Btrfs - The Next Generation Filesystem on Linux

Neependra Khare

STEC India

November 5, 2011

Agenda

- Existing Filesystems On Linux
- Challenges with existing file-systems
- Btrfs
- Features of Btrfs
- Rodeh's Research
- Btrfs Design
- Demo

Existing Filesystems On Linux

- Ext2,3,4
- XFS
- ReiserFS
- ...

What is Filesystem?

What is Filesystem?

- How do we create?

What is Filesystem?

- How do we create?
- How do we expand?

What is Filesystem?

- How do we create?
- How do we expand?
- How do we shrink?

What is Filesystem?

- How do we create?
- How do we expand?
- How do we shrink?

What is Volume Management?

Challenges with existing file-systems

- No defense against silent Data corruption
- Recovery
- Hard to manage
  - Partitions
  - Volumes
  - Grow or Shrink
- Performance Issues
- File-system and volume managers are separate products.

Beginning of BtrFS

- In 2007, Chris Mason, a former ReiserFS developer at SUSE who had recently moved to Oracle, was given the opportunity to design and create a next-generation Linux file system.
- After seeing Rodehs research presentation at USENIX FAST 07, Mason had the idea to create everything in the file systeminodes, file data, directory entries, bitmaps, the works[as] an item in a copy-on-write b-tree

Btrfs Features

- checksumming of data and metadata (CRC)
- built-in device/space management (spanned across devices) (so multiple device support no need for lvm)
- support for raid0, raid1, raid10 and single at this point (with raid5/6 in the works)
- ability to independently span metadata and data across these devices
- copy on write(COW) for both data and metadata
- writable snapshots
- create filesystem in existing btrfs pool without need to worry about device management
- online resize of filesystem (both grow and shrink)
- transparent compression, you can even specify for each file, or across all (lzo or zlib)
- ability to defrag files and/or directories

Btrfs Features

- balance command to balance filesystem chunks in a path across multiple devices if needed
- online add and remove devices to/from filesystems
- support for trim and SSD optimizations
- in place conversion from ext3/4 to btrfs
- file-based or object based cloning support with reflink (per file clone)
- file allocation is extent based with B-tree directory structures
- for the little details :
- Max file size 16 EiB
- Max number of files 2$\hat{6}$4
- Max volume size 16 EiB

Rodehs Research

- Remove links between leaf nodes in a b+-tree so that the entire tree does not have to be copied during copy-on-write.
- Provide a way to perform COW
- Support a large number of clones efficiently

Rodehs Research : COW

Enable COW by shadowing:

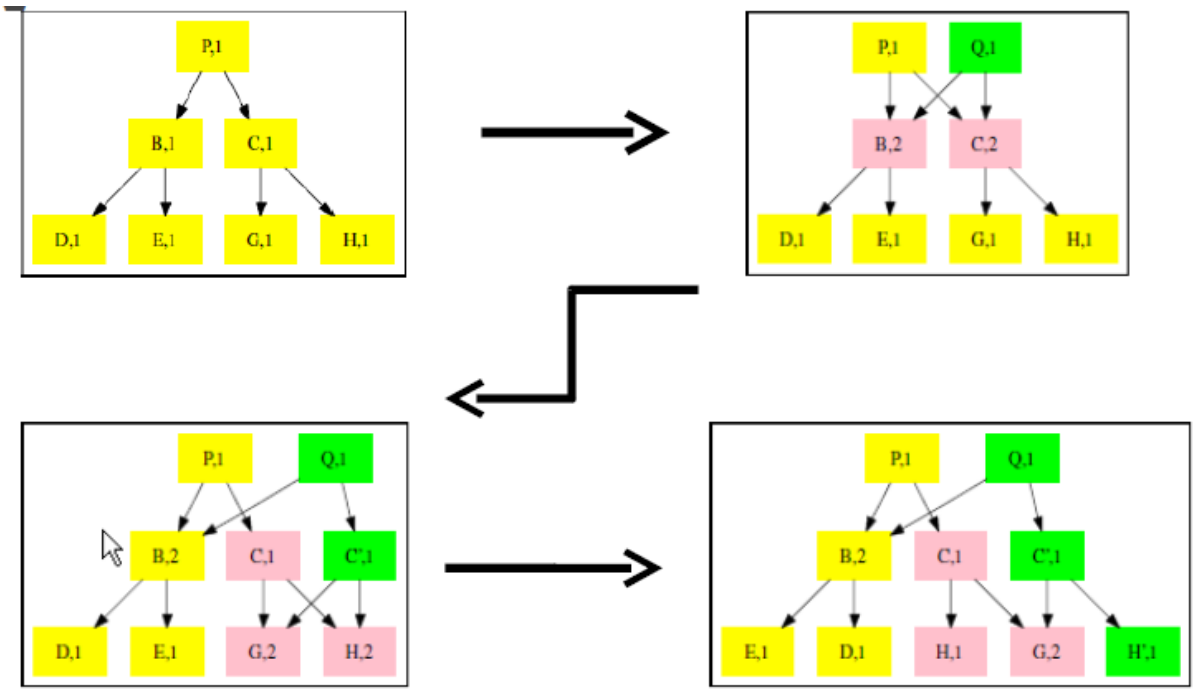
- Each change to a page is performed on a copy of the page in memory and a log of each operation is kept on disk
- Occasionally perform a checkpoint and write all changed pages to disk in one batch
- If a crash occurs before a checkpoint is performed, the operations stored in the long are performed to update the data

Rodehs Research : Clones

To clone a b-tree means to create a writable copy of it that allows all operations: lookup, insert, remove, and delete. Desirable properties:

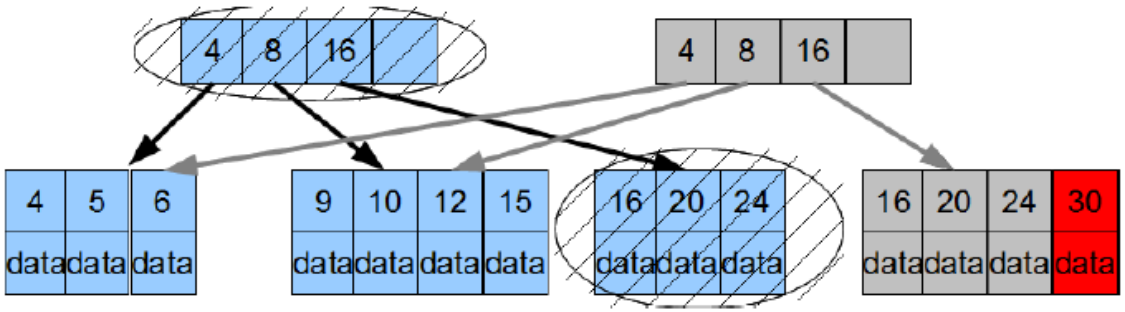
- Space efficiency: sharing of common pages
- Speed: clone creation should be fast
- Number of clones: a large number of clones should be supported
- Clones as first class citizens: a clone can in turn be cloned

Rodehs Research : Clones



Btrfs BTrees

- Stores key/item pairs
- B+ Tree without linked leaves
- Modified in Copy-On-Write (COW) manner
- Reference Counting for filesystem tree
- Collection of btrees
  - device tree
  - extent tree
  - checksum tree
  - data relocation tree .....



Design of Btrfs

Three basic Data Structure

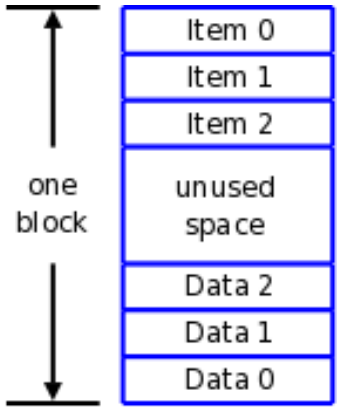
```
struct btrfs_header {
    u8 csum[32];
    u8 fsid[16];
    __le64 blocknr;
    __le64 flags;

    u8 chunk_tree_uid[16];
    __le64 generation;
    __le64 owner;
    __le32 nritems;
    u8 level;
}
```

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

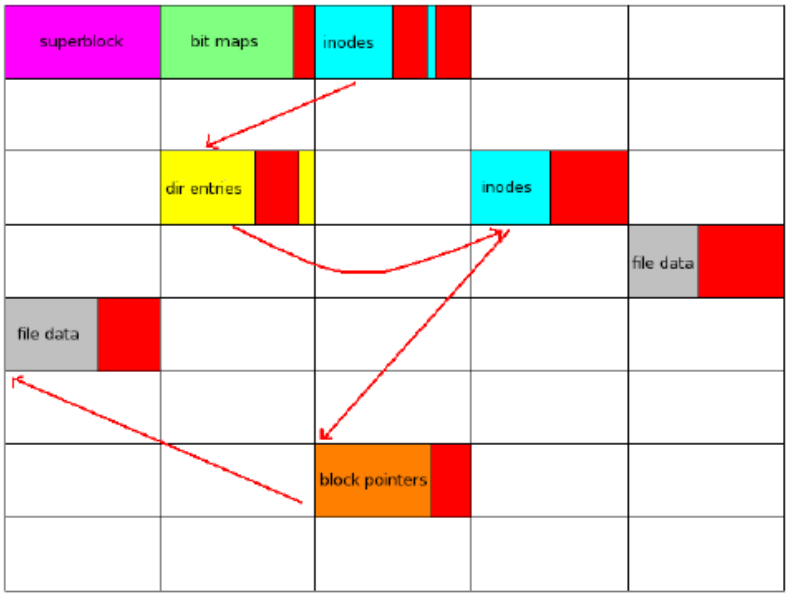
```
struct btrfs_item {
    struct btrfs_disk_key key
    __le32 offset;
    __le32 size;
}
```

Design of Btrfs

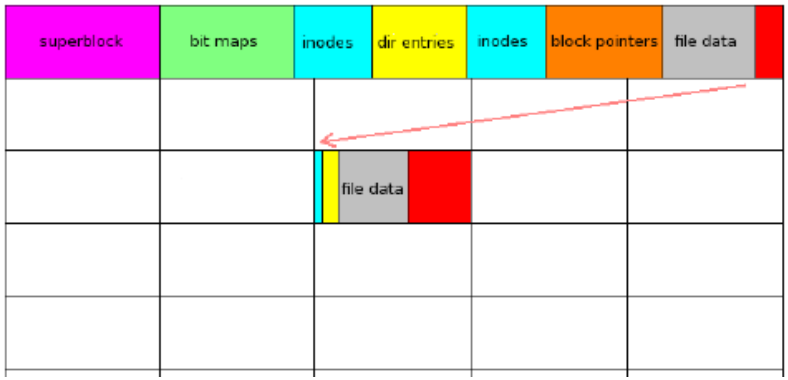
- Interior tree nodes consist only of <key, block pointer> pairs, just like a normal b+-tree
- The tree leaves are extents that contain both items and item data in space efficient storage
- Each leaf may hold any type of data, each kind of which has its own unique type id



Old School Filesystem Data Management



Btrfs Data Management



Sub-Volumes and Snapshot

- Subvolumes in Btrfs are sub-trees of the primary Btrfs filesystem tree. They are created in-place in the existing filesystem, but can be treated like separate filesystems
- Snapshot is specific type of subvolume

References

http://en.wikipedia.org/wiki/Btrfs
http://www.linuxfoundation.org/news-media/blogs/browse/2009/06
http://www.usenix.org/events/lsf07/tech/rodeh.pdf
wiki.whamcloud.com/display/PUB/btrfs