Lab2 test report

成员：冯昂、谢博、魏瑛洁

一、实验的性能结果检测：

（一）实验概要：

多线程编程是实现一个高并发服务器的基础，实验二实现一个简单的支持高并发服务器，并使用 ab 工具对比较不同请求数，不同的并发数情况下测试这个简单的服务器的性能差异

（二）程序输入

通过 ab 工具所提供的相关命令往服务器发起若干个请求，同时可以通过设置 ab 工具的参数指定并发数，例如：

ab -n 5000 -c 500 http://127.0.0.1:8888/index.html

上述命令指定并发数位 500，总共要向服务器发起 500 个请求

（三）性能指标

实验以服务器处理完所有请求所用的时间作为性能指标

（四）实验环境

实验中共一个实验环境：ENV1

ENV1:linux 内核版本为：Linux ubuntu 4.4.0-31-generic；1GB 内存；CPU 型号为 Intel(R) Core(TM) i5-5257U CPU @ 2.70GHz，共一个物理 CPU；每个 CPU 上有 2 个核心；不适用超线程技术。

（五）代码实验版本

本实验公一个版本，见 github

（六）性能测试

程序的性能会受到诸多因素的影响，其中包括软件层面的因素和硬件层面的因素。本节主要比较不同线程，不同请求，不同并发数之间的区别。

（七）Test how many HTTP request your server can process per second, when running on various server machine environments. For example, change the number of server CPU cores, enable/disable hyper-threading, etc.（测试服务器在各种服务器机器环境上运行时每秒可以处理多少 HTTP 请求。例如，

改变服务器 CPU 内核的数量，启用/禁用超线程，等等.）

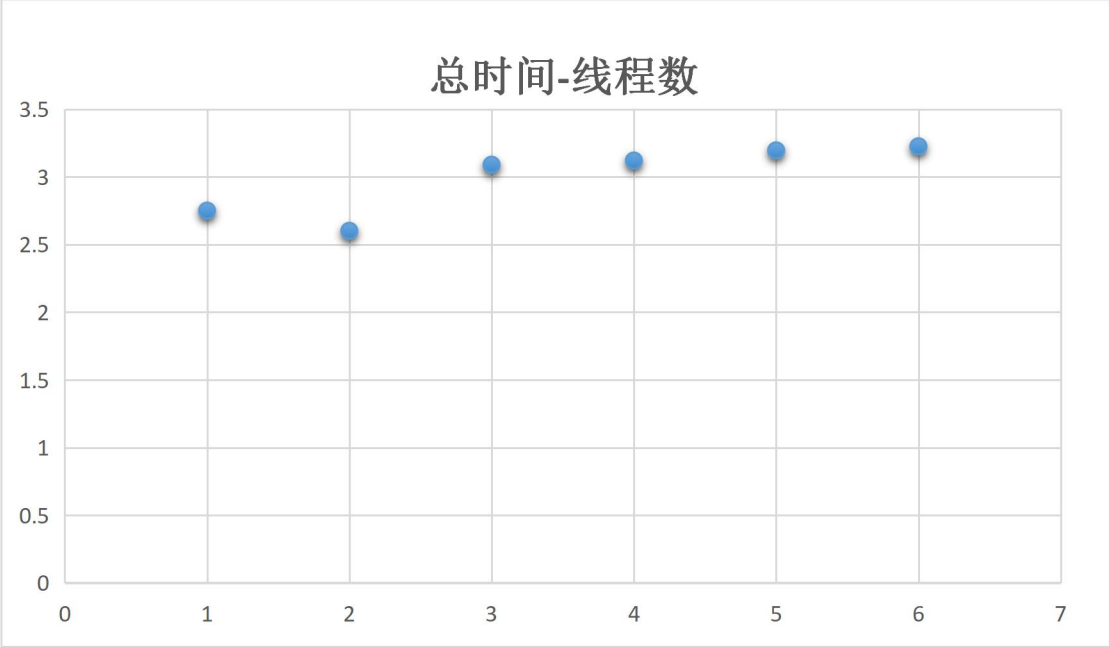
不同线程数之间的比较：

```
Document Path:          /index.html
Document Length:        68 bytes
```

这里客户端移动向服务器端发送将近 5000 个请求，每个请求所请求的字节数位 68bytes，且并发数为 200，测试服务器端不同线程数的实现情况。

线程数	完成时间
1	2.750s
2	2.601s
3	3.089s
4	3.120s
5	3.195s
6	3.225s

多线程程序能使 CPU 的多个核心并行 运作，因此，多线程能够充分发挥多核 CPU 的优势。在一定范围内，加速比会随 着线程数的增加而增长，即时间开销越少、效率越高。当线程数超过 CPU 核心数 时，性能会有所下降。

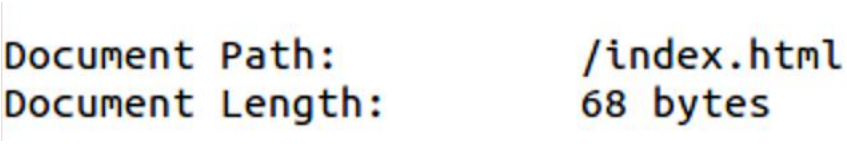


从表上可以看出当线程数小于等于 2 的时候，总的时间是逐渐减少的，因为此时线程数小于等于 cpu 核数，所以 cpu 核的利用率越来越高，但是当线程

数大于 cpu 核数后，所用的时间变得更长，因为此时会出现较快的线程等待较慢的线程的情况，反而会降低效率

（八）Test how many HTTP request your server can process per second, by varying the number of concurrent clients that send request to your server simultaneously.（通过改变同时向服务器发送请求的并发客户机的数量，测试服务器每秒可以处理多少 HTTP 请求。）

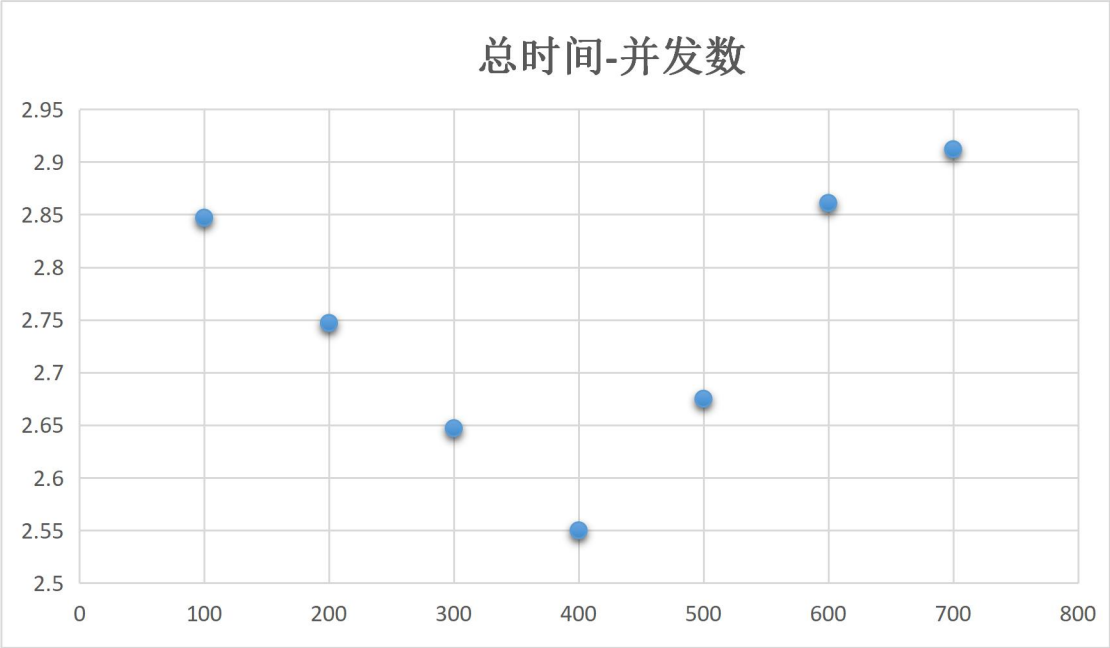
不同并发数之间的比较：



这里将服务器端线程数设置为 2，然后总的请求仍然为 5000，每个请求的数据长度仍然为 68 字节，但是会不断改变并发数，即同时向服务器发起的请求数

并发数	完成时间
100	2.847s
200	2.747s
300	2.647s
400	2.550s
500	2.675s
600	2.861s
700	2.921s

当并发数小于等于 400 时，完成时间随着并发数的增大而减小，当大于 400 后，总的时间随着并发数的增加而逐渐增大，可以看出并发出对服务器的性能也有着比较大的影响。



可以明显的看出并发数为 400 的时候效果最好，当同时向服务器发起请求数大于这个数时会出现客户端等待现象，所以导致总的时间会变大。

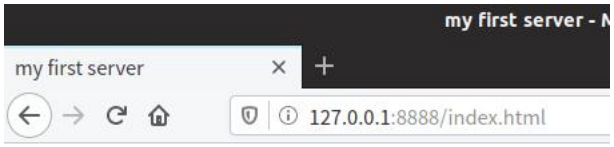
二、实验内容运行过程：

Make 并且运行 HTTP 服务器：

```
wyj@wyj-virtual-machine:~/CloudComputingGroup-master/CloudComputingLabs/Lab2$ ma
ke
g++ -o httpserver main.o http_conn.o threadpool.h http_conn.h locker.h  -pthread
wyj@wyj-virtual-machine:~/CloudComputingGroup-master/CloudComputingLabs/Lab2$ ./
httpserver --ip 127.0.0.1 --port 8888 --number-thread 8 --proxy https://www.CS06
142.com:80
```

访问 HTTP 服务器（Using GET method）

通过打开 web 浏览器并转到适当的 URL 来检查 HTTP 服务器是否工作正常。



Welcome!

正常。

使用 curl 程序发送 HTTP 请求（curl -i -X GET http://127.0.0.1:8888/index.html）

例：

```
wyj@wyj-virtual-machine:~/CloudComputingGroup-master/CloudComputingLabs/Lab2$ cu
rl -i -X GET http://127.0.0.1:8888/index.html
HTTP/1.1 200 OK
Content-Length: 149
Connection: close

<!DOCTYPE html>
<html>
    <title>my first server</title>
    <head>
        <body>
            <h1>Welcome!</h1>
        </body>
    </head>
</html>wyj@wyj-virtual-machine:~/CloudComputingGroup-master/CloudComputingLabs/L
```

如果请求页面不存在，HTTP 服务器应该返回 404 not Found 错误消息.

例：

```
wyj@wyj-virtual-machine:~/CloudComputingGroup-master/CloudComputingLabs/Lab2$ curl -i -X GET htt
P 0.1:8888/index1.html
HTTP/1.1 404 Not Found
Content-Length: 49
Connection: close

The requested file was not found on this server.
```