云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« linode 广告时间 | 返回首页 | 在不同的 lua vm 间共享 Proto »

Skynet 新的 socket.channel 模式

大部分外部网络服务都是请求回应模式的，skynet 和外部数据库对接的时候，直接用 socket api 编写 driver 往往很繁琐。需要解决读取异步回应的问题，还需要正确处理连接断开后重连的问题。

这个周末，我试着给 skynet 的 socket 模块加了一个叫做 channel 的模式，用来简化这类问题的处理。

可以用 socket.channel { host = hostname, port = port_number } 创建出一个对象。这个对象用来和外部服务器通讯。

对于 redis 的协议模式，我们每次发送一个请求，就对应着可以读到一个回应包。请求队列和回应队列是次序一致的。

那么，就可以使用 channel:request(request, response) 这个 api 获取回应信息。

这里 request 是一个字符串，即需要发送给服务器的请求内容。而 response 是一个函数，要求它可以返回两个值：第一个是一个 boolean ， true 表示回应内容正确，这时第 2 个返回值就是返回的对象。

如果 response 函数返回 false，第 2 个返回值是错误对象。这会导致 channel:request 将它以异常形式抛出。

而 mongo 的协议模式，并不保证请求和回应之间的次序，它是用一个 session id 来表识的。有可能出现先发出的请求，后收到回应。这时，我们应该在 channel 初始化的时候提供一个 response 包解析函数。

socket.channel { host = hostname, port = port_number , response = dispatcher }

这里的 dispatcher 就是这样的一个函数，它需要返回 3 个值：

第一个是 session ，表示这个包对应的是哪一次请求。

第二三个返回值的含义和前面所描述的 response 函数相同：一个 boolean 加一个返回对象。

在发起请求的时候，使用 channel:request(request, session) 即可。这里的第 2 个参数是这次请求的 session 。返回值的含义是一致的。

如果请求不需要接收返回包（mongo 的通讯协议中，有许多这样的协议），可以调用 channel:request(request) 即可。如果之后需要收取回应包，可以再调用 channel:response(response) 。这种用法现在被用来实现 redis 的 watch 模式 (pubsub 模式) 。

channel 模块会捕获 response 函数处理过程中的异常，包括网络异常断开的请求。任何在处理回应数据的过程中产生的异常都会强制连接断开，尝试重新连接服务器。重连完成后，未收到回应的请求将被重新发送。

如果每次连接服务器需要做一些认证等流程，可以在 channel 初始化的时候给出一个 auth 方法，底层会在每次成功连接上后调用。

有了 channel 模式后，socket 的 lock 机制一般就不需要在使用了。因为无论是请求和回应的处理都是 coroutine 安全的。请求被要求在一次 socket 写操作中完成；而回应的处理被统一安排在一个 coroutine 内分发。

多个 coroutine 同时提起 request 不会相互阻塞，所以 redis driver 原来提供的 batch 模式就意义不大了，这次重写 redis driver 我将其删去（如果有需要做流水线命令，可以考虑以新的方式实现）。

由于这次改动较大，所以我暂时提交到了一个临时的新分支 channel 上。希望有在用 skynet mongo driver 的同学帮忙测试下新版本的 driver 有没有明显的问题。

云风 提交于 March 23, 2014 04:13 PM | 固定链接

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