

# 云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

[« 增强了 skynet 的 socket 库 | 返回首页 | 给 skynet 添加 mongo driver »](#)

## coroutine 的回收利用

这几天在 lua 和 luajit 的邮件列表上有人讨论 coroutine 的再利用问题。

前几天有个用 skynet 的同学给我写了封邮件，说他的 skynet 服务在产生了 6 万次 timeout 后，内存上升到了 50M 直到 gc 才下降。

这些让我重新考虑 skynet 的消息处理模块。skynet 对每条消息的相应都产生了一个新的 coroutine，这样才能在消息处理流程中，可以方便的切换出去让调度器调度。诸如 RPC/ socket 读写这些 api 才能在使用起来看成是同步调用，却在实现上不阻塞线程。

读源码可知，lua 的 coroutine 非常轻量（luajit 的略重）。但依旧有一些代价。频繁的动态生成 coroutine 对象也会对 gc 造成一定的负担。所以我今天花了一点时间优化了这个问题。

简单说，就是用自己写的 co\_create 函数替换掉 coroutine.create 来构建 coroutine。在原来的主函数上包裹一层。主函数运行完后，抛出一个 EXIT 消息表示主函数运行完毕。并把自己放到池中。如果池中有可利用的旧 coroutine，则可以传入新的主函数重新利用之。

为了简化设计，如果 coroutine 中抛出异常，就废弃掉这个 coroutine 不再重复利用。为了防止 coroutine 池引用了死对象，需要在主函数运行完后，把主函数引用清空，等待替换。

具体实现参见[这个 patch](#)。

ps. coroutine poll 故意没实现成弱表，而是在相应 debug GC 消息时再主动清空。

云风 提交于 July 25, 2013 03:00 PM | [固定链接](#)

## COMMENTS

@lpk

加上超时后会增加业务层特别多的复杂度,得不偿失.

如果必须要, 可以不用 call 而用 send 自己模拟. skynet 已经提供了足够的机制把超时做出来.

Posted by: Cloud | (2) [August 9, 2013 07:31 PM](#)

能否给coroutine加上超时限制呢,比如skynet.call,跨机器调用别一个服务,而被调用的服务没有响应的話,如果调用次数足够多,这会引起大量coroutine挂起.

Posted by: lpk | (1) [August 9, 2013 04:01 PM](#)

## POST A COMMENT

非这个主题相关的留言请到: [留言本](#)

名字:

Email 地址:

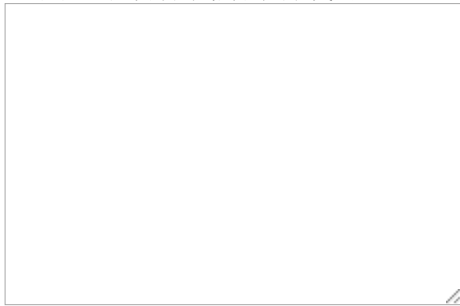
为了验证您是人类, 请将六加一的结果(阿拉伯数字七)填写在下面:

URL:

☐ 记住我的信息?

留言:

(不欢迎在留言中粘贴程序代码)



提交