Lab2 test report

1. 实验概要

使用从课程中学到的网络编程知识，从头开始实现基于 HTTP/1.1 的 HTTP 服务器。另外，尝试使用从类中学习的高并发编程技能来保证 web 服务器的性能。

目标：

练习基本的网络编程技巧，如使用 socket API，解析数据包；

熟悉健壮的高性能并发编程。

1.1 HTTP Server 概述

从网络的角度来看，HTTP 服务器应该实现以下功能：

A、创建侦听套接字并将其绑定到端口

B、等待客户端连接到端口

C、接受客户端并获取新的连接套接字

D、读入并分析 HTTP 请求

E、开始交付服务：1）处理 HTTP GET/POST 请求，并在发生错误时返回错误消息。2） 将请求代理到另一个 HTTP 服务器（高级版本可选）。

1.2 HTTP Server 请求

在这个实验中，只需要在 HTTP 服务器中实现 GET 方法和 POST 方法。也就是说，如果 HTTP 服务器接收到 HTTP 请求，但请求方法既不是 GET 也不是 POST，HTTP 服务器只需要返回一条 501 未实现的错误消息（响应行状态代码为 501 的响应消息）。

不需要处理请求中的标题行（但需要识别它，以便不会将其与下一个请求的起始行混合）。此外，可以随意填充 HTTP 响应中的任何（或零）标题行。

1.3 多线程处理

本次实验应使用多线程提高并发性

HTTP 服务器应该使用多个线程来处理尽可能多的并发客户端请求。至少有以下三个选项来设计多线程服务器：

On-demand thread:可以在新客户机进入时创建一个新线程，并使用该线程

处理所有客户机的任务，包括解析 HTTP 请求、获取页面文件和发送响应。在客户端完成后，可以销毁线程，例如，通过 TCP recv（）进行检测。但是，在 HTTP 层中检测客户端完成可能并不容易。

A pool of always-on threads：可以在 HTTP 服务器程序中使用固定大小的线程池来同时处理多个客户端请求。如果没有任务，则这些线程处于等待状态。如果新的客户机进入，则分配一个线程来处理客户机的请求并向其发送响应。如果分配的线程正忙，可以使用工作队列缓冲请求，并让线程稍后处理它。

Combined：将以上两种风格结合在一起。例如，可以使用线程池接收请求和发送响应，并使用按需线程获取大型页文件。

1.4 实验环境

实验使用 Ubuntu 16

2. 性能测试

2.1 HTTP Server 运行结果

网页测试：

请求与响应：

```
# ling @ ling-VirtualBox in ~ [2:51:50] C:7
$ curl -i -X GET 127.0.0.1:8888/index.html
HTTP/1.1 200 OK
Content-Type: text/html
Server: My Web Server
Content-length: 246

<html><head>
<title><CS06142></title>
</head><body>
<h1>CS06142</h1>
<p>our own httpserver<br />
</p>
<hr>
<address>Http Server at ip-127-0-0-1 port 8888</address>
</body></html>
curl: (18) transfer closed with 67 bytes remaining to read
```

```
# ling @ ling-VirtualBox in ~ [2:56:01] C:148
$ curl -i -X GET 127.0.0.1:8888/l/
HTTP/1.1 404 Not Found
Content-Type: text/html
Server: My Web Server
Content-length: 217

<html><title>404 Not Found</title><body bgcolor=ffffff>
  Not Found
<p>Couldn't find this file:/l/
<hr><em>HTTP Web server</em>
</body></html>
curl: (18) transfer closed with 74 bytes remaining to read
```

```
# ling @ ling-VirtualBox in ~ [2:56:05] C:18
$ curl -i -X DELETE http://127.0.0.1:8888/index.html
HTTP/1.1 501 Not Implement
Content-Type: text/html
Server: My Web Server
Content-length: 242

<html><title>501 Not Implemented</title><body bgcolor=ffffff>
  Not Implemented
<p>Does not implement this method:DELETE
<hr><em>HTTP Web server</em>
curl: (18) transfer closed with 78 bytes remaining to read
</body></html>%
```

2.2 多线程性能测试

测试工具： ab - Apache HTTP server benchmarking tool

线程池中有 8 线程时:

请求 1000 次,每次并发 100;

```
# ling @ ling-VirtualBox in ~ [4:33:12] C:22
$ ab -n1000 -c100 127.0.0.1:8888/index.html
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 127.0.0.1 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests
```

```
Server Software:        My
Server Hostname:        127.0.0.1
Server Port:            8888

Document Path:          /index.html
Document Length:        179 bytes

Concurrency Level:      100
Time taken for tests:   0.677 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      267000 bytes
HTML transferred:       179000 bytes
Requests per second:    1477.08 [#/sec] (mean)
Time per request:       67.701 [ms] (mean)
Time per request:       0.677 [ms] (mean, across all concurrent requests)
Transfer rate:          385.14 [Kbytes/sec] received
```

根据工具的测试结果，可以看到，并发级别为 100 的情况下，测试总花费时

间为 0.677s，平均每秒完成的请求数为 1477.08，从用户角度完成一次请求的时间为 67.701，服务器完成一个请求的时间为 0.677.

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   3.0      0      16
Processing:     2   20  15.7     16     130
Waiting:        1   18  14.4     12     128
Total:          2   21  15.8     17     130

Percentage of the requests served within a certain time (ms)
  50%     17
  66%     26
  75%     32
  80%     34
  90%     43
  95%     51
  98%     63
  99%     65
 100%    130 (longest request)
```

可以看到，大多数请求（99%）在 65ms 以内就完成了，最长的响应时间则达到了 130ms。

请求 1000 次,每次并发 1000;

```
Server Software:        My
Server Hostname:        127.0.0.1
Server Port:            8888

Document Path:          /index.html
Document Length:        179 bytes

Concurrency Level:      1000
Time taken for tests:   0.656 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      267000 bytes
HTML transferred:       179000 bytes
Requests per second:    1524.16 [#/sec] (mean)
Time per request:       656.101 [ms] (mean)
Time per request:       0.656 [ms] (mean, across all concurrent requests)
Transfer rate:          397.41 [Kbytes/sec] received
```

可以看到，测试的总花费时间基本不变，每秒处理的请求数也提升不大，只有用户角度的每个请求响应时间显著增大。

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   40  33.4     36     104
Processing:    29  228 104.6    273     370
Waiting:       25  224 104.9    270     365
Total:        131  268  76.9    287     399

Percentage of the requests served within a certain time (ms)
  50%    287
  66%    310
  75%    323
  80%    331
  90%    362
  95%    377
  98%    393
  99%    395
 100%    399 (longest request)
```

可以看到，响应时间大多数都在 300ms 以内。