Basic Unix Project 01: Networking

Mimi Chen < mchen6@nd.edu >
Andrew Paek < apaek1@nd.edu >

April 20, 2016

Summary

For the Networking project, **thor.py** and **spidey.py** were created. **thor.py** is an HTTP client that hammers a remote HTTP server by making multiple requests. **spidey.py** is a HTTP server that supports directory listings, static files, and CGI scripts.

Work was distributed equally between the members. We collaborated in person and also worked individually by pushing our work to Bitbucket. Working individually was sometimes less efficient since it was harder to communicate online. For example, when we both worked on the same line of code and pushed it, there was a merging error. The files then had to be stashed or renamed to fix the merging error.

Latency

Latency is the measurement of how long it takes to make a request. To measure the latency of the different types of requests for a server using a single connection and forking, the shell script ***latency.sh*** was written.

```
#!/bin/sh
    #latency.sh

            echo "Usage: $0 URL"
    }

    while getopts h argument; do
            case $argument in
                    h) usage
                            exit
                            ;;
                    *) usage
                            exit
                            ;;
            esac
    done

    if [ -z $1 ]; then
            usage
    else
            URL=$1
    fi


    ./thor.py -r 10 -p 10 -v $URL 2>&1> /dev/null | grep "Average" | awk 'BEGIN{FS="|"} {print $2}' | awk
    'BEGIN{FS=":"}{print $2}' | awk 'BEGIN{FS=" "}{sum += $1; n++}END{ if (n>0) print sum/n}'
```

Before executing the shell script, **spidey.py** is ran. The shell script uses **thor.py** to send 10 processes of 10 requests to **spidey.py**. The shell script requires a command line argument for the URL. Since the URL is a command line argument, the type of request can be specified with an URL directing to the path of the directory listing, static file, or CGI script.

Type	URL
Directory listing	http://student00.cse.nd.edu:9114
Static file	http://student00.cse.nd.edu:9114/www/songs/hanging.txt
CGI script	http://student00.cse.nd.edu:9114/www/cgi-bin/env.sh

The output of **latency.sh** is the average elapsed time for each process. To measure the latency of a single connection, **spidey.py** is ran with no flags set. To measure the latency of forking, **spidey.py** is ran with the flag '-f' set. The data is then condensed into a table.

Type	Single Connection (seconds)	Forking Mode (seconds)
Directory listings	0.00152794	0.00487385
Static files	0.0049382	0.0156512
CGI Script	0.00782454	0.0198443

Table 1: Measuring latency of **spidey.py** with **thor.py**

latency.plt consisting the rules in creating a histogram png file was written.

```
#latency.plt
reset

set title "Latency of Single Connection and Forking Mode"
set auto x
set xlabel "Type"
set ylabel "Time (seconds)"
set auto y
set style data histogram
set style histogram cluster gap 2
set style line 1 lt 1 lc rgb "cyan"
set style line 2 lt 1 lc rgb "gray"
set style fill solid border -1
set key left
set boxwidth 0.9
set output "latency.png"
set term png
plot "latency_data.txt" using 2:xtic(1) ti col, '' u 3 ti col
```

Gnuplot was used to convert latency.plt (which called the latency data table) to a png histogram plot. The command used was ”gnuplot -persist latency.plt”.

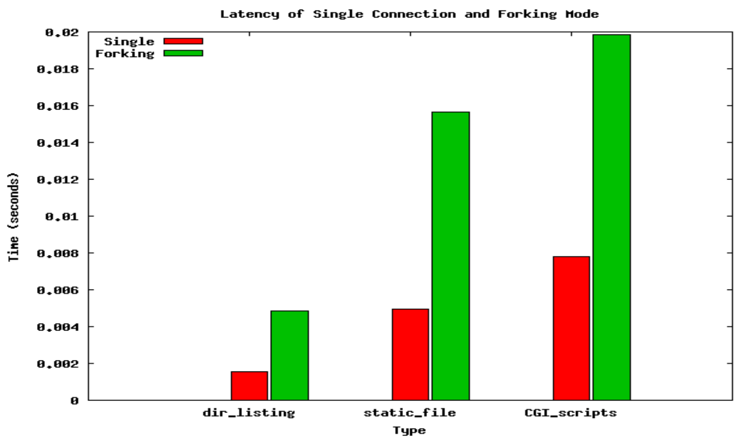

Figure 1: Measuring latency of **spidey.py** with **thor.py**

The latency of handling the requests by forking is much longer than the latency of handling the requests by a single connection. When forking, an additional process (the child process) is created by the parent process. The creation of the child process, the copying the resources, and changing the identity child process, all of which is time consuming.

Throughput

Throughput is how much data can be transferred in a given time. It is usually determined in bits/second. We decided to use thor.py and access files of size 1KB, 1MB, and 1 GB and measure the time taken. To create files of size 1KB, 1MB, and 1GB, we used the dd command to create random bytes. To measure the time taken, we used a shell script to access the 1KB and 1MB files 100 times, using 10 process and 10 requests in thor. Unfortunately, because the 1GB file was too large, we were only able to take a single sample. The results are shown in the following table and summarized in the following figure:

File Size	Single Connection (seconds)	Forking Mode (seconds)
KB	0.003149745	0.0129337947
MB	2.36890281	2.222617403
GB	2219.45399	2215.661636

Table 2: Measuring Throughput of **spidey.py** with **thor.py**

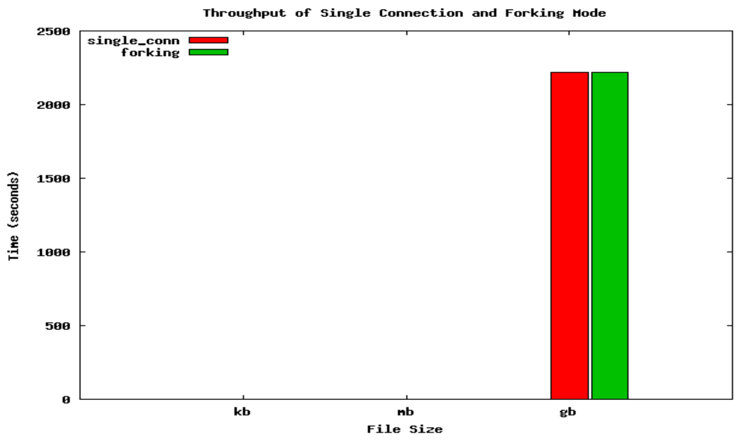

Figure 2: Average Throughput

The following is the shell scipt used to run the 1KB file 100 times:

```
./thor.py -p 10 -r 10 -v $URL/www/throughput/kilobyte 2>&1 > /dev/null
| grep "Average" | awk -F" " '{print &8}' >> throughput_kilo_test.txt'
```

Analysis

The experiments show that for smaller requests, forking took more time. This is probably because forking requires time to create a child process, copy the parent's information, and change what the child is doing. However, for large requests, it seems like forking takes less time, especially if there are a large number of requests. However, overall the data seems to indicate that the difference in time taken is quite small. Forking also provides more benefits like being able to continue interacting with the UI while requests are going through, instead of waiting for each request to finish. Therefore, we believe that the forking model is more advantageous than the single connection model.

Conclusion

Creating the HTTP client and the HTTP server furthered our understanding on how the web works. Networking is just sending and receiving packets of information. The client first sets up a socket, connects to the web server with an IP address and port, and sends a request for information to the web server.The web server also creates a socket, binds the socket to a public host and port, and listens for a request. When the web server receives a request from the client, it accepts the connections and does something with the client socket.

In creating web pages, we also learned basic HTML coding. We included an HTML5 audio element from YouTube to our 404 error page. By adding ”?autoplay=1” to the end of the URL of the YouTube file, we were able to play the audio file automatically after the client receives information back from the server.

The assignment served as a review of material learned previously in class. It required us to use knowledge from the beginning of the semester, such as creating shell scripts to experiment with spidey.py and thor.py, revisiting the concept of latency, writing plt files, creating GNUplots, and writing in LATEX.