Hints for ICS ProxyLab

谭亦轩
2022/12/21

Overview

ProxyLab 中，你需要实现一个带有 cache 的 HTTP 代理服务器。其的位置在 HTTP client 与 HTTP server 之间，对于 HTTP client 来说，你的 proxy 是 server，而对于 HTTP server 来说，你的 proxy 是 client。

你需要做的是对 HTTP client 发送的 `HTTP/1.0 GET` 请求进行处理，若 cache hit，直接返回 cache 中的内容；若 cache miss，则去从 HTTP server 中请求对应内容，并更新 cache。

Implementation Suggestion

参照 csapp 11.6 节实现 proxy 的功能。

参照 12.3.8 节或 12.5.5 节实现并发（推荐后者，前者对于每个请求都会创建一个线程，若请求过多可能导致线程过多而很慢甚至操作系统崩溃）。

对于 cache，**我认为** 全局数据结构通过一个大锁同步在 ICS 课程中是可以接受的（这点我的观点和 handout 不一样），当然如果能设计出不需要一个大锁进行同步的数据结构是更好的。

关于 cache 的替换策略，有一种经典的 LRU 的近似：Clock 算法，可以参见这篇博客。

另外，ProxyLab 需要自己设计文件结构，自己写 `Makefile`。建议大家先构思好每个模块要干什么，要对外提供什么接口，然后再开始写代码（Think ten times, code once）。

Debug Suggestion

好习惯是做单元测试，但我相信大家肯定不愿意干（我也不愿意在这个小 lab 上写单元测试）。可以借鉴单元测试的思路，比如每个模块在对外提供好接口以后，先用一种 naive 的方法实现，以此来确定这种模块以外其他模块的正确性。
比如 cache 模块，我对每个 find 请求都返回 cache miss，这样等于所有操作都不经过 cache，就可以验证其他模块的正确性。

运行时 `gdb` 调试在多线程情况下不是一个好选择（至少我用的时候头很大）。

建议的调试方法是善用输出函数，自己输出重要变量，以及进行一些 `assert`。（例如 MallocLab 的 `mm_checkheap`）

Grading

还是参见之前的评分标准（比如每个文件的头部注释，每个函数的注释，一行 80 个字符 等等）。

单独说明的一点是，模块设计是有分的。这个我不会扣得太严，但你不要一个 `.c` 文件就把所有函数全写进去了，这种情况肯定得扣分。

More Information

曾经的 ProxyLab 写完是能够作为浏览器代理在 B 站看视频的，现在大部分视频网站都用 HTTPS 了，就不太行了。当然，如果你能找到一些还在用 HTTP 的网站，这个 lab 是完全能够作为真正的 proxy 处理 `GET` 请求的。

我去年想过能不能用 C++，我不确定 autolab 上是否允许，建议大家还是用 native C 写，符合 ICS 这门课程的要求。

评测机会比本地慢，大家实现的时候注意速度（可以开启编译器优化，但代码中一些 Undefined Behavior 会导致开启编译器优化后出错）。