The Btrfs Filesystem

Chris Mason

ORACLE®

The Btrfs Filesystem

- Jointly developed by a number of companies
  - Oracle, Redhat, Fujitsu, Intel, SUSE, many others
- All data and metadata is written via copy-on-write
- CRCs maintained for all metadata and data
- Efficient writable snapshots
- Multi-device support
- Online resize and defrag
- Transparent compression
- Efficient storage for small files
- SSD optimizations and trim support

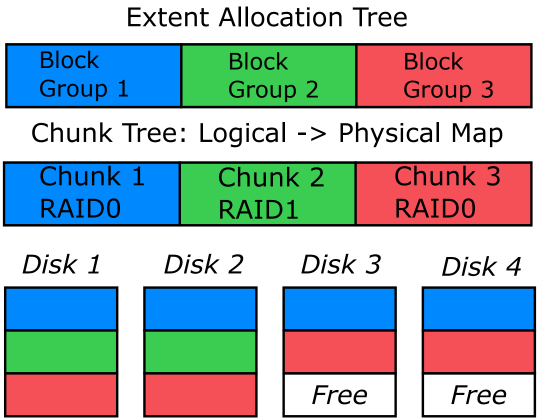
Btrfs Progress

- Extensive performance and stability fixes
- New and improving repair tool
- Background scrubbing
- Automatic repair of corrupt blocks
- RAID restriping
- Configurable metadata block sizes
- Improved IO error handling infrastructure

The Btrfs Structures

- Btrfs only stores one kind of metadata block
- Btree blocks store key/value pairs
- Metadata structures use specific keys to store related items close together on disk
- Logical address layer translates data and metadata blocks to physical areas of the storage
- Metadata for different files and directories can all be stored in the same btree block

Storage Allocation



- Storage allocated in chunks to create specific raid levels
- Data and metadata can have different raid levels

New Storage Technologies – Flash

- Flash lifetime is limited by the number of write cycles to a cell
- Small writes require internal read/modify/write cycles in the device
  - A 1MB write might count as small
  - Each small write may be amplified into multiple larger writes
- Flash lifetime can be increased if the FS works together with the device

Hints For the Storage

- Discard and Trim allow the device to ignore blocks the FS isn't using
- Devices may be tiered internally
  - Frequently modified or deleted blocks stay on faster cells
  - Long lived blocks moved to less expensive storage
- New APIs and standards will allow the FS to give hints to the device
- Large arrays and high end flash can use the hints to improve performance
- Low end flash can use hints to increase cell lifetime
- Btrfs block group layout separates shorter lived metadata from data

Discard/Trim

- Trim and discard notify storage when we are done with a block
- Btrfs supports both real-time trim and batched trim
- Real-time trims blocks as they are freed
- Batched trims all free space via an ioctl
- Newer kernels will have less penalty for online discard

Btrfs Restriper

- Newly introduced in 3.3
- Advanced control over block groups and storage
- Balance data across drives to select new RAID levels
- Balance filtering by usage for thin provisioning
- Ex: btrfs filesystem balance start -dconvert=raid1

When Bad Things Happen to Good Data

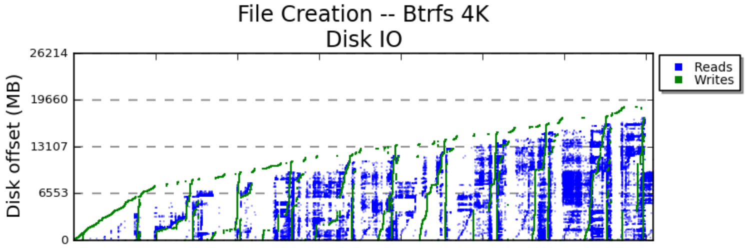
- Barrier bugs in Btrfs lead to most of the corruptions seen with kernels before v3.2
- Filesystem repair tool in btrfs-progs git
  - Repairs extent allocation tree corruptions in place
  - More repair modes in progress
- Filesystem recovery tool from Josef Bacik
  - Risk free – copies data out of the corrupt FS
- Tree root history log to recover from many hardware errors
  - Jumps back to older versions of the tree roots

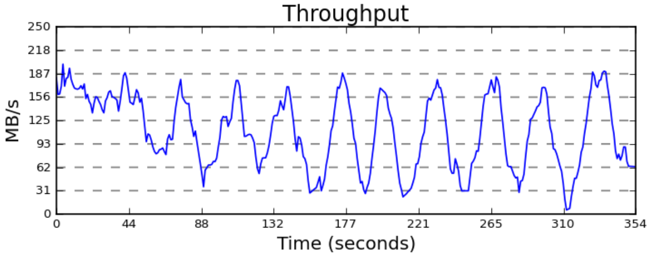
Larger Metadata Blocks

- Btrfs btree uses key ordering to group related items into the same metadata block
- COW tends to fragment the btree over time
- Larger blocksizes provide very inexpensive btree defragmentation
- Larger blocksizes reduce extent allocation overhead (fewer extents)
- Larger blocksizes allow metadata on raid5/6

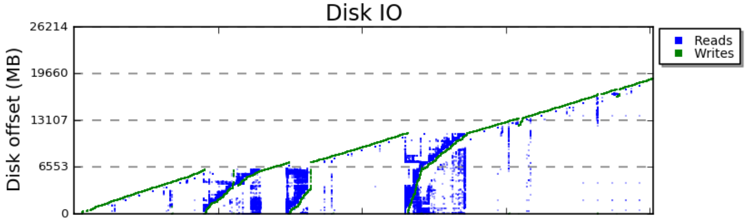
Metadata Blocksizes and Writeback

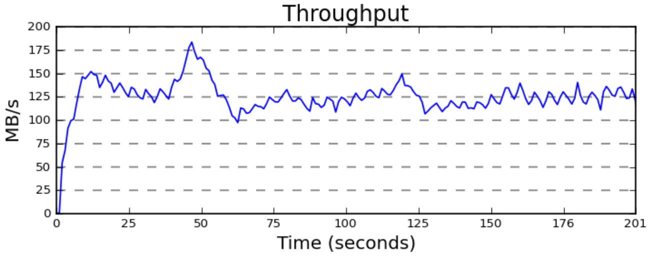
- Btrfs COW tends to allocate and free pages often
- The Linux VM was keeping our stale pages on the LRU too long
- In metadata heavy workloads Btrfs did many more reads than it should have
- After fixing metadata writeback and enabling larger blocks:
  - Create 32 million empty files
  - Btrfs – 170K files/sec (16KB metadata)
  - Btrfs – 150K files/sec (4KB metadata after LRU fixes)
  - XFS – 115K files/sec
  - Ext4 – 110K files/sec (256MB log)

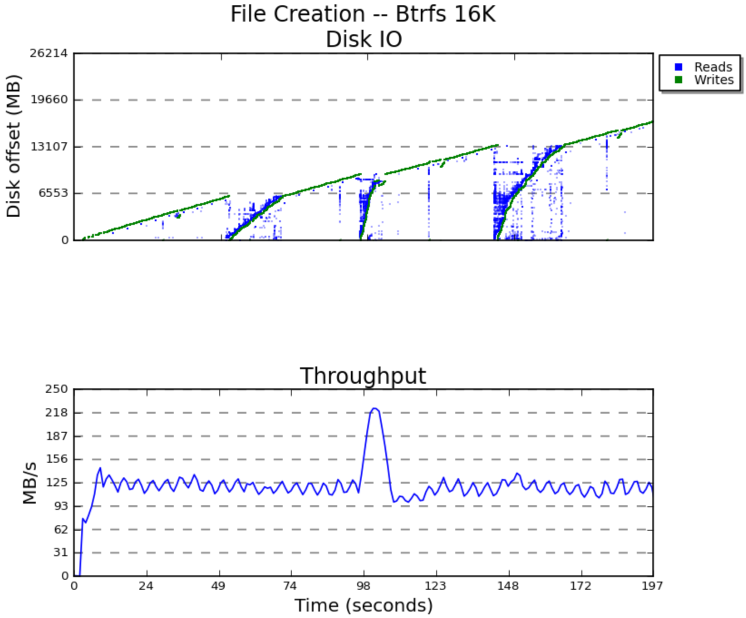

File Creation -- Btrfs 4K
Disk IO
Disk offset (MB)
26214
19660
13107
6553
0
Reads
Writes


Throughput
MB/s
250
218
187
156
125
93
62
31
0
0
44
88
132
177
221
265
310
354
Time (seconds)

File Creation -- Btrfs 4K (v3.4)







File Creation -- Btrfs 16K
Disk IO
Disk offset (MB)
26214
19660
13107
6553
0
Reads
Writes
Throughput
MB/s
250
218
187
156
125
93
62
31
0
0
24
49
74
98
123
148
172
197
Time (seconds)

IO Animations

- Ext4 is bottlenecked on reading the inode tables
- Both XFS and Btrfs are CPU bound
- XFS is walking forward through a series of distinct disk areas
- Both XFS and Ext4 show heavy log activity
- Btrfs is doing sequential writes and some random reads

Scrub

- Btrfs CRCs allow us to verify data stored on disk
- CRC errors can be corrected by reading a good copy of the block from another drive
- Scrubbing code scans the allocated data and metadata blocks
- Any CRC errors are fixed during the scan if a second copy exists
- Will be extended to track and offline bad devices

Seed Devices

- A readonly device can be used as a filesystem seed
- Read/write devices can be added to store modifications
- Changes to the writable devices are persistent across reboots
- The readonly device can be removed at any time
- Multiple read/write filesystems can be built from the same seed

Embedded Systems

- Btrfs is fairly friendly to small machines
- Very little memory is pinned by the filesystem
- Btrfs works very well overall on low end flash

Thank You!

- Chris Mason <chris.mason@oracle.com>
- http://btrfs.wiki.kernel.org