Project 1

1. Concurrency Design

Create a thread pool of 10 and a queue of 50. All the 10 thread will be created during initialization. Then all the thread pending on the queue. Whenever the listen thread receive a TCP request, it will put the socket file descriptor into the queue and wake up one working thread.

We create thread pool to save the effort to create thread for every tcp connection.

1. Features

* CGI Implementation  : We implement the CGI functionality in our simple web server. We create a cgi-bin folder under www folder and put a simple file adder.c there. The compiled file adder of this c file is used to add two integer and return the result to client.
* Fighting DOS attacks  : We use select before call recv() function and set a timeout value of 10 seconds. If the select timeout, then we will close the connection.
* Benchmarking your server

We use httperf to test the performance of our web server. We send 1000 connections to our server. Following is the result of the httperf test:

httperf --hog --client=0/1 --server=128.237.209.204 --port=12121 --uri=/ --rate=100 --send-buffer=4096 --recv-buffer=16384 --nu

m-conns=1000 --num-calls=1 --burst-length=2

Maximum connect burst length: 3

Total: connections 1000 requests 1000 replies 1000 test-duration 10.010 s

Connection rate: 99.9 conn/s (10.0 ms/conn, <=24 concurrent connections)

Connection time [ms]: min 6.0 avg 32.8 max 1258.0 median 11.5 stddev 107.5

Connection time [ms]: connect 24.6

Connection length [replies/conn]: 1.000

Request rate: 99.9 req/s (10.0 ms/req)

Request size [B]: 68.0

Reply rate [replies/s]: min 99.8 avg 99.9 max 100.0 stddev 0.1 (2 samples)

Reply time [ms]: response 7.7 transfer 0.6

Reply size [B]: header 64.0 content 879.0 footer 0.0 (total 943.0)

Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0

CPU time [s]: user 5.22 system 4.67 (user 52.2% system 46.7% total 98.8%)

Net I/O: 98.6 KB/s (0.8\*10^6 bps)