云风的 BLOG

思绪来得快去得也快，偶尔会在这里停留

« 招聘 Windows/Linux SA 一名 | 返回首页 | MongoDB 的 Lua Driver »

skynet 的网关模块的一点修改

skynet 有一个叫做 gate 的模块，用来解决外部连接数据读取的问题。它最初是用我随手为 ringbuffer 示例 而写的一段代码改造成的。

最初我认为，用 epoll 去处理读事件足够了。至于写数据，完全可以用阻塞写的方式进行。因为 skynet 可以将事务分到多线程中，所以特定几个 socket 发生阻塞，也并不会把系统阻塞住。也可以简单的理解为，单线程读，多线程写。

随着我们的游戏的开发，这样做的弊端逐渐显露出来。大量玩家聚集的场景里，广播数据会突然同时塞满多条连接。skynet 的工作线程数据固定，这样就有可能因为同时写数据而阻塞住所有的工作线程。

这个问题发现有一段时间了。当时蜗牛同学顺手改了一下，把外部连接设为非阻塞状态，并调大了缓冲区。一旦发生缓冲区慢的情况，就关闭连接。

这个改法并不彻底，而且在关闭连接后未能发送消息通知 agent ，导致业务逻辑不正确。上周仔细考虑了一下，决定把当初偷懒而没有完成的代码重新实现一下。

一开始，我在发送数据的模块中增加了写缓冲区。一旦系统缓冲区满，就放进应用层的缓冲区。等下次发起写请求时，再将以前没有发生完的数据发完。对于网络游戏来说，这样做基本够了。但隐患是，有可能最后一小片数据永远没有发送出去。（在 MMORPG 中，这种可能性几乎为零，即使发生，对用户也没有太坏的影响）

为了堵住这点纰漏，我设置了超时继续的 timer 。一旦应用层缓冲区有数据，timer 会在一定时间后处理它。

但这个做法引起了晓靖同学的不满。他觉得不统一用 epoll 来解决大量写事件而用 timer 这种事情是不可接受的。OK ，我也认同这点，但我这不是不想去动已经跑的好好的代码么？

周末还是忍不住动手了。

一开始，我想另外写一个模块整合所有的写事件。不过，读写模块分开有个问题：当用户想关闭 socket 时，它无法知道应用层写缓冲区里是否还有没发完的数据。对于 MMORPG 应用来说，有数据没发完就断开连接也不是啥大问题，不过对于 skynet 这个层次的东西来说，这样做粗暴了点。

我也想过另一个方案：重新做一个模块，利用 epoll 监听读写事件，并把事件发送给需要的服务，而自己并不真正读写数据。这样可以完美的把系统的 socket 读写 api 转换为 skynet 友好的接口。这样做的坏处是，需要更大的内存缓冲区（因为每个连接独立了）；对现有代码改动较大；多了一个间接层，可能对性能有些影响。

我认为这可能是最佳方案，如果这样做，还可以把原来 skynet 中别的 IO 操作的部分也整合在一起。但最后我还是放弃了。

昨天，我决定把所有的 socket 写操作都统一交给 gate 模块处理，并保留原有分开处理的方式做兼容（等新代码跑一段时间，再去掉旧代码)。以前的发送模块继续保留，为一个连接设立一个服务负责发送数据到客户端。但它们不再直接写 socket ，而是将数据打包转发给 gate 。

这样修改后数据的处理流会多一次复制的环节，但考虑到日后还需要对数据加密和压缩，这个环节倒不算冗余。

代码写好后，已经提到到 github 。kqueue 的部分暂时还没有环境测试，希望有测试环境的同学可以帮我看看 ：）

云风 提交于 May 26, 2013 05:32 PM | 固定链接

COMMENTS

@cloud，十分感谢！
有问题再请教，在用lua过程中也有很多细节。

Posted by: cd | (17) June 19, 2013 06:11 PM

@cd

sorry, 我昨天可能搞错了.
今天重新 review 了代码.

如果用 dont copy 后， 接收方除了要自己 free 还需要让 callback 函数返回 1.

否则 callback 函数调用结束后， 会重复 free.

换句话说， 如果 callback 返回 0, 就会默认调用 free 清理消息. 如果你的消息不是 malloc 的， 那么， 就可以自己销毁， 但要 callback 返回 1 阻止 skynet 用 free 清理.

Posted by: Cloud | (16) June 19, 2013 03:32 PM

@cd,

1. 简单的 C service 可以在 stack 上构造 message.
如果你自己 malloc, 让 skynet 复制, 那么在 skynet_send 后立刻 free 就可以了.

2. 有内存泄露, 前面我提到过, dont copy 是一个性能优化选项. 使用它必须收发方协作. 发送方负责 malloc, 接收方就必须 free.

3. 假设接收和发送方都在同一进程内(可以通过 service id 高位识别, 但如果建立过 tunnel 就可以有意外) , 那么你甚至可以传送复杂结构或指针, 接收方可以释放它们就可以了.

如果跨进程, skynet 的默认 harbor 无法得知如何复制 message , 所以一定会出问题. 这种消息需要自己定制 harbor 才能转发.

简单说, 设置了 dont copy 后, 如果跨机器, message 还是会被 memcpy 的, 但由于 harbor 也是用 malloc , 所以可以正确工作.

Posted by: Cloud | (15) June 18, 2013 03:54 PM

多谢cloud，，，第二个问题，应该是我自己内存越界，我再细查细查。

第一个问题，小细节请教：
Q1：A做了malloc，生成msg-a，msg传递给B，如果不设置donotcopy，则skynet自己又malloc了一个msg-a的副本msg-b压入B的队列，然后对B进行cb调用后释放的是msg-b。
没有发现msg-a在哪里释放，这里有内存泄露么？能否提示下在哪里释放的？
Q2：A做了malloc，生成msg-a，msg传递给B，如果设置donotcopy并且B的cb返回为1，则skynet直接将msg-a压入B的队列，然后对B进行cb调用后又不释放msg-a。
没有发现msg-a在哪里释放，这里有内存泄露么？能否提示下在哪里释放的？

Q3：A是否可以传递一个非malloc的固定内存给B?

十分感谢!

Posted by: cd | (14) June 18, 2013 03:48 PM

@cd

1. 由接收方 free , 这必须收发双方有共同的约定,且只能在本机传输. 这是一个纯优化性能的选择, 其代价就是有很多限制. 如果性能不是问题, 那么不建议使用.

2. 最近 gate 才加入发送模块, 可能测试还不充分. 但是我这里还没有发现问题.如果你可以 fix 它, 可以给我一个 pull request .

问题出在 mread_push 的 free 中的话, 那么它 free 的是 service-src/service_client.c 的 _cb 函数中 malloc 出的 tmp 指针. 希望这个线索可以帮助你查到 bug .

Posted by: Cloud | (13) June 18, 2013 03:18 PM

在编译和使用你的框架，有如下疑问：
1、如果我自己service对传递的消息做了malloc，type有传递为not copy的话，这块内存什么时候释放？
2、你现有的gate模块，经常用了一段时间就出现这个提示：

执行的时候，出现如下提示：

*** glibc detected *** ./skynet: free(): invalid next size (fast): 0x0000000013d280e0 ***
======= Backtrace: =========
/lib64/libc.so.6[0x3b4ce722ef]
/lib64/libc.so.6(cfree+0x4b)[0x3b4ce7273b]
./service/gate.so(mread_push+0xbd)[0x2afd522ce967]
./service/gate.so[0x2afd522d1417]
./skynet[0x4040cb]
./skynet(skynet_context_message_dispatch+0xfa)[0x4041ed]
./skynet[0x40587d]
/lib64/libpthread.so.0[0x3b4da06617]
/lib64/libc.so.6(clone+0x6d)[0x3b4ced3c2d]

Posted by: cd | (12) June 18, 2013 02:01 PM

做了这么久的服务器？怎么第一印象就是用阻塞的I/O？第二印象是调整发送缓冲区？第三印象是搞timer？ 你的skynet都包含啥？难道做网络层框架时没实现EPOLLOUT ？

Posted by: cui | (11) June 13, 2013 11:29 PM

@Dave

谢谢

我一直没有搭建 Mac OSX 下的环境, 所以这个版本没有测试过.

刚才我在我的 MacMini 上搭建好了开发环境, 找到 bug 修复了.

Posted by: Cloud | (10) June 12, 2013 01:32 AM

你好git 到你最新的skynet运行是报错（mac环境）
GDB调试信息如下 在mread.c:450 貌似出现EXC_BAD_ACCESS。

(gdb) info break
Num Type Disp Enb Address What
1 breakpoint keep y 0x000b99dc in _read_one at mread.c:450
breakpoint already hit 1 time
(gdb) p ret
$1 = (struct socket *) 0xffffffff
(gdb) p &ret
$2 = (struct socket **) 0xb0184d98
(gdb) p ret
$3 = (struct socket *) 0xffffffff
(gdb) p ret->fb
There is no member named fb.
(gdb) p ret->fd
Cannot access memory at address 0xffffffff
(gdb) list
445 writeflag = flag & EVFILT_WRITE;
446 readflag = flag & EVFILT_READ;
447 #endif
448 ++ self->queue_head;
449 if (writeflag) {
450 client_send(&ret->client, ret->fd);
451 try_close(self, ret);
452 }
453 if (readflag)
454 return ret;
(gdb) r
The program being debugged has been started already.

Start it from the beginning? (y or n) n
Program not restarted.
(gdb) n

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0xffffffff
0x000b99df in _read_one (self=0x9dea00) at mread.c:450
450 client_send(&ret->client, ret->fd);

Posted by: Dave | (9) June 11, 2013 11:29 PM

“需要更大的内存缓冲区（因为每个连接独立了）”...,这和调大socket发送缓冲区有啥区别:)

Posted by: donkey | (8) June 7, 2013 03:29 PM

你看错了, EAGAIN 就返回了.

Posted by: Cloud | (7) June 5, 2013 05:08 PM

当write遇到 EAGAIN时,为何不把它加到EPOLL? 而是用while继续写? 加到 epoll不是效率更高么.

Posted by: why | (6) June 5, 2013 09:58 AM

推荐timerfd和eventfd

Posted by: tinyzhang | (5) May 31, 2013 05:45 PM

timer里面写pipe，配合epoll，治疗洁癖の黑暗方式，呵呵

Posted by: Anana | (4) May 31, 2013 03:31 PM

推荐使用timerfd和eventfd，可以和epoll配合实现异步计时器和消息队列。缺点是bsd系统不支持timerfd和eventfd。

Posted by: zcpro | (3) May 27, 2013 12:59 PM

重写代码有时候也是一个好选择

Posted by: c语言 | (2) May 27, 2013 09:43 AM

不要害怕重写，有时候重写是一个更好的方案

Posted by: c语言小程序 | (1) May 27, 2013 09:40 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