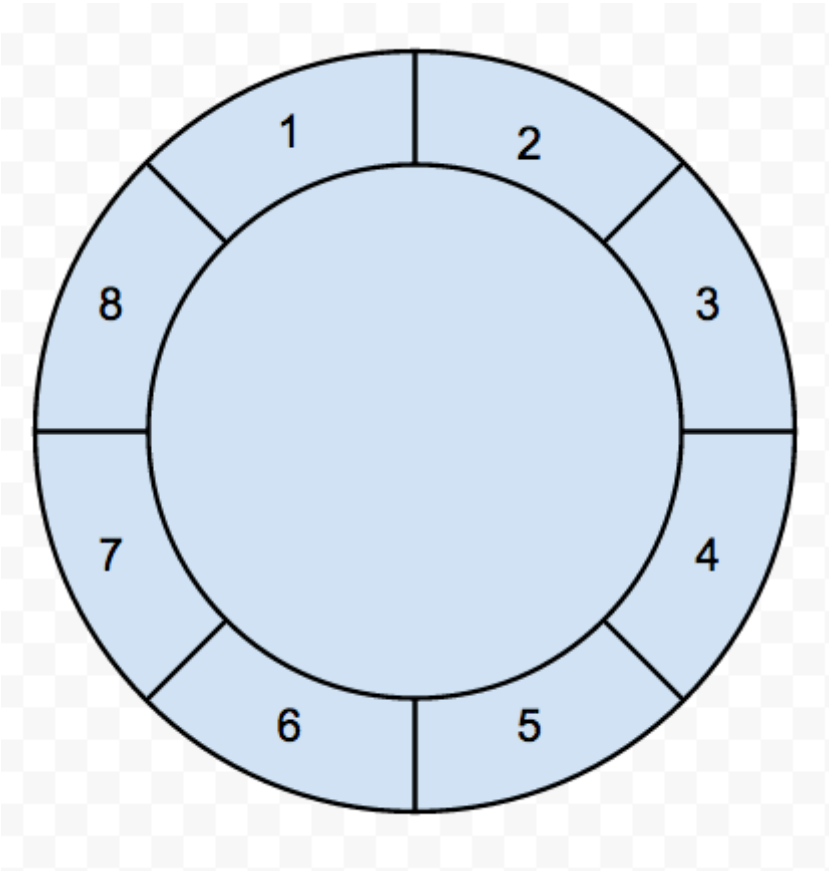
HashedWheelTimer 原理

HashedWheelTimer 是根据

Hashed and Hierarchical Timing Wheels: Data Structures
for the Efficient Implementation of a Timer Facility

这篇论文做出来的.

HashedWheelTimer 主要用来高效处理大量定时任务, 他的原理如图



可以将 HashedWheelTimer 理解为一个 Set<Task>[] 数组, 图中每个槽位(slot)表示一个 Set<Task>

HashedWheelTimer 有两个重要参数

tickDuration: 每 tick 一次的时间间隔, 每 tick 一次就会到达下一个槽位

Design Pattern(10)
Git/Github(2)
Guava(23)
Guava User Guide(7)
JAVA(144)
Linux(18)
Maven(2)
MyBatis(7)
MySQL(31)
Netty(12)
Nginx(1)
PhotoShop(3)
PHP(82)
PHP面向对象,模式与实践(6)
Redis(1)
Spring(17)
Test(3)
Thinking in Java(10)
ZooKeeper(2)
代码(5)
计算机网络(12)
计算机知识(3)
鸟哥的Linux私房菜(1)
算法与数据结构(70)

随笔档案(339)
2015年4月 (2)
2015年3月 (1)
2015年1月 (2)
2014年12月 (2)
2014年11月 (2)
2014年10月 (3)
2014年9月 (3)
2014年8月 (6)
2014年7月 (3)
2014年6月 (4)
2014年5月 (3)
2014年4月 (8)
2014年3月 (2)
2014年2月 (7)
2014年1月 (3)
2013年12月 (1)
2013年11月 (2)
2013年10月 (16)
2013年9月 (27)
2013年8月 (23)
2013年7月 (31)
2013年3月 (3)
2013年2月 (5)
2013年1月 (4)
2012年12月 (2)
2012年11月 (13)
2012年10月 (40)
2012年9月 (32)
2012年8月 (32)
2012年7月 (4)
2012年6月 (10)
2012年5月 (40)
2012年4月 (3)

最新评论
1. Re:SimpleDateFormat的线程安全问题与解决方案
这个断点--有点不好找，具体为止JDK1.7版本 java.text.CalendarBuilder 的 establish（）方法中，有 clear(),

ticksPerWheel: 轮中的 slot 数

上图就是一个 ticksPerWheel = 8 的时间轮, 假如说 tickDuration = 100 ms, 则 800ms 可以走完一圈

在 timer.start() 以后, 便开始 tick, 每 tick 一次, timer 会将记录总的 tick 次数 ticks

我们加入一个新的超时任务时, 会根据超时的任务的超时时间与时间轮开始时间算出来它应该在的槽位.

例如 timer.newTask(new Task(10, TimeUnit.SECONDS));

表示加入一个 10s 后超时的任务, 那么, 先计算他应该在的槽位

```
// deadline = 当前时间 + 任务延迟 - timer启动时间 = timer启动到任务结束的时间
long deadline = System.currentTime() + timeout - timerStartTime;

// calculated = tick 次数
long calculated = deadline / tickDuration;
// tick 目前已经 tick 过的次数
final long ticks = Math.max(calculated, tick); // Ensure we don't schedule for past.
// 算出任务应该插入的 wheel 的 slot, slotIndex = tick 次数 & mask, mask = wheel.length - 1, 默认即为 511
stopIndex = (int) (ticks & mask);
// 计算剩余的轮数, 只有 timer 走够轮数, 并且到达了 task 所在的 slot, task 才会过期
remainingRounds = (calculated - tick) / wheel.length;
```

其中 stopIndex 为它所在的槽位

remainingRounds 为它从 timer 启动时应该经过的轮数

当 timer tick 到 task 所在的槽位, 并且这个槽位的 remainingRounds <= 0 , 则说明这个 task 超时, 然后执行超时任务, 否则 remainingRounds--

---

为什么要使用时间轮的环形结构? 因为环形结构可以根据超时时间的 hash 值(这个 hash 值实际上就是ticks & mask)将 task 分布到不同的槽位中, 当 tick 到那个槽位时, 只需要遍历那个槽位的 task 即可知道哪些任务会超时(而使用线性结构, 你每次

可以在establish（）返回值为止设置断点。...
--junyi5257
2. Re:JOIN与EXISTS(子查询)的效率研究
exists的时候要select 一个常数 不要查询* 数据里大的时候性能差的还是挺多的
--CrazyRisk
3. Re:MyBatis使用Collection查询多对多或一对多结果集bug
学习了～
--规格严格-功夫到家
4. Re:[转]Nginx.conf介绍
关键是 在实践中调节这些配置参数了，才能印象深刻~
--程序员的文娱情怀
5. Re:SimpleDateFormat的线程安全问题与解决方案
看一下能不能改变昵称
--扭断翅膀的猪

阅读排行榜
1. Apache配置正向代理与反向代理(81125)
2. MyBatis使用Collection查询多对多或一对多结果集bug(56097)
3. MyBatis 缓存(53289)
4. MySQL分区表(40543)
5. Java Web 项目获取运行时路径classpath(30884)

评论排行榜
1. SimpleDateFormat的线程安全问题与解决方案(8)
2. MyBatis使用Collection查询多对多或一对多结果集bug(6)
3. 利用MyBatis的动态SQL特性抽象统一SQL查询接口(4)
4. PHP中的 clone()(4)
5. MyBatis 缓存(4)

推荐排行榜
1. Apache配置正向代理与反向代理(7)
2. MyBatis 缓存(6)
3. SimpleDateFormat的线程安全问题与解决方案(4)
4. MyBatis使用Collection查询多对多或一对多结果集bug(4)
5. HttpAsyncClient的连接池使用(3)

tick 都需要遍历所有 task), 所以, 我们任务量大的时候, 相应的增加 wheel 的 ticksPerWheel 值, 可以减少 tick 时遍历任务的个数.

详细代码参考 Netty 的实现:
https://github.com/netty/netty/blob/master/common/src/main/java/io/netty/util/HashedWheelTimer.java

分类: JAVA,Netty

ZimZz
关注 - 1
粉丝 - 149
+加关注

« 上一篇: Photoshop 使用曲线
» 下一篇: Two-Phase Commit (两阶段提交)

posted on 2014-08-21 22:59 ZimZz 阅读(4900) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

发表评论

昵称: Kingson_Wu

评论内容:

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

最新IT新闻:

· 酷我或遭淘汰，腾讯音乐资本运作的牺牲品?
· Alphabet董事长：不要雇佣太多“胶水人” 招聘人才看这两点
· EA超级福利：Origin游戏全体限免7天，正版大作扎堆
· 微信新功能：公众号文章语音支持进度条拖拽
· B站开启游戏实名认证：主站账号暂时不受影响
» 更多新闻...

最新知识库文章:

· 小printf的故事：真正的程序员?
· 程序员的工作、学习与绩效
· 软件开发为什么很难
· 唱吧DevOps的落地，微服务CI/CD的范本技术解读
· 程序员，如何从平庸走向理想?
» 更多知识库文章...

历史上的今天:

2012-08-21 PHP 合并排序