

“被安排了” 组-Lab2 test report

1. 实验概要

在本实验中，我们将不提供任何基本代码。因此，您应该从头开始实现满足以下要求的HTTP服务器（基于HTTP / 1.1）

1.1 HTTP服务器概述

从网络角度来看，您的HTTP服务器应实现以下功能：

1. 创建一个监听套接字并将其绑定到端口
2. 等待客户端连接到端口
3. 接受客户端并获取新的连接套接字
4. 读入并解析HTTP请求
5. 开始提供服务：1）处理HTTP GET / POST请求，如果发生错误，则返回错误消息。2）将请求代理到另一个HTTP服务器（高级版本可选）。
6. 服务器将处于非代理模式或代理模式。它不会同时做两件事。

1.2 处理HTTP请求

在本实验中，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，但是请求方法既不是GET也不是POST。HTTP服务器仅需要返回501 Not Implemented错误消息（响应消息的响应行的状态码为501，请参见2.2）。不需要处理请求中的标题行（但是需要识别它，以便不会将其与下一个请求的起始行混合使用）。另外，可以随意在HTTP响应中填充任何（或零个）标题行。

1.2.1 处理HTTP GET请求

HTTP服务器应该能够处理html页面的HTTP GET请求。

1、如果HTTP请求的路径与html页面文件相对应，则以200 OK和文件的完整内容进行响应。例如，如果请求GET /index.html，并且在文件目录中存在一个名为index.html的文件。您还应该能够处理对文件目录子目录中文件的请求（例如GET /images/hero.jpg）。

2、如果HTTP请求的路径对应于一个目录，并且该目录包含一个index.html文件，则以200 OK响应，并在该文件夹中显示index.html文件的全部内容。

3、如果请求的页面文件不存在，或者请求的目录不包含index.html文件，则返回404 Not Found响应（HTTP正文是可选的）。

1.2.2 处理HTTP POST请求

HTTP服务器应该能够处理HTTP POST请求。在本实验中，处理HTTP POST的方法非常简单。

1、您应该构造一个3.1.7.2包含2个键的HTTP POST请求（请参阅参考资料）：“名称”和“ID”（请分别填写您的姓名和学生编号），然后将POST请求发送至/Post_show（例如http://127.0.0.1:8888/Post_show，服务器的IP地址127.0.0.1和服务端口是8888）。然后，如果HTTP服务器收到此POST请求（请求URL为/Post_show，且键为“Name”和“ID”），则响应为200 OK，并回显已发送的“Name”-“ID”否则（即请求URL不是/Post_show或键不是“Name”和“ID”），返回404 Not Found响应消息。

1.2.3其他要求

对于其他请求方法（例如DELETE，PUT等），仅返回501 Not Implemented错误消息。

1.3使用多进程增加并发

您的HTTP服务器应使用多个线程来处理尽可能多的并发客户端的请求。您至少具有以下三个选项来构建多线程服务器：

1、**按需线程**。每当有新客户端进入时，您都可以创建一个新线程，并使用该线程来处理所有客户端的任务，包括解析HTTP请求，获取页面文件和发送响应。客户端完成后可以销毁线程，例如，通过TCP进行检测recv()。但是，在HTTP层中检测客户端是否完成可能并不容易。

2、**永远在线的线程池**。您可以在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。如果有新客户端进入，请分配一个线程来处理该客户端的请求并发送响应。如果分配的线程繁忙，则可以使用工作队列来缓冲请求，然后让线程稍后对其进行处理。

3、**合并**。将以上两种样式结合在一起。例如，您可以使用线程池来接收请求和发送响应，并使用按需线程来获取大页面文件。

随意选择以上三个中的任何一个，或使用您认为很酷的其他多线程体系结构。

1.4指定参数

您的程序应启用长选项以接受参数并在启动期间指定这些参数。它们是--ip，--port和--proxy（可选）。

1. --ip -指定服务器的IP地址。
2. --port -选择HTTP服务器侦听传入连接的端口。
3. --proxy -选择一个“上游” HTTP服务器进行代理。该参数可以在冒号后面有一个端口号（例如https://www.CS06142.com:80）。如果未指定端口号，则默认为端口80。

1.5运行HTTP服务器

程序应该能够在终端启动。

1.6访问HTTP服务器

1.6.1使用GET方法

1. 可以通过打开Web浏览器并转到相应的URL来检查HTTP服务器是否正常工作。[注] IP 127.0.0.1是指本地主机的IP。因此，您可以使用127.0.0.1在同一台本地计算机上测试您的HTTP服务器。

2. 还可以使用curl程序发送HTTP请求。如何使用curl的示例是：

```
curl -i -X GET http://127.0.0.1:8888/index.html
```

3. 如果请求页面不存在，则您的HTTP服务器应返回404 Not Found错误消息。

1.6.2使用POST方法

1. 可以通过使用curl程序发送HTTP请求来检查POST方法是否有效。在终端上键入命令：

```
curl -i -X POST --data 'Name=HNU&ID=CS06142' http://127.0.0.1:8888/Post_show
```

2. 如果请求URL不是/Post_show或键不是“名称”和“ID”，则会收到404 Not Found错误消息。

3. 还可以构造POST HTTP请求，并使用某些浏览器插件工具将请求发送到HTTP服务器。

1.6.3其他方法

除GET请求和POST请求外，HTTP服务器将不处理HTTP请求。

如果您发送HTTP DELETE（或PUT，HEAD等）请求以删除指定的资源，则会收到501 Not Implemented错误消息

1.7性能指标

1、在各种服务器计算机环境上运行时，测试您的服务器每秒可以处理多少个HTTP请求。例如，更改服务器CPU内核数，启用/禁用超线程等。

2、通过更改同时向服务器发送请求的并发客户端数，测试您的服务器每秒可以处理多少个HTTP请求。一定要改变客户的工作量。例如，测试客户端何时将新的TCP连接用于新请求，或何时客户端将旧的TCP连接重新用于新请求。此外，如果实施高级版本，请尝试更改同一客户端的TCP连接上的出站请求数。您可以编写一个自己发送HTTP Get的简单客户端（可以在同一台计算机上运行多个客户端程序来模拟多个客户端），也可以使用一些现有的HTTP客户端测试工具，例如ab-Apache HTTP服务器基准测试工具。

2. 性能测试

硬件环境如 CPU 主频（其它如物理 CPU 个数、物理核心数等）的不同，会导致同一个程序在不同的硬件环境中有不同的表现。对单个 CPU 物理核心，其主频越高，运行速度越快。

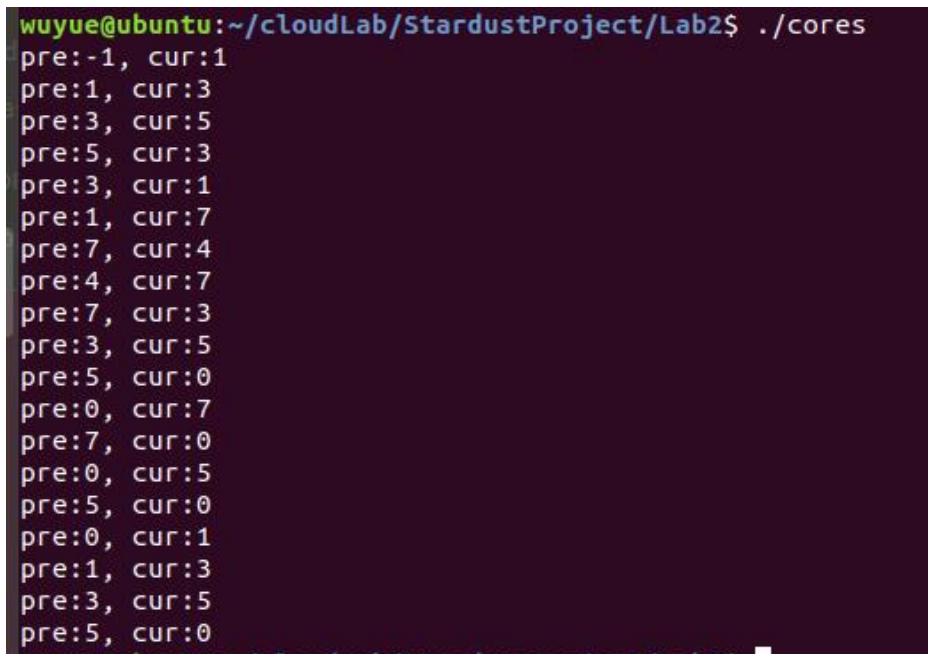
1、在各种服务器计算机环境上运行时，测试您的服务器每秒可以处理多少个HTTP请求。例如，更改服务器CPU内核数，启用/禁用超线程等。

Cores.c这段代码中启动了一个线程，通过sched_getcpu函数不停检测当前占用的逻辑核心编号。

```
// build script: gcc -lpthread bind_core.c -o bind_core
#include <stdio.h>
#include <pthread.h>
#include <sched.h>
void* thread_routine(void* arg) {
    int cpu = -1;
    while(1) {
        int cur_cpu = sched_getcpu();
        if (cur_cpu != cpu) {
            printf("pre:%d, cur:%d\n", cpu, cur_cpu);
            cpu = cur_cpu;
        }
    }
};

int main() {
    pthread_t thread;
    pthread_create(&thread, NULL, thread_routine, NULL);
    pthread_detach(thread);
    sleep(100);
}
```

输出如下:

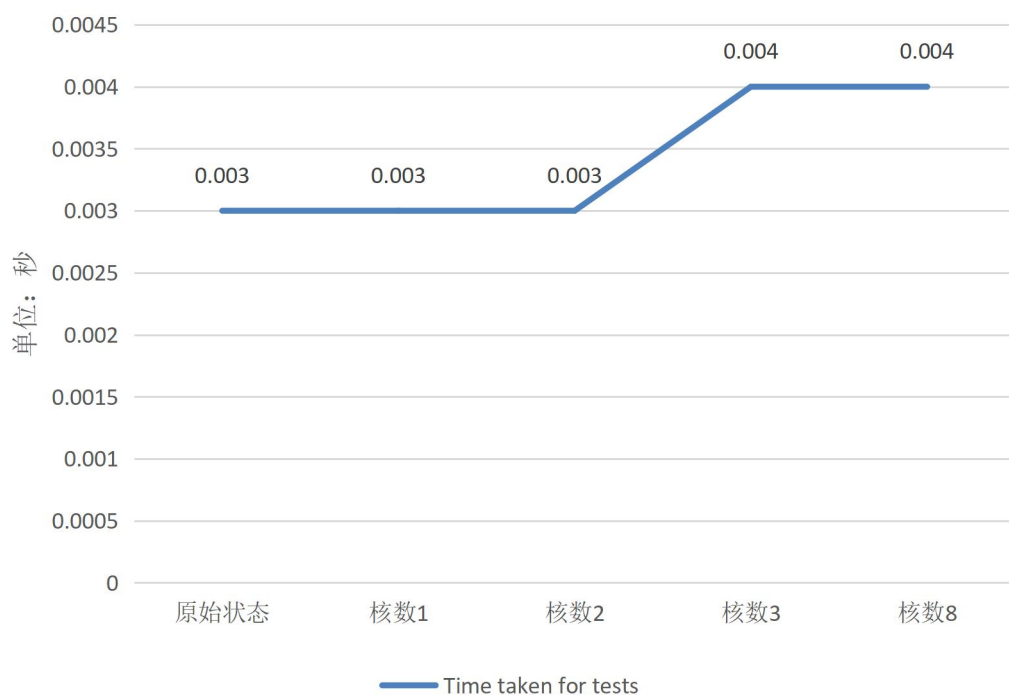


```
wuyue@ubuntu:~/cloudLab/StardustProject/Lab2$ ./cores
pre:-1, cur:1
pre:1, cur:3
pre:3, cur:5
pre:5, cur:3
pre:3, cur:1
pre:1, cur:7
pre:7, cur:4
pre:4, cur:7
pre:7, cur:3
pre:3, cur:5
pre:5, cur:0
pre:0, cur:7
pre:7, cur:0
pre:0, cur:5
pre:5, cur:0
pre:0, cur:1
pre:1, cur:3
pre:3, cur:5
pre:5, cur:0
```

上图可以看出，程序分别在：0, 1, 2, 3, 4, 5, 6, 7号逻辑核上运行过。为了让CPU在固定的核心上执行，我们可以使用taskset指令，让程序绑定逻辑核心。

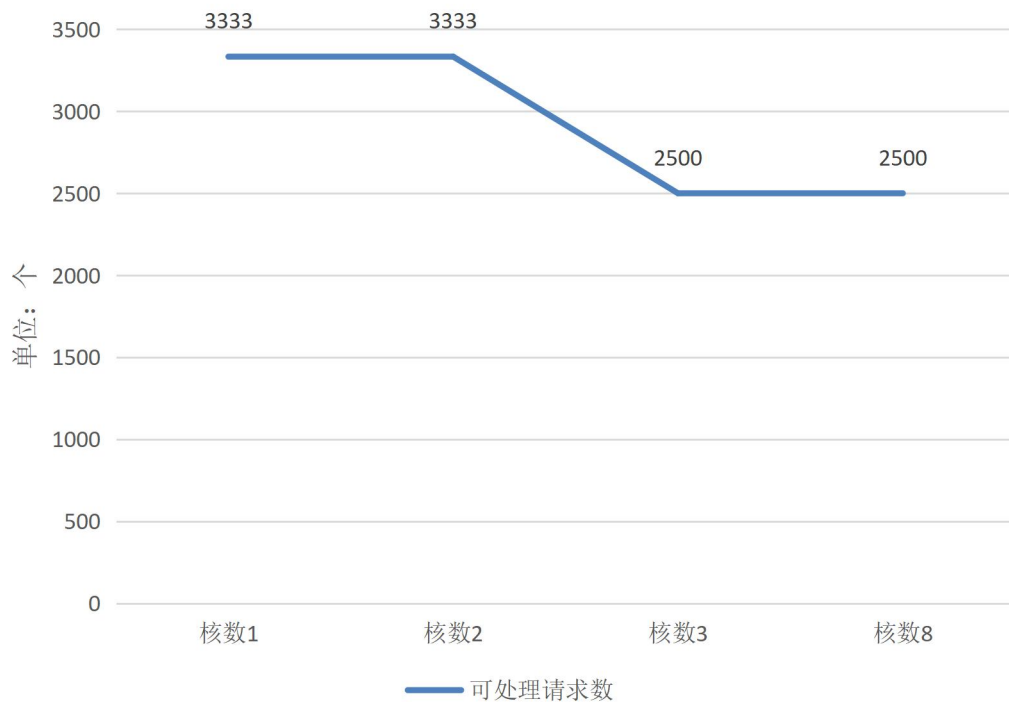
如图2. 1，此次测试HTTP请求均为十个，且都请求成功，并发请求都为1.

图2.1：不同核数处理相同数量请求耗费的时间



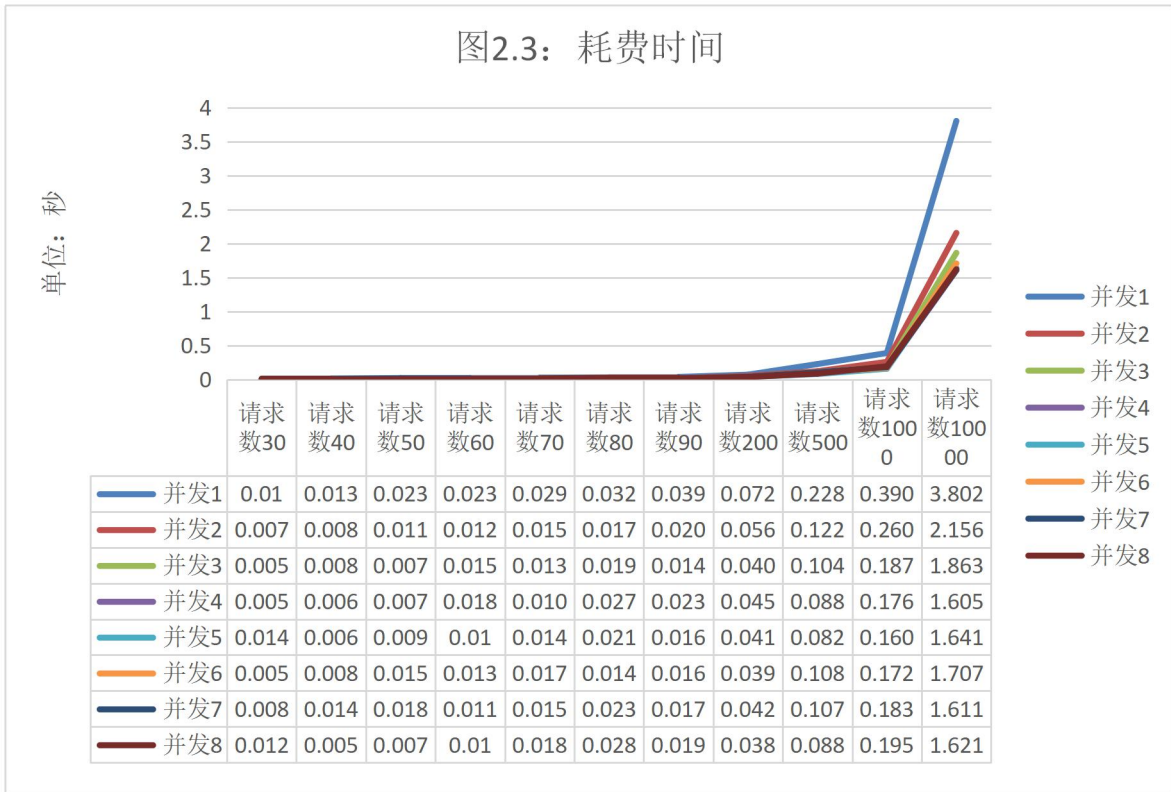
如图2.2 服务器核数不同时每秒可处理的 HTTP 请求

图2.2：服务器核数不同时每秒可处理的HTTP请求



2、通过更改同时向服务器发送请求的并发客户端数，测试您的服务器每秒可以处理多少个HTTP请求。一定要改变客户的工作量。例如，测试客户端何时将新的TCP连接用于新请求，或何时客户端将旧的TCP连接重新用于新请求。

测试结果如图2.3



可以看到随着请求数的增多耗费时间越多，而同一请求数随着并发数的增多耗费时间会少一些。可知增加线程个数会让性能有所提升。