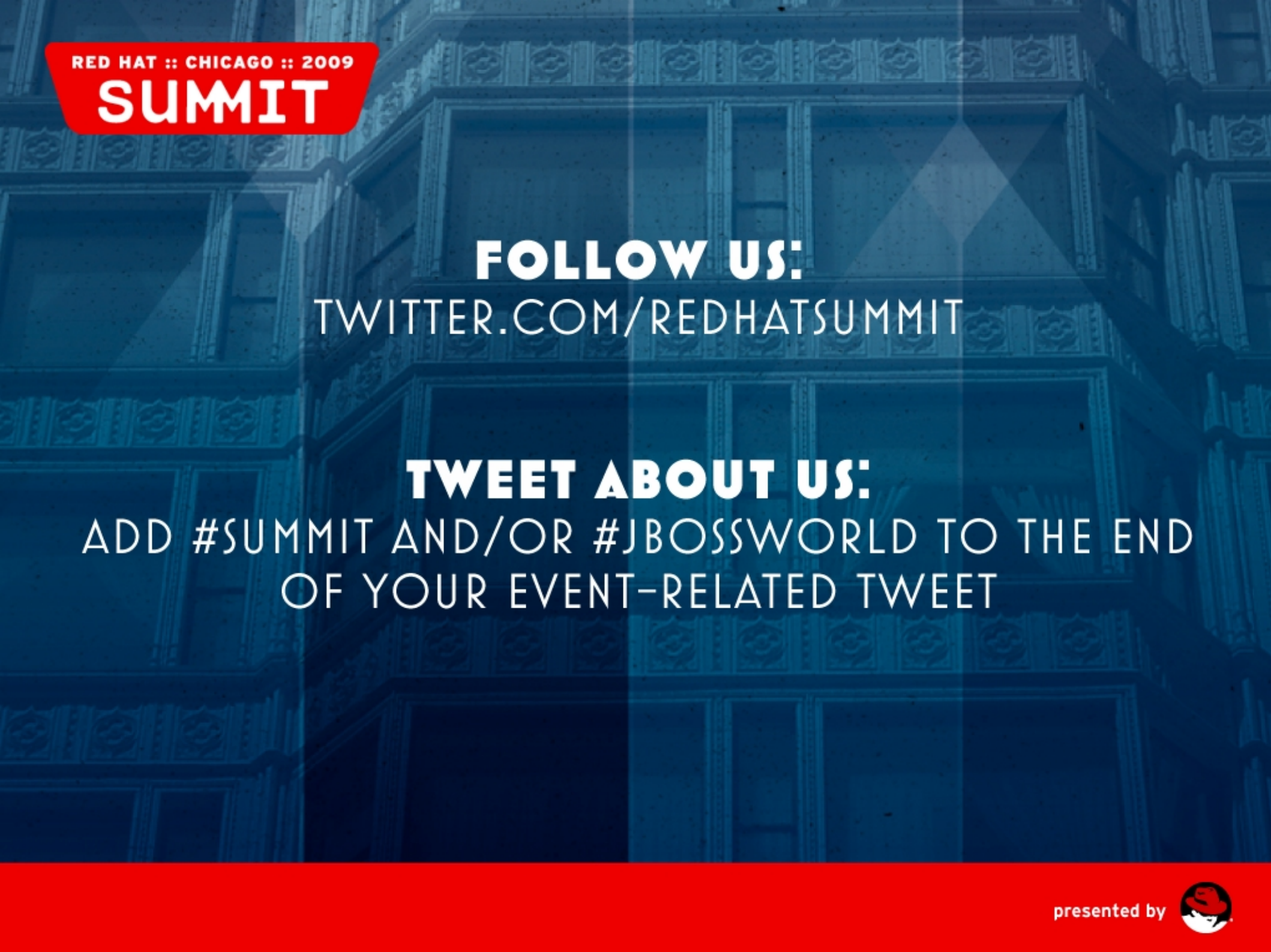

RED HAT :: CHICAGO :: 2009
SUMMIT
FOLLOW US:
TWITTER.COM/REDHATSUMMIT
TWEET ABOUT US:
ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END
OF YOUR EVENT-RELATED TWEET
presented by

Red Hat Enterprise Linux File Systems: Today and Tomorrow

David Egts, RHCA, RHCSS
Principal Solutions Architect
Red Hat

September 3, 2009

presented by

Overview

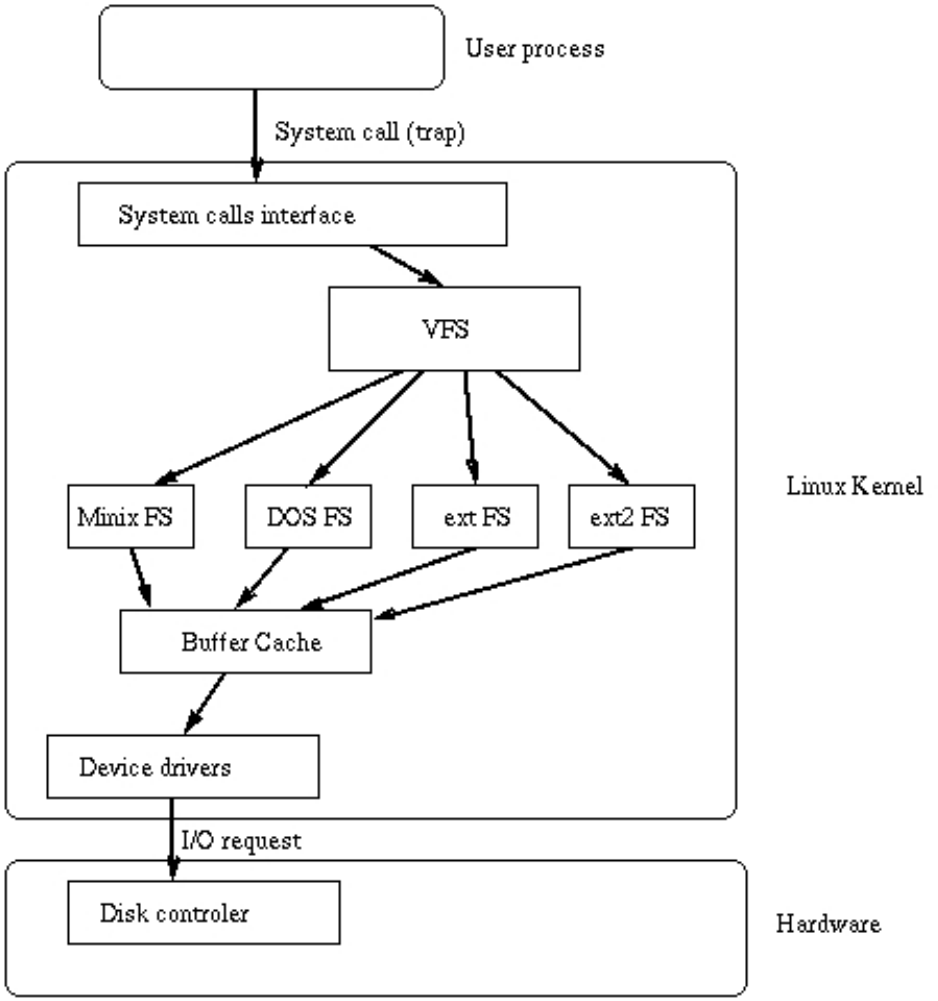
- Brief history of a few Linux file systems
  - Minix file system, ext, ext2, ext3
- Limitations of the current file systems
- Addressing the limitations with new file systems
  - XFS, ext4, Btrfs

Early Linux file systems: The Minix file system [1]

- Introduced in Minix in 1987 by Andrew S. Tanenbaum
- First Linux supported file system in 1991
- Linux was cross developed on Minix
- Performance and size issues
  - Max file system and file size: 64 MB!
- Met goal as a teaching aid
  - Similar to the Unix File System (UFS)
  - Simple and straightforward design for academic study
  - ... but not ideal for general purpose use

Early Linux file systems: The Linux Virtual File System (VFS)

- Initially written by Chris Provenzano, later rewritten by Linux Torvalds
- First used by ext



Source: [1]

Early Linux file systems: ext (the Extended File System) [1]

- Introduced in 1992 by Rémy Card, added to Linux 0.96c
- Supported file systems up to 2 GB and 255 character file names
- No support for separate timestamps for
  - Access
  - Inode modification
  - Data modification
- Linked lists to keep track of free blocks and inodes
  - Unsorted lists and file system fragmentation
  - Bad performance with extended use

ext2 [1]

- Introduced 1993 by Rémy Card, Theodore Ts'o, and Stephen Tweedie
- Major rewrite of and follow on to ext
- Allows for extension of file system functionality while maintaining internal structures
  - Eases development for ext3, ext4, ...
- Adopted advanced ideas from other file systems
  - Berkeley Fast File System (FFS)

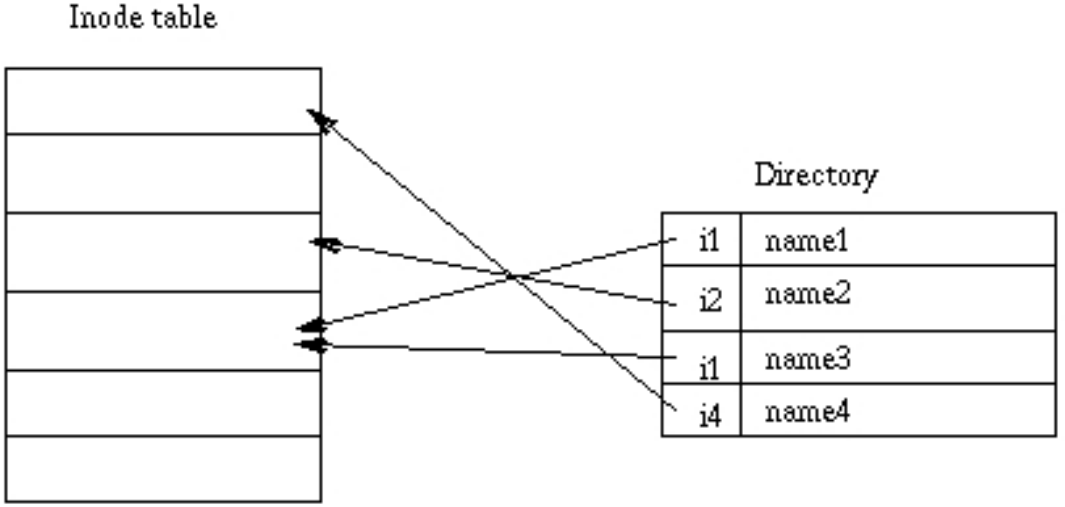
ext2 major features [1]

- Fast symbolic links – target stored in inode
- Choice of block sizes (1024, 2048, 4096 bytes)
- Extended attributes and POSIX ACLs
- File system state tracking
  - **Not Clean** = Read / write mount
  - **Clean** = Unmount or read only mount
  - **Erroneous** = Kernel detects inconsistency to force fsck
  - Can fsck based upon mount count and check interval
- Good for limited write cycle, flash-based storage media
  - Lack of journal minimizes writes

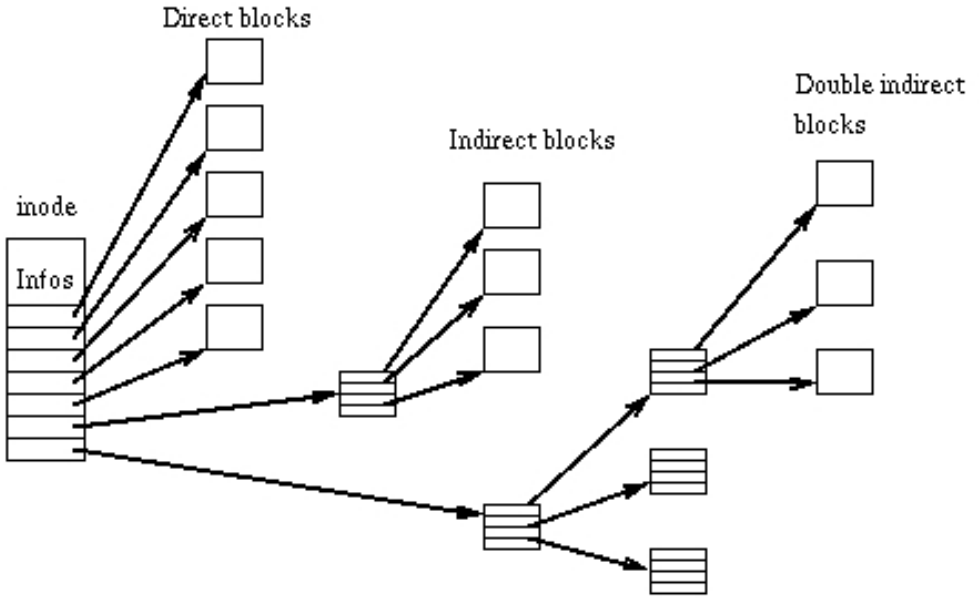
ext2 definitions [2]

- **Blocks** as basic unit of storage
- **Inodes** keep track of files and system objects
- **Block groups** split the disk into more manageable sections
- **Directories** provide hierarchical organization of files
- **Block bitmaps** and **inode bitmaps** keep track of allocated blocks and inodes
- **Superblocks** define the parameters of the file system and its overall state

Inodes and directories



Source: [1]



Source: [1]

ext3

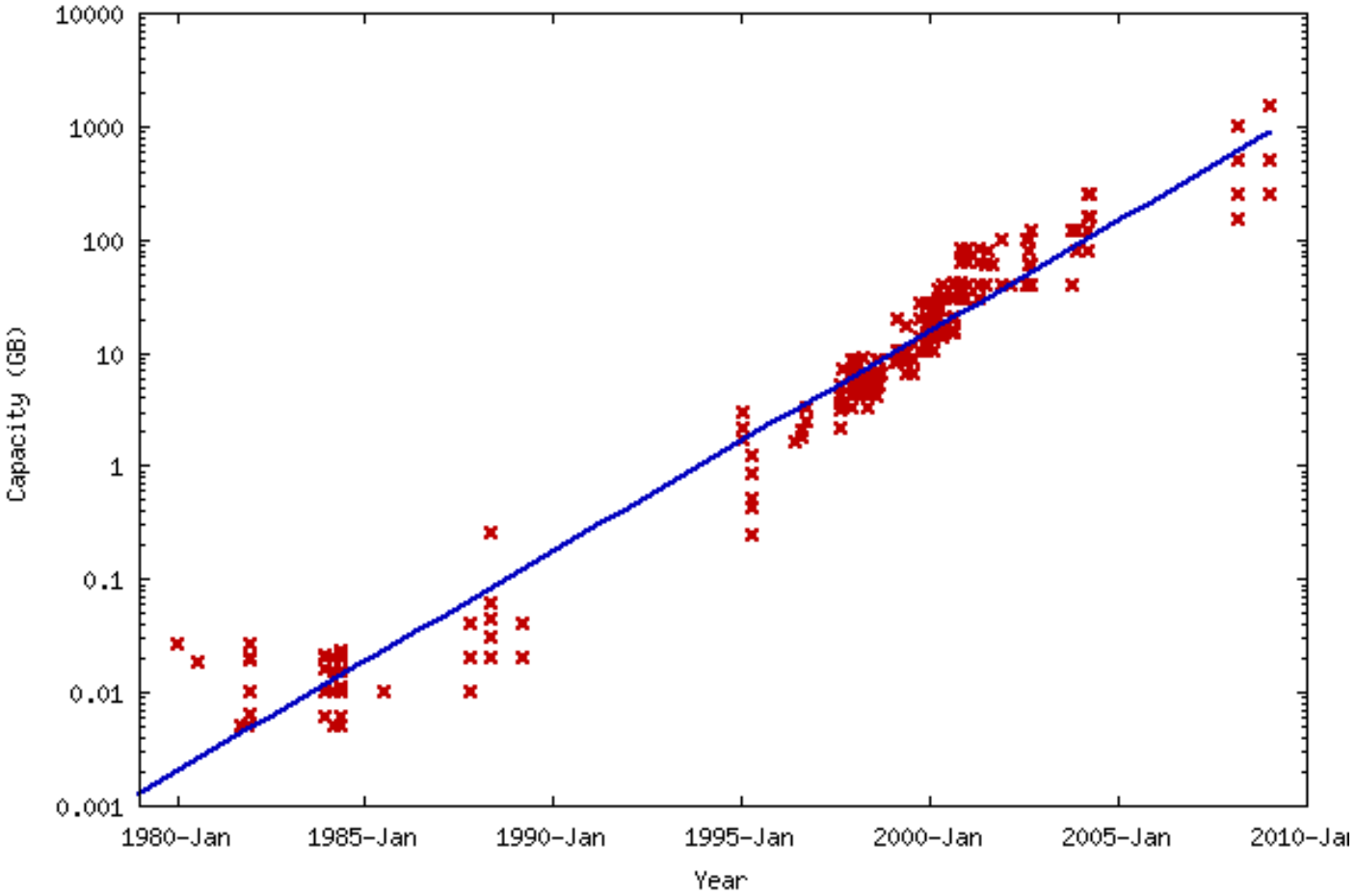
- Introduced in 2001 by Stephen Tweedie in the 2.4.15 Linux kernel
- Default file system for Red Hat Enterprise Linux and many other Linux distributions
- Red Hat Enterprise Linux 5 supports up to 2 TB files and 16 TB file systems

ext3 major features

- ext2 + journaling
  - Speeds up system recovery by shortening fsck times [3]
  - Journaling options [4]
    - **Ordered** (default) – only the metadata
    - **Journal** – metadata and data
    - **Writeback** – only the metadata but no commit order guarantee
    - External journal
  - Straightforward conversion between ext2 and ext3
  - Shares time tested and mature e2fsprogs with ext2
- Online file system growth
- HTree indexing – 50-100x faster w/larger directories [5]

The challenges

1. Disk drive density increases over time



"Unfortunately, every doubling of disk capacity leads to a doubling of recovery time when using traditional filesystem checking techniques." [3]

Source: [6]

The challenges (cont)

- “Kryder's Law” [7]
  - Magnetic disk areal storage density doubles annually
  - Some consider Kryder's law is flawed [8, 9, 10]
    - “Technology is approaching the superparamagnetic limit”
  - Still, general purpose file systems invented years ago were designed to work great on the disk drive capacities of their era

Year	File System	Capacities	Best Value
1992	ext	40 MB – 510 MB	$3.50 / MB
1993	ext2	40 MB – 510 MB	$1.68 / MB
2001	ext3	10 GB – 75 GB	$2.65 / GB
2006	ext4	80 GB – 500 GB	$0.64 / GB

The challenges (cont)

2. Disk speeds don't keep up with density growth
   - Interface and bus bandwidths
   - Rotational speeds
   - The gap widens
     - “Although disk drives are becoming faster each year, this speed increase is modest compared with their enormous increase in capacity.” [3]
   - An fsck in 2009 takes “longer” than an fsck in 1993

Year	File System	Capacities	Best Value	Bus Technology	Bus Bandwidth
1992	ext	40 MB – 510 MB	$3.50 / MB	ATA-1	8.3 MB / sec
1993	ext2	40 MB – 510 MB	$1.68 / MB	ATA-1	8.3 MB / sec
2001	ext3	10 GB – 75 GB	$2.65 / GB	ATA-5	66 MB / sec
2006	ext4	80 GB – 500 GB	$0.64 / GB	SATA	150 MB / sec

The challenges (cont)

- Fibre Channel or Infiniband to the rescue?
  - Adds speed
    - Faster interfaces
    - More interfaces per system
  - Allows for even more TB behind each interface
  - Problem is worsened

XFS [11]

- Development started by SGI in 1993
- “xFS” – the extension for EFS
  - “x” for to-be-determined (but the name stuck)
- First released with IRIX 5.3 (1994)
  - One of the oldest journaling file systems for UNIX
- Released under the GPL in 2000
- 2.6 Linux kernel released with full XFS support in 2003

XFS: Extreme scalability [11, 12]

- Journaled file system with sub-second recovery
- 64-bit file system
  - Max file size: 9M TB
  - Max file system size: 9M TB
  - Millions of files per directory
- Variable block sizes
  - 512 bytes – system page size
- Can grow live file system on the fly
- Online defragmentation (if needed)
- Fast metadata check and repair times

XFS: Extreme performance [11, 12]

- B+ trees for fast file searches and space allocation
- Allocation groups for improved parallelism
- Parallel direct I/O
- IOs and extent allocs along stripe unit/width boundaries
- Extent based allocation
- Pioneered delayed allocation for buffered writes
  - High resistance to space fragmentation
- Persistent file pre-allocation
- Close to raw I/O performance
  - Multiple GB/sec on multi-TB systems

Noteworthy XFS tools [12]

- **xfsdump / xfsrestore**
  - Like dd or tar on steroids for XFS
  - Maintains 64-bit inode numbers, file lengths, holes, etc.
  - Can interrupt and resume dump or restore w/o pain
  - Can dump and restore across several drives simultaneously in parallel
- **fsck.xfs** – "Do nothing, successfully"
- **xfs_check** – checks file system consistency
  - Deprecated in favor of xfs_repair (lower memory requirements)
- **xfs_repair** – checks metadata and repairs file system
- **xfs_fsr** – defragments file system online
- **xfs_growfs** – grows file system

XFS considerations

- Can't shrink file system
- Journal only includes metadata, not the data itself
  - Done for speed
  - Only metadata is guaranteed consistent post crash
  - Buffered I/O which has not been sync'ed could be lost
  - ext3 and JFS do this too by default

XFS considerations (cont)

- Sometimes slower than other file systems
  - Creation of directory entries
    - Empty files, subdirectories, etc.
  - Deletion of directory entries
  - Generally faster than most other file systems in other categories
- 16 TB file and file system maximums for 32-bit Linux
  - Limitation of 32-bit Linux not XFS
  - 64-bit Linux just fine

XFS and Red Hat Enterprise Linux

- Limited availability in Red Hat Enterprise Linux 5.4
- x86_64 only
- Layered offering

ext4 [13]

- Developers include Mingming Cao, Andreas Dilger, Alex Tomas, Dave Kleikamp, Theodore Ts'o, Eric Sandeen, Sam Naghshineh, and others
- Started as a series of backward compatible extensions
  - Address the scalability, performance, and reliability issues of ext3
  - Fork of ext3 in 2006 to not affect current ext3 users
  - Added as stable in the 2.6.28 Linux kernel in 2008
- Default file system for Fedora 11 and other distributions
  - Fedora 11 – first distribution shipping a testing version
  - Red Hat Enterprise Linux 5.1 – first EL w/tech preview

ext4 features [13]

- Compatibility with ext2 and ext3
  - Can mount existing ext2 and ext3 file systems as ext4
  - Can mount ext4 as ext3
    - ... assuming extents were never used
- Delayed allocation
  - Don't allocate blocks until data is going to be written to disk
  - Improves performance
  - Reduces fragmentation – allocations based upon actual file size

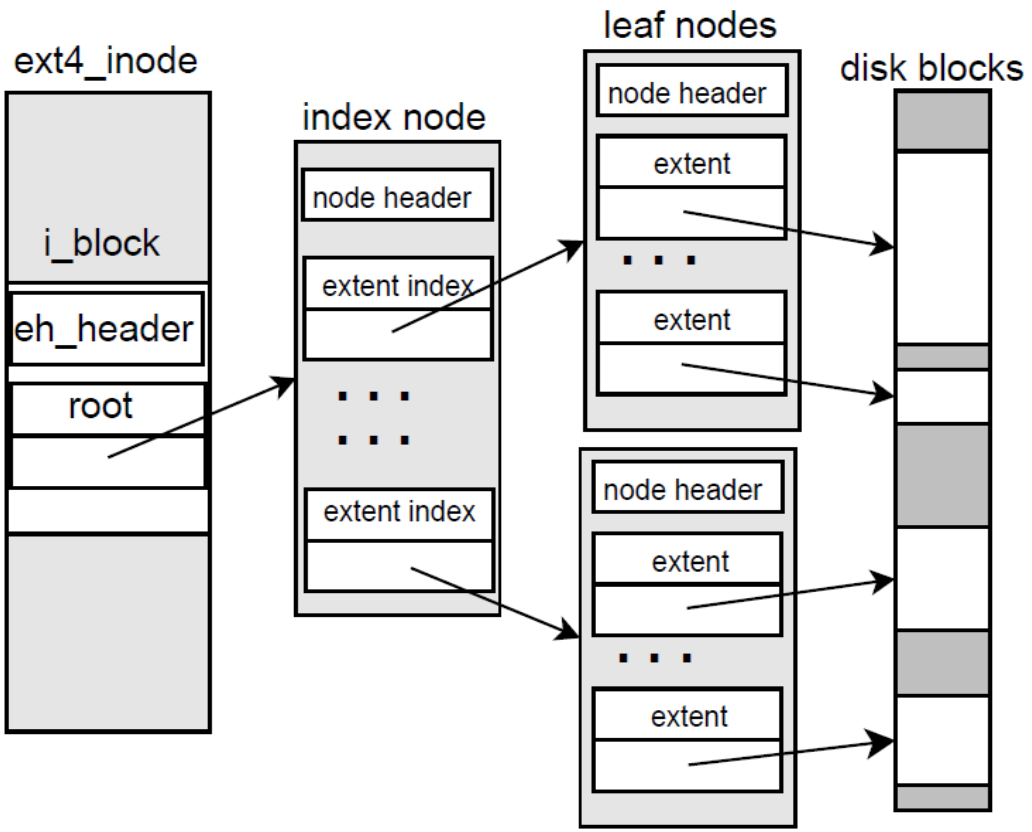
ext4 features (cont) [13]

- Multiblock allocator
  - Multiple blocks for a file allocated in a single operation
  - Reduces fragmentation by choosing contiguous blocks
- Delayed and multiblock allocation observed benefits
  - Improved throughput by 30%
  - Reduced CPU usage by 50%

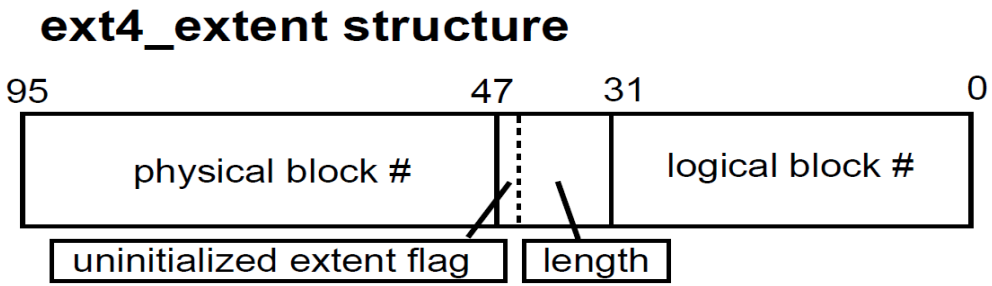
ext4 features (cont) [13]

- Extents
  - Replaces block mapping as used in ext2 and ext3
  - Range of contiguous blocks
    - Start/length pairs
    - 2^{15} blocks $\rightarrow$ 128 MB with 4 KB block size
  - Improves large file performance and reduces fragmentation
  - Up to 4 extents can be stored in the inode, additional indexed in an Htree
  - Extents w/48-bit block numbers (vs. 32-bit in ext3)...
    - 16 TB (ext3) $\rightarrow$ 1 EB (ext4 with 4 KB block size) max file size

ext4 data structures



Source: [13]



Source: [13]

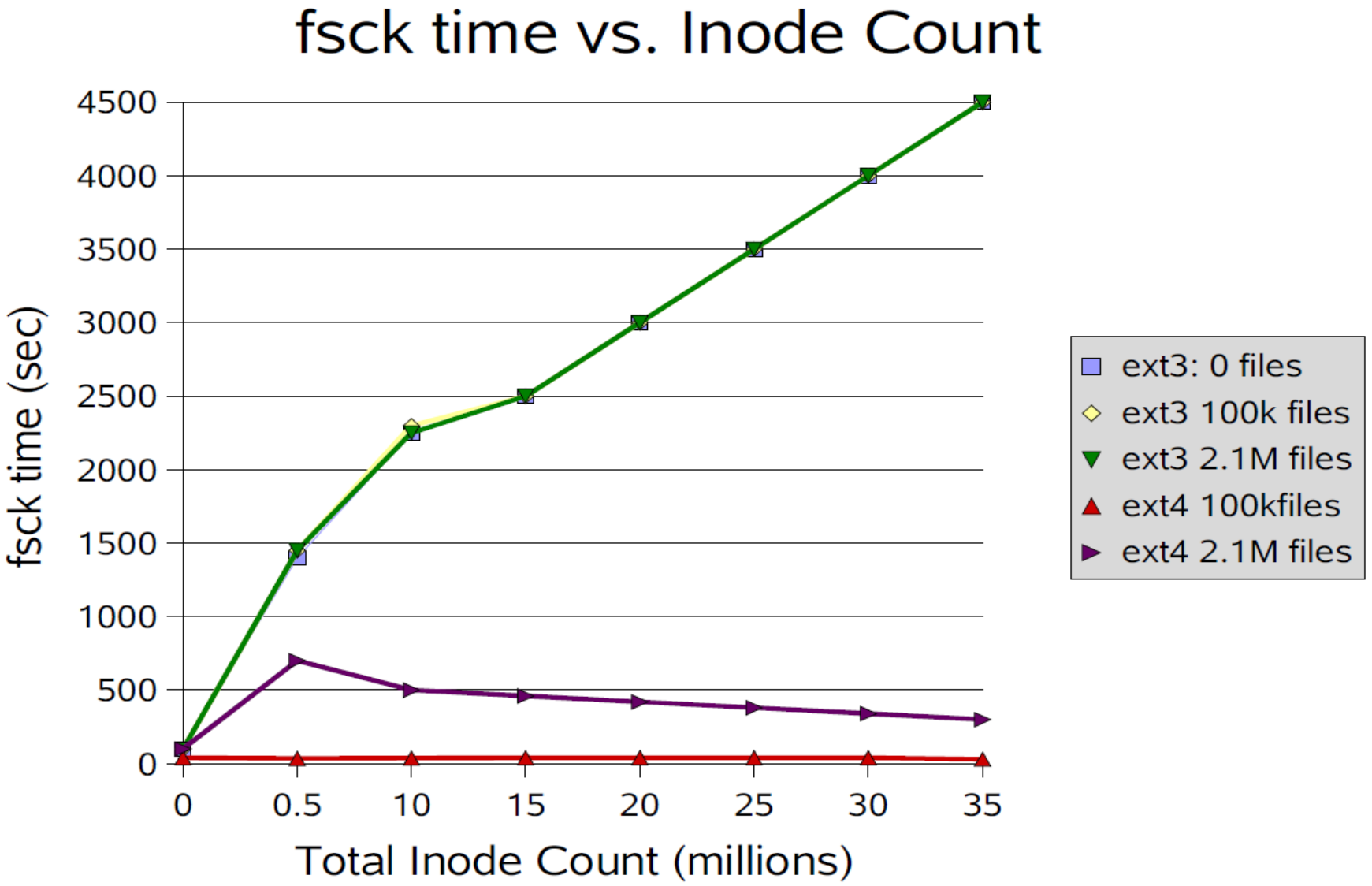
ext4 features (cont) [13]

- Persistent file pre-allocation
  - Old way – write a file full of 0s
  - New way – reserve but don't take time to write out 0s
    - Guaranteed available (unlike a sparse file)
    - Likely contiguous
    - Ideal for media streaming and databases
- 32,000 → 64,000+ subdirectories in a directory
- Improved timestamps
  - Second → nanosecond granularity
  - Year 2038 problem deferred to 2514

ext4 features (cont) [13]

- Online defrag of individual files or entire file system
  - e4defrag
  - Currently in git and Fedora Rawhide
- Journal and block group checksums
- Faster fsck
  - Problem: full fsck of 2 TB ext3 on high end RAID → 2 to 4 hours to days
  - Unallocated block groups and inodes are marked and don't need to be fsck'ed
  - Consequence: 2x to 10x+ speed up
  - Enabled by default or via -O uninit_groups

fsck performance comparison

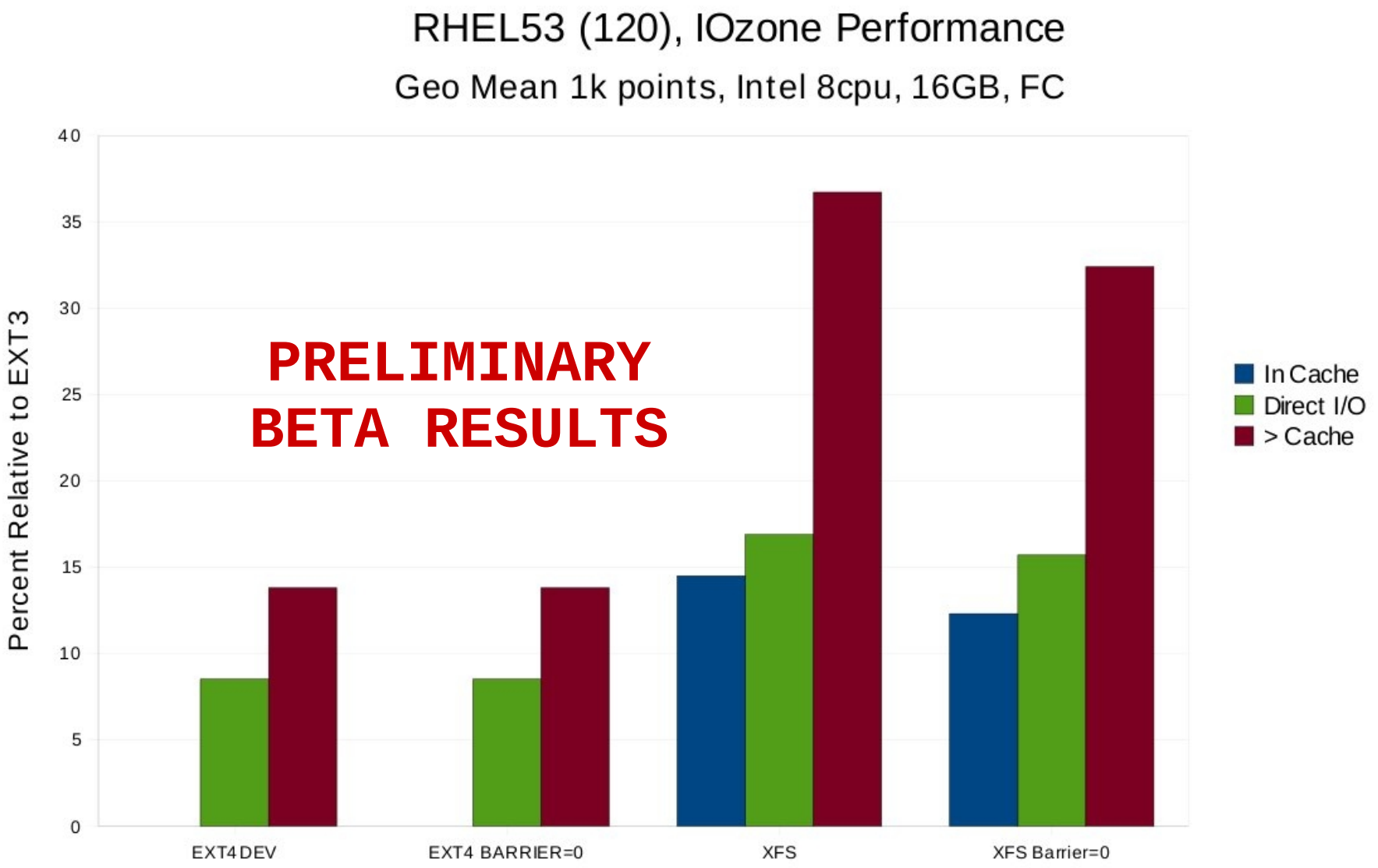


Source: [13]

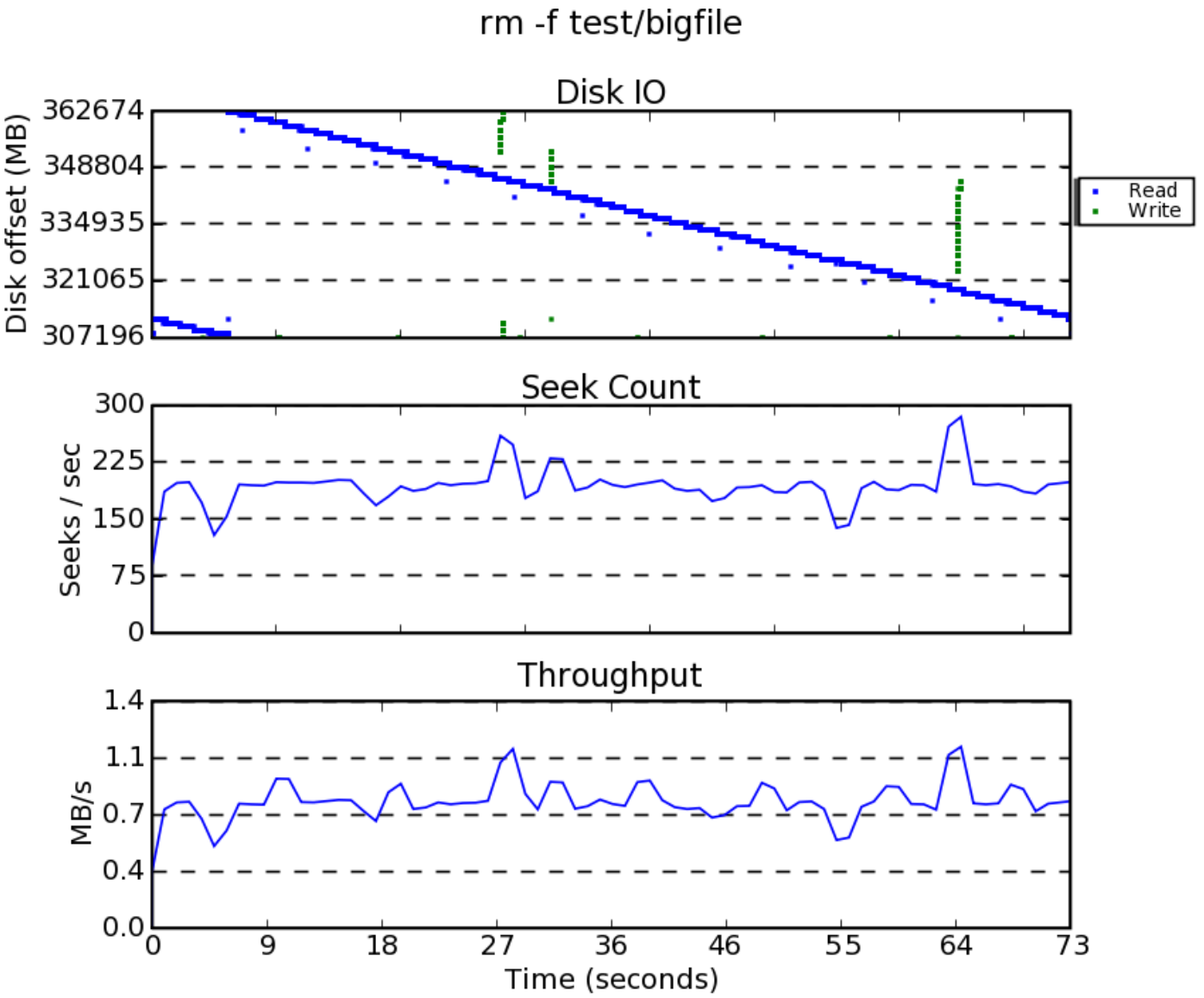
ext4 and Red Hat Enterprise Linux

- Tech preview in Red Hat Enterprise Linux 5.3 and 5.4
- Included as core part of OS (not a layered product)

RHEL5.3 IOzone EXT3, EXT4, XFS eval

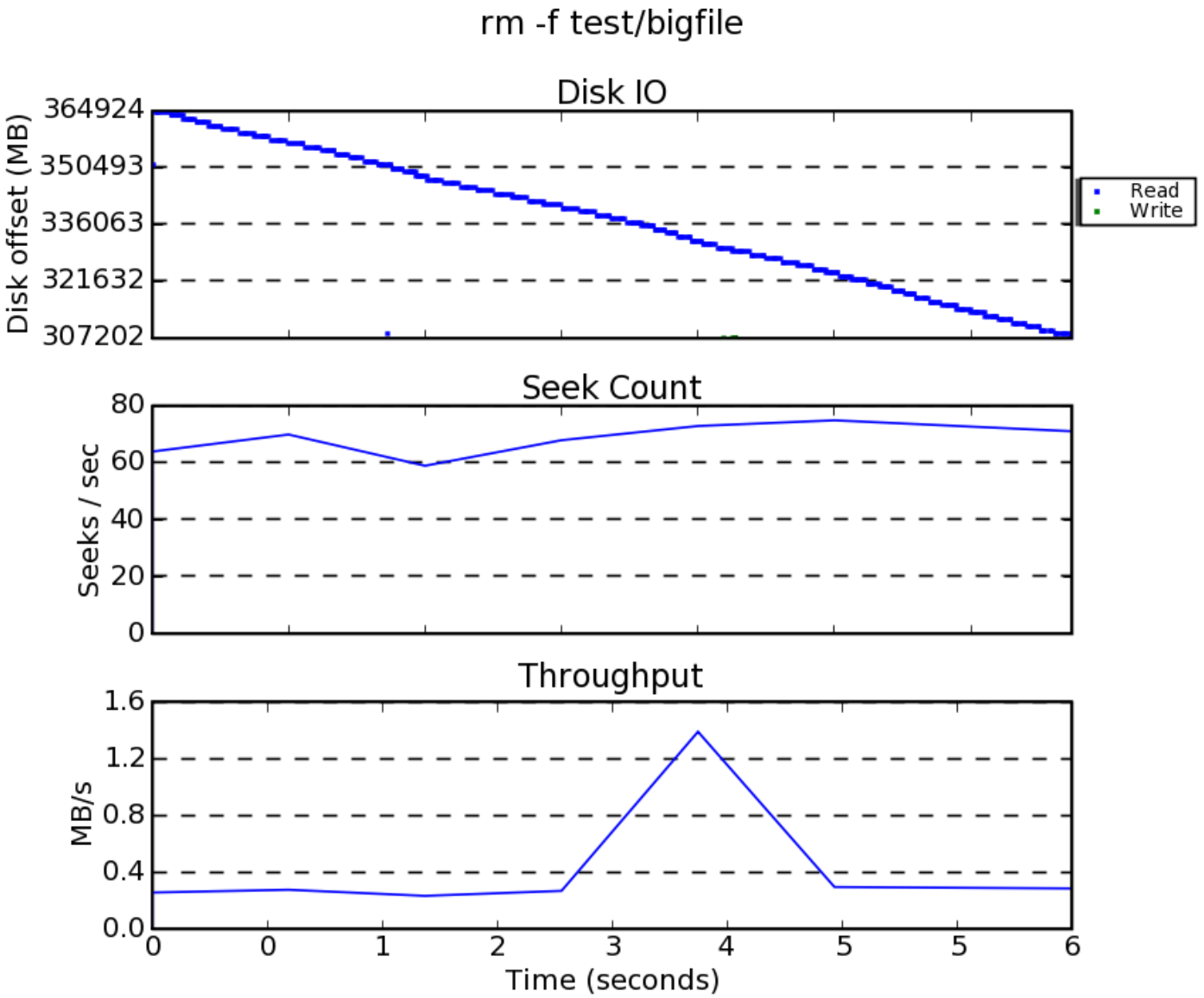


Best case deletion of 56 GB file: ext3



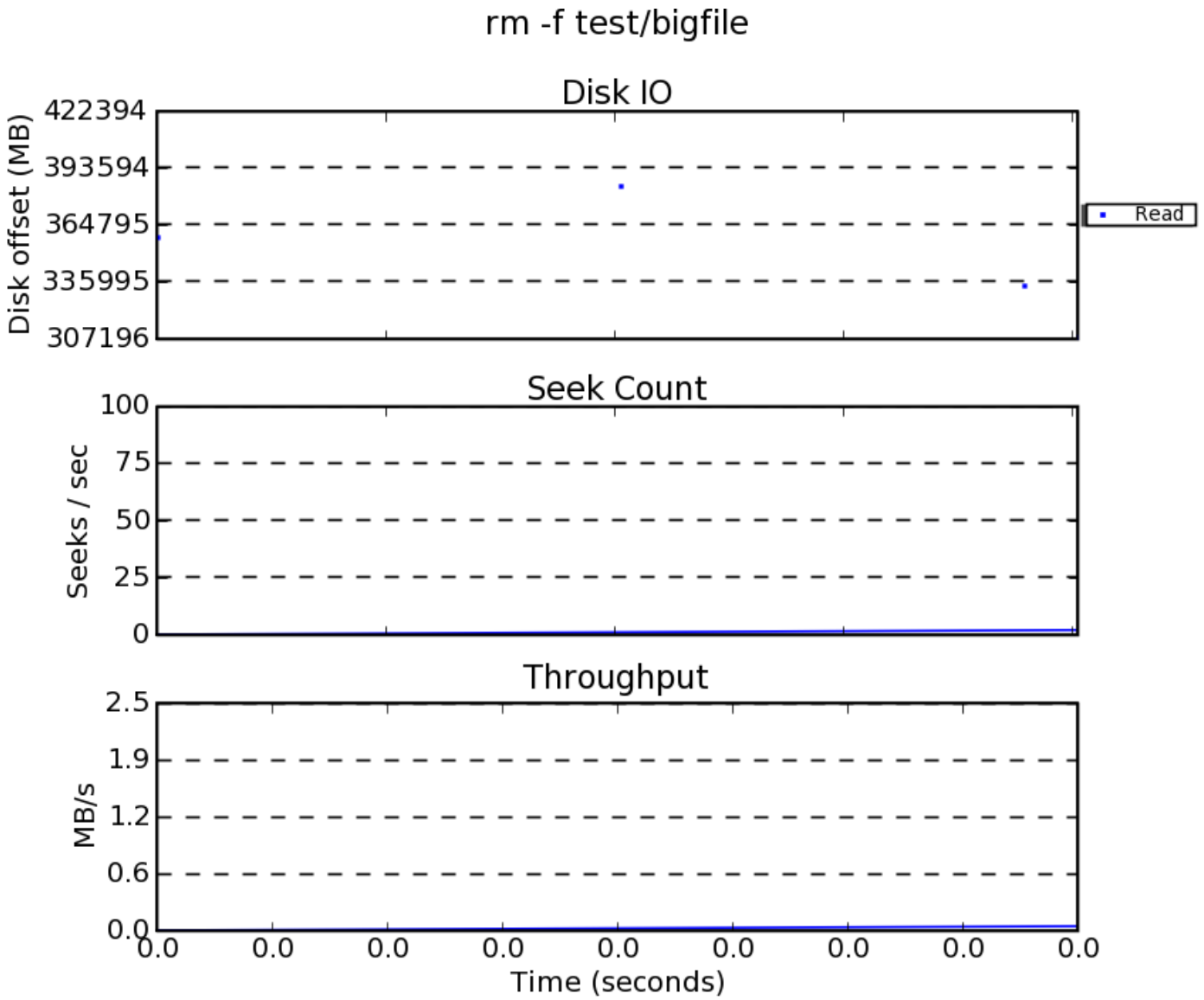
- Must delete from EOF to satisfy journal constraints
- Must read entire direct/ indirect tree
- 56 MB reads
- 14,000 IOs
- 73 seconds

Best case deletion of 56 GB file: ext4



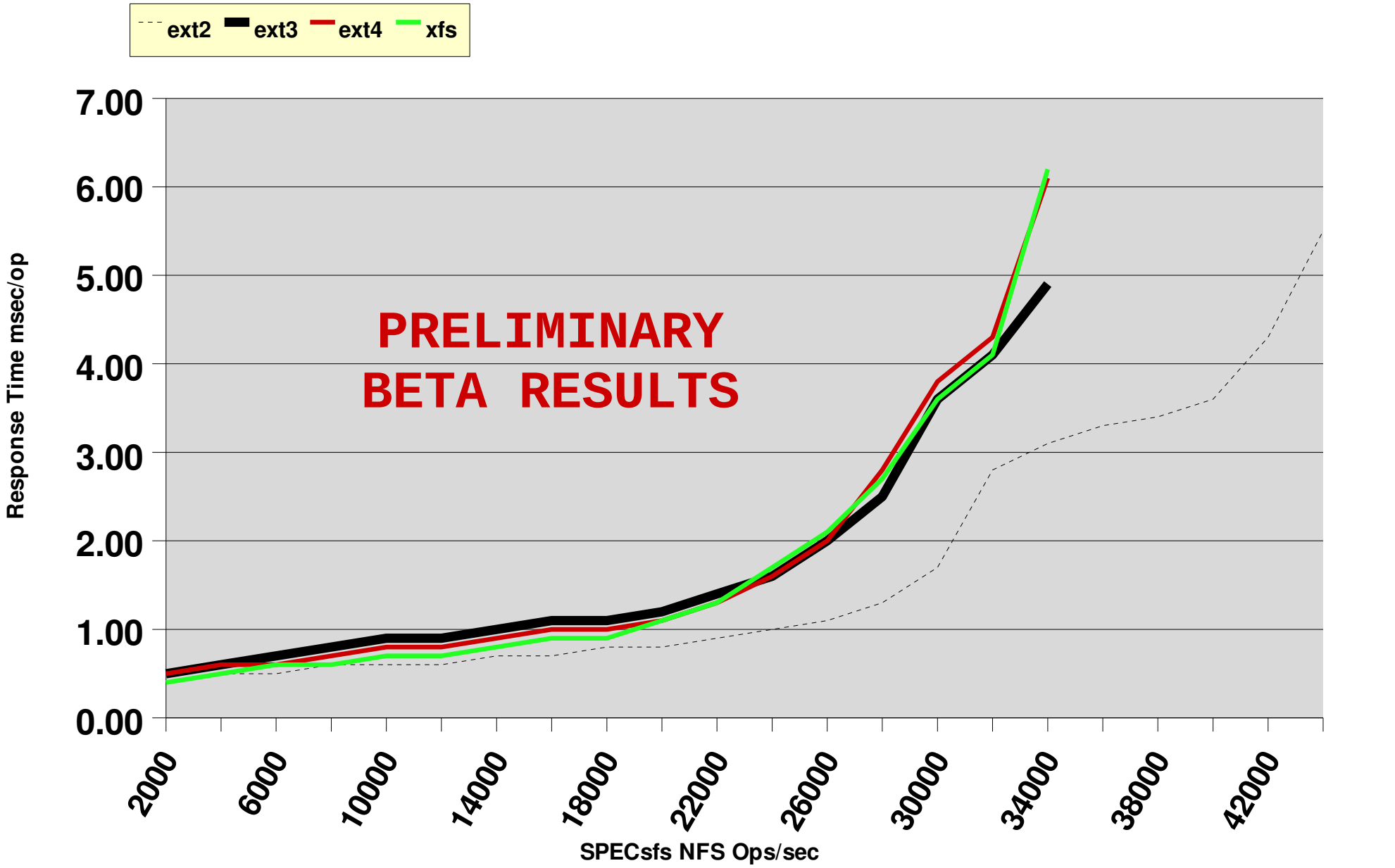
- 448 extents
- 1,820 KB reads
- 455 IOs
- 6 seconds

Best case deletion of 56 GB file: XFS

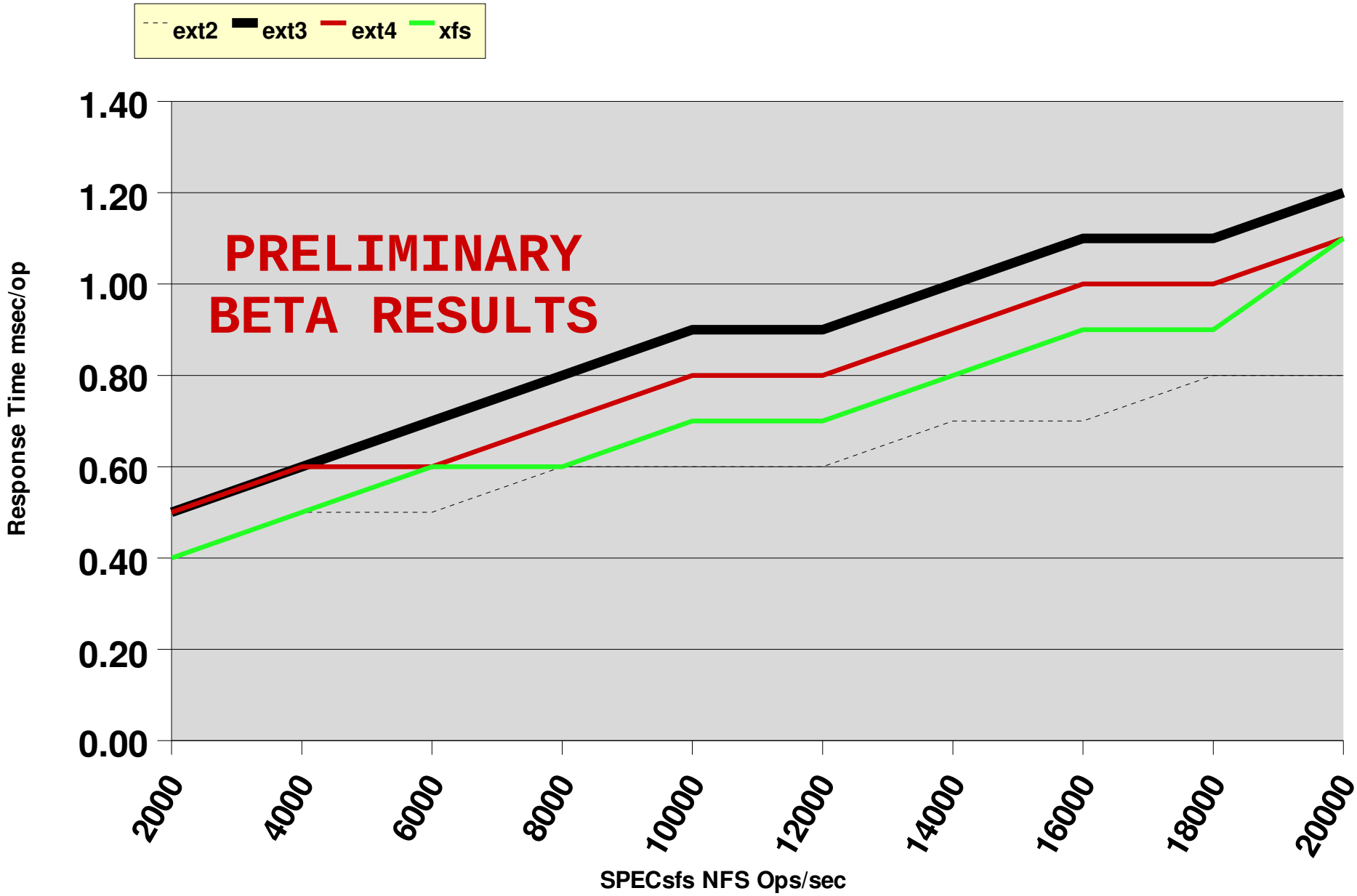


- All data stored in a single extent
- 36 KB reads
- 9 IOs
- “Near instantaneous”

SPECsfs NFS benchmark with RHEL5.4 -156 kernel on BIGI testbed
4 clients/52 processes per client/52 filesystems
DL580 16GB/4FC/4HP-MSA1000/4Gbit NICS with jumboframes

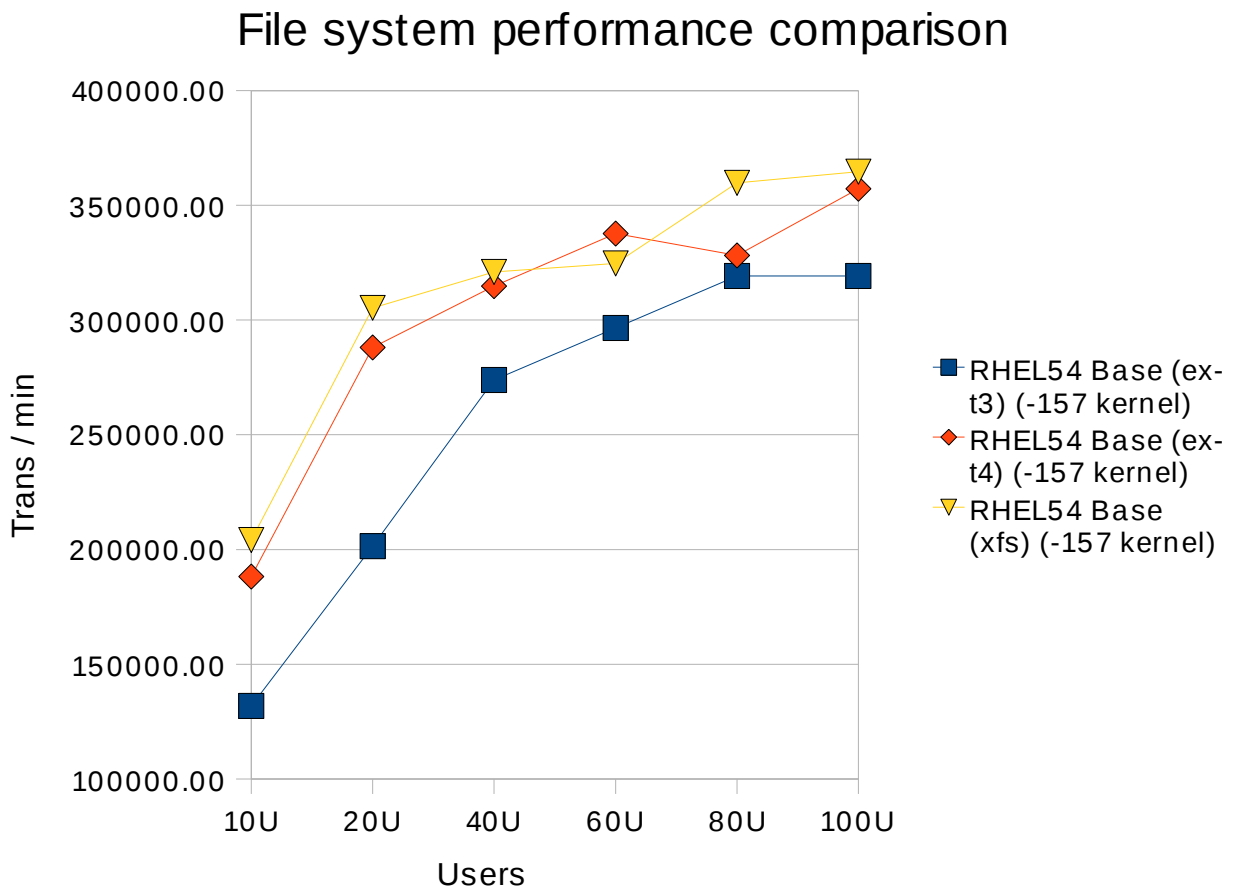


SPECsfs NFS benchmark with RHEL5.4 -156 kernel on BIGI testbed
4 clients/52 processes per client/52 filesystems
DL580 16GB/4FC/4HP-MSA1000/4Gbit NICS with jumboframes



Maximizing AMD Six-core Opteron™ Processor Performance with Red Hat Enterprise Linux

- Friday 9:45-10:45 AM
- Bhavna Sarathy (AMD) and Sanjay Rao (Red Hat)



ZFS? [14]

- Introduced in OpenSolaris in 2005 by Sun Microsystems
- File system + volume manager + RAID in one
  - RAID 0, RAID 1, RAID-Z (3+ devs), RAID-Z2 (4+ devs)
  - Can transparently utilize SSDs as cache
  - Automatic striping for balanced write load
  - Variable block sizes up to 128K
  - Built in compression – save up to 2-3x space and I/O time at cost of CPU time
  - Copy-on-write transactional model
    - Facilitates (read only) snapshots and (read / write) clones
    - No need to fsck

ZFS on Linux?

- Linux Kernel license (GPL) isn't compatible with the ZFS license (CDDL)
- Works with FUSE
  - File system runs in user space
  - Performance penalty?
  - http://www.wizy.org/wiki/ZFS_on_FUSE

Btrfs [15, 16]

- Announced by Chris Mason of Oracle in 2007
- Goals
  - Keep up with massive storage devices coming out in the next 10 years
  - Flexible storage management
  - Ensure normal administrative tasks can be done online
- Currently under heavy development
  - Merged into the 2.6.29 Linux kernel

Btrfs features [15, 16]

- Copy on write snapshots
- Enhanced built-in RAID
  - Fast recovery
    - Rebuild only use blocks in use by file system
  - Data chunk RAIDed – not whole disks
    - Decouples RAID level from number of disks
    - Easily easy add, remove, restripe over time
- Example – RAID 1 metadata and RAID 10 data
  - mkfs.btrfs -m raid1 -d raid10 /dev/sd{a,b,c,d}
- Good comparison of Btrfs vs. ZFS see [16]

Concluding thoughts

- Choose the file system that best fits your performance requirements
- ext3 is very mature, but is showing signs of age
- XFS is time tested and is an available consideration
- ext4 is positioning itself to replace ext3 as the next de facto Linux file system
- ZFS is compelling but is currently incompatible with the GPL, and would take time to be proven on Linux
- Btrfs is positioning itself to be the next generation, large scale file system for Linux

References

[1] Design and Implementation of the Second Extended Filesystem

http://e2fsprogs.sourceforge.net/ext2intro.html

[2] The Second Extended File System: Internal Layout

http://www.nongnu.org/ext2-doc/ext2.html

[3] Journaling the Linux ext2fs Filesystem

http://e2fsprogs.sourceforge.net/journal-design.pdf

[4] Ext3 Filesystem

http://www.kernel.org/doc/Documentation/filesystems/ext3.txt

[5] State of the Art: Where we are with the Ext3 filesystem

http://ext2.sourceforge.net/2005-ols/2005-ext3-paper.pdf

References (cont)

[6] Hard drive capacity over time

http://en.wikipedia.org/wiki/File:Hard_drive_capacity_over_time.svg

[7] Kryder's Law

http://www.scientificamerican.com/article.cfm?id=kryders-law

[8] Kryder's Law: A rule of thumb for hard drive growth

http://www.mattscomputertrends.com/Kryder%27s.html

[9] Hard Disk Trends

http://www.mattscomputertrends.com/harddrives.html

[10] Hard Disks - Historical Pricing

http://www.mattscomputertrends.com/harddiskdata.html

References (cont)

[11] XFS Overview and Internals

http://oss.sgi.com/projects/xfs/training/index.html

[12] XFS for Linux Administration

http://techpubs.sgi.com/library/manuals/4000/007-4273-003/pdf/007-4273-003.pdf

[13] The new ext4 filesystem: current status and future plans

https://ols2006.108.redhat.com/2007/Reprints/mathur-Reprint.pdf

[14] ZFS

http://en.wikipedia.org/wiki/Zfs

[15] The Btrfs file system

http://www.h-online.com/open/The-Btrfs-file-system--/features/113738

[16] Linux Don't Need No Stinkin' ZFS: BTRFS Intro & Benchmarks

http://www.linux-mag.com/id/7308

Special Thanks!

- Eric Sandeen, Red Hat
- Sanjay Rao, Red Hat
- Barry Marson, Red Hat
- Douglas “Shak” Shakshober, Red Hat