Linux File Systems

Terse Overview

Miguel Filipe - April 2012 - Amazon AWS - Database Services - DynamoDB

Summary

- Ideas & Concepts
  - Journaling
  - Block vs Extent allocation
- Mature Filesystems
  - Differentiating characteristics
  - Issues
- Recent developments
  - Ext4
  - BTRFS

Journaling

- Tolerance to:
  - power failures
  - Kernel panic
- Faster recovery
- Improved FS integrity

- Journaling level?
  - Journal
  - Ordered (data 1st)
  - Write-back

Alternatives to Journaling

- Log Structured Filesystems
- Copy-on-Write Filesystems

Block or Extent

- External Fragmentation
- What block size?
- Atomicity of writes?
- Free vs used space tracking and management

- Internal fragmentation
- What extent size?
- On disk allocation?

Other Ideas & Concepts

- DirectIO
- Discard & Trim
- SSD vs rotating media
- Delayed Allocation
- Snapshots
- Fallocate()
- Ftruncate()
- Sparse files

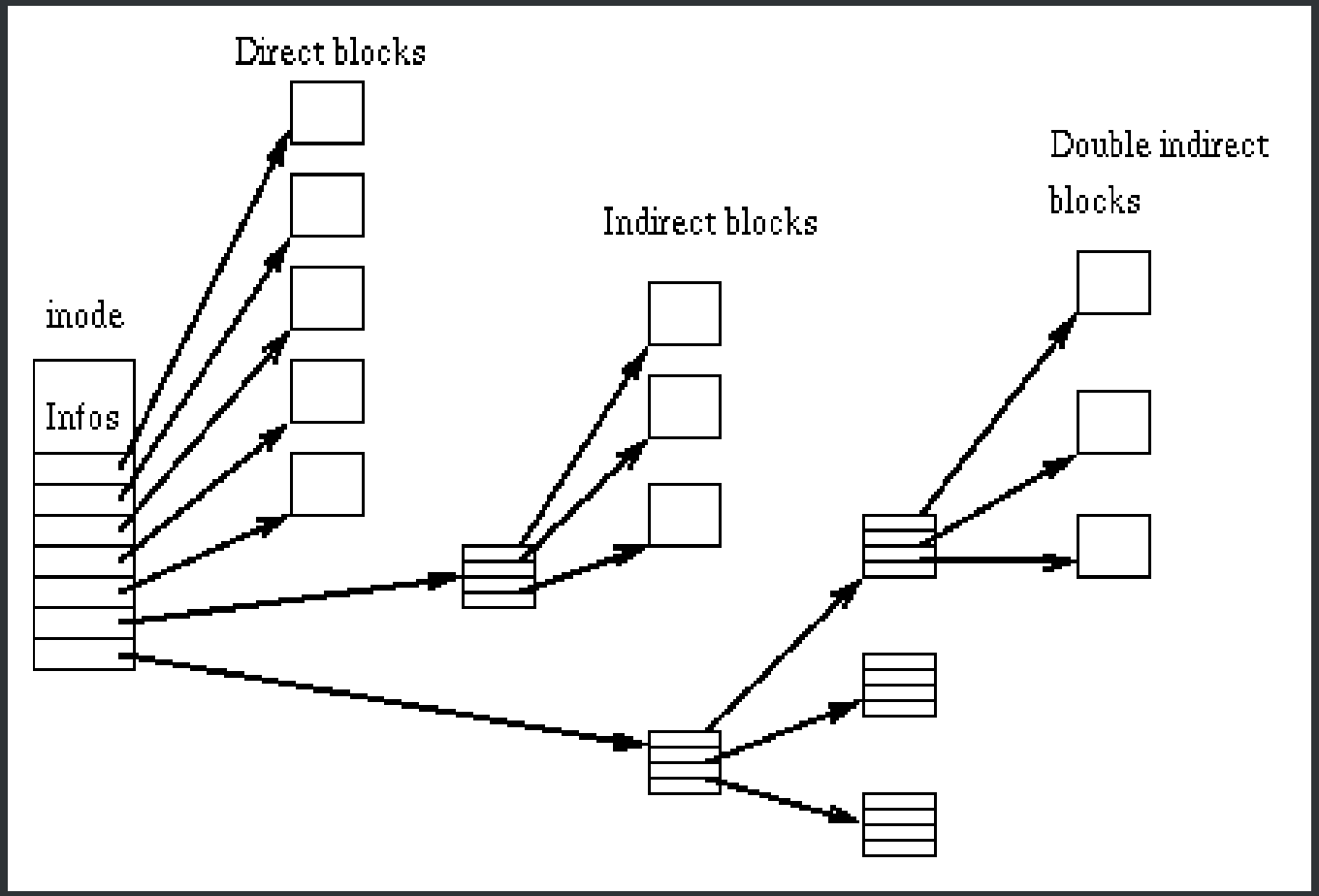
Mature FileSystems

- Ext2 & **Ext3**
- **XFS**
- ReiserFS
- ... others ...

Ext3

- Evolution of Ext2
- Added Journaling to the FS
- Quite fast on metadata operations
- Balanced performance on a large variety of workloads
- Quite reliable and robust to “crashes”
- Simple design and comprehensive understanding (when compared to some of the alternatives)

A file in EXT3



Ext3- Limitations

- Directories:
  - With +32k sub-dirs (counter limit)
  - With + ~32k files (performance degradation)
  - (2009) Recently uses Htrees, problems @ ~1M
- *Data intensive workloads*
- Maximum Nº of files is determined at creation time
- Can't *cope* very well with:
  - Large files (>1Gb)
  - Large filesystems
- Doesn't cope at all with silent data corruption

Motivation for Alternatives

Gap between increasing capacity & platter speed

versus constant seek time.

Need to scale:

- File/Directory count and size
- Disk sizes
- Creation time, Fsck, repair time, defragmentation
- Faster HW: *SSD/Raids* ..
- Bigger IO loads
- Bigger throughtput loads

Support for new features: fallocate(), discard(), ...

XFS

- Came from SGI's IRIX, created in the 90s
- Designed for large HPC systems

  (+300TB disk arrays)
- Optimized for high volume, highly parallel IO

  (HPC applications)
- Journalled, extent-based filesystem, very feature rich

XFS

Optimized for "*data intensive workloads*"

- Handles very well:

  Large Files & Filesystems (2^64/64bit)

  - Low latency data-streaming
  - Streaming at *platter speed*
  - Large disk arrays/raids
- Scales well on SMP and SMP cc/numa
- Supports multi-disk stripping & journal offloading
- Best performing Linux FS for MySQL/InnoDB

XFS – limitations

- Harder to maintain, smaller developer community
- Complex code - *port*/evolution from the XFS present in IRIX (*non-linux-kernel-native-code*)
- Not so good at metadata operations (better now)
- By *design* the journaling system is less tolerant
- Doesn't cope with silent data corruption
- Less used, not as tried & tested as EXT3

New requirements

- Current HW densities and die shrinks mean more *bad blocks* e more **silent** *data corruption*
  - Coping with data corruption (active & passive)
- Backup Management
- Disk and Volume management
- Achieve good performance on Solid State Disks
  - Matching behaviour to underlying HW design
- Support/Implement new user *API*s: sparse files, data dedup, hole punching, scrubbing.

Wanted features

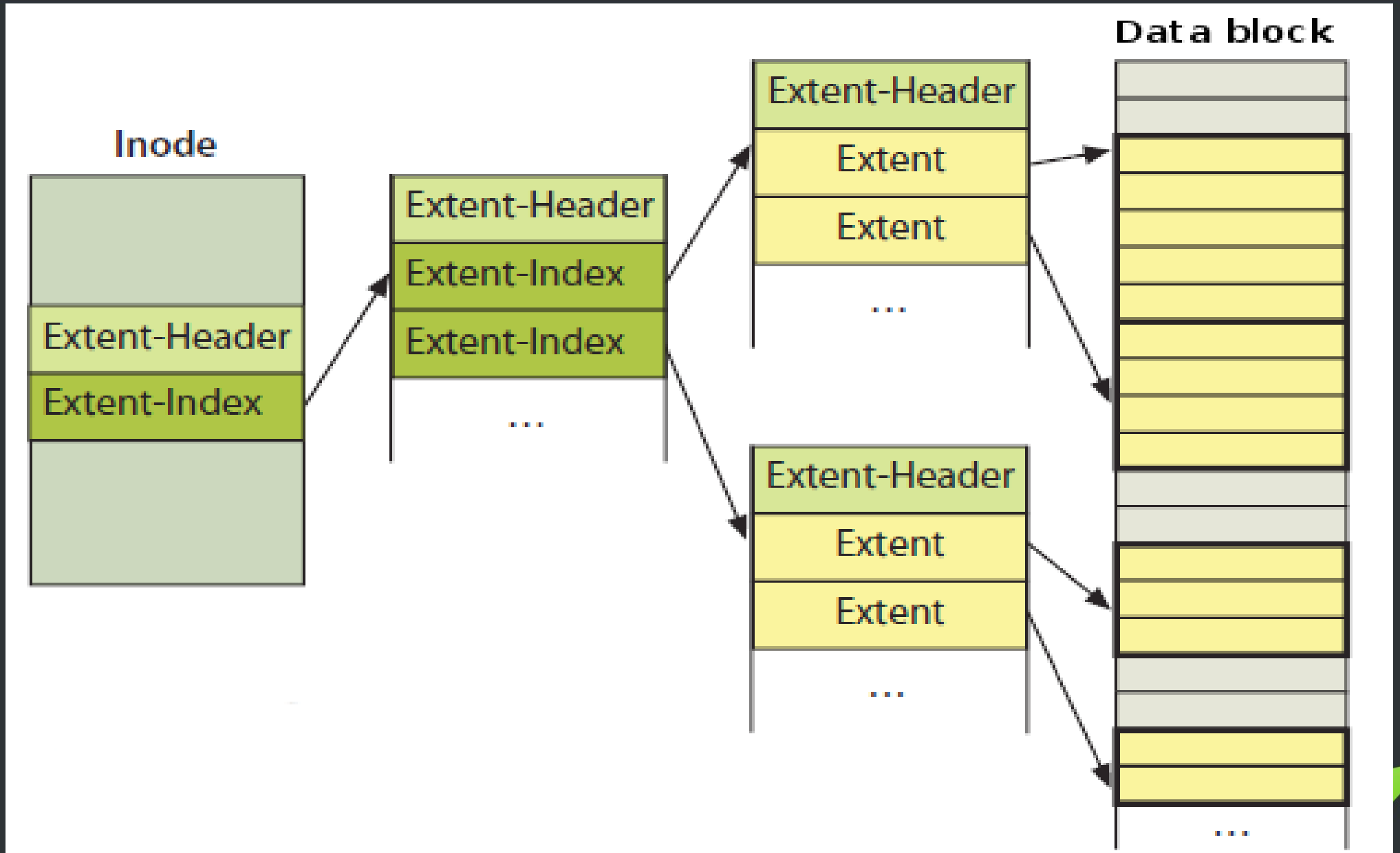
- Handling corruption
  - Data
  - Metadata
- How ?
  - Internal redundancy
  - CRC codes
  - Canaries
- Achieve HW speed

- Built into the *FS*:
  - Volume management
  - Disk management
  - Backups
  - Snapshots
  - Compression
  - Encryption
  - Data *deduppplication*

Ext4

- Evolution of Ext3, backwards compatible.
- *Stable* in Linux-2.6.28, 2008-12 (RHEL5.6 2011-01)
- Handles better bigger files and filesystems.
  - *Extent based filesystem*
  - *Delayed Allocation*
  - *Persistent Pre-allocation*
  - *Faster fsck (*proportional to the data and not FS size*)*
  - *Online Defragmentation*
  - *Htrees for extents, directory entries & links*

A file in Ext4



Ext4

- Better tolerance to HW/disk faults:
  - Journal Checksumming
  - *Live inodes have* canaries/health checks
- Support for bigger *things* (>1 EB):
  - Volumes e Files (2^48/48bits)
- Nanosec time resolution
- Faster Fsck()
- Journal is optional (thanks Google!)
- *Implements new APIs: discard, sparse files, ...*

What about...

- Better handling of **DATA** corruption?
- Snapshots?
- Built-in compression and/or encryption?
- Disk & Volume management?
- Even better performance?

XFS & Ext4 fail to meet all requirements...

BtrFS

- New Linux FS, started in 2007, mainline in Linux-2.6.29 (experimental). Created by Chris Mason, kernel hacker for Oracle
- Well received by the kernel hackers comunity
- Targeted successor for Ext4
- Developed within the linux open source comunity at large & with paid support/development by:
    - Oracle, HP, IBM, RedHat, SuSE, others

BtrFS

- *Extent Based, Copy on Write, Delayed Allocation*
- Gigantic *things* (2^64/64bit)
  - nº of files, file size, # volumes, # snapshots ...
- Multi-Disk & Multi-Volume
- *Writable Snapshots*
- **Data** & Metadata *checksumming*
- Plugable compression, encryption
- Data *dedup, scrubbing, rebalancing*

BtrFS

- Online fsck e defragmentation
- Fast offline fsck – promised, didn't land yet
- B-Tree for everything:
  - Sub-Volumes, Snapshots
  - data (extents), Dir *entries, inodes*
- Multi-threaded, multi-processor friendly
- Design contemplates SSDs from the go
- Big focus in minimizing seek count
- Distinct levels of redundancy for data & meta-data

BtrFS

- Modular design, we can enable/disable:
  - Data Checksumming
  - Metada Checksumming
  - Copy-on-Write (not so good for DBs)
  - *SSD* or *spindle* mode
- Tight integration with Linux kernel components
  - Multi-disk: md layer: raid5, raid6 e others (planned)
  - Vfs, Bio, thread-pool, locking, rcu, buffer cache, etc

BtrFS

- The name BtrFS comes from:
  - B-Tree based
  - *Better* FS
- It's possible to live migrate a ext3/4 volume to btrfs using the free space in the mounted volume
- Quite solid by now, SuSE stamps it production ready
- http://btrfs.wiki.kernel.org
- http://en.wikipedia.org/wiki/Btrfs

Others

- LogFS - http://logfs.org/
- NILFS - http://www.nilfs.org/

Recent Ext4 improvements

- Ext4 in RHEL5 doesn't have DISCARD support
- There are several modes of DISCARD support
  - On free (2.6.33), Periodic (2.6.27?), batched (2.6.37)
- Punch-hole support (3.0)
- Scalability/Throughput improvements (up to 3.1)
- Clustered Allocation

  (lower latencies specially on appends)
- SMP scalability improvements (2.6.39)

Recent Improvements

- Transparent huge pages support
- DISCARD support across the stacks
- VFS scalability
- Increased default TCP recv window + more
- Network scalability (recv packet+flow steering)
- XFS delayed logging
- Filesystem ↔ writeback perf improvements
- Filesystem ↔ b-io & block layer improvements

DynamoDB wanted FS features

- Fallocate() - pre allocation of contiguous space
- Batched Discard support for SSDs ?
- Direct IO and Asyncronous IO
- Minimal metadata IO overhead (journals are BAD)
- Log based or Copy-on-Write?
- Volume management?
- Snapshot support?
- *Hole-Punching* – release/delete parts of files