File Size Distribution on UNIX Systems—Then and Now

Andrew S. Tanenbaum, Jorrit N. Herder*, Herbert Bos
Dept. of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
{ast@cs.vu.nl, jnherder@cs.vu.nl, herbertb@cs.vu.nl}

ABSTRACT

Knowledge of the file size distribution is needed to optimize file system design. In particular, if all the files are small, the disk block size should be small, too, to avoid wasting too large a fraction of the disk. On the other hand, if files are generally large, choosing a large block size is good since it leads to more efficient transfers. Only by knowing the file size distribution can reasonable choices be made. In 1984, we published the file size distribution for a university computer science department. We have now made the same measurements 20 years later to see how file sizes have changed. In short, the median file size has more than doubled (from 1080 bytes to 2475 bytes), but large files still dominate the storage requirements.

1. INTRODUCTION

Twenty years ago we published a study of static file sizes at the Computer Science Dept. of the Vrije Universiteit (VU) [1]. These files represented the totality of files owned by students and faculty members on the Dept.'s UNIX machines. For another research project, we recently had a need to revisit these measurements to see how much file sizes have changed in the past 20 years. With the recent popularity of digital photos, music, and videos, it is certainly possible that this data is now dated. In this note we present new measurements on the Computer Science Dept.'s UNIX file systems and compare them to the old ones. As a second source of data, we ran the same measurements on a Linux machine running an Apache Web server.

2. THE DATA

The results of the three studies are presented in Fig. 1.

Length	VU 1984	VU 2005	Web
1	1.79	1.38	6.67
2	1.88	1.53	7.67
4	2.01	1.65	8.33
8	2.31	1.80	11.30
16	3.32	2.15	11.46
32	5.13	3.15	12.33
64	8.71	4.98	26.10
128	14.73	8.03	28.49
256	23.09	13.29	32.10
512	34.44	20.62	39.94
1024	48.05	30.91	47.82
2048	60.87	46.09	59.44
4096	73.51	59.13	70.64
8192	84.97	69.96	79.69

Length	VU 1984	VU 2005	Web
16,384	92.53	78.92	86.79
32,768	97.21	85.87	91.65
65,536	99.18	90.84	94.80
131,072	99.84	93.73	96.93
262,144	99.96	96.12	98.48
524,288	100.00	97.73	98.99
1,048,576	100.00	98.87	99.62
2,097,162	100.00	99.44	99.80
4,194,304	100.00	99.71	99.87
8,388,608	100.00	99.86	99.94
16,777,216	100.00	99.94	99.97
33,554,432	100.00	99.97	99.99
67,108,864	100.00	99.99	99.99
134,217,728	100.00	99.99	100.00

Fig. 1. Percent of files smaller than a given size (in bytes).

The first column of each table gives file sizes from 1 byte to 128 MB. The second column tells what percentage of all files in 1984 were that size or smaller. For example, 48.05% of all files in 1984 were 1024 bytes

* This work supported in part by the Netherlands Organisation for Scientific research under grant 612-060-420.

or less. This puts the median file size at just over 1 KB. Put in other terms, a file system with a 1-KB block size could store almost half of all files in a single disk block. Over 85% of the files were 'small' files in the sense of fitting in 10 1-KB disk blocks, and thus not needing indirect blocks in the i-nodes to keep track of them. In 1984, all files were 524,288 bytes or smaller.

The third column gives the data for 2005. Files are definitely larger now, with the median file size in 2005 being 2475 bytes and the percentage of files that are 'small' (10 1-KB disk blocks or fewer) being 73%. Furthermore, the largest file recorded was now 2 GB, more than 4000 times larger than in 1984.

While our university file system is probably typical of other UNIX file systems at major research universities with 1000+ login names, it may not be typical of other applications. To provide some idea of what the distribution looks like for other applications, we ran the same measurements on a Linux file system being used as a Web server (for *www.electoral-vote.com*). This system, which was at a commercial hosting service in upstate New York, had only 53,000 files (vs. 1.7 million at the VU). The results are given in the fourth column. Somewhat surprisingly, on the whole, the Web server's files were smaller than the university's, with a median file size of 1180 bytes. In part this result is due to the relatively large number of icons, small .gif files, and short Perl scripts on the server.

Another way of looking at the data is by graphing it. In Fig. 2 we see the data of Fig. 1 in graphical form. We see more clearly here that the shape of the curve for the three measurements is largely the same, with the 2005 measurements offset to the right to indicate the growth in file size. The Web has relatively more small files than the university measurements, but for larger sizes, it falls in between the 1984 and 2005 measurements.

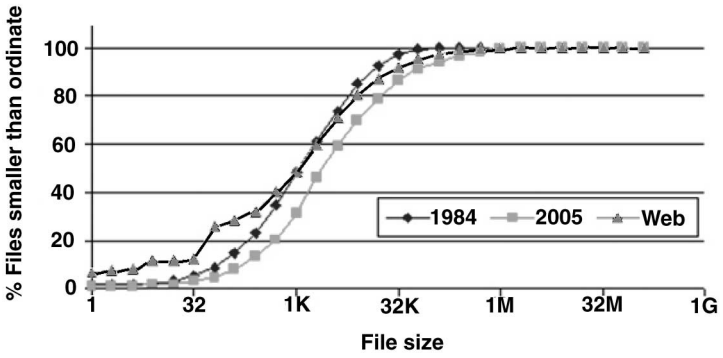

Fig. 2. Data of Fig. 1 shown graphically.

Another way of presenting the data is to look at it by decile, as is done in Fig. 3 for the 2005 VU files. Here the files are grouped into deciles by size. Each decile contains 10% of all the files. The first decile contains files from 0 to 164 bytes, the second decile contains files from 165 bytes to 485 bytes, and so on. Put in other words, 10% of all files are 164 bytes or smaller, 20% of all files are 485 bytes are smaller, half of all files are 2475 bytes or smaller and 90% of all files are 56,788 bytes or smaller.

3. THE EFFECT OF BLOCK SIZE ON STORAGE REQUIREMENTS

The reason for collecting this data was to determine how big disk blocks should be. File systems are generally free to choose any size they want since block size is transparent to user programs, but the efficiency of the file system depends on the block size. Large disk blocks are more efficient than small ones since the seek and rotational latency is then amortized over more bytes gathered. This fact argues for using a very large disk block. On the other hand, since most files are small, a large disk block wastes a great deal of space on the disk. From Fig. 3, we see, for example, that even a 1-KB disk block wastes more than half the block for the 20% smallest files. With a 16-KB disk block and files of under 512 bytes, over 95% of the block is wasted. This fact argues for a small block size. To make a reasonable choice, we need to look at the actual file size distribution in detail.

For a given block size, each file occupies a certain number of disk blocks. For example, with a 1-KB block size, a file of 2300 bytes occupies 3 blocks and 3072 bytes of storage. With a 2-KB block size it uses 2

Decile	Largest	Block size				
		1 KB	2 KB	4 KB	8 KB	16 KB
10%	164	0.08%	0.16%	0.31%	0.60%	1.13%
20%	485	0.17%	0.33%	0.66%	1.28%	2.43%
30%	968	0.26%	0.51%	1.01%	1.97%	3.74%
40%	1622	0.43%	0.69%	1.37%	2.66%	5.04%
50%	2475	0.64%	0.94%	1.72%	3.35%	6.35%
60%	4316	0.97%	1.31%	2.10%	4.04%	7.65%
70%	8211	1.55%	1.93%	2.81%	4.73%	8.95%
80%	18,149	2.68%	3.10%	4.07%	6.18%	10.40%
90%	56,788	5.52%	5.97%	6.99%	9.19%	13.65%
100%	2.14 GB	100.00%	100.00%	100.00%	100.00%	100.00%

Fig. 3. File size distribution in bytes and weight by decile. Column 2 gives the largest file in the decile. Columns 2-5 give the percentage of the disk used by files of that decile and smaller for various disk block sizes.

blocks and 4096 bytes of storage. With an 8-KB block size it uses only 1 block, but 8192 bytes of storage. Thus the block size affects how much storage is needed for the 1.7 million files measured.

The third column of Fig. 3 works with the assumption of 1-KB disk blocks. What it shows is the collective size of the 10% smallest files together take up 0.08% of the disk (excluding unused blocks), the collective size of the smallest 20% of the files take up 0.17% of the disk, and the collective size of the smallest half of the files take up only 0.64% of the disk. In fact the 90% smallest files (1.5 million files) take up only 5.52% of the disk in use. The largest 10% of the files use the other 94.48% of the disk. In short, virtually the entire disk is filled by the 10% largest files; the small files hardly matter at all.

The fourth column assumes 2-KB disk blocks. For the very smallest 10% of the files (all 164 bytes or less), doubling the block size does not change the number of blocks they occupied, but it does double the number of KB of storage used, since these 170,000 files now each occupy 2 KB instead of 1 KB. In percent terms, the first decile now occupies 0.16% of the disk instead of 0.08% of the disk. But as we move to larger files, the effect is much smaller. With a file above 1 MB, going from 1 KB to 2 KB might at most add another 1 KB to the length, but as a percentage of what the file needs, it is negligible. All in all, with 2-KB blocks, the smallest 90% of the files use only 5.97% of the occupied blocks.

The next three columns show the same calculations for 4-KB, 8-KB, and 16-KB disk blocks. Even with 16-KB blocks, which is too big for more than 3/4 of the files, the bottom 90% of the files still account for only 13.65% of the allocated blocks. In short, while using large disk blocks is very inefficient for small files, despite the fact that most files are small, the total amount of wasted space on the disk is still very small. The reason is simple: the disk is mostly occupied by very large files, whose space efficiency is largely independent of the block size.

One other figure that is interesting is how the total amount of disk storage rises with block size. Remember that the smallest 30% of the files are 968 bytes or less, so going from 1 KB to 2 KB increases the total amount of storage each of these files uses by a factor of two. For large files, the percentage increase is very small, of course. We have calculated how much additional storage is needed for the entire file system when going from 1 KB to larger sizes. For 2 KB, 4 KB, 8 KB, and 16 KB, the percent increase in total disk storage required is 1%, 2%, 4%, and 10% respectively. Thus going from a 1-KB block to a 16-KB block means the VU file system increases by 10%, hardly a huge amount.

4. DYNAMIC FILE SIZE DISTRIBUTION

While the ideal block size depends partly on the static file size distribution and the waste incurred by choosing a certain block size, these are not the only factors. For instance, if files larger than 4 KB are hardly ever accessed, it makes no sense to make the blocks larger than 4 KB. In other words, it is also important to consider the dynamic file size distribution: how often are files of a certain size read and written?

In order to answer this question, we logged all NFS traffic to the faculty's NFS servers for four days. The logs contain all read and write operations, but also operations that do not incur disk block transfers to the client and that are therefore given a transfer size of 0 (e.g., create, remove, and setattr operations). The total number of operations in the logs was 4.3 million. Note that these operations do not represent micro measurements, such as NFS packets on the wire. Rather, they represent full operations. For instance, a copy of a 10-MB file from the NFS server to a local disk results in a single entry in a log.

As expected, the logs are dominated by reads and writes: the number of write operations was approximately equivalent to the number of all other operations excluding reads combined (0.54 million) and the number of reads was six times higher even than that (3.2 million). Concentrating on reads and writes, we measured the transfer sizes of each operation and grouped them by decile. The results are shown in Fig. 4. For example, 20% of the reads were for files of length 1965 bytes or less.

Decile	Reads	Writes
10%	889	300
20%	1965	2821
30%	3055	4096
40%	5265	4096
50%	6629	4503
60%	9170	4767
70%	13,477	5511
80%	18,562	6918
90%	33,522	16,281
100%	235,507,712	1,131,274,484

Fig. 4. File size distribution (in bytes) for read and write operations by decile.

The results show that 50% of all read operations and 80% of all write operations are for files smaller than 8 KB. Note that we should be cautious in drawing conclusions about the ideal block size from these figures, as we do not know the exact behavior of the disk controller, the scheduling of requests on the disk, and indeed the transfers between physical disk and the NFS server. However, we do observe that choosing a block size of 8 KB on average results in a single-block transfer for 50% of all read operations and 80% of all write operations. Similarly, a 16 KB block may be expected to serve 80% of all reads and 90% of all writes in a single block transfer.

5. OPERATING SYSTEM CACHE SIZE AND PERFORMANCE

Another point worth considering is the effect of block size on the performance of the operating system's buffer cache. By increasing the block size, we decrease the number of blocks in the cache (assuming a constant amount of memory for the cache). This will reduce the hit rate. To investigate this effect, we constructed a 1-KB file system on a PC, set the operating system's cache size to 0.5 MB, compiled a large collection of programs, and measured the total compilation time. We then repeated the experiment for cache sizes of 1 MB, 2 MB, 4 MB, and 8 MB with block sizes of 1 KB, 2 KB, 4 KB, and 8 KB. The results are presented in Fig. 5. For example, with a 2-MB cache and 4-KB blocks, it took 90 seconds to compile all the test programs. While the first row seems anomalous, we repeated the measurements and kept getting the same numbers. We do not have an obvious explanation. It is probably an artifact of this particular experiment, however.

The effect here is dramatic for small caches. When the cache is too small to hold the various passes of the compiler, the include files, and the libraries, having large blocks (meaning few files in the cache) is a huge performance loss. When the cache gets to 2 MB, everything fits and block size is no longer so important.

This experiment puts a premium on cache space because we kept calling the same compiler passes and used the same standard header files over and over. When the cache was too small to store them, disk activity shot up and performance plunged. A different, less cache-intensive test, might give different results.

Cache (MB)	Block size			
	1 KB	2 KB	4 KB	8 KB
0.5	584	538	539	625
1	119	140	212	399
2	88	87	90	99
4	80	82	81	86
8	78	81	79	83

Fig. 5. Execution time in seconds for various cache and block sizes.

6. DISCUSSION

Somewhat surprisingly, files at a university computer science department are still quite small, although larger than they were in 1984. The most likely reason is that students keep their photo, music, and video files on their home computers, and use the university computers mostly for small programs they are working on for lab courses. The shape of the cumulative distribution of file sizes (shown in Fig. 2) is remarkably similar in the two years measured, 1984 and 2005, suggesting a similar usage overall.

The dynamic measurements also suggest that larger block sizes may help speed up a significant number of operations. For instance, compared to a block size of 4 KB, a 16 KB block would serve 1.5–2 times as many requests in a single disk block. Unfortunately we did not perform the dynamic measurements in our previous study, so we cannot compare the situation in 1984 with the current environment. A study of the Windows NT file system based on dynamic measurements has been presented by Vogels [2]. Interestingly the file sizes seen in the file size distribution for the dynamic measurements are higher than for the static measurements. We speculate that this might be because most reads or writes of small files incur a read of one or more large files first, because most files are accessed using fairly large applications. However, this is just a guess.

This discussion of block sizes does not lead to a mathematical optimum for trading off disk performance against wasted space. Still, the data show that even for large block sizes (16 KB), the total file system expands by only 10%. This suggests that large blocks are a good idea. However, the data of Fig. 5 suggest that even with an 8-MB cache, there is a measureable performance loss with 8-KB blocks, something not observed with 4-KB blocks. In summary, In terms of wasted disk space, block size hardly matters at all, but the cache performance suffers somewhat with 8-KB blocks.

Our tentative conclusion is that probably 4 KB is the sweet spot balancing all the factors. With this size, almost 60% of all disk files fit in a single disk block and the amount of disk space required to store all the files increases only 2% over 1-KB blocks. Considering files actually read, though, only 35% of the files are 4 KB or smaller. Finally, with operating system caches of 4-MB or more, there is no measureable performance difference between 1-KB and 4-KB blocks.

7. REFERENCES

[1] Mullender, S.J., and Tanenbaum, A.S.: ‘‘Immediate Files,’’ *Software—Practice and Experience*, vol. 14, pp. 365-368, April 1984.

[2] Vogels, W.: ‘‘File System Usage in Windows NT 4.0,’’ *Proc. ACM Symp. on Operating System Principles*, ACM, pp. 93-109, 1999.