

A Distributed Proxy Server
CS8803: Graduate Introduction to Operating Systems
Omkar Bellare GTID#902835686
Vishnu Akhilesh Venkataraman GTID#902832656

Overview

The proxy server that we built checks if the file being requested is of jpg type and if it is, then it sends the complete url of the jpg file to the RPC server which downloads the file, compresses the image and sends back the compressed image to forward it back to the client. We tested our code on the www.ece.gatech.edu page and the screenshots clearly show the original page on the top and the page with the compressed images on the bottom.

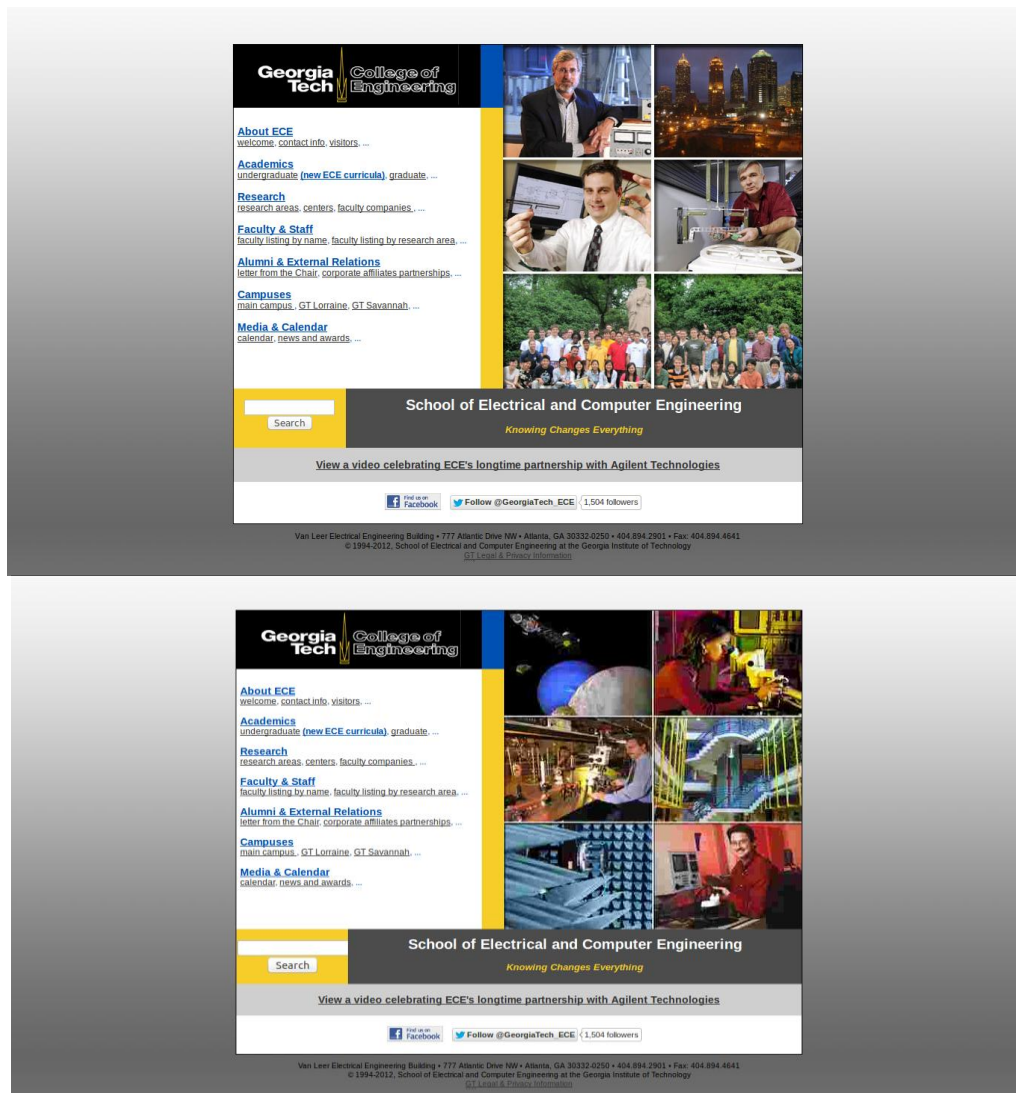


Fig 1: Top image is original page, bottom is page after compression

To test the memory usage of the proxy RPC servers, we queried for an image that was of size 1.7MB whose compressed image size was 68KB using 20 client threads each making 5 requests. Simultaneously we ran top to find the memory usage that our server was using and below are the screenshots for the experiment.

```

omkarb@ubuntu: ~/Documents/GIOS/Project 3/P3_RPC
omkarb@ubuntu: ... x omkarb@ubuntu: ... x omkarb@ubuntu: ... x omkarb@ubuntu: ... x
top - 13:22:06 up 2:47, 5 users, load average: 0.07, 0.04, 0.05
Tasks: 161 total, 1 running, 160 sleeping, 0 stopped, 0 zombie
%Cpu(s): 13.5 us, 3.8 sy, 0.0 ni, 82.7 id, 0.0 wa, 0.0 hi, 0.0 st, 0.0 st
KiB Mem: 3037056 total, 1713212 used, 1323844 free, 95124 buffers
KiB Swap: 3172348 total, 0 used, 3172348 free, 659556 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  6556 omkarb    20   0 57744 25m  904 S   7.6   0.8   0:00.97 compress_server
  2193 omkarb    20   0 1030m 46m  19m S   4.6   1.6   0:41.81 nautilus
  1154 root       20   0 246m 113m 8444 S   2.3   3.8   3:33.01 Xorg
  2798 omkarb    20   0 804m 123m 38m S   1.3   4.2  13:11.48 firefox
  2169 omkarb    20   0 1058m 93m  34m S   1.0   3.2   2:16.32 compiz
  2597 omkarb    20   0 529m 22m  12m S   1.0   0.8   0:31.27 gnome-terminal
  2319 omkarb    20   0 537m 29m  11m S   0.3   1.0   0:11.67 unity-panel-ser
  2386 omkarb    20   0 511m 15m  11m S   0.3   0.5   0:00.28 goa-daemon
  3061 omkarb    20   0 523m 57m  15m S   0.3   1.9   1:59.16 gedit
  3546 omkarb    20   0 1039m 198m 28m S   0.3   6.7   0:42.71 opera
  6303 omkarb    20   0 24784 1620 1164 R   0.3   0.1   0:03.98 top
    1 root      20   0 24600 2536 1376 S   0.0   0.1   0:01.15 init
    2 root      20   0 0 0 0 S   0.0   0.0   0:00.00 kthreadd
    3 root      20   0 0 0 0 S   0.0   0.0   0:00.76 ksoftirqd/0
    6 root      rt   0 0 0 0 0 S   0.0   0.0   0:00.00 migration/0
    7 root      rt   0 0 0 0 0 S   0.0   0.0   0:00.39 watchdog/0
    8 root      0 -20 0 0 0 S   0.0   0.0   0:00.00 cpuset

strcat(requestLine, filename);
strcat(requestLine, " HTTP/1.1\r\n");
strcat(requestLine, "Host: fc03.deviantart.net\r\n");
strcat(requestLine, "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:17.0) Gecko/20100101 Firefox/17.0\r\n");
strcat(requestLine, "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n");
strcat(requestLine, "Accept-Language: en-US,en;q=0.5\r\n");
strcat(requestLine, "Accept-Encoding: gzip, deflate\r\n");
strcat(requestLine, "Proxy-Connection: keep-alive\r\n\r\n");

buf = (char*)malloc(BUFFER_LENGTH);

//Sending the request

```

```

omkarb@ubuntu: ~/Documents/GIOS/Project 3/P3_RPC
omkarb@ubuntu: ... x omkarb@ubuntu: ... x omkarb@ubuntu: ... x omkarb@ubuntu: ... x
top - 13:22:31 up 2:47, 5 users, load average: 0.12, 0.05, 0.05
Tasks: 161 total, 1 running, 160 sleeping, 0 stopped, 0 zombie
%Cpu(s): 16.9 us, 4.6 sy, 0.0 ni, 77.8 id, 0.7 wa, 0.0 hi, 0.0 st, 0.0 st
KiB Mem: 3037056 total, 1735216 used, 1301840 free, 95212 buffers
KiB Swap: 3172348 total, 0 used, 3172348 free, 660376 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  6556 omkarb    20   0 122m 42m  904 S  13.2   1.4   0:03.19 compress_server
  2193 omkarb    20   0 1030m 46m  19m S   3.3   1.6   0:42.57 nautilus
  1154 root       20   0 248m 115m 8444 S   1.6   3.9   3:33.84 Xorg
  2798 omkarb    20   0 804m 123m 38m S   1.3   4.2  13:11.77 firefox
  2597 omkarb    20   0 529m 22m  12m S   1.0   0.8   0:31.48 gnome-terminal
  2169 omkarb    20   0 1058m 93m  35m S   0.7   3.2   2:16.95 compiz
  3546 omkarb    20   0 1039m 198m 28m S   0.7   6.7   0:42.78 opera
  6303 omkarb    20   0 24784 1620 1164 R   0.7   0.1   0:04.06 top
  1528 nobody   20   0 33072 1552 1304 S   0.3   0.1   0:00.07 dnsmasq
  2319 omkarb    20   0 537m 29m  11m S   0.3   1.0   0:11.72 unity-panel-ser
  2321 omkarb    20   0 639m 9224 4368 S   0.3   0.3   0:07.85 hud-service
  2386 omkarb    20   0 511m 15m  11m S   0.3   0.5   0:00.29 goa-daemon
  3061 omkarb    20   0 523m 57m  15m S   0.3   1.9   1:59.56 gedit
  4367 root      20   0 0 0 0 S   0.3   0.0   0:07.73 kworker/0:0
    1 root      20   0 24600 2536 1376 S   0.0   0.1   0:01.15 init
    2 root      20   0 0 0 0 S   0.0   0.0   0:00.00 kthreadd
    3 root      20   0 0 0 0 S   0.0   0.0   0:00.76 ksoftirqd/0

strcat(requestLine, filename);
strcat(requestLine, " HTTP/1.1\r\n");
strcat(requestLine, "Host: fc03.deviantart.net\r\n");
strcat(requestLine, "User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:17.0) Gecko/20100101 Firefox/17.0\r\n");
strcat(requestLine, "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n");
strcat(requestLine, "Accept-Language: en-US,en;q=0.5\r\n");
strcat(requestLine, "Accept-Encoding: gzip, deflate\r\n");
strcat(requestLine, "Proxy-Connection: keep-alive\r\n\r\n");

buf = (char*)malloc(BUFFER_LENGTH);

//Sending the request

```

Fig 2: Top image was taken before bottom image (observe compress_server)

From the above screenshots, we clearly observe that the `compress_server` is using a large amount of memory which it does not free as time progresses. The memory will be freed automatically but only after some time. The previous experiment took the memory readings when the files were still being requested for. In the next experiment we tried requesting for just one large file, keep the `compress_server` reading and check for memory usage after probably 4 seconds after the file was sent to the client. This time we observed that the memory goes down soon after the processing is done, meaning Sun RPC probably has a memory cleanup thread which runs when the server is not doing anything else. Screenshots are again shown below:



Fig 3: Top image shown while RPC server was working on compression, bottom shows 4 seconds later

Experiment 2: Testing the impact of varying the image file size

Setup: 20 Client Threads with each making 5 requests.

Image Size (KB)	Number of bytes after compression (B)	Time (sec)	Throughput (Mb/s)	Requests per sec
10	360000	27.292	0.013191	3.664
20	921500	77.799	0.011844	1.285
64	1087400	116.667	0.009320	0.857
108	1020500	129.567	0.007876	0.772
151	786752	150.526	0.005226	0.664

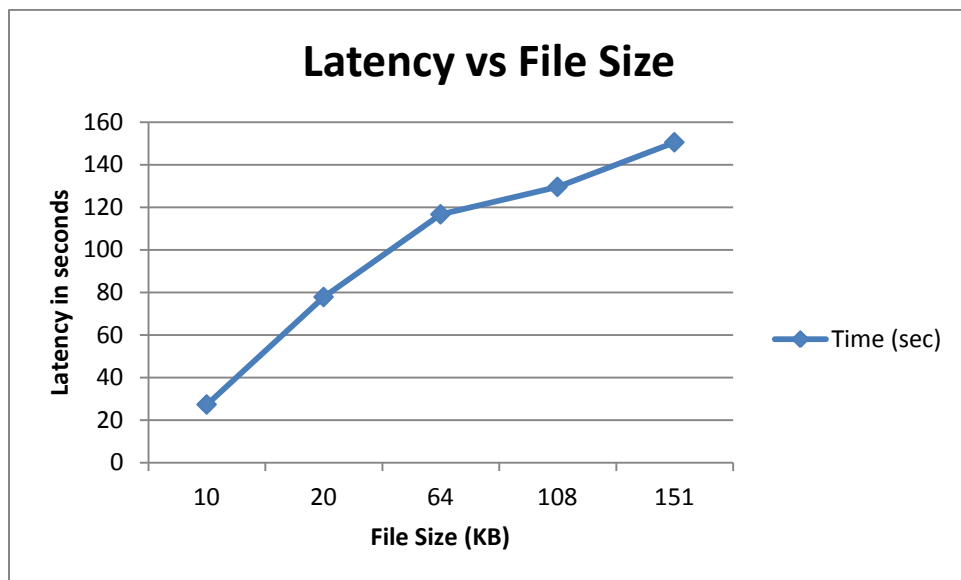


Fig 4: Latency vs File size

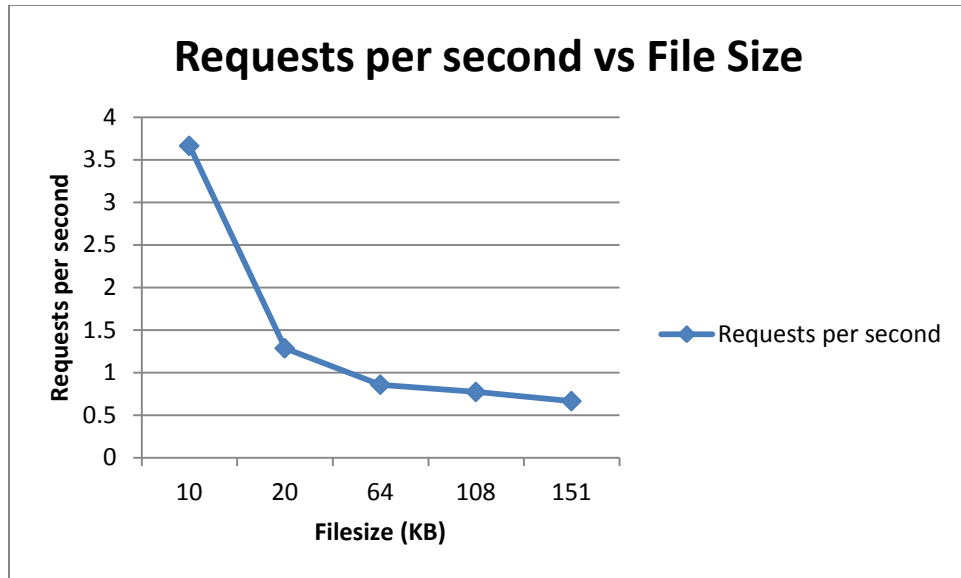


Fig 5: Requests per second vs Filesize

Conclusion

As we can see there is almost an exponential drop in the number of requests that can be handled by the RPC server per second. This is because the server is currently single threaded and processes requests sequentially. We can also see an almost linear trend in the throughput of the proxy. Also we think if we increase the number of RPC servers the time to process will be lower, and should ideally be half but will be a little more due to the RPC overhead. Thus the use of RPC for distributing the load on proxy shows that the bottleneck is not the network but the CPU.