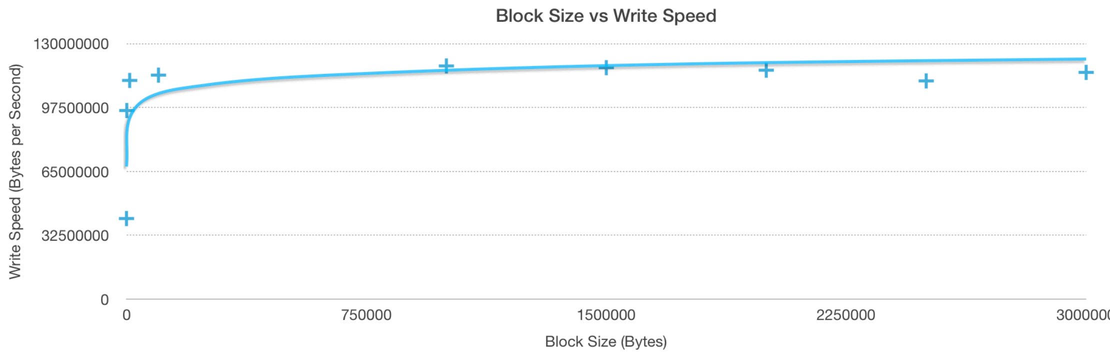
# CSCD43 IO Investigation

**Brandon & Henry**

## Introduction

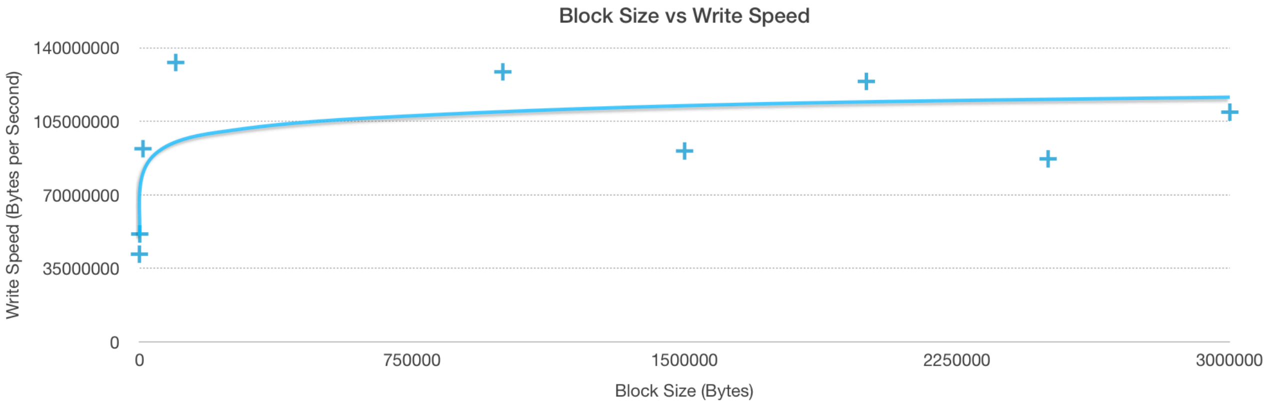
For this assignment, we were asked to explore the effects of IO on the disk, and to look at the speed of reading and writing from the disk using different block sizes. As we created our code and tested it on various machines, we noted some interesting results that surprised us! See below for our results.




## Writing to a NVME SSD on OS X

Testing the experiment on a laptop with a PCI-E SSD and writing a file of size 10MB, we can see that the graph approximates a logarithmic curve, and that after a block size of around 1 MB, we see negligible gains in performance. Furthermore, we also see that you need at least 1KB of block size to achieve acceptable (SSD) speeds.

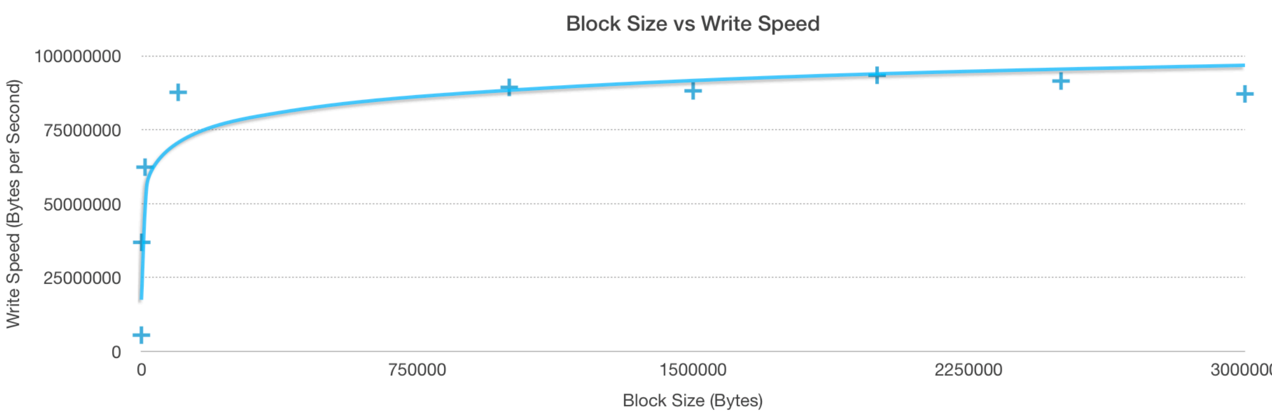
| BLOCK SIZE | BYTES PER SECOND |
|---|---|
| 100 | 41011340 |
| 1000 | 95967448 |
| 10000 | 111259457 |
| 100000 | 113958815 |
| 1000000 | 118635220 |
| 2000000 | 116472740 |
| 3000000 | 115344245 |
| 1500000 | 117753730 |
| 2500000 | 111011201 |




## Writing to a HDD on Linux

Firing up our school computers, we ran our script on their hard disks and the results we found were interesting to say the least. It appears that they were writing faster than our laptops! Furthermore we see that the block size is optimal at around 1MB.

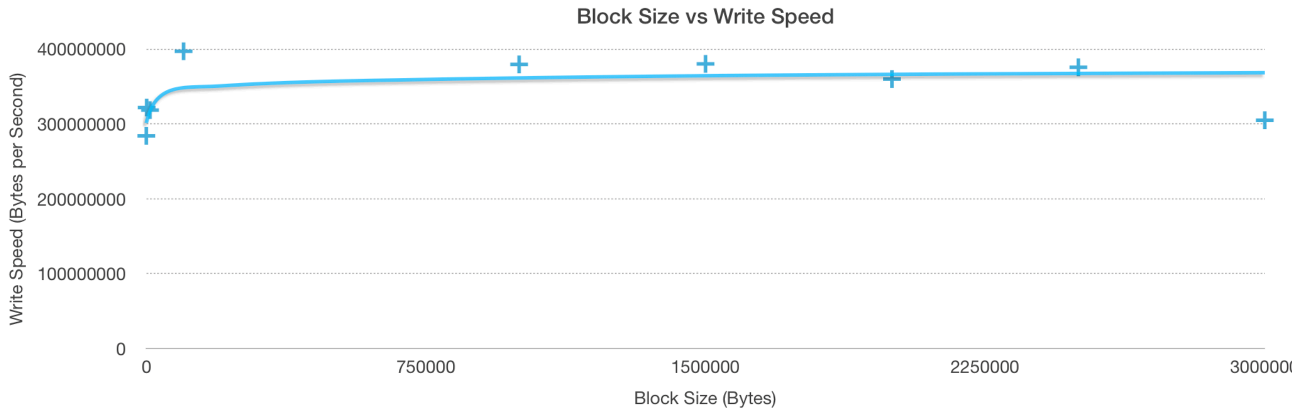
| BLOCK SIZE | BYTES PER SECOND |
|---|---|
| 100 | 41790995 |
| 1000 | 51360805 |
| 10000 | 91943032 |
| 100000 | 132987566 |
| 1000000 | 128524792 |
| 2000000 | 123994098 |
| 3000000 | 109357742 |
| 1500000 | 90883478 |
| 2500000 | 87114085 |




## Writing to a USB on Linux

Finally we plugged a USB drive into the lab machines and ran our script, which resulted in a similar result, with the optimal block size appearing to be around 1.5MB

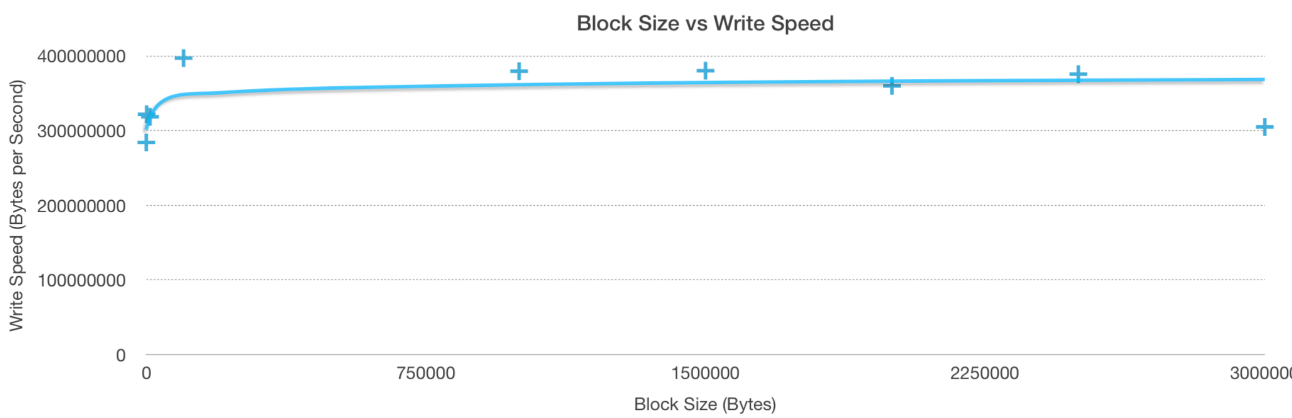
| BLOCK SIZE | BYTES PER SECOND |
|---|---|
| 100 | 5518182 |
| 1000 | 36907996 |
| 10000 | 62344917 |
| 100000 | 87670080 |
| 1000000 | 89338360 |
| 2000000 | 93440478 |
| 3000000 | 87133820 |
| 1500000 | 88161653 |
| 2500000 | 91480428 |




## Reading from a NVME SSD on OS X

From observing the read rates, the first thing we see is that block size does not significantly impact their rates, however the optimal size is still somewhere between 1MB and 1.5MB.

| BLOCK SIZE | BYTES PER SECOND |
|---|---|
| 100 | 284082839 |
| 1000 | 321864238 |
| 10000 | 318461196 |
| 100000 | 397030214 |
| 1000000 | 379506641 |
| 2000000 | 359880520 |
| 3000000 | 304887344 |
| 1500000 | 380170316 |
| 2500000 | 375629179 |




## Reading from a HDD on Linux

From observing the read rates, the first thing we see is that block size does not significantly impact their rates, however the optimal size is still somewhere between 1MB and 1.5MB. It's interesting to note that the speeds were close to or faster than the SSD, so something seems wrong

| BLOCK SIZE | BYTES PER SECOND |
|---|---|
| 100 | 331191627 |
| 1000 | 374405631 |
| 10000 | 379679550 |
| 100000 | 370041445 |
| 1000000 | 369890882 |
| 2000000 | 353769413 |
| 3000000 | 404858300 |
| 1500000 | 392603353 |
| 2500000 | 375629179 |

# Comparison & Thoughts

It appears that both the read & write speed of our drives were similar (which is not what we expected since they are rated to be much different). This leads to a few questions and interesting points that we have thought about. First of all, it appears that Linux overall had much better performance in comparison to Mac OS, which suggests maybe EXT4 is better than HSF+. Furthermore, it appears as though the function “fwrite()” may not directly flush to disk, and instead may flush to cache instead, leading to the similar performance between a laptop SSD & desktop HDD, or perhaps the Linux computers contain a super fast network drive in RAID. Another explanation could be that “fwrite()” does not wait for the file to be written to disk, so writing to disk and immediately recording the time does not account for the full time taken, leading to inflated numbers. Furthermore, upon further investigation we believe there is some sort of inefficiency in the code or the compiler optimized code, as performing a direct write to disk by performing an operation such as “dd if=/dev/zero of=/dev/null bs=1024” yields much different numbers than ours, with the SSD’s performing much closer to their expected read/write speeds.