# 云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« 招聘 平台开发工程师 | 返回首页 | 简悦招聘 Unity3D 程序员 »

## skynet 中 Lua 服务的消息处理

最近为 skynet 修复了一个 bug ， Issue #51 。经查，是由于 redis driver 中的 batch 模式加锁不当造成的。

有同学建议把 batch 模式取消，由于历史原因暂时还保留。在很多其它 redis driver 的实现中也不提供类似机制。也就是依次提交多个数据库操作请求，不用等回应，最后再集中处理数据库返回的信息。

我的个人建议是在目前的 redis driver 基础上再实现一个独立服务，里面做一个连接池，让系统不同服务对数据库的读写工作在不同连接上，这样可能更好些。如果简单的实现一个数据库代理服务而不采用连接池的话，可能会面对一些意想不到的情况。

这是由 skynet 的 lua 模块工作方式决定的，下面解释一下。

---

skynet 中的不同服务是利用系统的多线程完全并行的。当你从服务 A 向 B 和 C 分别各发送一条消息时，并不能保证先发的消息先被处理。

而当你从服务 A 向 B 依次发送两条消息，那么先发的消息一定会被 B 先处理。

用 lua 实现的服务只是一个内嵌了 Lua 虚拟机的服务，也遵守上面的规则。但目前的实现中，由于使用了 lua 的 coroutine 机制，问题变得更复杂一些了。

如果 B 是一个 lua 服务，当 A 向 B 发送了两条消息 x 和 y 。skynet 一定保证 x 先被 B 中的 lua 虚拟机收到，并为 x 消息生成了一个 coroutine X ，并运行这个 coroutine 。然后才会收到消息 y ，重新生成一个新的 coroutine Y ，接下来运行。

大多数情况下系统是会保证运行次序的。可一旦 coroutine X 中调用了skynet 提供的 socket IO 处理，或是调用了 skynet.call skynet.sleep （注：skynet.send 不会导致挂起）等会导致 coroutine 挂起的指令，那么消息处理的执行流就被暂时挂起了。

和 erlang 的 process 不同，此时 skynet 挂起的是 lua vm 中的 coroutine ，服务 B 本身是可以继续处理消息的。这个时候，一旦 y 消息抵达，一个新的 coroutine Y 就会被创建并运行。看起来，B 中就有两条执行流程并行了。

从这个意义上来说 skynet 中 lua 虚拟机里的 coroutine 才能看成是 erlang 中 process 的（不完全）等价物。lua 虚拟机，也就是 skynet 中的一个服务，提供了一个共享环境，让不同的 coroutine 之间可以共享状态。这也是很多 bug 的滋生之处。

---

如果这个行为让你困扰的话，那么我建议另外实现一个 lua 库，去掉 coroutine 的部分，转而实现一个类似 erlang 的 mailbox 机制来接收 skynet 的 C 层转发过来的消息会更好一些。

或者，我今天给 skynet 增加了一个叫 mqueue.lua 的库，可以给每个服务定义一个消息队列，消息队列里的消息会被依次处理。

见 testqueue.lua 以及 pingqueue.lua 可以看到基本用法。

云风 提交于 October 30, 2013 04:58 PM | 固定链接

### COMMENTS

云风哥， 我在看skynet lua-seri.c代码， 是从你的lua-serialize修改而来， 我注意到在unpack的时候， 你注释说: // Need not free buffer

为什么呢， lua-serialize是free掉buffer的。 skynet的free放在了什么地方呢？

Posted by: van9ogh | (12) October 17, 2014 02:54 PM

云风大哥

我们现在的项目也遇到类似的问题了，或者更严重点，我们的char在登录的过程中分别会向pcl和db分别发起一条数据query gs和pcl，db之间均为异步 我们的语句大致是这样 sendonequerytppcl sendonequerydb ;pclreq的返回数据可能会insertdb or updb,我们更希望得到up之后的数据 ，但是问题是我们又不能将querydb挂在pclreq响应后才去激活dbreq （我们无法确保这个时间有多长） 这样直接导致了dbreq返回的数据有可能不及时，甚至返回的是错误的数据 在这样的情况下我们当如何处理？

Posted by: Injur | (11) November 9, 2013 08:55 PM

云风大哥
我们现在的项目也遇到类似的问题了，或者更严重点，我们的char在登录的过程中分别会向pcl和db分别发起一条数据query gs和pcl，db之间均为异步 我们的语句大致是这样 sendonequerytppcl sendonequerydb ;pclreq的返回数据可能会insertdb or updb,我们更希望得到up之后的数据 ，但是问题是我们又不能将querydb挂在pclreq响应后才去激活dbreq （我们无法确保这个时间有多长） 这样直接导致了dbreq返回的数据有可能不及时，甚至返回的是错误的数据 在这样的情况下我们当如何处理？

Posted by: Injur | (10) November 9, 2013 08:54 PM

其实我挺佩服高代码的，人家销售业绩300W不是人人都有的。http://www.haobitou.com/new/300W.html#00 看看去。。

Posted by: 如何管理好团队？ | (9) November 4, 2013 11:47 AM

终于又有更新了，两个星期没来看看了支持！！

Posted by: 好笔头 | (8) November 4, 2013 11:46 AM

谈下登录服务，可以在一个service里面通过多个coroutine以连接池的方式读取各自数据（无序的）,但我觉得还不如开多个service绑定db连接，毕竟，service最多只有一个线程资源可以被调度，多个service才可以真正的并行。
聊天服务的话，本身好像来没有call等IO操作，几乎是池内从来就只有一个coroutine，无所谓有序无序了。
总体来说，我更倾向于多开service，coroutine尽量保持有序，虽然有些场景无序会高效些，但不应该过早优化这个问题，让开发人员陷入困惑。

Posted by: wesom | (7) November 1, 2013 03:36 PM

@wesom 比如agent发送modify部分的数据给dbservice两次，分别为a、b，均为当前某个属性剩余值，那么必然b包要比a包后处理，那这里必然就有时序的要求，即这两个coroutine得排队处理
——这种情况是保证时序的，先处理A，再处理B。mqueue解决的是A消息处理过程中被挂起[yield]的情况【见云风上面的分析】
应该只有与agent类似的并行的服务才能用这套机制吧，其它的，例如Map服务、登录服务、聊天服务、好友服务、组队服务都应该慎用吧，其实弱弱地觉得dbproxy也应该慎用的。

Posted by: mike | (6) November 1, 2013 11:38 AM

我觉得skynet可以的啊，这里面向讨论全局的tab ,对于2次msg操作同一个tab的某一个字段（当然如果非同字段也就不存在问题了吧？除非设计tab内的key->val运算），我们是否可以序列化msg,就算我们先处理了coroutine Y再处理coroutine X，这时发现X的序列是低于当前序列的，是否可以不针对某些冲突event做响应呢？

Posted by: injur | (5) October 31, 2013 09:50 PM

我说的是大部分场景，不是绝对。像某些应用场景，如排名系统，可以接受无序性，利用coroutine提升性能

Posted by: wesom | (4) October 31, 2013 03:21 PM

to @mike, 我说的场景和表设计无关，比如agent发送modify部分的数据给dbservice两次，分别为a、b，均为当前某个属性剩余值，那么必然b包要比a包后处理，那这里必然就有时序的要求，即这两个coroutine得排队处理

Posted by: wesom | (3) October 31, 2013 03:11 PM

@wesom 如果消息的处理是同步处理的，的确应该保证有序处理。但如果消息中途会异步出去，保证有序性会牺牲掉服务器的性能，影响服务器的吞吐量。为了解决这个问题，只能通过多开Service的方法来解决。据我所知，微信的当

前框架就是这样的模型，但在某些关键Service上面，机器的要求很大，出现瓶颈。现在已经部分服务coroutine化。
再回到DB的话题上，我觉得可能你有点误解了应用场景。这个BUG的发生是出现在一个玩家的数据被存在多个表里面的，存一个玩家数据有多个表要一下子写下去，而不能一个写完再写另外一个。对于这种DB设计思路，我持反对的意义，如果是相关性的几个表，要同时读入同时写进去，放在一个表更加安全一些。否则其中一个表写失败了，肿么办啊肿么办？

Posted by: mike | (2) October 31, 2013 02:41 PM

大部分dbproxy的职责决定了它的处理必须是有序的。比如玩家某个属性改变了两次，这两coroutine的完成必须是有序的，或者自己做队列如最新的socket.lock()的方式，或者使用新的mqueue。在多个连接上，也最好是不同的服务对应各自的单个连接。如果同一类数据读写实现连接池，会增加复杂度，而且也没什么必要。
btw，感觉大部分场景下service都是需要有序的处理，coroutine带来的好处不多。细想下erlang的设计，不无道理。

Posted by: wesom | (1) October 31, 2013 11:31 AM

## 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