Quo vadis Linux File Systems: An operations point of view on EXT4 and BTRFS

Udo Seidel

Agenda

- Introduction/motivation
- ext4 – the new member of the extfs family
  - Facts, specs
  - Migration
- BTRFS – the newbie .. the hope
  - Facts, specs
  - Migration
- Summary

File systems challenges

- Speed
- Size/growth
- Flexibility
- sustainibility

Linux file systems

- More than 50 file systems shipped with Linux kernel
  - Local
  - Remote
  - Cluster
  - ...
- A few as standard for root directory
  - ext2, ext3
  - XFS

Operational challenges with Linux file systems

- ReiserFS
  - sun-setted
  - Big installation base
- Ext3
  - Stable, robust
  - Slow
- XFS
  - Additional costs?
- Changes in recent Enterprise distributions

Chances, possibilities, choices for the next Linux file systems

- New version of the ext family -> ext4
  - Marked as stable
  - Shipped with Enterprise distributions
- New approach with BTRFS
  - Still experimental in vanilla kernel
  - Shipped as production ready by some Enterprise distributions

4th extended file system

- Shipped since 2.6.19
- Stable since 2.6.28
- To overcome limits of ext3
  - Size
  - Performance

Ext4 - facts

- Max volume size: 1 EB = 1024 PB (ext3: 16 TB)
- Max file size: 16 TB (ext3: 2TB)
- Max length of file name: 256 Bytes
- Support of extended attributes
- No encryption
- Not really compression
- Partially 64bit

Ext4 – operational first glance

- OPROC/procedure changes?
- Behaviour changes?
- Migration

Ext4 – OPROCs/procedures

- Known tools
  - *mkfs*
  - *fsck*
  - *tune2fs*
  - *e2label*

=> *Easy to re-use or adapt of existing :-)*

Ext4 – Behaviour changes ...

- Extents
- Improved block allocation
- Journaling
- File system check

Ext4 – from blocks to extents

- Common addressing for modern file systems
- Contiguous area of blocks
  - Less management information needed
  - Less meta data operations
  - Less “fragmentation”
- Requires change of on-disk format :-(
  - Migration paths?

Ext4 – delayed allocation

- Use cache information for placement
- Less fragmentation
- Risk of data loss in early versions => improved since 2.6.30
- No change of on-disk format! :-)

Ext4 – “clever” allocation

- Support of system call *fallocate()*
  - Application reserves blocks ahead
  - File system ensures disk space availability
- Allocation information in extent structure
  - Only usable with extents!

Ext4 – consistency via JBD2

- Transactions have checksums
- 64 bit ready
- Deactivation possible
  - Keep other ext4 features
  - Comparison: ext3 ... -> ext2

Ext4 – repair

- Improved *fsck()*
  - No check of unused blocks
    - information stored in block group header
    - Information secured via checksums
    - (de)activation possible at any time
  - First run possibly as slow like in ext3

Ext4 – other news

- Nano second precision time stamps
  - Unix millennium bug shifted to 2514
- More subdirectories
  - Up to 65000
  - More than 65000 ... with limitation

Ext4 – background for migration

- 2 kind of changes compared to ext3
  - change of on-disk format:
    - Extents
    - Only enabled for new files via *tune2fs*
    - Additional tasks needed
  - On-disk format not relevant:
    - block allocation
    - Immediately enabled via *tune2fs*

Ext4 – general migration paths

- *mkfs()* and backup/restore
  - Clean new file system structure
  - Only way for file systems other than ext2/3
  - Extended outage
- Conversion via *tune2fs*
  - Partial only
  - Possible for ext2/3 family only
  - Faster/easier

Ext4 – migration strategy

- Both options
  - Significant installation base of non-ext2/3
  - Big landscape (2500+ Linux servers)

Ext4 – migration via mkfs()

- For all *new* deployments
- for existing file systems
  - On demand
  - By chance

Ext4 – migration via tune2fs

- Results in mix of ext3 and ext4 structure
- Access via ext3 driver impossible
- *fsck()* needed

parameter	description
extent	Extent based block allocation
flex_bg	Flexible placement of meta data
uninit_bg	Flag uninitialized blocks for faster fsck
dir_nlink	Infinite number of sub directories
extra_isize	Timestamps with nano seconds

Ext4 – migration hints

- *fsck()* recommended
- /boot – booting from ext4 possible?
- Rescue media enabled for ext4?
- Backup/restore of ext4?

Ext4 – summary

- Good successor of ext3
- Manages higher amount of data
- Faster
  - Performance
  - recovery
- Safer
- Sufficient migration options from ext2/3

Better/b-tree file system

- Shipped since 2.6.29
- Still experimental
  - In vanilla kernel
  - But .... production ready in some Enterprise distributions
- Supposed to replace ext3/4
- New storage management approach

BTRFS - facts

- Max file/volume size: 16 EB
- Max length of file name: 256 Bytes
- Support of
  - Extended attributes
  - No encryption yet
  - Snapshot
  - Compression
  - Copy-on-Write

BTRFS – OPROCs/procedures

- Known tools
  - *mkfs ... with limitation*
  - *fsck ... with limitation*
  - *tune2fs -> nope*
  - *e2label -> nope*

=> *No easy to re-use or adapt of existing :-(*

BTRFS – behaviour changes ...

- Device management
- Snapshots
- Compression
- File system check

BTRFS – device management

- Included volume manager
  - RAID-0, RAID-1, ...
  - Add/remove devices
  - ...
- New tool
  - change of OPROC's/procedures :-(
  - Change to GUI or new GUI tools :-(

BTRFS – snapshots

- Not really new ... but not used so far
- Integration into package management
  - automatic - no additional tasks
  - Faster roll-back
  - Post change tracking
- New tool
  - New OPROC's anyway :-|
  - Minimal human intervention :-)

BTRFS – compression

- Transparent
- Flexible:
  - different algorithms possible
  - Easy to switch on/off
- Operations:
  - Minimal changes on admin side
  - No changes on other levels :-)

BTRFS – file system check

- Very very basic ... almost not existing
- Reliable?
- Alternatives?
  - Backup/restore
    - Possible longer outage :-(
    - covered in OPROC's :-)
  - Snapshots -> new at all :-(

BTRFS – what else

- Support of POSIX ACL's
- Online grow/shrink
- Online add/removal of disks
- SSD-aware
- Management tool evolution (*btrfsctl* -> *btrfs*)
- *du/df* not fully BTRFS-aware

BTRFS – migration paths

- *mkfs()* and backup/restore

  OR

- In place from ext3/4 via tool *btrfs-convert*
  - Via *libe2fs*
  - BTRFS meta data location flexible
  - Old ext3/4 organized in snapshot
  - Roll-back possible to date/time of conversion

BTRFS – migration strategy

- In place via btrfs-convert
  - Fast
  - Ext3/4 a given in data centre by then
  - Change management friendly

BTRFS – migration hints

- New file system – check everything!
- General knowledge/awareness
- */boot* – booting from BTRFS possible?
- Rescue media enabled for BTRFS?
- Backup/restore of BTRFS?

BTRFS summary

- Still experimental
- Meets standard file systems requirements
- Bridges existing gaps
- Easy migration from ext3/4 possible
- New approach to storage management
- Prospects
  - SSD
  - Compression

Summary

- Improvement moving to ext4
- Safe switching to ext4
- In place migration from ext3 possible
- Future is BTRFS
- In place migration from ext3/4 to BTRFS possible

References

- http://ext4.wiki.kernel.org
- http://btrfs.wiki.kernel.org

Thank you!