一个特殊超时设置.在启动后第一次心跳前会使用该值来临时替代nimbus.task.timeout.secs. |
| 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.file.copy.expiration.secs | nimbus判断上传/下载链接的超时时间，当空闲时间超过该设定时nimbus会认为链接死掉并主动断开 |
| ui.port | Storm UI的服务端口 |
| drpc.servers | DRPC服务器列表，以便DRPCSpout知道和谁通讯 |
| drpc.port | Storm DRPC的服务端口 |
| supervisor.slots.ports | supervisor上能够运行workers的端口列表.每个worker占用一个端口,且每个端口只运行一个worker.通过这项配置可以调整每台机器上运行的worker数.(调整slot数/每机) |
| supervisor.childopts | 在storm-deploy项目中使用,用来配置supervisor守护进程的jvm选项 |
| supervisor.worker.timeout.secs | supervisor中的worker心跳超时时间,一旦超时supervisor会尝试重启worker进程. |
| supervisor.worker.start.timeout.secs | supervisor初始启动时，worker的心跳超时时间，当超过该时间supervisor会尝试重启worker。因为JVM初始启动和配置会带来的额外消耗，从而使得第一次心跳会超过supervisor.worker.timeout.secs的设定 |
| supervisor.enable | supervisor是否应当运行分配给他的workers.默认为true,该选项用来进行Storm的单元测试,一般不应修改. |
| supervisor.heartbeat.frequency.secs | supervisor心跳发送频率(多久发送一次) |
| supervisor.monitor.frequency.secs | supervisor检查worker心跳的频率 |
| worker.childopts | supervisor启动worker时使用的jvm选项.所有的”%ID%”字串会被替换为对应worker的标识符 |
| worker.heartbeat.frequency.secs | worker的心跳发送时间间隔 |
| task.heartbeat.frequency.secs | task汇报状态心跳时间间隔 |
| task.refresh.poll.secs | task与其他tasks之间链接同步的频率.(如果task被重分配,其他tasks向它发送消息需要刷新连接).一般来讲，重分配发生时其他tasks会理解得到通知。该配置仅仅为了防止未通知的情况。 |
| topology.debug | 如果设置成true，Storm将记录发射的每条信息。 |
| topology.optimize | master是否在合适时机通过在单个线程内运行多个task以达到优化topologies的目的. |
| topology.workers | 执行该topology集群中应当启动的进程数量.每个进程内部将以线程方式执行一定数目的tasks.topology的组件结合该参数和并行度提示来优化性能 |
| topology.ackers | topology中启动的acker任务数.Acker保存由spout发送的tuples的记录，并探测tuple何时被完全处理.当Acker探测到tuple被处理完毕时会向spout发送确认信息.通常应当根据topology的吞吐量来确定acker的数目，但一般不需要太多.当设置为0时,相当于禁用了消息可靠性,storm会在spout发送tuples后立即进行确认. |
| topology.message.timeout.secs | topology中spout发送消息的最大处理超时时间.如果一条消息在该时间窗口内未被成功ack,Storm会告知spout这条消息失败。而部分spout实现了失败消息重播功能。 |
| topology.kryo.register | 注册到Kryo(Storm底层的序列化框架)的序列化方案列表.序列化方案可以是一个类名,或者是com.esotericsoftware.kryo.Serializer的实现. |
| topology.skip.missing.kryo.registrations | Storm是否应该跳过它不能识别的kryo序列化方案.如果设置为否task可能会装载失败或者在运行时抛出错误. |
| topology.max.task.parallelism | 在一个topology中能够允许的最大组件并行度.该项配置主要用在本地模式中测试线程数限制. |
| topology.max.spout.pending | 一个spout task中处于pending状态的最大的tuples数量.该配置应用于单个task,而不是整个spouts或topology. |
| topology.state.synchronization.timeout.secs | 组件同步状态源的最大超时时间(保留选项,暂未使用) |
| topology.stats.sample.rate | 用来产生task统计信息的tuples抽样百分比 |
| topology.fall.back.on.java.serialization | topology中是否使用java的序列化方案 |
| zmq.threads | 每个worker进程内zeromq通讯用到的线程数 |
| zmq.linger.millis | 当连接关闭时,链接尝试重新发送消息到目标主机的持续时长.这是一个不常用的高级选项,基本上可以忽略. |
| java.library.path | JVM启动(如Nimbus,Supervisor和workers)时的java.library.path设置.该选项告诉JVM在哪些路径下定位本地库. |