Ely Schoenfield

Doctor Dixon

CSCI-440

14 December 2023

Testing Different File Systems Read Performance of Small Files

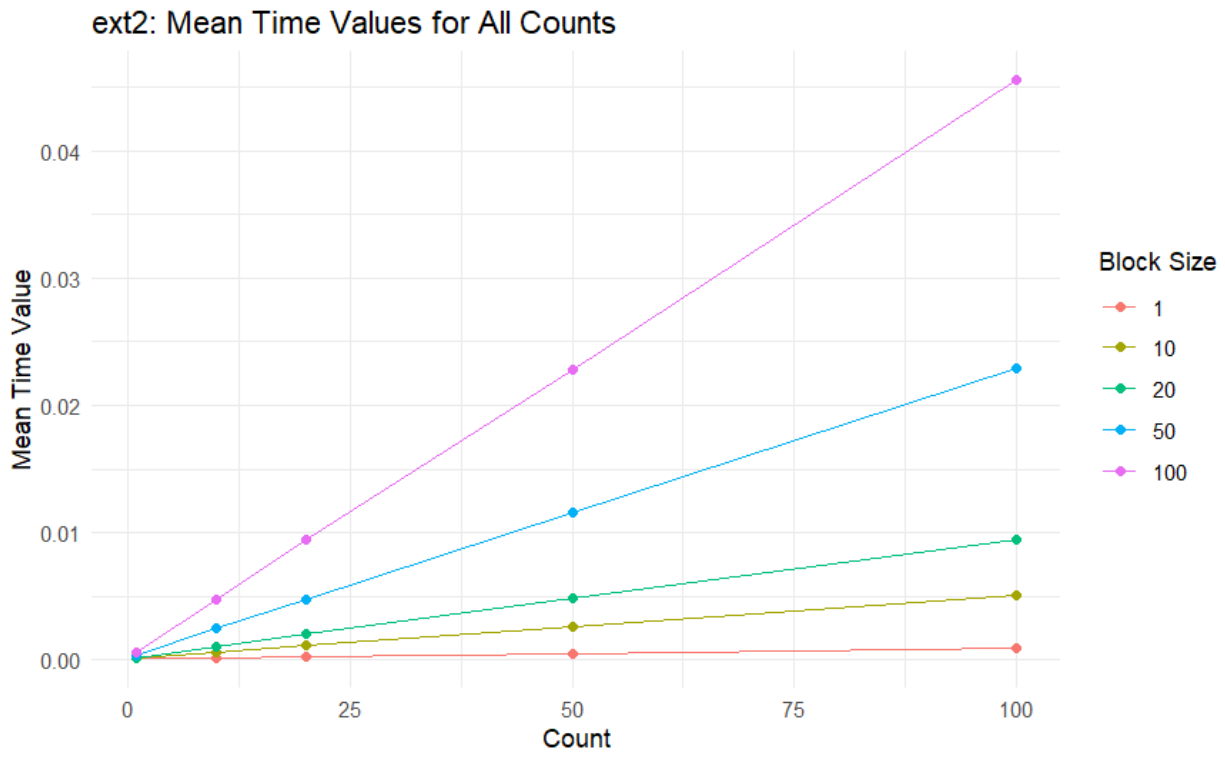
Within the domain of Linux file systems, choosing the most effective file system solution largely depends on the expected characteristics of the files that are to be read and written. By comparing the performance of the ext2, ext4, XFS, ZFS, and FAT32 file systems, this research aims to provide insight for what file system performs the best for reading small files on Linux.

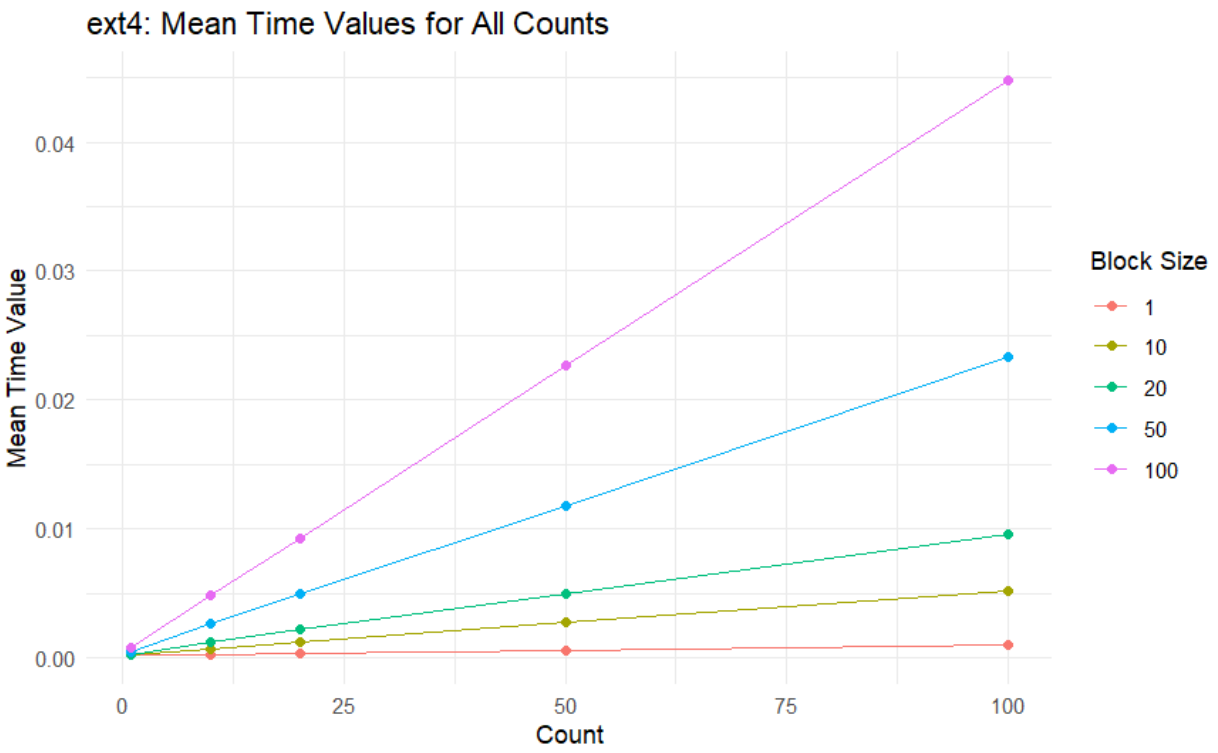
The experiment was set up on a Google Cloud virtual machine (VM) instance. The VM was of type n1-standard-1 (1 vCPU, 3.75GB memory) and had an added 15 GB standard persistent disk for testing the file systems on. While ext2, ext4, XFS, and FAT32 were all partitioned and tested on the same standard persistent disk there was a Google Cloud ssh issue that prevented the testing of ZFS on the same standard persistent disk as the rest of the file systems. This difference may be negligible since both virtual machine instances were provisioned with the same type, specs, and location.

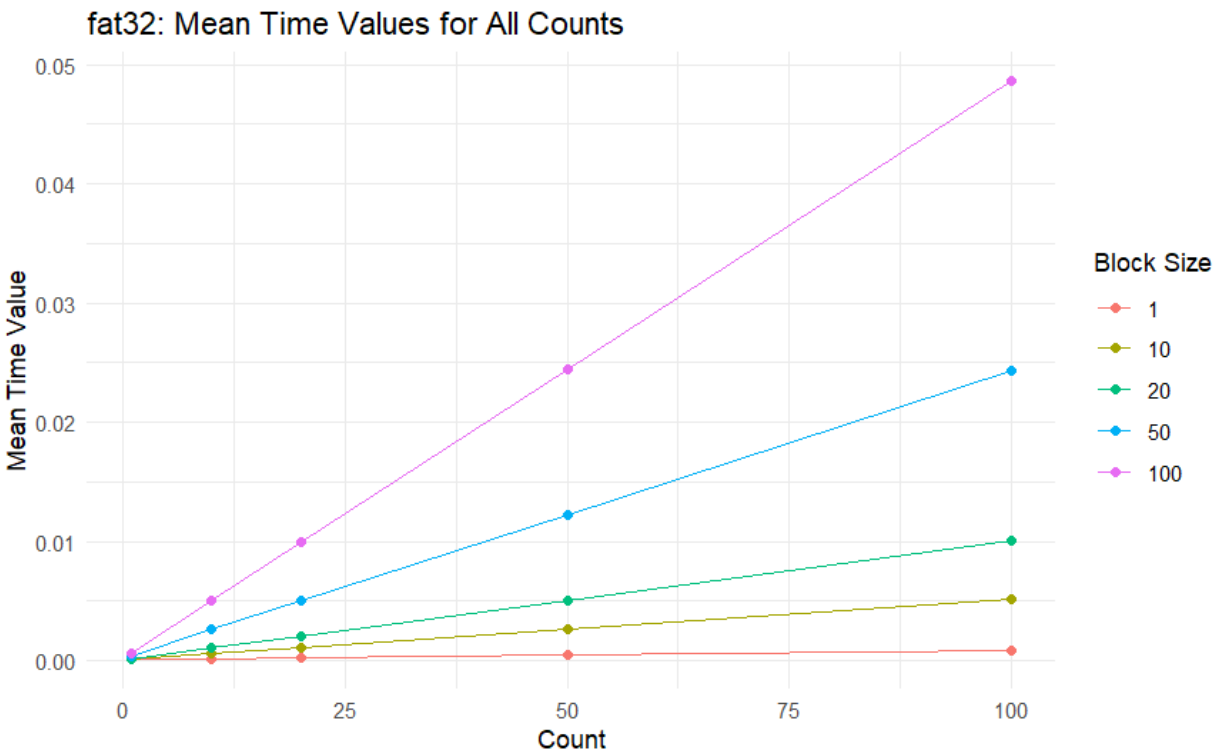
The tests for the experiment were run with a python script I developed. The VM's standard persistent disk was partitioned with a file system and the python script calls the linux dd command to convert and copy random bytes to a sample text file. The script then copies the output of the dd command and writes the block size, file size, and time to a csv file. The script would test variation in the dd command's block size and counts

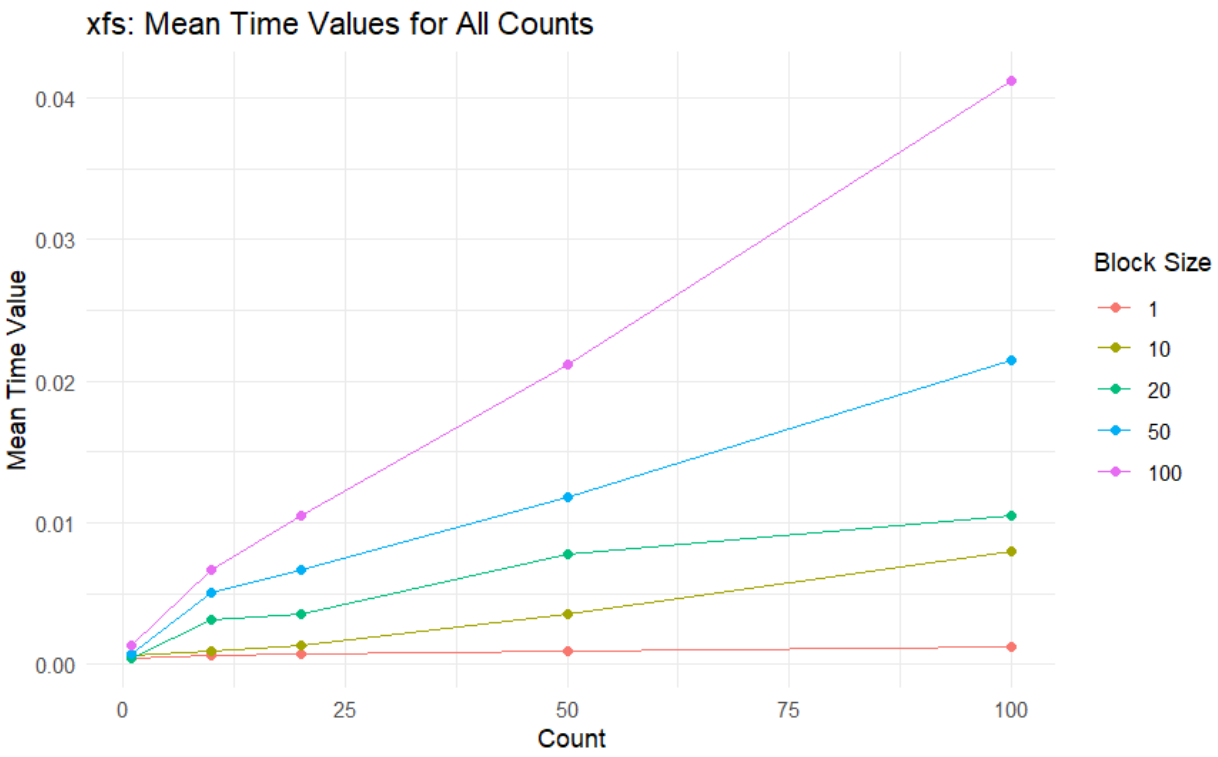
with a set of 1, 10, 20, 50, 100. To make sure the results were accurate each block size and count combination was run a thousand times and the mean was calculated for each graph.

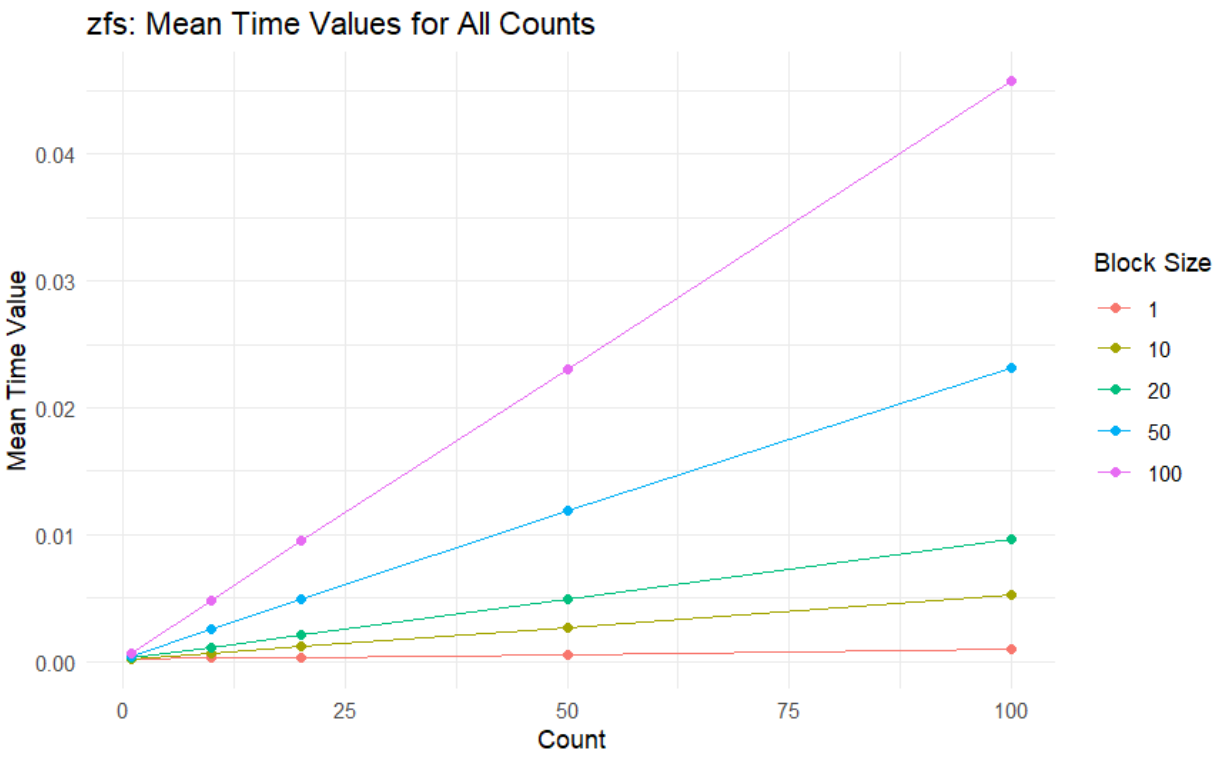
The following five graphs are of each individual file system's mean time for all block sizes and counts. All graphs were made in R-Studio from the csv files created from the testing python script.

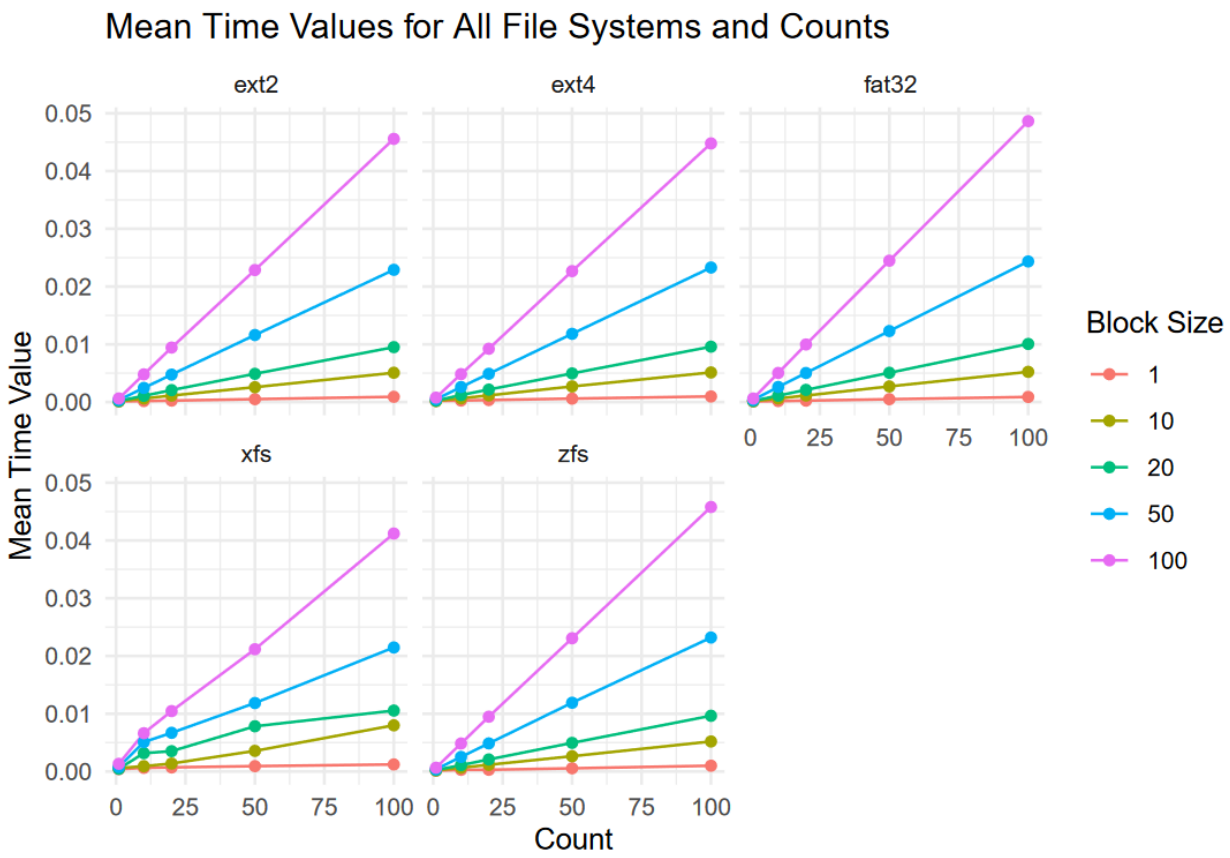



ext4: Mean Time Values for All Counts
Mean Time Value
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100
Count
Block Size
1
10
20
50
100

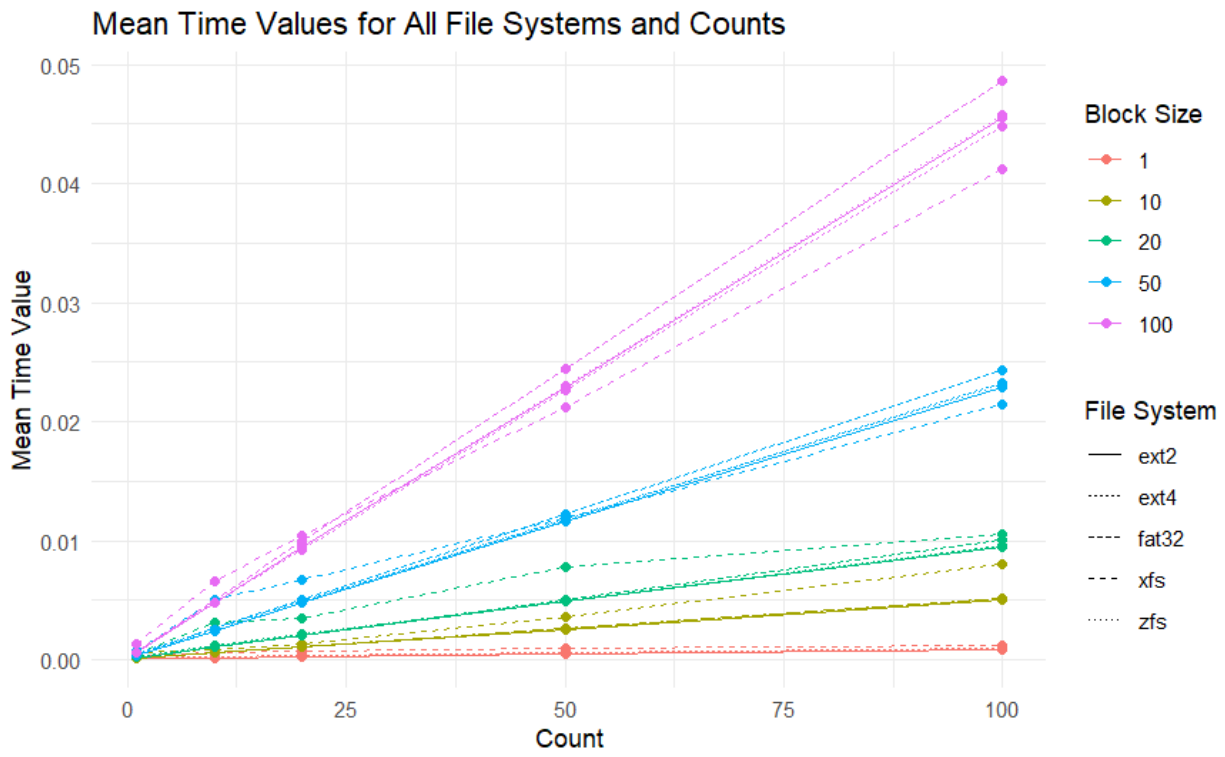

fat32: Mean Time Values for All Counts
Mean Time Value
0.05
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100
Count
Block Size
1
10
20
50
100


xfs: Mean Time Values for All Counts
Mean Time Value
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100
Count
Block Size
1
10
20
50
100


zfs: Mean Time Values for All Counts
Mean Time Value
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100
Count
Block Size
1
10
20
50
100



The above graph is all the file system's mean times for each block size and count compared to one another. Closely looking at the graph we can see that some file systems perform better than others when it comes to certain block sizes and counts.



We learned by observing the graphs on page five and six, that it's apparent XFS had the worst mean time for the smallest block sizes and counts. This may be because XFS is good at storing large files, but not small files. Looking further into the results, FAT32 out performed XFS for using on small files because it's compatible across various operating systems. ZFS out performed FAT32 for use on small files. This may be caused by the overhead that comes with ZFS's data integrity check and advanced features. The results collected from ext4 shows it out performs ZFS. This could be from ext4 use of multiblock allocation, delayed allocation, and journal checksum. It's not noticeable from the graphs, but while observing the combined mean time table it was noted that ext2 out performed ext4 for reading small files. This is because ext2 has very low overhead compared to all the other tested file systems.

To conclude, listing the file systems that perform the best to worst for reading small files on Linux is as follows; ext2, ext4, ZFS, FAT32, then XFS. This list is based on thorough analysis of the benchmarks that were recorded when testing with python script. Further testing of different file systems may reveal a file system that's faster than ext2 at reading small files on Linux.

Works Cited

- https://askubuntu.com/questions/384062/how-do-i-create-and-tune-an-ext4-partition-from-the-command-line
- https://github.com/CSUChico-CSCI345/CSCI345-Course-Materials/blob/main/assignments/lab3.md
- https://linux.die.net/man/1/dd
- https://realpython.com/python-subprocess/
- https://docs.python.org/3/library/re.html
- https://ggplot2.tidyverse.org/