# Performance Evaluation of a Web Server

Hamza Mughal
Department of Computer Science
George Mason University
Fairfax, VA USA
hmughal2@masonlive.gmu.edu

## ABSTRACT

Web server deployments have exponentially grown in the 21st century with the introduction of cloud server providers such as Amazon. Web servers have been increasingly becoming more and more important to the profitability of companies. To put it into perspective Amazon's AWS currently hosts 4.7% of all websites (currently there are approximately 1.8 billion hostnames and 178 million websites).

As seen from figure 1, the total number of websites has drastically increased (and as a result so has the number of web servers hosting these websites!).

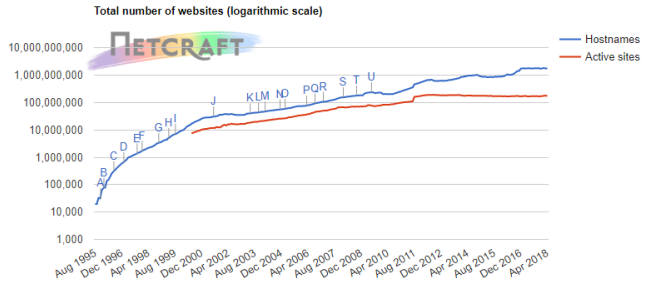



Figure 1. Netcraft February 2018 Web Server Survey

With web servers being readily available for low cost amongst multiple cloud service providers such as Amazon, Cloudhost, Vultr etc., the necessity of a performant web server is also needed to keep up with an increased customer population to a webpage.

This paper analyzes a web servers' ability to keep up with increased concurrent demand as files are requested and subsequently served to multiple users at the same time.

## 1 Performance Problem Being Studied

### 1.1 Vultr

Vultr is a cloud service provider which was launched in 2014. Vultr provides a variety of web servers of different capacity and price ranges. A few of the price ranges can be seen below:

1. 10 GB SSD, 1 CPU 2GHz, 512 MB Memory, .50 TB Bandwidth @ $2.50 per month
2. 25 GB SSD, 1 CPU. 1024 MB Memory, 1 TB Bandwidth @ $5.00 per month
3. 320 GB SSD, 6 CPU, 16 GB Memory, 5TB Bandwidth @ $80.00 per month
4. 1600 GB SSD, 24 CPU, 96 GB Memory, 15 TB Bandwidth @ $640.00 per month

### 1.2 www.hamzamughal.com

My personal webpage was deployed in early 2018 on Vultr with the server specs 20 GB SSD, 512 MB Memory, 500 GB Bandwidth. Additionally, it has been developed node.js and express. The sole purpose of having my own webpage was to have a virtual presence. Thousands of individuals have personal web pages online describing who they are, what they do, how to contact them etc.

With cloud servers being extremely cheap to deploy, it made sense for me to join he bandwagon back in 2018 as it would allow others to find me via the internet easily. On average my webpage averages about 100 visitors per month. Needless to say some of the users attempt to crash my server – some of their attempts succeed however the server always automatically restarts itself.

Currently there are three routes set up which serve static HTML pages (these pages are KBs in size):

1. a contact page
2. a home page
3. a GitHub/games page

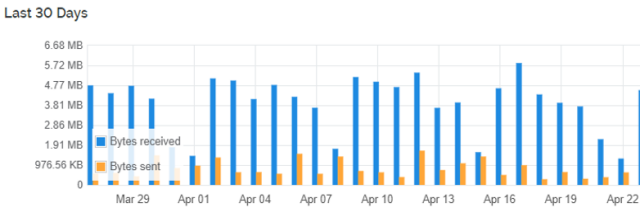



Figure 2. hamzamughal.com bytes sent/received

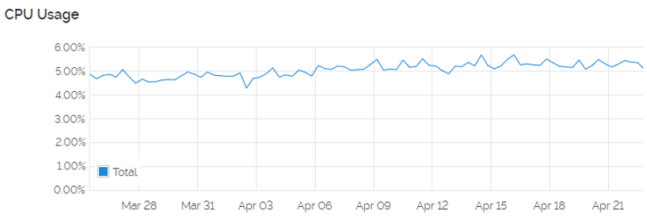



Figure 3 – hamzamughal.com CPU usage

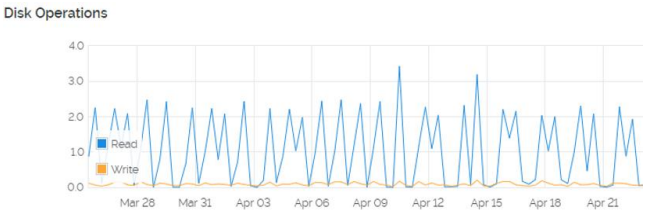



Figure 4 – hamzamughal.com disk IOs

As can be seen from the three figures above, there is not much happening in terms of CPU/disk utilization as serving HTML files which are KBs large are not as taxing as in the past on hardware. Because these static files are not physically taxing on the hardware, thousands of concurrent users could very easily be supported on the website which no dropped requests.

Instead of serving static HTML files to users, three new routes have been implemented to serve files of different types and sizes:

1. /displayFile
2. /downloadFile
3. /downloadZip

The first route is used to serve a 250 MB PDF file for browser viewing. The second route is used to serve a 30 MB and 250 MB PDF file for downloading. The third route is used serve a 66 MB and 282 MB ZIP file. With these routes serving files that are MBs large, the performance of the web server can be measured to determine if it can handle multiple concurrent requests at a decent enough response time.

# 2 Data Collection Procedures

## 2.1 Apache Bench

Apache Bench, or better known as ab, is a load testing tool used for benchmarking a Apache Hypertext Transfer Protocol (HTTP) server [2]. Apache Bench is generic enough to be used with any server as well. It is designed to give an impression of how a web server performs as it has the ability to display how many requests per second a web server is capable of serving.

A sample run of Apache Bench can be seen in figure 5 below.

```
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking hamzamughal.com (be patient).....done


Server Software:        Apache/2.4.29
Server Hostname:        hamzamughal.com
Server Port:            80

Document Path:          /
Document Length:        4072 bytes

Concurrency Level:      1
Time taken for tests:   0.052 seconds
Complete requests:      1
Failed requests:        0
Total transferred:      4401 bytes
HTML transferred:       4072 bytes
Requests per second:    19.28 [#/sec] (mean)
Time per request:       51.862 [ms] (mean)
Time per request:       51.862 [ms] (mean, across all concurrent requests)
Transfer rate:          82.87 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       16   16   0.0     16      16
Processing:    36   36   0.0     36      36
Waiting:       35   35   0.0     35      35
Total:         52   52   0.0     52      52
```

Figure 5 – Apache Bench output

The sample run was output running the command above:

ab -n 1 -c 1 http://hamzamughal.com/

-n indicates the number of requests to perform and -c indicates the concurrency level and the number of requests to perform at a time.

Time taken for test indicates the time taken from the moment the first socket connection is created to the moment the last response is received. Requests per seconds is the number of requests per second. This value is the result of dividing the number of requests by the total time taken. The higher this number is the better.

## 2.2 Apache JMeter

Apache JMeter is another load testing tool for analyzing and measuring performance specifically for web servers. A sample run of Apache JMeter can be seen in Figure 6 below.

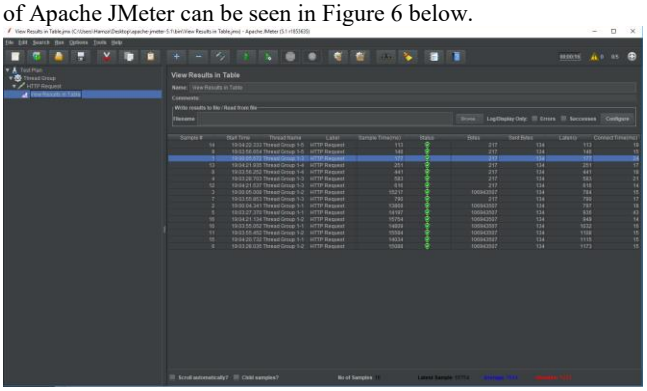


Figure 6 – Apache JMeter

For the purpose of analyzing the web server, Apache Bench is mainly used however Apache JMeter is used to validate the output of Apache Bench.

## 2.3 dstat

Dstat is a replacement for more popularly known tools such as iostat, vmstat etc. The output of dstat is designed for real time consumption where the output can additionally be piped to a CSV file for human consumption. A sample output of dstat can be seen in figure 7.

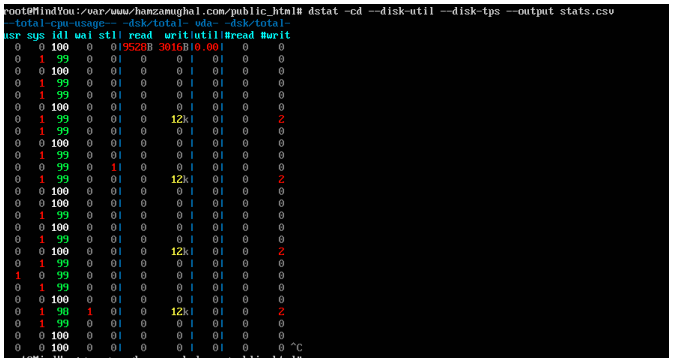



Figure 7

The data above was produced through the command:

dstat -cd -disk—util -disk-tps -output stats.csv

where it outputs the CPU usage, bytes read/write, disk usage, and disk read/writes.

## 2.4 Second Vultr Server

Another Vultr server of similar specs to the web server was deployed to run the Apache Bench tool. This is because is made sense to the the Vultr bandwidth to send requests as it would result in non-skewed results. If I had chosen to run the command locally to send requests, the results would have been greatly skewed as the download speeds would be bottlenecked by my own bandwidth.

# 3 Results Gathered

A total of 1204 requests went sent to the web server downloading files of various concurrency levels ranging from 1 to 100. I now discuss the results for each of the different files downloaded.

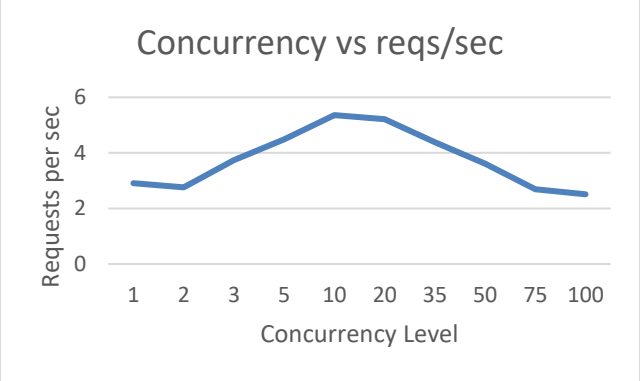
## 3.1 30 MB PDF

Table 1 displays the various data gathered from sending 1-100 concurrent users each executing one request for the PDF file.

| # conc. | Req/ sec | Time/req (sec) | Avg IO | Bytes Read (avg) | CPU util. usr | CPU util sys | Disk Util |
|---|---|---|---|---|---|---|---|
| 1 | 2.91 | .343228 | 2 | 49153 | 13.86 | 21.78 | 0 |
| 2 | 2.76 | .72563 | 0 | 49103 | 14.04 | 63.45 | 0 |
| 3 | 3.74 | .801553 | 2 | 192512 | 15.94 | 24.90 | .4 |
| 5 | 4.49 | 1.1135 | 3 | 201653 | 21.50 | 34.50 | 0 |
| 10 | 5.36 | 1.86688 | 3 | 241364 | 39.36 | 53.74 | 0 |
| 20 | 5.21 | 3.84080 | 2 | 2515313 | 36.50 | 48.09 | 0 |
| 35 | 4.38 | 7.99682 | 1.6 | 2580480 | 41.91 | 50.21 | .8 |
| 50 | 3.61 | 13.8369 | 0 | 320152 | 42.78 | 39.43 | 0 |
| 75 | 2.69 | 27.8495 | 1 | 696320 | 42.84 | 32.23 | 1.2 |
| 100 | 2.51 | 36.9929 | .22 | 816311 | 48.09 | 30.86 | 3 |

Table 1

A visual depiction of the concurrency vs reqs/sec reveals an important insight:




The reqs/sec gradually increased hitting a global maximum at 10 concurrent users however it then gradually decreased as the concurrency level increased.

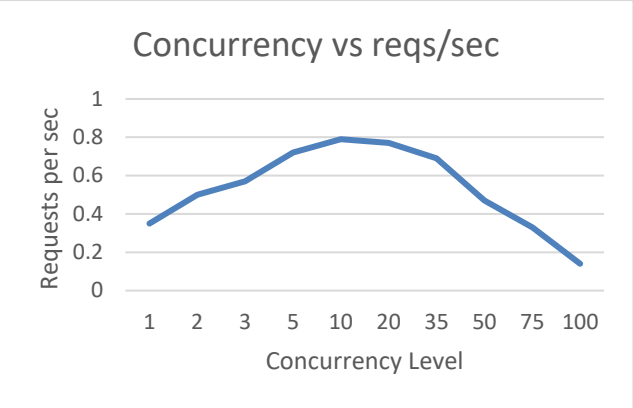
## 3.2 250 MB PDF

Table 2 displays the various data gathered from sending 1-100 concurrent users each executing one request for the PDF file

| # conc. | Req/ sec | Time/req (sec) | Avg IO | Bytes Read (avg) | CPU util. usr | CPU util sys | Disk Util |
|---|---|---|---|---|---|---|---|
| 1 | .35 | 2.89196 | 406.5 | 68597077 | 21.93 | 49.66 | 25.87 |
| 2 | .5 | 3.97006 | 353 | 43434379 | 23.92 | 51.92 | 6.4 |
| 3 | .57 | 5.25446 | 342 | 43357525 | 26.97 | 58.12 | 2.3 |
| 5 | .72 | 6.92093 | 299.42 | 37390629 | 31.11 | 59.93 | 1.14 |
| 10 | .79 | 12.6550 | 130.87 | 16330240 | 32.94 | 54.23 | .2 |
| 20 | .77 | 26.0549 | 70.93 | 8816401 | 38.38 | 56.33 | .42 |
| 35 | .69 | 21.0771 | 43.23 | 5067461 | 42.49 | 52.16 | .52 |
| 50 | .47 | 106.945 | 33.58 | 3746253 | 45.59 | 41.35 | .8 |
| 75 | .33 | 227.118 | 9.40 | 1145479 | 47.29 | 31.73 | .05 |
| 100 | .14 | 310.542 | 6.44 | 79942.5 | 49.40 | 30.39 | .02 |

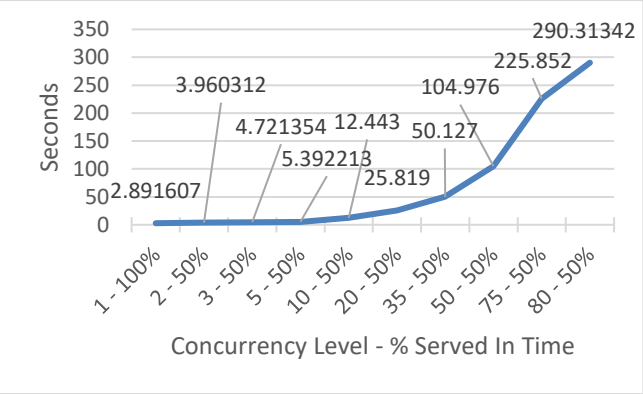
Table 2

A visual depiction, once more, of concurrency vs req/sec reveals the same observation made earlier:




The number of req/sec gradually hits a peak at 10 concurrent users and then dies off as concurrency increases. This is telling as it explains that the system hits a bottleneck at 10 users however let us continue with the results gathered.

Another visualization is concurrency vs. request served in a time period:




As the concurrency level increased, at least 50% of the current requests being processed took at most the time shown above. For example, for a concurrency level of 50 at least 50% of the requests took 104 seconds or more to complete.

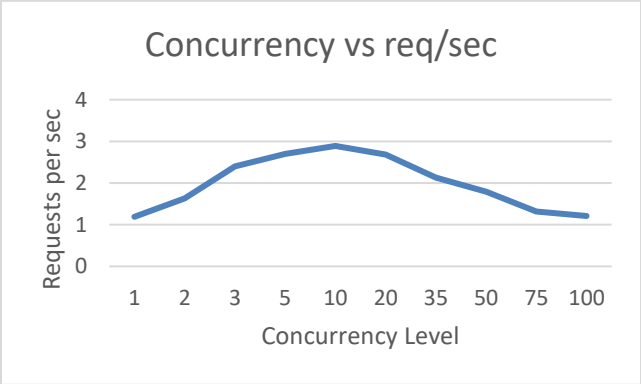
## 3.3 66 MB ZIP

Table 3 displays the various data gathered from sending 1-100 concurrent users each executing one request for the ZIP file.

| # conc. | Req/ sec | Time/req (sec) | Avg IO | Bytes Read (avg) | CPU util. usr | CPU util sys | Disk Util |
|---|---|---|---|---|---|---|---|
| 1 | 1.19 | .839253 | 3.5 | 53248 | 17.5 | 25.5 | 0 |
| 2 | 1.63 | 1.22795 | 3.5 | 61440 | 31.38 | 30.34 | 0 |
| 3 | 2.4 | 1.24855 | 1.5 | 10240 | 22.51 | 32.53 | 0 |
| 5 | 2.7 | 1.85093 | 0 | 43230 | 39.50 | 53.5 | 0 |
| 10 | 2.89 | 3.46432 | 0 | 64123 | 31.90 | 53.19 | 0 |
| 20 | 2.68 | 7.46732 | 2.25 | 49664 | 37.28 | 51.42 | 0 |
| 35 | 2.13 | 16.4598 | 12.82 | 1559130 | 40.09 | 49.09 | .64 |
| 50 | 1.79 | 27.9566 | 8.89 | 866742.9 | 45.52 | 39.68 | .142 |
| 75 | 1.32 | 56.8189 | 2.403 | 112891.5 | 44.47 | 31.94 | .02 |
| 100 | 1.21 | 82.5022 | 7.48 | 708608 | 51.40 | 30.82 | .14 |

Table 3

A visual depiction, once more, of concurrency vs req/sec reveals the same observation made earlier:




Once more, the fact that the server bottlenecks in requests per sec once 10 concurrent users is hit is further corroborated.

## 3.4 282 MB ZIP

Table 4 displays the various data gathered from sending 1-100 concurrent users each executing one request for the ZIP file.

| # conc. | Req/ sec | Time/req (sec) | Avg IO | Bytes Read (avg) | CPU util. usr | CPU util sys | Disk Util |
|---|---|---|---|---|---|---|---|
| 1 | .28 | 3.60830 | 431.2 | 56595251 | 32.25 | 39.27 | 11.36 |
| 2 | .35 | 5.65679 | 314.2 | 40460873 | 32.14 | 47.79 | 10.22 |
| 3 | .58 | 5.13877 | 335.5 | 42280960 | 29.56 | 56.13 | 2.87 |
| 5 | .63 | 7.89253 | 296 | 35822592 | 34.99 | 63.24 | 1.3 |
| 10 | .76 | 13.1841 | 156.1 | 19653778 | 32.27 | 61.78 | .65 |
| 20 | .61 | 33.0078 | 71.42 | 8819805 | 38.02 | 56.52 | .36 |
| 35 | .52 | 67.1434 | 48.41 | 5728898 | 41.72 | 50.46 | .38 |
| 50 | .43 | 115.869 | 38.45 | 4619264 | 48.84 | 41.38 | .4 |
| 75 | .31 | 241.237 | 11.62 | 1685266 | 49.04 | 33.07 | .12 |
| 100 | .17 | 314.731 | 6.41 | 981543 | 54.60 | 29.31 | .11 |

Table 4

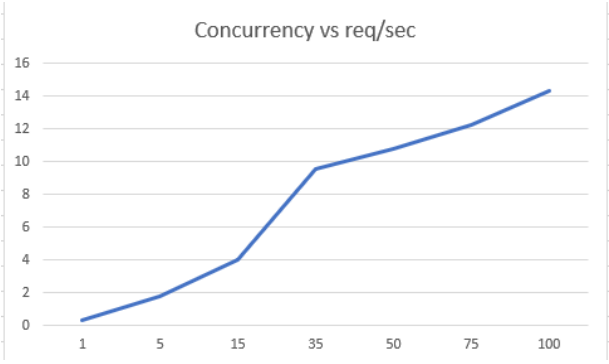
## 3.4 Streaming a 250 MB PDF

The focus of this project was on the download of files, however I choose to include this as an extra piece of data collected as I was interested how streaming a large PDF file affected utilization etc (none of the data in Table 5 is used to solve a model, it is purely for informative value).

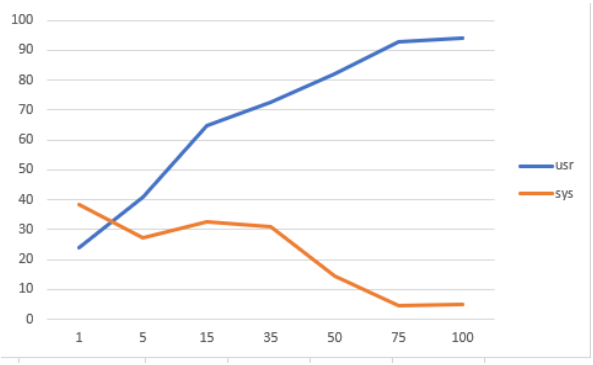
| # conc. | Req/sec | Time/req (sec) | CPU util. usr | CPU util. sys | Disk util. |
|---|---|---|---|---|---|
| 1 | .33 | 2.989 | 24 | 38.39 | 224.4 |
| 5 | 1.76 | 2.834 | 40.77 | 27.40 | 44 |
| 15 | 3.98 | 3.770 | 64.78 | 32.64 | 37.9 |
| 35 | 9.54 | 3.670 | 77.65 | 31.00 | 40.8 |
| 50 | 10.78 | 4.813 | 82.3 | 14.6 | 41 |
| 75 | 12.25 | 6.124 | 93 | 4.6 | 45.4 |
| 100 | 14.31 | 6.986 | 94.2 | 5 | 50.6 |

Table 5

There are some differences that can be spotted when the PDF file is streamed to the browser vs. when it is downloaded:




The first difference is that instead of the req/sec dropping at a concurrency level of 10, it gradually increased!




The second difference is that CPU usage spiked and continued to increase at a steady rate rather than slowly increasing to a maximum of around 50%. The same can be said for disk utilization – rather than hovering in the single digits it increased into the double digits (~50%).


Compared to the average disk IOs for downloads, streaming the 250 MB PDF hit a max IO of ~1500. This intuitively makes sense considering the browser has to continuously fetch data from the server as a user scrolls through a document.

## 3 Solving a Model

Now that data has been gathered, a queueing network can be solved.

| CPU Util PDF | CPU Util ZIP | Disk Util PDF | Disk Util ZIP |
|---|---|---|---|
| 38.99 | 40.80 | 2.1575 | 1.4118 |

A total of 1204 requests were logged during a 30-minute interval, thus we calculate the system throughout using the Service Demand law [1]

$$D[i] = U[i]/X0$$

Where X0 is given to be .6689 (1204/ (30*60)).

The following service demands are found:

| | PDF | ZIP |
|---|---|---|
| CPU | .583052311 | .610083972 |
| Disk (SSD) | .032254983 | .021106645 |

Solving a closed QN with the above service demands results in the following data for concurrency levels 0-200:

| # Conc. | Throughput PDF | Throughput ZIP | Resp. Time PDF | Resp. Time ZIP |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 20 | .86 | .82 | 23.329 | 24.397 |
| 40 | .86 | .82 | 46.650 | 48.800 |
| 60 | .86 | .82 | 69.972 | 73.203 |
| 80 | .86 | .82 | 93.294 | 97.606 |
| 100 | .86 | .82 | 116.616 | 122.010 |
| 120 | .86 | .82 | 139.939 | 146.416 |
| 140 | .86 | .82 | 163.261 | 170.817 |
| 160 | .86 | .82 | 185.583 | 195.220 |
| 180 | .86 | .82 | 209.905 | 219.623 |
| 200 | .86 | .82 | 233.227 | 244.027 |

Solving a closed QN for 2 CPUs results in the following service demands:

| | PDF | ZIP |
|---|---|---|
| CPU | .29153 | .30504 |
| Delay Resource | .29153 | .30504 |
| Disk (SSD) | .032254983 | .021106645 |

The resulting data from solving the QN with 2 CPUs:

| # Conc. | Throughput PDF | Throughput ZIP | Resp. Time PDF | Resp. Time ZIP |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 20 | 1.71 | 1.64 | 11.668 | 12.195 |
| 40 | 1.71 | 1.64 | 23.329 | 24.396 |
| 60 | 1.71 | 1.64 | 34.990 | 36.598 |
| 80 | 1.71 | 1.64 | 46.651 | 48.799 |
| 100 | 1.71 | 1.64 | 58.312 | 61.001 |
| 120 | 1.71 | 1.64 | 69.973 | 73.203 |
| 140 | 1.71 | 1.64 | 81.634 | 85.404 |
| 160 | 1.71 | 1.64 | 93.295 | 97.606 |
| 180 | 1.72 | 1.64 | 104.956 | 109.80 |
| 200 | 1.72 | 1.64 | 116.617 | 122.00 |

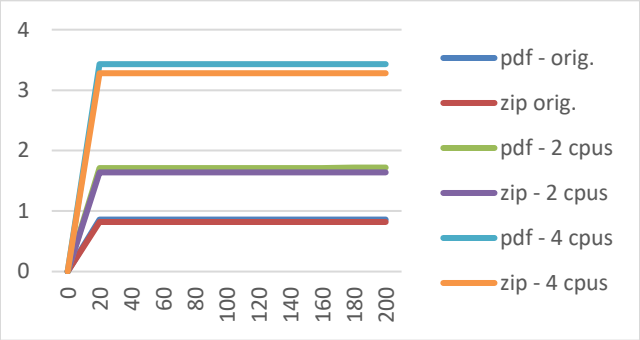
Solving a closed QN for 4 CPUs results in the following service demands:

| | PDF | ZIP |
|---|---|---|
| CPU | .14576 | .15252 |
| Delay Resource | .43729 | .45756 |
| Disk (SSD) | .032254983 | .021106645 |

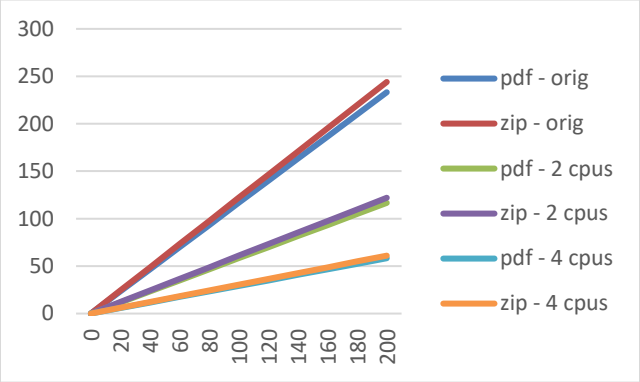
The resulting data from solving the QN with 4 CPUs:

| # Conc. | Throughput PDF | Throughput ZIP | Resp. Time PDF | Resp. Time ZIP |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 20 | 3.43 | 3.28 | 5.8388 | 6.093 |
| 40 | 3.43 | 3.28 | 11.6688 | 12.194 |
| 60 | 3.43 | 3.28 | 17.4992 | 18.295 |
| 80 | 3.43 | 3.28 | 23.3296 | 24.395 |
| 100 | 3.43 | 3.28 | 29.1601 | 30.496 |
| 120 | 3.43 | 3.28 | 34.9906 | 36.597 |
| 140 | 3.43 | 3.28 | 40.8211 | 42.698 |
| 160 | 3.43 | 3.28 | 46.6516 | 48.799 |
| 180 | 3.43 | 3.28 | 52.4821 | 54.899 |
| 200 | 3.43 | 3.28 | 58.3126 | 61.000 |

A visual depiction of all the throughputs graphed among the different CPU configurations:



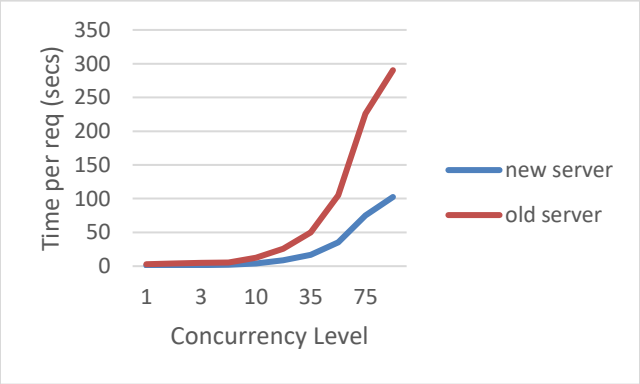

A visual depiction of all the response times graphed among the different CPU configurations:




From the above image, the response time (when using 4 CPUs compared to the original 1 CPU) drastically decreases to reasonable response times.

## 3.1 A test run with a 4 CPU Vultr server

As noted above, having 4 CPUs drastically decreases the response time as the concurrency level increases. I decided to deploy a 4 CPU Vultr server for a simple test to see exactly how much of a reduction was had:



## 4 Conclusion

In conclusion, this whole process was quite intriguing as I never thought I would really be evaluating my own web server this thoroughly. A few key items were noticeable throughout the process.

The first item is that with the original web server specifications, the req/secs hit a maximum value at10 concurrent users however there was a gradual decline as the concurrency level increased. This indicates a bottleneck starting to occur whenever 10 or more concurrent users download a file.

The second item of nice is that as concurrency increase, so did the CPU utilization however disk utilization did not really increase.

The service demands, for the original web server, for PDF and ZIP was .583052311 and .610083972 respectively. The service demand for ZIP files was higher than PDF files which indicates a bottle neck when serving files for downloads (although this could be down to the fact that the ZIP files used were slightly larger in file size thus causing the difference in service demands). The solution to lower service demands, and consequently lower response times, is to add more CPU cores.

Adding more CPU cores has a drastic impact on the response time on downloads as more cores results in faster downloads for more concurrent users. For example, 1 CPU had a PDF response time of 233 secs, 2 CPUs had a PDF response time of 116 secs, and 4 CPUs had a PDF response time of 58 seconds. This decrease in response time is definitely noticeable.

Despite analyzing the performance of my web server serving large files and learning that it is not the best hardware wise, the current specifications serve the current server usage well: to serve static HTML pages for people to know who I am/be able to contact me.

If, in the future, I were to start serving large files I would be able to certainly know that I would need to upgrade my server specifications to match the increased demand. This web server analysis has been an interesting experience into how actual performance evaluation is done.

## REFERENCES


[1] D. A. Menascé, V. Almeida, and Larry W. Dowdy, Performance by Design: Computer Capacity Planning by Example, Prentice Hall, 2004.

[2] Apache Bench (ab) - https://httpd.apache.org/docs/2.4/programs/ab.html