链滴

[开源] gev （支持 websocket 啦）：Go 实现基于 Reactor 模式的非阻塞网络库

作者：Allenxuxu

原文链接：https://ld246.com/article/1571884776910

来源网站：链滴

https://github.com/Allenxuxu/gev

gev 是一个轻量、快速、高性能的基于 Reactor 模式的非阻塞网络库，底层并不使用 golang net 库而是使用 epoll 和 kqueue。

现在它支持 WebSocket 啦!

支持定时任务，延时任务!

□□arrow_down□□arrow_down□□arrow
down

特点

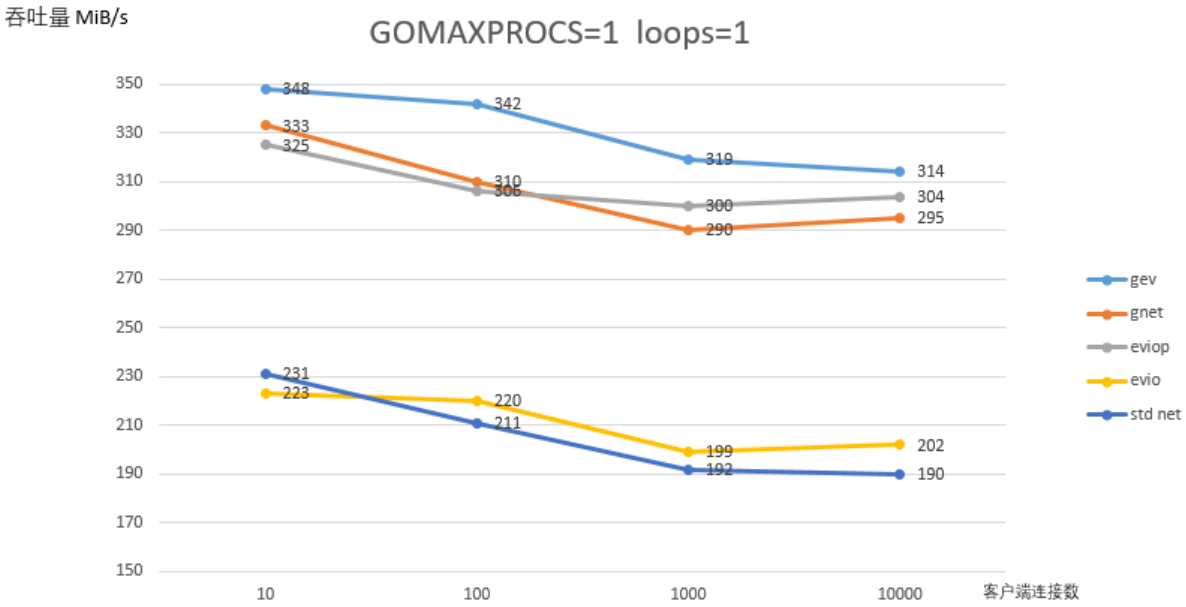
- 基于 epoll 和 kqueue 实现的高性能事件循环
- 支持多核多线程
- 动态扩容 Ring Buffer 实现的读写缓冲区
- 异步读写
- SO_REUSEPORT 端口重用支持
- 支持 WebSocket
- 支持定时任务，延时任务

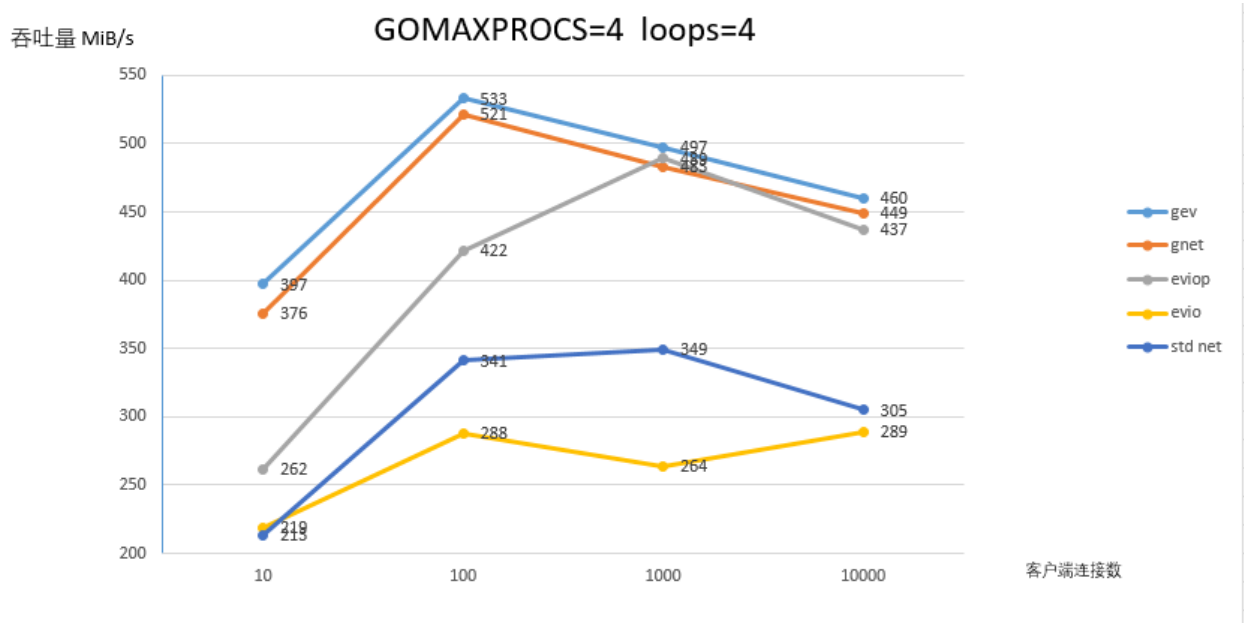
性能测试

测试环境 Ubuntu18.04

- gev
- gnet
- eviop
- evio
- net (标准库)

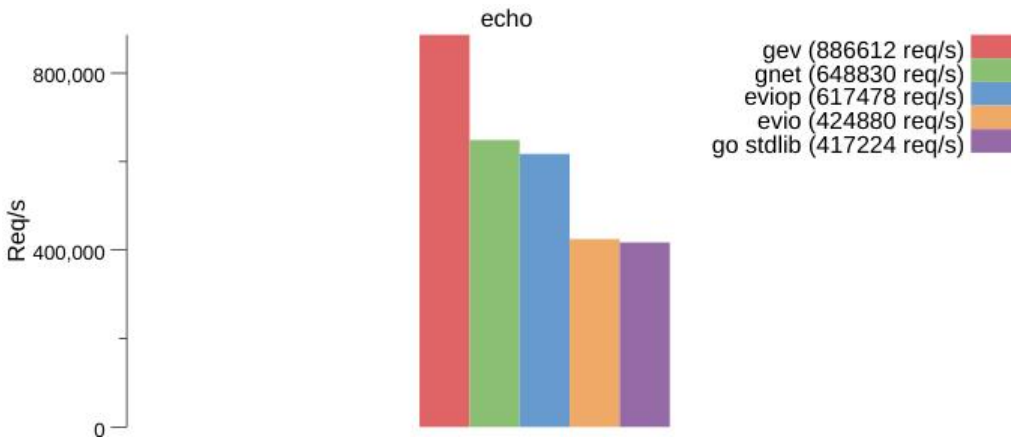
吞吐量测试





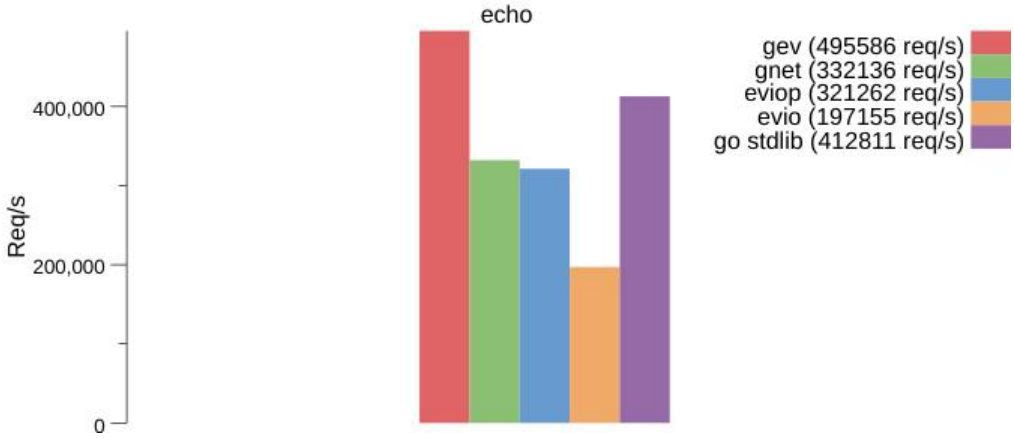
evio 压测方式:

限制 GOMAXPROCS=1，1 个 work 协程

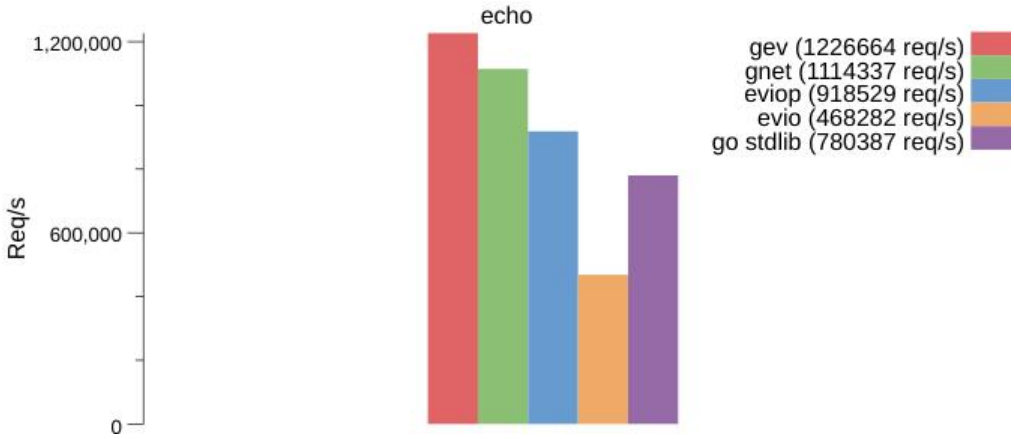


原文链接：[开源] gev （支持 websocket 啦）: Go 实现基于 Reactor 模式的非阻塞网络库

限制 GOMAXPROCS=1，4 个 work 协程



限制 GOMAXPROCS=4，4 个 work 协程



安装

```
go get -u github.com/Allenxuxu/gev
```

示例

TCP

```
package main

import (
    "flag"
    "strconv"
    "log"

    "github.com/Allenxuxu/gev"
    "github.com/Allenxuxu/gev/connection"
    "github.com/Allenxuxu/ringbuffer"
)

type example struct{}
```

```
func (s *example) OnConnect(c *connection.Connection) {
    log.Println(" OnConnect :   ", c.PeerAddr())
}
func (s *example) OnMessage(c *connection.Connection, buffer *ringbuffer.RingBuffer) (out []
yte) {
    //log.Println("OnMessage")
    first, end := buffer.PeekAll()
    out = first
    if len(end) > 0 {
        out = append(out, end...)
    }
    buffer.RetrieveAll()
    return
}

func (s *example) OnClose() {
    log.Println("OnClose")
}

func main() {
    handler := new(example)
    var port int
    var loops int

    flag.IntVar(&port, "port", 1833, "server port")
    flag.IntVar(&loops, "loops", -1, "num loops")
    flag.Parse()

    s, err := gev.NewServer(handler,
        gev.Network("tcp"),
        gev.Address(":"+strconv.Itoa(port)),
        gev.NumLoops(loops))
    if err != nil {
        panic(err)
    }

    s.Start()
}
```

WebSocket

```
package main

import (
    "flag"
    "github.com/Allenxuxu/gev/ws"
    "log"
    "math/rand"
    "strconv"

    "github.com/Allenxuxu/gev"
    "github.com/Allenxuxu/gev/connection"
)
```

```
type example struct{}

func (s *example) OnConnect(c *connection.Connection) {
    log.Println(" OnConnect :   ", c.PeerAddr())
}
func (s *example) OnMessage(c *connection.Connection, data []byte) (messageType ws.Mess
geType, out []byte) {
    log.Println("OnMessage:", string(data))
    messageType = ws.MessageBinary
    switch rand.Int() % 3 {
    case 0:
        out = data
    case 1:
        if err := c.SendWebsocketData(ws.MessageText, data); err != nil {
            if e := c.CloseWebsocket(err.Error()); e != nil {
                panic(e)
            }
        }
    case 2:
        if e := c.CloseWebsocket("close"); e != nil {
            panic(e)
        }
    }
    return
}

func (s *example) OnClose(c *connection.Connection) {
    log.Println("OnClose")
}

func main() {
    handler := new(example)
    var port int
    var loops int

    flag.IntVar(&port, "port", 1833, "server port")
    flag.IntVar(&loops, "loops", -1, "num loops")
    flag.Parse()

    s, err := gev.NewWebSocketServer(handler,
        gev.Network("tcp"),
        gev.Address(":"+strconv.Itoa(port)),
        gev.NumLoops(loops))
    if err != nil {
        panic(err)
    }

    s.Start()
}
```

相关文章

- evio源码解析

- Golang 网络库 evio 一些问题/bug和思考
- gev: Go 实现基于 Reactor 模式的非阻塞 TCP 网络库

仓库地址： https://github.com/Allenxuxu/gev