XFS:

There and Back....

.... and There Again?

Dave Chinner
<david@fromorbit.com>
<dchinner@redhat.com>

Overview

- Story Time
- Serious Things
- These Days
- Shiny Things
- Interesting Times

Story Time

- Way back in the early '90s
- Storage exceeding 32 bit capacities
- 64 bit CPUs, large scale MP
- Hundreds of disks in a single machine

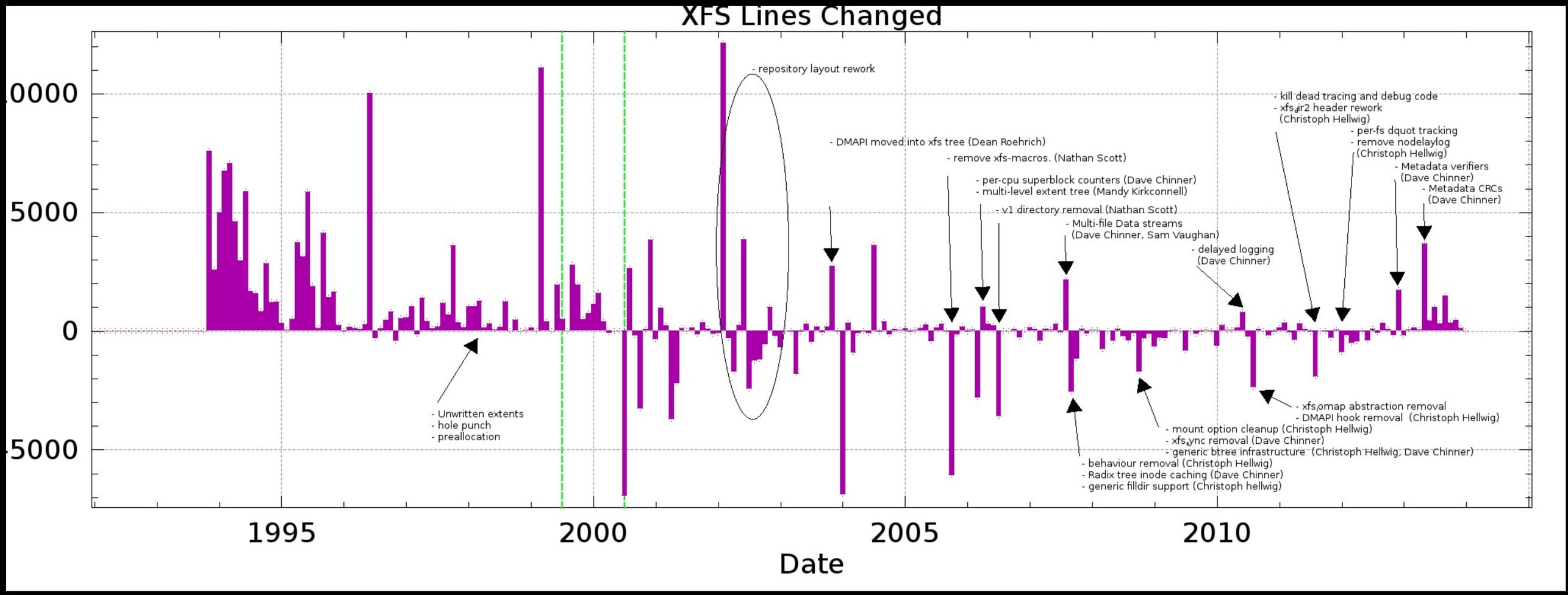
"x" is for Undefined

xFS had to support:

- Fast Crash Recovery
- Large File Systems
- Large, Sparse Files
- Large, Contiguous Files
- Large Directories
- Large Numbers of Files

```
- Scalability in the XFS File System, 1995
http://oss.sgi.com/projects/xfs/papers/xfs_usenix/index.html
```

The Early Years



The Early Years

- Late 1994: First Release, Irix 5.3
- Mid 1996: Default FS, Irix 6.2
- Already at Version 4
  - Attributes
  - Journalled Quotas
  - link counts > 64k
  - feature masks

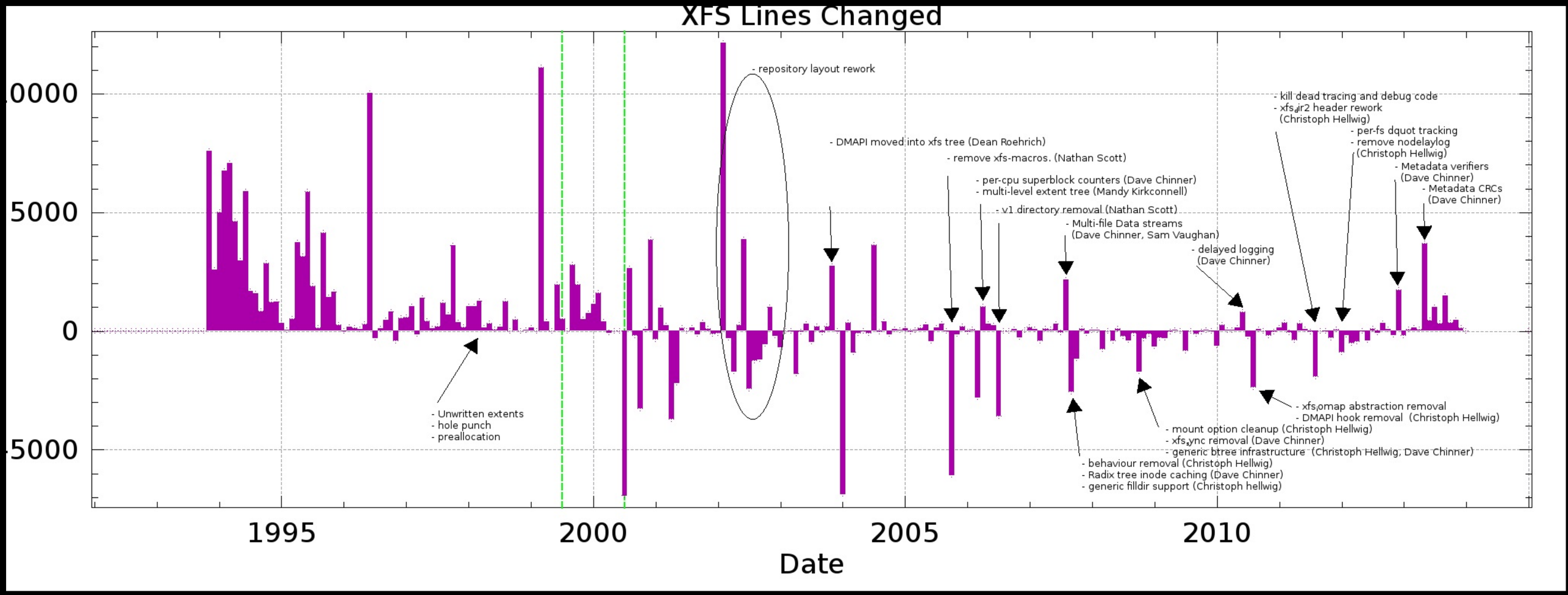
The Early Years

- Allocation alignment to storage geometry (1997)
- Unwritten extents (1998)
- Version 2 directories (1999)
  - mkfs time configurable block size
  - Scalability to tens of millions of directory entries

What's that Linux Thing?

- Feature development mostly stalled
- Irix development focussed on CXFS
- New team formed for Linux XFS port!
  - Encumberance review!
  - Linux was missing lots of bits XFS needed
  - Lot of work needed

That Linux Thing?



Light that fire!

- 2000: SGI releases XFS under GPL
- 2001: First stable XFS release
- 2002: XFS merged into 2.5.36
- JFS follows similar timeline

Bonnie++ Speed Racing

AKA: the origin of "noatime,nodiratime,logbufs=8" meme

"I decide that the best combination of options appears to be a XFS filesystem created with 45 allocation groups, but a 64 megabyte log, and mounted with 8 log buffers, and atime disabled (mount -o noatime,nodiratime,logbufs=8)"

– The Custodian, 25 June 2003

http://everything2.com/title/Filesystem+performance+tweaking+with+XFS+on+Linux

Back to the Features!

- group quotas, diverges from Irix
- V2 log format (2002)
- Inode cluster delete (2002)
- Configurable sector sizes (2002)
- 2.4/2.6 dev tree unification (2004)
- feature mask expansion (2004)

Major Achievement Unlocked

SLES 9 ships with full XFS support in 2004

Flux Capacitors

"I was actually told about this by an XFS engineer, who discovered this the hardware. Their solution was to add a power fail interrupt and bigger capacitors in the power supplies in SGI hardware, and in Irix, when the power fail interrupt triggers, the first thing the OS does is to run around frantically aborting IO transfers to the disk."

– Ted T'so, 19 December, 2004
http://zork.net/~nick/mail/why-reiserfs-is-teh-sukc

Magic Zeroing #1

"Unfortunately, the XFS filesystem just doesn't allow undeletion - for example, all non-allocated blocks are automatically zeroed, so all data on them is lost."

– Oozi, 25 January, 2004
http://forums.justlinux.com/showthread.php?121069-XFS-data-recovery-really-impossible

Magic Zeroing #2

"This issue is completely different from the XFS issue of zero'ing all open files on an unclean shutdown, of course."

– Ted T'so, 19 December, 2004
http://zork.net/~nick/mail/why-reiserfs-is-teh-sukc

Back to the Features 2

- dynamic attribute fork size (2005)
- code unification with Irix (2005)
- project quotas on Linux again (2005)
- per-cpu superblock counters (2006)

10GB/s on 2.6.16 on 24p Altix machine

XFS DESTROYS!

"XFS leaves garbage in file if app does write-new-then-rename without f(data)sync."

- Ubuntu bug 37435, opened 31 March 2006, closed: WONTFIX
https://bugs.launchpad.net/ubuntu/+source/linux-source-2.6.15/+bug/37435

The O_PONIES wars have begun....

Back to the Features 3

- lazy on-disk superblock counters (2007)
- filestreams allocator (2007)
- radix tree based inode caching (2007)
- behaviour layer removal (2007)
- Ascii case insentivity (2008)

Null files on crash problem fixed!

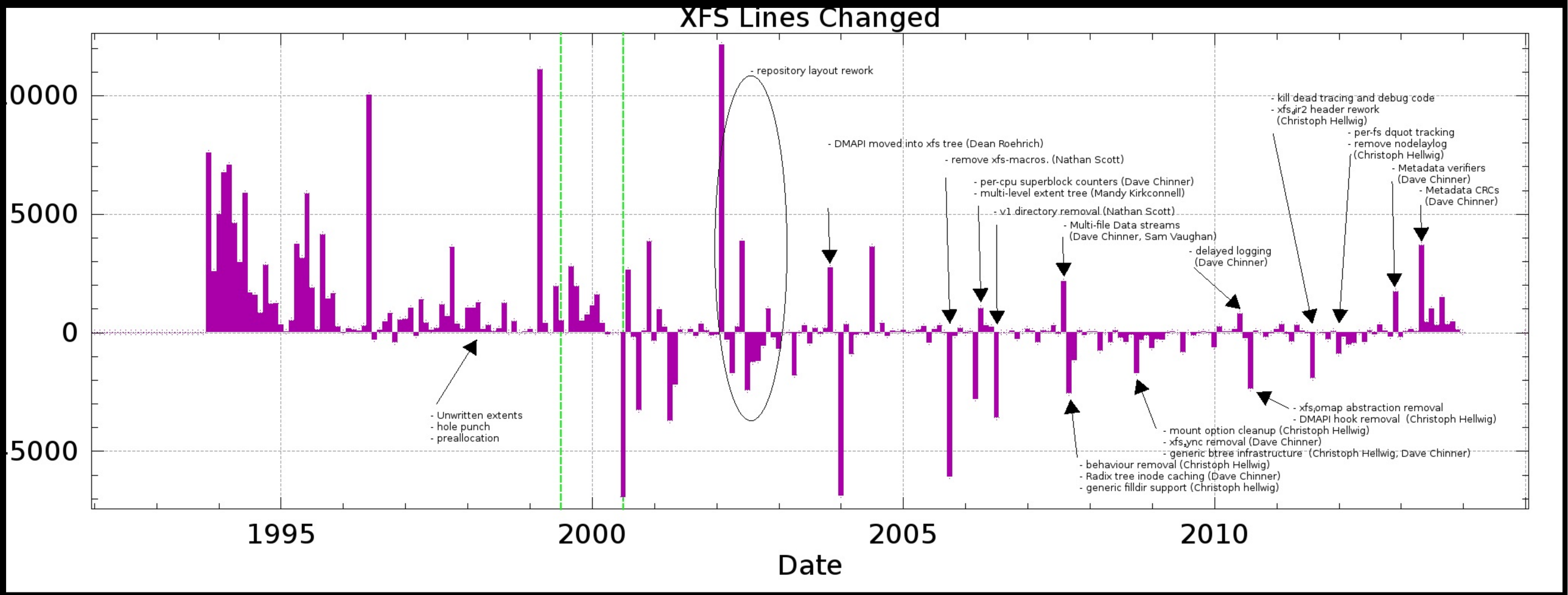
Sad Days

- SGI Linux XFS engineering team disbanded in 2009
- Community steps in to maintain XFS until SGI reorganises
- SGI retains intermittent maintainership until end of 2013
- XFS development shifts to the wider community

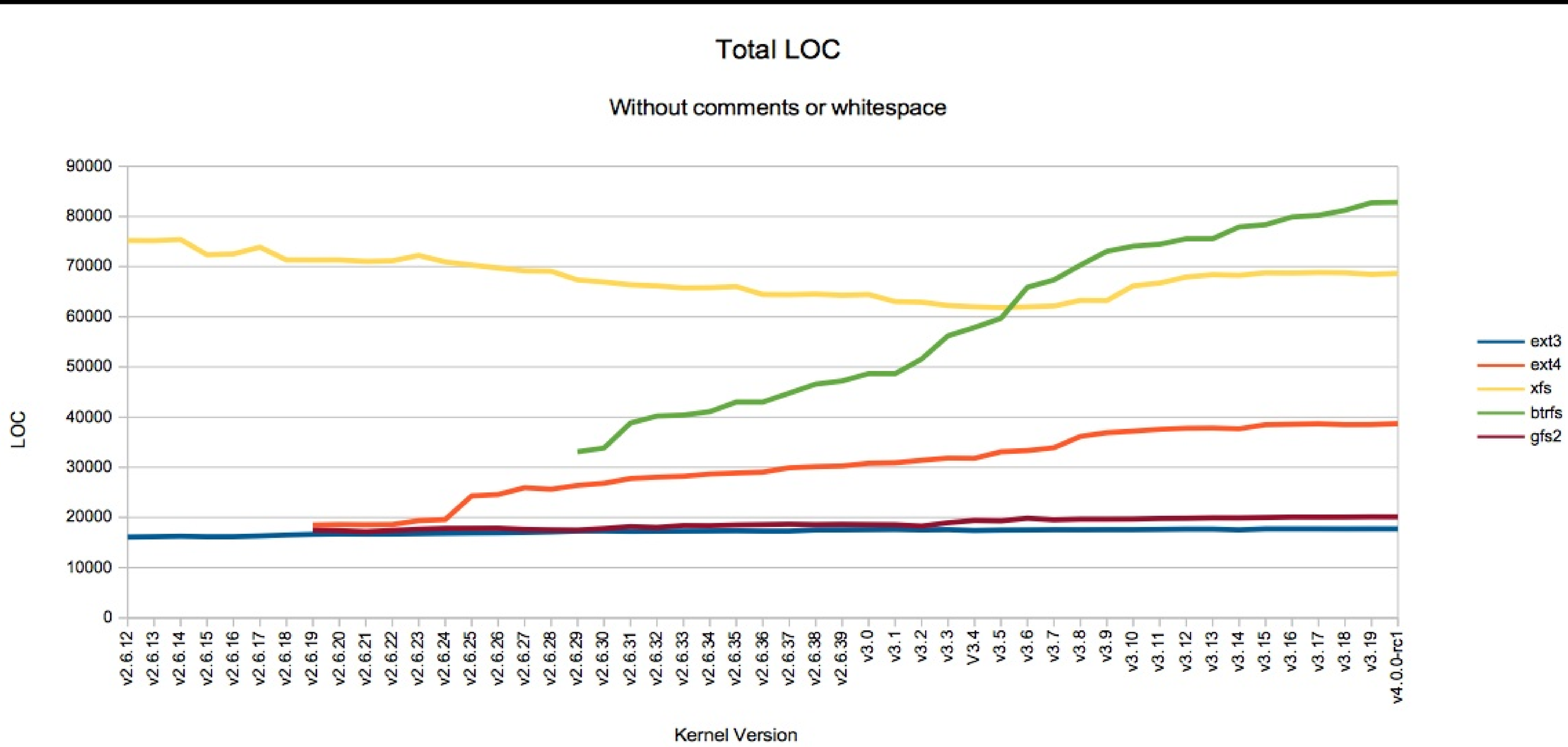
Back to the Features 4

- generic btree consolidation (2009)
- tracing hooked to new generic kernel tracing infrastructure (2009)
- RHEL 5.4 adds support for XFS (2009) (Finally!)
- delayed logging (2010)
- 32-bit project quota support (2010)
- dynamic speculative allocation (2011)
- strict metadata verification at IO time (2012)
- Version 5 superblocks created for CRC support (2012)
- concurrent user, group and project quota support (2013)
- dirent type support (2013)
- free inode btree index (2014)
- PNFS server block layer support (2015)

Back to the Present



Back to the Present



Top Developers

1096 Christoph Hellwig (2002-present)
1061 Dave Chinner (2005-present)
957 Nathan Scott (1999-2006)
956 Doug Doucette (Second XFS commit - 1999)
950 Steve Lord (1997-2003)
661 Adam Sweeney (First XFS commit - 1996)
431 Eric Sandeen (2000-present)
221 Supriya Wickrematillake (1994-1997)
208 Michael Nishimoto (1993-1995)
175 Ray Chen (1994-1998)

These Days

- sparse inode chunk allocation
- generic/XFS quota API unification
- reverse mapping btrees
- reflink support
- defragmentation improvements
- ranged inode bulkstat
- extent manipulation offloads via fallocate
- filesystem repair improvements
- user/kernel code sharing simplifications
- DAX support

These Days

- Plotting the next five years
- Long term lines of development
- Similar to 2008 documents on xfs.org
- Emerging technologies could be game changers
- Known technologies require diametrically opposed optimisations

Shiny Things?

- SMR drives
  - require fundamental changes
    - high latency, low throughput media
  - SMR in RAID5/6 is a long way off
  - Is XFS really the filesystem for SMR devices?
  - What is a good-enough solution to get us through to smr-native filesystems are written?

Shiny Things?

- Persistent memory
  - already available as NVDIMMS
  - Byte addressable, low latency, high bandwidth
  - Near term capacity measured in TB
  - CPU cache coherent
  - DAX (Direct Access) via mmap() gets applications 99% performance of a native filesystem solution
  - Do we need to do anything more in XFS?

Shiny Things

- Block device integration
    - thin provisioning capacity awareness
    - block clone/copy/compress offloads
    - block unsharing for exclusive access
    - snapshot awareness and control
    - preallocation/reservation of blocks
    - much more....

Shiny Things

- Improving reliability
  - reconnection of orphaned objects
  - proactive corruption detection
  - partial filesystem isolation
  - online repair of simple corruptions
  - online repair of AG free space corruptions

Interesting Times

- Spinning rust progression:
    - mid 1990s - 8GB, 7ms drives
    - mid 2010s - 8TB, 15ms drives
    - mid 2030s - 8PB, 30ms drives?

If this occurs, what does it mean?

Interesting Times

- Solid State progression:
    - 2005 - slow, unreliable 30GB drives @ $10/GB
    - 2015 - 512TB in 3U @ <$1/GB
      (Sandisk InfiniFlash, 780kiops, 7GB/s)
    - 2025 - 8EB in 3U @ < $0.1/GB ???

- $1/GB = cost of enterprise SAS disk storage
- Large scale solid state storage is:
    - cost competitive
    - order of magnitude denser
    - order of magnitude lower power
    - order of magnitudes higher performance

Interesting Times

- Persistent memory will be
  - denser than solid state storage
  - faster than solid state storage
  - cost competitive with solid state storage
- At least five years until pervasive deployments

- Memristors are the wildcard in this field.....

Interesting Times

- Where does this projection leave spinning rust based storage?

- What does it mean for XFS development?

....and There Again?

- solid state storage will push capacity, scalability and performance much faster than SMR technologies

- SMR is a low density bulk storage technology, not a high performance storage medium

- 64bit address space exhausted in 10-15 years.
  - Both for filesystems *and CPUs*

- Wait, What?

....and There Again!

- By 2025, XFS will be hitting against:
  - capacity and addressibility limits
  - architectural performance limits
  - unnecessary complexity limits
  - an architecture unsuited to emerging storage technologies

....and There Again!

- Places hard limits on XFS development life
  - succession planning must start within 5-10 years
  - implies this 5-7 year dev cycle will be the last

- Large scale rework for SMR makes little sense
  - large, lots and fast is being driven entirely by solid state storage.

- In 20 years time, XFS will be a legacy filesystem

Thank You for Listening!

- Questions welcome