One Billion Files:

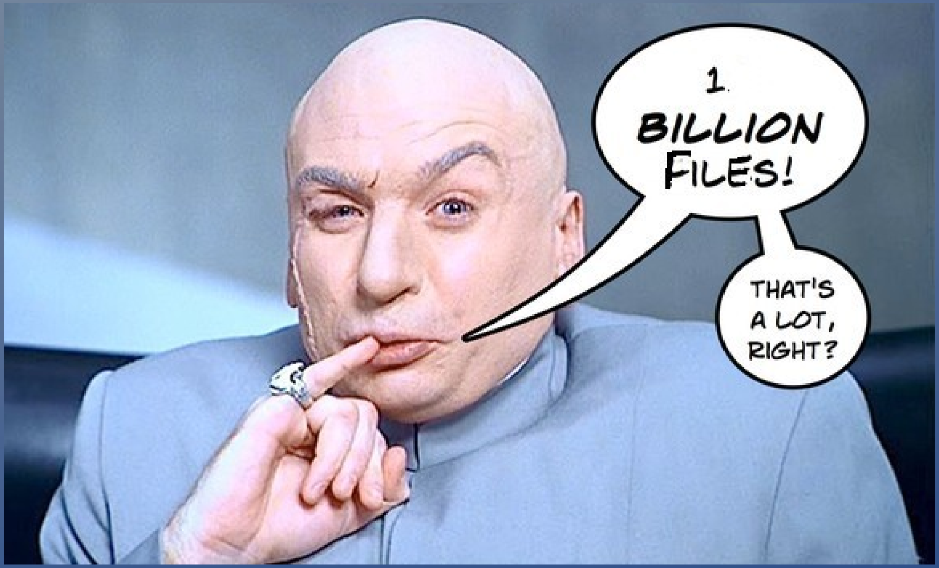
Scalability Limits in Linux File Systems

Ric Wheeler
Architect & Manager, Red Hat
August 10, 2010

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 Much Storage Do 1 Billion Files Need?

Disk Size	10KB Files	100KB Files	4MB Files	4TB 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	3
100 TB	**10,000,000,000**	**1,000,000,000**	25,000,000	25
4,000 TB	**400,000,000,000**	**40,000,000,000**	**1,000,000,000**	1,000

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?

- Pushes the problem from the file system designers down
  - 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 (16+ MB)
  - Streaming read/write performance of a single S-ATA drive can sustain roughly 100MB/sec
  - Seek latency bounds random IO to the order of 50-100 random IO's/sec
- This is the classic platform that operating systems & applications are designed for
- High end 2TB drives go for around $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 without hitting spinning media
- 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 and 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

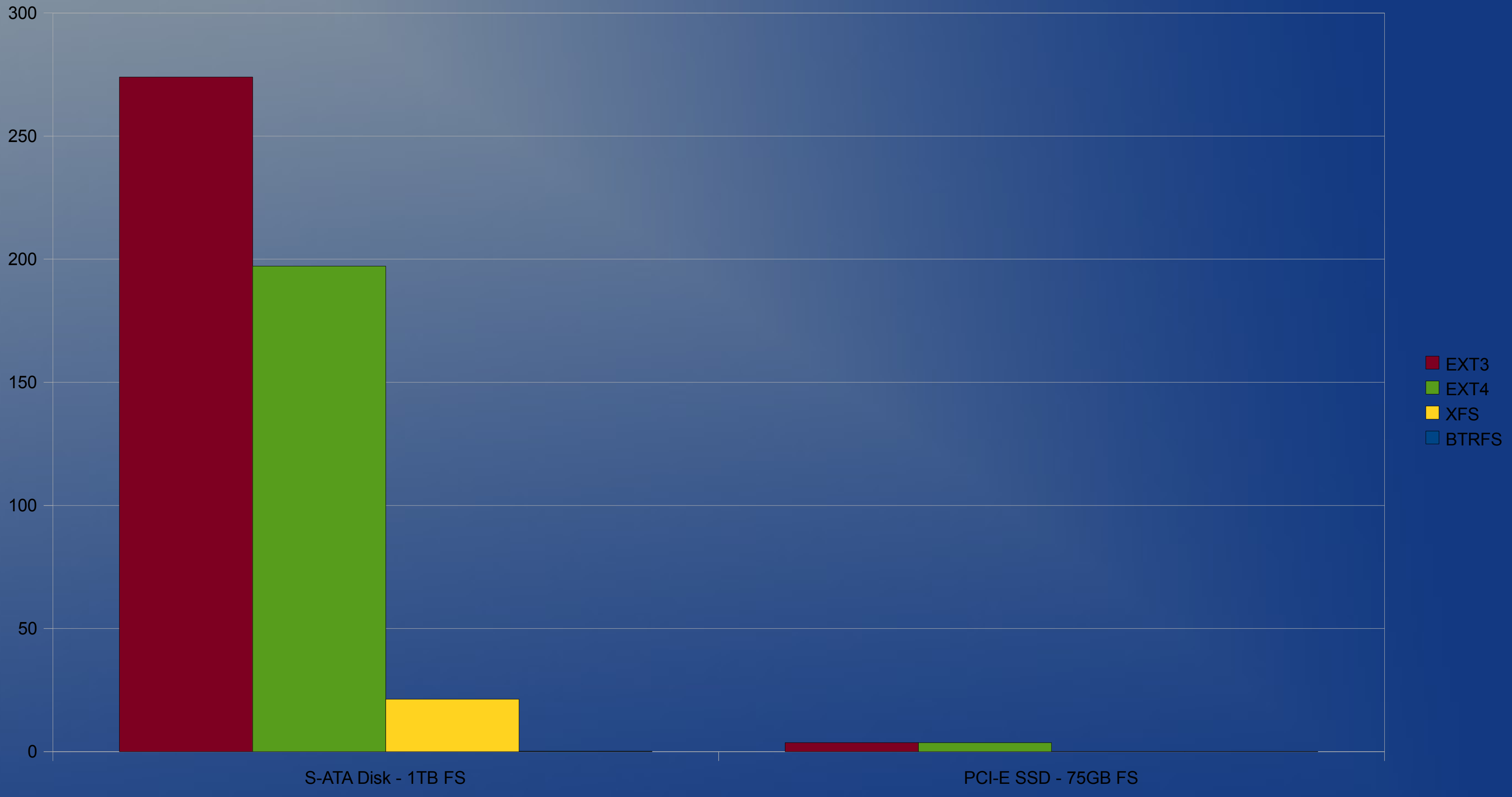
Overview

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

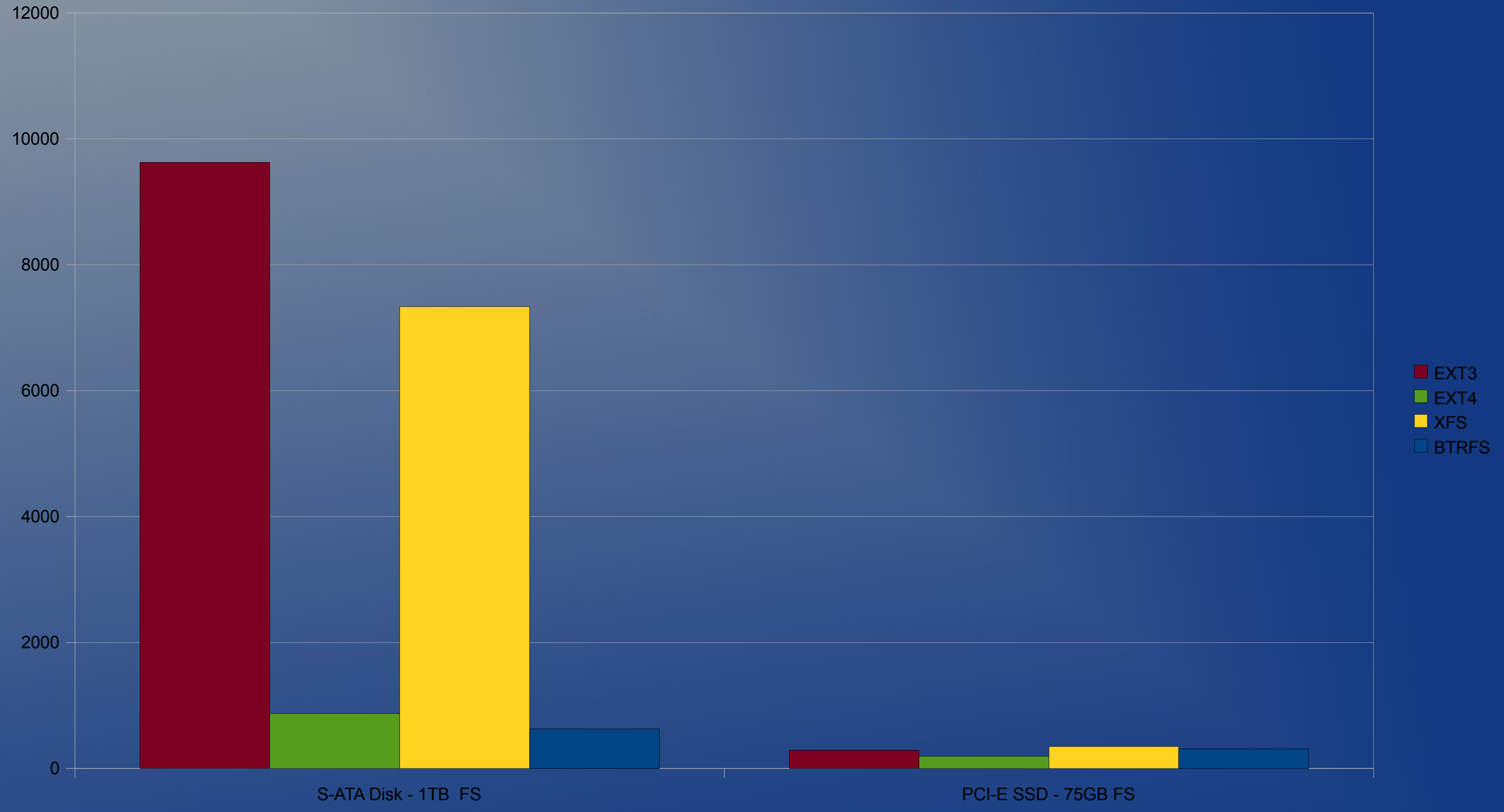
File System Life Cycle

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

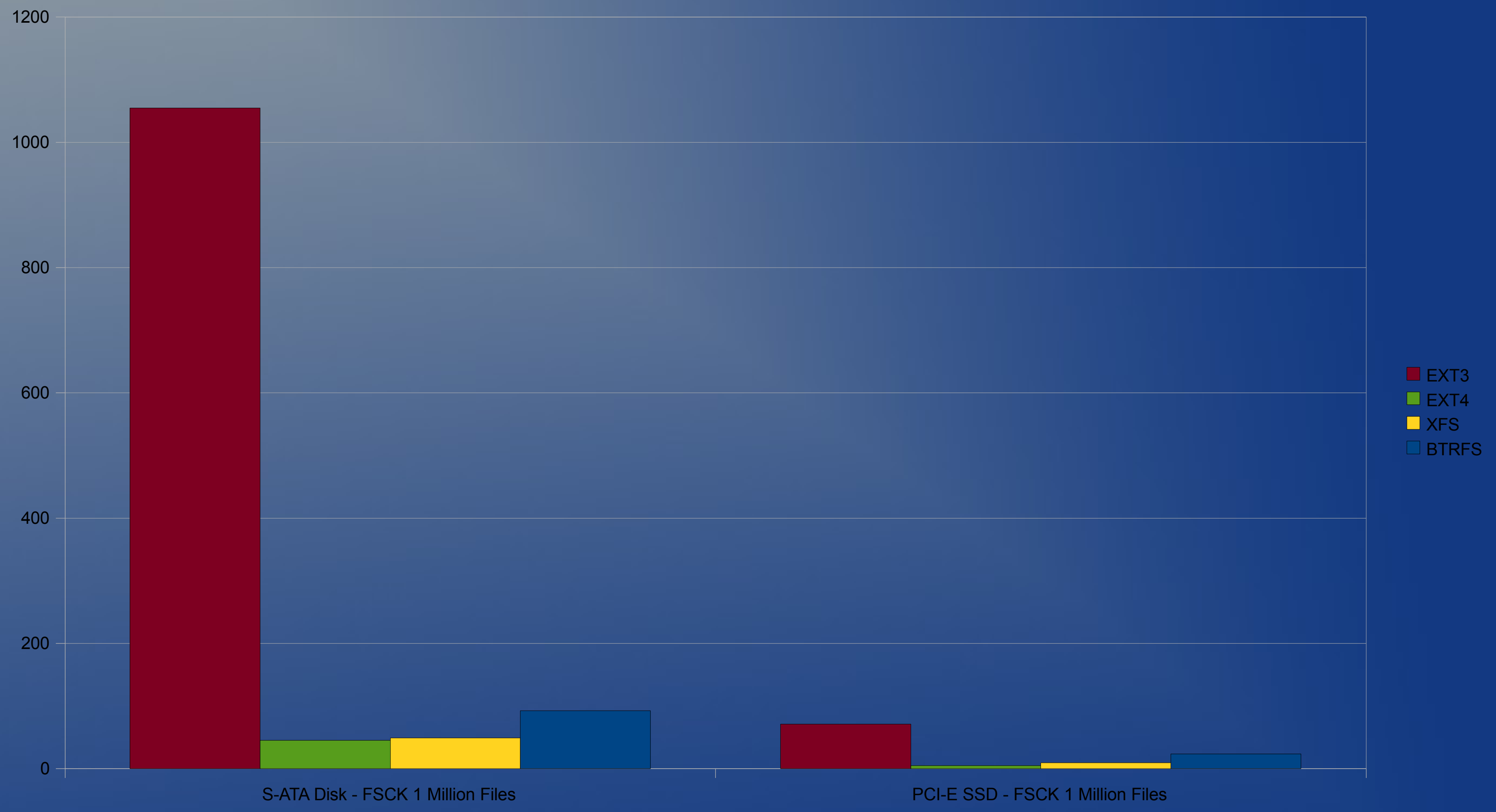
Making a File System – Elapsed Time (sec)



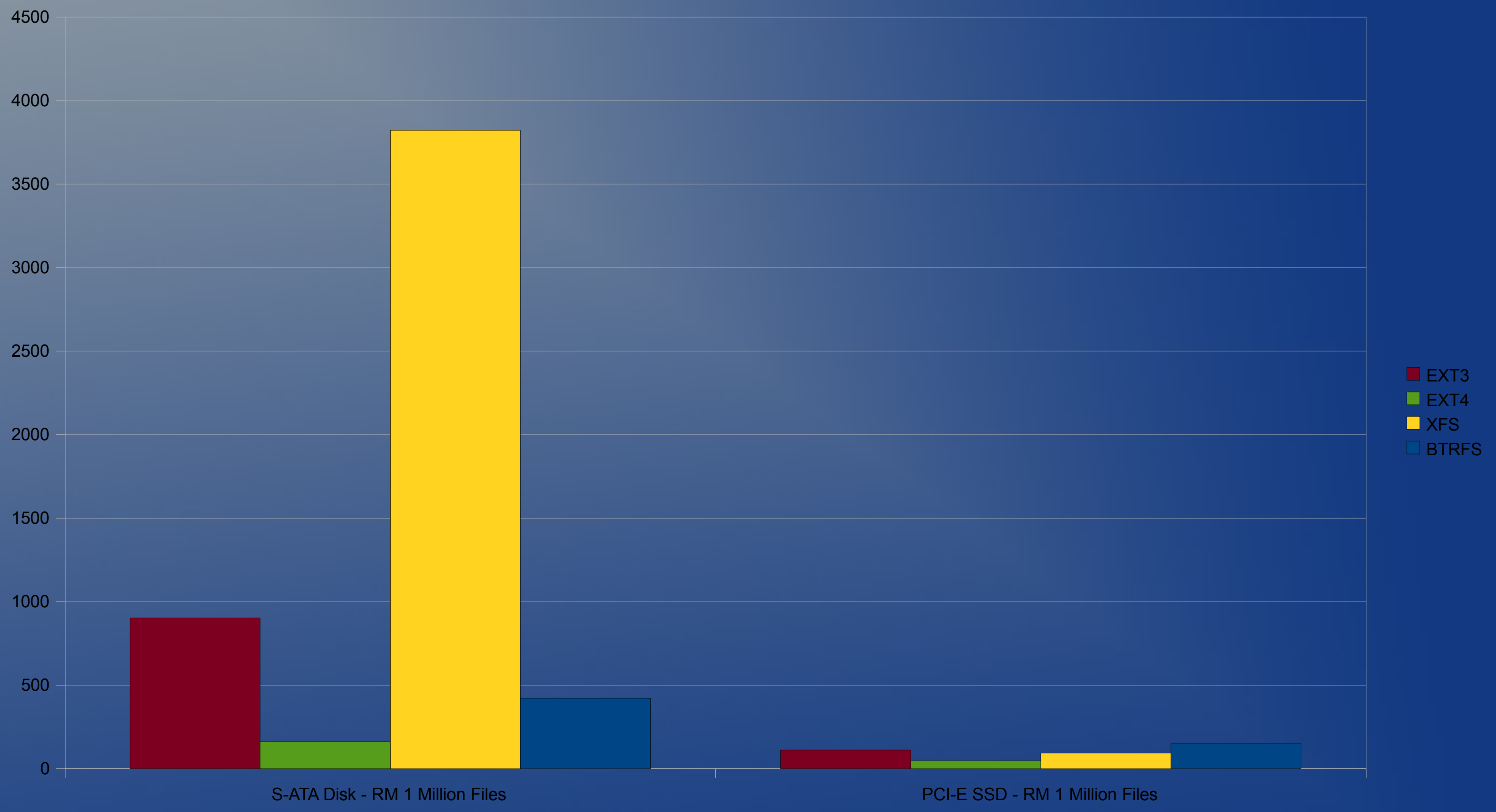
Creating 1M 50KB Files – Elapsed Time (sec)



File System Repair – Elapsed Time



RM 1 Million Files – Elapsed Time



What about the Billion Files?

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

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

What about the Billion Files?

“Strangely enough, I have been testing ext4 and stopped filling it at a bit over 1 billion 20KB files on Monday (with 60TB of storage). Running fsck on it took only 2.4 hours.”

My reply post on the ext3 mailing list, 9/14/2009.

Billion File Ext4

- Unfortunately for the poster an Ext4 finished earlier that week
  - Used system described earlier
- MKFS
  - 4 hours
- Filling the file system to 1 billion files
  - 4 days
- Fsck with 1 billion files
  - 2.5 hours
- Rates consistent for zero length and small files

What We Learned

- Ext4 fsck needs a lot of memory
  - Ideas being floated to encode bitmaps more effectively in memory
- Trial with XFS highlighted XFS's weakness for meta-data intensive workloads
  - Work ongoing to restructure journal operations to improve this
- Btrfs testing would be very nice to get done at this scale

Overview

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

Size the Hardware Correctly

- Big storage requires really big servers
  - FSCK on the 70TB, 1 billion file system consumed over 10GB of DRAM on ext4
  - xfs_repair was more memory hungry on a large file system and used over 30GB of DRAM
- Faster storage building blocks can be hugely helpful
  - Btrfs for example can use SSD's devices for metadata & leave bulk data on less costly storage

Iteration over 1 Billion is Slow

- "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
- Performance of enumeration of small files
  - Runs at roughly the same speed as file creation
  - Thousands of files per second means several days to get a full count

Backup and Replication

- Remote replication or backup to tape is a very long process
  - Enumeration & read rates tank when other IO happens concurrently
  - Given the length of time, must be done on a live system which is handling normal workloads
  - Cgroups to the rescue?
- Things that last this long will experience failures
  - Checkpoint/restart support is critical
  - Minimal IO retry on a bad sector read