Go 网络库并发吞吐量测试

作者：Allenxuxu
原文链接：https://ld246.com/article/1569146985090
来源网站：链滴

https://github.com/Allenxuxu/gev

本文主要测试 gev 网络库和其他三方 Go 网络库以及标准库的吞吐量对比。

测试对象

- gev ： 一个轻量、快速的基于 Reactor 模式的非阻塞 TCP 网络库
- eviop： evio 的优化版本
- evio： Fast event-loop networking for Go
- gnet： eviop 的网络模型替换版本
- net 标准库

测试方法

采用陈硕测试 muduo 使用的 ping pong 协议来测试吞吐量。

简单地说，ping pong 协议是客户端和服务器都实现 echo 协议。当 TCP 连接建立时，客户端向服器发送一些数据，服务器会 echo 回这些数据，然后客户端再 echo 回服务器。这些数据就会像乒乓一样在客户端和服务器之间来回传送，直到有一方断开连接为止。这是用来测试吞吐量的常用办法。

测试的客户端代码： https://github.com/Allenxuxu/gev/blob/master/benchmarks/client/main.o

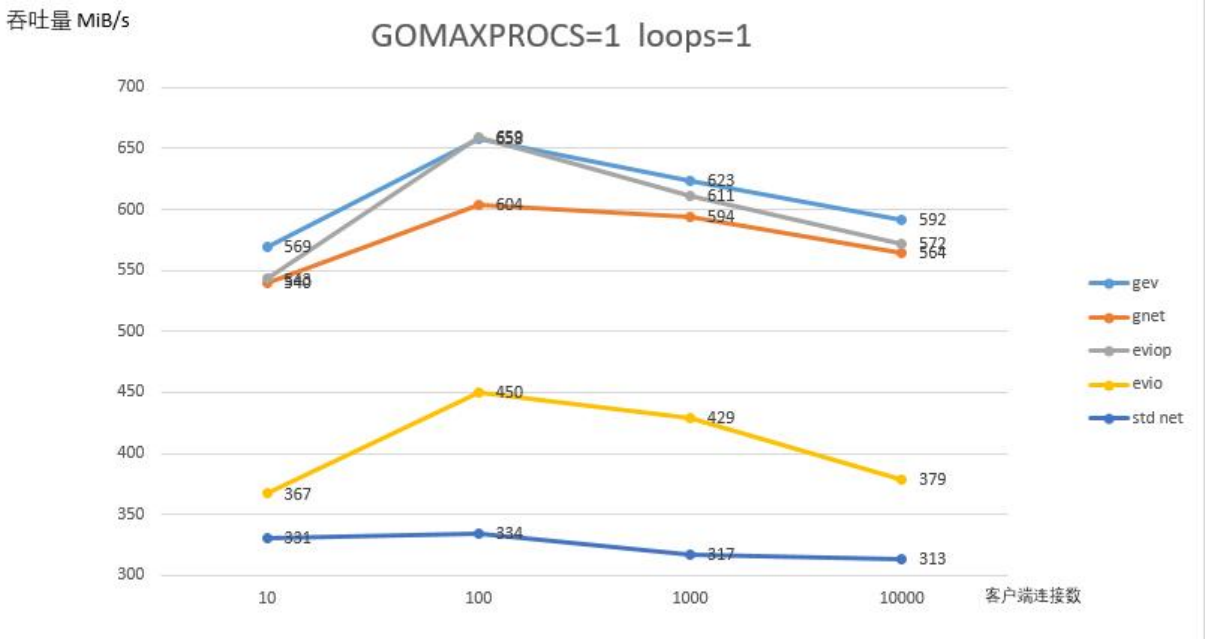
测试脚本：https://github.com/Allenxuxu/gev/blob/master/benchmarks/bench-pingpong.sh

主要做两项测试：

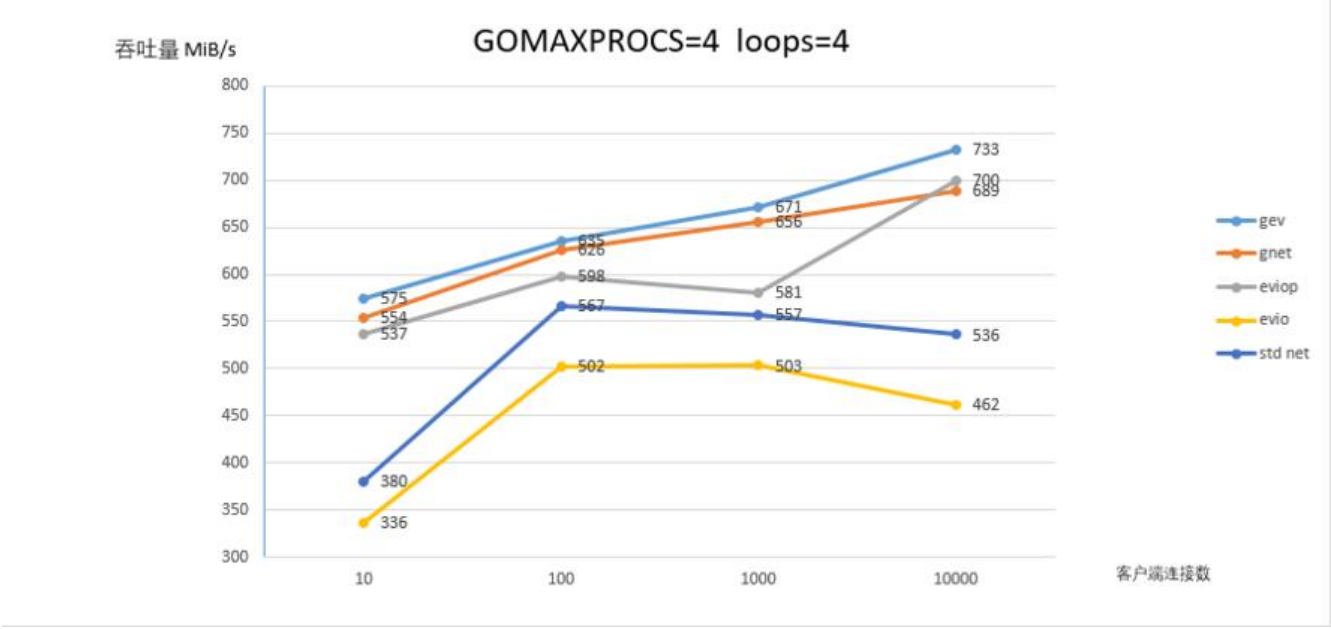
- 单线程单个 work 协程测试，测试并发连接数为 10/100/1000/10000 时的吞吐量
- 4线程4个 work 协程测试，测试并发连接数为 10/100/1000/10000 时的吞吐量

所有测试中，ping pong 消息的大小均为 4096 bytes，客户端始终是4线程运行。

测试结果



原文链接：Go 网络库并发吞吐量测试



总结与思考

无论是单线程，还是多线程模式下，gev 都比其他网络库吞吐量略高出一些。

evio 因为 epoll 使用一些 bug 和可优化之处，所以在 linux 环境中的吞吐量远不如优化版本 eviop 。

eviop 是我对 evio bug 修复和优化的版本，所以其性能也是比 evio 提升不少。我曾尝试在 eviop 替换 evio 的网络模型（evio 利用 accpet 的惊群现象工作），但是因为其代码耦合度过高，修改成过大，最终决定一边完善 eviop（维持网络模型不变）一边自己借鉴muduo 的网络模型重新撸一个的 -- gev 。

gnet 是研究了 eviop 的代码，继续在其之上替换网络模型的版本。但是网络模型的优势在单线程模中并没有体现出来，吞吐量反而比 eviop 小一些。在多线程模式下，网络模型的优势得以体现。

gev 与其他使用 epoll 构建的基于事件驱动的网络库在逐步的优化中，相信性能都差不多。因为作者的不同，网络库不同的设计，优势点都会不同。我研究 evio，最终自己撸了 gev ，也是因为想要一在内存占用低前提下，速度足够快，能负载更多连接的网络库。

如果对 gev 网络库感兴趣，欢迎提意见和 PR 。□□arr
w_right https://github.com/Allenxuxu/gev