Bài tập Socket - Đề #1

Thông tin

- Bài tập môn Mạng máy tính - CSC10008(18CLC3) - FIT @ HCMUS
- **Mã số sinh viên:** 18127185
- **Họ và tên:** Bùi Vũ Hiếu Phụng

Danh sách công việc

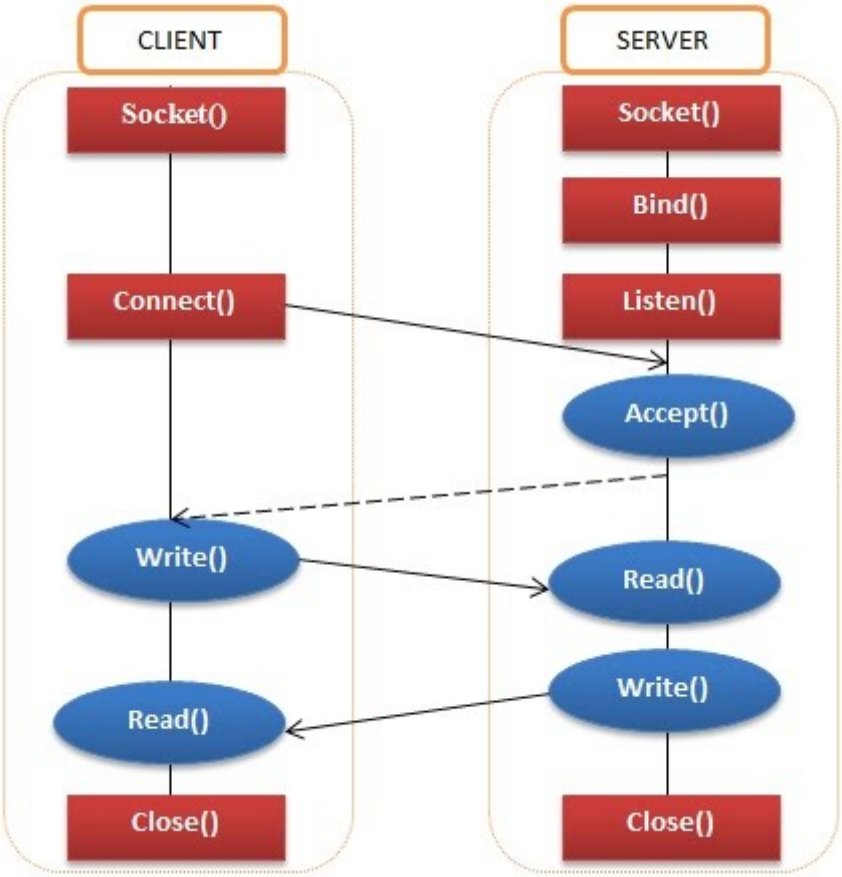
STT	Ngày	Công việc	Tiến độ	Note
1	26/4/2020	Lập trình mô hình server - client cơ bản	✔	
2	27/04/2020	Try - catch Tách hàm Crawl web	✔	(*) scrapy
3	28/04/2020	Hoàn thiện crawl web p2 Xử lí file json Truy vấn 'h'	✔	
4	29/04/2020	Truy vấn <TenTinhThanh> Xử lí tiếng việt Truy vấn <TenTinhThanh> <số vé>	✔	(*) unicode
5	30/04/2020	Server tự động update data (crawl) khi được khởi động	✔	
6	1/5/2020	Viết báo cáo		

Báo cáo

- **Hệ điều hành:** Windows
- **Ngôn ngữ lập trình:** Python
- **IDE/Code editor:** Visual Studio Code

Giao thức kết nối

- Ta có hai giao thức kết nối có thể cài đặt là UDP và TCP. Mỗi giao thức có những điểm mạnh và yếu khác nhau. Ở bài tập này, em chọn giao thức TCP để cài đặt hệ thống tra cứu xổ số của mình vì:
    - Sau khi người dùng gửi truy vấn thì kết quả trả về của truy vấn đó phải được gửi lại đầy đủ, tránh mất mát thông tin
    - Người dùng phải thiết lập kết nối trước khi được server phục vụ



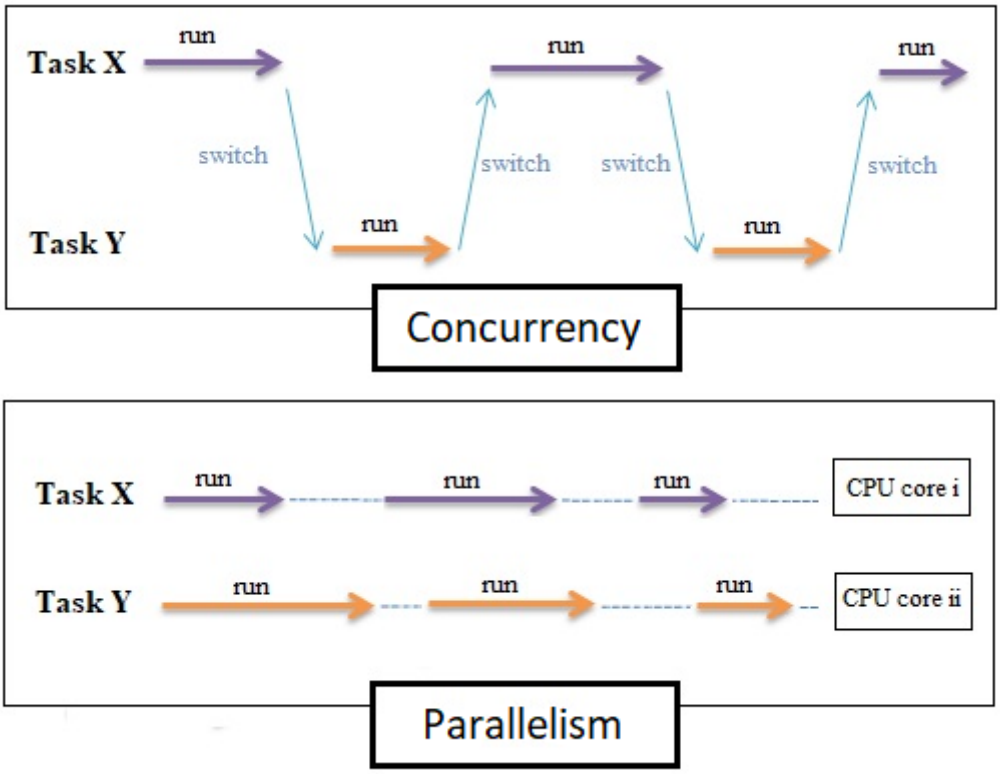
Mô hình bài làm

- Có 3 mô hình xử lí các kết nối: Tuần tự, giả song song, song song.
    - Đối với mô hình xử lí tuần tự thì các client gửi truy vấn sau sẽ phải đợi cho client gửi truy vấn trước thực hiện xong mới được thực hiện khiến thời gian chờ của client kéo dài
    - Đối với mô hình xử lí song song, ta có thể sử dụng multiprocessing. Nhưng việc tạo ra một process và thực hiện nó song song với các process khác có chi phí rất cao, nhất là khi mỗi process có một vùng nhớ riêng.

  → Em chọn mô hình xử lí giả song song hay đồng thời (concurrency) sử dụng multithreading.

  ⇒ **Mỗi client là một thread.**

- Multithreading trong python bị giới hạn bởi GLI (Global Interpreter Lock) nên các thread không thể xử lí song song được, nhưng các thread có thể trao quyền sử dụng CPU cho thread khác. Điều này khiến cho tất cả các client nghĩ rằng mình được phục vụ một cách công bằng như nhau. Ngoài ra thread cũng dùng ít tài nguyên hơn process



Mô tả tóm tắt các hàm chính

Giao thức truyền nhận tin

Vì khi truyền nhận dữ liệu giữa server - client, ta phải định nghĩa độ dài tin nhắn được nhận nên trong chương trình này có giao thức truyền tin riêng

- Khi tin nhắn được chuyển đi
    - Xử lí tin nhắn của người gửi thành 1 package gồm 2 phần: header và tin nhắn. Trong đó, header có độ dài là 64 bytes, chứa thông tin về độ dài của tin nhắn sắp tới
- Khi tin nhắn được nhận, mở header để biết độ dài tin nhắn sắp tới rồi nhận tin nhắn với độ dài đó

Crawl data từ web

- Hỗ trợ việc crawl dữ liệu một lần duy nhất từ web https://xskt.com.vn/ trong vòng đời chương trình khi server được khởi động
- Crawl theo từng vùng (Bắc / Trung / Nam) và lấy 7 ngày gần nhất có xổ số của tỉnh đó vì trong 7 ngày thì tất cả các tỉnh đều được xổ ít nhất một lần
- Sử dụng thư viện scrapy

server.py

- `def main()`
    - Tạo socket từ hàm `create_socket()`
    - Update database bằng việc tạo ra một subprocess để chạy `scrapy crawl`
    - Load data lên các list để phục vụ người dùng

    - Lắng nghe các kết nối sắp tới bằng hàm `start()`
- `def create_socket()`
    - Tạo socket và bind socket với địa chỉ IP
- `def start()`
    - Lắng nghe kết nối tới
    - Khi có kết nối với client mới thì tạo một Thread và target thread đến hàm `handle_client()`
- `def handle_client()`
    - Nhận request từ client bằng phương thức truyền nhận tin đã nói ở trên
    - Xử lí request bằng hàm `handle_request()`
    - Gửi kết quả truy vấn lại cho client
- `def handle_request()`
    - Xử lí yêu cầu của client, truy xuất dữ liệu đã có

client.py

- `def main()`
    - Tạo socket bằng hàm `create_socket()`
    - Nhận truy vấn từ người dùng
    - Gửi yêu cầu tới server bằng phương thức truyền nhận tin đã nói ở trên
- `def create_socket()`
    - Tạo socket và thiết lập kết nối với server
- `def handle_server()`
    - Nhận kết quả truy vấn từ server

Kịch bản chương trình

- Chạy file `main.py`, kết nối với tư cách là server hoặc client
    - Server: Tạo socket → Crawl dữ liệu về database → Load dữ liệu lên để phục vụ → Lắng nghe các kết nối
    - Client sẽ tạo socket → Kết nối đến server

      → Server chấp nhận kết nối của client
- Client gửi truy vấn đến server
- Server xử lí truy vấn → kết quả
- Server gửi kết quả cho client
- Client nhận kết quả
- Client chọn tiếp tục truy vấn hoặc ngắt kết nối với server

Cách chạy chương trình

Lưu ý: Nên chạy *ở môi trường ảo* của python

Giả sử lệnh chạy python trên máy là `python` ở phiên bản 3.x trên Windows. Tại folder `../18127185`

- Install những thư viện cần thiết

  `pip install -r dist/requirements.txt`
- Chuyển đến thư mục chứa src

`cd src`

- Chạy chương trình

  `python main.py`

- Chọn kết nối là client hay server (Chỉ có 1 server duy nhất và server phải tồn tại trước thì client mới có thể kết nối vào được)
- Tại client gửi truy vấn, server sẽ tự động gửi lại kết quả

Giới hạn chương trình

Chương trình chưa hỗ trợ việc tắt hẳn server. Vì ở Windows không thể bắt KeyboardException khi server đang listen được, nên em bỏ qua chức năng này.

Nếu server bị bắt buộc phải ngắt, các thread cũng chết theo vì đã set `daemon = True`