BTRFS

Linux’s Filesystem of the Future
OVERVIEW

- B-tree copy-on-write filesystem
- Started by Chris Mason of Oracle
- Expected features comparable to ZFS
  - Writable snapshots
  - Checksumming
  - Volume management
- Experimental - still feature incomplete!
No truly modern Linux filesystem
  - ext2/3/4 based on 30+ year old format

Started by Chris Manson of Oracle.
  - Worked on Reiserfs

Design based on COW-friendly b-trees
  - COW-friendly b-trees invented by Ohad Rodeh
  - All objects in btrfs use this structure
FEATURES

- **Subvolumes**
  - Each filesystem can have independent child filesystems – similar to ZFS

- **Snapshots**
  - Snapshots are clones of subvolumes
  - Use same data-structures

- **Multi-device filesystems – RAID 0, RAID 1**
  - RAID 5/6 planned
MORE FEATURES

- Checksums
- Online Balancing
- Online defragmentation
- Volume Growth/Shrinking
- Block Device Add/Removal
- SSD Optimizations
- Compression
- In-place conversion from ext3/4
## Feature Comparison – ZFS

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZFS</th>
<th>BTRFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-filesystems</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Snapshots</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAID 0/1</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAID 5/6</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Quotas</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Block Device filesystems</td>
<td>Yes</td>
<td>NO</td>
</tr>
<tr>
<td>Checksums</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Compression</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Device Add/Remove</td>
<td>NO</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Three types of on-disk structures:
- Block Headers, Keys, Items
- Nodes in tree are only keys and block headers
  - Keys point to items
  - Block headers point to nodes on disk
- Leaves are items
  - keys and data
OLD ON-DISK FORMAT
BTRFS DISK FORMAT
struct btrfs_header {
    u8 csum[32];
    u8 fsid[16];
    __le64 blocknr;
    __le64 flags;
    u8 chunk_tree_uid[16];
    __le64 generation;
    __le64 owner;
    __le32 nritems;
    u8 level;
}

struct btrfs_disk_key {
    __le64 objectid;
    u8 type;
    __le64 offset;
}

struct btrfs_item {
    struct btrfs_disk_key key;
    __le32 offset;
    __le32 size;
}

- **Item**
  - Each item has a key

- **Key**
  - Objectid – each object in btrfs has unique ID – like inode numbers.
  - Type – inode, directory entry, extent, file data

- Each extent in btree also contains back-references to all extents that reference it
ADVANTAGES

- Snapshots
  - Reference count + checksum
  - Snapshots are writable – Copy-on-Write

- Multi-device
  - Back-references allow easy migration of data
  - Easy to remove devices from filesystem

- Checking
  - Back-refs allow incremental checking of FS
BASIC COMMANDS

- mkfs.btrfs
- btrfsctl – old control command
- btrfs – new control command
- btrfsck – offline filesystem check
- btrfs-convert
- btrfs-vol – old device management
- btrfs-bcp – Copy-on-write files
root@host # mkfs.btrfs /dev/sda
root@host # mkfs.btrfs -m raid0 -d raid0 /dev/sdb /dev/sdc

root@host # btrfs subvolume create <path>
root@host # btrfs subvolume snapshot <source> [dest/]<name>
root@host # btrfs subvolume delete <path>

root@host # btrfs filesystem resize +/-<newsize> <filesystem>