HPC File Size Distributions

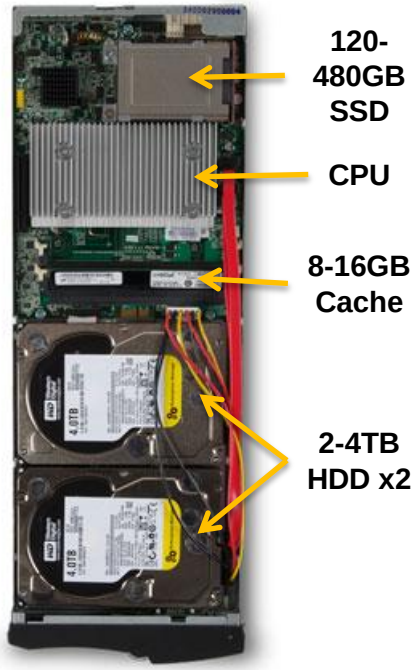
OPTIMIZING A HYBRID SSD/HDD HPC STORAGE SYSTEM BASED ON FILE SIZE DISTRIBUTIONS

MSST 2013

BRENT WELCH, CTO

HYBRID HHD/SSD OSD

- **OSD – Object Storage Device**
  - Manages a collection of variable sized, byte addressed objects
  - Implemented much like a traditional file system that manages inodes, blocks, and snapshots
- **Panasas OSD was originally a PC with 2 SATA drives**
  - 2009 we had a model with 1 SATA, 1 SSD
  - 2012 we introduced a model with 2 SATA, 1 SSD
- **Question, what's the right stuff to store on the SSD?**
  - We suspected there were a lot of small files that we could cheaply store there, and use HDD for large data extents
  - How much SSD do we need?



ActiveStor 14 Storage Blade

FILE SIZE STUDY - FSSTATS

panasas

- **fsstats is a perl script that walks a file system and collects information about file length, file capacity, directory sizes, file name lengths, use of symlinks**
  - www.pdsi-scidac.org/**fsstats**/
- **Today we are primarily concerned with the file size and file capacity histograms**

- **<u>Most files are small, but most of the capacity is in large files</u>**
  - Because a 1 GB file is a million times bigger than a 1KB file
- **In this presentation:**
  - 1KB is 1024 bytes (X axis of the charts is in KB)
  - 1MB is 1024 KB, or 1,408,576
  - 1GB is 1024 MB, or 1,073,741,824
  - 1TB is 1024 GB, or 1,099,511,627,776 (“one million megabytes”)

PRIOR WORK

- There are several old studies of desktops and workstations
- The most complete recent one is from Microsoft, but is still a survey of desktops

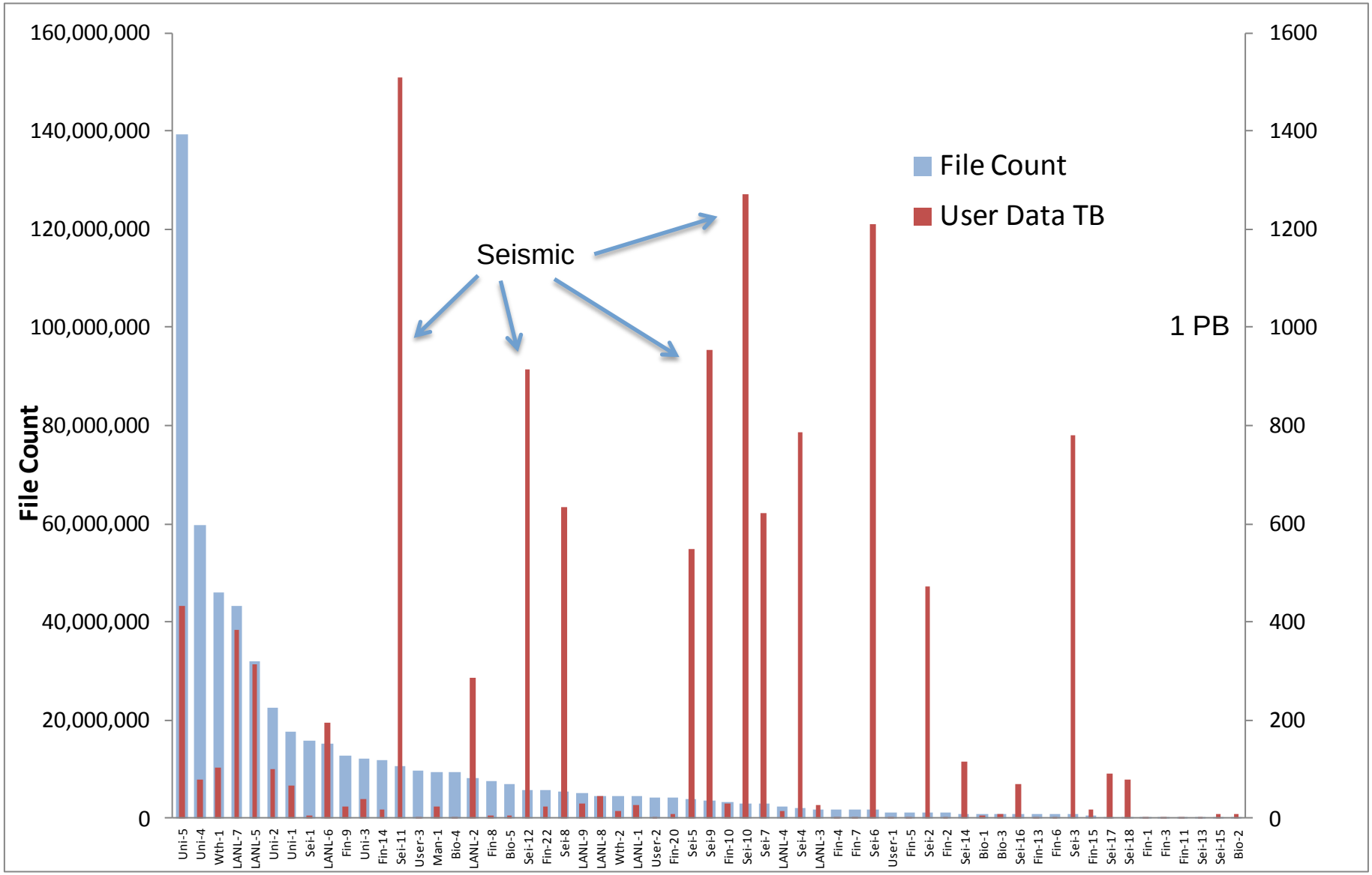
- The work presented today focuses on file systems that are found in large HPC environments

- We got this data simply by asking customers to run fsstats

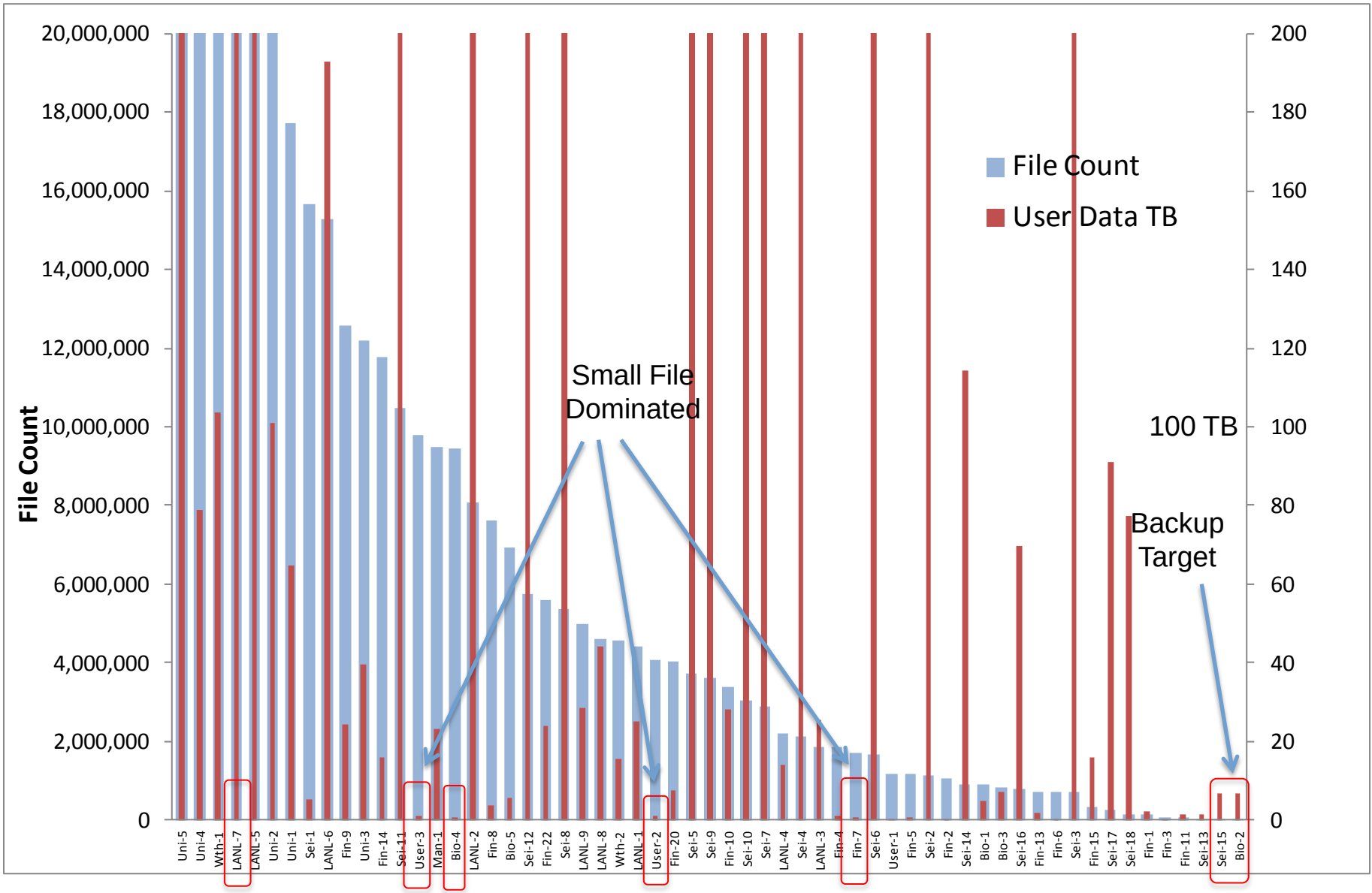
OVERVIEW OF THE DATA

- **600 million files total, 12.4 PB of user data**
  - 10's to 100's of millions of files in each system
  - Individual files larger than 1TB
  - Lots and lots of very small files
- **65 different "file systems" from 13 customers**
  - Most of these file systems are Panasas Volumes that may or may not be sharing physical OSD with other volumes – we can't tell from the data
    - Panasas Volumes are subtrees of the namespace with quota and managed by different metadata services
    - Volumes may be co-mingled with each other in a physical storage pool we call a BladeSet
  - Some of these file systems are from Lustre and GPFS systems
- **The datasets have identifiers that loosely classify them**
  - FIN – Financial market modeling and risk analysis
  - BIO – Bioinformatics associated with sequencing instruments
  - UNI – Univerisity (mixed workloads)
  - USER – Home directories, log files, shared binaries
  - LANL – Los Alamos National Labs
  - SEI – Seismic data processing

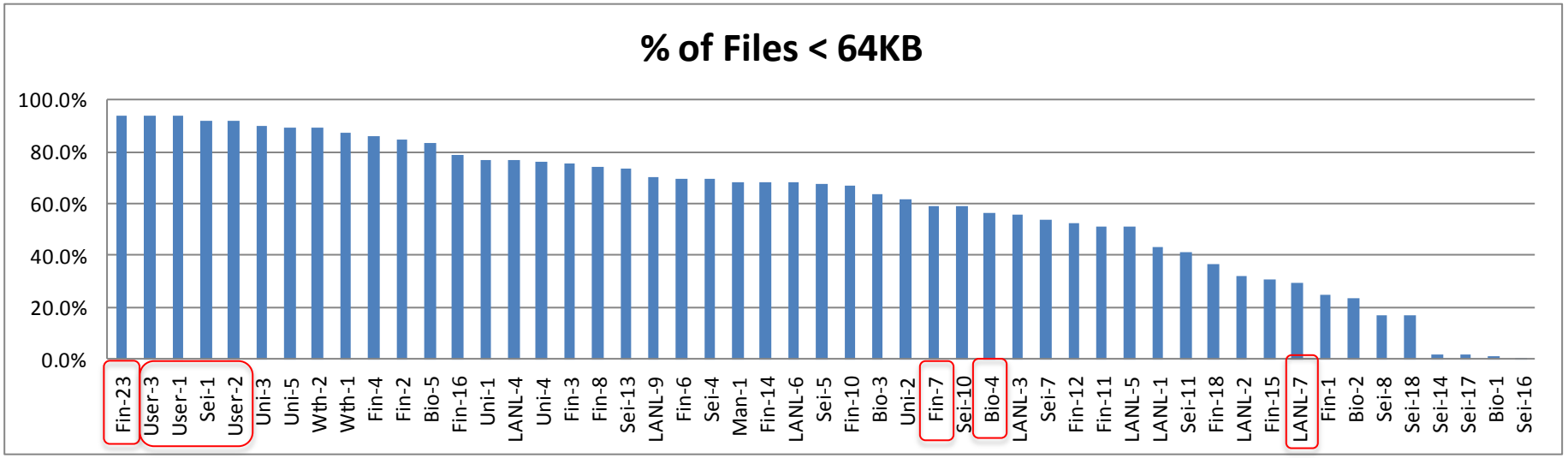
FILE COUNTS AND USER DATA TB

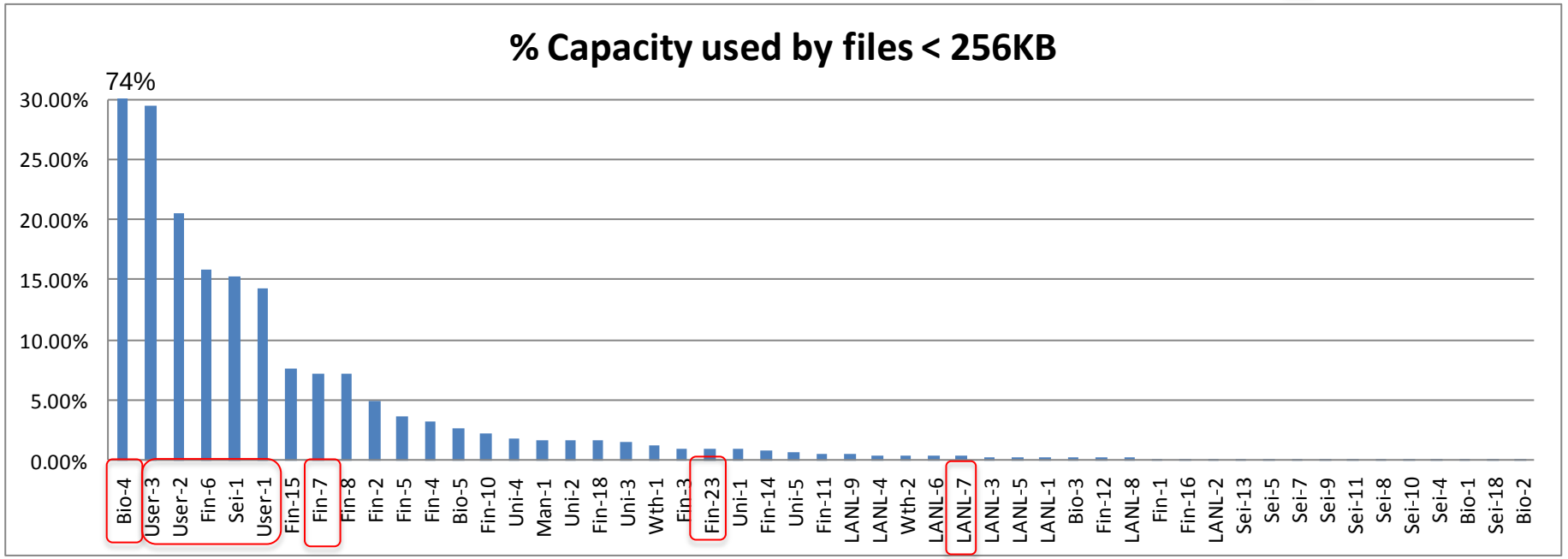


FILE COUNTS AND USER DATA TB (ZOOM IN)



MANY SMALL FILES, NOT MUCH CAPACITY





WHAT MAKES 1TB

- **1TB is 1 billion kilobytes**
  - 256 million 4K disk sectors or SSD pages
- **single-platter disk drive ($70)**
  - 2 surfaces, each 500 GB
  - 350,000 tracks/surface
  - Each track on average 1.5 MB in capacity
  - 120+ MB/sec streaming bandwidth at 7200 RPM
    - About 2.3 hours to read the complete device
- **MLC SSD ($1000 to $3,000)**
  - 64Gb NAND FLASH die @ 22 nanometer process
  - 8 die/package => 64 GB
  - 16 packages per SSD => 1TB
  - 3,000 program/erase cycles
    - 1.5 PB lifetime writes after wear leveling (only 1 month at 500 MB/sec)
    - ~1 complete device write each day for 3 years (only 12 MB/sec)

HYBRID STORAGE SYSTEMS

panasas

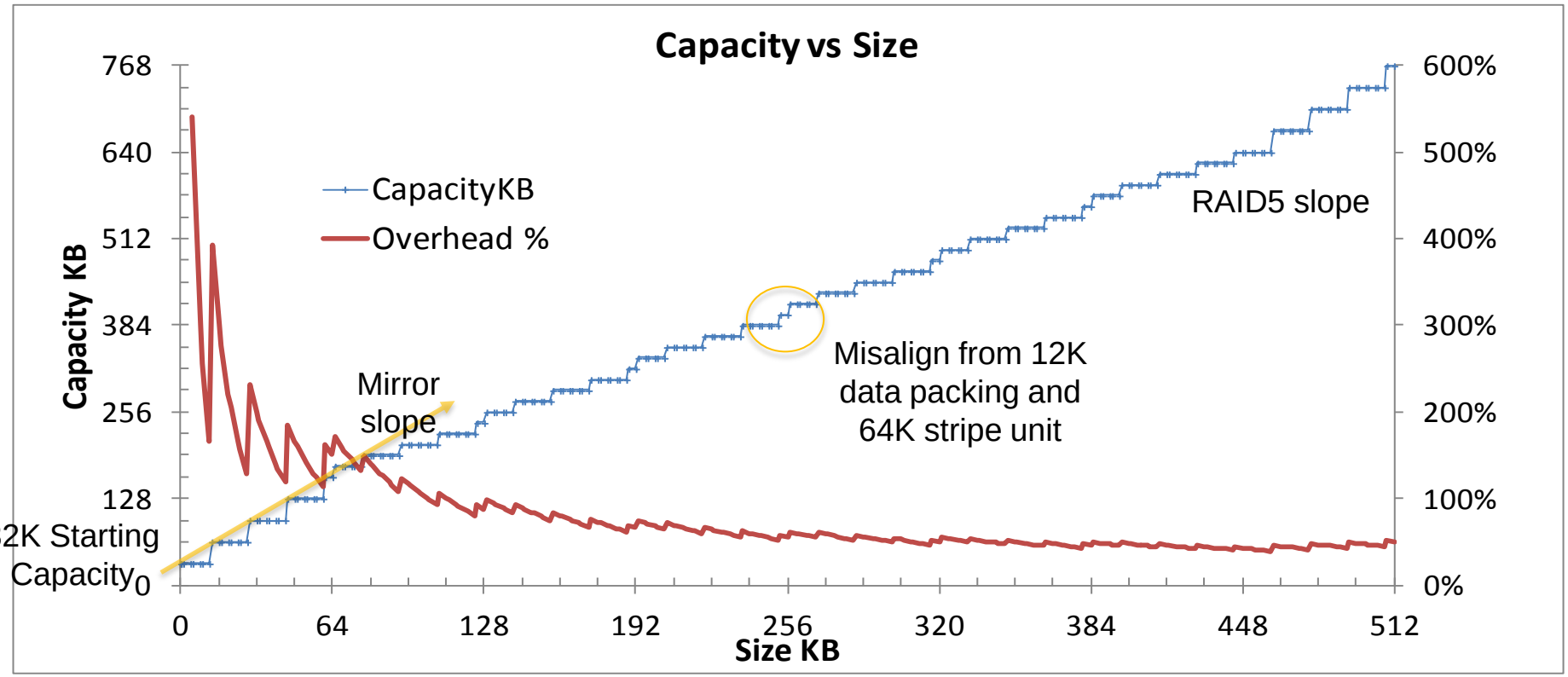
- **How do you combine cheap HDD storage with expensive SSD storage?**
- **How much SSD do you need?**
- **Will it last? (Come to this evening's talk)**
- **Will the HDD be fast enough?**
- **What kind of data structures and algorithms do you need?**

PANASAS APPROACH

- **We propose putting small files and metadata onto SSD, and large file data extents onto HDD**
  - First did this in 2009, recently released 2nd generation AS-14 hybrid
- **File system data structures are seek-intensive**
  - B-Trees
  - Allocation Maps
  - Object descriptors
  - Indirect block pointers
- **Small files are seek intensive**
  - lots of work just to read a small amount of data
- **Suppose large files are extent allocated, and cost "1" seek to get a large chunk of data off of a hard drive**
  - Large data tracks live permanently on HDD and prolong SSD lifetime
- **OK, But…**
  - The way we store small files is pretty expensive in capacity overhead

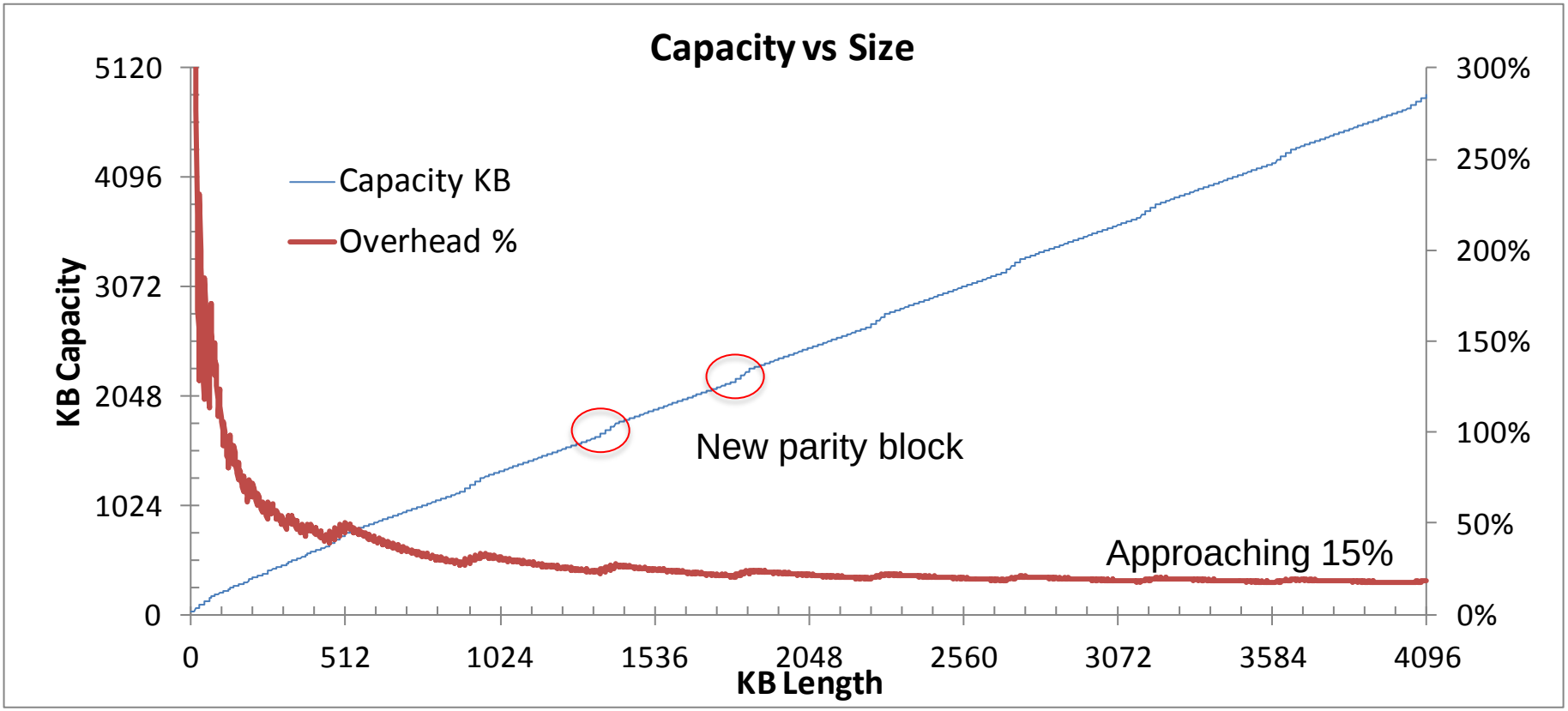
CAPACITY OVERHEAD OF SMALL FILES

- **File System Overhead**
  - PanFS uses a full 16K file system block for object descriptor
  - But packs 12K of object data into the object descriptor
  - PanFS mirrors small files in two component objects on different OSD
  - Larger files (> 64K) are striped RAID-5 with 64K stripe unit



CAPACITY VS FILE SIZE

- **For files > 1MB, capacity overhead approaches 15%**
  - As we shall see, capacity is dominated by files like this
  - Any RAID system has capacity overhead from parity components



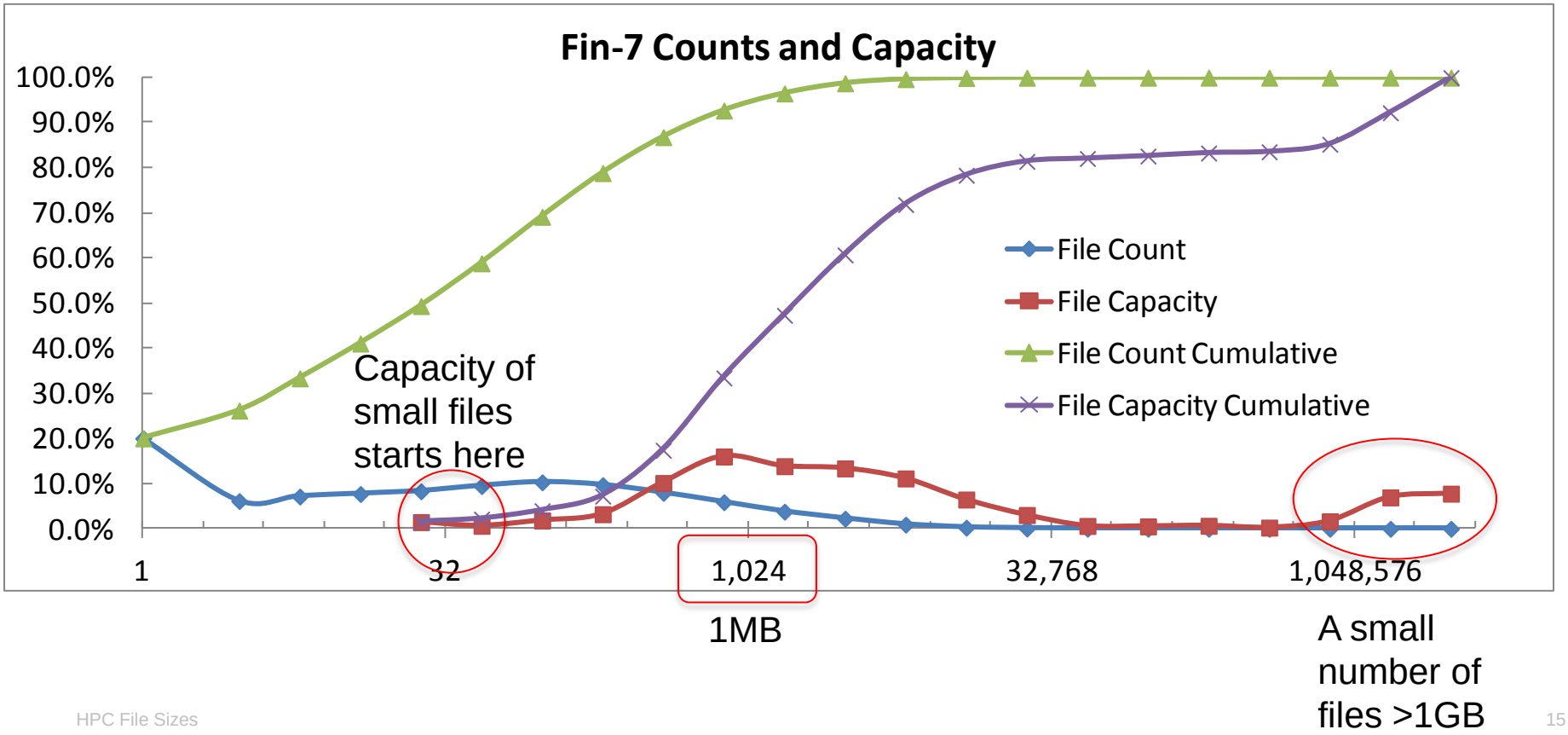
FILE SIZE VS CAPACITY, ACTUAL DATA

panasas

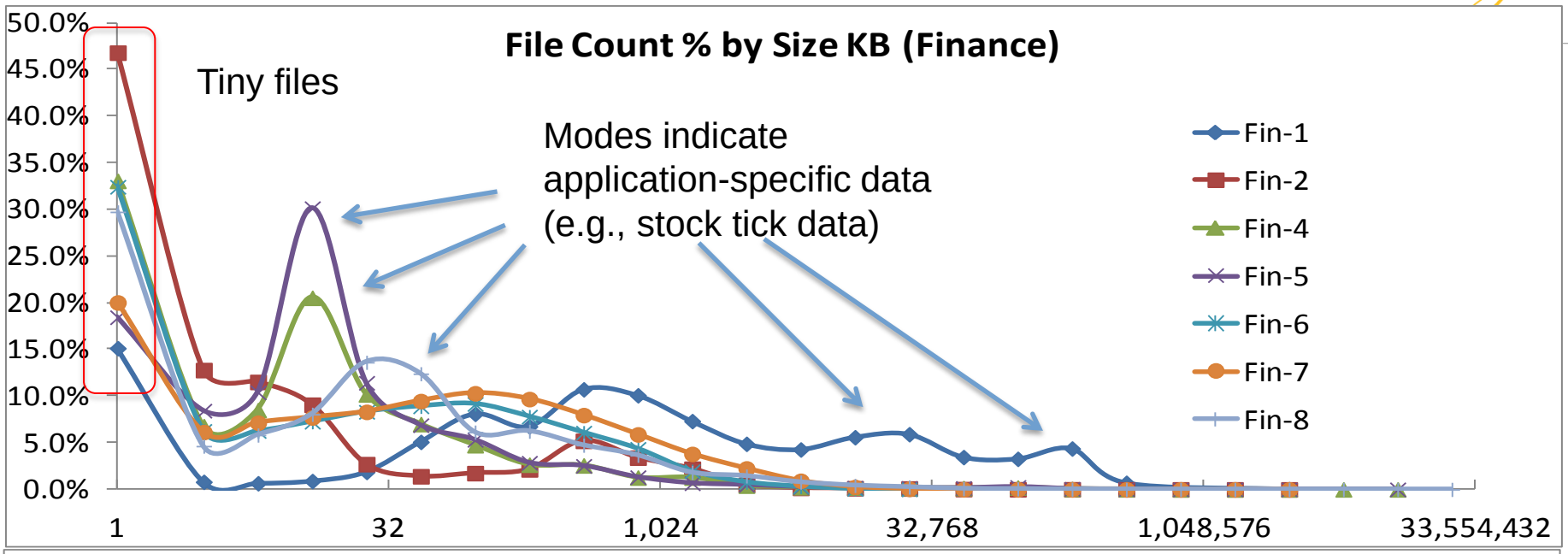
- **Now that we've worried about the overhead of small files, let's look at the data to see if it matters**
- **And, let's try to get an idea of how much file system overhead and small stuff there is to see if it can cheaply fit onto SSD**
- **The datasets have identifiers that loosely classify them into**
  - FIN – Financial market modeling and risk analysis
  - BIO – Bioinformatics associated with sequencing instruments
  - UNI – Univerisity (mixed workloads)
  - USER – Home directories, log files, shared binaries
  - LANL – Los Alamos National Labs
  - SEI – Seismic data processing

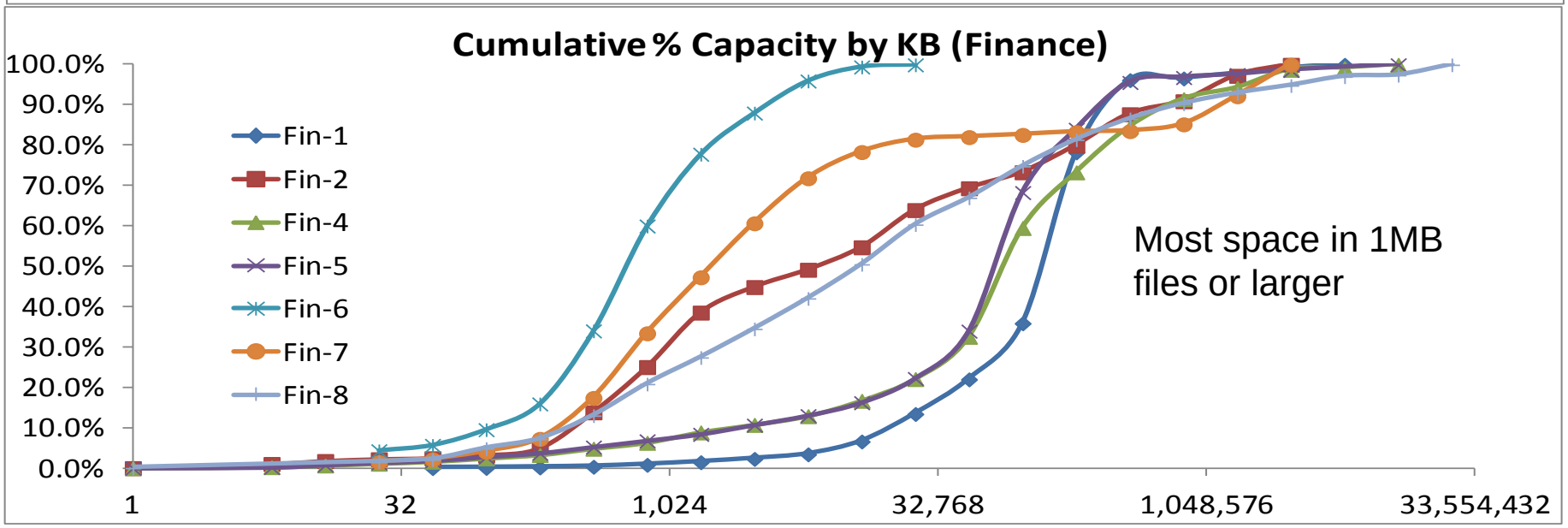
FIN-7

- **"Traditional" mix with plenty of small files**
  - 1.7 million files, 665 GB
  - Most files are less than 1MB in length (the green curve)
  - Most capacity in files 4MB or larger (the purple curve)
  - Mode at about 1GB from about 100 large DB files is 20% of the capacity



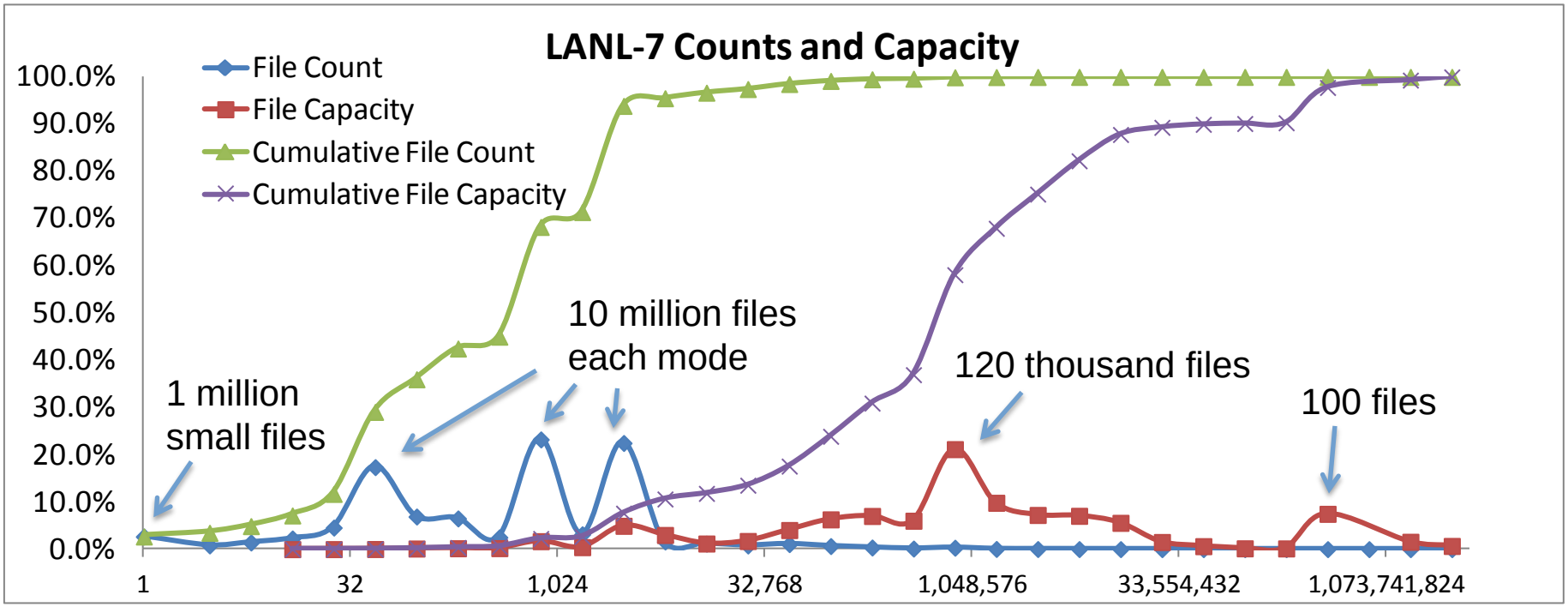
FINANCE DISTRIBUTIONS



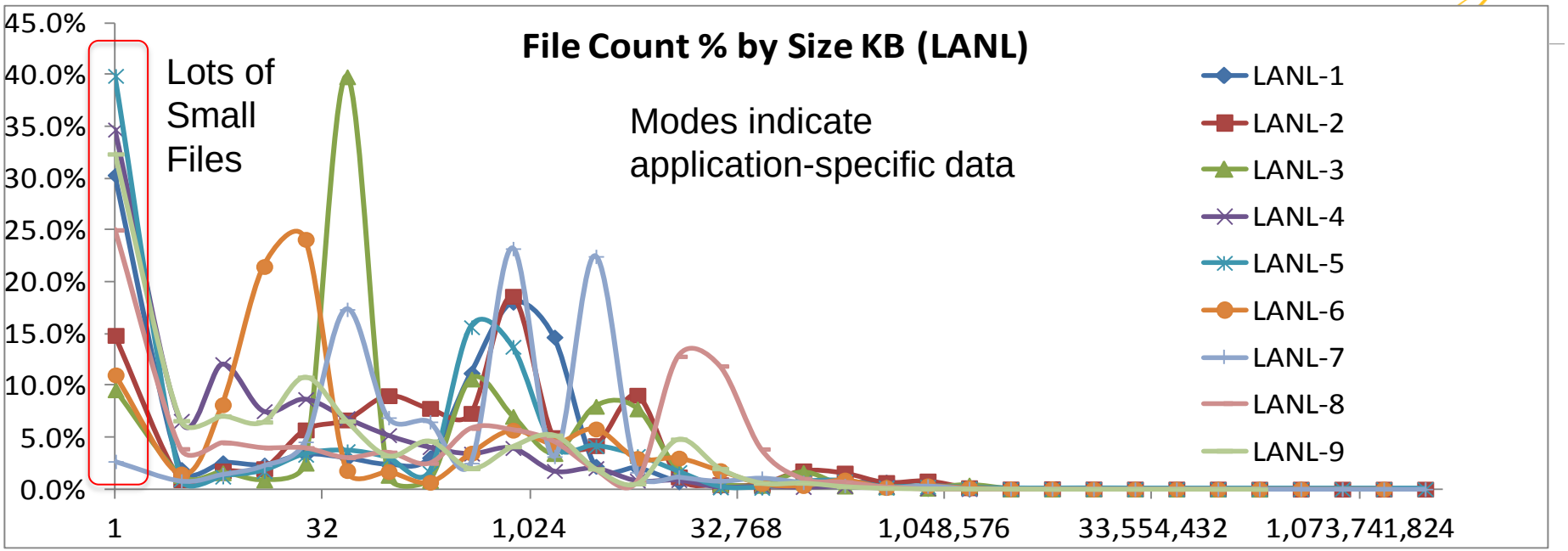


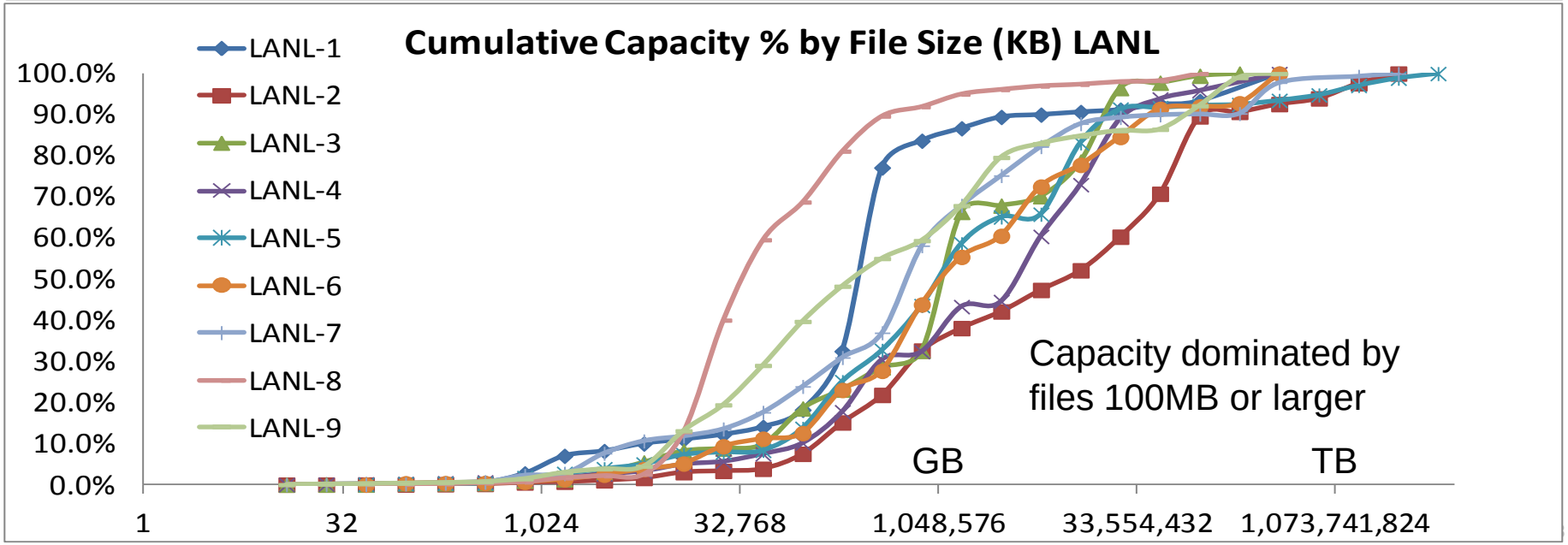
LANL-7 (LARGEST OF LANL DATASETS)

- **43 Million files, 382 TB**
  - A million tiny files
- **Modes at 64K, 1MB, 4MB, 1GB, 400 GB file sizes**
- **Largest file 4TB**



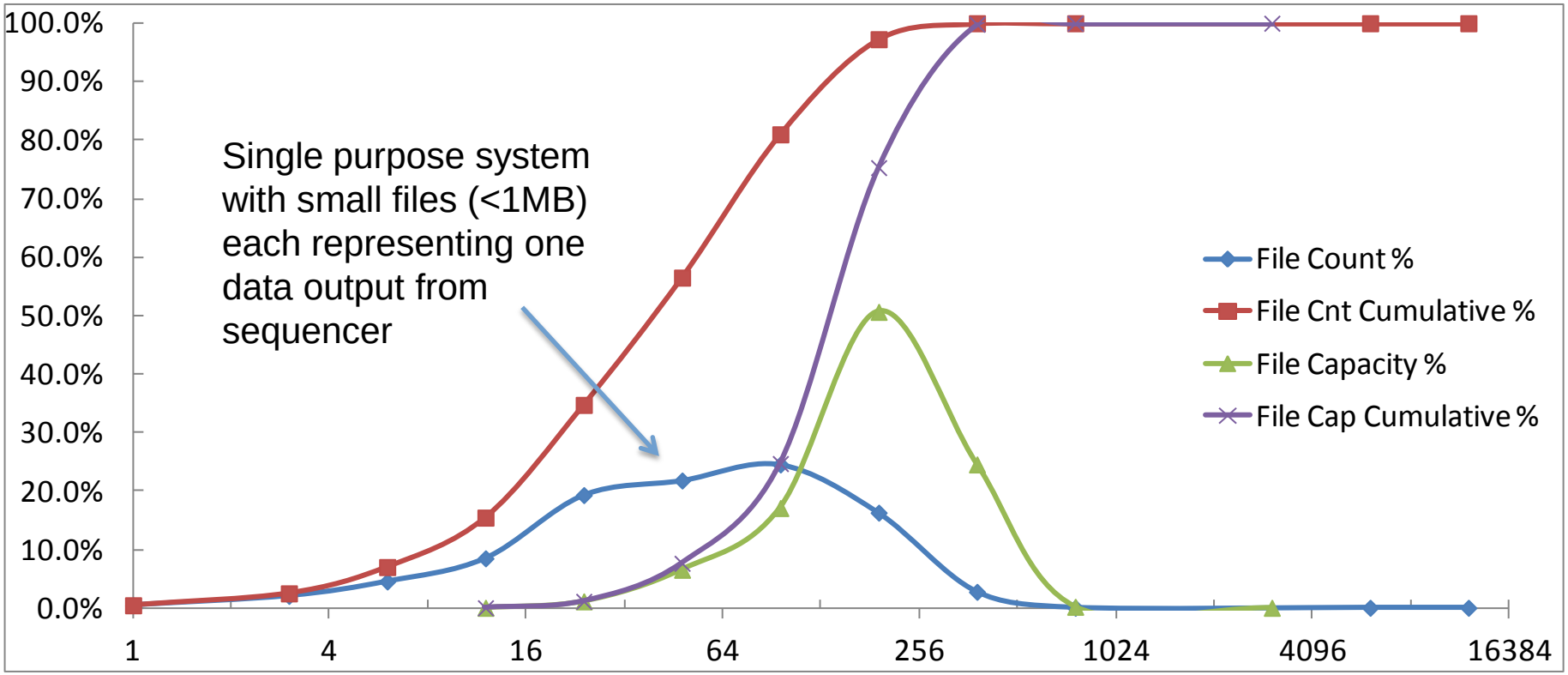
LANL DISTRIBUTIONS





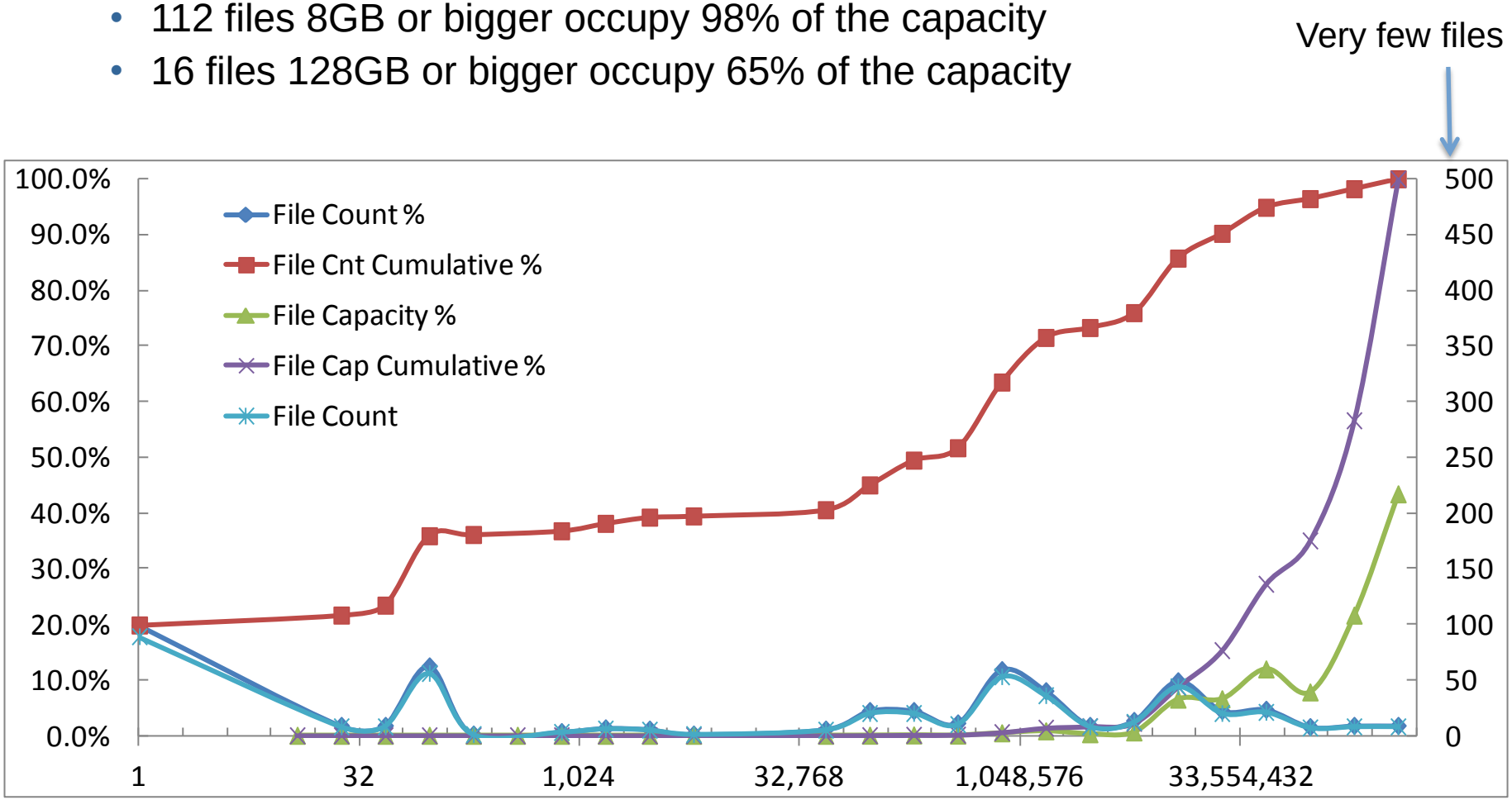
BIO-4

- **9.5 million files, 73KB average size**
  - Genomic sequence data
- **681 GB User Data, 1.2 TB Capacity used (43% overhead)**



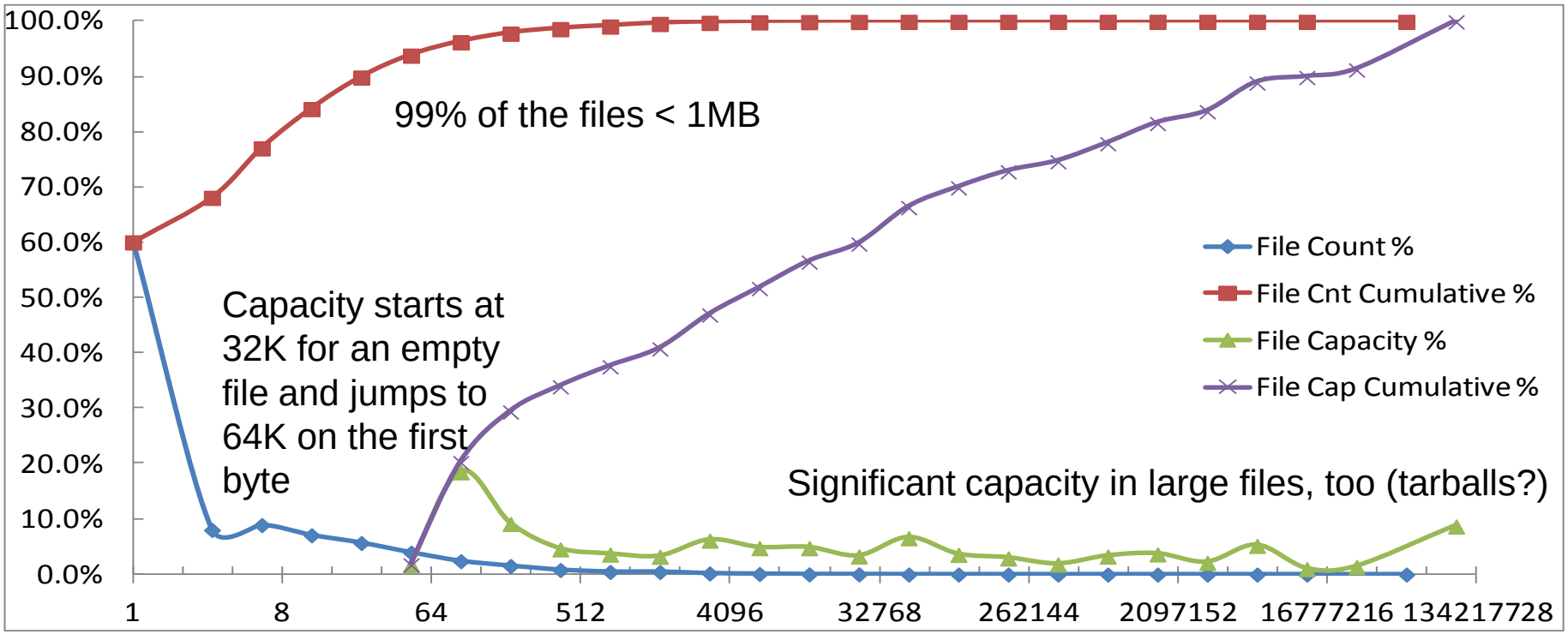
BIO-2

- **6.8 TB, 449 files**
  - Backup target for tarballs?
  - 112 files 8GB or bigger occupy 98% of the capacity
  - 16 files 128GB or bigger occupy 65% of the capacity

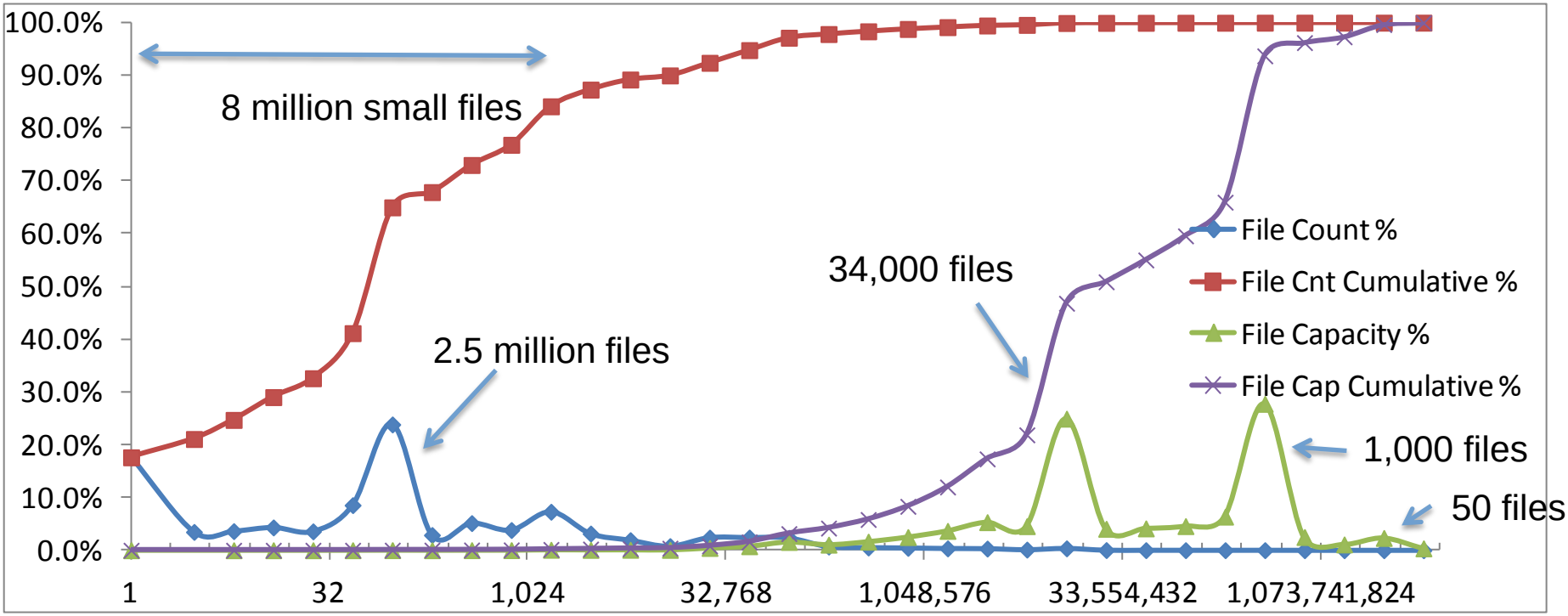


USER-3 (JOB LOG FILES)

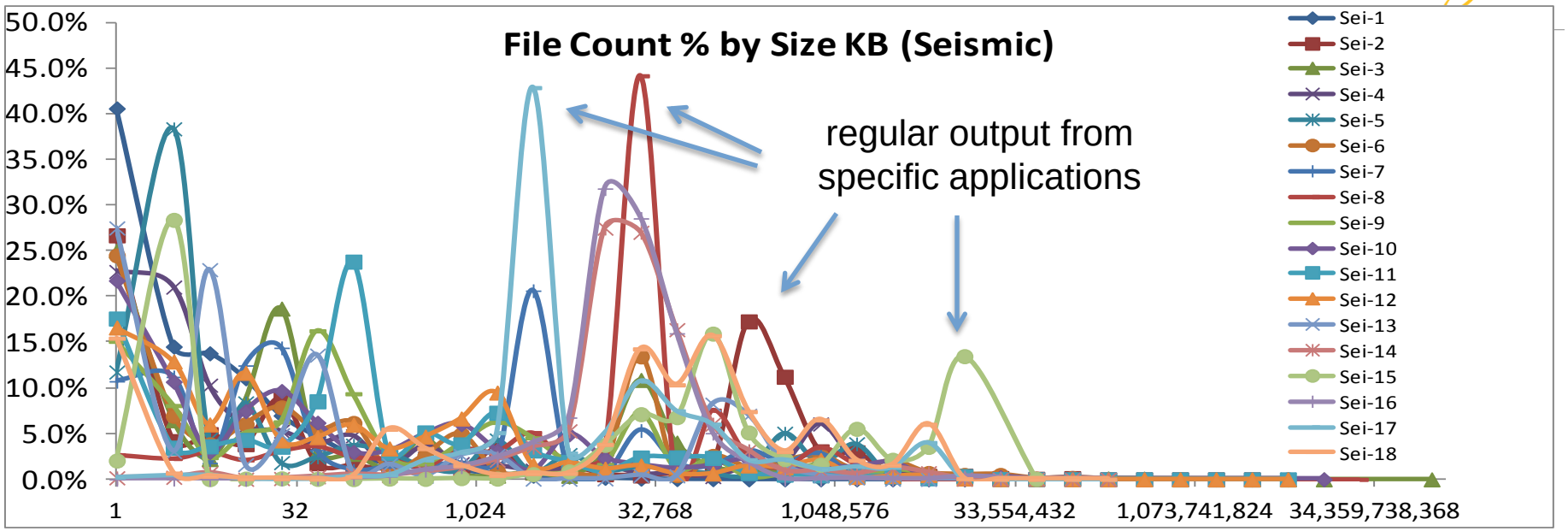
- **9.7 million files, 86KB avg size**
  - Per-job log files from large seismic processing cluster
- **2.4 million directories**
  - Average 6 names each, maximum 50,000 names
- **820 GB User Data, 2.36 TB Capacity Used (70% overhead)**

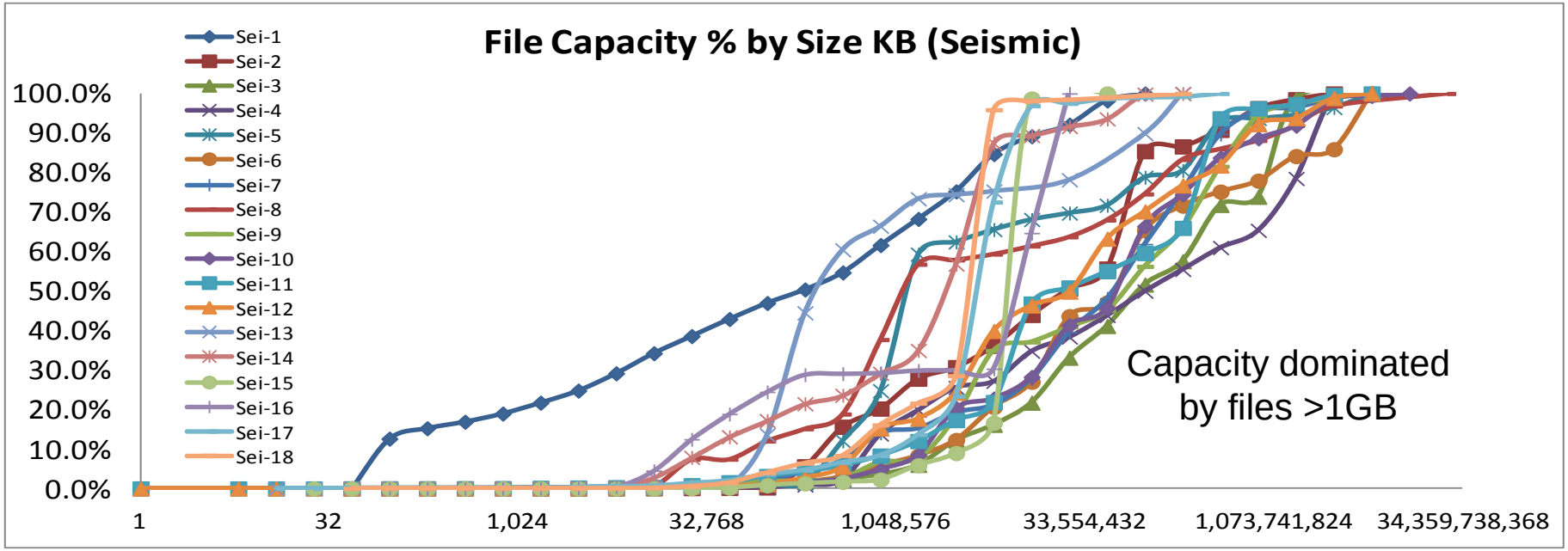


- **10.4 million files, 1.5 PB User Data**
  - 1000+ files bigger than 256 GB
  - 31 files bigger than 1TB. Max file 4.8 TB. Evidence of sparse files.
  - 65% files less than 128KB. 84% less than 2MB
  - 88% of the capacity in files 1GB or larger.



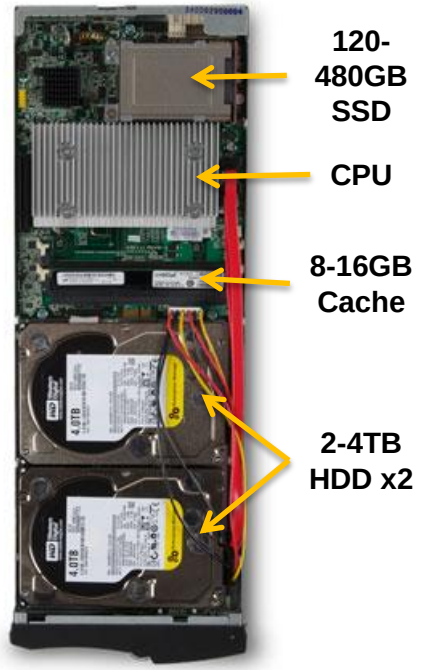
SEISMIC DISTRIBUTIONS





WHAT WE DID IN OSDFS

- **Set a soft boundary between a metadata zone and the data zone within the OSDFS partition**
- **All B-tree, object descriptors, and indirect blocks are allocated out of the metadata zone**
  - Did this before we had SSD to reduce disk fragmentation
- **All data extents > 2 blocks are allocated from the data zone**
- **With a hybrid OSD**
  - Concatenate the SSD and HDD(s) into one logical device
  - Simply line up the metadata zone with the SSD
  - Use existing OSDFS data structures, except for data packing
- **16K block size to shrink B+Trees and allocation map relative to our 4K block format**
  - Put 12K of object data into the object descriptor block to make better use of the SSD



ActiveStor 14 Storage Blade

SUMMARY

- **HPC file size distributes follow the old rule of thumb**
  - There are lots (and lots) of small files
  - But all the capacity is occupied by large (very large) files
- **There is wide variation among systems**
  - Different applications and user communities create different file distributions
  - Everyone can generate lots of small files – one file per core
- **Concentrating file system data structures and small objects on SSDs is a cost efficient way to boost performance**

- **Contact welch@panasas.com, or try Google, to find the data**