**NgSql分享模板**

## 项目组

NgSql项目组

## 分享或调研方向

线程池

## 分享人

Sean

## 分享内容

(可包含以下内容：架构、心得或其他内容)

**架构：**

内容

**心得：**

预备知识：

java多线程知识

java线程池的内容

http://coach.iteye.com/blog/855850 这篇文章可以阅读到java线程池的内容。

大致步骤就是 先new 一个ThreadPoolExecutor 线程池对象。 然后调用execute(Runnable task);把任务加入到线程池的工作任务队列当中。线程池就会自动执行任务了。

1. **NameableExecutor**

NameableExecutor extends ThreadPoolExecutor

NameableExcutor 这个是线程池类。调用它的execute(Runnable)可以将任务加入到线程池的任务队列中。

1. **ExecutorUtil**

使用这个类来创建线程池

1. **CobarServer**

CobarServer 现在开始研究这个类， 看看他到底是做神马的

这应该是cobar真正的服务类。 从CobarStartup类中的main函数就可以看出，整个cobar是从CobarServer来启动的，启动函数就是CobarServer类中的startup函数来启动的。 另外CobarServer是一个单例。

我现在主要关注的里面的四个变量

**private final Timer timer;//Timer是一个定时器类,通过该类可以为指定的定时**

**任务进行配置**

**private final NameableExecutor managerExecutor; //处理来自9066端口 任务**

**的线程的线程池 是一些管理命令的任务**

**private final NameableExecutor timerExecutor; //定时任务的线程池**

**private final NameableExecutor initExecutor;**

在看CobarServer的私有构造函数：

**private CobarServer() {**

**this.config = new CobarConfig();**

**//可以通过system变量来读取配置文件的内容**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**this.timer = new Timer(NAME + "Timer", true);**

**this.initExecutor = ExecutorUtil.create("InitExecutor",**

**system.getInitExecutor());**

**this.timerExecutor = ExecutorUtil.create("TimerExecutor", system.getTimerExecutor());**

**this.managerExecutor = ExecutorUtil.create("ManagerExecutor", system.getManagerExecutor());**

**\*\*\*\*\*\*\*\***

**}**

我们可以看出在startup之前，就已经准备好线程池了。

现在在看看CobarServer 的函数startup源码

源码就不贴出来了。 对照着源码看一下的内容即可。

可以从以下几点来看线程池的工作：

1、timer.schedule(processorCheck(), 0L, system.getProcessorCheckPeriod());

这个processorCheck函数里面就调用了timerExecutor.execute(Runnable)函数， 这里已经开始启动工作线程了。 实现的功能是：定时检查任务

2、timer.schedule(dataNodeIdleCheck(),0L,

system.getDataNodeIdleCheckPeriod());timer.schedule(dataNodeHeartbeat(), 0L,system.getDataNodeHeartbeatPeriod());

dataNodeIdleCheck 数据节点定时连接空闲超时检查任务，这些任务交给timerExecutor来处理dataNodeHeartbeat 把数据节点定时心跳任务 交给timerExecutor这个线程池来处理

3、 timer.schedule(clusterHeartbeat(), 0L, system.getClusterHeartbeatPeriod());

从这里开始要开启服务了。clusterHeartbeat() 把集群节点定时心跳任务 交给timerExecutor这个线程池来处理

1. **全局搜索 NameableExecutor**

在所有源码中搜索NameableExecutor

做这一步的目的主要是看看那些地方在使用 线程池。 发现NIOProcessor里面存在。 于是进去看看。 很明显这个类肯定是涉及到来处理NIO的，关于NIO的使用可以参考之前的内容。

CobarServer里有个 NIOProcessor的数组，而初始化大小的个数是通过配置文件配置的。查看配置文件，把这个NIOProcessor解释为：Cobar内部处理器。而NIOProcessor的handler和executor（均为线程池来的，是NameableExecutor的对象）的个数也通过配置来确定。 在CobarServer中初始化每个NIOProcessor后就调用其startup函数（这里又有startup函数了， 先看看源码再说）看代码先：

public void startup() {

reactor.startup();

}

这个reactor 是一个NIOReactor来的，而它的startup如下：

final void startup() {

new Thread(reactorR, name + "-R").start();

new Thread(reactorW, name + "-W").start();

}

而reactorR 和reactorW 是NIOReactor两个实现了Runnable接口内部类的对象。所以这个函数就是启动线程，一读一写。从解释来看这NIOReactor 这是一个网络事件反应器。

NIOProcessor有两个变量，一个是handler，一个是executor。可以预想其他类中肯定会调用这两个线程池来执行任务的。 通过查看FrontendConnection的 handler（final byte[]） 中就有这么一句代码：

processor.getHandler().execute(new Runnable())

这里正是在把任务加到工作队列中。

## 当前进度(选填)

1. 通过学习java线程池的内容，以及研究Cobar的几个关于线程池的类。现在大致把握了cobar的线程池是如何工作的

## 下阶段计划(选填)

1. 继续研究源码以及有关设计模式，从具体的功能上去研究线程池是如何工作的