SUMMIT JBoss WORLD

PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

ONE BILLION FILES:

Pushing Scalability Limits of Linux

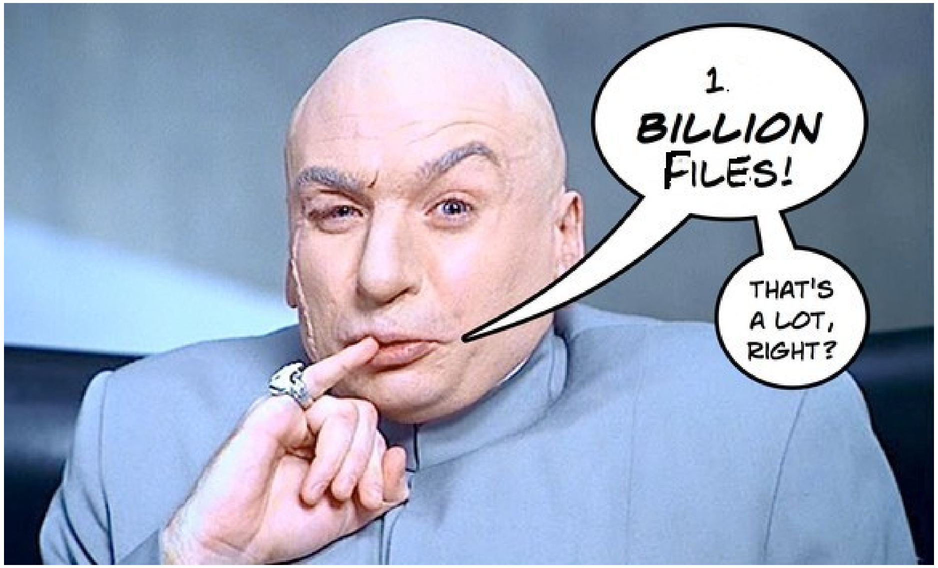
File Systems

Ric Wheeler
Architect and Manager, Red Hat
May 5, 2011

Overview

- ***Why Worry about 1 Billion Files?***
- Storage Building Blocks
- Things File Systems Do & Performance
- File System Design Challenges & Futures

Why Worry about 1 Billion?



- 1 million files is so 1990
- 1 billion file support is needed to fill up modern storage!

How Many Files are Needed to Fill a File System?

FS Size	10KB Files	100KB Files	4MB Files	2TB Disk Count
1 TB	100,000,000	10,000,000	250,000	1
10 TB	**1,000,000,000**	100,000,000	2,500,000	5
100 TB	**10,000,000,000**	**1,000,000,000**	25,000,000	50
4,000 TB	**400,000,000,000**	**40,000,000,000**	**1,000,000,000**	2,000

Why Not Use a Database?

- Users and system administrators are familiar with file systems
  - Backup, creation, etc are all well understood
- File systems handle partial failures pretty well
  - Being able to recover part of the stored data is useful for some applications
- File systems are “cheap” since they come with your operating system!

Why Not Use Lots of Little File Systems?

- Moves the problem from the file system designers off to application developers and users!
  - Application developers then need to code multi-file system aware applications
  - Users need to manually distribute files to various file systems
- Space allocation done statically
- Harder to optimize disk seeks
  - Bad to write to multiple file systems at once on the same physical device

Overview

- Why Worry About 1 Billion Files?
- ***Storage Building Blocks***
- Things File Systems Do & Performance
- File System Design Challenges & Futures

Traditional Spinning Disk

- Spinning platters store data
  - Modern drives have a large, volatile write cache
  - Streaming read/write performance roughly 100MB/sec
  - Seek latency limits drive to about 50-100 random IOPs
- This is the classic disk that file systems design for
- S-ATA 2TB drives go for under $200

External Disk Arrays

- External disk arrays can be very sophisticated
  - Large non-volatile cache used to store data
  - IO from a host normally lands in this cache
- Performance changes
  - Streaming reads and writes are vastly improved
  - Random writes and reads are fast when they hit cache
  - Random reads can be very slow when they miss cache
- Arrays usually start in the $20K range

SSD Devices

- S-ATA interface SSD's
  - Streaming reads & writes are reasonable
  - Random writes are normally slow
  - Random reads are great!
  - 1TB of S-ATA SSD is roughly $1k
- PCI-e interface SSD's enhance performance across the board
  - Provides array like bandwidth & low latency random IO
  - 320GB card for around $15k

How Expensive is 100TB?

- Build it yourself
  - 4 SAS/S-ATA expansion shelves which hold 16 drives ($12k)
  - 64 drives 2TB enterprise class drives ($19k)
  - A bit over $30k in total
- Buy any mid-sized array from a real storage vendor
- Most of us will have S-ATA JBODS or arrays
  - SSD's still too expensive

Overview

- Why Worry About 1 Billion Files?
- Storage Building Blocks
- ***Things File Systems Do & Performance***
- File System Design Challenges & Futures

Common Wisdom on the Web

> “Millions of files may work; but 1 billion is an utter absurdity. A file system that can store reasonably 1 billion small files in 7TB is an unsolved research issue...,”

Post on the ext3 mailing list, 9/14/2009

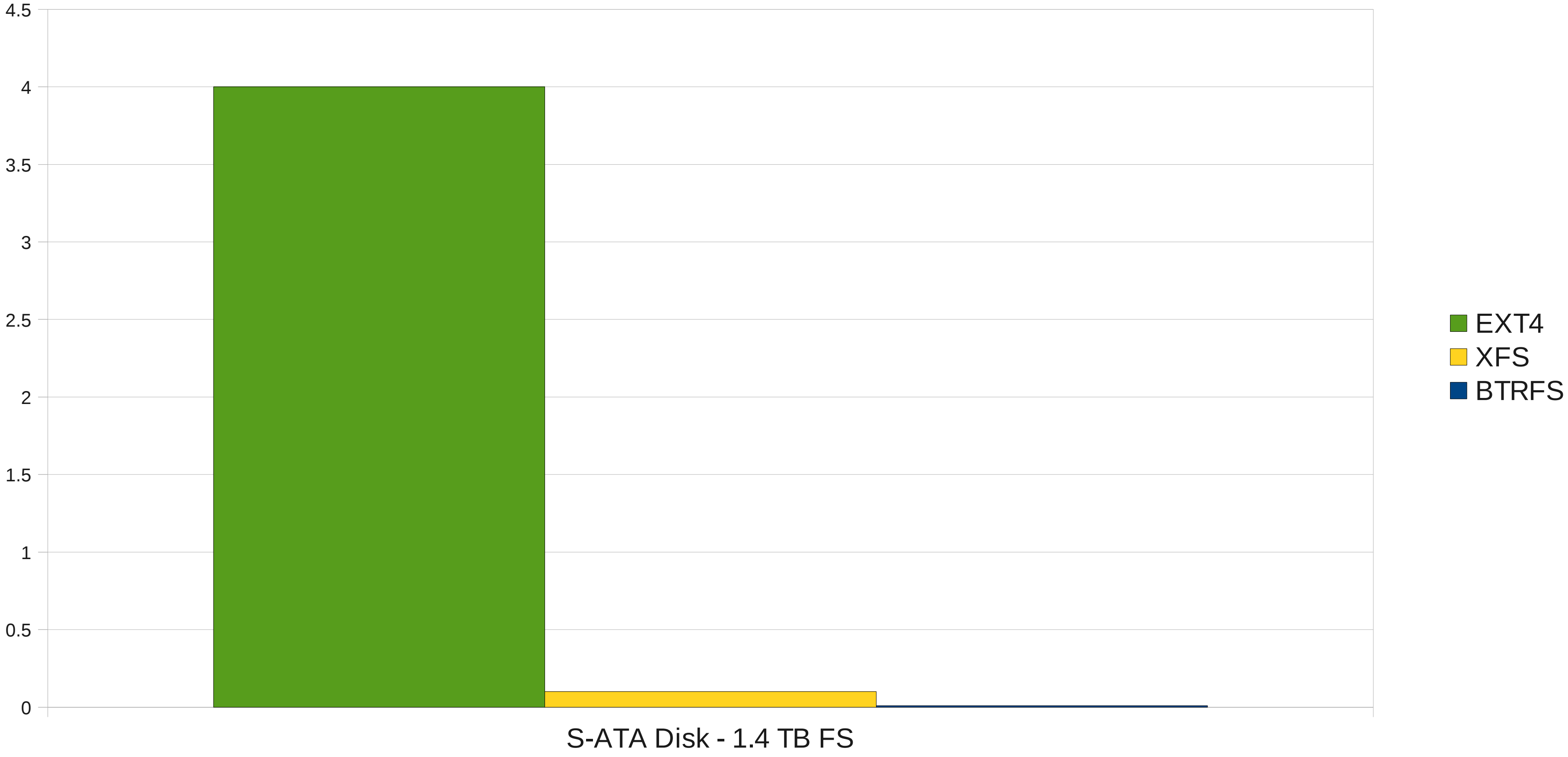
File System Life Cycle

- Creation of a file system (mkfs)
- Filling the file system
- Iteration over the files
- Repairing the file system (fsck)
- Removing files

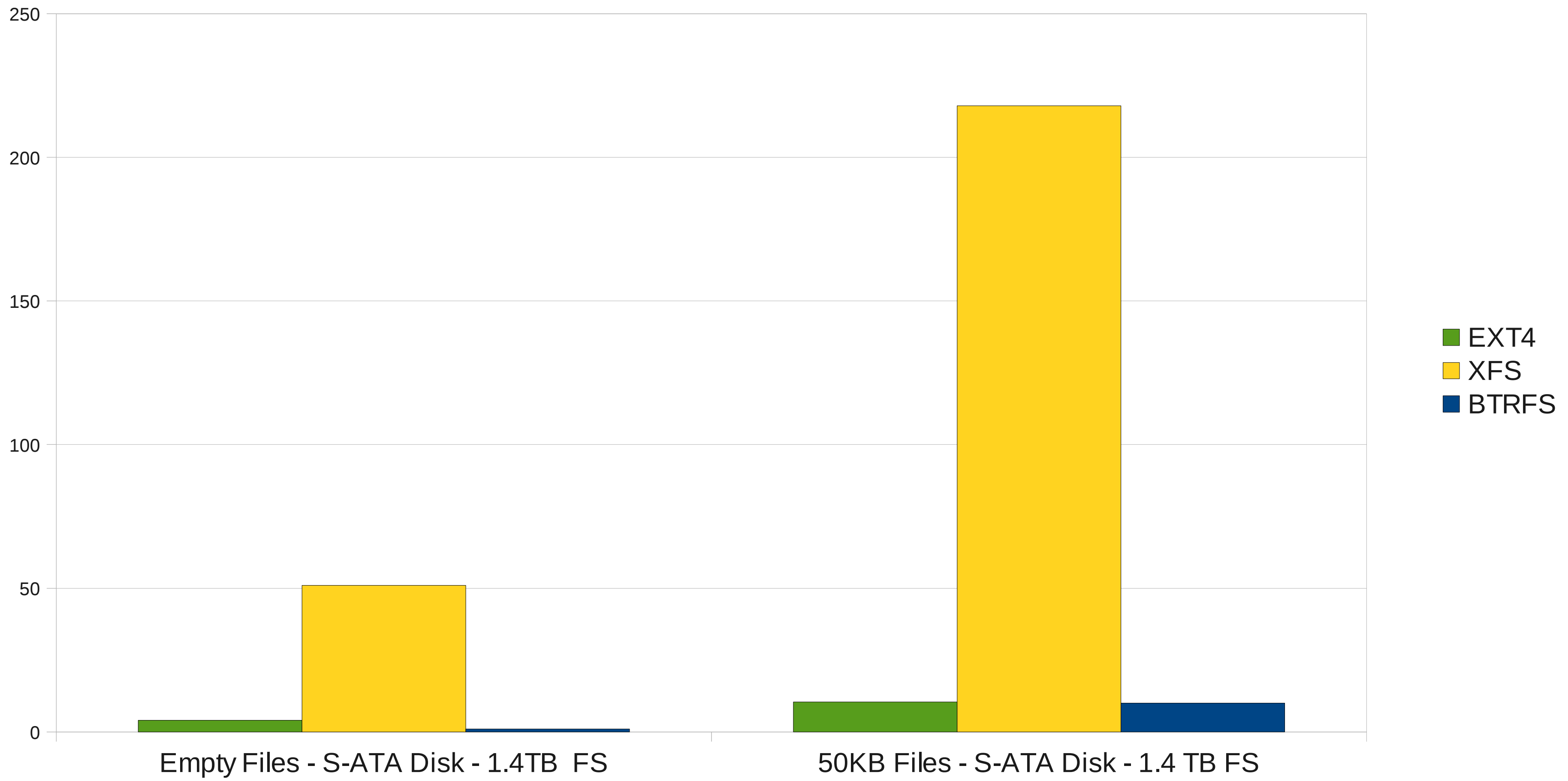
Starting Small – Just 1 Million Files

- Avoid the obvious bottlenecks
  - Spread files over 100 directories
  - Filled with “fs_mark” command
- Tested on a desktop class machine running Fedora 15
  - Intel(R) Core(TM)2 Quad CPU Q6600, 2.40GHz
  - 4GB DRAM
  - 1.5 TB S-ATA Seagate Disk (7200 RPM, 32 MB Cache)
  - 2.6.38.3-18 F15 kernel
- Expect billion file times to be 1,000 times longer!

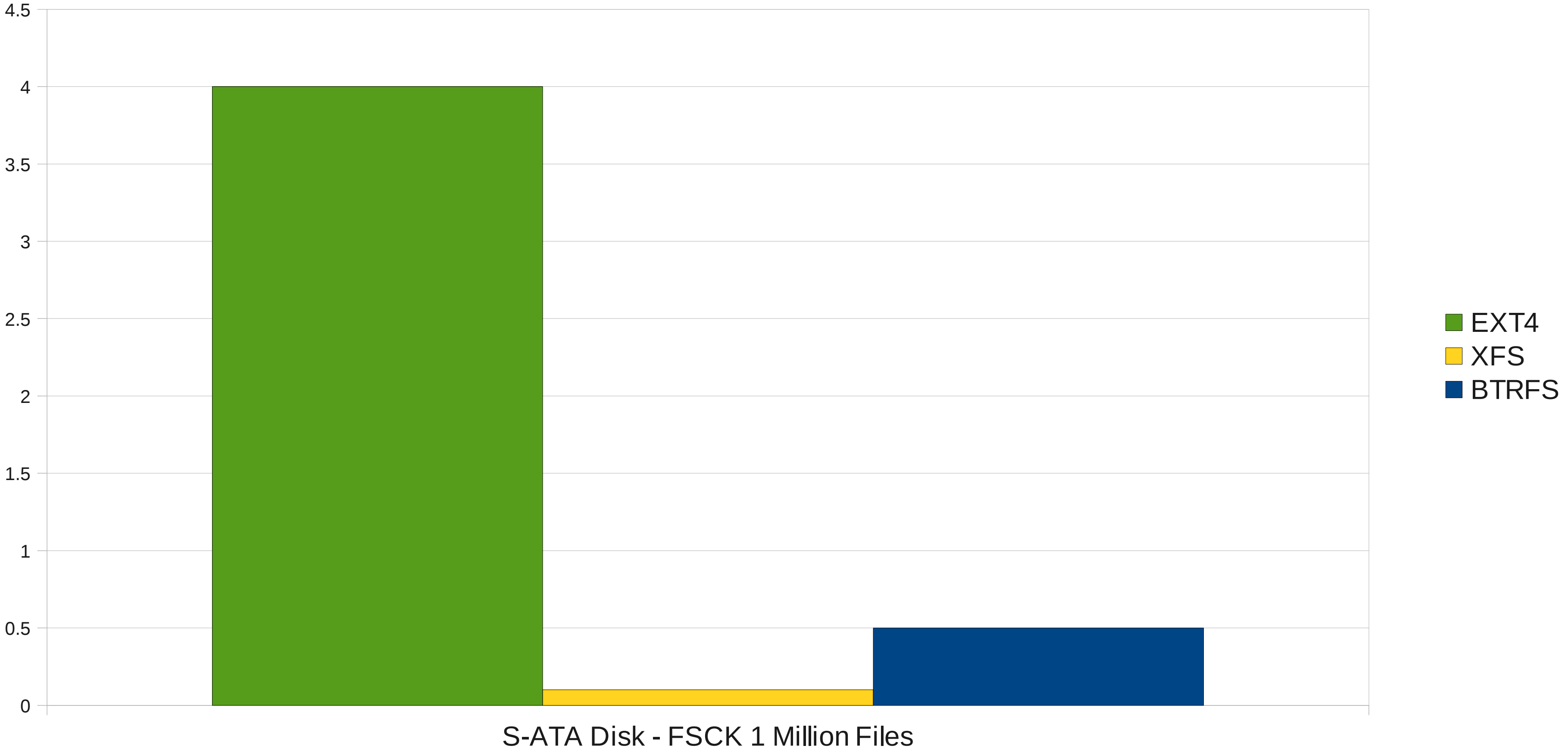
Making a File System Elapsed Time (Minutes)



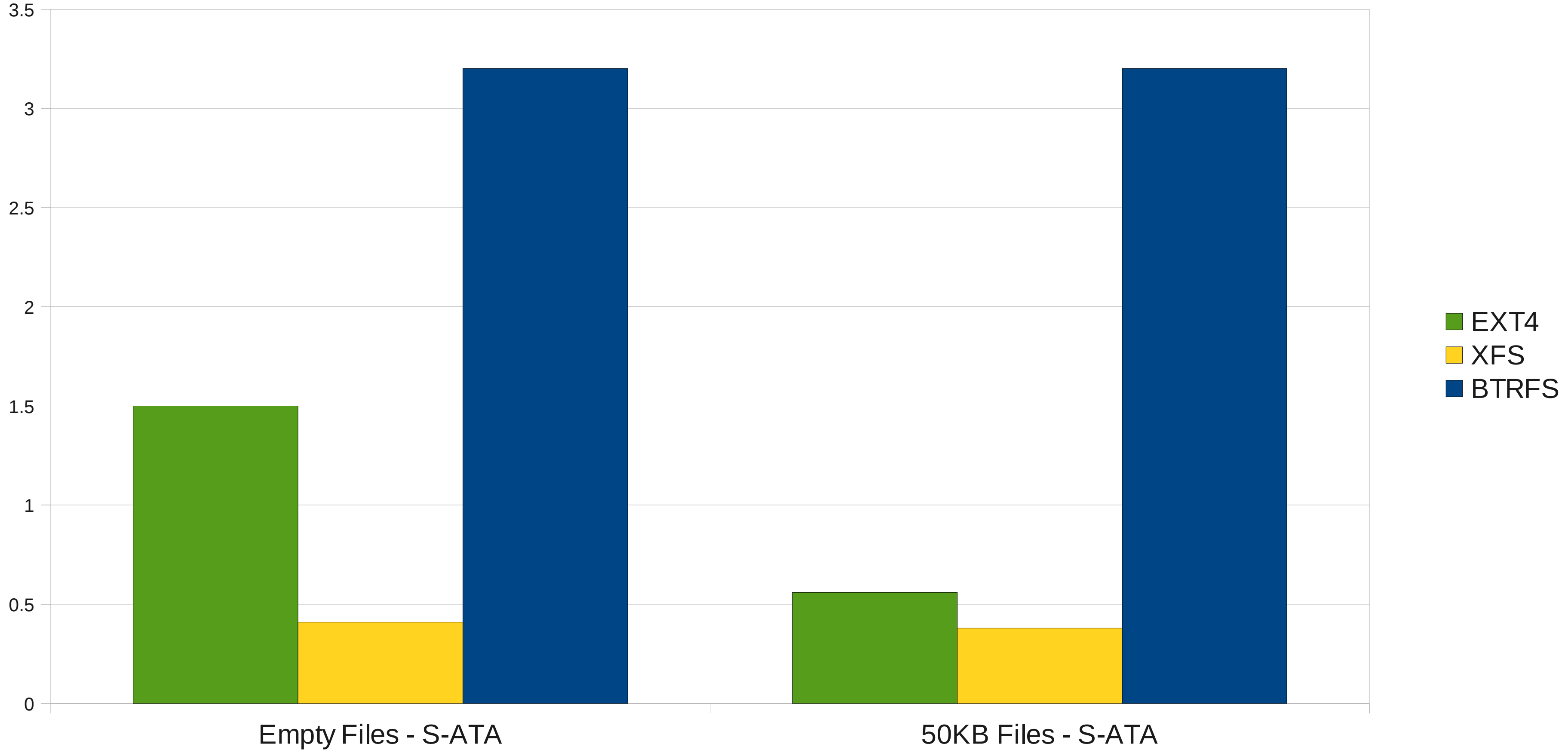
Creating 1M Files
Elapsed Time (Minutes)



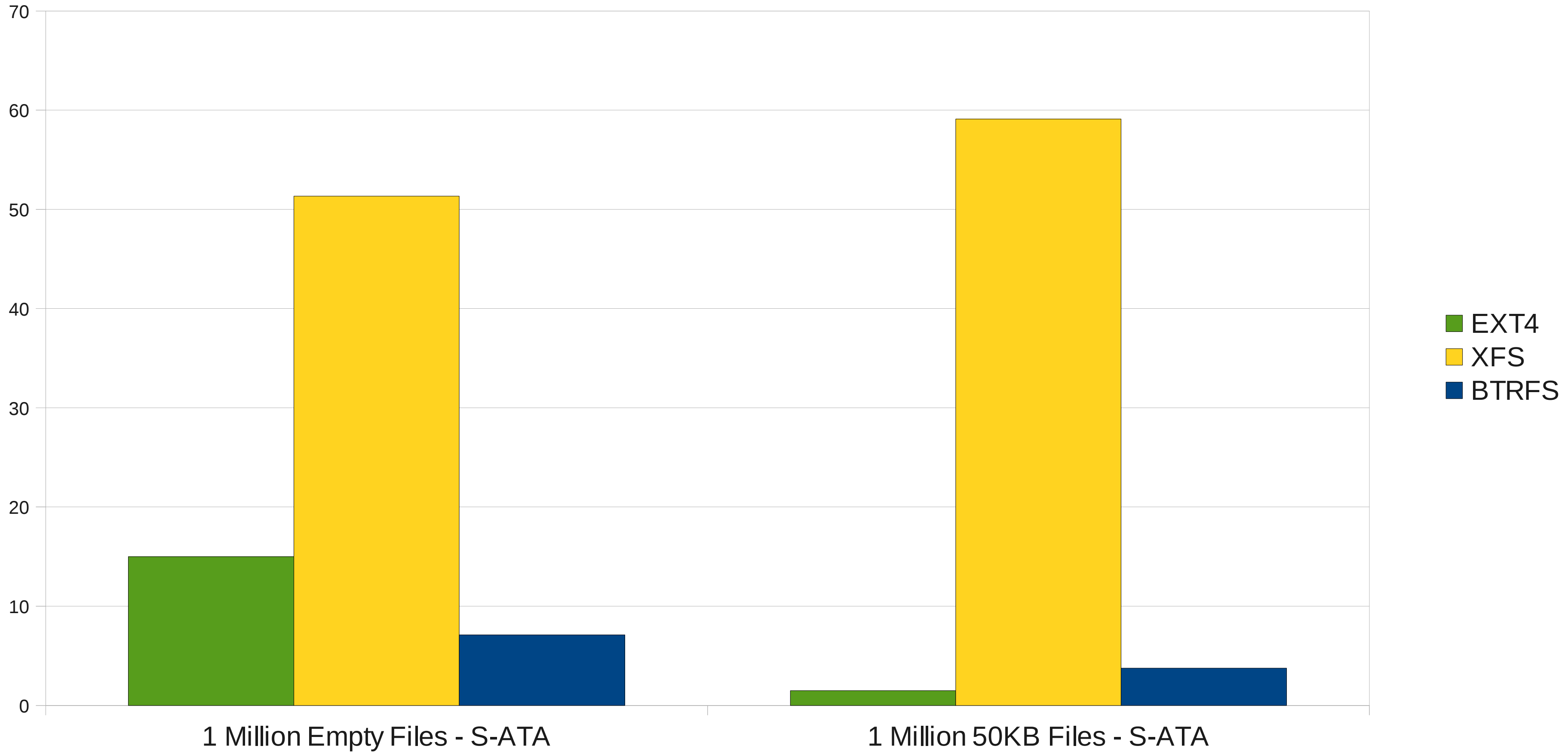
File System Repair
Elapsed Time (Minutes)



File System Iteration
Elapsed Time (Minutes)



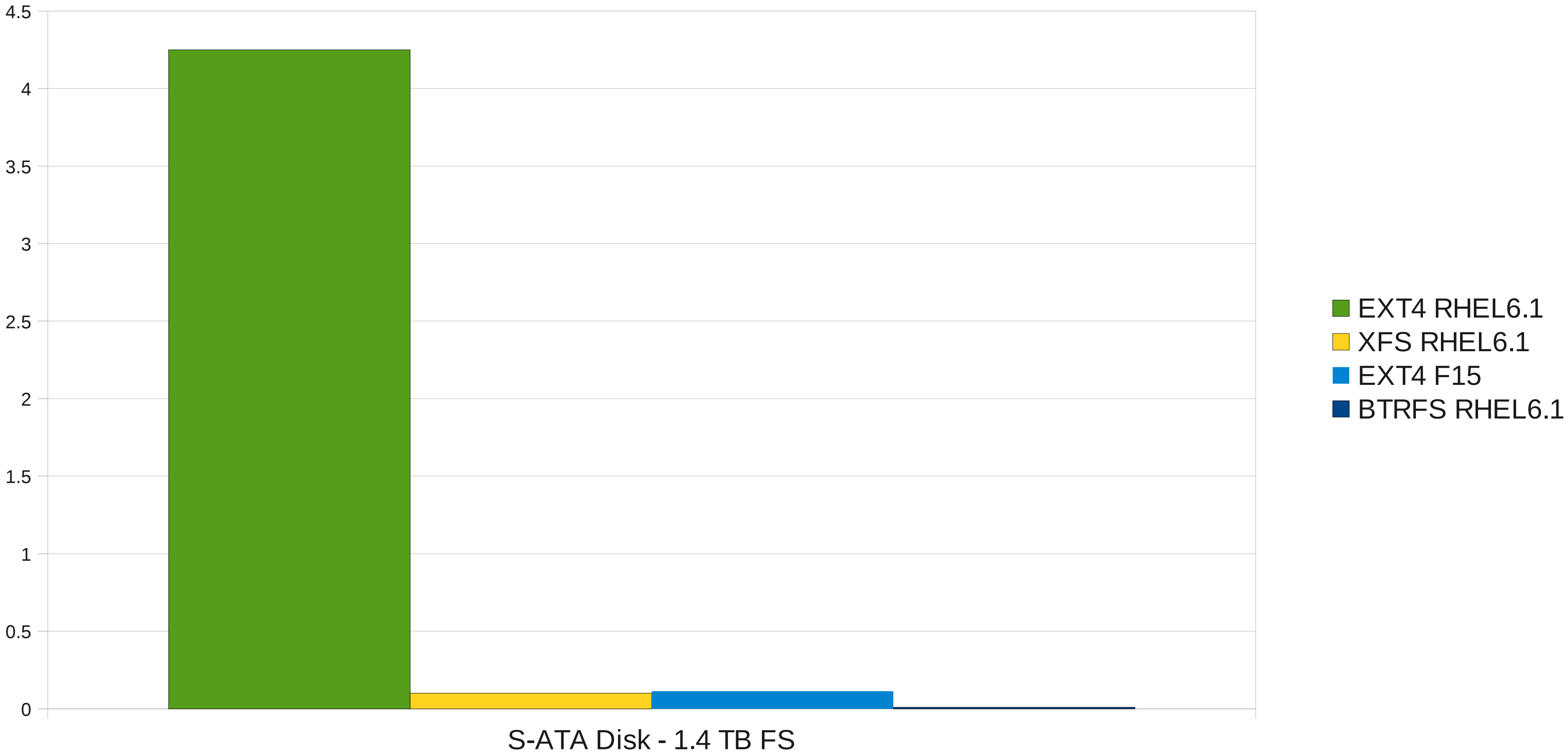
File System Deletion Elapsed Time (Minutes)



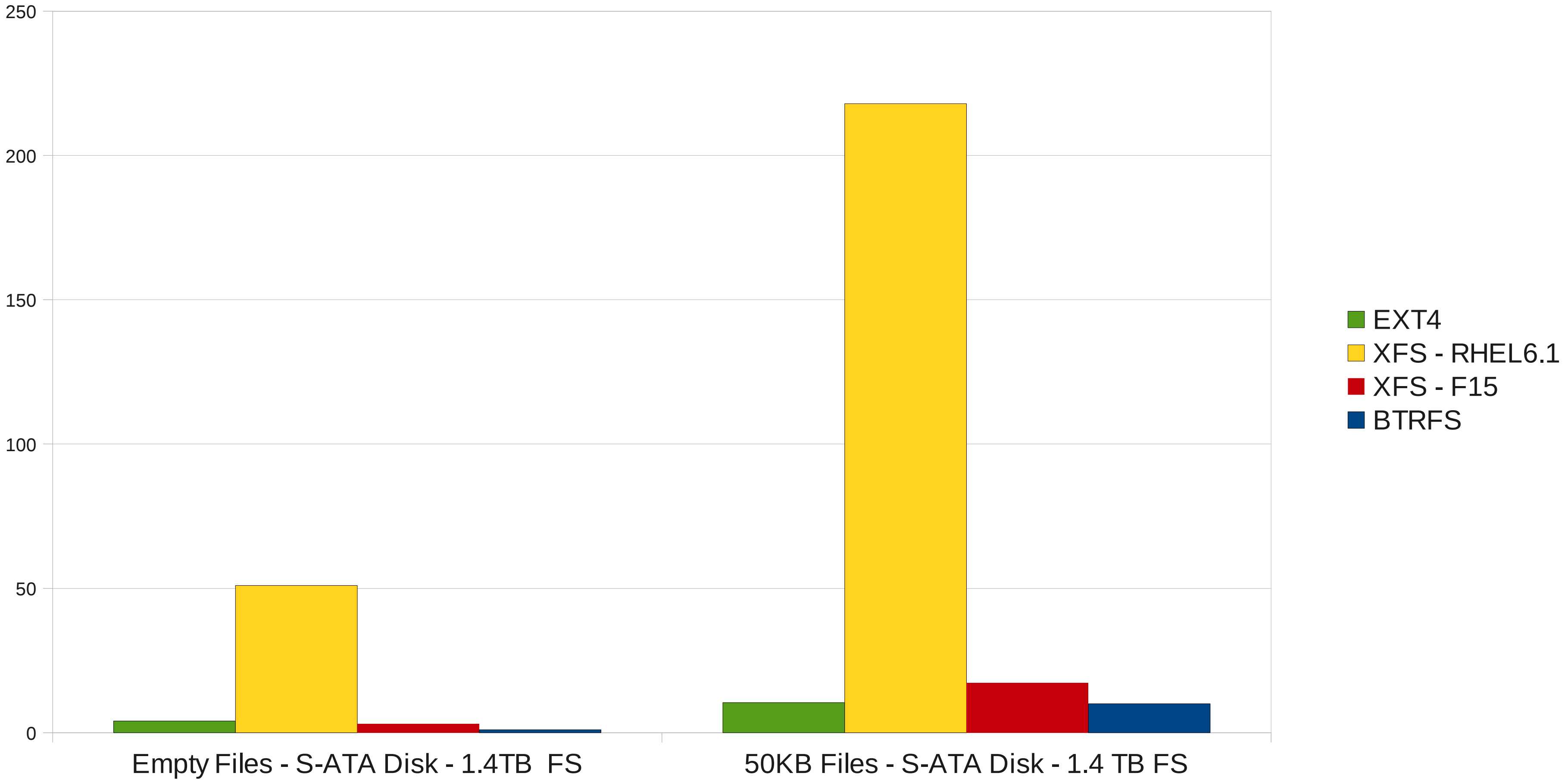
We Need to Do Better!

- Aiming for 1 billion files
- These performance challenges are in RHEL6.1
  - EXT4 is very slow at file system creation (mkfs)
    - Over 50 minutes to mkfs
  - XFS is slow at file creation and removal
    - 150 days to hit 1 billion 50KB files
  - BTRFS needs btrfsck to be finished
- Red Hat's Dave Chinner and Lukas Czerner tackled the XFS and EXT4 issues respectively

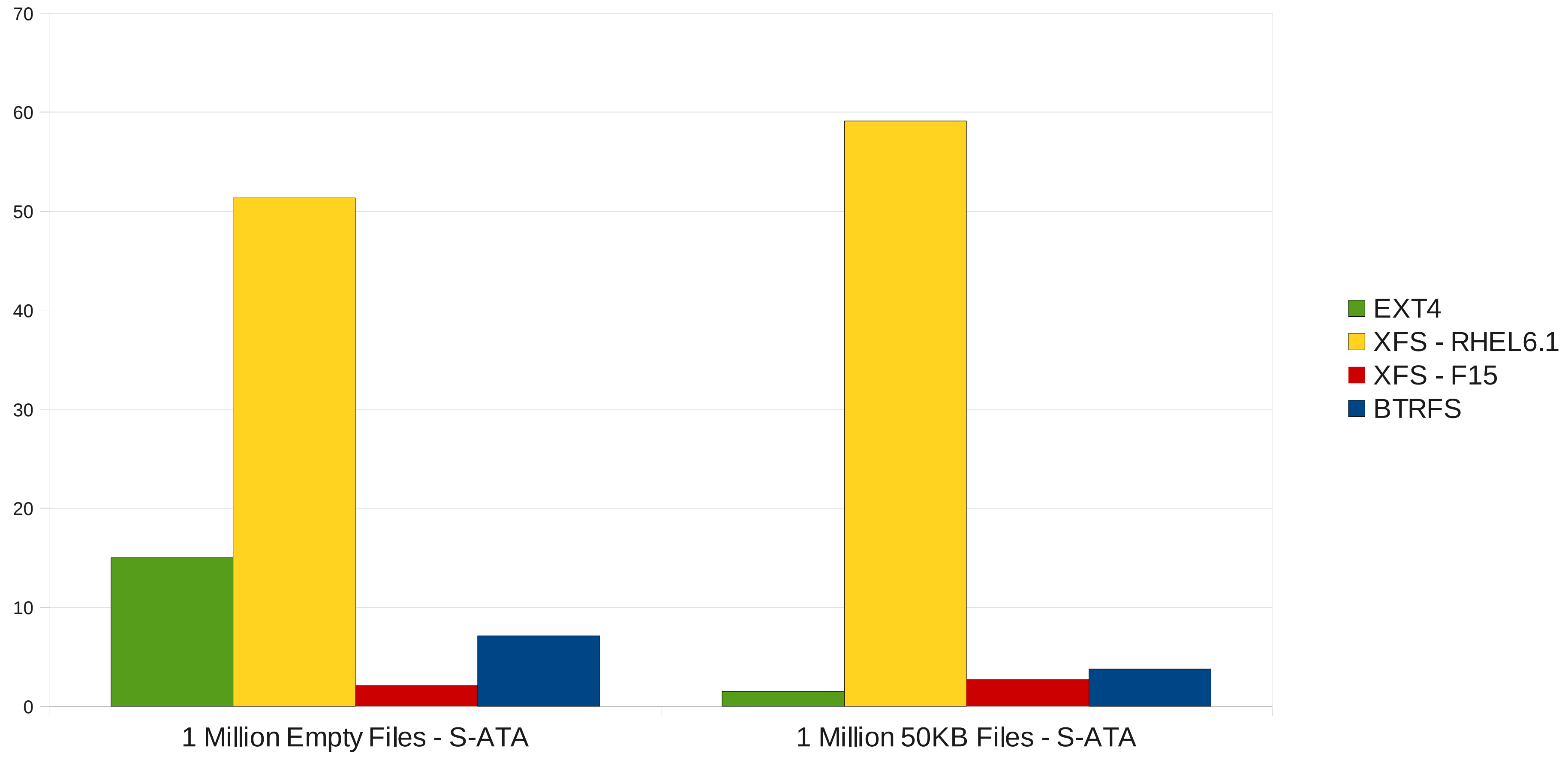
EXT4 MKFS Improvements Elapsed Time (Minutes)



XFS Creating Files Improvements Elapsed Time (Minutes)



File Deletion Improvements Elapsed Time (Minutes)



Summary of Upstream Performance Gains

- XFS file creation & deletion
  - Delayed logging mode (-o delaylog)
  - 12-17x faster for file creation
  - 21-24x faster for file deletion
- EXT4 File System Creation
  - Lazy inode initialization feature
  - 38x faster
- Path to RHEL: Upstream first, Fedora then RHEL...
  - RHEL6.2 is the target for the above features

Billion File Testing - What Rate is the Goal?

Goal	Rate Needed
24 Hours	11,500 Files/sec
1 Hour	277,777 Files/sec
1 Minute	16,666,666 Files/sec

- Rate of file creation can slow as file system ages
  - Some types of storage slow down when fully utilized
- FSCK must finish in a reasonable time – 4 hours?

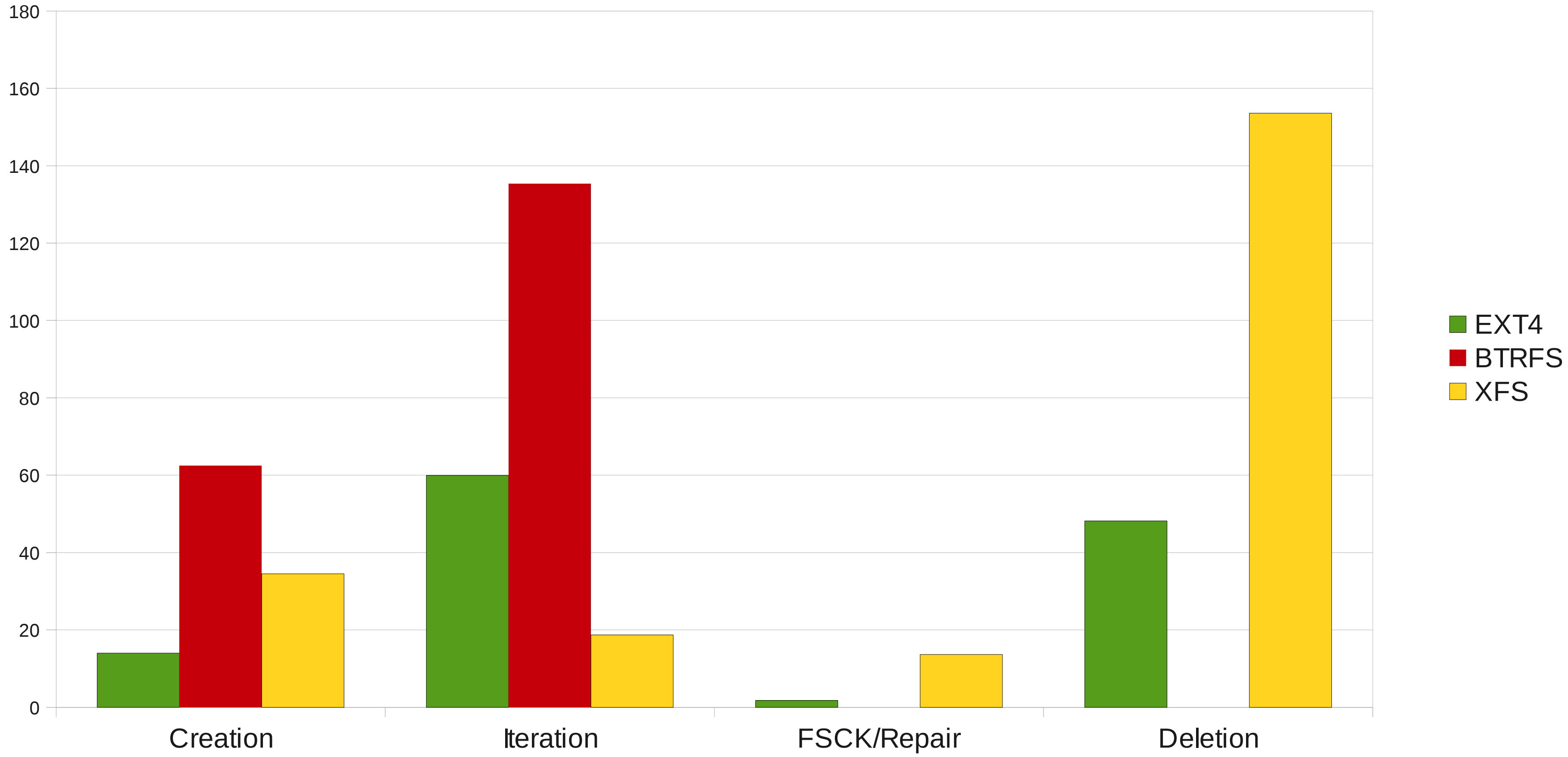
Hardware Impact on FS Zero Length File Creation Files/sec (Bigger is better)

File System	S-ATA	S-ATA SSD
XFS	5,599	10,442
EXT4	5,857	19,602
BTRFS	18,229	19,501

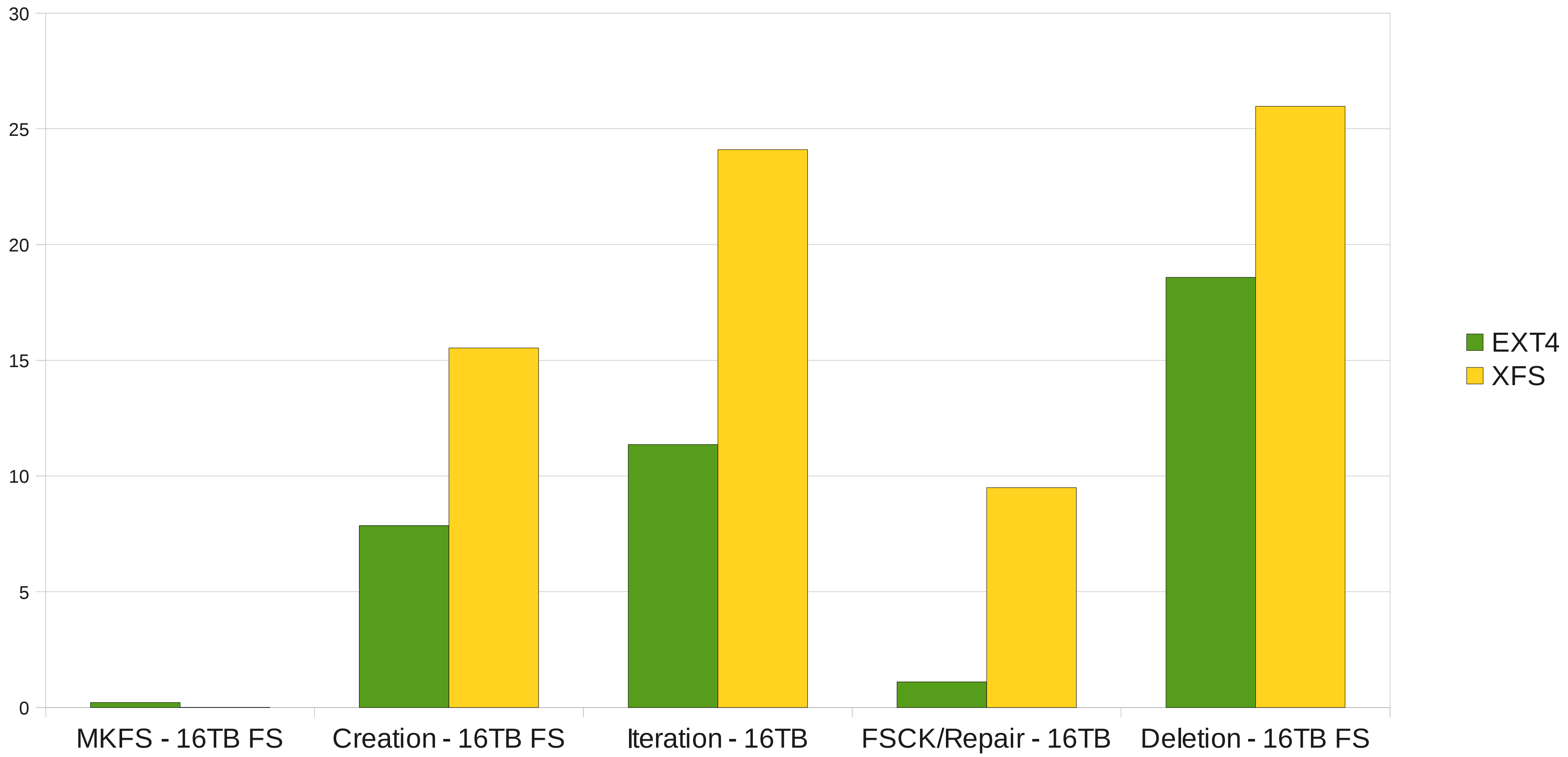
Billion File Testing Hardware Upgrade

- Enterprise storage settings (nobarrier mount option)
- Desktop class test as a baseline on RHEL6.1
  - 2 CPU, 2GB DRAM KVM guest
  - Single Near-line SAS drive with RAID card
- Server class hardware test with RHEL6.1 & Upstream
  - RHEL6.1 alpha kernel (2.6.32-122) vs 2.6.39-rc4
  - 16TB FS
    - 12 Near-line SAS drives & battery backed RAID card
    - 8 CPU, 8GB DRAM KVM guest

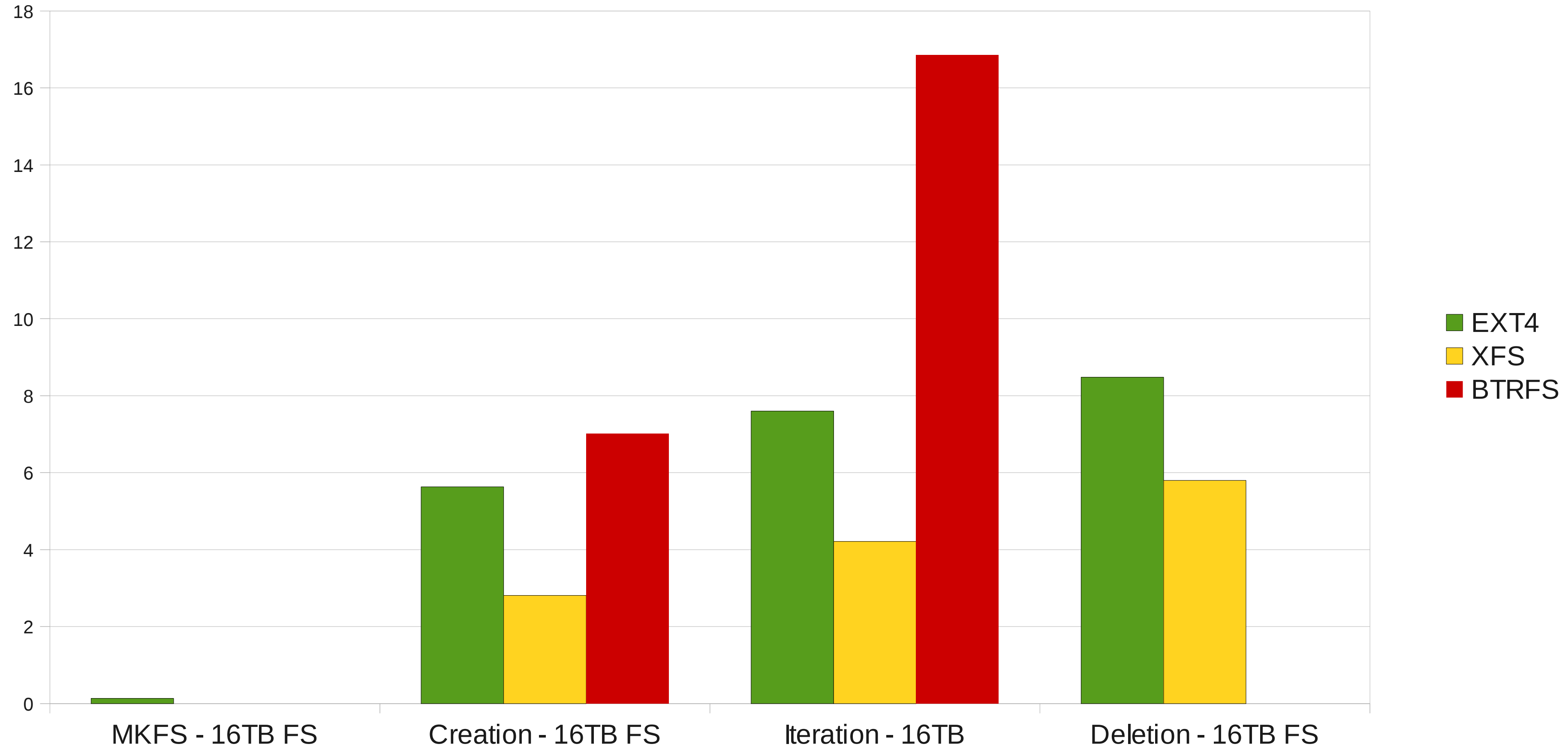
Desktop RHEL6.1 Billion Empty File Elapsed Time (Hours)



Server RHEL6.1 Billion Empty Files Elapsed Time (Hours)



Server 2.6.39-rc4 Billion Empty Files Elapsed Time (Hours)



Overview

- Why Worry About 1 Billion Files?
- Storage Building Blocks
- Things File Systems Do & Performance
- ***File System Design Challenges & Futures***

Pick Your Hardware Carefully!

- Big storage requires really big servers
  - xfs_repair can run in limited memory but runs faster with more DRAM
  - DRAM is relatively cheap so avoid paging!
- Faster storage building blocks can be hugely helpful
  - Small file work loads are very IO/sec limited
  - Using high performance, low latency storage helps
  - Highest performance storage is still small capacity

Kernel Challenges

- “ls” is a really bad idea
  - Iteration over that many files can be very IO intensive
  - Applications use readdir() & stat()
  - Supporting d_type avoids the stat call but is not universally done
- Lock scalability
  - With faster storage hardware, lock contention has become an issue for FS & IO stack
- Block caching schemes mix of expensive SSD and high capacity, cheap disks

Things to Keep in Mind

- One Billion Files is really still quite challenging
  - Expect to wait for hours (if not days!)
- Remote replication & backup are painful at this scale
  - Iteration & read rates hurt by concurrent IO
  - Done on a live system - cgroups can definitely help
- Things that last this long will experience failures
  - Checkpoint/restart support
  - Robust IO error handling needed

Questions?

Ric Wheeler
rwheeler@redhat.com

Resources

- Red Hat videos (search for “ext4” or “XFS”)
  - https://access.redhat.com/knowledge/videos
- Mailing lists include
  - linux-ext4@vger.kernel.org, xfs@oss.sgi.com, inux-btrfs@vger.kernel.org
- Linux kernel coverage: http://lwn.net

Summit File System Talks

- Wed. 5:30pm: File System Performance – John Shakshober
- Wed. 2:00PM: NFS – The Next Generation – Steve Dickson
- Thurs. 3:10 - Tuning the Red Hat Enterprise Linux 6 I/O Subsystem & Using I/O cGroups – Jeff Moyer and Vivek Goyal
- Wed. 4:20PM - Building a Cloud Filesystem – Jeff Darcy and Mark Wagner

LIKE US ON FACEBOOK

www.facebook.com/redhatinc

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#redhat

READ THE BLOG

summitblog.redhat.com

GIVE US FEEDBACK

www.redhat.com/summit/survey