# Characterizing HEC Storage Systems at Rest

Shobhit Dayal

CMU-PDL-08-109
July 2008

**Parallel Data Laboratory**
Carnegie Mellon University
Pittsburgh, PA 15213-3890

**Acknowledgements:** I'd like to thank my advisor Dr. Garth Gibson for working closely with me through this study and providing invaluable guidance. I wish to acknowledge James Nunez from LANL for providing timely comments and feedback on the report and helping gather data from LANL over several iterations, and Akbar Mokhtarani from NERSC and Evan Felix from PNNL for providing statistics from repeated runs over months. This report will be revised into a paper for future submission to a conference and James Nunez, Akbar Mokhtarani and Evan Felix will be coauthors on it. I also wish to thank the following people who ran fsstats on their respective sites and uploaded results to us: Brad Havel from ARSC, Michael Stroucken from CMU, Alfred Torrez and Gary Grider from LANL, Katherine Vargo from PSC and Philip Roth from ORNL, and the members and companies of the CyLab Corporate Partners and the PDL Consortium (including APC, Cisco, EMC, Google, Hewlett-Packard, Hitachi, IBM, Intel, LSI, Microsoft, Network Appliance, Oracle, Seagate, Symantec, and VMware) for their interest, insights, feedback, and support. This material is sponsored in part by the Department of Energy, under Award Number DE-FC02-06ER25767.

**Keywords:** filesystems, storage, HPC, HEC, statistics

## Abstract

*High-performance parallel file systems are a critical component of the largest computer systems, are primarily proprietary, and are specialized to high end computing systems that have many access patterns known to be unusual in enterprise and productivity workplaces. Yet little knowledge of even the basic distributions of file systems and file ages are publicly available, even though significant effort and importance is increasingly associated with small files, for example. In this paper we report on the statistics of supercomputing file systems at rest from a variety of national resource computing sites, contrast these to studies of the 80s and 90s of academic and software development campuses and observe the most interesting characteristics in this novel data.*

## 1 Introduction

In 2006 we started a project at CMU whose goal was to make available tools and services that facilitate worldwide data collection on static file tree attributes and aggregate this data into a large database that can be queried and viewed by anyone. In the past, people have collected data on how files within file systems change in terms of file size, access time, modification time, filename length and various other attributes [17, 13, 15, 4, 11, 19, 6, 16, 7, 20, 3, 2]. Our goal is for users to be able to gather this data for themselves and to facilitate sharing of this data. In the process we collected statistics from several file systems at various national laboratories and HPC sites, and some very large file systems at CMU.

In this paper we present statistics collected from 13 file systems from five supercomputing sites and from 2 file system at a local file server in our department. We compare and contrast this data set with previous studies and provide some insight to where data at HPC sites are different and where they are similar to workstation data. A challenge in collecting these statistics has been that since these file systems are large, data collection tools may take a very long time to complete and disrupt normal activity at the site. It is also difficult to write one tool that may work at any supercomputing site. Often file system metadata in high performance file systems are stored in Databases, and out of band querying of the data base is much more efficient than using the normal POSIX interface to the file systems. Finally the system administrator may already have the metadata we need in a flat file, that is also used for backup, and it would be less disruptive for the site to run a tool against the flat file than the live file system. For this reason some of our collaborators chose to use their own methods and tools in conjunction with our tool than just running our tool in a straightforward manner.

We collected statistics on file size, capacity used, directory size, overhead, symbolic links, hard links, access, modification and change time and filename length. Here we don't present analysis and data for symbolic links, hardlinks and filename length and concentrate on the remaining properties of files that are more interesting.

Section 2 introduces our graphs and how to read them. Section 3 describes our tool used to collect statistics and our data collection methodology. Section 4 provides a survey of related work. Section 5 describes in detail the hardware and software environment at each site where we gathered data. Section 6 presents our analysis of the gathered data. Section 7 summarizes our study and provides conclusion. Finally section 8 expands and on what we would like to do next, and appendix A provides definitions for values returned in stat system call.

## 2 Reading our graphs

Most of our graphs show empirical cumulative distribution functions; that is, the fraction of samples (usually files, sometimes total size) whose property of interest (often size, sometime overhead or age) is less than a given size, age or overhead. It is common to see a clustering of a large amount of data point in the zero to ten percentile range, or ninety to hundred percentile range. For this reason, wherever pertinent, we plot the zero to hundred percentile range as 3 sections in the graph. The first section represents the zero to ten percentile range where the Y-axis is a log scale, the second section represents the whole range from zero to hundred percentile with a linear Y-axis and finally the third section represents the ninety to hundred percentile range where the Y-axis is again log scale. The X-axis is always log scale. Not all graphs have all three sections, their representation is chosen based on the need to expand. Since different sections of the same graph may have linear or non linear Y-axis, the slope of lines in different sections may be different.

We did not attempt to curve fit our data as many previous studies for primarily two reasons. We wanted our study to derive a simple models that researchers can use as parameters for designing file systems. For instance, instead of deriving whether the HPC file size distribution is log-normal, we wanted to document more simple values such as its mean and median and present a distribution that shows what fraction of files are a given size or age. We also wanted to characterize very large storage systems, typically those used in HPC environment, and so our sample space is small. For instance we collected data from about 13 HPC file systems and one local file system at our department. This sample set is not large enough to subject to curve fitting. Nevertheless, we would like to curve fit at least the file size distribution for completeness since it has been curve fitted by various previous studies [17, 6, 8] that have each found a different analytical function to best fit this distribution.

### 2.1 Legends

We use the following legends to represent lines in all our graphs:

- **lanl-scratch1, lanl-scratch2, lanl-scratch3:** These are 3 different volumes, all used as scratch space by the LANL scientists. Their detailed description can be found at 5.1
- **nersc-projects:** This is a GPFS volume at NERSC, used for projects by NERSC scientists. Its described in section 5.2
- **pnnl-home:** This is the home directory of users at the PNNL, described in section5.3
- **pnnl-dtemp:** This is the scratch space for users at the PNNL, described in section5.3
- **pnnl-nwfs:** This is the archival file system for users at the PNNL site, described in section5.3
- **arsc-projects:** A SAM-QFS file system at ARSC, described in section 5.4. It is used to for holding projects running at ARSC.
- **arsc-seau1, arsc-seau2, arsc-nanu1:** These are SAM-QFS file system volumes at ARSC, described in section 5.4 They are used to archival HPC data at ARSC.
- **psc-scratch:** This is the scratch space for users at the PSC, described in section5.5
- **psc-bessemer:** This is another scratch file system, for more long term data. Described in detail in section5.5
- **pdl1 and pdl2:** These are volumes at our storage lab, at CMU. Described in more detail in section 5.6

## 3 Data Collection Methodology

### 3.1 FSstats

To collect file statistics we built a tool called FSstats. FSstats is a perl tool that runs through a file system and creates size statistics on file attributes such as file EOF (file size), file capacity used, file positive and negative overhead (where file capacity used is more or less than file size), directory size in entries and in bytes, file name length, hard links, symbolic link length and file age. We were careful that FSstats created and uploaded only anonymous data for e.g. a distribution of file size but not file extension. This was done to make our collaborators comfortable with running the tool and submitting results to us for analysis. For this reason our analysis can not be extended to include distribution of file name extension and its corelation to file usage. FSstats can checkpoint to a temporary file and resume from the checkpoint if killed midway. It writes histograms of collected statistics in a CSV format that can be uploaded to our website. When the user uploads, we also ask for a form to be filled that collects context information around the file system and its users, such as: what type of data it is, what is its storage hardware and file system software, any data redundancy schemes etc. FSstats was written to be fast and easy to run and be able to collect statistics with minimal permissions on a file subtree (execute permission on all sub-directories).

FSstats uses the POSIX 'stat' system call to collect summary on individual files. It walks a file tree recursively gathering stat information on all files encountered in the file sub-tree. From the stat returned values we use: number of hard links, total size of file in bytes, number of blocks allocated to file, time of last access, time of last modification and time of last status change to build file statistics. We use the Inode number returned to detect hard-links. See appendix A for an exact definitions of these fields. Below is a description of the size distributions we present in the paper.

- **File size histogram:** This histogram bins data based on size of regular files on the target file-tree. The size calculation is based on the length of the file in KBs. We define this as the end of file address (EOF), as against capacity used by a file which is defined as the number of disk blocks allocated to a file, times block size. For instance on a file system with 4KB block sizes at least 2 disk blocks, or more if file system does preallocation, must be allocated for storing 5 KBs of file data. In this case 5KB is the file size (EOF) and the blocks allocated times 4KB is its capacity used. File size is not necessarily less than capacity used, for e.g. in sparse files, where ranges of file are assumed to be zero filled and so do not have any physical disk blocks allocated to them. Hard links are counted just once for this histogram. We show fraction of files smaller than given size and fraction of total user bytes (sum of EOF of all files) in files smaller than given size. On the graphs, the given bin size includes all files up to that size but not including that size. For e.g. xtic 2K in figure 2 represents all files less than 2K but not those that are exactly 2K in size. They are accounted in the next bucket. for a POSIX definition of the stat field used to calculate file size, see appendix A.

- **Capacity used histogram:** This is a histogram that bins data based on the actual space that a file occupies on disk in terms of disk blocks. Bins are capacity used of files. We show fraction of total files using given capacity and fraction of total capacity used in files of given capacity. As mentioned above, capacity used may be more than file size when file systems do preallocation, or due to internal fragmentation. There is another contributor to capacity overhead, use of data redundancy schemes, e.g. RAID in storing files. Most file systems leave redundancy mechanisms to the disk subsystem, and so stat calls on those files will not show overhead due to redundant data. Some file systems, e.g. Panasas, do RAID per file and thus end up reporting parity overhead in capacity used. Such file systems will typically show a higher overhead in capacity used as compared to others. On the graphs, the given bin size includes all files up to that size but not including that size. For e.g. xtic 2K in figure 4 represents all files less than 2K but not those that are exactly 2K in size. They are accounted in the next bucket. For a POSIX definition of the stat field used to calculate capacity used, see appendix A.

- **Directory size histogram:** We size directories in two ways, by the number of entries in it and by its size in bytes. Since directories are treated as files on UNIX systems, its size in bytes is the same as the size of a regular file (EOF). Figures 16 and 17 show directory distribution by entries whereas figures 18 and 19 show directory distribution by size in bytes. Bins are size of directory in entries or bytes. On the graphs that plot distribution of directory by size in entries, the given bin size includes all directories up to and including that size. For e.g. xtic 32 in figure 16 represents all directories whose size is less than or equal to 32 entries. On the graphs that plot distribution of directory by size in bytes, the given bin size includes all directories up to that size but not including that size. For e.g. xtic 32K in figure 18 represents all directories less than 32K but not those that are exactly 32K in size. for a POSIX definition of the stat field used to calculate directory size in bytes, see appendix A.

- **atime, mtime and ctime histogram:** We collect time statistics of files by recording their atime, mtime and ctime and show distribution of files by age, i.e. fraction of files older than given age, and distribution of user bytes by age, i.e. fraction of total user bytes (sum EOF of all files) older than a given age. To see how POSIX defines atime, mtime and ctime for files and the system calls that affect these fields, see appendix A. Briefly, atime is time of last access or read but other system calls may also change it like execve, mtime is time of last modification or write but other system calls such as mknod may change it and change time defined as time of time of last attribute change like changing link count or time of last write. Again, this histogram considers only regular file and ignores duplicate hard-links. Bins are days since last access, modification or change to file. On the graphs, the given bin size includes all files up to that age but not including that age. For e.g. xtic 32D in figure 10 represents all files less than 32 days but not those that are exactly 32 days in age. They are accounted in the next bucket.

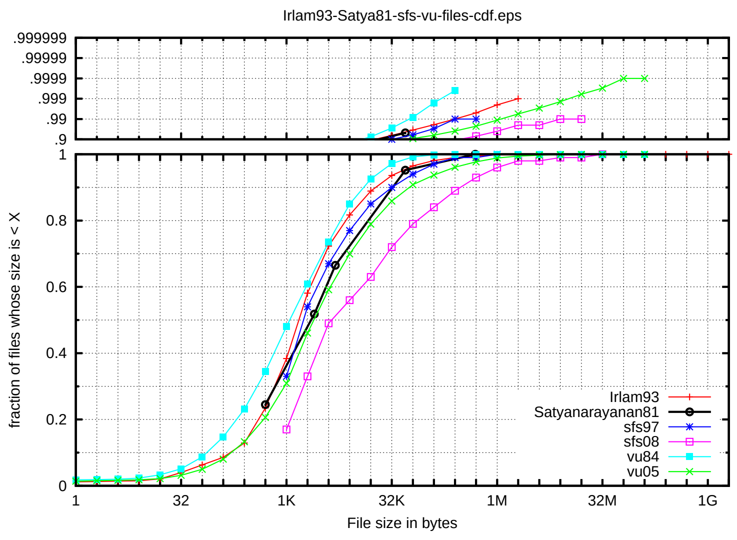
- **Positive and negative overhead histogram:** Files may have their EOF greater or less than their capacity used. We create a distribution of such files in two histograms: positive and negative overhead. Positive overhead records each file whose EOF is greater than or equal to its capacity used, whereas negative overhead records each file whose EOF is less than its capacity used. Using a distribution of positive and negative overhead we show fraction of sparse files or dense files in the file system and the amount of overhead in terms of space (fraction of total positive or negative overhead bytes in files whose size is less than given). These histograms bin not just regular files, but also directories and symbolic links and thus cannot directly be compared with the file-size or capacity-used histograms. Bins are size of file(EOF). On the graphs, the given bin size includes all files up to that size but not including that size. For e.g. xtic 2K in figure 6 represents all files less than 2K but not those that are exactly 2K in size. They are accounted in the next bucket. Overheads are calculate using file size and file capacity used. For a POSIX definition of stat fields used to calculate these value see appendix A

## 4 Related Work

File size distributions are a well studied phenomena because of their importance to file system design parameters. At least once every decade a study reviews how sizes have changed [17, 13, 15, 4, 5, 11, 19, 6, 20, 3, 2]. Similarly, there has also been an extensive study of usage patterns of file systems which also covered some aspect of file size distribution [4, 9, 15, 21, 16, 7]. In the past these studies have significantly impacted the design of file system[12, 14]. This paper presents the first significant collection of statistics on the largest and highest bandwidth file systems, those used in supercomputing systems. Common wisdom is that these systems are primarily huge, short-lived files. Our evidence suggests that this is too simplistic a model; supercomputer file systems have surprisingly many small files and a complex concept of file size.

Perhaps the most widely cited early study examined a single DEC PDP10 running TOPS-10 and supporting the entire computer science department at Carnegie Mellon University [17]. With only 1.6 GB of disk space, space was in demand and staff performed periodic archive and purge exercises leading to 36,000 online files and 50,000 archived files. This study also defined the term 'file functional life time' which is the difference in time from last modification to last access and is considered a measure of the usefulness of the file's data. The (cumulative) file size distribution on this machine is shown on the left in Figure 1. Interestingly later studies [21, 19, 6] found the associated time attributes of the file to be unreliable and completely under the control of the application. This is something that we observe too and treat our analysis of file age with caution.

In 1984 Mullender and Tanenbaum conducted a study of file size distribution on a file server belonging to the computer science department at Vrije university [13]. This study was repeated in 2006 and the results were compared with the previous study [20]. They found that most storage still went to large files though



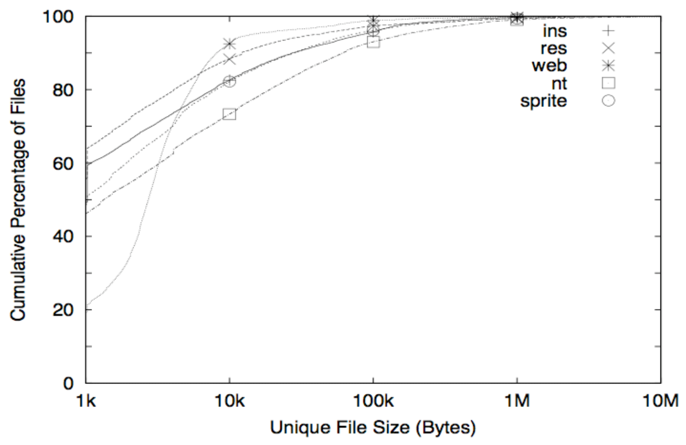


Figure 1: *This figure summarizes file size data reported in earlier papers. Satyanarayanan81 refers to a 1981 study by Satayanarayanan [17], Irlam93 refers to the 93 study by Gordon Irlam [11], sfs97 and sfs08 refer to file size data gathered from the SFS publications [2] and vu84 and vu05 refer to a file size distribution study by Mullender et al. in 1984 [13] and Tanenbaum et al. in 2005 [20]. On the left is very similar file size data captured in 1981, 1984, 1993 and 2005 studies and the file size distributions used in the SPEC benchmark for NFS file servers, SFS97 and SFS2008. The 1981 study examined a 1970s era PDP-10 file system on 8 disks of 200 MB each supporting a computer science department's software development and paper writing [17]. The 1984 and 2005 papers gathered and compared data from a local UNIX file server at the computer science department at Vrije university [20, 13]. The 1993 study examined an anonymous collection of contributed file systems from November 1993 [11]. The SFS benchmark documentation reports industry gathered data on NFS file system distributions appropriate to 1997 and 2008 workloads [2]. On the right is Roselli's Figure 7 "Unique File Size" which is the cumulative distribution function of the files accessed at least once during their tracing periods [16]. The five data sets are "ins", a coursework cluster of 20 machine, "res", 13 researchers' desktops, "web", a single digital library web server, "nt", 8 desktops in a local police station, and "sprite", a similar 8-day trace study done a decade earlier in the same computer science department [4]. The traces gathered for this paper were mostly taken over 31 days (24 for the web server) in 1997 (2000 for the police station desktops).*

the median file size had changed from 1080 to 2475 bytes. Surprisingly the median file size in the 81 study by Satanarayanan is almost as big as the later study in 2005 and slightly bigger than the median size from 97 data collected by SFS [2] and the 93 study by Irlam [11].

In November 1993 a Usenet posted request for a script to be widely run and reported back led to a largely anonymous data set of 12 million file sizes [11]. Shown in the same graph as the 1981 data, on the left in Figure 1, This study summarized:

- file size distribution is heavily skewed toward small files,
- median file size is less than 2 KB,
- files larger than 512 KB comprise more than 50% of the total size of the files, and
- Roughly 89% of files take up 11% of the total size, and 11% of files take up 89% of the total size.

Starting in the mid 80s a number of file systems studies interested in designing memory-based file caches focused on the dynamic patterns; that is, the characteristics of the data accessed, including the sizes of file accesses [15, 4, 16, 7]. While these differ significantly from the statistics of stored files because large fractions of the storage are not accessed during the relatively short periods studied (days to weeks rather than years), they nevertheless report a related file size distribution, the size distribution of accessed files. The data on the right in Figure 1 is drawn from a 3-day trace in the late 1980s and a few 30-day traces in 1997 and 2000 [16]. Interestingly, these distribution curves are quite similar to those of the earlier studies.

In September of 1998 researchers at Microsoft captured the file system statistics on 4,801 desktops with the help of 4,418 volunteers (22% of employees at Microsoft's main campus). This provided data on 10,568 file systems, 141 million files totaling 10.5 TB in size [6]. While much of this paper addressed fitting the observed data to standard distributions, it did present a lot of data and comparisons to prior data. For example, in contrast to even the 1981 data, their study looked at files smaller than 1 KB, and reported that, except for an impulse of 1.7% zero sized files, the lower half of a log-normal distribution is present in their data; that is, prior studies (and this study as well) tend to group a very large fraction of the files in the smallest bucket (at say, 1 KB, 2 KB or 4 KB), obscuring the variation in file size frequencies for very small files.

A 2002 study by Kylie M. Evans and Geoffrey H. Kuenning [8], focused mainly on multimedia files, found that a simple log-normal distribution did not fit the size distribution of media files and suggested that a more complex lambda distribution may be used. This study was also unique in that it gathered data from a heterogeneous environment of Windows, Linux, MacOS and UNIX file servers. They also found the mean and median to have gone up considerably in a two year period since the 1999 study of Douceur and Bolosky [6].

More recent studies of static file size distributions have been used to re-examine the appropriate disk allocation block size, such as the 2005 dataset VU2005 [20] and to construct the file size distribution of the widely used SPEC SFS benchmark for NFS file server machines [2]. SPEC SFS97 and SPEC SFS2008 file size distributions are shown, along with the 2005 VU2005 distribution, in the left graph of Figure 1. The dramatic increase in file sizes in the SFS2008 file size distribution is most interesting; we shall see that it falls inside the variation of our HPC file size distributions.

## 5 File Systems Studied

### 5.1 LANL

Los Alamos National Lab (LANL) is a supercomputing site in New Mexico. We collected file statistics from this site on three different file systems labeled lanl-scratch1, lanl-scratch2 and lanl-scratch3. See a summary of these volumes in table 1. All three are scratch space and use the Panasas PanFS [22] file system. PanFS is a fault tolerant high performance cluster file system that supports per file RAID. The lanl-scratch space is a general purpose file system used mainly for I/O testing, but can be used as scratch space on preproduction clusters, i.e. consisting of user who were porting and testing their code. There is no active purge policy for this file system. It has predominately RAID5 files with 64 KB stripe size, but may have some RAID10 files on them. The storage nodes are organized as 7 shelves each with 10 OSDs (Object Store Device). A single OSD has 500 GB raw space created using two 7200 RPM 250 GB SATA drives. The compute main cluster that uses lanl-scratch has 246 dual processor 1.6 Opteron nodes. The lanl-scratch2 and lanl-scratch3 are used by a different cluster called Flash/Gordon and the Yellow Rail system. These file system use RAID 5 with a stripe unit of 64KB. The lanl-scratch2 has 8 shelves of 800 GB OSDs and lanl-scratch3 has 16 shelves of 500GB OSDs. They're both purged periodically (60 days or older) or on a need basis (to keep capacity used below 90 %) and users are encouraged to archive the files they want to keep. Though users can set RAID levels on a per file basis, in reality, very few do this.

| Label | Date (2008) | Type | File System | Total Size TB | Total Space TB | # files M | # dirs K | max size GB | max space GB | max dir ents | max name bytes | avg file MB | avg dir ents |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Satyanarayanan81 | ¡1981 | Home | TOPS10 | | ¡.0016 | .086 | | | | | | .012 | |
| Irlam93 | Nov 1993 | | | | .259 | 12 | | | | | | .022 | |
| SFS97 | ¡1997 | | NFS | | | | | .001 | | 30 | | .027 | 30 |
| Douceur99 | Sept 98 | Desktops | NTFS | 10.5 | | 141 | | | | | | .079 | |
| VU2005 | 2005 | Home | UNIX | | | 1.7 | | 2 | | | | .327 | |
| SFS2008 | ¡2008 | | NFS | | | | | .032 | | 30 | | .531 | 30 |
| CMU gg1 | 4/10 | OS | HFS+ | .044 | .046 | 1.0 | 258 | 2.1 | 2.1 | 100,344 | 252 | .046 | 5 |
| CMU gg2 | 4/10 | Home | HFS+ | .0098 | .0099 | .028 | 3.2 | .328 | .328 | 448 | 123 | .37 | 10 |
| CMU gg3 | 4/10 | Media | HFS+ | .065 | .066 | .042 | 2.6 | 2.2 | 2.2 | 536 | 129 | 1.6 | 17 |
| CMU pdl1 | 4/9 | Project | WAFL | 3.93 | 3.68 | 11.3 | 821 | 37.7 | 23.4 | 56,960 | 255 | .37 | 15 |
| CMU pdl2 | 4/9 | Project | WAFL | 1.28 | 1.09 | 8.11 | 694 | 37.7 | 23.4 | 89,517 | 255 | .17 | 14 |
| NERSC | 4/8 | Project | GPFS | 107 | 107 | 20.5 | 917 | 616 | 523 | 143,365 | 152 | 5.3 | 23 |
| PNNL nwfs | 3/17 | Archival | Lustre | 265 | 264 | 13.7 | 1,824 | 1,074 | 1,074 | 57,114 | 232 | 19.3 | 9 |
| PNNL home | 3/17 | Home | ADVFS | 4.7 | 4.3 | 10.1 | 682 | 268 | 35 | 23,556 | 255 | .46 | 16 |
| PNNL dtemp | 3/17 | Scratch | Lustre | 22.5 | 19.2 | 2.2 | 51 | 1,074 | 1,075 | 8,004 | 89 | 10.3 | 44 |
| PSC scratch | 3/27 | Scratch | Lustre | 32 | 32 | 2.07 | 451 | 173 | 173 | 64,010 | 160 | 15.6 | 6 |
| PSC bessemer | 3/27 | Project | Lustre | 3.7 | 3.7 | 0.38 | 15 | 51 | 51 | 8,226 | 89 | 9.6 | 26 |
| LANL scratch1 | 4/1 | Scratch | PanFS | 9.2 | 10.7 | 1.52 | 120 | 134 | 154 | 14,420 | 90 | 6.0 | 14 |
| LANL scratch2 | 4/10 | Scratch | PanFS | 25 | 26 | 3.30 | 241 | 978 | 1,076 | 50,000 | 73 | 8.2 | 15 |
| LANL scratch3 | 4/10 | Scratch | PanFS | 26 | 29 | 2.58 | 374 | 998 | 1,099 | 45,002 | 65 | 10.9 | 8 |
| ARSC seau1 | 3/13 | Archival | SAM-QFS | 305 | 4.3 | 10.5 | 326 | 386 | 13.7 | 62,803 | 245 | 29 | 34 |
| ARSC seau2 | 3/14 | Archival | SAM-QFS | 115 | 4.6 | 5.3 | 116 | 366 | 7.0 | 25,008 | 144 | 21.7 | 47 |
| ARSC nanu1 | 3/12 | Archival | SAM-QFS | 69 | 4.5 | 6.7 | 338 | 601 | 13.6 | 58,648 | 234 | 10.4 | 21 |
| ARSC projects | 3/13 | Archival | SAM-QFS | 32 | .93 | 6.2 | 898 | 171 | 3.7 | 24,153 | 81 | 5.2 | 8 |

Table 1: *This table summarizes the statistics gathered from all 15 file systems and older studies. We report here the date when data was gathered, the type of file system, total file space and capacity used, number of files and directories, file with biggest size and capacity used, biggest directory size in entries, biggest filename and average file size and directory size in entries. Direct comparisons may be made in mean file size, mean directory size etc. within file systems studied by us. A shift in trend in statistics from older studies is also visible. For example Douceur et al., found avg file size to be 0.079 MB where as average file sizes in the HPC data are bigger by an order of magnitude.*

## 5.2 NERSC

NERSC (The National Energy Research Scientific Computing Center) is a scientific computing facility for the Office of Science in the US Department of Energy, located at Lawrence Berkeley National Laboratory in Berkeley, California.

We collected data from the NERSC Global File System (NGF) which provides shared storage accessible from five supercomputers: Franklin, PDSF, Jacquard, Bassi, and DaVinci. Franklin is a Cray XT4 with 9,660 dual Opteron nodes. See a summary in table 1. PDSF is 275 nodes with a mixed collection of dual x86 CPUs. Jacquard is a Linux Networx cluster of 356 dual Opteron nodes. Bassi is an IBM p575 POWER5 cluster of 122 8-processor nodes. DaVinci is a SGI Altix 350 with 32 Itanium-2 processors. While all these supercomputers have local scratch disk space of different sizes using different file system software, we had access to only the shared NGF file system.

NGF is a IBM GPFS file system [18] with 96 volumes built on top of 2 Data Direct Networks (DDN) S2A 9550 disk arrays and 4 IBM DS4500 disk arrays. The DDN disk arrays are both configured as 8 data + 2 ”parity” RAID-6 arrays; one has 16 RAID sets (each 8+2 disks) using 250 GB 7 Krpm SATA disks and the other has 24 RAID sets using 300 GB 10 Krpm FC disks. Two IBM DS4500 disk arrays each have 16 RAID sets each a RAID-5 array of 4 data + 1 parity disks configured with 512 KB stripe units and using 250 GB 7 Krpm SATA disks. The other two IBM 4500 disk arrays each have 12 RAID sets each a RAID-5 array with 5 + 1 disks configured with 64 KB stripe units and using 300 GB 10 Krpm FC disks. The result is about 132 TB usable space with another 31 TB for RAID overhead (24%). Files in NGF are not automatically archived and purged except when a project becomes inactive.

The users of NGF are over 100 science projects funded through the Office of Science's SciDAC and INCITE programs. Example projects include Interaction of Turbulence and Chemistry, Full Vehicle Wind-noise Simultion, Plasma Based Accelerators, Climate-Science Computation, X-Ray Free Electron Lasers, Modeling the Earth System, Clouds in Global Clime, Thermahaline Circulation, Magnetohydrodynamic Modeling, Quantum Simulations of Nanostructures, Turbulent Combustion, and Supernovae.

## 5.3 PNNL

PNNL (Pacific Northwest National Laboratory), is a U.S. Department of Energy (DOE) government research laboratory in Richland Washington.

We collected attribute data from three different file systems at PNNL, labeled as pnnl-dtemp, pnnl-nwfs and pnnl-home. See a summary of these volumes in table 1 The pnnl-dtemp dataset is a general purpose global scratch space. Its purge policy deletes a file after 30 days since last use but this policy is not currently enforced since free space is ample. The volume uses the Lustre [1] file system. Lustre, for this volume, runs over 32 storage servers using 64 logical LUNs. The disk array backend is a HP EVA3000 with FC disks, configured with RAID-5 LUNs. The compute cluster attached to this volume is the mpp2 cluster of 946 dual Itanium HP RX2600 nodes used by projects in the Environmental Molecular Sciences Laboratory such as preteomics, chemistry, biology, and biochemistry. The pnnl-home is home directories for scientists who have projects approved to run on the mpp2 cluster. Files in both of these volumes are manually archived to the pnnl-nwfs volume on a need basis. The pnnl-nwfs volume is backed by 70 storage servers each with 2 to 4 RAID-5 LUNs built by a 3ware 9000 series controller. These LUNs use a mix of 400GB, 750GB, and 1TB drives, all 7Krpm SATA drives.

## 5.4 ARSC

ARSC (Arctic Region Supercomputing Center) is part of the DoD (Department of Defense) and the university of Alaska Fairbanks for high performance computing. Its goal is to support computational research in science and engineering with emphasis on high latitudes and the arctic. We collected file systems statistics from four major file systems here, labeled: arsc-sea-u1, arsc-sea-u2, arsc-nan-u1 and arsc-nan-projects. The volume arsc-sea-u1 is for academic user archival storage on the DoD systems, arsc-sea-u2 is for DoD user archival storage on the DoD systems, arsc-nan-u1 is for academic user archival storage on academic (non-DoD) systems and arsc-nan-projects is for academic user archival storage who have requested actual shared project space, e.g. National Weather Service, Bureau of Land Management etc. Table 1 contains of summary of the file systems. Volumes arsc-sea-u1, arsc-sea-u2 and arsc-nan-u1 usually contain data archived from scratch but may come from anywhere. Client mount these file systems using the SUN SAM-QFS file system. SAM-QFS can hide the file's real storage (disk or tape) from the user and so a stat on a file on tape can return a large file size (EOF) but no capacity used. SAM-QFS inodes also track multiple copies of a file existing on various disk and tape systems and a stat returns a sum of this information. The arsc-nan-projects is used for shared (multiple owner) projects, not much file system information was available to us about arsc-projects. Volumes arsc-sea-u1, arsc-sea-u2, and arsc-nan-u1 use two storage servers, seawolf and nanook, which are configured with the same equipment. Each of these three file systems consists of 5.5TB (5 x 1.1TB LUNS, each of which are 4+1 RAID5 300GB FC 10k/rpm drives) drive space on two separate

StorageTek FLX380 drive array. These drives use a 512KB segment size, giving a 2MB stripe size for any write. Two main compute clusters are concurrently using this storage, Midnight and Iceberg. Midnight is a Sun cluster comprised of 2312 compute Opteron processors in 415 shared memory nodes. Iceberg is 800 processor IBM System spread over a mix of 98 IBM servers.

### 5.5 PSC

PSC (Pittsburgh Supercomputing Center) is a joint effort of Carnegie Mellon University and the University of Pittsburgh together with Westinghouse Electric Company to provide resources for high performance computing to universities, government, and industrial researchers. We collected file statistics from two volumes used as scratch space in the PSC clusters. They are labeled psc-scratch and psc-bessemer. Both are used for output from parallel jobs running on the compute cluster. Users of psc-scratch archive files that they need and delete those they don't. PSC deletes files from here on a need basis, generally those older than 21 days. Whereas psc-bessemer is used for special projects that need more disk space for a longer period of time. Files are deleted only after the project is completed. See a summary of the volumes in table 1. Both file systems run Lustre, a high performance file system for shared clusters. The psc-scratch is used by BigBen, a Cray XT3 MPP system with 2068 compute nodes and twenty-two dedicated IO processors. The storage is backed by 1 Lustre DDN 8500 in a 8+1 configuration, serving 24 2TB luns to 8 lustre OSS nodes. The psc-bessemer used 3 Lustre DDN 9550 in a 8+2 configuration serving 24 6TB luns to 12 lustre OSS nodes. All drives in both configurations are 400GB 7200RPM. The psc-bessemer is also attached to BigBen, Pople: an SGI Altix 4700 with 768 processors and 1.5 TB of shared memory, and Salk: an SGI Altix 4700 with 144 processors and 288 gigabytes memory.

### 5.6 CMU

We also collected data from our storage laboratory at CMU called PDL (Parallel data laboratory). PDL is composed of a group of graduate students and scientists at CMU who work on storage systems related research and projects. File statistics were collected from a storage server that manages a variety of volumes storing data typical to university development and research environment. This data is represented as pdl1 and pdl2 on the Graphs. These volumes store user home directories, CVS repository, OS distributions, music and video for file system testing, internal projects and trace data collected from industry as well as CMU labs. See a summary of PDL volume in table 1. Its backend store is a NetApp filer using WAFL [10] with Dual parity RAID LUNs created out of SATA disks. We also collected statistics from a volume that is used as PDL scratch space, labeled as pdl-scratch.

## 6 Results

### 6.1 File Size

The mean file size ranges from 169 KB to 29 MB. Ignoring the 3 file systems with the smallest mean size: pdl2, pdl1 and pnnl-home, of mean sizes 169 K, 374 K and 463 K, the smallest mean value is 5.1 M. This means that 80% file systems have a mean file size that varies from 5.1 M to 29 M. The mean file size range in HPC data is considerable bigger than the mean file size in a 99 study. Douceur et al., found mean file size to range from 64K to 128K, which was still 4 times bigger than a similar study in 94 [19]. This shows, as expected, that files on average are many orders of magnitude bigger in HPC environment than on workstations. Figure 2 shows that the median file size ranges from 0 to 256K, for all file systems. Ignoring one file system, psc-bessemer, the range of median is 2K to 256K. The large mean value as compared to the median value is due to the presence of large files as also noted by Sienknecht et al. Our median file size

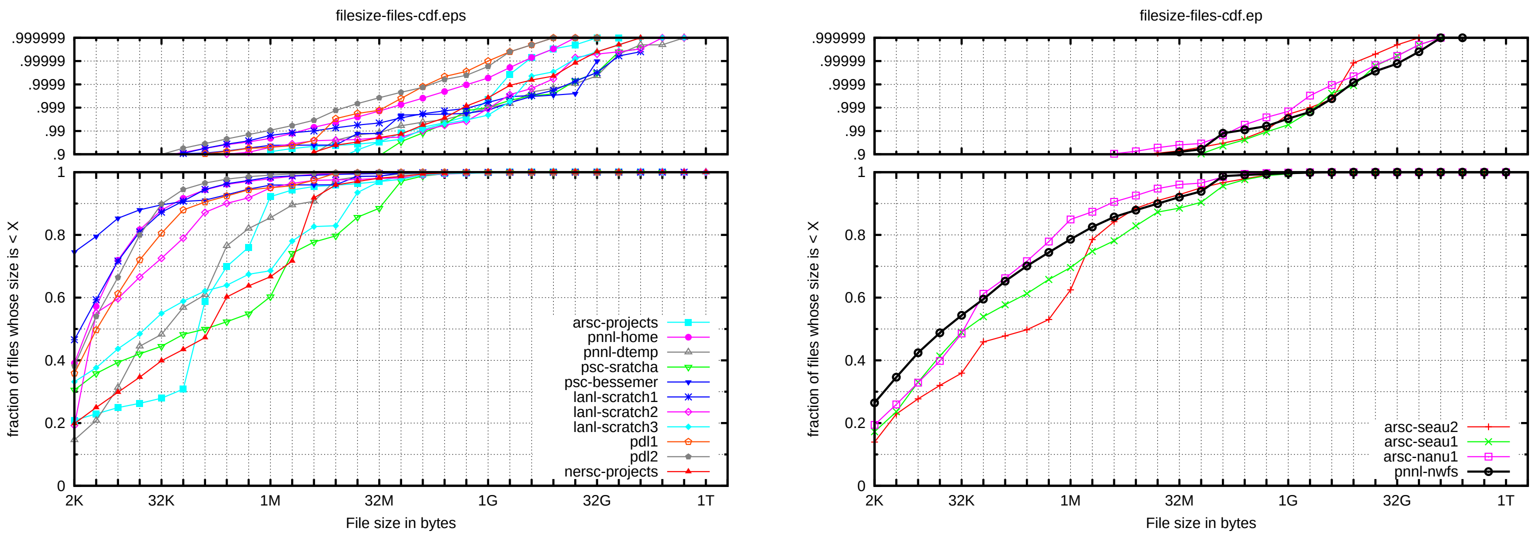


Figure 2: This figure shows a CDF of files of given size across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3). The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is linear from 0 to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. Files across file systems show a wide variety of behavior. There are file systems with only 20% of files less than 2K to file systems with almost 75% files less than 2K. The archival file system lines are much more herded and almost linear as file size grows.

range is also considerably bigger than the one previously reported from workstation studies by Douceur et al. They found median file size on workstations to be 4K. Figure 2 shows that 90% of files are relatively small, varying from 0 to 64M in size, as compared to the remaining 10% that varies from 32K to 1T in size or 25 binary orders of magnitude. This is a wide variation in just a 10% range. We see here the advantage of plotting the graphs as multiple sections to better view the activity in upper 10 percentile range.

Looking at the graphs in figure 3 that plots distribution of total file space consumed, as a function of file size, we see that the median value across all file systems ranges from 4M to 32G. The Archival file systems have a narrower median range, between 256M to 1G.

There is a clear shift in the range of median values when graphed as a distribution of total file space compared to total files. This shift in median value shows, as often noted in previous studies, that while most files are small, most bytes are in large files. For example, in the archival file systems, almost 90% space is consumed by files 32M or greater where as 90% files are smaller than 32M.

## 6.2 Capacity used

The mean capacity used across all file systems ranges from 144K to 19 M. Of this, 7 file systems, 46% of all file systems have a mean value that ranges from 144 K to 868 K. The remaining file systems, more than 50% have a mean value that ranges from 5M to 19M. Some of the smallest mean values: 868, 404, 149 and 674 bytes, are in the SAM-QFS file systems: arsc-seau2, arsc-seau1, arsc-projects and arsc-nanu1ARSC. Since SAM-QFS transparently manages files between tapes and disks a large capacity of files is actually on tapes and not on disks and so does not get accounted from the stat system call. The SAM-QFS inode will not show capacity residing on tape as allocated. Again the presence of large mean values as against the median values indicate the effect of large files. Some file systems that have a small mean value for capacity used

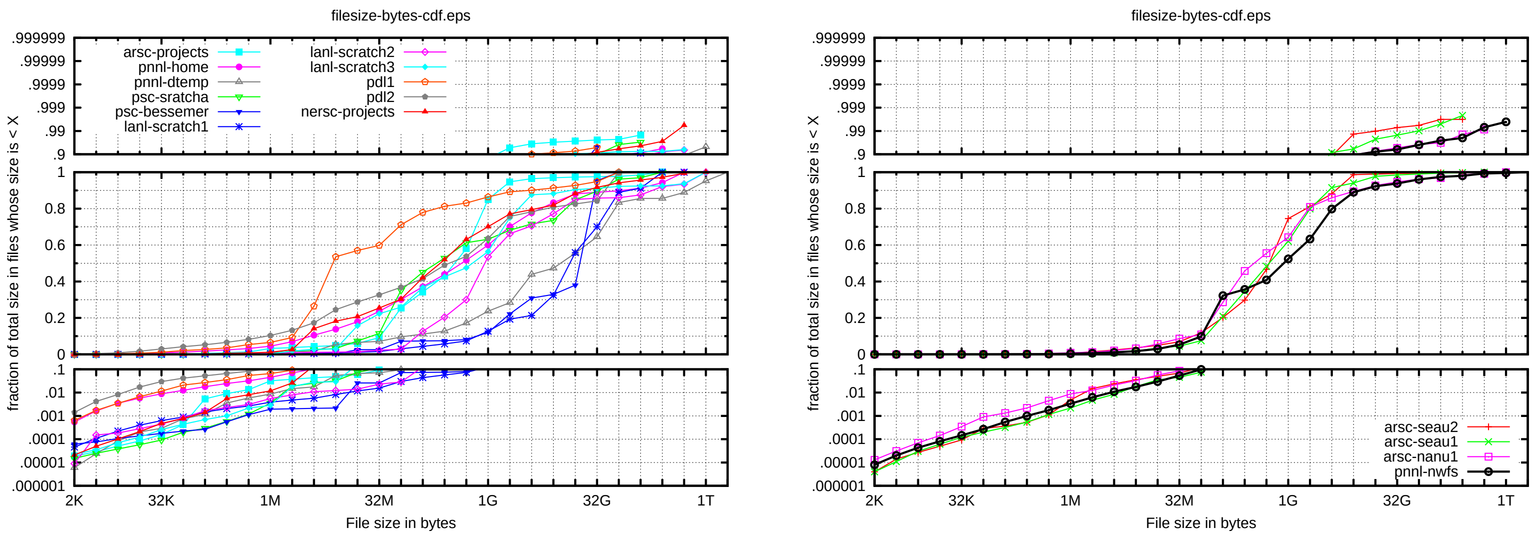


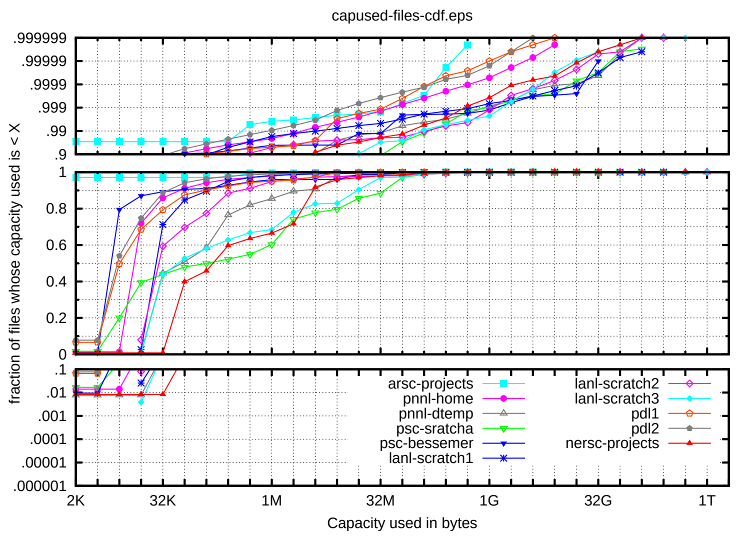
Figure 3: This figure shows a CDF of total file space in files of given size, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. In the non-archival file systems, there is considerable activity in the lower 10 percentile range, showing that 10% space is consumed in files whose size range from 0 to less than 1G. There is a clear shift in lines as compared to the previous figure that graphs fraction of files of given size, indicating that most space is consumed by large files. For example in the archival file systems, almost 90% space is consumed by files 32M or greater where as 90% files are smaller than 32M.

are pdl1 and pdl2, 350K and 144K, which is expected given that they are file systems for department at a university.

Figure 4 graphs files as a function of capacity used on disk. The median value for capacity used across all file systems ranges from 32K to 256K.

The archival file systems show some interesting characteristics. In 75% of file systems almost 80% to 92% files use up less than 2K capacity. Since we have a single bin from 0 to 2K we cant tell if these files are actually 0 bytes in size, but it is likely since they are on archival file systems arsc-seau1, arsc-seau2 and arsc-nanu1. A completely horizontal line until the 2M point on X-axis implies that there were no files that used more than 2K and less than 2M capacity. But we can see from figure 2 that there are only 15 to 30% files whose size is less than 2K. This anomaly is due to the fact that these 3 file systems are SAM-QFS, which transparently moves data to tape. As a result 'stat' will not show any disk space allocated but will show a large file space to exist. Volume arsc-projects, though categorised as non-archival, shows similar behavior since it is a SAM-QFS file system.

Looking at the non-archival file systems, the capacity used range is comparatively wider in the top 10 percentile range. 90% of files use just 64M or less capacity, but the maximum capacity used is 1T. This suggests that there is a narrow band of capacity usage in which most files lie, but there are a few files that are spread across a relatively large range of sizes. There was a similar narrow band for file size with 90% files less than 64M in size. Thus there is a typical value for capacity used and file size, which suggests that there must be optimizations in file systems design for this common case.



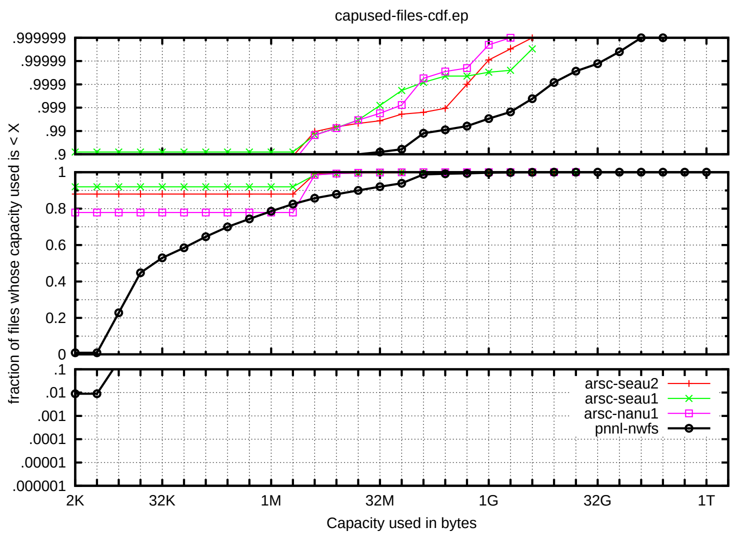


Figure 4: This figure shows a CDF of files with given capacity used, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. The SAM-QFS file systems show an interesting characteristic. Since they don't report allocated space for files on tape to stat calls, a large amount of file space appears to be compressed in a small amount of disk space.

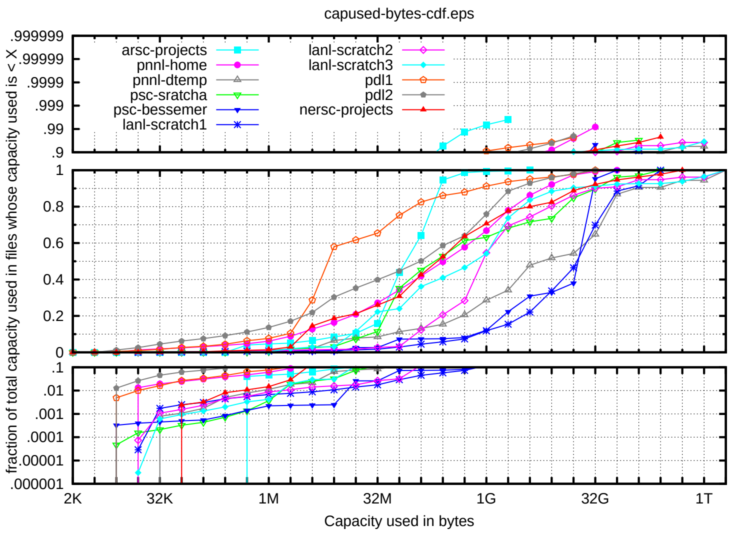
The median value for total capacity used ranges from 4M to 32G, i.e., 50% of total capacity used is in files whose capacity used ranges from less than 4M to less than 32G, for all file systems. The comparatively large range of median values for this distribution over the previous one (32K to 256K) shows again that large files dominate capacity requirement on disk.

## 6.3 Positive overhead

The mean positive overhead ranges from 2K to 1463K, across all file systems. Note that the positive overhead histogram only counts files that have some positive value of overhead, including 0. All files whose capacity used is less than their EOF are counted as negative overhead. So the above mean is the average amount by which a files capacity used exceeds its EOF address.

Figure 6 shows that the median ranges from 0 to 64K. It means 50% of files with overhead, across all file systems, have a size that ranges from 0 to 64K. Note that this includes files that have no overhead, since we count files with 0 overhead in the positive overhead histogram. This allows us to get a mean overhead per file, in files that have overhead, thus the 0 in the 0 to 64K range. 90% of files with overhead have a size that ranges from 0 to 16M though the absolute file size ranges is much wider; 0 to 1T. While it appears that most files with overhead are small in size, we should remember that most files are small anyway. The overhead per file is useful only when seen together with the total number of files with overhead. Across all 14 file systems (excluding NERSC from the 15 studied), files with positive overhead ranges from 0.39 Million to 15.57 Million. Remember that positive overhead files include directories and symlinks and not just regular files. We do not have file overhead data form NERSC yet.

A useful contrast is to see where most overhead bytes are. Figure 7 shows the fraction of overhead



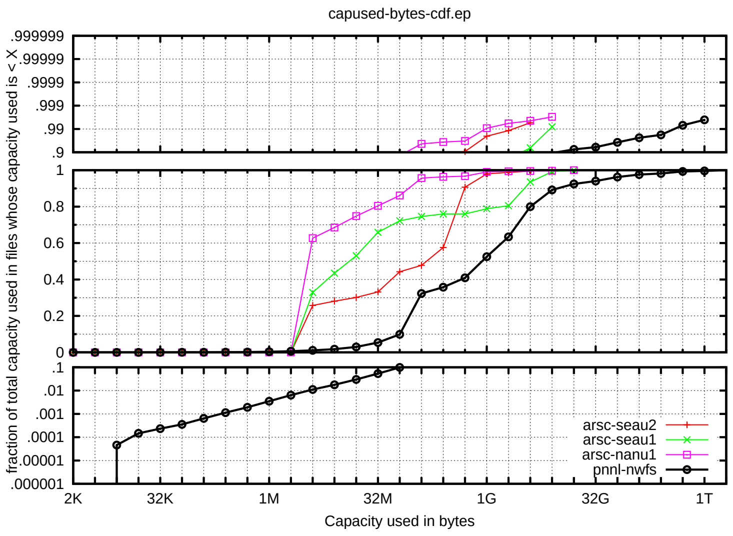
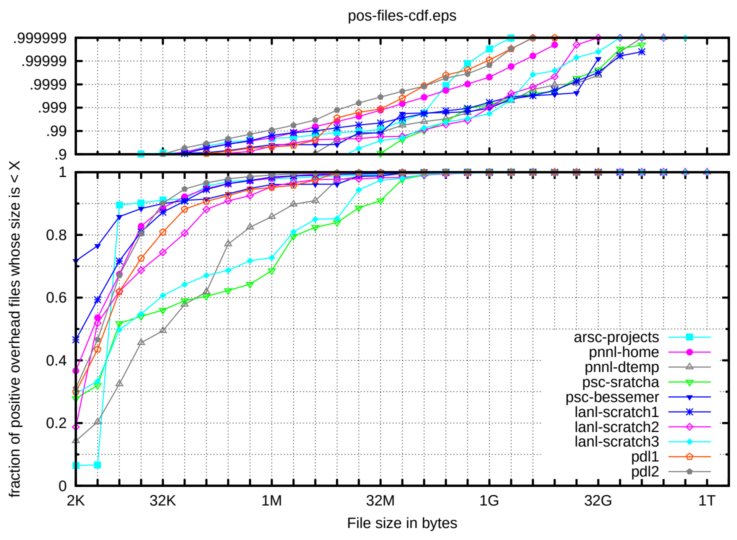


Figure 5: This figure shows a CDF of total capacity used in files of given capacity used, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. A narrow band of size in the 20 to 90% range suggests that most files are relatively close to each other in size and only a few files are spread out in the wide band of capacity used or file size spectrum.

bytes in files of given size. We can see that the median has a wide range from 0 to 16G. Which means that 50% overhead in file systems may be anywhere in between files less than 2K to less 16G. Looking just at the archival file system, the range of median is narrowed substantially when one file system, pnnl-nwfs, is excluded. I.e., 75% archival file systems have a median value for overhead bytes in between 16K to 64K. LANL, that has panfs, has most of its overhead in relatively big files, while the other file systems have overhead in relatively smaller files. This is likely because none of the other file systems are counting overhead due to parity and metadata in their capacity used. While panfs accounts for overhead due to RAID parity and metadata overhead when it reports capacity used to the stat system call. Since the big files take up the most space on disk also have big overheads. This pushes the median much further in the range of big files for LANL, as compared to others. The parity overhead will be visible in statistics in disk subsystem though, for the other file systems.

## 6.4 Negative overhead

The mean negative overhead, across all file systems, ranges from 58K to 1.05 K, i.e., of all files with negative overhead, the average value by which the EOF exceeds the capacity used in these files varies from 58K to 1.05G. Figure 8 shows the distribution of files with negative overhead as a function of file size. Only files with negative overhead are included. As can be seen, in 100% file systems, 90% files with negative overhead have a size that ranges from less than 2K to less than 256M. Many of them have 90% overhead files less than 2K in size. Our guess is that this could be due to some optimization such as stuffing small file data in the Inode of the file (for e.g. in the case of symbolic links). If the file system does not include the metadata overhead in the capacity used, then these files will appear to have negative overhead. For example volume



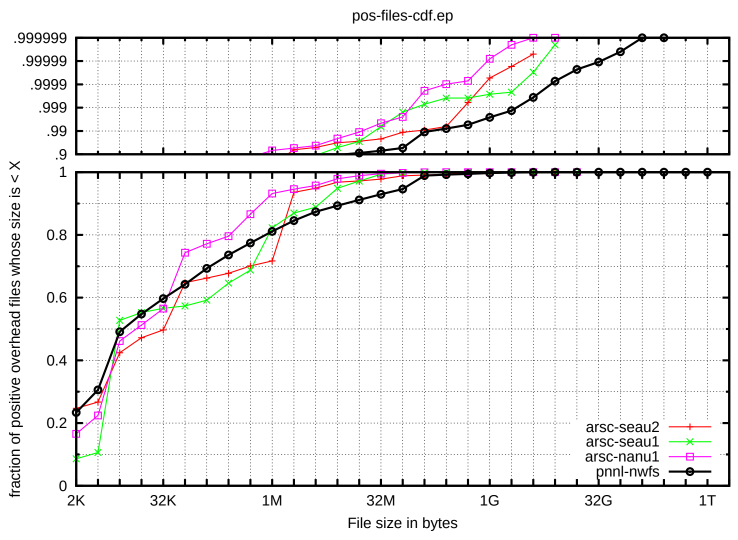


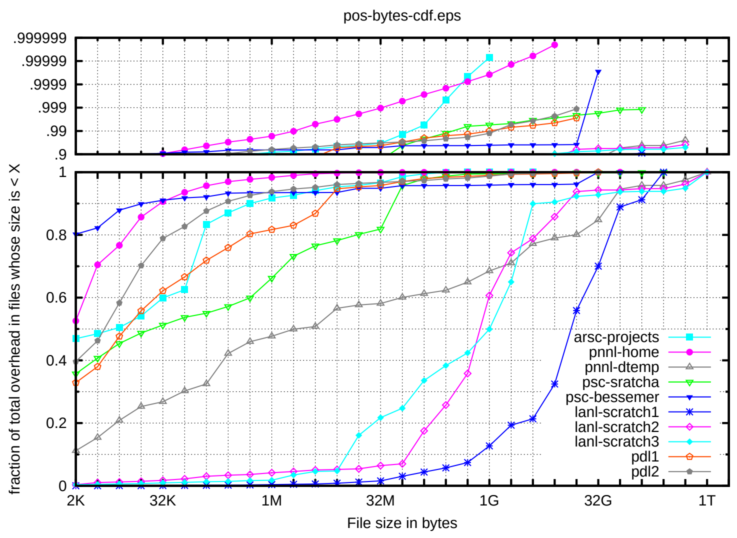
Figure 6: This figure shows a CDF of files with positive overhead in files of given size, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is linear scale from 0 percentile to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. It appears that most overhead is in small files, but small files are the most abundant on file systems anyway.

pnnl-nwfs has has over 99.5% overhead files, less than 2K in size. In absolute terms, there are 17447 negative overhead files in the 0 to 2K size range that together account for 410.4K of negative overhead space. It has 19624 symlinks in the file system, suggesting that most of the negative overhead files are actually symlinks.

We only consider 14 of the 15 file systems we studied since overhead statistics from NERSC were not available. lanl-scratch1 had only 1 file with a negative overhead. Incidentally LANL had a few very big empty files, several Tera bytes in size, almost all of which was a negative overhead. I.e., their capacity was almost zero. This skewed the negative overhead statistics for LANL data. LANL removed them after discovering them via fsstats. These files had been accidentally left on the system for a long time. The current data does not include those files.

In the non-archival file systems, lanl-scratch2, lanl-scratch3 and arsc-projects have negative overhead in comparatively larger files. In the HPC environment, often a checkpoint file is created, and many cores seek to different regions of the file and write to it. This seek will cause whole in the files that otherwise unfilled will create sparseness in the file. This could be a reason for large files with negative overhead in the HPC environment.

In the archival file systems, barring pnnl-nwfs, there is a more uniform distribution over size in files with negative overhead. The median value ranges from 16K to 512K (excluding pnnl-nwfs). Figure 9 shows the distribution of the total negative overhead (in bytes), as a function of file size. The first thing we notice is the clear shift in the curves to right, when compared with the distribution of files with negative overhead. This shows that though most files with negative overhead are small in size, most savings in space are coming from the large files. This has not been been discovered in any previous study. This trend is obvious in both archival and non-archival data. This is also evident in the median. The median for distribution of total negative overhead, across all file systems, ranges from 256M to 512G. This is relatively large when compared to the median in the distribution of negative overhead files, whose range was 16K to 512K.



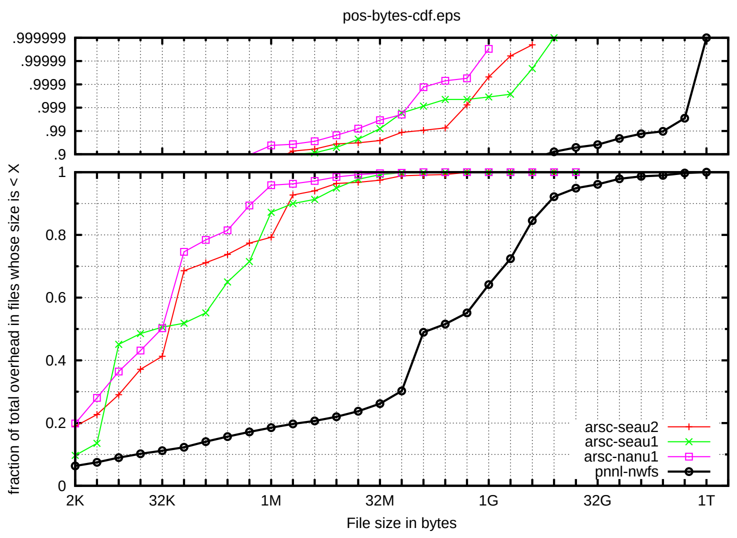


Figure 7: This figure shows a CDF of total positive overhead in files of given size, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is linear scale from 0 percentile to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. LANL, that has panfs running on its systems, seems to have most of its overhead in big files. This is due to the fact that panfs is likely the only file system that accounts for RAID parity overhead and metadata overhead, such as inode and indirect blocks, in the value it reports for capacity used. Since big files use up majority of disk space, as can be seen from previous graphs, the big files also end up paying for most overhead. This overhead though is not visible in file systems that do not account for overhead due to parity and metadata in file system stat calls.

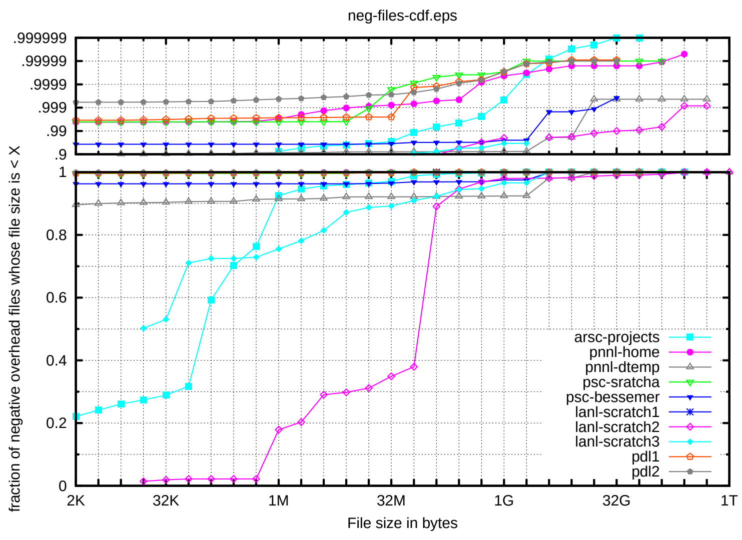
The archival graph in figure 9 shows that the archival file systems, barring pnnl-nwfs, have a narrower range for median, varying from 256M to 1G. I.e., 75% archival file systems have relatively less variability in the median value. There is some bias here from the fact that the remaining 3 archival file systems, when we exclude pnnl-nwfs, are from the same site, ARSC.

## 6.5 File Age

The age of files in terms of modification time, access time and change time are believed to be indicators of usefulness of data in that file [17]. But as has been noted earlier and found in our own study, file time attributes are unreliable. This may be seen from files that have future time stamps or appear to be too old, sometimes older than the machine itself. File time attributes are under the control of application and can be set using UNIX system calls. For this reason, it is hard to say how accurate and reliable is the time data retrieved from stat calls. For example, files with obviously wrong values may be ignored easily but those with not so obviously wrong values will be hard to discern.

### 6.5.1 Distribution of files as a function of access time

Figure 10 shows the access age of files. The mean access time ranges from 9 days to 838 days. This is a very large variance in access time. Of this, the SAM-QFS file file systems have the largest mean value for



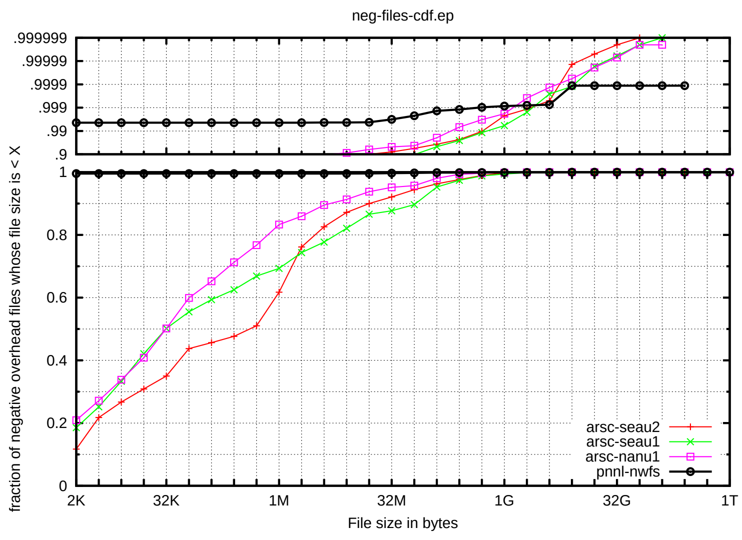


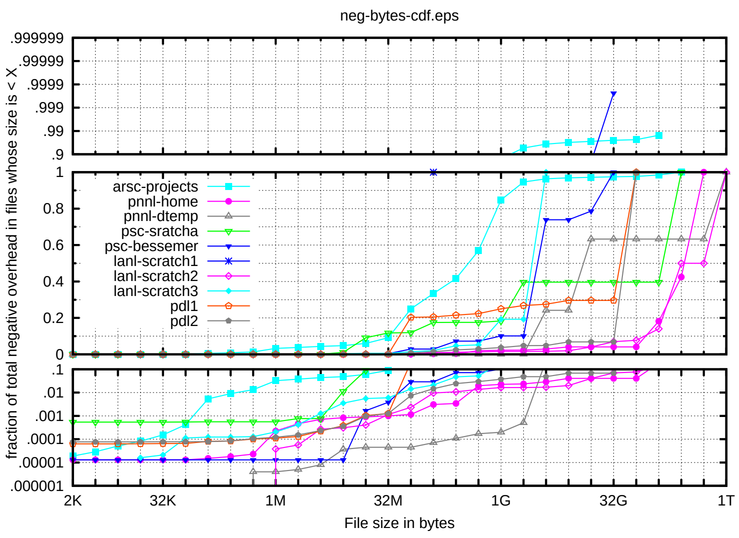
Figure 8: This figure shows a CDF of files with negative overhead in files of given size, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is linear scale from 0 percentile to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. pnnl-nwfs has 99.56% negative overhead files smaller than 2K in size. It is possible that they are all symlinks with optimizations such a Inode stuffing obscuring the capacity used on these files. pnnl-nwfs has 19,624 symlinks. There are other file systems with similar statistics.

access time ranging from 598 days to 838 days. The median access time ranges from 32 days to 1024 days for the archival file systems and from 4 days to 1024 days for the other file systems. This suggests that the files in the archival file system are much older in terms of access time, as expected. This data should be read cautiously since access times are under application control and may not contain the real access time for the files. For instance the lanl-scratch2, lanl-scratch3 and pdl1 file systems have access age in negative for some files, which means the access time is set to a future date. The minimum access time on both lanl-scratch2 and lanl-scratch3 is -32555, that affect their mean access time. Although the fraction of files that have access time in the negative is not substantial, just 134 files out of 3.3 million for the lanl-scratch2 and 867 out of 2.6 million for lanl-scratch3. But it does suggest that using access time to summarize file functional lifetime may be an unreliable method.

### 6.5.2 Distribution of total file space (sum EOF) as a function of access time

Figure 11 shows the access age of total file space, sum of EOF of all files on disk. The mean value ranges from -474 days to 863 days. Ignoring the negative value, the mean value ranges from 7 days to 863 days. This is again a wide variance in access time of different file systems. The largest mean values are again in the SAM-QFS file system suggesting that the oldest bytes live there.

The median value for the archival file system ranges from 32 days to 1024 days and for the non-archival file systems ranges from 4 days to 512 days. Of the 4 archival file system, 3 (75%) have access time greater than 512 days. Median value for the remaining file systems varies from 4 days to 512 days. At least 4 of these file systems, 36% of non archival file systems, have a median value ranging from 256 to 512 days, suggesting much older bytes. This means that for the HPC data, the access time show a wide variance and



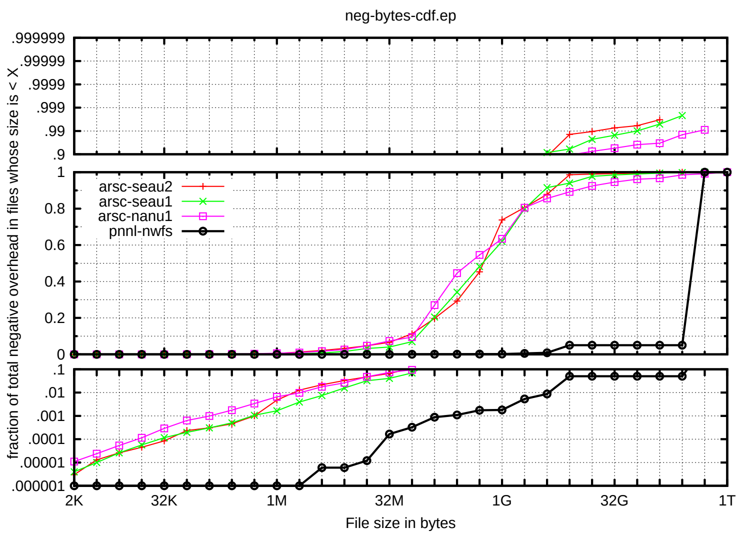


Figure 9: This figure shows a CDF of total negative overhead in files of given size, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. A clear shift in the distribution of negative overhead bytes, when compared to distribution of negative overhead files, shows that most significant sparseness is in big files, though most files that are sparse are small in size.

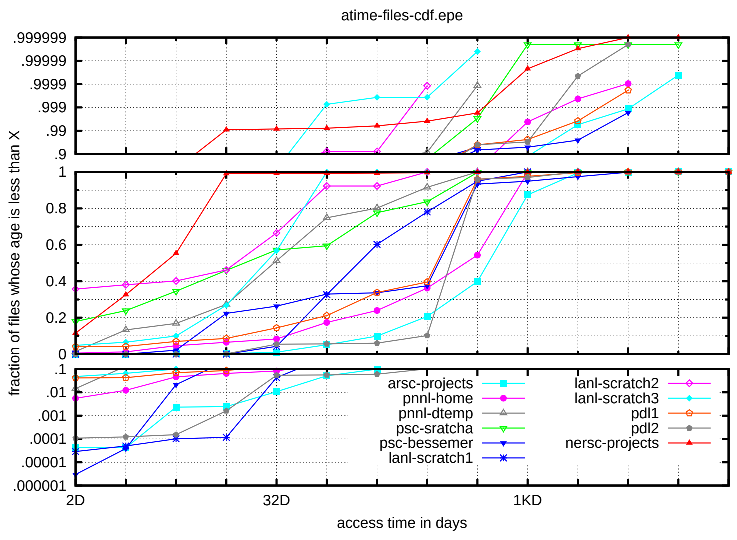
are probably dependant on the purge policy at each site.

#### 6.5.3 Distribution of files as a function of modification time

Figure 12 graphs the file modification age for files in file systems. The mean file modification time in our data set ranges from 64 days to 1169 days. Again there are several file systems that have some files with negative file modification time: arsc-seau1, lanl-scratch2, lanl-scratch3, pnnl-dtemp, pnnl-nwfs, pnnl-home. These files have future time stamps for the same reason as we discussed in the previous section. The fraction of files with future time stamps are very few, ranging from 1 to a few hundred in millions of files.

As expected, the mean file modification time in the archival file systems is comparatively high, ranging from 816 days to 940 days. Other file systems with comparatively old file modification time are arsc-projects, which is expected since it is a SAM-QFS file system and pdl1 and pdl2 which is surprising. The mean file modification time for pdl1 and pdl2 is 1128 and 1169 days. pdl1 and pdl2 are departmental file server but also host directories that store a collection of media files (audio and video) for file systems testing. It is possible that these files cause the mtime average to be skewed. There are also some trace files contributed by industry research lab that are several years old. This high average figure can be seen even in the ctime stats. We should note here that pdl1 and pdl2 show several similar values in some histograms in the same histogram bin, leading us to believe that they may have a bunch of copied files from each other.

Figure 12 shows that the median modification time ranges from 16 to 1024 days. Of this, 5 out of 6 scratch file systems (83% of all scratch file systems) have 50% files younger than 256 days and 90% files younger than 512 days. This suggests, as expected, that files are comparatively younger in the scratch file systems than projects and home at HPC sites.



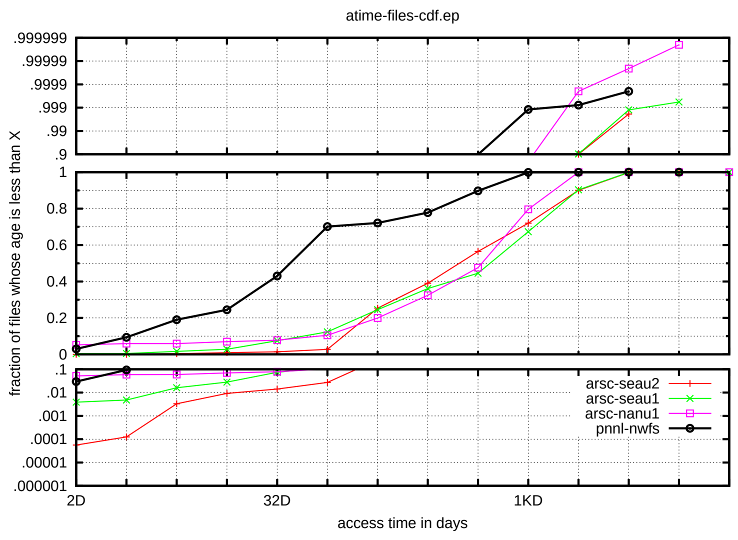


Figure 10: This figure shows a CDF of files whose access time is less than given age, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. The files in archival file systems are relatively older when compared to the non-archival ones. Access time of less than 0 days indicates future time stamps on file, which implies that the time attribute associated with files is not always reliable when held to POSIX definitions.

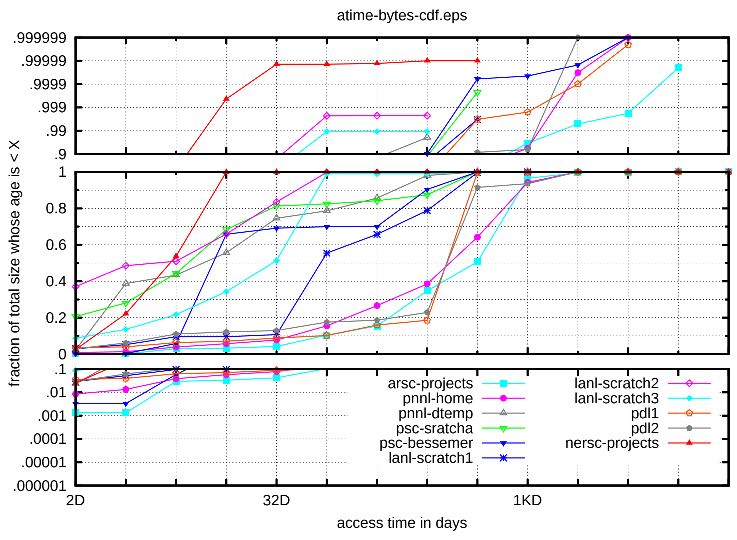
Douceur et al. [6] found the median file age to range from 1.5 to 388 days on 90% file systems. The file age they were referring to was time since last creation or modification. Our median value ranges from 16 days to 1024 days for the project and scratch file systems and from 512 days to 1024 days for the archival file systems. This shows that data at HPC sites may have a much wider range in age and relatively older files as compared to workstation data.

The mean file age for scratch file systems varies from 64 days to 667 days, a much narrower range, but still wide enough to indicate that file modification time in HPC systems can be heavily influenced by the purge policy of the HPC site.

### 6.5.4 Distribution of total file space (sum EOF) as a function of modification time

Figure 13 shows the distribution modification age of total file space. The mean modification time for all file systems ranges from 34 days to 1180 days. The median value for the non-archival file systems ranges from 8 days to 2048 days while the median value for the archival file systems range from 512 to 1024 days. There is a clear difference in age between the archival data and the scratch and project data from the HPC sites. For the non archival file systems, 90% of all file space is 32 to 2048 days old where as 10% of all file space is 0 to 512 days old . For the archival file systems, 80% of all the space in all the file systems is 1024 to 2048 days old and 10% of all space in all the file systems are 0 to 256 days old (ignoring negative age values in both case). The non archival file systems show considerable activity in the 0 to 10% and 90 to 100% range as can be seen in section 1 and 3 of graphs in Figure 13.

The wide range in the median value for the non archival file system (8 binary orders of magnitude) suggests that HPC data shows much wider variety in age of bytes as compared to workstation data [19, 6].



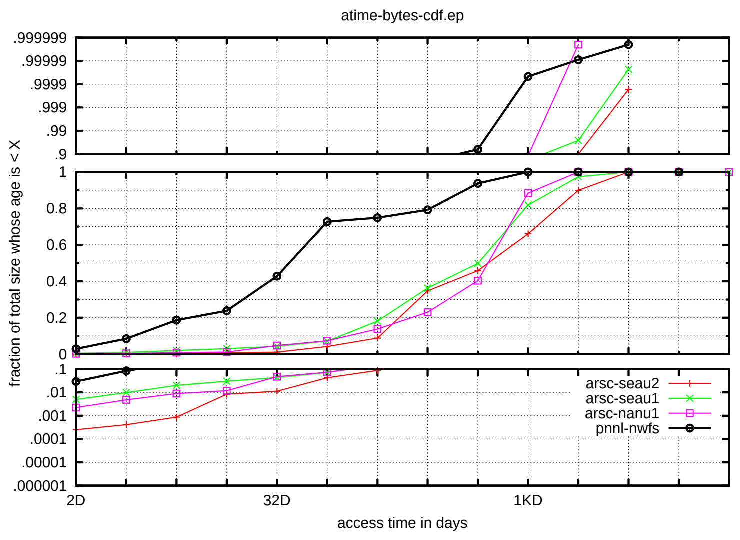


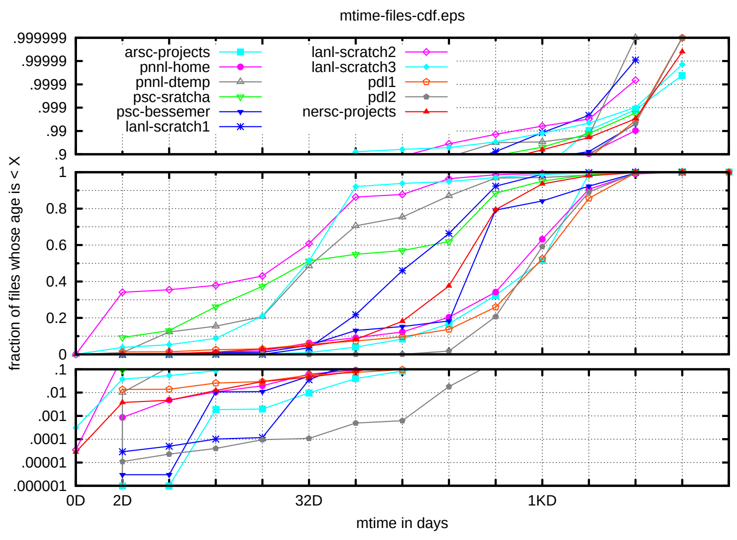
Figure 11: This figure shows a CDF of total file space (sum EOF) whose access time is less than given age, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. The access time for the median bytes, that is the 50 percentile mark, shows a wide variance across file systems. This suggests that time attribute on HPC data may be strongly influenced by the purge policy at the HPC site.

### 6.5.5 Distribution of files as a function of change time

Figure 14 shows distribution of files as a function of change time across all file systems. The file systems are split in two groups, archival and non-archival. The mean age for file change time across all file systems varies from 27 days to 754 days. Ignoring mean time from file systems that have files with future time stamps: lanl-scratch2 and lanl-scratch3, the mean value ranges from 74 to 754 days. Although there are only a few files with future timestamps; 134 in 3.3 Million files in lanl-scratch2 and 865 in 2.58 Million in lanl-scratch3. The highest mean time, 754 days, is in the pdl2 file system.

Figure 14 shows that the median change time ranges from 16 days to 1024 days which means that 50% files are older than 16 to 1024 days across all file systems. Splitting this statistic we see that for the non-archival file systems this range is 16 to 1024 days but for the archival file systems this range is only 128 to 512 days.

A closer look at the non-archival file systems shows that the change time is not older than 256 days for 50% files on 60% (6 out of 10) file systems. All these are scratch file systems. Which means that 100% of all scratch file systems have 50% files no older than 256 days in change time. And 90% of all file in 100% of all scratch file systems have files no older than 512 days in change time. Similar to modification time, this suggests that files on HPC scratch volumes are much younger as compared to project and home directories. They are younger than workstation files too. This is expected since modification time is set primarily on write and change time on write as well as on change of file attribute (see appendix A). Scratch file systems typically have a purge policy associated with them and do not allow files to become very old. Though this purge policy is not strictly enforced at all HPC sites, there is still some amount of churn in file creation and deletion. The excluded file systems, i.e. no scratch, are pdl1, pdl2, arsc-projects and pnnl-home. pdl1 and



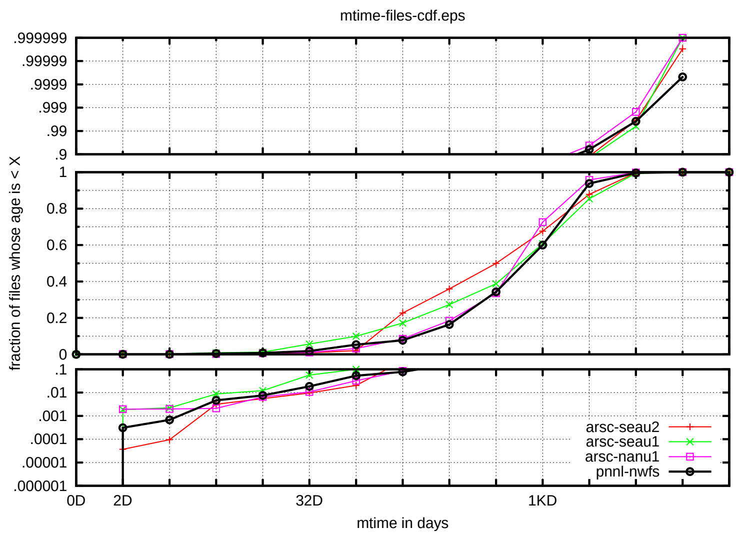


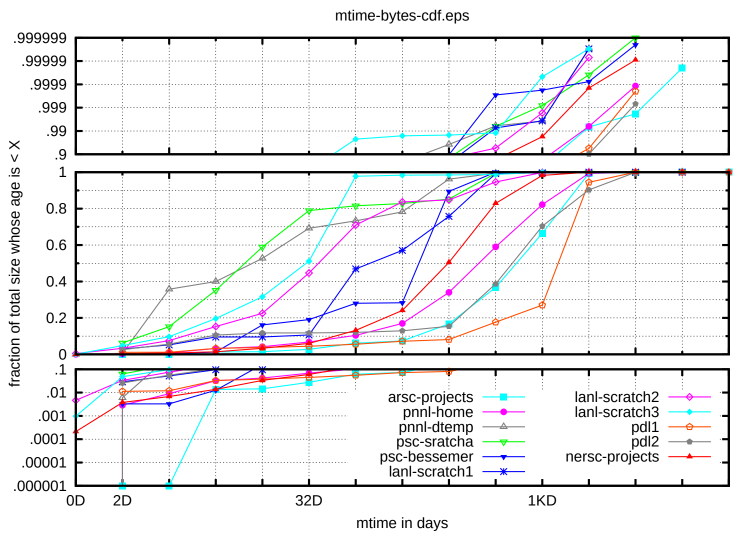
Figure 12: This figure shows a CDF of files whose modification time is less than given age, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. Scratch file systems have files that are much younger than project and home volumes. 50% files younger than 256 days and 90% files younger than 512 days as compared to the project and home volumes where files are as old as 1024 days. File modification time similar to access time show a wide variance at the median indicating influence of purge policy on file age at the HPC site. A few hundred files in several millions have future timestamps indicating application control on timestamp values.

pdl2 may have their statistics skewed due to presence of media files and trace data , which will have old change time. Also note that overall, 90% files on 70% file systems are less than 512 days old.

### 6.5.6 Distribution of total file space (sum EOF) as a function of change time

When the total file space is viewed as a distribution with respect to change time then the mean value ranges from -47 days to 638 days. Disregarding the file system with files with future timestamps: lanl-scratch2 and lanl-scratch3, the mean ranges from 57 to 638 days.

For all file systems, the median ranges from 4 days to 1024 days. Which means 50% of total space across all file systems is older than 4 to 1024 days. Figure 15 shows that for the non-archival file systems, there are two classes: scratch and non-scratch. The non-scratch file systems: pdl1, pdl2, arsc-projects and pnnl-home are clearly older than all scratch space. pdl1 and pdl2 have similar behavior, that is almost 80% to 90% of their file space is older than 256 days. On the other hand 4 out of the 6 scratch file systems have had 80% of their total space changed in the last 128 days. The remaining 2 file systems, psc-bessemer and lanl-scratch1 fall somewhere midway with 80% of their total space changed sometime in the last 512 days but about 78% of their file space not changed in the last 32 days.



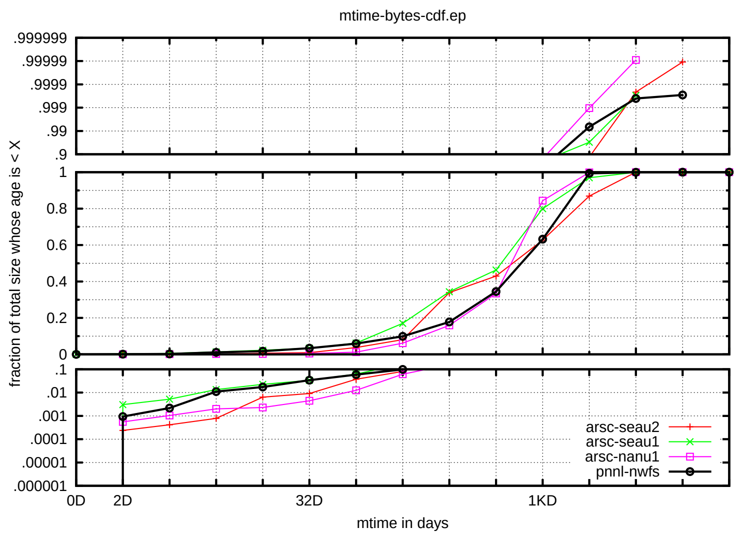


Figure 13: This figure shows a CDF of total file space (sum EOF) whose modification time is less than given age, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. There is considerably more activity, and range covered in age, in the lower (0 to 10 and upper (90 to 100) 10 percentile range for the non-archival file systems as against archival file systems.

## 6.6 Directory Size

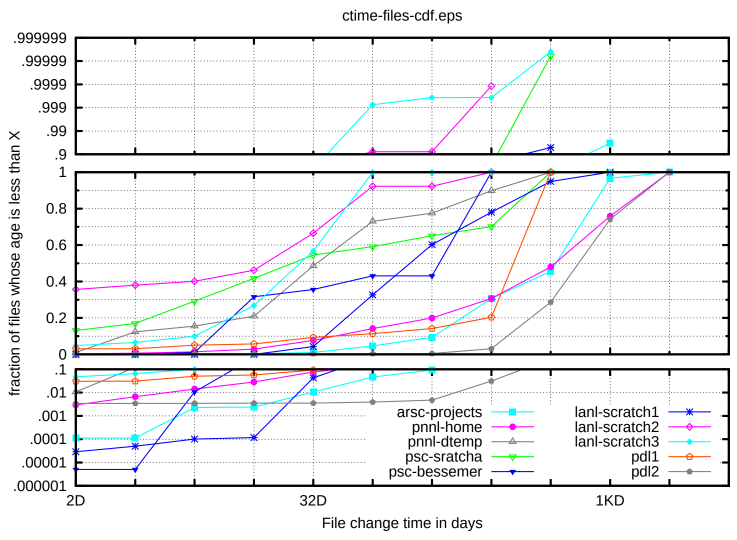
Directory size in terms of both: number of entries and size in bytes have implications on file system data structures. Parallel file systems for the largest HPC sites have the most demanding IO accesses to directories. While study of dynamic data, i.e. number of creations per second etc, is one important way to understand demand, the other is to study data at rest. Typical size of directories give insight to how big directories are in HPC environment.

Though we do not gather directory depth data, it is another important factor and something we would like to report on in a future study.

Figure 16 and 17 show distribution of entries in directories. For all file systems, the mean number of entries per directory ranges from 6 to 47. The median across all file systems, ranges from 0 to 8 entries. Douceur et al. in the 99 study found the median directory size to be 2 entries. They found that on 50% file systems the median ranges from 1 to 4 files and on 90% file systems it ranged from 0 to 7 files. This is not much different from our data, suggesting that the distribution of directories in HPC systems is not much different from workstation. In sheer size, the biggest directories at HPC sites are many orders of magnitude bigger.

Figure 16 shows that for the non-archival file systems, 90% directories are 2 to 128 entries in size, but the maximum number of entries in a directory can be as large as 128 K. Graph in Figure 17 shows that only 10% entries are in directories of size 1 to 128 entries. This implies that most entries are in a few large directories.

For the archival file systems, 90% directories have between 8 to 64 entries per directory, a shorter range but not significantly different from the non-archival file systems.



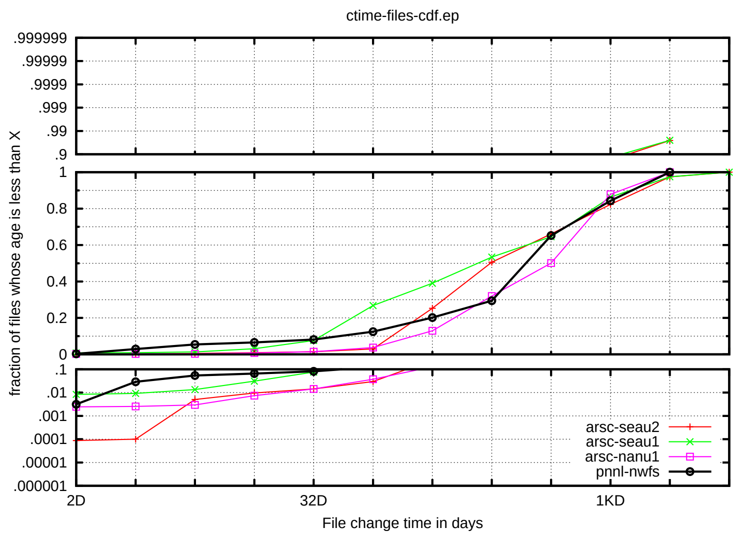


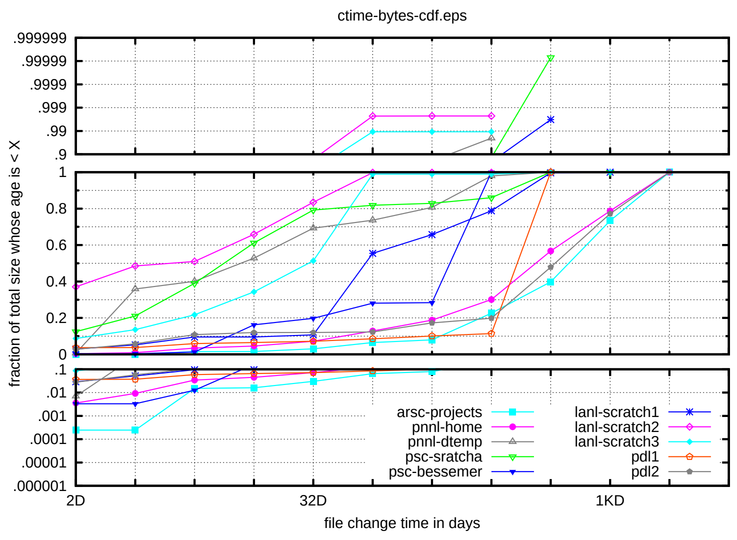
Figure 14: This figure shows a CDF of files whose change time is less than given age, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. NERSC does not store ctime in its backup files which were used to generate these histograms. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. Though the median ranges from 16 days to 1024 days 100% of all scratch file systems have 50% files no older than 256 days in change time. And 90% of all file in 100% of all scratch file systems have files no older than 512 days in change time. Similar to modification time analysis, this suggests that scratch files are younger in age as compared to projects, and home volumes on HPC sites. They are also younger than our workstation files from pdl1 and pdl2.

Figure 17 shows fraction of entries in a directory of given size. The median point for the non-archival file systems is in the range 8 to 1024. Similarly for the archival file systems the range for the median values is size 16 to 1024, which is not much different. But if exclude the pnnl-nwfs from the archival file systems, then the range for median is narrower, 128 to 1024. That is, 75% of the archival file systems have a 50% entries in directories whose size vary from 0 to 1024 entries.

Another way to study directories is by its size in bytes. The mean size of directories for all file systems ranges from 4 KB to 39 KB. Figure 18 shows that the smallest directory size in bytes is 4K. The median size is 4K to 8K. For the non-archival file systems, directory size ranges from 4K to 128M. For the archival file systems the size ranges from 4K to 4M.

Across all file systems, 80% to 99.5% directory are between 4K to 8K in size. In the archival file systems, almost 94% to 99.5% directories are in the range of 4K to 8K in size. In 12 out of all 15 file systems (80% of studied file systems), 90 to 99.5% are between 4K to 8K in size. This is by far the most pronounced behavior, with only a marginally small set of directories, in almost all file systems, bigger than 8K in size.

Across all file systems, the total covered range of directories varies by a factor of 15, ranging from 4K to 128M. In terms of entries the total covered range varies from 1 to 256K entries, 18 binary orders of magnitude.



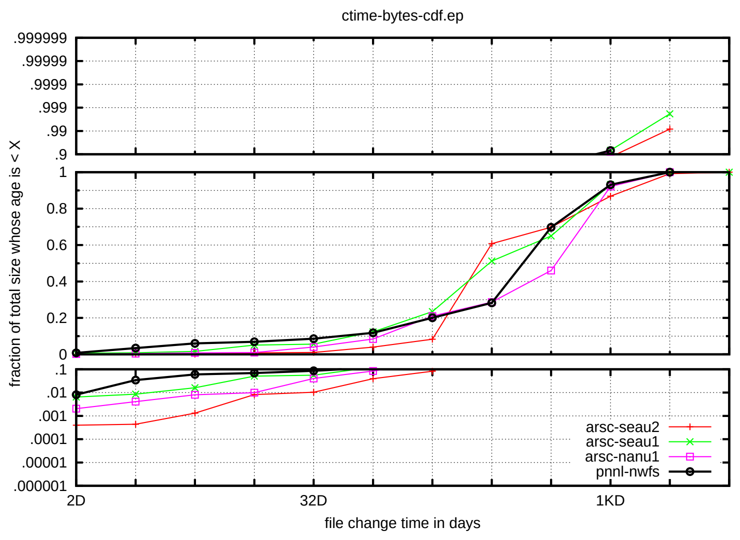


Figure 15: This figure shows a CDF of total file space (sum EOF) whose change time is less than given age, across all file systems. On the left is a graph for 10 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. NERSC does not store ctime in its backup file which was used to create these histograms. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. For the non-archival file systems, there are two classes: scratch and non-scratch. The non-scratch file spaces are clearly older. pdl1 and pdl2 have similar behavior, that is almost 80% to 90% of their file space is older than 256 days. On the other hand 4 out of the 6 scratch file systems have had 80% of their total space changed in the last 128 days.

## 7 Conclusion

We collected statistics from 15 file systems, of which 13 belong to a high end computing environment and 2 belong to departmental file servers at CMU. We collected statistics on file size, age, directory size in bytes and entries and capacity used of files. This is the first ever study on files in a HEC environment and also first ever study to report statistics on such large file systems.

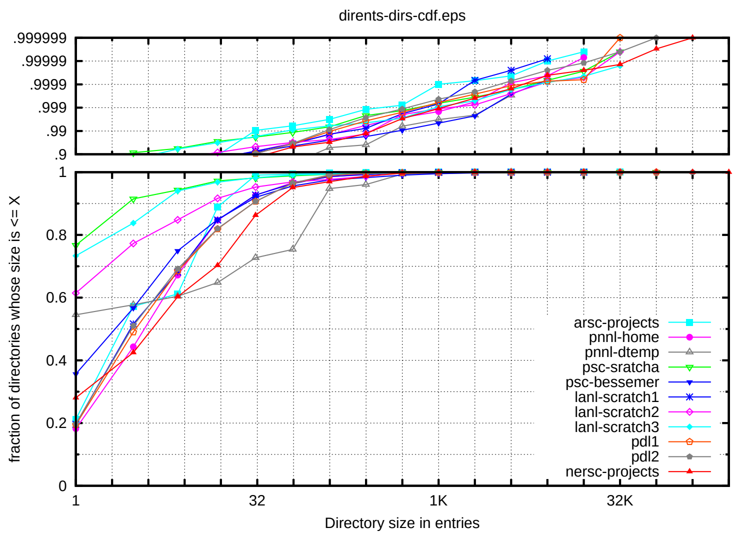
We report on data from file systems of size several hundred Tera bytes and files within file systems whose size is greater than a Tera byte. We compared our data with previous studies [6, 19]. Our mean file size is larger than previously reported [6, 19] and the range for median is bigger than those previously reported. We found directories to be very typical in size.

Our analysis on file overhead shows that positive overhead may not always be obvious from 'stat' like system calls since some file systems are not reporting all the overhead.

We also analyze file age characteristics and find that age of files differ significantly depending on whether they are from scratch file systems or belong to project or home volumes.

We further classify our file systems as archival and non-archival and study their characteristics separately. We received valuable statistics from ARSC (see 5.4), from its archival file systems running SAM-QFS, that enabled this study.

Our primary contribution is the size statistics from a number of different HEC sites. We have already made this data publicly available on our website ¡http://www.pdsi-scidac.org/fsstats/¿. Our goal in fact, is to enable easily sharing contributions from everyone.



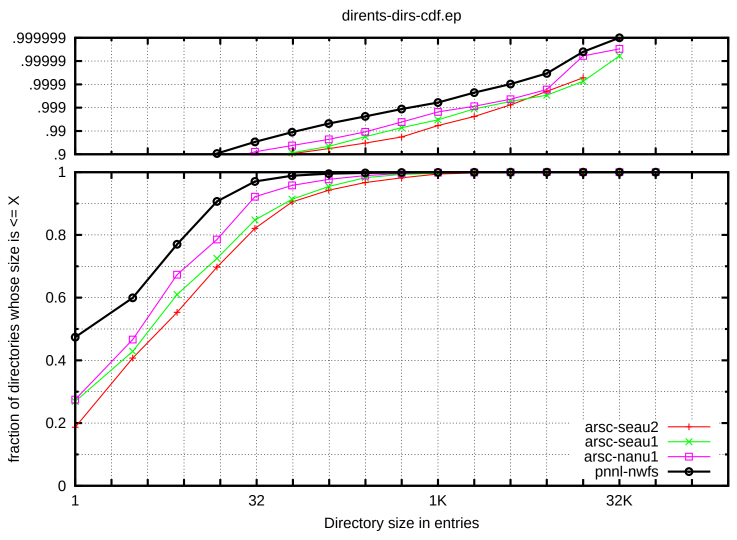


Figure 16: This figure shows a CDF of directories whose size in entries is less than given size, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is linear scale from 0 percentile to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. The median across all file systems ranges from 0 to 8 entries. 50% directories are 0 to 8 entries in size. 90% directories are 0 to 128 entries in size

.

Our second contribution is analysis of data at rest from HPC sites and its comparison with previously published papers on user workstations and departmental file servers.

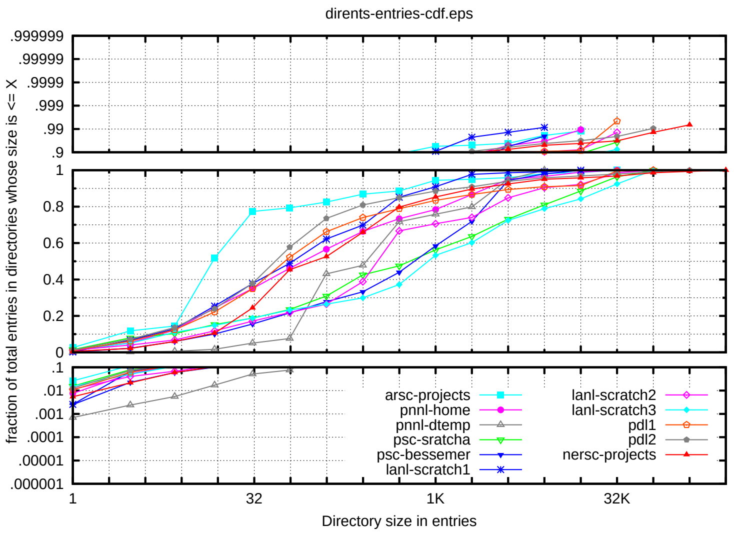
Our third contribution is the analysis of file positive and negative overhead that has so far not been done in any previous study.

## 8 Future Work

There are a number of areas concerning file statistics that were not addressed in this paper. Our future research will focus on completing study in these areas. Some of the work that we plan to achieve in future are:

- Working at building a repository that enables people to publicly share file statistics information. We already have a web site where people are contributing data. Since statistics change over time, we also want to enable generating data from the same file systems over a period of time and comparing with older statistics, i.e. we want to generate longitudinal stats from same file systems. We already have statistics in our website, that were gathered from the same file system separated by a period of weeks to months. Our future study will present analysis from statistics spaced apart in months to years and show how particular file systems are changing in time.

- Studying dynamic workloads and usage patterns are an important aspect to understanding characteristics of files and file systems. In HPC environment, data is often generated in one place, like the scratch file system, and makes it way to an intermediate storage in a second file system, before getting



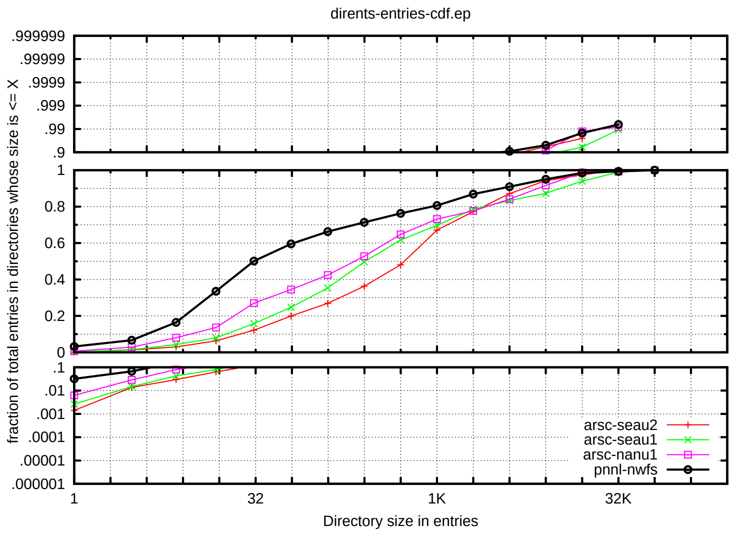


Figure 17: This figure shows a CDF of total entries in directories of given size, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. Across all file systems, 50% entries are in directories whose size vary from 0 to 1K entries. And 90% entries are in directories whose size varies from 0 to 32K entries, though directories can be as big as 256K entries.

archived in a third. In future we intend to study this data flow where we trace data from birth to long term rest.

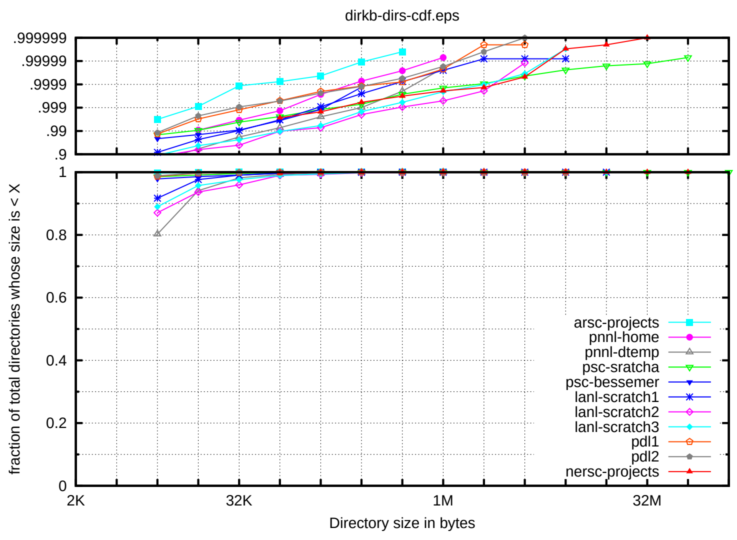
- Though we did not focus on curve fitting in this paper, we would at least like to curve fit the file size distribution. Different studies in different periods of time have curve fitted file size distributions [17, 6, 8] and have actually found different curves to be best fit for the data they studied. Satyanarayanan found the file size distribution best fit by hyperexponential distribution, while Douceur et al. found the log-normal distribution to be the best fit. A 2002 study by Kylie M. Evans and Geoffrey H. Kuenning [8] showed that even log-normal is not a good fit for file size and a more complex lambda distributions provide the best fit. We would like to curve fit the file size distribution and see what is the best analytical function that represents file size distribution on HPC file systems.

- Though we studied directory size in terms of entries and bytes, since directories enable hierarchies, an important aspect of studying directories is the directory depth distribution. We have left this exercise for a future study.

## Appendix

## A Interpretation of values returned by the POSIX 'stat' system call

The following is a description of some of the values returned by the POSIX stat system call. The description below has been taken directly from the Linux Manpage (2). We provide a description of only those fields



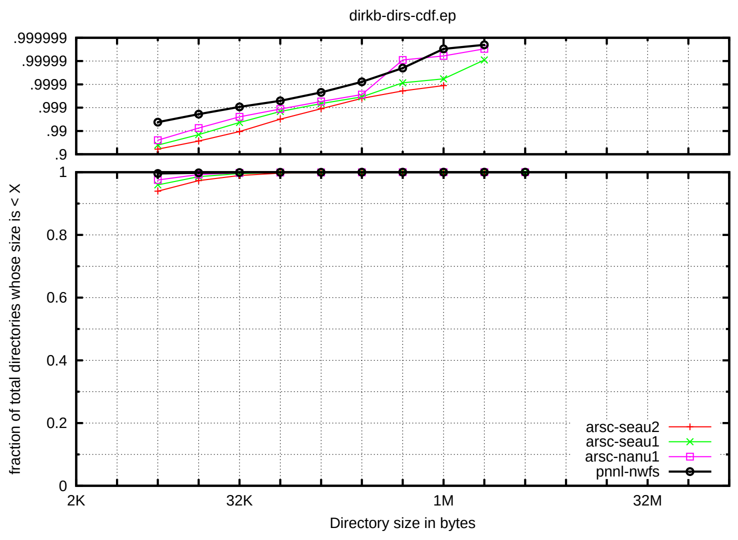
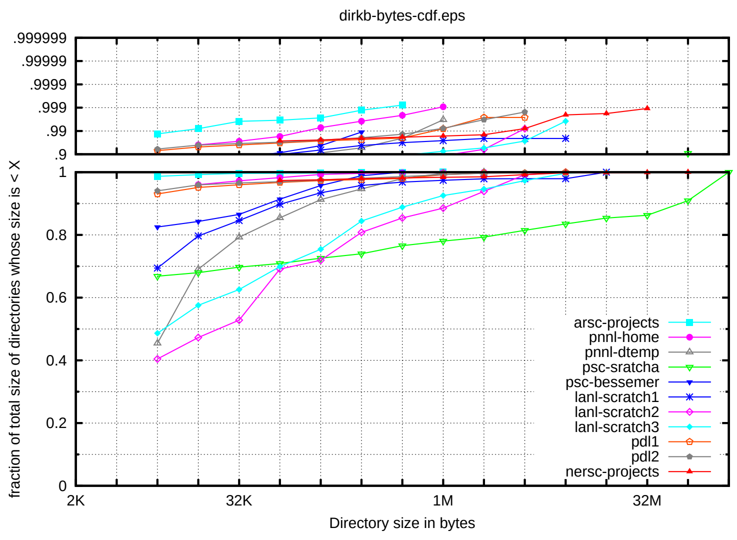


Figure 18: This figure shows a CDF of directories whose size in bytes is less than given size, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 2 sections. It is log scale from 0 percentile to 100 percentile in section 1 and log scale from 90 to 99.9999 percentile in section 2. The Legends are explained in section 2. Across all file systems, 80% to 99.5% directory are between 4K to 8K in size. Only a marginally small fraction is greater than 8K in size though directories can be as big as 128M in size. The median size is 4K to 8K and the mean size ranges from 4K to 39K.

that were used by the fsstats program.

Note that not all of the file systems implement all of the time fields. Some file system types allow mounting in such a way that file accesses do not cause an update of the st_atime field. (See 'noatime' in mount(8).)

- st_ino Inode number.
- st_nlink number of hard links.
- The st_size field gives the size of the file (if it is a regular file or a symbolic link) in bytes. The size of a symlink is the length of the pathname it contains, without a trailing null byte. This field was used to calculate the file size (EOF).
- The st_blocks field indicates the number of blocks allocated to the file, 512-byte units. (This may be smaller than st_size/512, for example, when the file has holes). This field was used to calculate the file capacity used.
- The field st_atime is changed by file accesses, e.g. by execve(2), mknod(2), pipe(2), utime(2) and read(2) (of more than zero bytes). Other routines, like mmap(2), may or may not update st_atime.
- The field st_mtime is changed by file modifications, e.g. by mknod(2), truncate(2), utime(2) and write(2) (of more than zero bytes). Moreover, st_mtime of a directory is changed by the creation or deletion of files in that directory. The st_mtime field is not changed for changes in owner, group, hard link count, or mode.



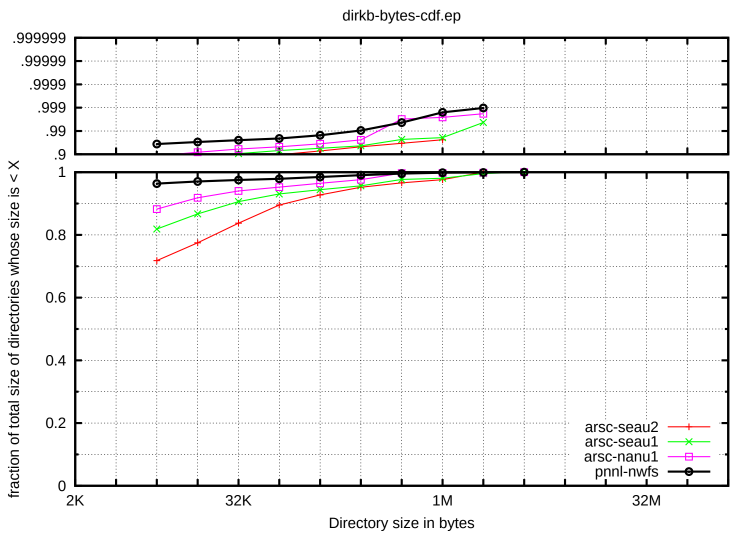


Figure 19: This figure shows a CDF total size (sum EOF of directories) in directories of given size, across all file systems. On the left is a graph for 11 non-archival file systems. These are mainly scratch, project and home volumes from HPC sites and two volumes from departmental file servers at CMU. On the right is a graph of 4 archival file systems, 3 from ARSC (see section 5.4) and 1 from PNNL (see section 5.3) The X axis is log scale of base 2. The Y-axis is split in 3 sections. It is log scale from 0.0001 percentile to 10 percentile in section 1. Linear from 0 to 100 percentile in section 2 and log scale from 90 to 99.9999 percentile in section 3. The Legends are explained in section 2. Median value ranges from 4K to 32K in bytes, i.e., 50% bytes are in directories whose sizes range from 4K to 32K. 90% bytes are in directories of size 4K to 2M.

- The field st_ctime is changed by writing or by setting inode information (i.e., owner, group, link count, mode, etc.).

## References

[1] Lustre File system: High-Performance Storage Architecture and Scalable Cluster File System. http://www.sun.com/software/products/lustre/docs/lustrefilesystem_wp.pdf.

[2] SFS2008 and SFS97_R1, Network File System Benchmark, Open Systems Group, Standard Performance Evaluation Corporation. http://www.spec.org/benchmarks.html#nfs, 2008.

[3] Agrawal, N., Bolosky, W. J., Douceur, J. R., and Lorch, J. R. A Five-Year Study of File-System Metadata. In *Proc. of the FAST '07 Conference on File and Storage Technologies* (San Jose CA, Feb. 2007).

[4] Baker, M. G., Hartman, J. H., Kupfer, M. D., Shirriff, K. W., and Ousterhout, J. K. Measurements of a Distributed File System. In *Proc. of 13th ACM Symposium on Operating Systems Principles (SOSP '91)* (Pacific Grove CA, Oct. 1991).

[5] Bennett, J. M., Bauer, M. A., and Kinhlea, D. Characteristics of Files in Nfs Environments. In *Proceedings of the 1991 ACM SIGSMALL/PC symposium on Small systems* (Toronto, Ontario, Canada, 1991).

[6] DOUCEUR, J. R., AND BOLOSKY, W. J. A Large-Scale study of File System Contents. In *Proc. of Joint International Conference on Measurement and Modeling of Computer Systems* (Atlanta, WA, 1999).

[7] ELARD, D., LEDLIE, J., MALKANI, P., AND SELTZER, M. Passive Nfs Tracing of Email and Research Workloads. In *Proc. of the FAST '03 Conference on File and Storage Technologies* (San Francisco CA, Mar. 2003).

[8] EVANS, K. M., AND H.KUENNING, G. A Study of Irregularities in File size Distributions. In *Proc. of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)* (san Diego, CA, 2002).

[9] GRIBBLE, S. D., MANKU, G. S., ROSELLI, D., BREWER, E. A., GIBSON, T. J., AND MILLER, E. L. Self Similarity in File Systems. In *Proc. of Joint International Conference on Measurement and Modeling of Computer Systems* (Madison, WI, 1993).

[10] HITZ, D., LOU, J., AND MALCOLM, M. File System Design for an nfs File Server Appliance. In *Proc. of the Winter USENIX Conference* (San Francisco, CA, 1994).

[11] IRLAM, G. Unix File Size Survey. http://www.gordoni.com/ufs93.html, 1993.

[12] MENDEL, R., AND OUSTERHOU, J. K. The design and Implementation of a Log-Structured File System. *TOCS 10, Number 1, P. 26-52* (1992).

[13] MULLENDER, S. J., AND TANENBAUM, A. S. Immediate Files. *Software-Practice Experience 14*, 4 (1984).

[14] NELSON, M. N., WELCH, B. B., AND OUSTERHOUT, J. K. Caching in the Sprite Network File System. *TOCS 6, Number 1, P. 134-154* (1988).

[15] OUSTERHOUT, J. K., COSTA, H. D., HARRISON, D., KUNZE, J. A., KUPFER, M., AND THOMPSON, J. G. A Trace Driven Analysis of the Unix 4.2 bsd File System. In *Proc. of 10th ACM Symposium on Operating Systems Principles (SOSP '85)* (Orcas Island WA, Dec. 1985).

[16] ROSELLI, D., LORCH, J. R., AND ANDERSON, T. E. A Comparison of File System Workloads. In *Proc. of 2000 USENIX Annual Technical Conference* (san Diego, CA, 2004).

[17] SATYANARAYANAN, M. A study of File Sizes and Functional Lifetimes. In *Proc. of 8th ACM Symposium on Operating Systems Principles (SOSP '81)* (Pacific Grove, WA, Dec. 1981).

[18] SCHMUCK, F., AND HASKIN, R. GPFS: A Shared-Disk File System for Large Computing Clusters. In *Proc. of the FAST '02 Conference on File and Storage Technologies* (Monterey CA, Jan. 2002).

[19] SIENKNECHT, T. F., FRIEDRICH, R. J., MARTINKA, J. J., AND FRIEDENBACH, P. M. The Implications of Distributed Data in a Commercial Environment on the Design of Hierarchical Storage Management. *Performance Evaluation 20(1-3), P. 3-25* (1994).

[20] TANENBAUM, A. S., HERDER, J. N., AND BOS, H. File Size Distribution on UNIX Systems - Then and now. *ACM SIGOPS Operating Systems Review 40*, 1 (2006).

[21] VOGELS, W. File System Usage in Windows nt 4.0. In *Proc. of 17th ACM Symposium on Operating Systems Principles (SOSP '99)* (Kiawah Island Resort SC, Oct. 1999).

[22] Welch, B., Unangst, M., Abbasi, Z., Gibson, G., Mueller, B., Small, J., Zelenka, J., and Zhou, B. Scalable Performance of the Panasas Parallel File System. In *Proc. of the FAST '08 Conference on File and Storage Technologies* (San Jose CA, Feb. 2008).