Nabil's Books

A load testing experiment

Nabil Darwich
Department of Computer Science
George Mason University
Fairfax VA USA
ndarwich@gmu.edu
http://www.nabild.com

ABSTRACT

Nabil's Books (http://www.nabild.com/books), a world-renowned bookstore, wants to expand its book distribution monopoly The bookstore is currently experiencing contentions from consumers due to latencies on its service to them. The owner of the bookstore wants to investigate the cause of these delays and find out how management should act.

1 Background

1.1 – Introduction, Nabil's Books

Nabil's Books is a bookstore that sells books of all types of media. The bookstore serves Text Books, Image Books, Audio Books, Video Books, and even, Miscellaneous Books. The bookstore has books of all sizes, with its smallest book in supply containing one million pixels (**2.86 MB**), and its largest book holding a staggering 111,960 KB (**109 MB**) of rare video footage. The bookstore is open to the world wide web, with clients requesting books from the Taj Mahal in India to the Statue of Liberty in America. Of course, as Nabil's bookstore's motto is to "Spread knowledge, wherever you are," it plans to always respond to incoming requests in a timely manner.

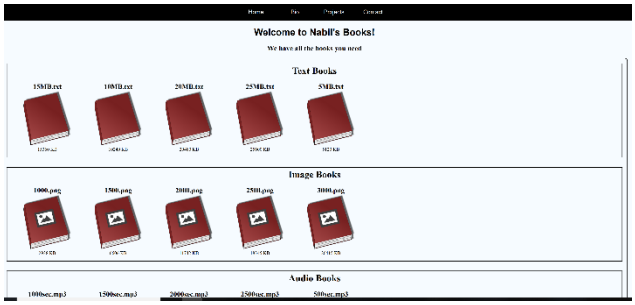

Figure 1: Welcome screen of Nabil's Books

The following is a description of different types of books distributed by Nabil's Books:

1.1.1 – Nabil's Text Books

The first type of media Nabil's Books shares with the world is plain text. Nabil's Text Books may be small in size, but they contain a wide array of knowledge. Nabil's Books gives 5 free Text Books, all in Latin, to anyone who requests them, These books range in sizes from 5,242,880 characters (**5 MB**) to 26,214,400 characters (**25 MB**)

1.1.2 – Nabil's Image Books

Pictures speak a thousand words, and Nabil's Books shares Image Books as well. These Image Books are not just for interpretation, but they are also for learning and understanding, especially for the visual learners. Nabil's Books is kind enough to offer 5 free Image Books to its clients. Image Books that are offered range in size from 1,000px by 1,000px (**2.86 MB**) to 3,000px by 3,000px (**25.MB**).

1.1.3 – Nabil's Audio Books

A lot of time is wasted by us. Whether it's to commute from one place to another, or to get a checkup at the dentist. To help pass time, Nabil's Books also provides Audio Books! There are 5 Audio Books available, and they range in length from 100 seconds (**1.52 MB**) to 1,000 seconds (**15.2 MB**). Why waste time impatiently waiting, when a whole world of knowledge is available at your fingertips.

1.1.4 – Nabil's Video Books

If a picture is worth a thousand words, videos are worth billions. After all, a video is simply a collection of frames. To share this massive amount of knowledge, Nabil's Books offers Video Books too! 5 videos recorded at a frame rate of 30 FPS and a resolution of 720p are available for the whole world to learn from. These Video Books also range in length from 700 seconds (**44.9 MB**) to 900 seconds (**109 MB**).

1.1.5 – Nabil's Miscellaneous Books

Of course, knowledge comes in many more mime formats than the ones previously listed, and that is why Nabil's Books offers Miscellaneous Books as well! Miscellaneous Books contain types like Zip Books, which hold other Books inside, and PowerPoint Books, which hold very educational slide decks inside. The 4 Miscellaneous Books that are available range in sizes from **391 KB** to **2784 KB**.

1.2 – The Problem

Nabil is the owner of Nabil's Books, and he has received many consumer complaints about the books taking too long to request, despite clients having a very fast internet connection. Online consumer reviews show that a large number of Nabil's clients are dissatisfied with the service due to its latency, with many considering a switch to Nabil's Books' main competitor, Daniel's Books, Nabil, of course, sees this as a problem and wants to

address it. So, Nabil sets up a checklist of goals his bookstore must achieve in order to prosper.

1.2.1 – Nabil's Goals

Nabil wants to achieve the following goals in order to keep his global customer base happy:

- Allow for as many as 1,000 concurrent requests without crashing

- Customer downloads should rarely wait more than 7 seconds from the server, only doing so when there is an unusually high traffic

Currently, all of Nabil's Books is hosted in a Vultr server that has one CPU and one Disk. He has numerous what if questions that he wants answered, including:
- What if Video Books are only served by a new identical disk to invest on?
- What if the current server has more than 1 CPU?
- If another disk is chosen, should it have more than 1 CPU?

2 Data

2.1 – Initial Analysis

2.1.1 – File Size Statistics

Book Num	Text	Image	Audio	Video	Misc
1	5120	2939	7814	46055	392
2	10240	6607	15627	49166	710
3	15360	11743	23439	65865	933
4	20480	18346	31252	69786	2785
5	25600	26416	39064	111960	

Figure 2.1: File Sizes (KB)

Figure 2.1 shows the sizes of each Book that is currently in inventory.

File Sizes (KB)	Text	Image	Audio	Video	Misc	Overall
Mean	15360	13210.2	23439.2	68566.4	1205	25320.79167
Median	15360	11743	23439	65865	821.5	16986.5
Stand. Dev.	7240.773	8390.734	11048.543	23558.133	932.2524	26557.51772
Variance	52428800	70404414	122070313	554985631	869094.5	705301747.6
Range	20480	23477	31250	65905	2393	111568
Minimum	5120	2939	7814	46055	392	392
Maximum	25600	26416	39064	111960	2785	111960
Sum	76800	66051	117196	342832	4820	607699
Count	5	5	5	5	4	24
T(1-A/2, N-1)	2.776445	2.776445	2.7764451	2.7764451	3.182446	2.06865761
1/2 95% CI	8990.608	10418.47	13718.579	29251.286	1483.422	11214.25622
Coef. of Var.	0.471405	0.635171	0.4713703	0.3435813	0.773653	1.048842314

Figure 2.2: File Size Statistics

An initial analysis of the sizes of different books shows interesting patterns regarding them. First off, the files in within the same category do not vary much from each other, having a coefficient of variation less than one. The largest intra-category variance is observed for Miscellaneous Books, as one book holds more than triple the median book size for the category. When it comes to the variance of the books as a whole, a high coefficient of variation (1.04) is observed, meaning that the standard deviation was greater than the mean. An explanation for this is that Video Books are, on average, significantly larger than other books. Likewise, Miscellaneous Books are significantly smaller than others. Having this disparity in sizes.

2.1.2 – Workload Characterization

Nabil's Books' management has conducted a log analysis to investigate what proportion of requests is for each type of book. The following results were obtained:

- 25% of user requests ask for Text Books

- 25% of user requests ask for Image Books

- 20% of user requests ask for Audio Books

- 20% of user requests ask for Video Books

- 10% of user requests ask for Miscellaneous Books

(Note: This behavior is internally captured through a pseudo RNG, for the URL http://www.nabild.com/books/getRandomBook)

From this analysis, it is clear that Video Books are on relatively high demand, but as all books are being served by the same disk and CPU, this adds evidence that the cause of these delays may be video books sharing the same disk as everything else.

2.2 – Data Collection

2.2.1 – Server Specifications

Nabil's Books is served by an Ubuntu 18.04 64-bit server. Specifics of the server are as follows:

- Hosted by Vultr- Located in New Jersey

- One CPU Core

- 1024 MB RAM

- 25 GB SSD

- Ubuntu 18.04 x64 OS

2.2.2 – Client Specifications

A client that will generate thousands of requests holds these specifications:

- Located in New Jersey

- 2 CPU Cores

- 4 GB RAM

- 80 GB SSD

- Ubuntu 18.10 x64 OS

The client specifications are mentioned in this report to indicate that the client can indeed act as a load generator for requests to the server.

2.2.3 – Load Testing Software

Numerous tools exist for load testing. It was very pleasing to find out how much the open source community has contributed to develop very powerful automated tools for load testing, available for anyone to download for free.

All in all, out of the available open source testing tools, a few stood out significantly when it comes to generating a load:

- Apache Bench (AB) – A tool designed for benchmarking an Apache HTTP server. This tool allows sending a number of requests and specifying the concurrency level in one command.

- Apache JMeter – A scalable Java framework that can target many protocols in addition to HTTP (e.g.: FTP, SMTP, POP3). It has a graphical user interface that allows for adjusting load parameters.

- Locust – A scalable Python framework designed to target web apps and provide an abundance of information. This tool stood out particularly due to the graphs it generates along with the testing, which provide a more direct visual interpretation of the results.

- NPM loadtest – Like Apache Bench, this tool allows for sending a number of requests and specifying a concurrency level with one command.

When it comes to the server, the tools vmstat and iostat helped monitor server activity in real time, outputting metrics such as CPU utilization, number of reads, number of writes, number of bytes read, number of bytes written, and so on.

2.2.3.1 – Apache Bench

AB was picked as the load generator as Nabil's Books is an Apache Web Server itself. AB provides logging information such as the time it took to execute all requests, the concurrency level, the transfer rate, the number of requests per second, and so on; all of which come in very handy when performing calculations.

2.2.3.2 - DSTAT

DSTAT is a UNIX command that combines output information from both vmstat and iostat, as well as netstat and ifstat. As it does the work of combining information from the aforementioned tools in a user-friendly CSV format, DSTAT was picked to record server statistics during each load test.

2.2.4 – Request Logs

```
res.download(bookPath, (err) => {
  timeStamp = new Date()/1000;
  if (err) {
    console.error("" + timeStamp + ": " +  thisClient + " fail " + bookAddress);
  }
  else {
    console.info("" + timeStamp + ": " +  thisClient + " success " + bookAddress);
  }
});
```

Figure 2.3: Server Request Logging Source Code

In addition to logged output from DSTAT, the server also tracks information through console logs whenever a request is made/finished. When a client makes a request for a book, they are assigned a unique ID and a new entry is appended the log, formatted as "***Timestamp***: ***ClientId*** *request* ***BookPath***". This format is self-explanatory and concise, recording sufficient information to begin calculations later on. When it comes to transaction completions and failures, the same format is followed, with the words "success" and "fail" respectively replacing the word "request". The following are possible log entries that may be generated by the server (pulled from the file data/logs/0.raw logs/server logs/300.log):

- 1556574306.544: 9016 request video/850sec.mp4

- 1556574319.335: 9016 success video/850sec.mp4

- 1556576642.222: 20764 fail audio/1000sec.mp3

Keeping necessary information in a simple format later proves to be a very beneficial step in building our model.

2.2.5 – Initial Sample

With the server, client, and logging system now in place, it was time for load testing this server. To get a general idea on how the server performs when one user sequentially downloads a thousand "random" (2.1.2) books, the server first forwards console log output from book requests to a local file *1.txt*, and then executes dstat:

nabil@NS:~/$ sudo dstat -c -d --disk-util --disk-tps --ouput ./1.csv

The above command stores the following columns in the local file *1.csv*: usr,sys,idl,wai,stl,read,writ,util,#read,#writ. The column idl stands for the CPU idle percentage (=1-CPU Utilization). The columns read and write hold the number of bytes read from/written to the disk. The column util is the disk utilization. The columns #read and #writes and the number of physical IOs on the disk.

By now the server is ready to receive requests, as all the necessary information is being actively logged. It is the client's turn to load test the server with 1,000 sequential requests. The ab command is executed on the client machine:

client@NabilsBooksClient:~/$ ab -n 1000 -c 1 http://nabild.com/books/getRandomBook > ./1.log

The above command executes 1000 requests with a concurrency level of 1 to Nabil's Books, storing the output in the file *1.log*. Outputs of this command are explained in 2.2.3.1.

Analyzing the outputs of the generated logs showed that the client, on average, was able to execute 6.16 requests per second and received 162899.48 KB per second.

The server, on the other hand, received and responded with a success to all 1,000 requests. The time it took to reply to all of them on the server was 162.252 seconds

2.2.6 - Concurrency

Of course, Nabil's Books has millions of clients, and odds are hundreds of them will be downloading books at the same time. And so, the same commands that were executed with different concurrency levels this time to simulate concurrent downloads. Logs for different concurrency levels were named according to the level, with file extensions .csv and .txt for the server, and .log for the client. For example, when testing 5 concurrent requests, the generated files would be named *5.csv* and *5.txt* in the server, and *5.log* in the client. The levels of concurrency that were chosen for testing simulate the levels that the server would experience

during operation: 1, 3, 5, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 300, 400, 500, 1000. During peak traffic (> 200 requests), Nabil's Books sends an RST packet to the client, terminating hundreds of queued requests. This was a great discovery to make during load testing rather than during actual operation.

2.2.6 – Sample Data

While a lot of raw data was gathered, it is of little use as it has not yet been transformed to solve our problem. Transforming this data would require it to be in a proper format first

2.2.6.1 – Discarded Data

It was also realized (through logging) that Nabil's server fails to serve hundreds of files during peak traffic (see *data/logs/0.raw logs/server/300.txt*). To not skew our computations, it is decided that only traffic during normal operation will be analyzed.

2.3 – Preprocessing

```
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking nabild.com (be patient)


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

Document Path:          /books/getRandomBook
Document Length:        3008737 bytes

Concurrency Level:      3
Time taken for tests:   141.247 seconds
Complete requests:      1000
Failed requests:        0
   (Connect: 0, Receive: 0, Length: 0, Exceptions: 0)
Total transferred:      25743647227 bytes
HTML transferred:       25743305521 bytes
Requests per second:    7.08 [#/sec] (mean)
Time per request:       423.740 [ms] (mean)
Time per request:       141.247 [ms] (mean, across all concurrent requests)
Transfer rate:          177988.32 [Kbytes/sec] received
```

Figure 2.4: Raw Data in *data/logs/0.raw logs\client logs/3.log*

2.3.1 – Data Cleanup

As a follow-up to 2.2.6, the data needs to be in a more interpretable format in order to start becoming meaningful. So, irrelevant information was removed from every file. Irrelevant information ranges from English sentences like "This is ApacheBench…" to unnecessary information like "Server Port: 80," and any additional newlines or symbols that are present. Data cleanup was done through finding regular expressions that match irrelevant data portions in the file.

```
Concurrency Level:      3
Time taken for tests:   141.247 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      25743647227 bytes
HTML transferred:       25743305521 bytes
Requests per second:    7.08 [#/sec] (mean)
Time per request:       423.740 [ms] (mean)
Time per request:       141.247 [ms] (mean, across all concurrent requests)
Transfer rate:          177988.32 [Kbytes/sec] received
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.8      0      10
Processing:    10  423 417.7    295    2099
Waiting:        4   14   9.2      8      44
Total:         11  423 417.6    295    2099
```

Figure 2.5: 2.4 Data after cleanup

2.3.2 – Structure

Following the removal of unnecessary attributes, the data needed to become structured to allow for extensive analysis. The structure that was followed was Comma Separated Values (CSV), which is compatible with Microsoft Excel Spreadsheets, the latter would help us compute many metrics and generate graphs. The way structuring was done is similar to data cleanup, which is through regular expressions that match words/spaces and replacing them with commas.

	A	B	C	D	E	F
1	Batch Stati	Concurren	Time Taken (secon	Completed	Failed Requ	Total Transferr
2	Failed	-1		575	425	
3	Passed	100	161.748	1000	0	28031363428
4	Passed	10	126.014	1000	0	25903522940
5	Passed	120	152.079	1000	0	26936939697
6	Passed	140	168.625	1000	0	26829930440
7	Passed	160	159.069	1000	0	25538011313
8	Passed	180	157.093	1000	0	25085942203
9	Passed	1	162.256	1000	0	27065818487
10	Passed	200	169.653	1000	0	27603185322
11	Passed	20	134.643	1000	0	25700863617
12	Failed	-1		610	390	
13	Passed	3	141.247	1000	0	25743647227
14	Failed	-1		591	409	
15	Passed	40	155.815	1000	0	27179388911

Figure 2.6: 2.5 Data after structuring, in CSV format

2.3.3 – Client Computations

(Note: *logs/2.calculations/client logs/all.xlsx* contains all averages and graphs shown here).

During normal business hours, 79 concurrent users should be expected to be downloading a file at the same time as a user. The time it should take to download a file is 0.152 seconds, and one should be able to download 6.6 books per second. The following graph was generated to illustrate how the throughput gets affected by concurrency level:

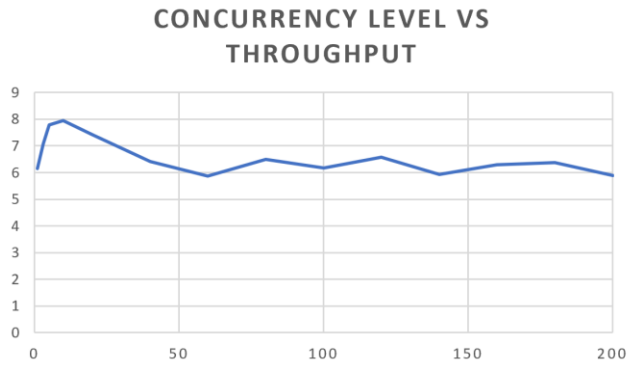

Figure 2.7: Concurrency Level vs. Throughput

As can be seen, there seems to be a "knee point" in the throughput when the concurrency level approaches 10, after which the server declines in performance slowly, eventually crashing and not being

able to handle any more requests when more than 200 concurrent requests are occurring

2.3.4 – Preparing for a QN Model

Recording metrics such as system throughput, overall CPU utilization, and overall disk utilization over dozens of "random" 1000 book requests reveals quite a bit of information. For one, as seen in figure 2.7, a local optimum is reached when the concurrency level is around 10. This detail suggests some rework being needed as the optimum should match the average concurrent requests in order to provide customer satisfaction.

3 Queuing Network (QN) Model

3.1 – Type of Model

Nabil's books, due to knowing the concurrency level of requests in its logs, can be thought of as close QN. Having 5 different classes of books makes it a Multi-Class Closed Queueing Network. Along with the 5 classes, the bookstore currently has only one CPU and one disk, either of which can be considered a culprit for the latencies experienced by the customers.

3.2 – Service Demands

In order to proceed with the model, service demands for each class need to be computed. The Service Demand Law states that Service Demand is equivalent to dividing the utilization by a class of a resource by the system throughput of that resource. One thing is missing however, no data was recorded to find out individual class utilizations of the either the CPU or the Disk, so an investigation is started to find out what was overlooked.

3.2.1 – Class Utilizations

To determine the utilization of a single class, only requests from that class must be monitored from the logs. Unfortunately, our "random" configuration from 2.1.2 is not enough to simulate that, especially due to concurrency, so a new request is implemented for the server. This time, the /books/getRandomBooks route accepts a sub path that specifies the type of book being requested. A sample URL is: http://nabild.com/books/getRandomBook/text to only request a Text Book. The possible sub paths that are handled are ["text", "image", "audio", "video", "misc"]. And so, batches of 200 book requests are made to each class, monitoring CPU Utilization and Disk Utilizations. The following results are obtained:

Utilizations Per Class

Class	Text	Image	Audio	Video	Misc
CPU Util	54.30991	69.24179	81.38261	90.83822	20.22611
Disk Util	0.011857	0.550625	0.810921	0.065459	0.090556

Figure 3.1: CPU and Disk Utilizations per class

3.2.2 – Service Demand Computations

As per service demand law,

$$D_{i,r} = U_{i,r}/X_{0,r}$$

Or in other words, the service demand of a resource is equal to its utilization divided by the system throughput. In our case, we found out that the system throughput is 6.6 in 2.3.3, so all that's left is dividing the utilizations found in Figure 2.8 by it to get the service demands, resulting in this:

Service Demands Per Class

Class	Text	Image	Audio	Video	Misc	Total
CPU Dema	0.082279	0.1049	0.123294	0.1376188	0.030642	0.478734
Disk Dema	1.8E-05	0.000834	0.001229	9.917E-05	0.000137	0.002317
Total	0.082297	0.105735	0.124522	0.1377179	0.030779	

Figure 3.2: Service Demands per class for disk and CPU

Sure enough, the bottleneck occurs in the CPU, with the highest user being none other than the videos that were discussed earlier. This gives a huge answer to Nabil. Investing in another disk will simply be a waste of money, as the all the contention occurs at the CPU. So it will be much wiser and more cost effective to purchase another CPU instead of another disk. Now onto solving this QN and answering the remaining What If? questions.

3.4 – Solving the QN

3.4.1 – Obtaining Class Throughputs

One of the input parameters for the QN is the throughput for each class. This is where our server's book request logs come in to play. This log included timestamps of when the request was started and when in was served. The time interval during which the server was busy was simply the maximum timestamp – the minimum, and the number of requests made is the number of requests present in the log. Thus, using these two metrics, and dividing the requests by the time interval gave us throughputs for each class.

3.4.2 – Distributing the Number of Requests

No. Requests per Class:	0.25	0.25	0.2	0.2	0.1

Figure 3.3: One concurrent request gets split "random"-ly

When determining the number of requests per class, as we are load testing our server under a normal workload, which follows the "random" proportions described in 2.1.2, we apportion the number of concurrent requests per class to follow the distribution

3.4.3 – Solving the QN

Conc Level	CPU Util	Disk Util	TEXT	IMG	AUD	VID	MISC	Queue Len	Queue Len
1	0.99547	0.00457	0.08194	0.10608	0.12516	0.13719	0.03078	0.995441	0.004559
3	0.99949	0.00462	0.24648	0.31533	0.37095	0.41235	0.09197	2.995374	0.004626
5	0.99981	0.00463	0.41103	0.52502	0.61737	0.68758	0.15323	4.995364	0.004636
10	0.99995	0.00463	0.82243	1.04944	1.23372	1.37566	0.30643	9.995357	0.004643
20	0.99999	0.00463	1.64521	2.09840	2.46659	2.75184	0.61285	19.99535	0.004646
40	0.99999	0.00463	3.29079	4.19639	4.93244	5.50421	1.22569	39.99535	0.004647
60	1.00000	0.00463	4.93637	6.29439	7.39830	8.25659	1.83854	59.99535	0.004648
80	1.00000	0.00463	6.58194	8.39240	9.86417	11.00896	2.45138	79.99535	0.004648
100	1.00000	0.00463	8.22752	10.49040	12.33004	13.76134	3.06423	99.99535	0.004648
120	1.00000	0.00463	9.87310	12.58841	14.79591	16.51371	3.67707	119.9954	0.004648
140	1.00000	0.00463	11.51867	14.68642	17.26178	19.26609	4.28992	139.9954	0.004648
160	1.00000	0.00463	13.16425	16.78443	19.72765	22.01847	4.90277	159.9954	0.004649
180	1.00000	0.00463	14.80983	18.88244	22.19352	24.77084	5.51561	179.9954	0.004649
200	1.00000	0.00463	16.45540	20.98045	24.65939	27.52322	6.12846	199.9954	0.004649

Figure 3.4: Reorganized QN Solution

With the throughputs, number of concurrent requests, and the correct proportions for each request, the Close QN was ready to solve. The solver was run 14 times to determine the metrics that were sought for the different concurrency levels. The output of the QN solver was taken and restructured to a readable model.

3.5 – Results

Note: results that are included have a computed margin of error (+/-). For preciseness, the following are the margins of error for the date points in these graphs:

:onc Level	Error
1	0.000140378
3	2.95908E-05
5	7.7309E-05
10	0.000188072
20	0.000402238
40	3.74691E-06
60	5.70611E-06
80	7.66392E-06
100	9.62118E-06
120	1.15781E-05
140	1.3535E-05
160	1.54917E-05
180	1.74483E-05
200	1.94049E-05

Figure 3.5: Error margin per concurrency level

What follows are graphs of different measures that were taken against the concurrency level. A notable detail from these graphs is that, as section 2.3.3 hinted at, a pattern starts to be followed after the concurrency level exceeds 10.

3.5.1 –CPU Queue length vs. Concurrency Level

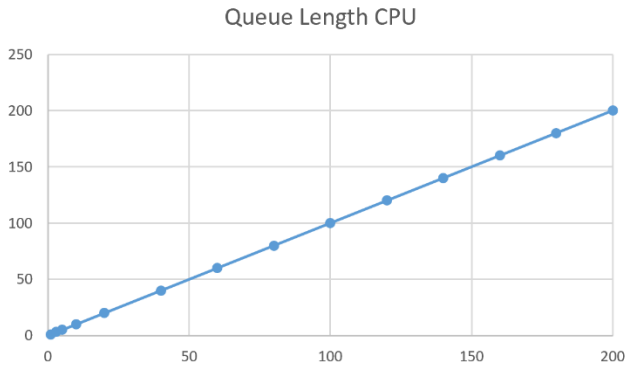

Figure 3.6: Linear CPU Queue Length

3.5.2 –Disk Queue length vs. Concurrency Level

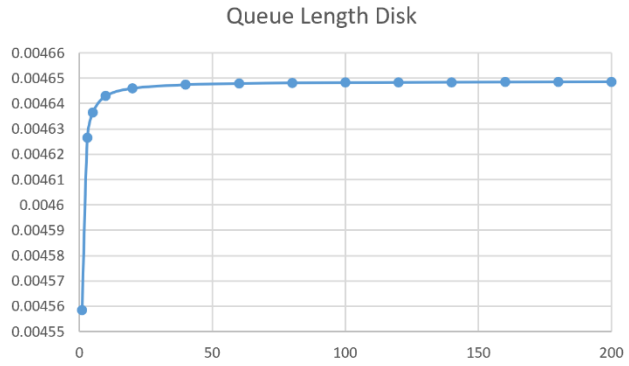

Figure 3.7: Asymptotic Disk Queue Length

3.5.3 – CPU Utilization vs. Concurrency Level

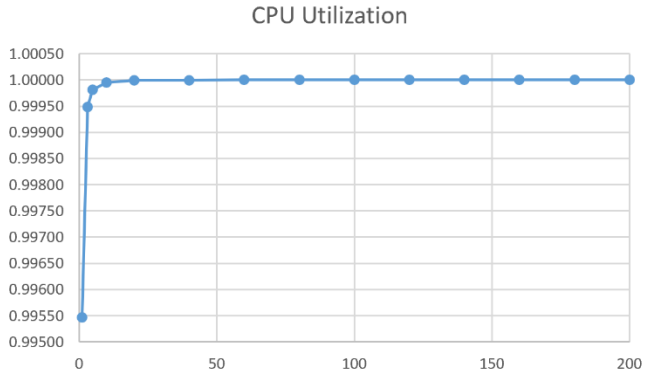

Figure 3.8: Asymptotic CPU Utilization

3.5.4 – Disk Utilization vs. Concurrency Level

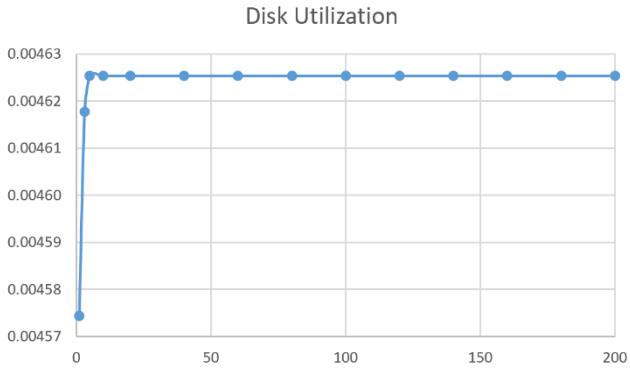

Figure 3.9: Asymptotic Disk Utilization

3.5.5 – Class Residence Times vs. Concurrency Level

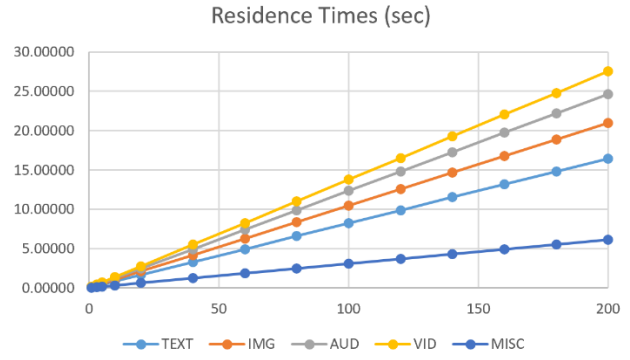

Figure 3.10: Linear Residence Times

3.5.6 – Result Interpretation

As can be seen, the bottleneck device is the CPU, with utilization reaching 100% when the concurrency level is only 10. Due to that, the queue length for the CPU starts growing linearly, meaning every new request would be ever waiting for the CPU. The disk, on the other hand, is barely being utilized.

3.6 – Solving the Problem

By now we are able to answer the what if questions Nabil initially asked.

- What if Video Books are only served by a new identical disk to invest on?
While video books are indeed the most demanding class, having an identical disk serve those specific books is of no use, as there is no significant bus contention on the current disk to begin with.

- What if the current server has more than 1 CPU?
This is highly needed. In order to meet ever increasing consumer demands, it is imperative to install more CPUs/cores until the CPU is no longer a bottleneck device.

- If another disk is chosen, should it have more than 1 CPU?
Definitely, the more CPUs available, the less contention there will be on an individual one of them. But again, for the time being, a new disk is not necessary at all as disk utilization barely exceeds 1%. In order to save on costs, simply buying more CPUs would solve problems.

4 Contribution

Nabil's Books is available on Github under the URL, https://github.com/ndarwich/nabildarwich.com.