Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions

Brent Welch, Geoffrey Noer
Panasas, Inc.
Sunnyvale, USA
{welch,gnoer}@panasas.com

Abstract - We studied file size distributions from 65 customer installations and a total of nearly 600 million files. We found that between 25% and 90% of all files are 64 Kbytes or less in size, yet these files account for less than 3% of the capacity in most cases. In extreme cases 5% to 15% of capacity is occupied by small files. We used this information to size the ratio of SSD to HDD capacity on our latest HPC storage system. Our goal is to automatically allocate all of the block-level and file-level metadata, and all of the small files onto SSD, and use the much cheaper HDD storage for large file extents. The unique storage blade architecture of the Panasas system that couples SSD, HDD, processor, memory, and networking into a scalable building block makes this approach very effective. Response time measured by metadata intensive benchmarks is several times better in our systems that couple SSD and HDD. The paper describes the measurement methodology, the results from our customer survey, and the performance benefits of our approach.

Keywords - file systems; OSD; SSD; parallel file system; HPC.

I. INTRODUCTION

Several years ago we developed a simple tool, *fsstats*, that walks a file system and creates histograms of file size and file age information[6]. This has been publically available for some time, and we have been asking some of our larger customers to use the tool to characterize their file systems. Using the information we sought to optimize current and future products. In this study we present data from 13 customers and 65 different file systems. The customers include national labs that have traditional HPC workloads such as physics simulations, Biotech companies that have large amounts of genomic sequencing data and derived data, financial institutions that have historical stock trading data and derived results, and seismic data processing companies that have large amounts of seismic data. While there are differences among the file size distributions, the overall rule of thumb holds across these customers: *most files are small, but most capacity is occupied by large files.* To reconcile this, it is helpful to remember that a 1GB file is 1 million times larger than a 1 KB file. Indeed, some of our customers had individual files that were many TB in capacity.

The Panasas system has a novel per-file data protection scheme that we call Object RAID[11]. In this system, files are broken up into component objects, and the objects contain both file data and parity information that protects those objects from loss or damage. By dispersing the parity groups for a file across a large pool of Object Storage Devices (OSD), and using a cluster of metadata servers to drive reconstruction, we have very fast recovery from failed OSD. The system uses an adaptive RAID scheme where small files (64KB or less) are mirrored in two component objects on different OSD, and larger files use a RAID5 striping pattern that reduces the capacity overhead from the redundant data. Because mirroring has an obviously higher capacity overhead, we wanted to understand just how much overhead this would be. As we introduced SSD into our OSD, we wanted to be sure that we could have a cost effective solution that could place small files onto the SSD region of the OSD, and utilize HDD for the larger component objects.

Using the data from our customers, we found that even with the overhead of mirroring small files, and the metadata overhead associated with each component object, that a system with about 1.5% SSD to HDD capacity could comfortably fit all the small files and metadata onto the SSD. Specifically, we used a 120 GB SSD and two 4 TB HDD inside each OSD for this 1.5% ratio. However, we found some customers with relatively more small files, and some with a very large proportion of small files. Using a 300 GB SSD and two 4 TB HDD we get about 3% SSD to HDD. Using a 480 GB SSD and two 2 TB HDD we get about 10% SSD to HDD. The system gracefully handles the scenario where the SSD is completely full by overflowing into HDD storage.

Our OSD are housed in a blade chassis that has 10 OSD blades and one metadata service blade. Up to 100 chassis can be combined into a single system. Thus the minimum capacity is 1TB SSD and 80 TB HDD, and the maximum capacity is 300 TB SSD and 8PB HDD. The physical system can be logically divided into different file systems that share the OSD with a dynamic and soft partitioning scheme that is part of the OSD standard.

II. RELATED WORK

There have been a few studies of Unix file size distributions dating back to the 1980's[7][8]. These studies reveal an obvious change in file systems that are now storing much, much larger files. While in the early studies a 1MB file was "large", and the average size was around 1KB, today a large file is measured in GB and even TB in capacity. Of course, the change in file sizes reflects the staggering change in technology over the last 30 years since those initial studies.

Some studies were done to create models of file size distributions that could be used in simulation and design of

future systems[4]. Reference [5] highlights the existence of modes (i.e, "bumps") in file size distributions caused by large audio, image, and video files. These studies indicate that there are lots of small files, but most capacity is occupied by larger files.

Baker et. al. presented dynamic measurements of file system operations that look at access patterns and the effectiveness of the Sprite caching model[2]. This work also compares its results with the 1984 study of Unix file systems[8], and notes the growth of average and maximum file sizes. While they did not study the static distribution of file sizes, they found that most accesses during their study period were to small files (10 KB or less), but most of the bytes transferred (90%) were to relatively large files of 1MB or larger.

Tanenbaum studied file size distributions in the university setting and found that most files are small, but most of the space is occupied by large files[10]. In this 2005 study the files were pretty small - 2K average size, with 99% of the files being 1MB or smaller. The largest files in their study was 2 GB, and there were about 1.7 million files in the study. They relate their data to a similar study from 1984[7], and they investigate the effect of having different file system block sizes (e.g., 1KB, 2KB, 4KB, 8KB or 16KB) on capacity.

Argawal et. al studied thousands of user file systems at Microsoft over a 5 year period (2000-2004)[1]. They also found large numbers of relatively small files, with most of the capacity occupied by large files. They suggest techniques such as variable block sizes and co-locating small files with metadata to make small file storage more efficient. They also study file ages and types, and the shape of the directory structure. The file sizes are smaller in general than our study, with 99% of their files being just 16MB or less in capacity. However, image, blobs, and databases become evident in the last years of their study as files with sizes ranging from 1GB to as much as 64GB, and these larger files occupy about 10% of the system in their latest year of study.

The primary difference with our study is that we study large HPC installations with many millions of files and distributions that include files up to TB in size. This is a very different user scenario than the personal workstations or web servers that are involved in most of these other studies. None the less, it is interesting to note that the general results hold. There are still relatively large numbers of relatively small files, yet most of the capacity is occupied by relatively large files.

III. File Size vs. File Capacity

Before we analyze the results of our customer surveys, we need to dig into the details about how a file system uses capacity to store files. File systems allocate in blocks; our system uses a 16KB allocation unit. File systems have overhead from allocation maps, B-tree indexes, and object descriptors (i.e., inodes). When a small file is stored, this block-level metadata overhead will dominate the capacity used by the file. In addition, the Panasas system uses Object RAID for data protection, so a file is divided into multiple component objects that store the data and parity for the file. The parity information is another source of overhead, especially in our system that uses mirroring to protect small files.

We define small files as 64KB or less. This is a special threshold in the Panasas system because files 64K or smaller are mirrored into two component objects. Larger files start using a RAID-5 pattern up to a full stripe width of between 8 and 11 component objects. The stripe width is chosen automatically by the system based on the overall number of OSD that are available. If a file grows beyond 2000 stripes, another set of component objects is allocated (if possible) for a new parity group. Once all possible OSD are used, the existing component objects just grow to accommodate more stripes.

In our system, each component object has an object descriptor that occupies a full file system block. This is different than many Unix file systems that only devote 128 or 256 bytes to an inode, and pack several inodes into a file system block. Because our snapshot facility is block-oriented, it is simpler if an object descriptor occupies a full block so that different snapshots of an object have copies of the object descriptor in different blocks. A 16KB block means a zero-length file occupies 32KB from the two object descriptors on two different OSD. A 1 byte file would occupy 64KB - two object descriptors and two data blocks that mirror the object. You can see that a larger block size has an obvious drawback for the storage of small files.

The first and obvious optimization we did was to pack small file data into the object descriptor. We devote the first 4K of the object descriptor to metadata, and pack the first 12K of the object data into that same block. If objects have more than 4K of metadata, additional full blocks are allocated to store it. In the common case, a zero-length file and a 12K file occupy the same space: 32 K from the replicated object on two OSD. Our data indicates there are quite a large number of very small files below 12K, so this is a simple optimization that is very effective at reducing the capacity overhead for small files.

There is additional overhead from the B-Tree that we use to index object descriptors, and the direct block pointers that we use to track data blocks. However, the object B-Tree occupies less than 1% of the space of the object descriptors (one leaf node references up to 256 object descriptors), and the direct blocks also occupy less than 1% of the data block storage (one direct block references up to 2048 blocks). So, we ignore these effects in the rest of our study. We did comparisons between our projected capacity utilization and the real capacity utilization in our systems, and they were within a few percent.

The second complexity in measuring the capacity used by files comes from the use of mirroring or RAID to protect file data. Files 64K or smaller are mirrored, and larger files use RAID-5. (Newer Panasas systems can use triplication and RAID-6, but the data we studied came from systems that used the classic RAID-1/5 system that is either mirroring or RAID-5.) Thus the overhead for small files is greater than 100%, but becomes much smaller for larger files. In a RAID pattern with 8 data stripe units and 1 parity stripe unit, the overhead is 12.5%. fsstats measures this overhead as the difference between Total User Data and Total Capacity, which is typically about 15% for PanFS file systems.

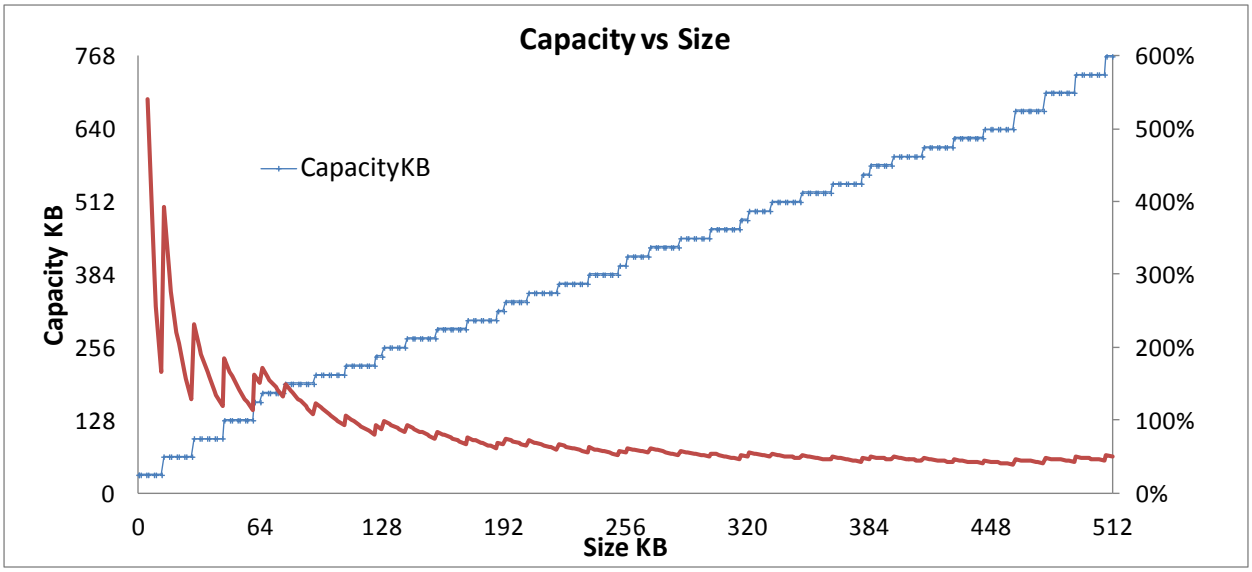

Fig. 1. Capacity used for a file as a function of its size, highlighting smaller files. The red line is the percent overhead (capacity vs size).

Fig. 1 and Fig. 2 plot the capacity used in a PanFS file system as a function of file size, and the overhead of that capacity compared with the file size. The irregular shape of the curves is from the allocation of additional blocks and additional objects with their object descriptor. This data was generated by creating files sized in increments of 4KB plus or minus 1KB, to highlight the stepwise increase in capacity from the 16KB block allocation and the packing of the initial 12KB into the object descriptor. For example, files of 0 bytes to 12KB occupy 32KB because it is mirrored in two objects, and all the data is packed into the object descriptors. After 64KB the system switches from RAID-1 and mirroring to RAID-5, so the steps change from 32KB to a single 16KB block. The anomalies in the steps around 64KB, 128KB, etc. stem from the initial 12KB allocation and the use of a 64KB RAID stripe unit. Fig. 2 shows the data for larger files, and the bumps in the graph occur as a new RAID stripe is added to the file and an extra 64KB is allocated for the parity unit in the new stripe.

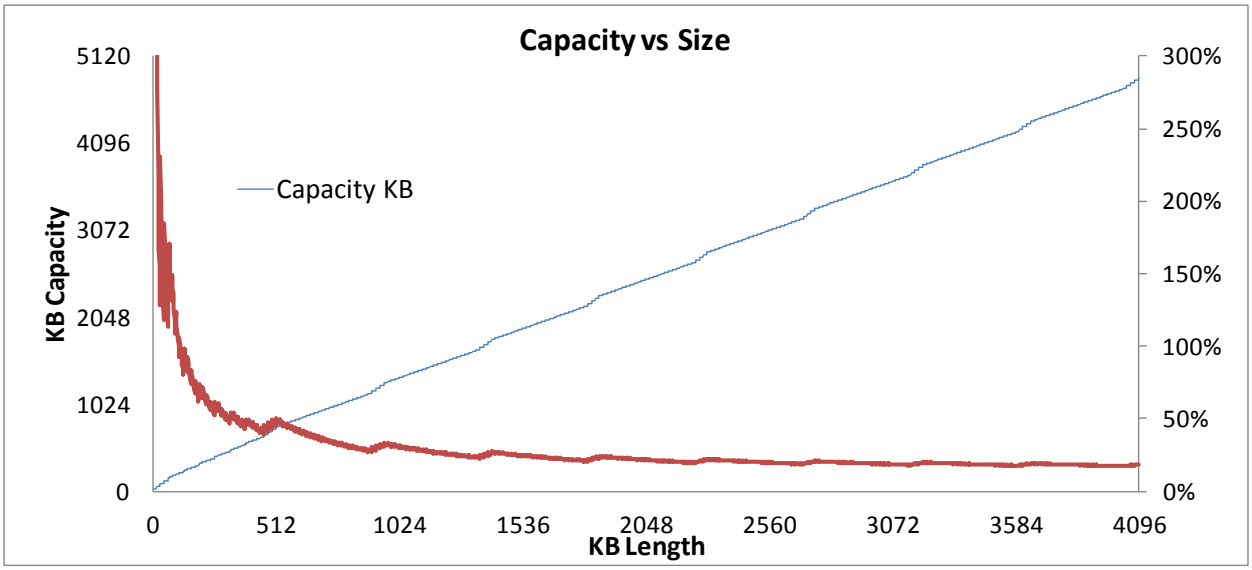

Fig. 2. File capacity used as a function of file size, highlighting files with multiple RAID stripes. The red line is the percent overhead (capacity vs size).

IV. Measurements

The fsstats tool is a Perl script that walks a file system and collects various information about the files and directories in the system[6]. It examines file length, capacity used, directory size, file age, and information about symlinks and hardlinks. For each metric, an average, min, max, and histogram are recorded. The histograms use powers of two buckets. The size buckets start with files less than 2KB, 2KB up to 4KB, 4KB up to 8KB, and so forth. A file exactly 16KB is counted in the 16KB to 32KB bucket. These are base-2 KB, so 1 KB is 1024 bytes, and 1MB is 1048576 bytes, etc. In the figures below, we plot the percentage of files in particular buckets, using the midpoint of the bucket as the X coordinate.

TABLE I. OVERALL CHARACTERISTICS OF THE DATA WE COLLECTED, SORTED BY NUMBER OF FILES IN THE SYSTEM. (PART I)

System	User Data	File Cnt	Avg Len KB	Max Len KB	Directory Count	Directory Entry		Name	
						Avg	Max	Avg	Max
Uni-5	433.60 TB	139,440,887	3,339	562,401,886	7,450,817	20	100,000	25	254
Uni-4	78.68 TB	59,550,373	1,418	908,517,159	6,780,002	10	3,188,509	26	255
Wth-1	103.73 TB	46,095,337	2,416	1,930,925,561	2,300,074	21	350,000	24	179
LANL-7	382.60 TB	43,237,353	9,501	2,798,494,990	359,023	120	349,058	11	133
LANL-5	313.11 TB	32,057,311	10,487	3,836,928,000	415,310	78	100,000	15	87
Uni-2	101.07 TB	22,324,039	4,861	183,997,492	1,279,045	19	263,622	22	207
Uni-1	64.59 TB	17,729,082	3,912	607,034,885	537,922	35	79,356	26	214
Sei-1	5.30 TB	15,664,895	363	113,736,330	1,364,076	13	64,311	15	173
LANL-6	192.79 TB	15,284,519	13,543	1,337,720,832	142,940	107	81,611	12	211
Fin-9	24.43 TB	12,565,235	2,087	55,334,804	689,640	20	191,419	24	167
Uni-3	39.44 TB	12,186,131	3,475	177,724,290	39,977	306	218,670	24	153
Fin-14	16.06 TB	11,754,641	1,466	66,018,402	94,508	134	350,000	26	49
Sei-11	1508.62 TB	10,469,009	154,729	4,799,672,727	601,450	18	62,925	20	159
User-3	820.30 GB	9,779,854	87	61,829,588	2,465,991	5	50,048	6	75
Man-1	23.22 TB	9,497,470	2,625	161,005,560	320,836	31	38,348	18	168
Bio-4	681.19 GB	9,464,391	75	12,896	10,132	935	1,713	16	24
LANL-2	286.34 TB	8,073,268	38,082	3,111,873,960	203,040	45	34,231	19	83
Fin-8	3.62 TB	7,615,883	511	27,040,766	40,947	187	46,272	26	180
Bio-5	5.49 TB	6,943,839	849	79,282,357	647,381	12	144,947	19	113
Sei-12	913.46 TB	5,742,360	170,804	7,021,072,265	144,602	41	51,154	17	121
Fin-22	24.03 TB	5,575,157	4,628	13,094,985	23,713	236	38,273	28	82
Sei-8	633.15 TB	5,361,453	126,800	21,453,090,816	18,904	285	2,342,911	19	150
LANL-9	28.35 TB	4,972,722	5,427	266,708,130	721,281	10	81,716	18	123
LANL-8	43.97 TB	4,618,230	10,222	89,600,128	296,183	16	85,585	18	133
Wth-2	15.34 TB	4,566,684	3,608	71,947,710	303,017	17	66,123	16	179
LANL-1	24.92 TB	4,403,273	6,076	298,936,790	62,430	73	30,153	16	69
Fin-19	7.69 TB	4,142,457	1,993	18,567,876	222,619	20	823	34	50
Fin-18	7.69 TB	4,135,559	1,996	1,539,463	223,746	19	872	34	50
Fin-17	7.60 TB	4,121,129	1,980	652,271	221,118	20	1,520	34	50
Fin-16	7.71 TB	4,113,341	2,012	2,462,291	222,748	19	922	34	50
User-2	932.54 GB	4,060,880	240	11,713,329	389,618	12	18,831	15	246
Fin-21	7.53 TB	4,050,959	1,997	16,994,257	220,351	19	2,925	34	50
Fin-12	7.45 TB	4,048,816	1,976	1,233,843	218,638	20	824	34	50
Fin-20	7.51 TB	4,042,590	1,993	479,283	219,610	19	1,147	34	50
Sei-5	546.95 TB	3,723,428	157,727	4,354,540,499	14,756	254	87,631	21	94
Sei-9	954.91 TB	3,619,943	283,243	5,670,650,134	30,176	121	119,528	20	202
Fin-10	28.25 TB	3,386,744	8,955	55,334,804	154,333	23	83,777	27	136

TABLE II. OVERALL CHARACTERISTICS OF THE DATA WE COLLECTED, SORTED BY NUMBER OF FILES IN THE SYSTEM (PART II)

Tag	User Data	File Cnt	Avg Len KB	Max Len KB	Dir Cnt	Directory Entry		Name	
						Avg	Max	Avg	Max
Sei-10	1272.13 TB	3,028,598	451,008	8,645,563,520	137,758	23	51,016	17	129
Sei-7	622.27 TB	2,900,395	230,368	7,517,228,594	16,800	174	50,864	22	118
LANL-4	14.02 TB	2,209,335	6,811	348,372,738	490,399	7	29,736	17	95
Sei-4	786.84 TB	2,141,093	394,593	4,612,501,768	21,016	103	99,636	21	159
LANL-3	25.58 TB	1,873,912	14,659	150,802,080	144,427	13	10,269	15	80
Fin-4	1.07 TB	1,863,089	619	9,710,116	125,345	16	18,268	22	125
Fin-7	665.13 GB	1,703,488	409	2,421,407	442	3,855	15,541	19	26
Sei-6	1210.78 TB	1,678,245	774,660	5,351,220,703	56,197	31	209,769	22	161
User-1	413.19 GB	1,170,349	370	20,033,536	75,510	17	11,634	18	112
Fin-5	670.82 GB	1,163,843	604	9,713,498	75,000	17	17,589	20	119
Sei-2	471.65 TB	1,136,579	445,572	2,375,836,562	25,547	46	129,143	23	161
Fin-2	227.48 GB	1,051,603	227	3,673,885	34,074	32	20,801	15	68
Sei-14	114.42 TB	899,342	136,606	206,277,071	244	3,687	39,050	58	94
Bio-1	4.96 TB	894,546	5,953	8,667,102	2,555	351	3,849	12	43
Bio-3	7.07 TB	838,504	9,053	61,992,400	8,569	99	95,513	33	82
Sei-16	69.80 TB	791,014	94,748	17,560,600	76	10,409	40,574	59	78
Fin-13	1.73 TB	701,426	2,641	620,472	15,253	47	28,552	15	50
Fin-6	103.99 GB	697,384	156	23,190	841	830	4,375	21	26
Sei-3	781.15 TB	697,119	1,203,178	85,308,625,086	15,531	46	98,497	15	96
Fin-15	15.94 TB	348,471	49,123	5,472,113	565	618	42,682	31	62
Sei-17	90.91 TB	261,316	373,555	256,822,213	221	1,183	109,509	54	104
Sei-18	77.43 TB	154,341	538,656	152,412,961	237	652	19,968	42	104
Fin-1	2.19 TB	129,480	18,168	4,157,022	1,038	126	7,990	29	100
Fin-3	147.87 GB	70,887	2,187	3,616,690	3,413	22	10,150	25	125
Fin-11	1.20 TB	67,626	19,006	6,665,833	287	237	17,028	30	52
Sei-13	1.34 TB	33,145	43,304	146,032,041	373	90	7,991	31	104
Sei-15	6.90 TB	3,684	2,011,364	50,938,445	39	95	798	27	54
Bio-2	6.88 TB	449	16,452,237	428,096,320	29	16	24	14	26

TABLE I. and TABLE II. summarize the datasets we collected. There are 65 datasets from 13 different customers. These are loosely categorized into University (Uni), User files (User), Weather forecasting (Wth), Seismic data processing (Sei), Financial modeling (Fin), Biotech (Bio), and Manufacturing (Man). The LANL datasets have been previously studied[3]. The number of files counted for each system ranged from 139 million to just a few hundred, and the amount of data in each system ranged from over 1500 TB to under 1TB.

A. General Observations

The seismic data sets have large files, with an average size ranging from 43MB to over 2000MB. These systems have very large individual files, with one system having an 85 TB file. However, small files are still common. The Sei-1 system had its largest file at 113 GB, but had over 6 million files in the zero to 2K length bucket.

The rest of the systems have an average file size is in the small number of MB in most cases. There are a few systems with less than 1MB average file size. Ignoring the obvious outlier in the last row (Bio-2), the averages of the non-seismic systems range from 75 KB to 49 MB.

Nearly all systems have very large files. Only 5 systems had a maximum file size less than 1GB. 16 systems measured their largest file in TB, including most of the seismic systems, a weather processing system, and some of the LANL systems.

Directories are large. While many systems have an average directory size that is under 30 entries, several systems averaged hundreds of files per directory. The largest directory was over 3 million files, and a maximum directory size over 100,000 was common.

Systems with mostly small files are relatively small in total capacity. While most systems in our study were many TB in total capacity, those dominated by small files were usually less than 1TB in size. In most cases, a customer that had a file system like this also had file systems dominated by large files. The User-1, User-2, and User-3 systems were from a seismic customer, for example. This suggests that sharing OSD among these different kinds of systems (small files vs. large files) would allow averaging out the metadata workload associated with having large numbers of files in a file system. This kind of sharing is commonly used in PanFS systems where a large physical pool of OSD is shared among different file systems.

There is one system (Bio-2) that has a small number of very large files (16GB average, 428 GB maximum). This system has only 449 files and occupies almost 7 TB. We assume this system is used as a backup target and stores large tarfile images.

The averages and maximums in Table 1 hide the nuances and differences among the systems. One fact seems clear: there are many different file size distributions among HPC users. The remaining sections look at the file size distributions in more detail.

B. Small Files

Small files post challenges to file systems. The ratio of metadata to data is high. Allocation strategies for small files may be different than for large files to increase storage efficiency and eliminate seeks. We are particularly interested in files 64KB and smaller because of our RAID striping strategy. Unfortunately, the fsstats histograms lump 64KB files with files up to 128KB, so we cannot directly measure how much capacity is occupied by 64KB and smaller files.

Fig. 3 shows the percentage of files less than 64KB for each of the systems we measured. The percentage ranges from about 25% to over 90%, although there are a few systems that had almost no small files. For example, Bio-1, had 65% of its files in the 1-4 MB buckets, and less than 1% of its files in the smallest buckets. It is associated with a genomic sequencing instrument.

The primary reason there are lots of small files is that it is just easy for users to have every thread in a parallel job open its one file (or files) and generate its data. While there are special libraries like HDF5 and netCDF that allow sophisticated shared output files so a parallel job can generate just a single file, these are not always widely used even in very mature HPC environments. For example, in the DOE tri-lab community of LLNL, Sandia, and LANL, only the LANL applications are big users of shared output files. The other communities have developed a usage model of having many files per job. Genomic sequencing information has a similar pattern. The instruments generate a large stream of relatively small files that each represent relative small data sample. It is common to see each genomic run create a single directory with large numbers of small files in it.

The fsstats program records the capacity used attribute (i.e., "ls -s"). In the case of PanFS, this attribute counts blocks used for object descriptors and for redundant objects, which means mirroring for these small files. Fig. 4 shows the capacity occupied by files that use less than 256KB. Because a file that is 64KB in length occupies 160KB in our system, we must use this bucket to approximate the capacity used by our small files. The results are an overestimation. These systems were measured before the introduction of our latest version that packs 12KB of data into the block used for the object descriptor. In other words, the cumulative capacity up to 128KB would underestimate the capacity of files less than 64KB, and the cumulative capacity up to 256KB overestimates the capacity used by files with length 64KB or smaller.

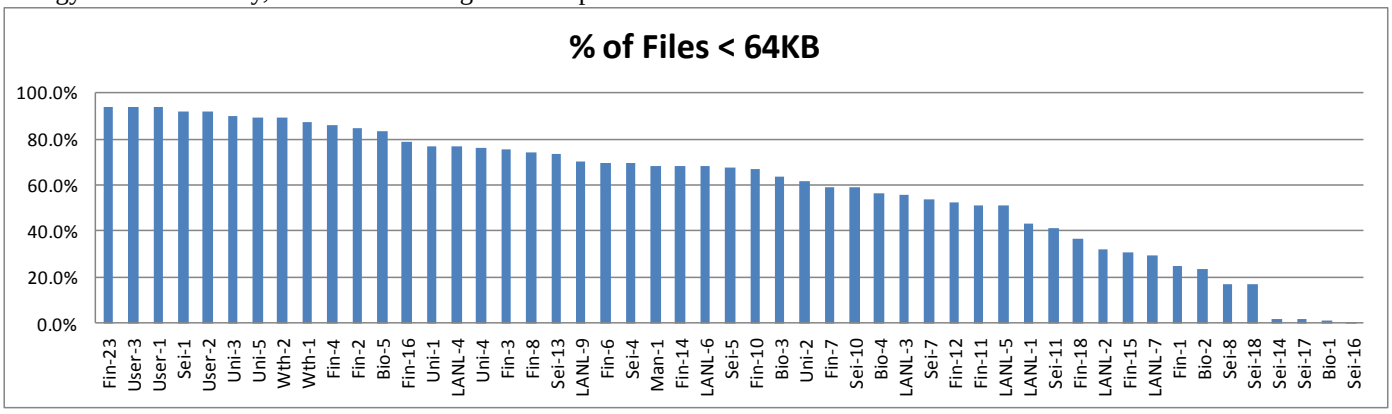

Fig. 3. The percentage of files of files less than 64KB.

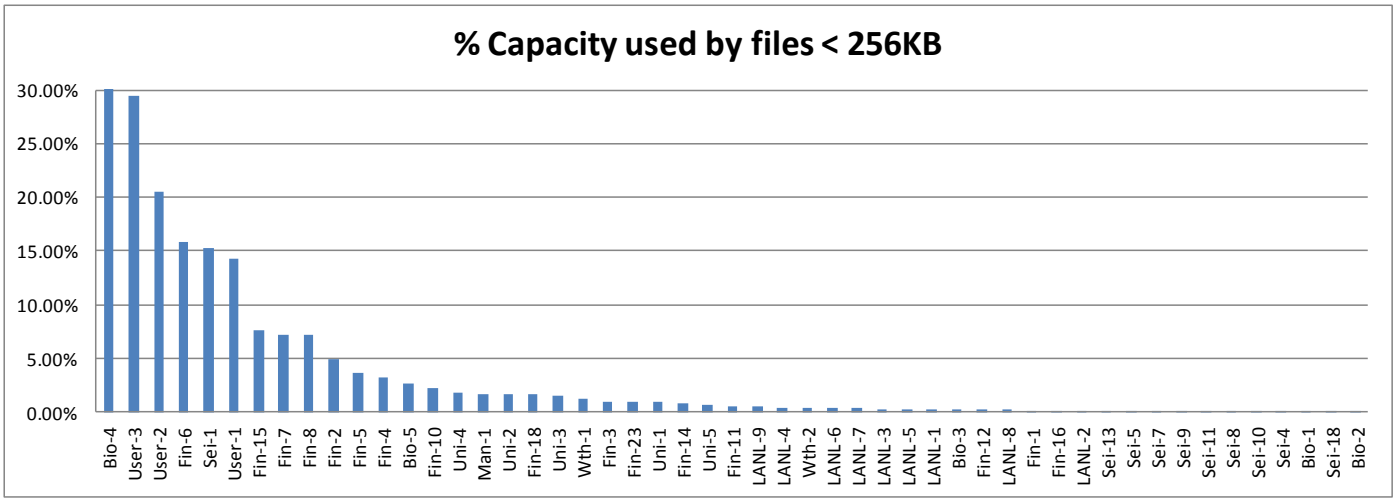

Fig. 4. The Capacity used by files of length less than 256KB. These measures include overhead from block and object allocation. The Bio-4 value is 75%.

Most systems have less than 2-3% of their capacity occupied by small files. The system with the largest percentage of capacity devoted to small files (75%) is Bio-4. This is a relatively small file system (681 GB) in terms of capacity, but it has over 9 million files. It also has the smallest average file size (75KB), and the smallest maximum file size (12MB). User-3 is used for log files, User-2 is home directories, and User-3 is shared binaries. They are relatively small, each being less than 1TB in capacity. Fin-6 is very small. Sei-1 has a different profile. It is larger with over 5TB in capacity, with some large files, but a large number (6 million) of tiny files, which are likely to be small job log files.

There is a cluster of systems in the finance sector with between 3% and 8% of their capacity in small files. These systems have "tick data" that represent fine grain resolutions of stock trade activity in individual files. This information comes from data feeds and is just stored in raw format in the file system.

C. *File Size Distributions*

The best insights come from plots of file size distributions. In this section we plot the file counts as a percentage of total for each histogram bucket, and file capacity as a cumulative percentage of total for each histogram bucket. In all graphs, we plot the midpoint of the histogram on the X axis. Smooth curves are used as a reminder that we don't have precise file sizes. The first two figures have four curves from each of two datasets (Fin-7 and LANL-7):

- File Count indicates how many files fell into this histogram bucket for file size. This comes from the *file size* histogram of fsstats.
- File Capacity indicates how much capacity was used by the files in this histogram bucket of file capacity. This comes from the *file capacity* histogram of fsstats.
- File Count Cumulative indicates how many files were this size or smaller.
- File Capacity Cumulative indicates how much capacity was used by files with this capacity or smaller.

It is important to note that the X axis is logarithmic. The sizes and capacity of the files toward the right of the graphs are a million times bigger, or more, than those on the left side of the graphs.

Fig. 5 plots data from a customer in the Finance sector (Fin-7 dataset) that had about 1.7 million files. The blue curve shows the counts of files at various sizes. The first point at 1K counts files between 0 bytes and 2K, and 20% of all files fall into this bucket. The red curve shows the capacity used by files in the various buckets. This curve starts at 24K, which is the mid-point of the 16K to 32K bucket. The smallest files land in this bucket due to block, object and parity overhead. The curve grows gradually to a hump around 1MB and shows another bump beyond 1GB. The green curve shows the cumulative file count; 60% of the files are 64K or less. However, the purple cumulative capacity curve indicates that these files occupy about 7% of the capacity. These capacity curves take into account the overhead illustrated in Figure 1 and 2. The cumulative capacity ramps up around the 1MB to 4MB file size, with 75% of the capacity occupied by files 16MB or less. The bump at the 1GB to 4GB file size range comes from less than 100 files in this capacity range, and they account for over 10% of the capacity.

Fig. 6 plots the same family of curves for the LANL-7 dataset. This is the largest LANL system with over 43 million files. There are interesting modes at 64KB,1MB, 4MB, 1GB, and 1TB which represent collections of similar sized files. While most files are less than 1MB, the capacity is dominated by files 1GB and larger. The modes (i.e., "bumps") result from applications that generate large numbers of similar sized files. This becomes more evident when looking at all the file size distributions.

Fig. 7 through Fig. 16 show the file size and cumulative capacity curves for all the datasets. The curves are grouped into roughly similar datasets. The non-cumulative file count figures highlight the modes of files in different size bands. The

cumulative capacity figures highlight how large files dominate the capacity used in these systems.

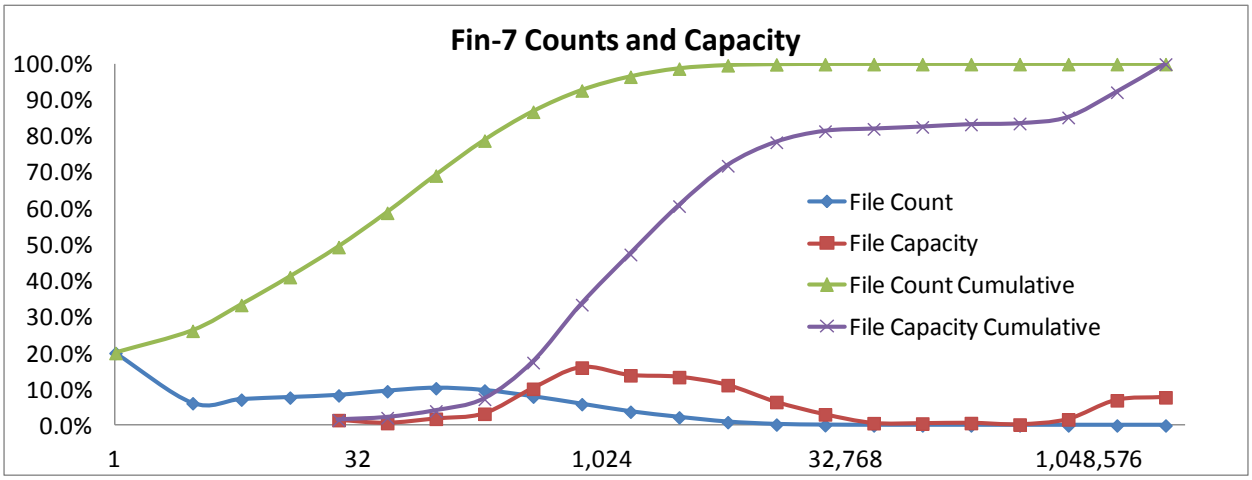

Fig. 5. Cumulative file counts and capacity for the Fin-7 dataset from a customer in the Finance sector.

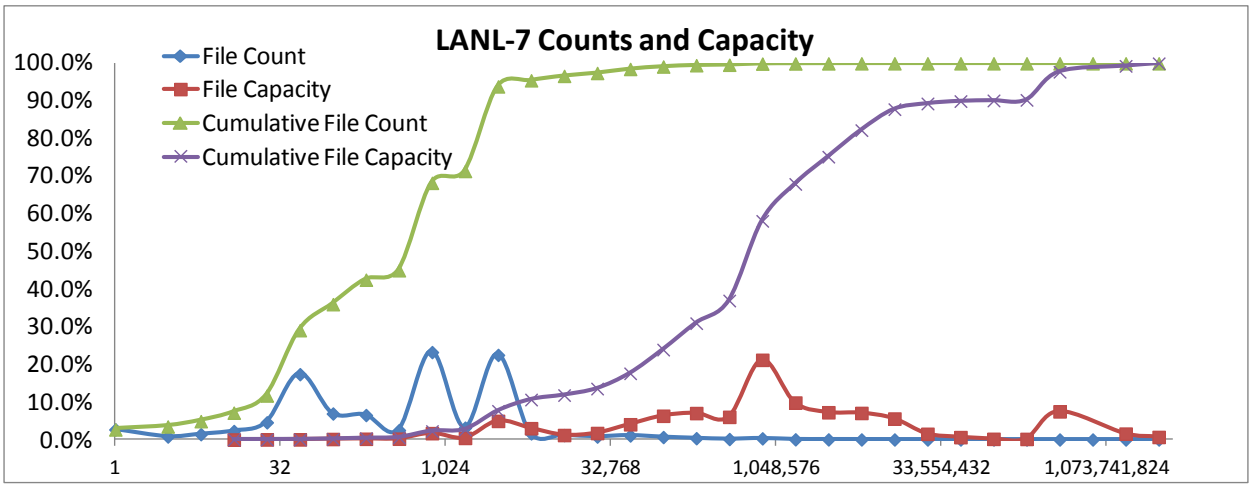

Fig. 6. Cumulative file counts and capacity for the largest LANL dataset, LANL-7.

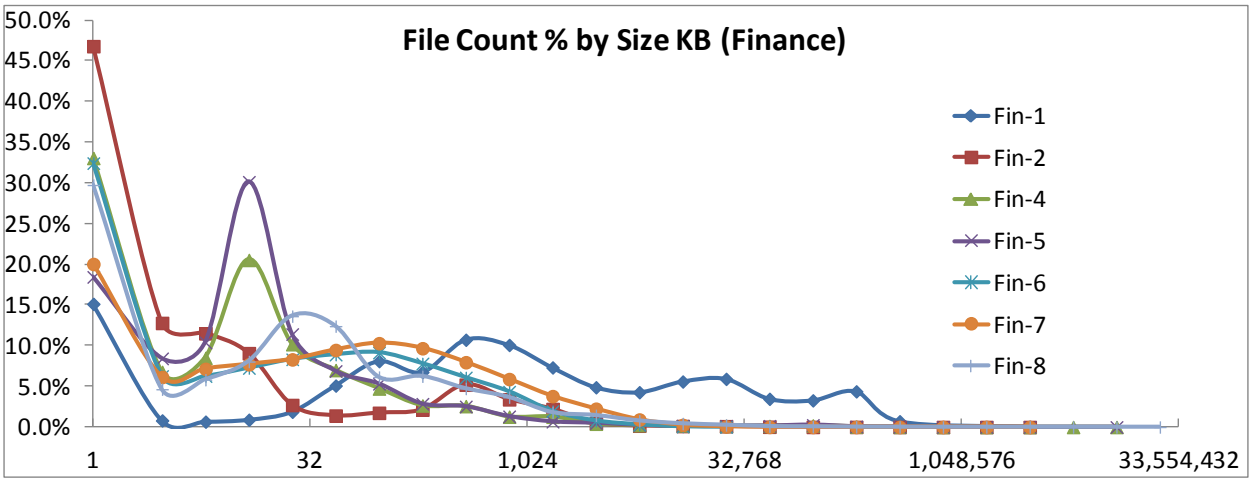

Fig. 7. File Counts for the Finance datasets.

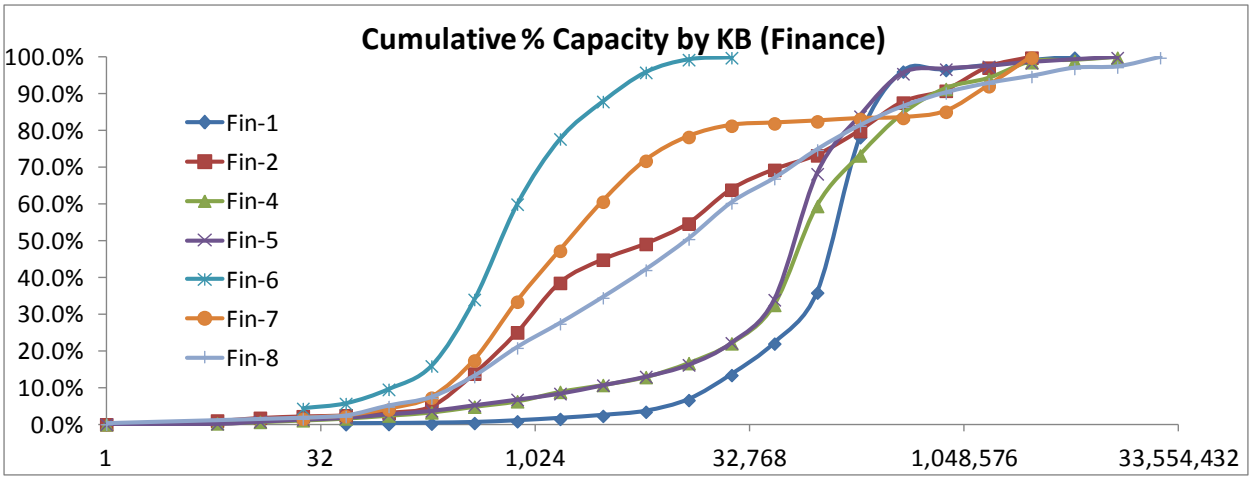

Fig. 8. Cumulative capacity for the Finance datasets.

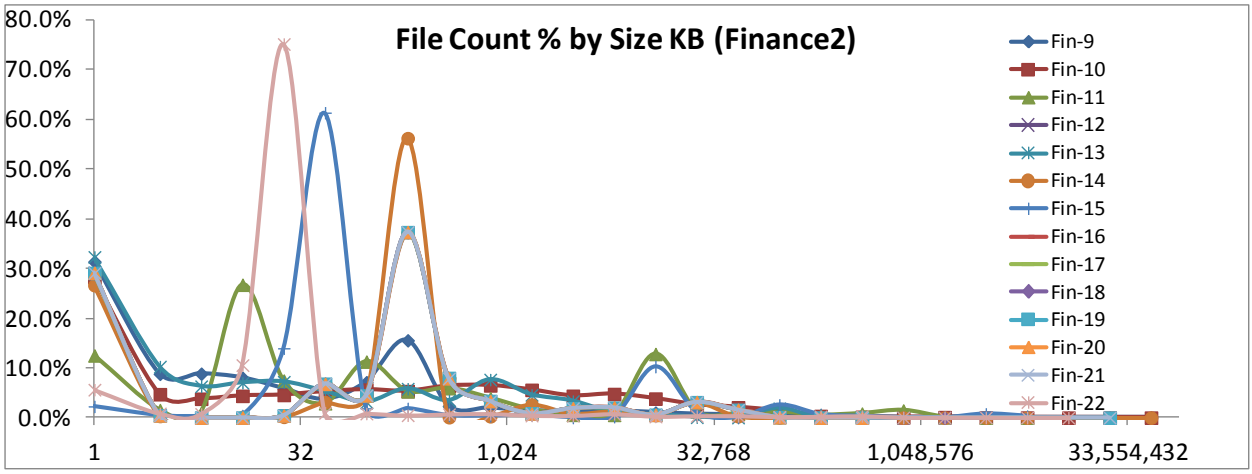

Fig. 9. File counts for the second set of Finance datasets. Fin-15 through Fin-21 have almost identical distributions.

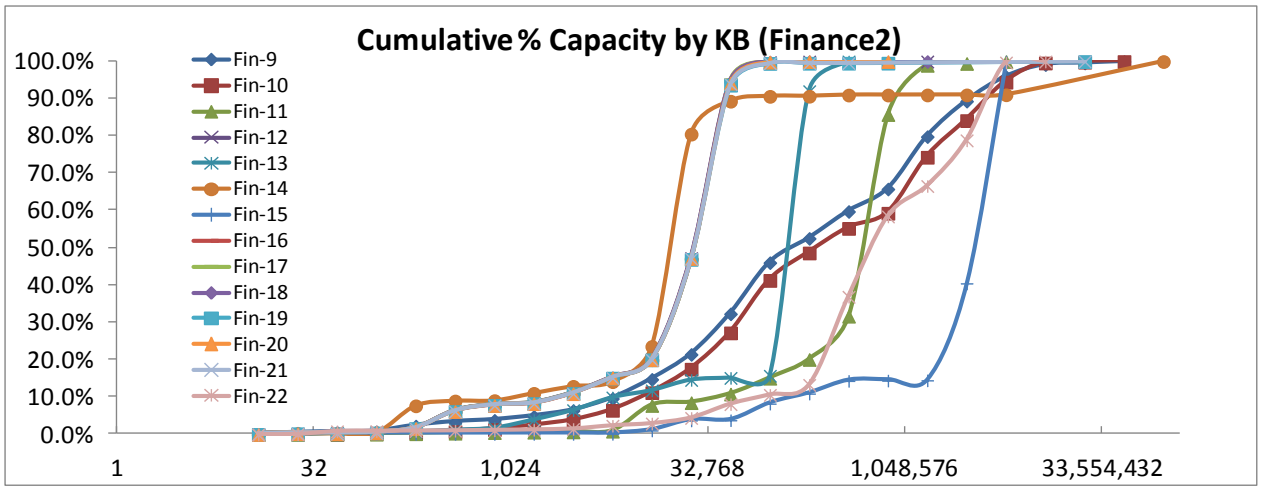

Fig. 10. Cumulative capacity for the second set of Finance datasets. Fin-15 through Fin-21 have almost identical distributions.

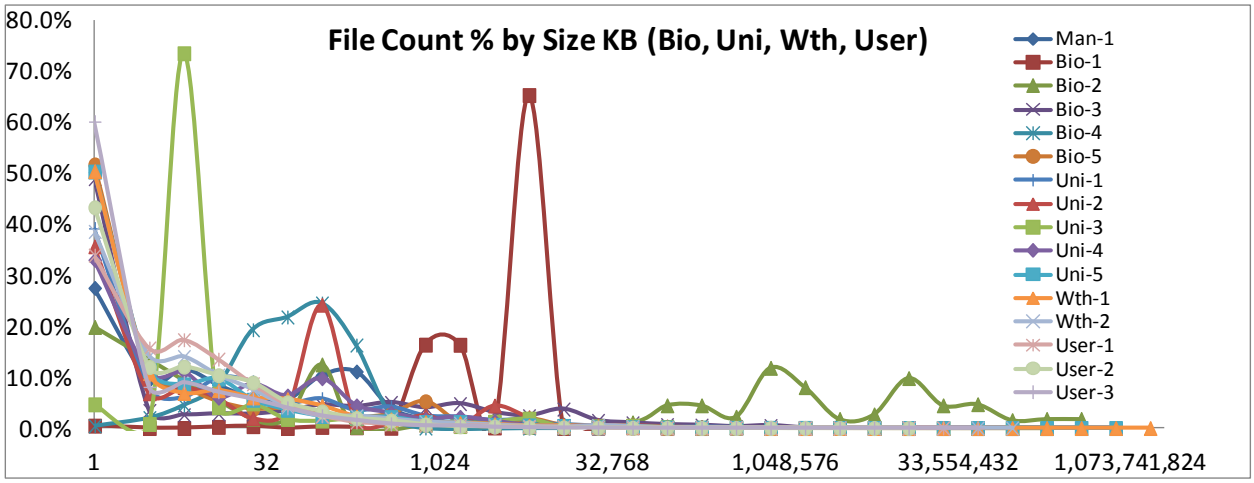

Fig. 11. File counts for the University, Bio, Weather, and User datasets.

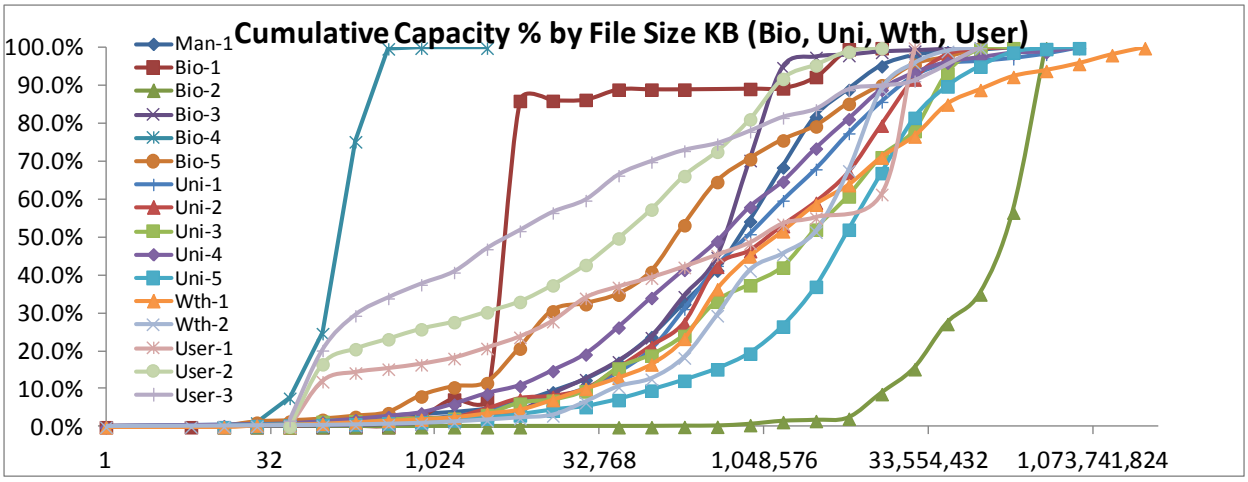

Fig. 12. Cumulative capacity for the University, Bio, Weather, and User datasets.

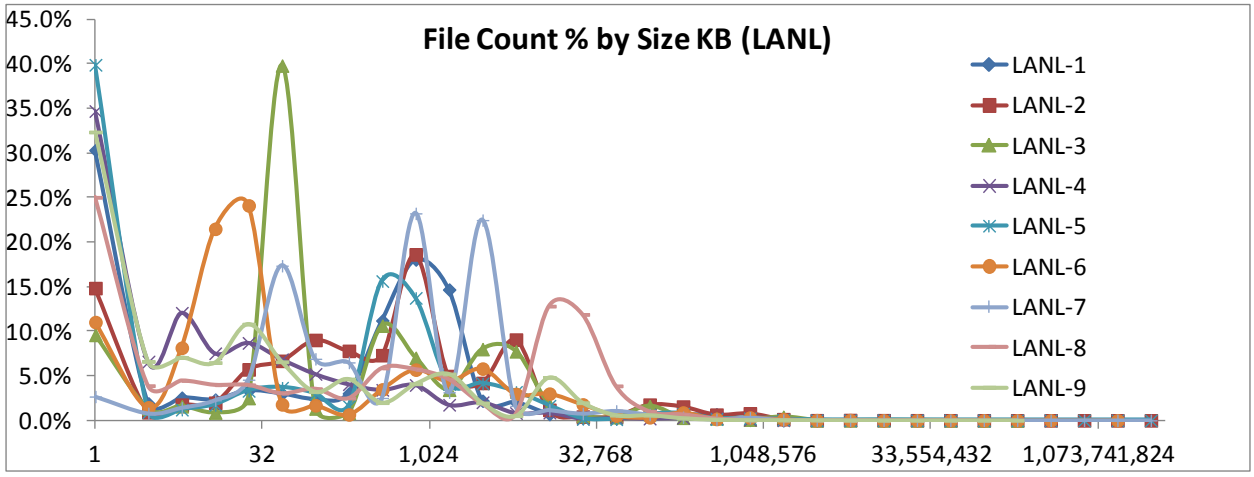

Fig. 13. File counts for the LANL datasets.

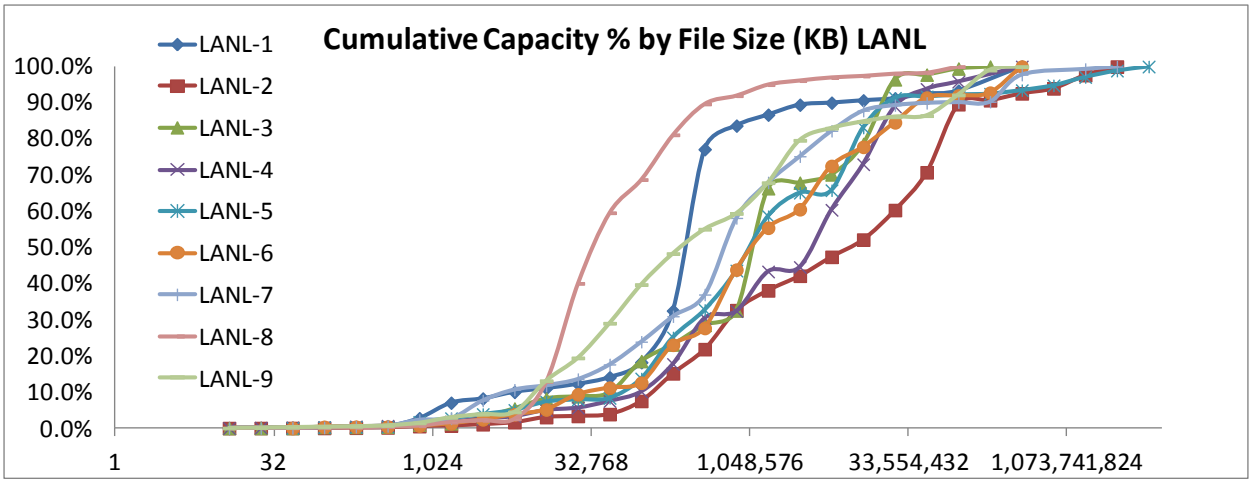

Fig. 14. Cumulative capacity for the LANL datasets.

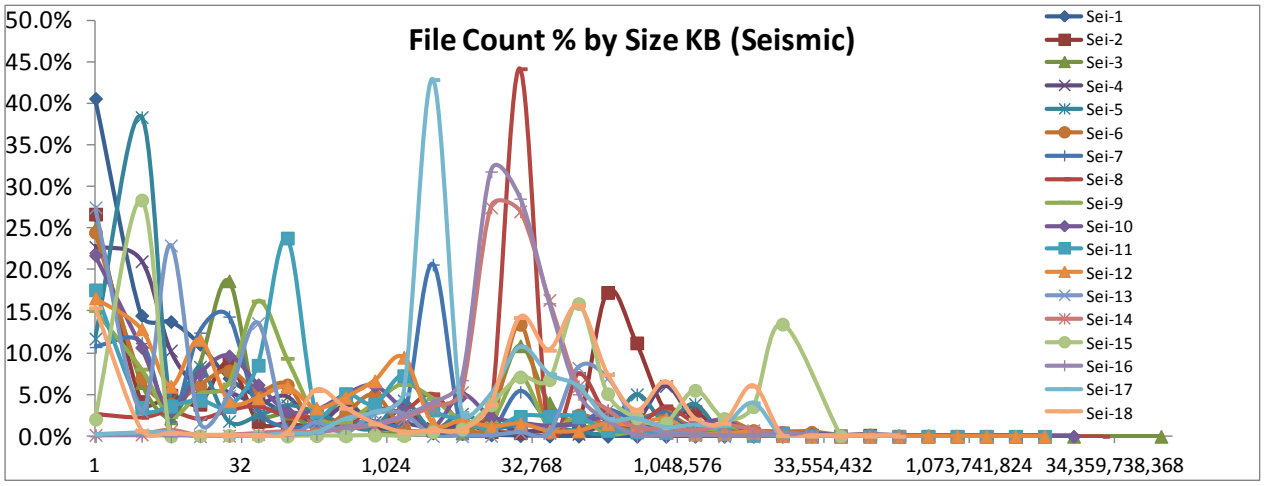

Fig. 15. File counts for the Seismic datasets.

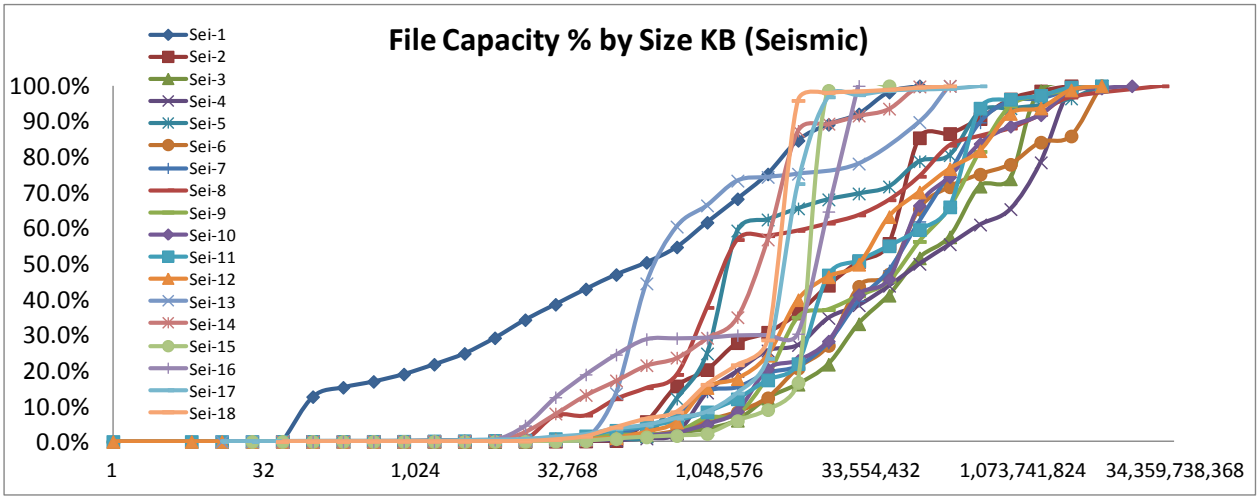

Fig. 16. Cumulative capacity for the Seismic datasets.

Even though the distributions vary, nearly every system has a significant number of files in the smallest size bucket. The smallest size bucket accounted for 15% to 50% of the files in most cases. However, there are also distinct peaks at other capacity points, which indicate that these systems are dominated by applications with specific file sizes. These peaks often account for 50% or more of the files within the entire file system.

Capacity is dominated by large files. Only a few systems had a significant percentage of their capacity in files below 1 MB, and these were relatively small systems (under 1TB total capacity).

V. PERFORMANCE

We used the amount of small files and metadata to size the SSDs in our AS-14 hybrid storage blade. This OSD has 8 GB DRAM, a 120 GB SSD and two 4TB SATA drives to create an OSD that has 1.5% of its capacity in SSD storage. A model with a 300 GB SSD has 3% SSD. For extreme performance the AS-14T has 16GB DRAM, a 480 GB SSD and two 2TB SATA drives, so 10% of the capacity is SSD. We compare the performance of these blades to the AS-12 that has 8GB DRAM and two 2 TB SATA drives. These ratios of 1.5%, 3%, and 10% cover most of the different file capacity distributions for small files that we observed in our survey.

We used metadata intensive benchmarks to evaluate the performance of adding SSD to the OSD; the performance of large file bandwidth is very similar between the two configurations. Our blade chassis with 10 OSD storage blades and dual 10GE uplinks delivers sustained bandwidth from disk to a collection of clients at about 1600 MB/sec, or 160 MB/sec/OSD. Table II gives a brief summary of the performance impact of adding SSD to the OSD.

TABLE III. PERFORMANCE IMPACT OF ADDING SSD TO THE AS-14 STORAGE BLADE.

Benchmark	AS-12	AS-14T	Speed Up
SFS 2008	9739 ops/s 9.3 ms RT	20745 ops/s 7.6 ms RT	2.1
20M FSRC	29336 sec	3638 sec	8.1
ls -l 1M	1236 sec	366 sec	3.4
60K reads	1129 ops/sec	8882 ops/sec	7.9

SpecFS 2008 is the industry standard benchmark[9]. Higher ops/sec and lower response time is better, and the performance more than doubles. The FSRC benchmark is an offline file system check of a volume with 20 million files, and this runs 8 times faster because it is a pure metadata workload. The listing of a million file directory is more than 3 times faster. We use prefetching of object attributes to gain some parallelism out of a serial workload. The read workload is a cold-cache workload that reads small files, and it is nearly 8 times faster. Except for SFS, which has a cache warming phase, these benchmarks were run with a cold cache to force disk or SSD accesses.

VI. CONCLUSION

This paper has looked at file size distributions in HPC systems measured with the fsstats tool. Unlike earlier studies that focused on personal workstations, we have measured very large HPC systems that feature individual files over 1TB in size. The 65 systems we surveyed are each different. The modes in file size distribution reflect the different applications run against the systems. There were extremes such as more than 9 million files all less than 12 MB and a total capacity less that 1TB, and a system with less than 500 files and almost 8TB in total capacity with an average file size of 16 GB. In most cases, however, average file sizes were a few MB, and maximum file sizes were measured in GB. The LANL and seismic systems had maximum file sizes in the TB, and average sizes in the 100's of MB.

We found the traditional pattern of lots of small files, with the capacity dominated by very large files. Based on these measurements, we sized the ratio of SSD to HDD in our latest OSD storage blade. SSD is used to optimize small files and metadata storage. We introduced data packing to put small file data into object descriptors so we optimize the use of SSD. Our measurements indicate that most systems can easily afford to put every object descriptor, index, allocation maps, and all small files onto the SSD with a configuration that has 1.5% to 3% of its capacity in SSD. These configurations allow us to add SSD in a cost effective manner, and yield significant performance results for metadata intensive workloads.

VII. REFERENCES

[1] Agrawal, N., Bolosky, W. J., Douceur, J. R., & Lorch, J. R. (2007). A five-year study of file-system metadata. ACM Transactions on Storage (TOS), 3(3), 9.

[2] Baker, M. G., Hartman, J. H., Kupfer, M. D., Shirriff, K. W., & Ousterhout, J. K. (1991, September). Measurements of a distributed file system. In ACM SIGOPS Operating Systems Review (Vol. 25, No. 5, pp. 198-212). ACM.

[3] Dayal, S. (2008). Characterizing HEC storage systems at rest. Parallel Data Lab, Carnegie Mellon University, Pittsburgh, PA, USA. CMU-PDL-08-109

[4] Downey, A. B. (2001). The structural cause of file size distributions. InModeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on (pp. 361-370). IEEE.

[5] Evans, K. M., & Kuenning, G. H. (2002, July). A study of irregularities in file-size distributions. In Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS).

[6] fsstats, http://www.pdsi-scidac.org/fsstats/

[7] Mullender, S. J., & Tanenbaum, A. S. (1984). Immediate files. Software: Practice and Experience, 14(4), 365-368.

[8] Ousterhout, J. K., Da Costa, H., Harrison, D., Kunze, J. A., Kupfer, M., & Thompson, J. G. (1985). A trace-driven analysis of the UNIX 4.2 BSD file system (Vol. 19, No. 5, pp. 15-24). ACM.

[9] SFS2008. www.spec.org/sfs2008/

[10] Tanenbaum, A. S., Herder, J. N., & Bos, H. (2006). File size distribution on unix systems-then and now. Operating systems review, 40(1), 100.

[11] Welch, B., Unangst, M., Abbasi, Z., Gibson, G., Mueller, B., Small, J., ... & Zhou, B. (2008, February). Scalable performance of the Panasas parallel file system. In *FAST* (Vol. 8, pp. 1-17).