EE 324, Programming Assignment #3

Implementing a web server that can handle large-scale requests

1. Overview

In this assignment, you need to design and implement a web server that runs with multi-threads and uses I/O multiplexing to handle concurrent HTTP requests from clients. Your ultimate goal is to build a high-performance web server that can process about 100,000 (100K) requests per second. The required program is an event-based server with I/O multiplexing to handle HTTP requests. The below figures show example overviews of a workflow. You can design your web server with a single task queue, multiple task queue, or the others. Figure 1 is a single task queue case, and the other one is about numerous task queues (queue per thread).

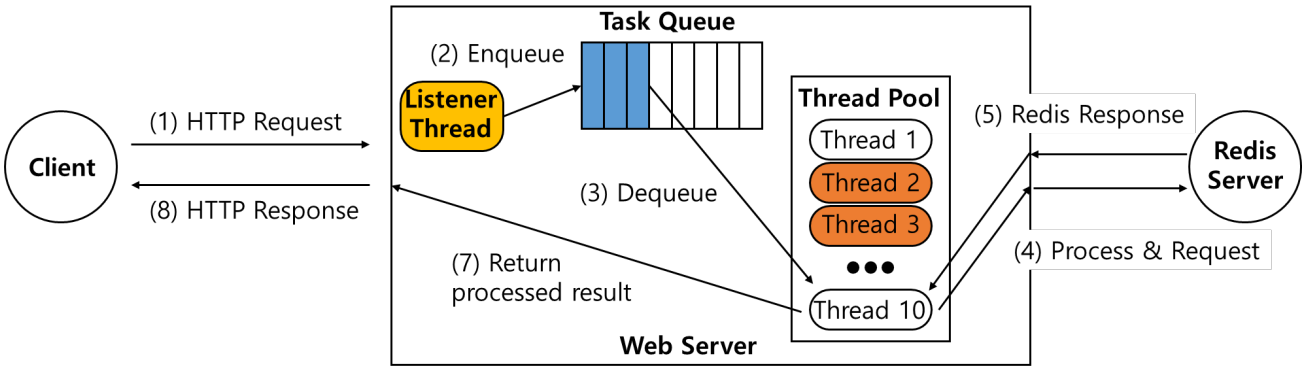

Figure 1. An example overview of a workflow with Single Task Queue

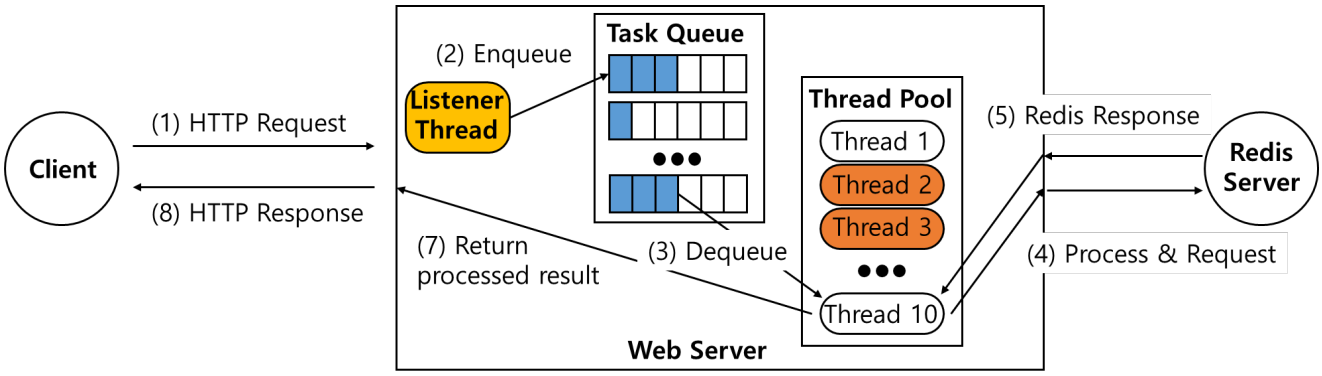

Figure 2. An example overview of a workflow with Multiple Task Queues

2. Web server (event-based server with I/O multiplexing)

- Your server binary (webserver) should get [Port # of your Web Server], [IP address of your Redis Server], [Port # of your Redis Server] from stdin.

  $ **./bin/webserver <webserver port> <Redis Server IP> <Redis Server port>**

- In the webserver, there is a listener thread that receives HTTP requests, which contain Redis requests, just like PA#2, from the client.
- Listener thread may manage multiple socket descriptors. **You have to use the functions from the "libevent" library to pick pending inputs**. (We recommend you to use "Buffered event" functions supported by libevent). It can be applied to other threads for data receiving events.
- When the listener thread received a request, it delivers that request to a specific thread in a thread pool. More clearly, the listener thread makes the connection with the client, and tasks caused by the requests are enqueued to the task queue.
- **NOTE. Design and manage your task queue (Single/Multi queue) for the threads to achieve the best performance.**
- Tasks (depend on your design, it can be the sockets or the received data from sockets) in the Task queue are dequeued one by one, then an idle thread in the thread pool should process that task and returns the result to the client.
- Each thread should send a Redis request packed by the received HTTP request to the Redis server and receive a response. Then, it creates and transmits a new HTTP response to the client.
- **Strict Requirement: Use 10 threads & 40 Redis connections**
- You should manage ten threads by your thread pool design and efficiently manage 40 Redis connection between a web server and a Redis server based on your policy for better performance. (e.g., max # of connections per thread, task allocation algorithm)

**** Reference: We recommend you to read ****

https://www.joinc.co.kr/w/Site/Thread/Advanced/ThreadPool (Korean)
https://docs.oracle.com/cd/E19253-01/816-5137/ggedn/index.html (English)
for the thread pool implementation.

http://wiki.pchero21.com/wiki/Libevent_R6:_Bufferevents:_concepts_and_basics (Korean)
http://www.wangafu.net/~nickm/libevent-book/Ref6_bufferevent.html (English)
for the buffered event handling of libevent library

3. Client: ApacheBench (ab)

** The testing and evaluation scenarios & tools are same as what you did at PA#2. **

The ApacheBench (ab) will be used to evaluate the performance. ab generates 'n' HTTP requests while having 'c' concurrent connections that downloads a same file at the same time. You can specify arguments for the URL to download, number of concurrency (c option), number of total requests (n option). For more detail, please enter the command “ab” to your shell to see its usage (man ab).

- ab with 1 concurrency 1,000 requests

  $ ab n 1000 server_ip:server_port

- ab with 100 concurrency, 10,000 requests

  $ ab c 100 n 10000 server_ip:server_port

After finishing benchmark, ab will print how many transactions were processed per second and whether the content of the file is legitimate. There are requests per second (#/sec) and transfer rate recorded. You will use these values to compare the performance between the different version of servers.

4. Instruction for Submission

- You should upload your code on GitLab. (You don’t need to submit any files on KLMS.) https://gitlab.ee324.kaist.ac.kr/assignments/assignment-3
- TAs will download your project when the due is over.
- **Fork the GitLab repository to your group and clone the forked repository to your local machine before you start.**
  - You need to make a group named after your student id (e.g., 20201234) before forking.
  - Your forked repository should be in the form of (if your ID is 20201234):
  - https://gitlab.ee324.kaist.ac.kr/20201234/assignment-3
  - Make sure you are not directly cloning from the assignment repository.
- In the GitLab, there are five files. You may add more files as you wish. However, the total size of the submission must not exceed 10MB. Please do not add unnecessary files to your commit (e.g., log files).

- README.md
- Makefile
- src/your_source_files

- Make sure that your Makefile correctly compiles your source code. The Makefile must reside in the root of the repository.

- In the README.md:
  - You should put your name, student ID.
  - Write your name and date on the ethics oath.
  - In addition, briefly describe how you designed and implemented your programs.
    - **NO EXPLANATION => We will not grade any part of your project!**
    - Let us know which design/method you used (task queue design, task allocation)
    - Let us know how to manage 10 threads & 40 Redis connections (connection allocation)
  - Describe whatever help (if any) you received from others while doing the assignment, and write the names of any individuals with whom you collaborated, as prescribed by the course Policy web page.

- Write concise comments in the source code to understand your program.

IMPORTANT 1: Please strictly follow the above structure and name policy; if not, TAs will not evaluate your program.

IMPORTANT 2: Please make sure that there is no compile and execution error. TAs will not give you any score in case of the problems.

5. Grading (Total 170 points)

1) Functionality – 80 points

I. Does web server use libevent library to pick pending connections? – 10 points

II. Can the web server process HTTP requests from the client? – 10 points

III. Can the web server send the result of the requests to the client? – 10 points

IV. Can the web server process 100 requests from the client using thread pool? – 50 points

2) Performance – 50 points (Max for the best performance)

I. We will send maximum 100,000 (100K) requests at one second using 100 threads (each thread sends 1,000 requests) as we did with Apache Benchmark at PA #2.

II. Value size of Redis request/respond can be small as 100 bytes or big as 4MB.

III. The grades for this performance part are scored through a relative evaluation manner. Students with better performance will get higher scores than those who do not. (The policy will be finalized after grading, based on statistics)

IV. Metric: “requests per second” of ab result.

3) Design explanation (readme file) – 20 points

I. Explanation about thread pool, task queue, and task allocation – 10 points

II. Explanation about Managing Redis connection, connection allocation – 10 points

III. NO EXPLANATION = NO PROJECT GRADING

4) Extra Credit!!! – 20 points

If you **use buffered event functions** for implementing web server, you will get extra credit.

6. Test Environment

- Language: C or C++
- **NOTE:** We will not consider your compilation and execution problems due to the different OS versions. (defaults: eelab machine, testing machine)

7. Due Date

- **11:59 PM, Nov. 22, 2020 (Sunday) with 10% late penalty per day**
- **Hard deadline: 11:59 PM, Nov. 26. 2020 (Thursday)**

- We will not receive additional submissions after the hard deadline.
- Exceptions: documented medical/personal emergency.
- Please DO NOT e-mail Prof. Han or TAs to submit your assignments.

8. Plagiarism

- You can discuss with your colleagues, but you should turn in your own programs

  ✓ Copy and Paste

  - Will run plagiarism detection on source code.
  - “Copy and paste” codes will get severely penalized
  - If detected, 0 points for all assignments (both providers and consumers)
  - But you will have a chance to defend yourself

9. Questions?

- Please use PIAZZA Q&A board to ask any questions.
- Or you can ask to TAs at official office hour
- Please carefully read specification described at PA #3 document.