Local file systems update

Red Hat
Lukáš Czerner
February 23, 2013

Agenda

1. Linux file systems overview
2. Challenges we're facing today
3. Xfs
4. Ext4
5. Btrfs
6. Questions ?

Part I

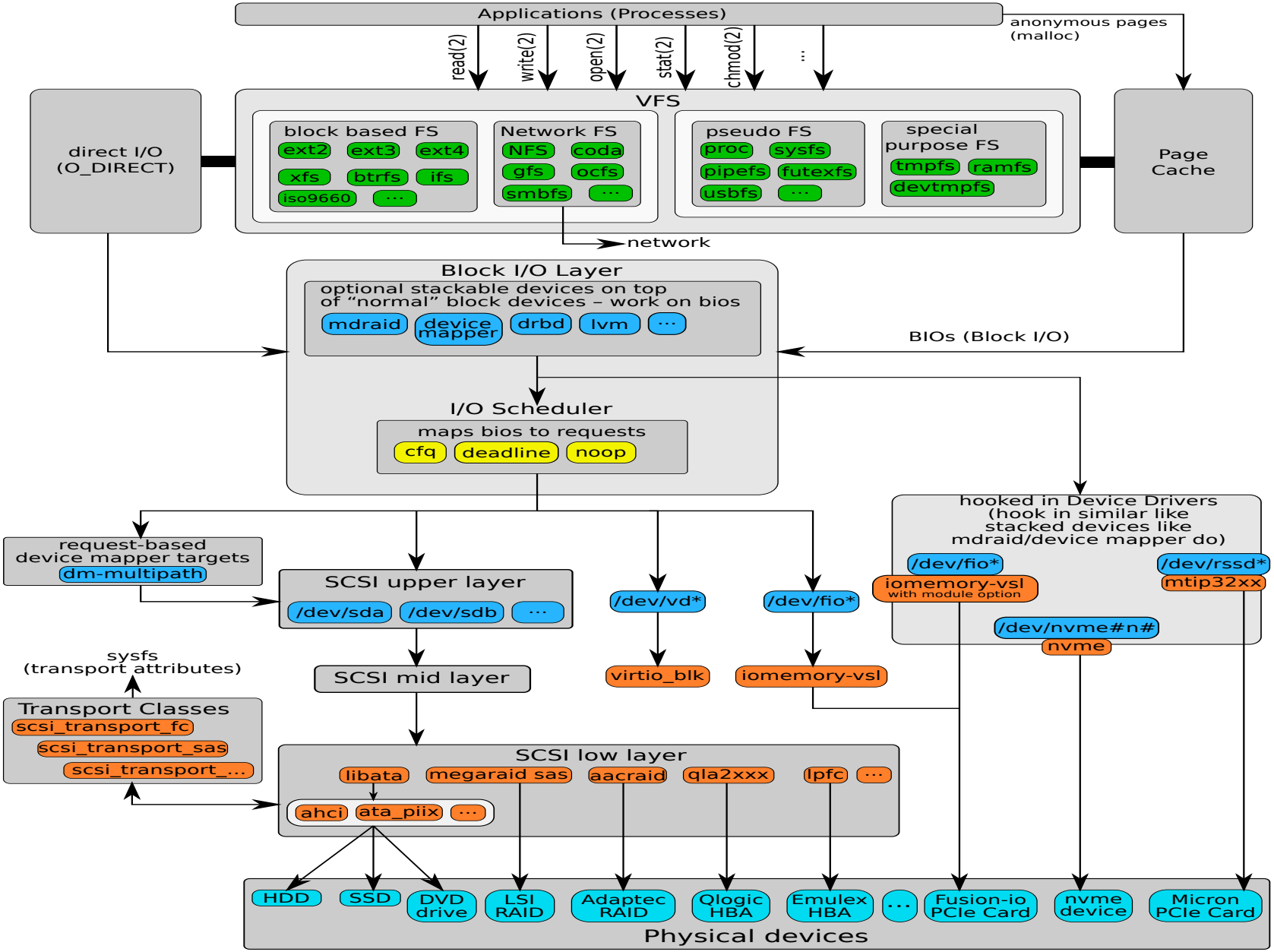
Linux kernel file systems overview

File systems in Linux kernel

- **Linux kernel has a number of file systems**
  - Cluster, network, local
  - Special purpose file systems
  - Virtual file systems
- **Close interaction with other Linux kernel subsystems**
  - Memory Management
  - Block layer
  - VFS - virtual file system switch
- **Optional stackable device drivers**
  - device mapper
  - mdraid

The Linux I/O Stack Diagram

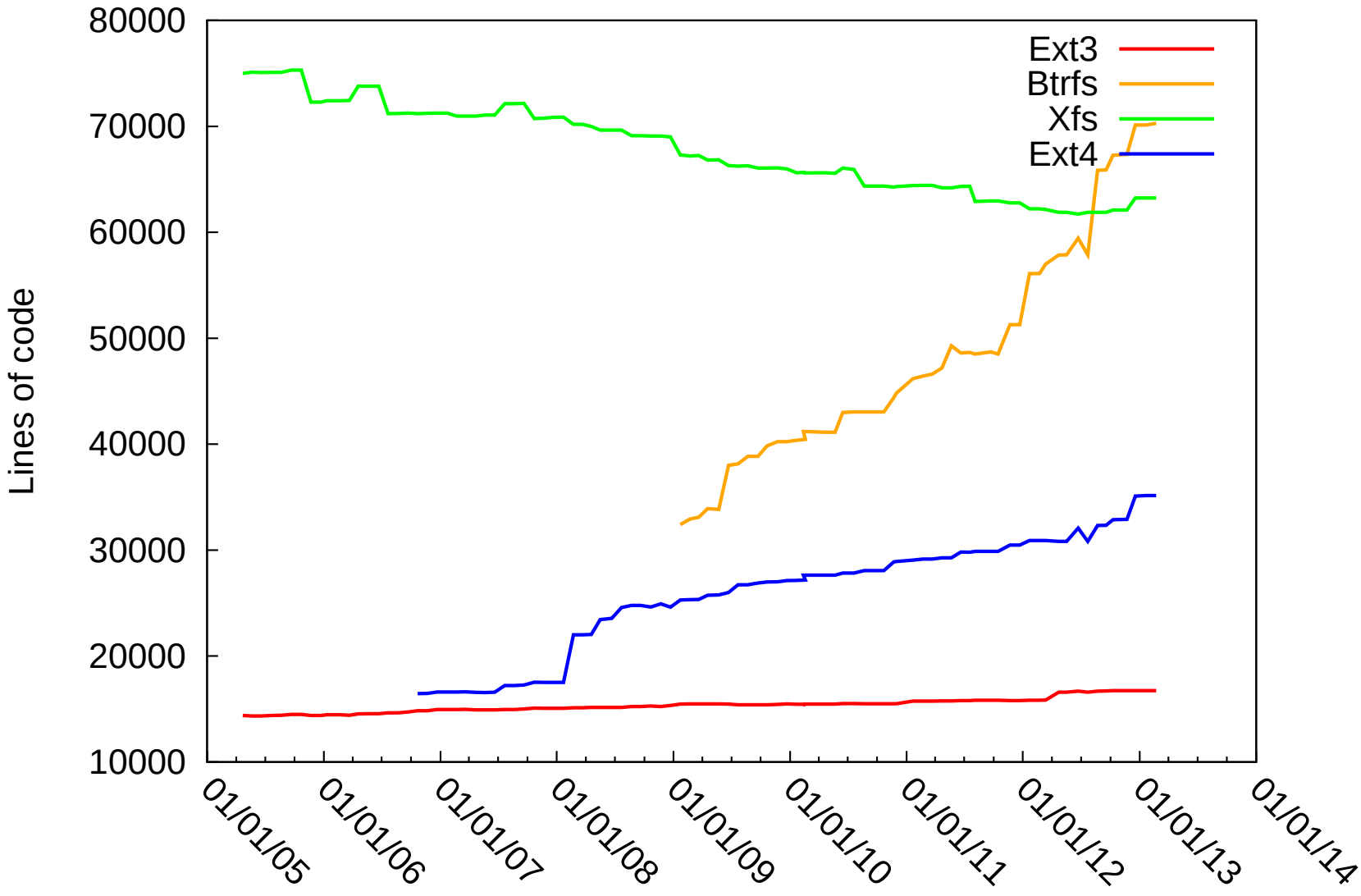
version 0.1, 2012-03-06
outlines the Linux I/O stack as of Kernel version 3.3



Most active local file systems

File system	Commits	Developers	Active developers
Ext4	648	112	13
Ext3	105	43	2
Xfs	650	61	8
Btrfs	1302	114	21

Number of lines of code



redhat.

Part II

Challenges we're facing today

Scalability

- **Common hardware storage capacity increases**
  - You can buy single 4TB drive for a reasonable price
  - Bigger file system and file size
- **Common hardware computing power and parallelism increases**
  - More processes/threads accessing the file system
  - Locking issues
- **I/O stack designed for high latency low IOPS**
  - Problems solved in networking subsystem

Reliability

- Scalability and Reliability are closely coupled problems
- **Being able to fix your file system**
  - In **reasonable time**
  - With **reasonable memory** requirements
- **Detect errors before your application does**
  - Metadata checksumming
  - Metadata should be self describing
  - Online file system scrub

New types of storage

- **Non-volatile memory**
  - Wear levelling more-or-less solved in firmware
  - Block layer has it's IOPS limitations
  - We can expect bigger erase blocks
- **Thinly provisioned storage**
  - Lying to users to get more from expensive storage
  - Filesystems can throw away most of it's locality optimization
  - Cut down performance
  - Device mapper dm-thinp target
- **Hierarchical storage**
  - Hide inexpensive slow storage behind expensive fast storage
  - Performance depends on working set size
  - Improve performance
  - Device mapper dm-cache target, bcache

redhat.

Maintainability issues

- **More file systems with different use cases**
  - Multiple set of incompatible user space applications
  - Different set of features and defaults
  - Each file system have different management requirements
- **Requirements from different types of storage**
  - SSD
  - Thin provisioning
  - Bigger sector sizes
- **Deeper storage technology stack**
  - mdraid
  - device mapper
  - multipath
- Having a centralized management tool is **incredibly useful**
- Having a central source of information is a **must**
- System Storage Manager http://storagemanager.sf.net

Part III

What's new in xfs

Scalability improvements

- **Delayed logging**
  - Impressive improvements in metadata modification performance
  - Single threaded workload still slower then ext4, but not much
  - With more threads scales **much** better than ext4
  - On-disk format change
- **XFS scales well up to hundreds of terabytes**
  - Allocation scalability
  - Free space indexing
- Locking optimization
- Pretty much the **best** choice for *beefy* configurations with lots of storage

Reliability improvements

- **Metadata checksumming**
  - CRC to detect errors
  - Metadata verification as it is written to or read from disk
  - On-disk format change
- **Future work**
  - Reverse mapping allocation tree
  - Online transparent error correction
  - Online metadata scrub

Part IV

What's new in ext4

Scalability improvements

- **Based on very old architecture**
  - Free space tracked in bitmaps on disk
  - Static metadata positions
  - Limited size of allocation groups
  - Limited file size limit (16TB)
  - Advantages are resilient on-disk format and backwards and forward compatibility
- **Some improvements with bigalloc feature**
  - Group number of blocs into clusters
  - Cluster is now the smallest allocation unit
  - Trade-off between performance and space utilization efficiency
- **Extent status tree for tracking delayed extents**
  - No longer need to scan page cache to find delalloc blocks
- Scalability is very much limited by design, on-disk format and backwards compatibility

Reliability improvements

- **Better memory utilization of user space tools**
  - No longer stores whole bitmaps - converted to extents
  - Biggest advantage for e2fsck
- **Faster file system creation**
  - Inode table initialization postponed to kernel
  - Huge time saver when creating bigger file systems
- **Metadata checksumming**
  - CRC to detect errors
  - Not enabled by default

Part V

What's new in btrfs

Getting stabilized

- **Performance improvements is not where the focus is right now**
  - Design specific performance problems
  - Optimization needed in future
- Still under heavy development
- Not all features are yet ready or even implemented
- File system stabilization takes a long time

Reliability in btrfs

- Userspace tools not in very good shape
  - Fsck utility still not fully finished
- Neither kernel nor userspace handles errors gracefully
- Very good design to build on
  - Metadata and data checksumming
  - Back reference
  - Online filesystem scrub

Resources

- **Linux Weekly News** http://lwn.net
- **Kernel mailing lists** http://vger.kernel.org
  - linux-fsdevel
  - linux-ext4
  - linux-btrfs
  - linux-xfs
- **Linux Kernel code** http://kernel.org
- **Linux IO stack diagram**
  - http://www.thomas-krenn.com/en/oss/linuxiostack-diagram.html

The end.

Thanks for listening.