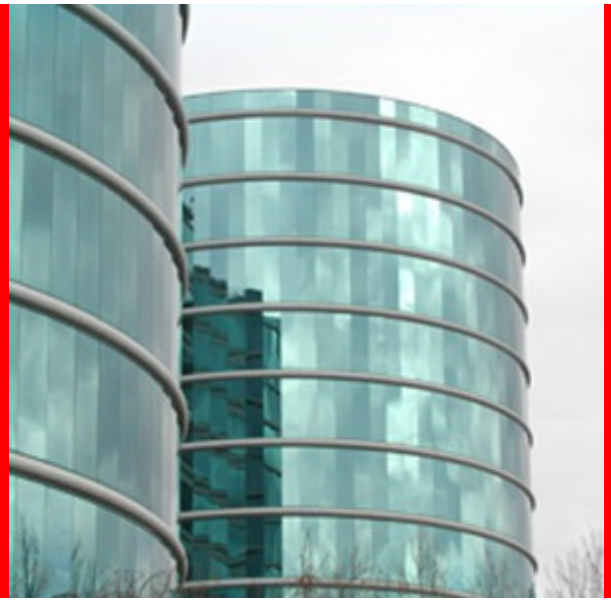


ORACLE®

Btrfs Filesystem

Chris Mason

Btrfs Goals

- General purpose filesystem that scales to very large storage
- Feature focused, providing features other Linux filesystems cannot
- Administration focused, easy to run and very fault tolerant
- Perform well in a variety of workloads

Btrfs Features

- Extent based file storage
- Copy on write metadata and data
- Space efficient packing of small files
- Optional transparent compression (zlib)
- Integrity checksumming for data and metadata
- Writable snapshots
- Online resize, defragmentation, device management
- Multiple device support
- Offline conversion from Ext3 and Ext4
- Specialized log for fast fsync and O_SYNC writes

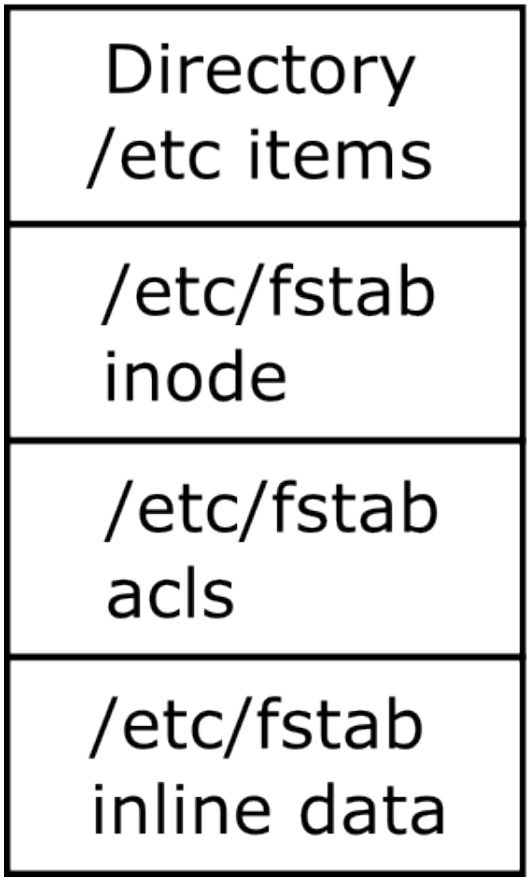
Btrfs Status

- Included in 2.6.29
- Generally usable in many workloads
- Generally stable
- No disk format changes planned
- Development team includes many companies and individuals
- Proper ENOSPC handling
- AIO/DIO support
- Snapshot assisted upgrades

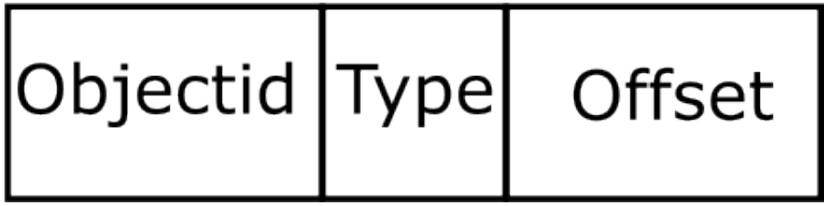
Btrfs Btree

- Generic key/value pair storage
- The same btree core used for all metadata
- Protected by copy on write for crash safety
- Transaction id stored in block headers and pointers
  - Allows efficient searches for recent changes
- Metadata from different files and directories is mixed together in a block
- All metadata is addressed by a key and searched for in the btree
- Key order keeps related items close together in the btree

Btree Block



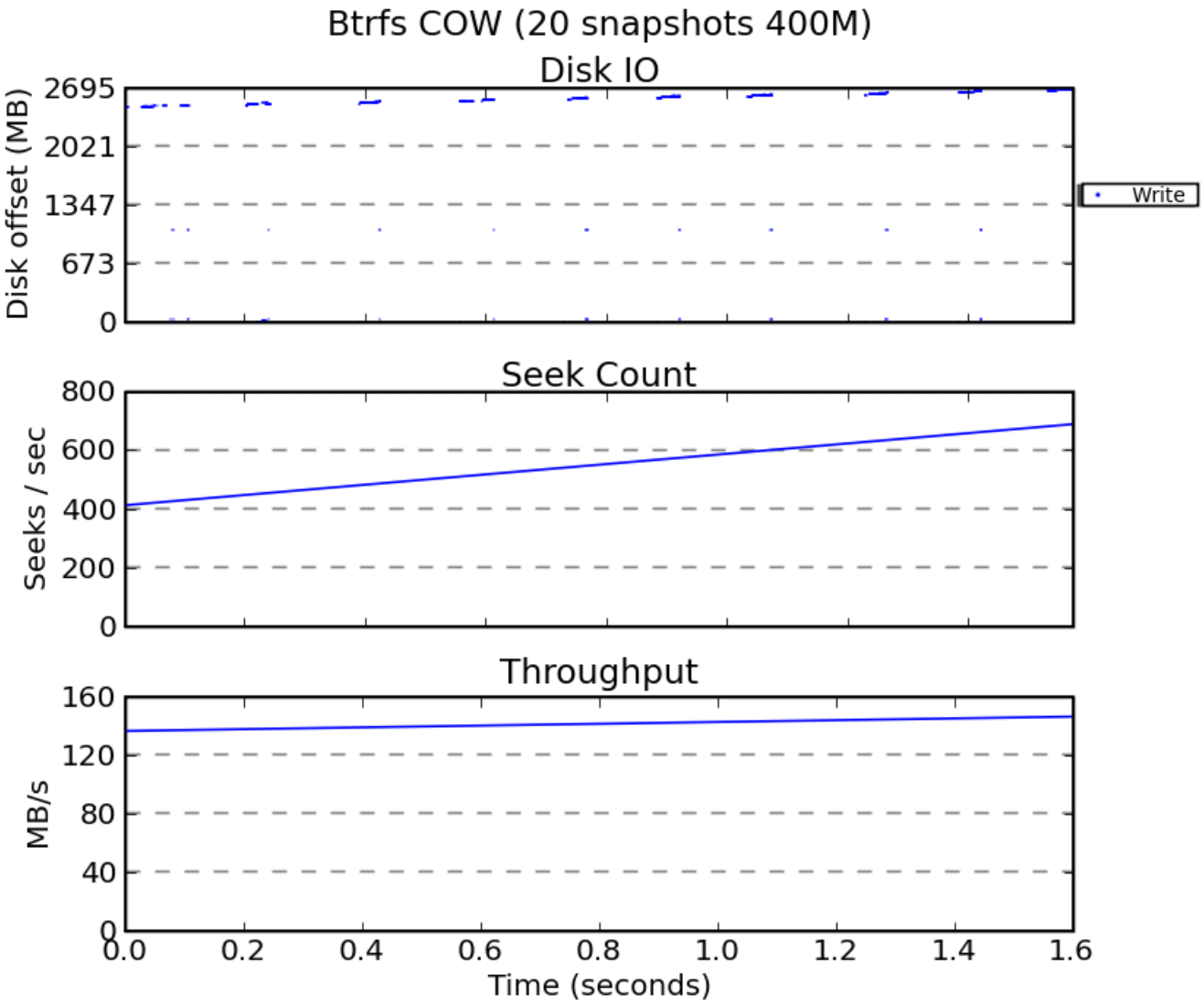
Item Key

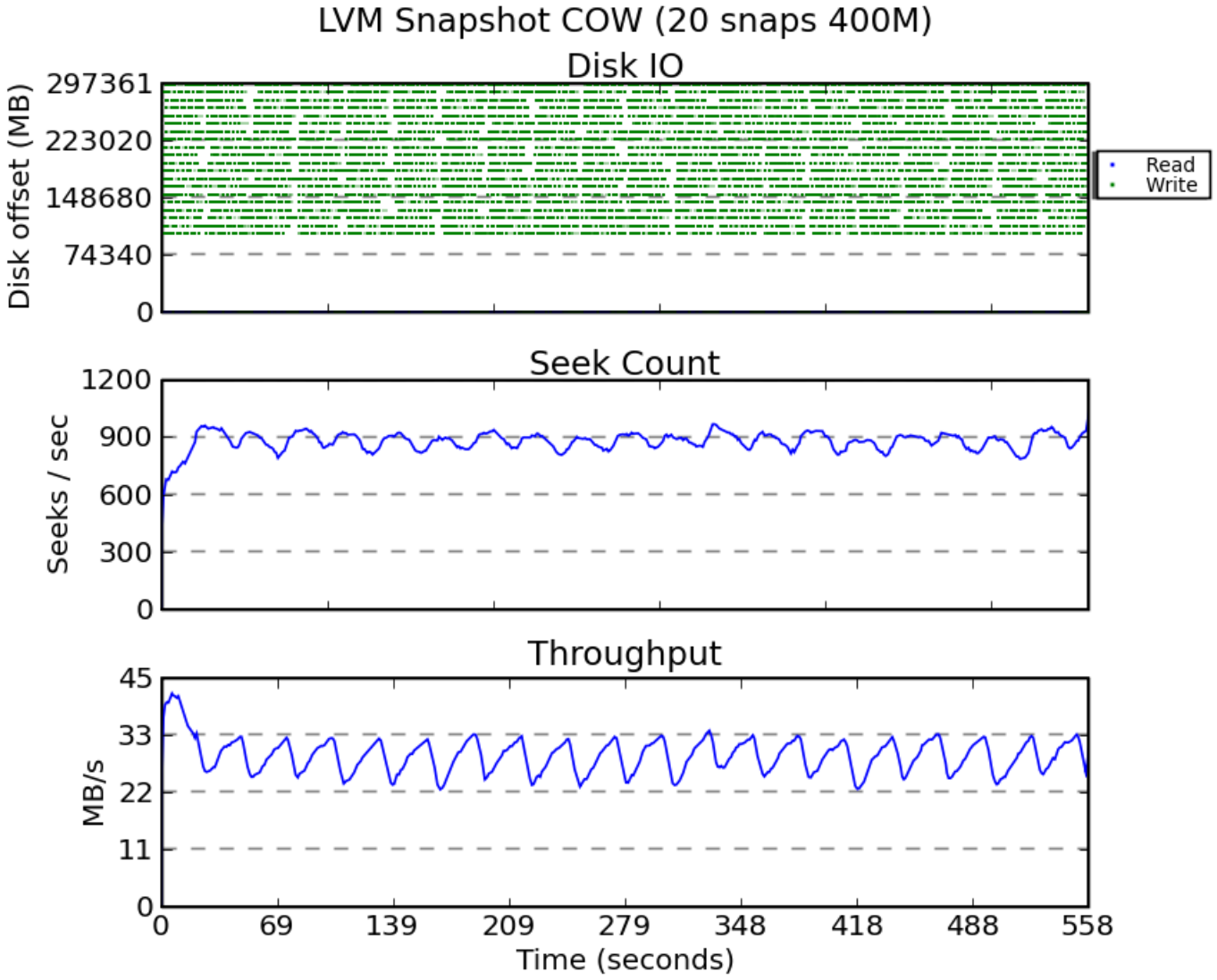


Key Types:
Directory Item
Inode Item
xattr
File extent pointer
Inode back reference
...

COW Comparison (next two graphs)

- Btrfs
  - Create 20 snapshots of a 400MB file
  - Overwrite the file
  - 400MB written to a new location on disk
  - Total time: 1.6s
- LVM
  - Create 20 snapshots of a LVM logical volume
  - Overwrite 400MB of the original
  - 400MB copied and written to exception table for each snapshot
  - Total time: 558s


Btrfs COW (20 snapshots 400M)
Disk IO
Disk offset (MB)
2695
2021
1347
673
0
Write
Seek Count
Seeks / sec
800
600
400
200
0
Throughput
MB/s
160
120
80
40
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Time (seconds)


LVM Snapshot COW (20 snaps 400M)
Disk IO
Disk offset (MB)
297361
223020
148680
74340
0
Read
Write
Seek Count
Seeks / sec
1200
900
600
300
0
Throughput
MB/s
45
33
22
11
0
0
69
139
209
279
348
418
488
558
Time (seconds)

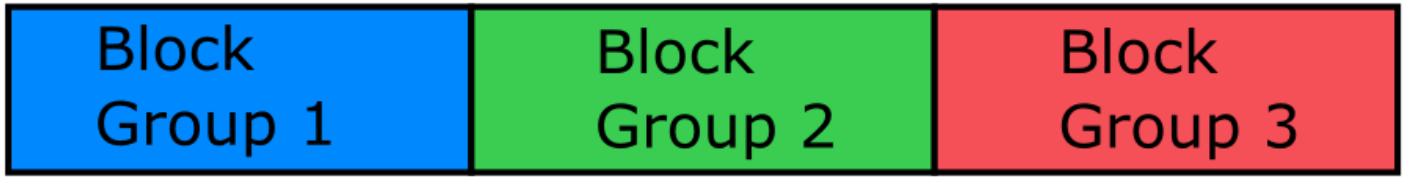
Snapshots and Subvolumes

- Subvolume is the unit of snapshotting
  - Individual files may be cloned without a full snapshot
  - Cloning support now in cp --relink
- Subvolumes may be created anywhere in the directory tree
- Reference counts and back references track every extent and btree block
- Snapshots can be written and snapshotted again
- Snapshots not suitable for continuous data protection

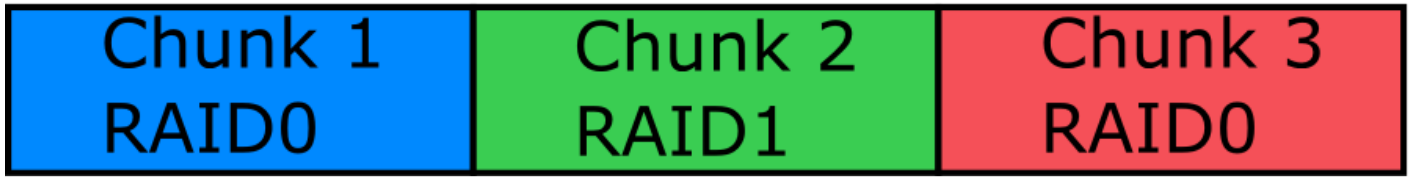
Multi-device Support

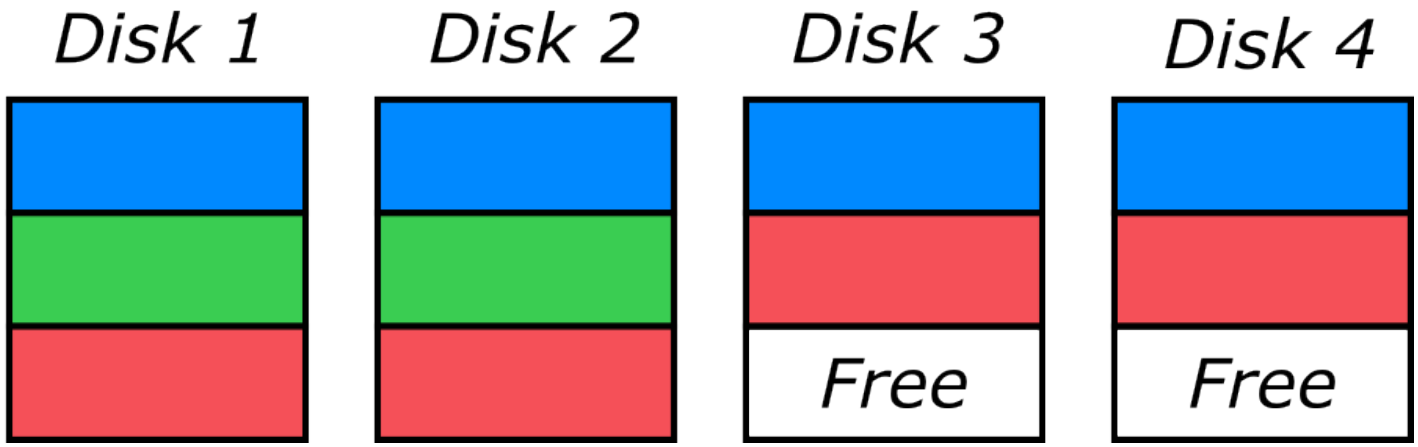
- Devices are added into a pool of available storage
- New logical address space is allocated with a specific RAID configuration and data storage flags
  - System (used by the volume management code)
  - Metadata
  - Data
  - Raid0, raid1, raid10, single-spindle-dup
  - RAID5,6 are coming
- Space is allocated from the storage pool in large chunks (1GB or more)
- Devices can be mixed in size and speed

Extent Allocation Tree



Chunk Tree: Logical -> Physical Map





Demo

- Quickly finding updated files and blocks
- File cloning
- Snapshot rollback

Synchronous Operations

- COW transaction subsystem is slow for frequent commits
  - Forces recow of many blocks
  - Forces significant amounts of IO writing out extent allocation metadata
- Write ahead log added for synchronous operations on files or directories
- File or directory items are copied into a dedicated tree
  - File back refs allow us to log file names without the directory
  - One log btree per subvolume

Synchronous Operations

- The log tree uses the same COW btree code as the rest of the FS
- The log tree uses the same writeback code as the rest of the FS, and uses the metadata raid policy.
- Commits of the log tree are separate from commits in the main transaction code.
  - fsync(file) only writes metadata for that one file
  - fsync(file) does not trigger writeback of any other data blocks

SSD Optimizations

- Mount -o ssd
  - Places new extents into areas that are mostly free
  - Combines new writes from many different files without trying to prevent fragmentation
  - Effective on high end SSD
- Mount -o ssd_spread
  - Places new extents into areas that are completely free
  - More likely to overwrite an entire erasure block in the SSD
- Trim
  - Mount -o discard (2.6.32)
  - Extents are trimmed in bulk at transaction commit time
  - Some hardware trims very slowly today

Conclusions

- Btrfs is ready for broader testing
- Many projects available for new contributors
- http://btrfs.wiki.kernel.org/
- chris.mason@oracle.com