BtrFS

Next Generation Linux Filesystem

Frank Sweetser fs@wpi.edu

March 2010

Standard Disclaimer

- BtrFS still under heavy development
- At least 2 slides are wrong
  - But which ones?
- Not all features implemented yet
- Reserves right to:
  - Corrupt your data
  - Eat your homework
  - Prank call your ex

BACKGROUND

Why Bother?

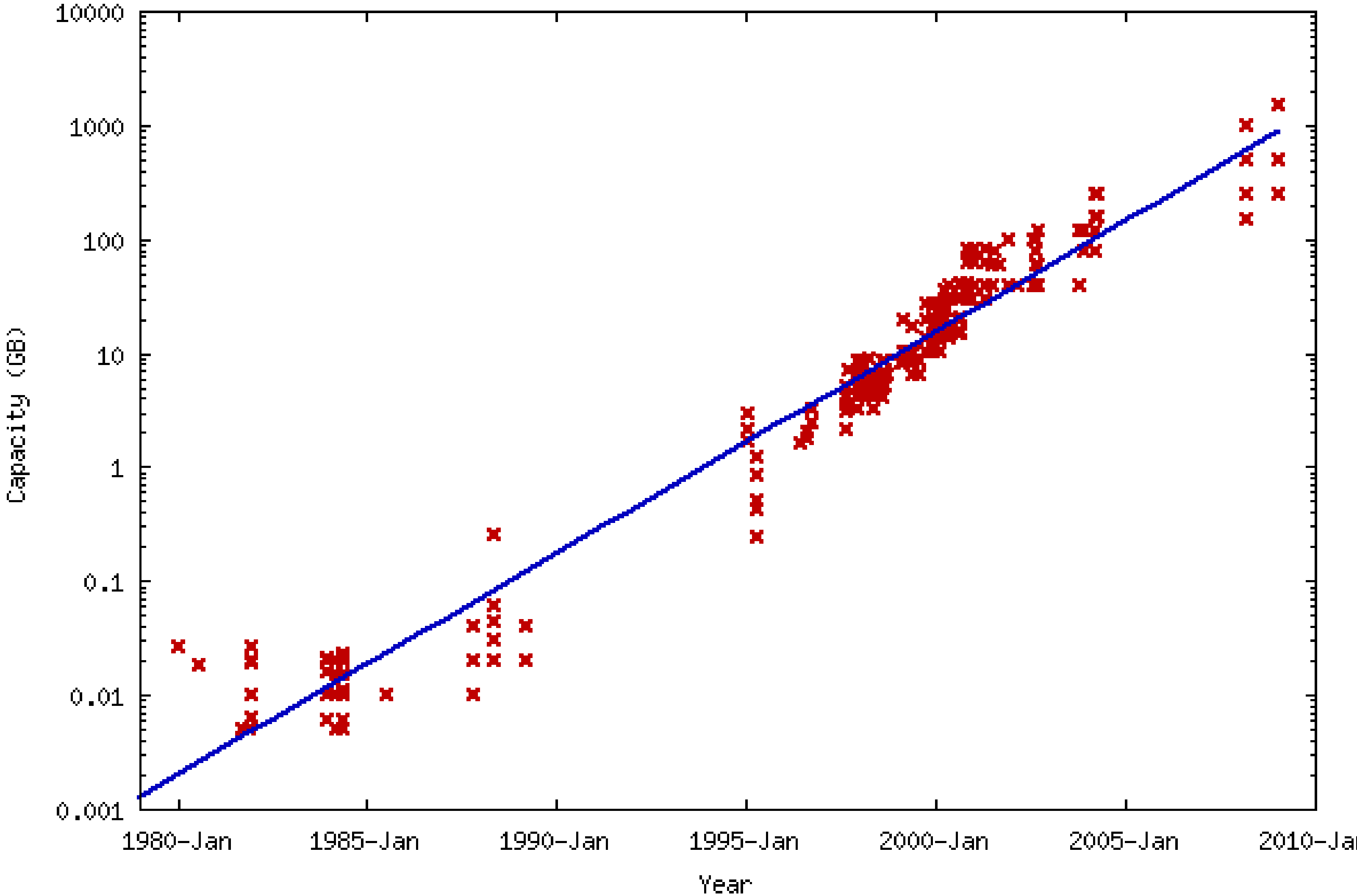
- Ext3
  - Reliable and well trusted
  - Journaled
  - 32TiB volumes, 2TiB file size, 2^{31} files
  - Mainstreamed in 2001
- Ext4
  - Evolution of Ext3
  - 1EiB volumes, 16TiB file size, 2^{32} files
  - Numerous performance tweaks

Today, Tomorrow, and Beyond…

- 2 TiB per disk
- 14 disks per shelf
- 12 shelves per rack
- 336 TiB per rack

What about 5 years from now? 10 years?

Exponential Storage Growth



Scalability

"Scaling is not just about addressing the storage but also means being able to administer and to manage it with a clean interface that lets people see what's being used and makes it more reliable."

- Sean Michael Kerner

More Than Just More Bits

- New media types
- Scale to huge volumes
  - 33 EiB (or more?) per rack in 10 years
  - Must maintain acceptable performance
  - Back up in reasonable time
- High availability
  - Time is $$$
  - Maintenance windows thing of the past

Alternative Filesystems

- ReiserFS3
  - Fragile - fsck can make corruption worse
  - Image files on disk may get merged
- ReiserFS4
  - Not yet in mainstream kernel
  - Future still somewhat in doubt
  - More of same…
- ZFS
  - Lots of good ideas
  - Released by ~~Sun~~ Oracle under CDDL license
  - Not compatible with GPL

BTRFS TO THE RESCUE!

BtrFS Origins

- Conceived in 2007 at Linux Storage and File Systems Workshop
- Nearby USENIX session provides copy on write B-Tree data structures
- Mainline kernel Jan 2009
- Current leading contender for future primary linux filesystem
- Primary author Chris Mason at Oracle

BtrFS Features

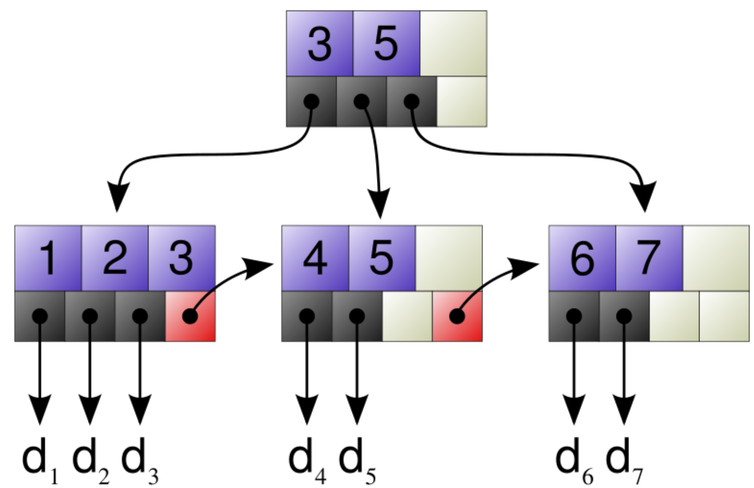
- Huge volume size scalability
  - Raw size
  - Performance
  - Efficiency
- Online operations
- RAID
- Subvolumes and Snapshots
- Migration from Ext3/4
- Efficient backup support
- SSD enhancements
- Transparent compression

Scalability

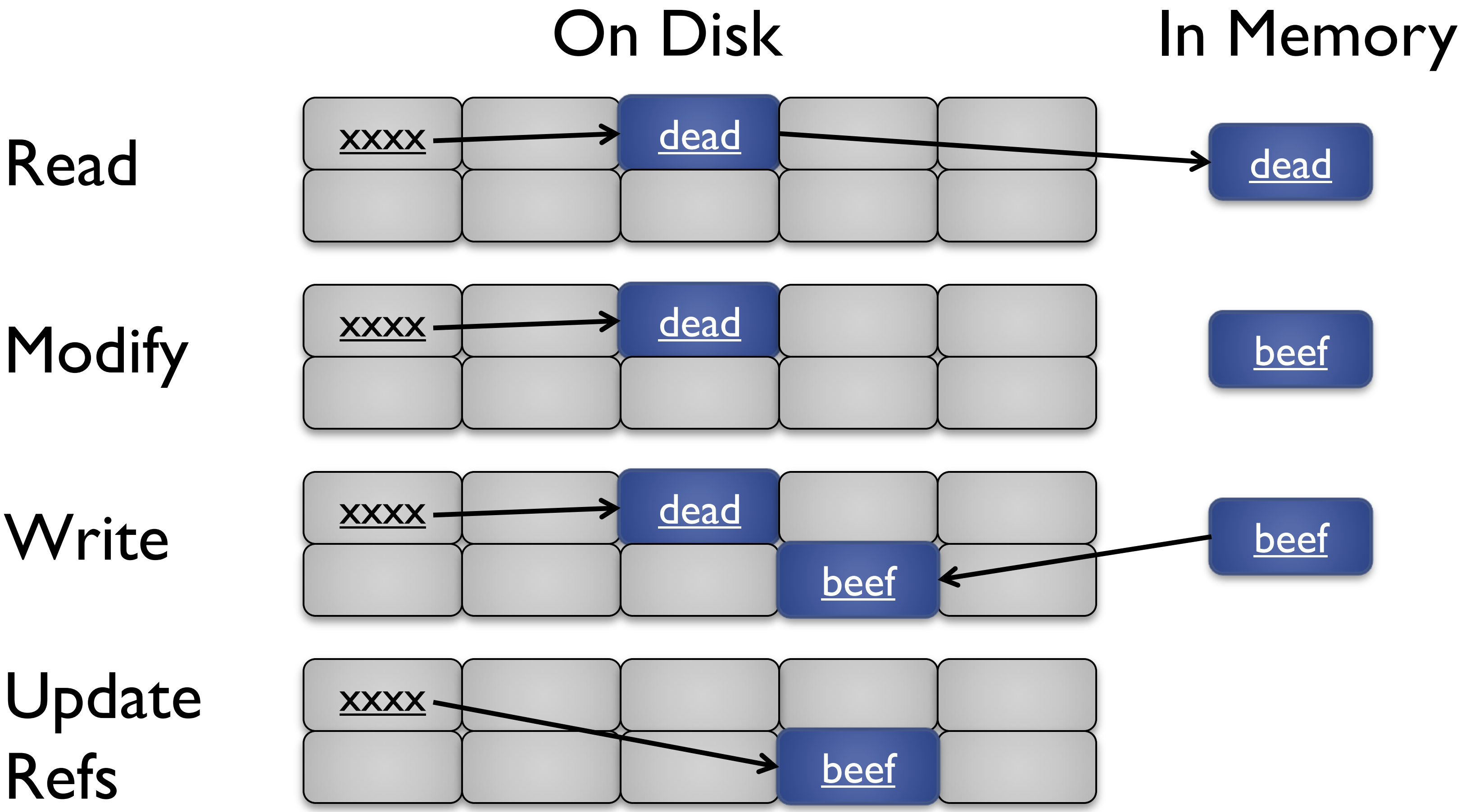
- Huge capacity
  - 2^{64} files per volume
  - 16 EiB max volume size
  - 16 EiB max file size
- Secret sauce: Copy on Write B-Trees
- On-demand inode allocation
- Extents based block tracking

B-Trees: The Butter in BtrFS

- Highly efficient data structure for organizing trees
- Well understood
- Scales to huge trees
- BtrFS uses copy on write safe variant



Copy on Write

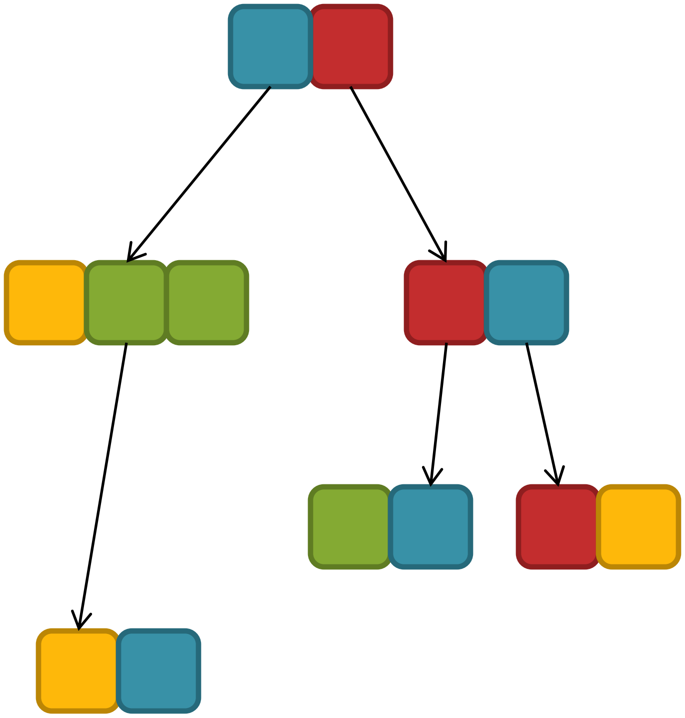


Copy on Write Benefits

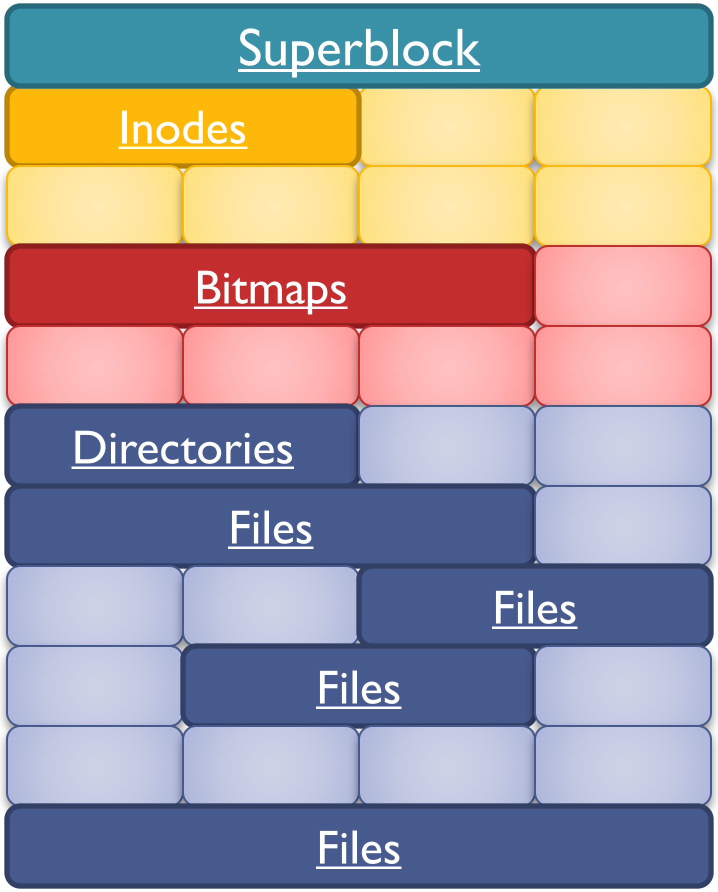
- Disk always in consistent state
- Applies to almost everything
  - Copy on write B-Trees
- (Almost) no fixed location
  - Relocating blocks becomes trivial
  - Metadata can move on the fly
- Essential to online operations
  - Resize
  - Rebalance
  - Migrate

~~Turtles~~ Trees All The Way Down

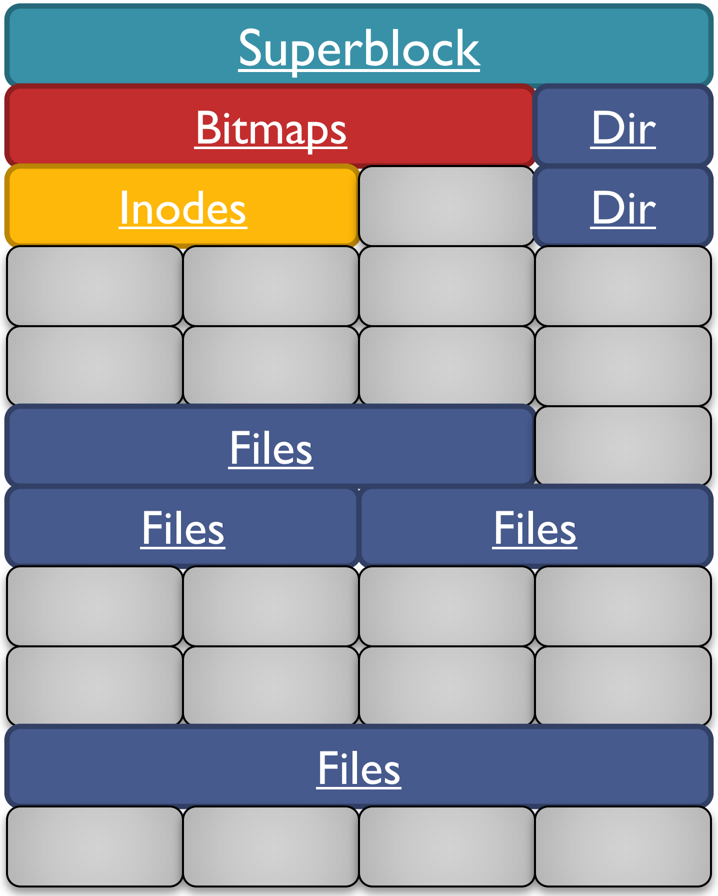
- B-Tree code is data agnostic
- Entire volume is just two B-Trees
  - One for metadata
  - One for data
- *Everything* gets checksummed



Flexible Block Usage



Static Block Types

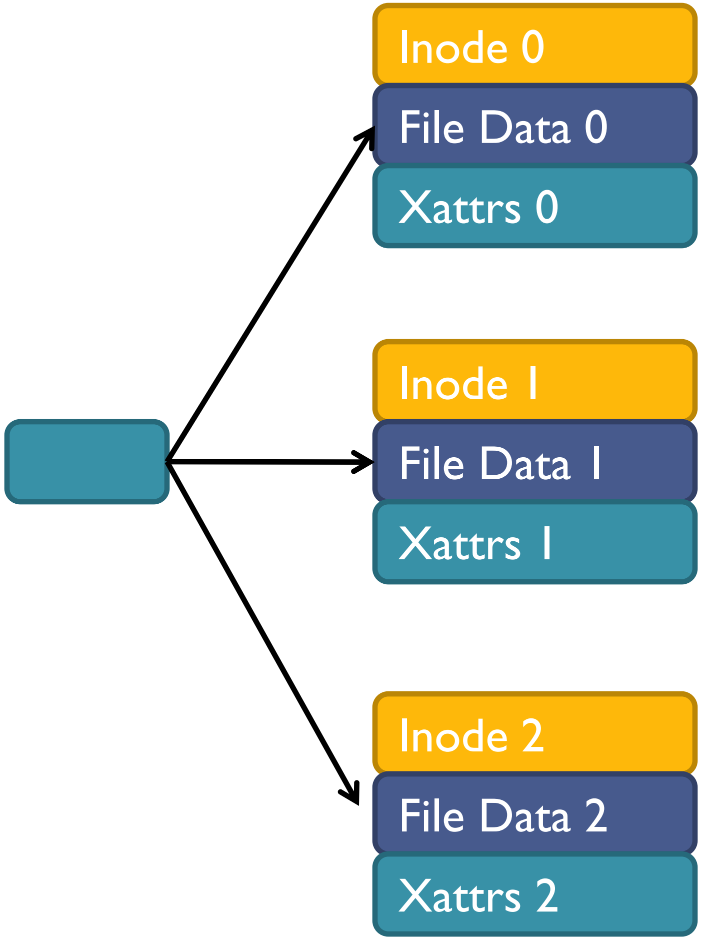


Dynamic Block Types

Clustering Related Blocks

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

- Sorted on objectid before type
- Related objects kept near each other on disk



On-Demand inode Allocation

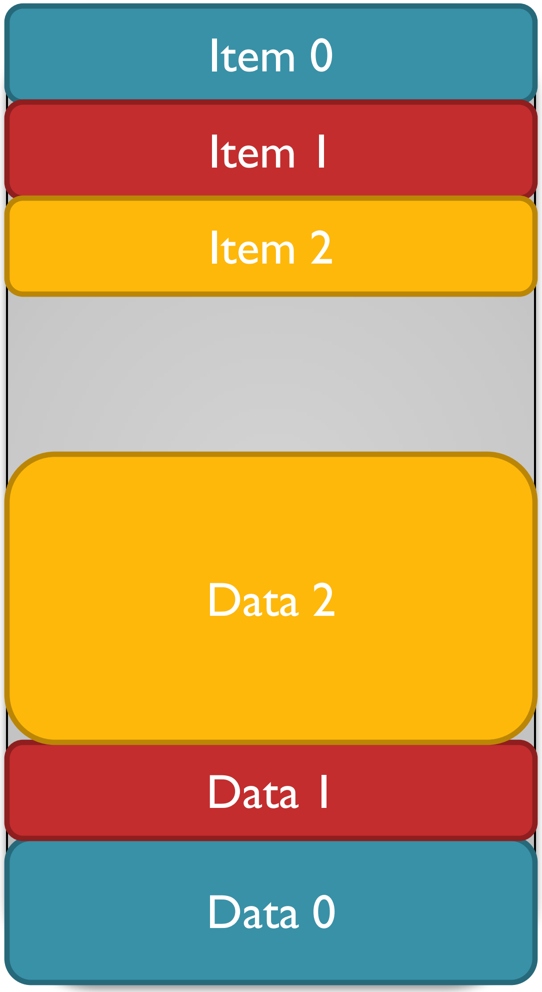
- Volume initially created with small number of inodes
- More created as needed
- Flexible
  - No longer locked into static file count
- Efficient
  - Less wasted disk space
  - ext3/4: ~1.5% pre-allocated to inodes
- Fast
  - 465GiB volume created in < 1s
  - 7GiB *not* used by inodes

Efficient Block Usage

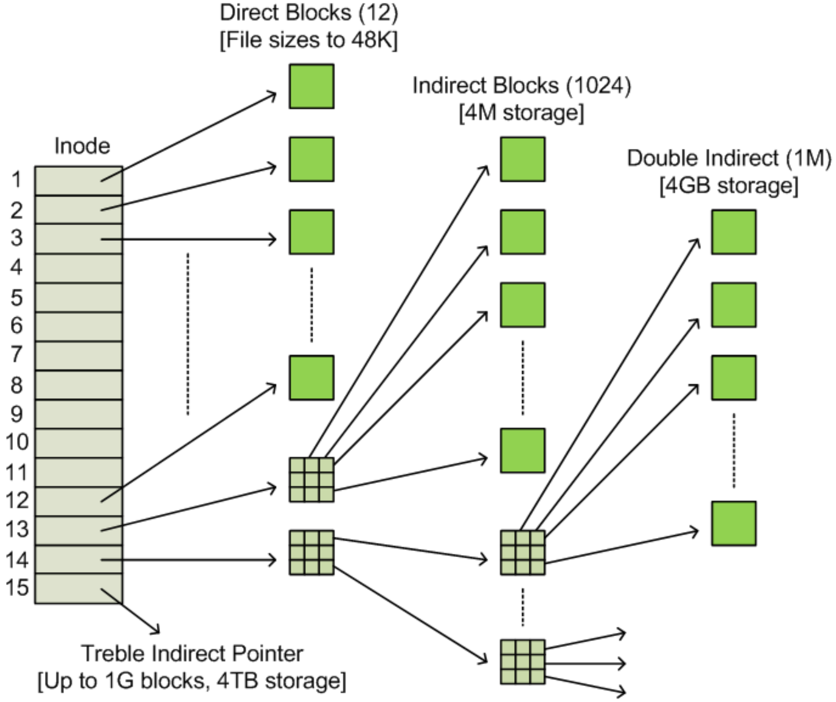
- Multiple leaves per block
- Items packed on top of block

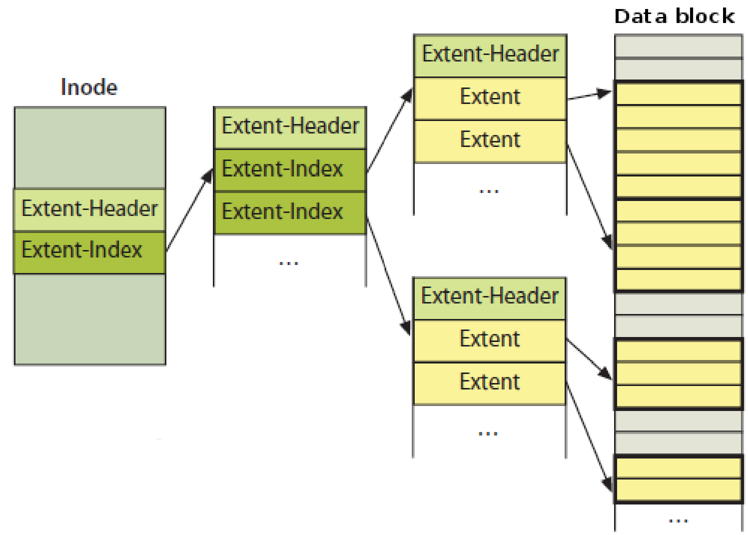
```
struct btrfs_item {
   struct btrfs_disk_key key;
   __le32 offset;
   __le32 size;
}
```

- Data packed on bottom of block



Indirect Blocks vs Extents





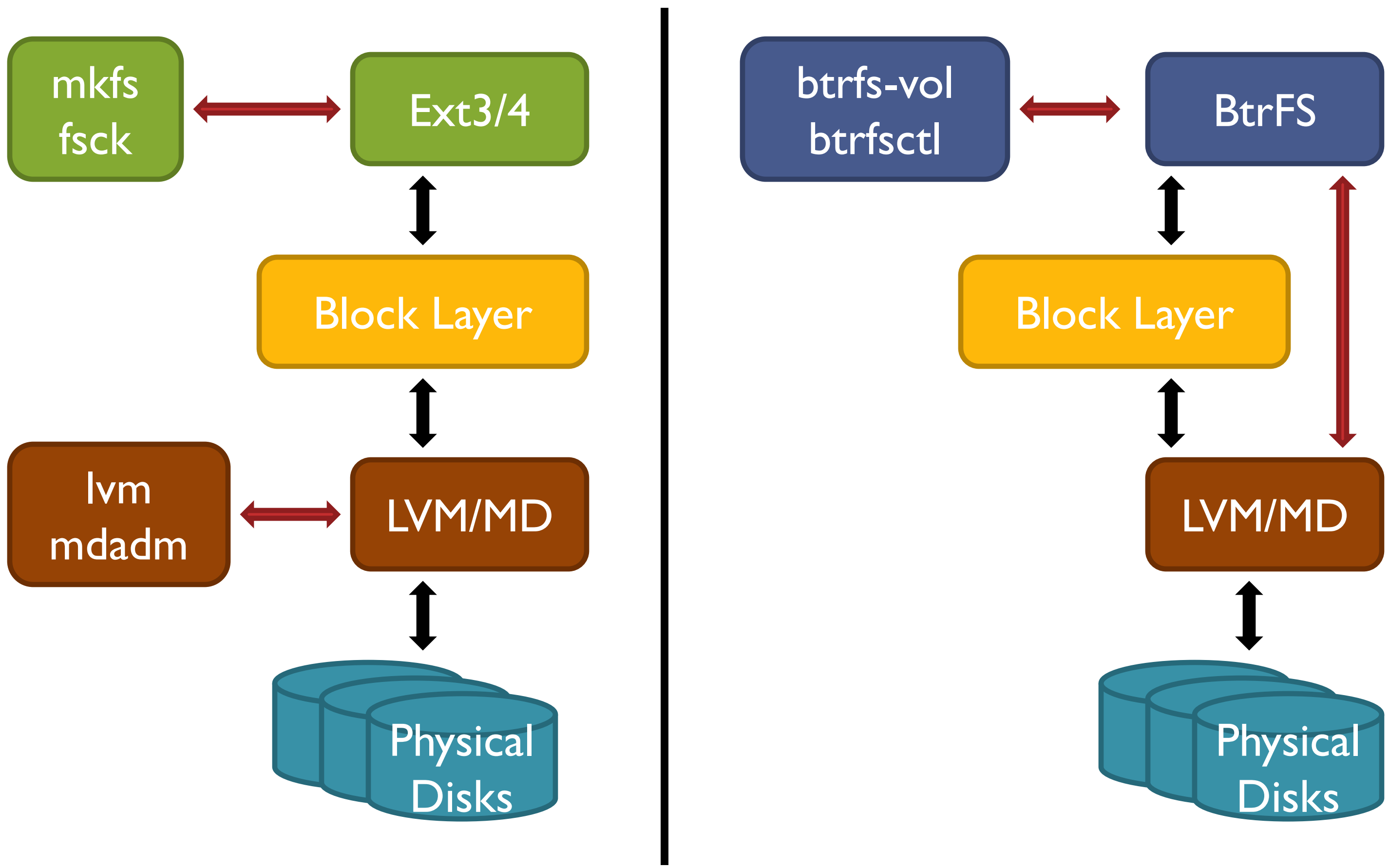
Benchmarks

- Phoronix benchmark published April 2009
  - Fedora 11 preview
  - 2.6.29 kernel
- BtrFS results comparable to Ext3
- Still behind Ext4, XFS
- Rapidly moving target
- Steps to reproduce in References

Online Operations

- Snapshots
- Resize volume
  - Grow
  - Shrink (ZFS can't!)
- Add/remove block device
- Defragmentation
- Rebalance between block devices
- fsck

LVM/MD Integration



Management Operations

Filesystem Data

LVM/MD Integration

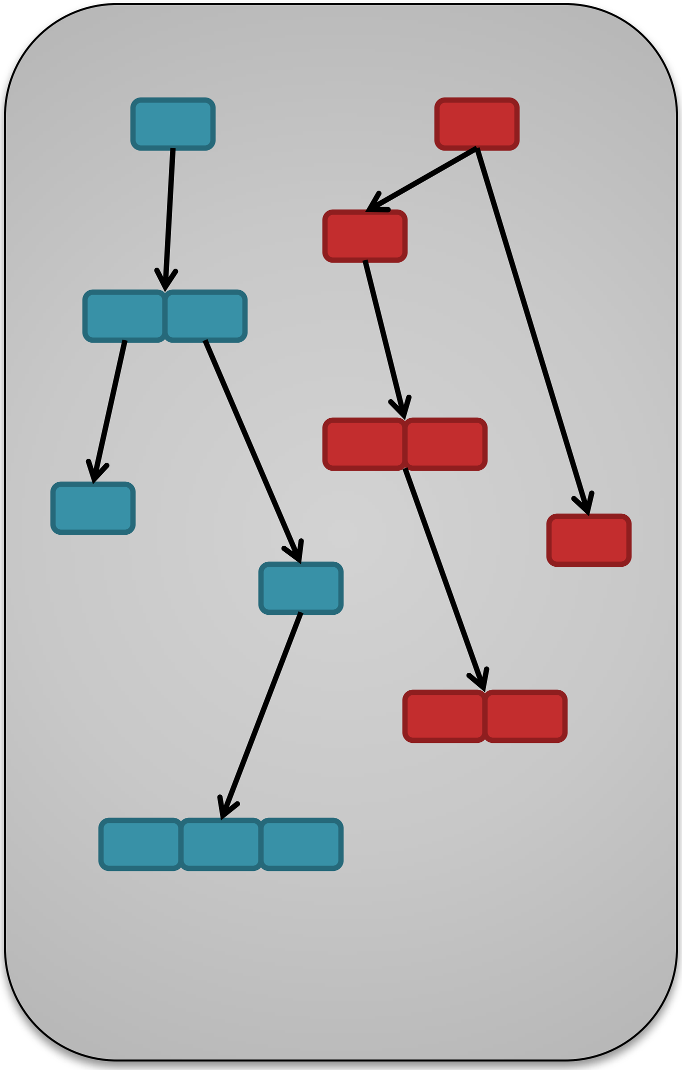
- Directly plugged into much of existing LVM and MD code
- “Rampant layering violation”, but…
  - Ignore unused blocks
  - Make use of TRIM
  - LVM/MD extensions available for future filesystems

RAID

- Currently supported
  - RAID 0, 1, 10
- Future plans
  - RAID 5, 6
- Replication per object, not per block
  - More efficient – ignores unused blocks
  - Different RAID policies per object
  - Default is stripe file data, mirror metadata

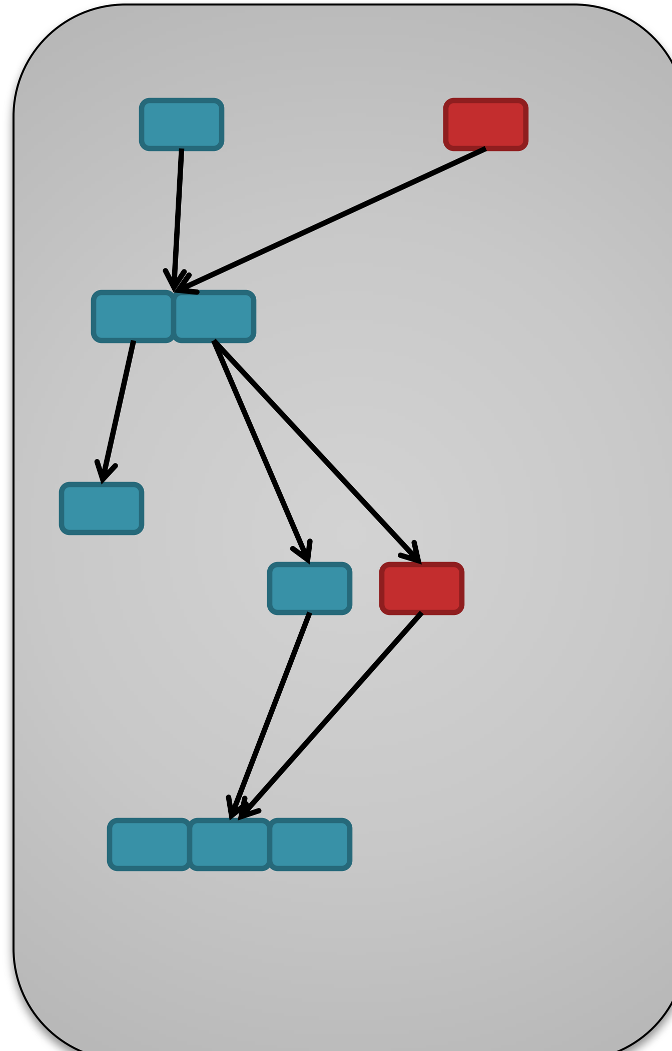
Subvolumes

- Named B-Tree within a BtrFS volume
- Allows multiple "roots" within single volume
- Optional per-subvolume disk quota
- "default" subvolume mounted if unspecified



Snapshots

- Copy on write clone of existing subvolume
- Changes isolated between snapshots
- Stored in blocks already allocated to filesystem



Migration from Ext3/4

- Offline conversion utility
  - Replicates ext3/4 metadata in BtrFS format
  - Data blocks unchanged
  - Places new metadata in unused blocks
- Original ext3/4 saved as subvolume snapshot
  - May be deleted to recover space
  - Trivial rollback
  - Modifications to BtrFS volume *not* propagated back to ext3/4 snapshot!

Backups

- Incremental backups
- Traditional backup algorithm
  - Scan entire volume to find new/changed files
  - Scales with total number of files
- Better: request list of changed files directly from filesystem
- (Not yet implemented)

Solid State Devices

- Already penetrated top end
- Different behavior
  - Wear leveling
  - Write block size vs erase block size
- TRIM command
  - Notifies SSD of unused blocks
  - Must be implemented in filesystem layer
- SSD mount mode
  - Future versions will autodetect SSD media
  - Currently causes performance drop

Compression

- Zlib compression of file data only
- Completely transparent to userspace
- Pros
  - More data stored
  - Less disk IO
- Cons
  - More CPU

EXAMPLES

Creating a BtrFS

- Create a filesystem across four drives

```
mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd /dev/sde
mount /dev/sde /mnt
```

- Don't duplicate metadata on a single drive

```
mkfs.btrfs -m single /dev/sdb
```

Converting Existing FS

- Convert from Ext3/4

```
btrfs-convert /dev/xxx
```

- Mount the resulting Btrfs filesystem

```
mount -t btrfs /dev/xxx /btrfs
```

- Mount the ext3/4 snapshot

```
mount -t btrfs -o subvol=ext2_saved
  /dev/xxx /ext2_saved
```

- Roll back the conversion

```
btrfs-convert -r /dev/xxx
```

- Use the original filesystem

```
mount -t ext3 /dev/xxx /ext3
```

Subvolumes and Snapshots

- Create a subvolume called NewSubVol

```
btrfsctl -S NewSubVol /mnt/test
```

- Create a snapshot of NewSubVol.

```
btrfsctl -s /mnt/test/NewSnapShot
  /mnt/test/NewSubVol
```

- Delete them both

```
btrfsctl -D NewSnapShot /mnt/test/
btrfsctl -D NewSubVol /mnt/test/
```

Add/Remove Devices

- Create and use a volume

```
mkfs.btrfs /dev/sdb
mount /dev/sdb /mnt
```

- Add another device

```
btrfs-vol -a /dev/sdc /mnt
```

- Redistribute extents

```
btrfs-vol -b /mnt
```

- Remove device

```
btrfs-vol -r /dev/sdc /mnt
```

Online Resize

- Add 2GiB to the volume

```
btrfsctl -r +2g /mnt
```

- Shrink the volume by 4GiB

```
btrfsctl -r -4g /mnt
```

- Explicitly set the volume size

```
btrfsctl -r 20g /mnt
```

- Use 'max' to grow the volume to the limit of the device

```
btrfsctl -r max /mnt
```

Replace Failed Device

- Create RAID volume

```
mkfs.btrfs -m raid1 /dev/sda /dev/sdb
```

- /dev/sdb is set on fire, mount without it

```
mount -o degraded /dev/sda /mnt
```

- 'missing' is a special device name

```
btrfs-vol -r missing /mnt
```

- Replacement device can be added normally

References

- "A short history of BrtFS", http://lwn.net/Articles/342892/
- http://en.wikipedia.org/wiki/Hard_disk_drive
- Kerner, Sean Michael (2008-10-30), "A Better File System For Linux", InternetNews.com, http://www.internetnews.com/dev-news/article.php/3781676/A+Better+File+System+for+Linux.htm, retrieved 2008-10-30
- http://btrfs.wiki.kernel.org/
- http://en.wikipedia.org/wiki/B-trees
- "B-Trees, Shadowing, and Clones" http://www.cs.tau.ac.il/~ohadrode/papers/LinuxFS_Workshop.pdf
- SSD TRIM command, http://en.wikipedia.org/wiki/TRIM

References

- "Btrfs Benchmarks: Btrfs Is Not Yet The Performance King", http://www.phoronix.com/scan.php?page=article&item=btrfs_benchmarks&num=1
- http://kerneltrap.org/Linux/Btrfs_Online_Resizing_Ext3_Conversion_and_More
- SSD Mode, http://www.phoronix.com/scan.php?page=article&item=btrfs_ssd_mode&num=1