

天津大学

《计算机网络》课程设计报告



HTTP 的设计与实现

学 号: 3020202184 3020244344

姓 名: 刘锦帆 李镇州

学 院: 智能与计算学部

专 业: 计算机科学与技术

年 级: 2020 级

任课教师: 石高涛

2022 年 4 月 11 日

目 录

第一章	报告摘要	1
第二章	任务需求分析	2
第三章	协议设计	4
3.1	总体设计	4
3.1.1	基础代码源文件架构分析	4
3.2	数据结构设计	4
3.3	协议规则设计	4
第四章	协议实现	5
4.1	第一周——实现简单的 echo web server	5
4.2	第二周——实现 HEAD、GET、POST 方法	5
4.3	第三周——实现 HTTP 的并发请求	5
4.4	第四周——实现多个客户端的并发处理	5
第五章	实验结果及分析	6
5.1	第一周——实现简单的 echo web server	6
5.2	第二周——实现 HEAD、GET、POST 方法	6
5.3	第三周——实现 HTTP 的并发请求	6
5.4	第四周——实现多个客户端的并发处理	6
第六章	个人总结	7
附录		

一 报告摘要

二 任务需求分析

在本次实验中，我们完成了一个完整的 HTTP 服务器。能够处理并发的，大量的符合 **HTTP 1.1** 的请求报文，并且能够支持对 **CGI** 请求做特殊处理。以下将分别介绍四周、以及选做的任务需求及分析。

第一周：实现简单的 echo web server 第一关的任务主要分为以下几点：

1. 搭建编程环境、熟悉 Socket 编程方法，为之后的任务奠定基础；
2. 完成消息解析模块，为服务器实现基本的功能做好准备；
3. 完成基本的错误码返回以及基础方法 HEAD、GET、POST 的判断，为以后分别处理不同的报文，同时测试模块化、解耦和的编程框架；

总地来看，第一周任务的主要目的，是让我们熟悉编程环境以及 socket 的基础功能。万事开头难，完成第一关的任务，将对我们完成剩下的任务奠定一个良好、坚实的基础。

第二周：实现 HEAD、GET、POST 方法 第二关的任务主要为以下几点：

1. 完善服务器的功能、能够建立并维系 HEAD、GET、POST 的持久连接；（此时还不需要进行 Pipelining）
2. 支持 4 种 HTTP 1.1 错误码、能够正确封装响应消息；
3. 妥善管理缓冲区、避免客户端请求消息过长导致缓冲区溢出，该项任务主要目的在于处理 1. 下一步 Pipelining 请求过长，以及 2. 后续 CGI 处理 POST 请求时，可能会有大量参数传递，导致溢出的情况；
4. 处理读写磁盘错误，防止因为错误导致服务器宕机等问题；
5. 进行格式化日志的记录，方便正式上线后的 Debug；

第二周的任务是对于 server 处理框架的优化，引导我们去实现独立的消息处理和日志模块等，解耦和、模块化的编程框架为之后的编程提供了便利和良好的可扩展性。

第三周：实现 HTTP 并发请求 第三关的任务主要为：

1. 服务器能够连续相应客户端的 Pipelining 请求；
2. 即使出现了错误的请求，也妨碍服务器进行剩下并发请求的处理；

第三周的任务是对于 server 端的能力进行提升，作为一个比较单独的功能，是根据 RFC2616 文档标准设计出来的并发请求模式。是我们实现标准 HTTP 服务器的关键之处。

第四周：实现多个客户端的并发处理 第四关的任务主要是：

1. 使用 select 函数实现多用户并发请求，在其他用户暂停发送时，服务器能察觉并转而给其他用户提供服务；
2. 服务器的最大连接数量设置为 1024，为 Linux 最大的文件描述符；

第四周的任务是 HTTP 1.1 中较为独立且较为困难的协议之一。在理论上来讲，我们只需要在第三周止步即可实现一个完整的服务端程序。然而第四周的 select 为多用户并发的情况进行了非常好的优化，是向市面上的服务端靠齐的一个关键。在这一关中，我们还将使用 apache bench 对服务器性能进行评测，这也是向正规服务器看齐的关键。

选做：CGI

1. 根据 RFC3875 以及 RFC2396 文档，实现 CGI 的请求；
2. CGI 请求的处理主要靠 URI 区别，且将用一个新的线程处理该请求；

该任务的完成，将标志着我们的服务器走向新的一个台阶：它将能够正确处理 POST 请求，且拥有作为后端程序与前端界面进行交互的能力。当然，对于 session 等更高级的网络编程工具的支持，尚未开发。至少通过我们的实现，我们将能够实现一个非常基础的用户注册、登陆接口。

由于我们将子线程同步采用 Python - 数据库的技术栈，将能够对用户信息进行记录，同时方便对以后想添加的其他功能进行扩展。

如图 ?? 我们将在自己的阿里云服务器上部署项目，方便测试与演示。

三 协议设计

3.1 总体设计

3.1.1 基础代码源文件架构分析

首先我们通过 tree 命令获得如图

3.2 数据结构设计

3.3 协议规则设计

四 协议实现

- 4.1 第一周——实现简单的 echo web server
- 4.2 第二周——实现 HEAD、GET、POST 方法
- 4.3 第三周——实现 HTTP 的并发请求
- 4.4 第四周——实现多个客户端的并发处理

五 实验结果及分析

- 5.1 第一周——实现简单的 echo web server
- 5.2 第二周——实现 HEAD、GET、POST 方法
- 5.3 第三周——实现 HTTP 的并发请求
- 5.4 第四周——实现多个客户端的并发处理

六 个人总结

附 录

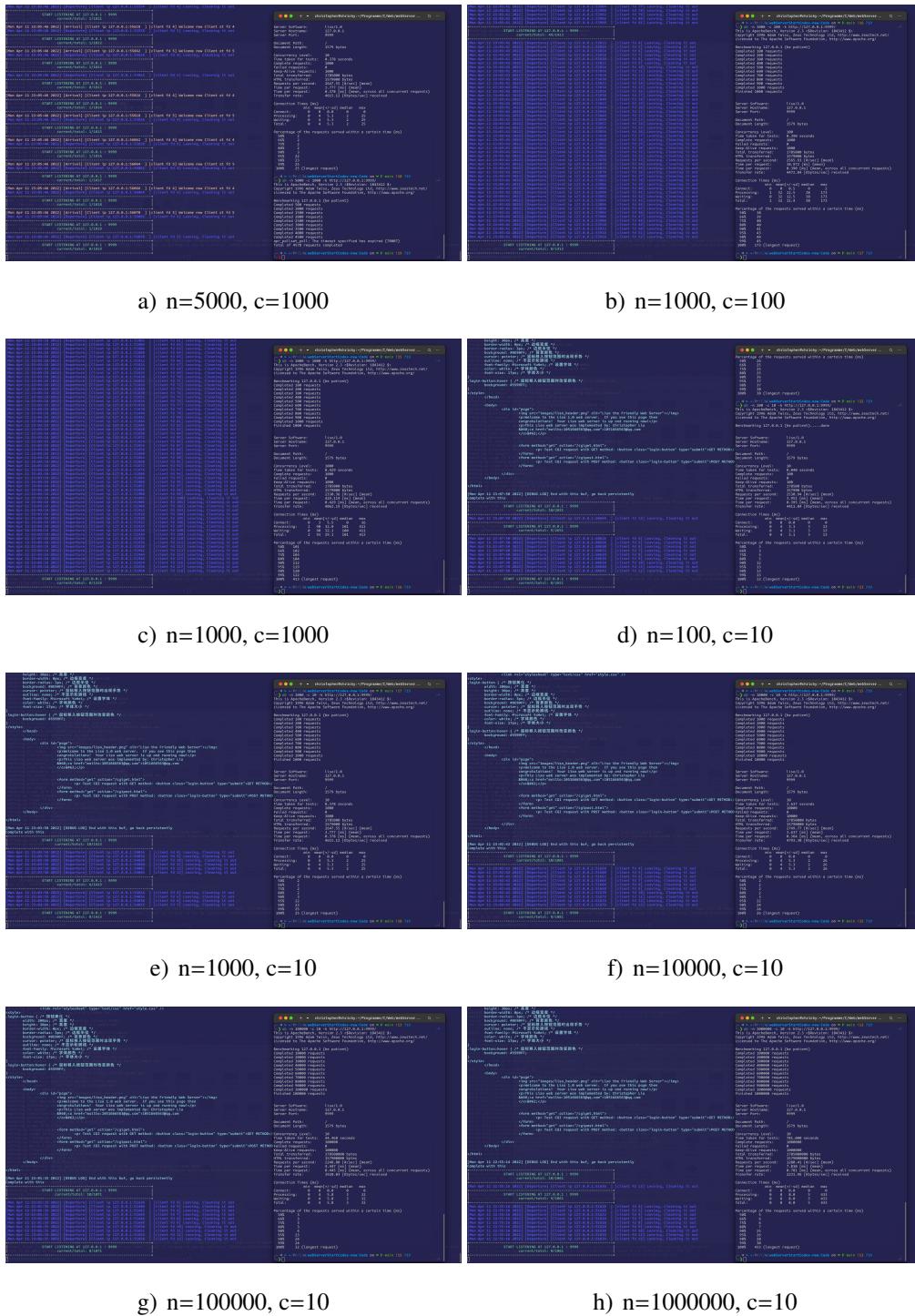


图 6-1 Apache Bench 测试