SchedulerX 常见问题及解决办法

> 作者：黄晓萌（学仁）
> 创作日期：2019-11-14
> 专栏地址：【稳定大于一切】 PDF 格式：SchedulerX 常见问题及解决办法

目录

机器繁忙

应用管理，连接机器，如果机器是红色的【繁忙】，说明机器繁忙了，可以点击繁忙/健康，看到当前机器的水位。

如果机器繁忙了，默认是无法触发任务的，想要触发任务，有如下解决方案：

1. 根据机器水位进行处理，比如磁盘超过 90%，就清理磁盘。如果 Load 或者 CPU 高，最好排查出是哪里比较耗 CPU，在业务层面解决，如果解决不了最好升级到高配的机器。
2. 如果只是测试，机器繁忙也想继续触发，可以在应用管理 -> 高级配置下，调整响应的阈值，或者关闭【不触发繁忙机器】的开关，如下图

暂无可用的机器（no worker available）

首先排除不是机器繁忙造成的，那么就是接入失败了。登陆自己的机器，找到 SchedulerX 的日志，

1. 日志路径在 `${user.home}/logs/schedulerx/worker.log`
    - 如果进程是通过 root 启动的，目录为 /root/logs/schedulerx/worker.log
    - 如果进程是通过 admin 账户启动的，目录为/home/admin/schedulerx/worker.log
2. 如果日志报 `groupId is not existed`，说明是配置问题，在 worker.log 里搜索 `Schedulerx WorkerConfig`，可以看到当前的配置，然后和控制台确认下，注意 namespace 要填 SchedulerX 控制台 -> 命令空间 -> 命名空间 Id。
3. 如果日志报了其他异常，基本是 Jar 包冲突引起的，可以看下 Jar 包冲突章节。

Jar 包冲突

可以在 worker.log 里搜一下 `maven dependencies`，可以看到每个 Jar 的版本和路径，帮助快速定位和解决 Jar 包冲突

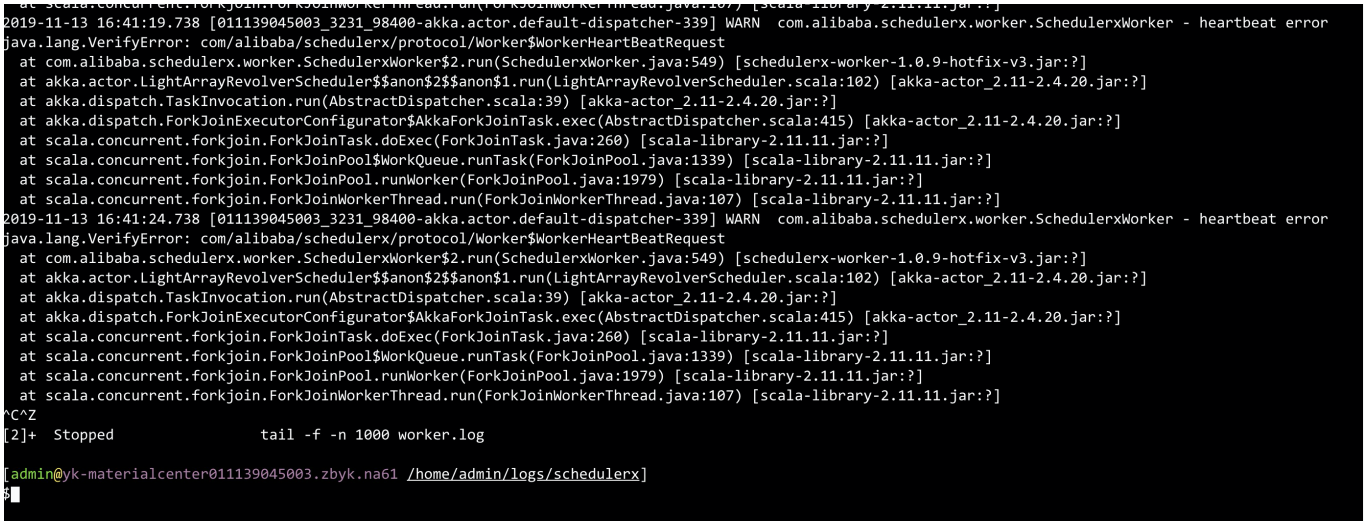
```
2019-06-11 10:29:23.702 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - Schedulerx Worker starting...
2019-06-11 10:29:23.705 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - ===maven dependencies===
2019-06-11 10:29:23.707 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - netty:jar:file:/Users/armon/.m2/repository/io/netty/netty/3.10.6.Final/netty-3.10.6.Final.jar!/org/jboss/netty/channel/socket/nio/
2019-06-11 10:29:23.708 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - protobuf-java:jar:file:/Users/armon/.m2/repository/com/google/protobuf/protobuf-java/2.6.1/protobuf-java-2.6.1.jar!/com/google/protobuf/
2019-06-11 10:29:23.712 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - javaassist:jar:file:/Users/armon/.m2/repository/org/javassist/javassist/3.18.2-GA/javassist-3.18.2-GA.jar!/javassist/compiler/
2019-06-11 10:29:23.712 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - commons-configuration:jar:file:/Users/armon/.m2/repository/commons-configuration/commons-configuration/1.10/commons-configuration-1.10.jar!/org/apache/commons/configuration/
2019-06-11 10:29:23.713 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - config:jar:file:/Users/armon/.m2/repository/com/typesafe/config/1.3.0/config-1.3.0.jar!/com/typesafe/config/
2019-06-11 10:29:23.713 [main] INFO  com.alibaba.schedulerx.worker.SchedulerxWorker - ==================
```

请对照 SchedulerX 使用的 Jar 包，把自己低版本的排掉。

Jar 包	版本
guava	20.0
com.typesafe.config	1.3
protobuf-java	2.6.1
io.netty	3.10.6.Final
javassist	3.21.0-GA
hessian	4.0.51
commons-configuration	1.10
commons-validator	1.4.0
akka	2.4.20
scala	2.11

下面放一些常见的 Jar 包冲突

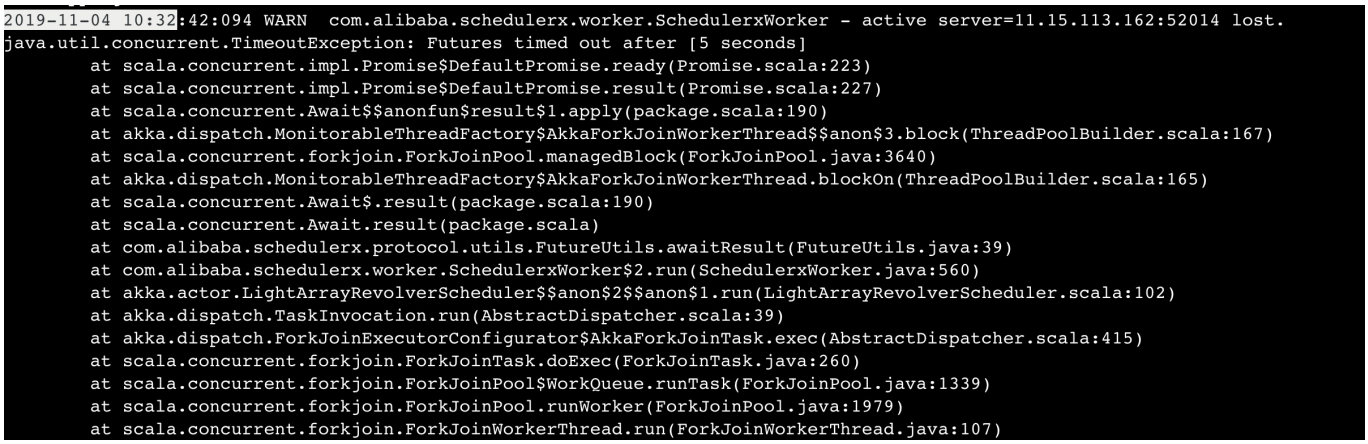
1. protobuf冲突

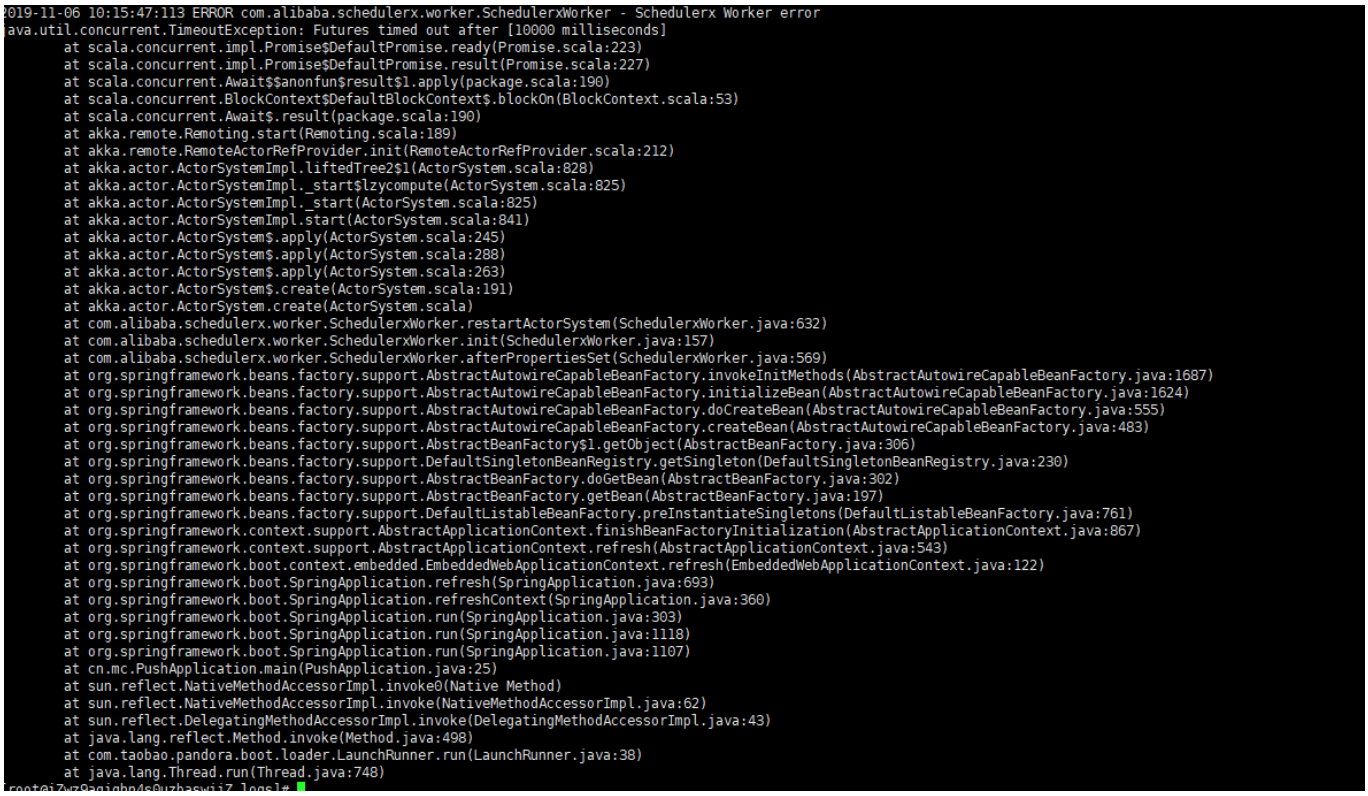


2. netty冲突

   Uncaught error from thread [030011244063_1935_44810-akka.remote.default-remote-dispatcher-6]: org.jboss.netty.channel.socket.nio.NioWorker.<init> (Ljava/util/concurrent/Executor;Lorg/jboss/netty/util/ThreadNameDeterminer;)V, shutting down for ActorSystem[030011244063_1935_44810]
   java.lang.NoSuchMethodError: org.jboss.netty.channel.socket.nio.NioWorker.<init> (Ljava/util/concurrent/Executor;Lorg/jboss/netty/util/ThreadNameDeterminer;)V

3. netty或者protobuf冲突





Spring 应用找不到 Bean

1. 应用管理链接机器看启动方式，确保是 Spring 或者 SpringBoot。
2. JobProcessor 要注入为 bean，比如加 @Component 注解。
3. 排查 pom 依赖如果依赖 spring-boot-devtools 请排除掉。
4. JobProcessor 和 process 方法不要加事务注解。

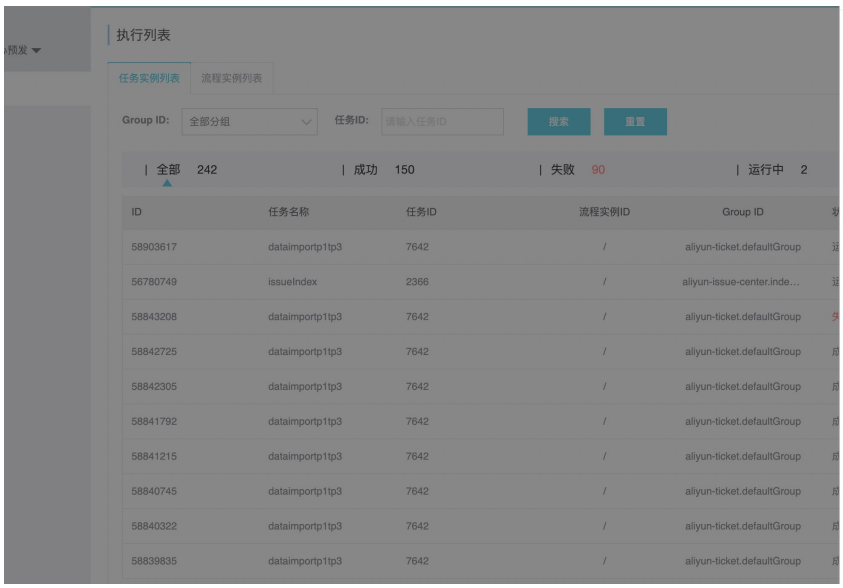
tablestore protobuf-2.4.1 和 SchedulerX 不兼容

https://help.aliyun.com/knowledge_detail/58568.html

任务运行中卡住

schedulerx-worker 执行每个 processor 的时候会把任务实例 Id 放到线程名中，方便查看线程栈。这里以分布式任务某个子任务卡住为例，单机执行/广播执行类似。

1. 首先控制台执行列表查看卡住的任务实例的详情，获取实例id



2. 登入卡住的机器，查看线程栈，执行 `jstack [pid] | grep [实例id] -A 20`，如下图所示，发现是业务自己卡住了。

```
$jstack 29191 |grep 58903617 -A 200
"Schedulerx-Container-Thread-58903617-0" #4093 prio=5 os_prio=0 tid=0x00002ad3edcb4800 nid=0x172f7 waiting on condition [0x00002ad443101000]
   java.lang.Thread.State: WAITING (parking)
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for  <0x000000072b837ae0> (a java.util.concurrent.CountDownLatch$Sync)
  at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(AbstractQueuedSynchronizer.java:836)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireSharedInterruptibly(AbstractQueuedSynchronizer.java:997)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(AbstractQueuedSynchronizer.java:1304)
  at java.util.concurrent.CountDownLatch.await(CountDownLatch.java:231)
  at com.aliyun.ticket.importworker.WOImportJob.startImport(WOImportJob.java:353)
  at com.aliyun.ticket.importworker.WOImportJob.doImportForAll(WOImportJob.java:242)
  at com.aliyun.ticket.importworker.WOImportJob.process(WOImportJob.java:163)
  at com.alibaba.schedulerx.worker.container.ThreadContainer.start(ThreadContainer.java:90)
  at com.alibaba.schedulerx.worker.container.ThreadContainer.run(ThreadContainer.java:60)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
  at java.lang.Thread.run(Thread.java:756)

"Container-Batch-Statues-Retrieve-Thread-58903617" #4092 prio=5 os_prio=0 tid=0x00002ad3edcb4000 nid=0x172f6 waiting on condition [0x00002ad44fe0e000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
  at java.lang.Thread.sleep(Native Method)
  at com.alibaba.schedulerx.worker.batch.BaseReqHandler$2.run(BaseReqHandler.java:74)
  at java.lang.Thread.run(Thread.java:756)

"TDDL-Druid-ConnectionPool-DestroyScheduler--2-thread-237" #4052 daemon prio=5 os_prio=0 tid=0x00002ad3f4bf0800 nid=0x16df5 waiting on condition [0x00002ad462e00
   java.lang.Thread.State: TIMED_WAITING (parking)
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for  <0x00000007436c7190> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
  at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
  at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.poll(ScheduledThreadPoolExecutor.java:1129)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.poll(ScheduledThreadPoolExecutor.java:809)
  at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1066)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1127)
```

推荐产品

- 分布式任务调度 SchedulerX

加入我们

【稳定大于一切】打造国内稳定性领域知识库，**让无法解决的问题少一点点，让世界的确定性多一点点。**

- GitHub 地址
- 钉钉群号：23179349
- 如果阅读本文有所收获，欢迎分享给身边的朋友，期待更多同学的加入！