Ext2/3/4, XFS, Btrfs, WTF?

René Pfeiffer

web.luchs.at

30. April 2012

Inhaltsübersicht - Wovon reden wir?

- Grundlagen
- GNU/Linux Plattform und Dateisysteme
- Ext*n* Familie
- XFS
- Btrfs
- Btrfs Beispiele
- Workloads
- Wahl des „besten“ Dateisystems für alle Situationen!
- **Keine** Benchmarks!

ZFS? Ist in Debian GNU/kFreeBSD enthalten!

Inhaltsübersicht - Wovon reden wir?

- Grundlagen
- GNU/Linux Plattform und Dateisysteme
- Ext*n* Familie
- XFS
- Btrfs
- Btrfs Beispiele
- Workloads
- Wahl des „besten“ Dateisystems für alle Situationen!
- **Keine** Benchmarks!

ZFS? Ist in Debian GNU/kFreeBSD enthalten!

Grundlagen von Dateisystemen

Aufgaben von Dateisystemen

- Management von Speicherplatz
- Verwaltung von Dateinamen und Verzeichnissen
- Verwaltung von Metadaten
- Berechtigungen
- Erhalten der Datenintegrität / Konsistenz
- Performance

Aufgaben von Dateisystemen

- Management von Speicherplatz
- Verwaltung von Dateinamen und Verzeichnissen
- Verwaltung von Metadaten
- Berechtigungen
- Erhalten der Datenintegrität / Konsistenz
- Performance

Daten und Metadaten

- Metadaten sind „Daten über Daten“
  - Dateigröße
  - Datei-/Verzeichnisnamen
  - Zeitstempel, Berechtigungen, . . .
- Metadaten und Daten werden verschieden behandelt
  - Inodes für Metadaten
  - „normale“ Blöcke für Daten
- Metadaten und Daten nur im Team sinnvoll
  - siehe Suche im `lost+found`
  - siehe Dateien gefüllt mit Nullbytes

Daten und Metadaten

- Metadaten sind „Daten über Daten“
  - Dateigröße
  - Datei-/Verzeichnisnamen
  - Zeitstempel, Berechtigungen, . . .
- Metadaten und Daten werden verschieden behandelt
  - Inodes für Metadaten
  - „normale“ Blöcke für Daten
- Metadaten und Daten nur im Team sinnvoll
  - siehe Suche im `lost+found`
  - siehe Dateien gefüllt mit Nullbytes

Daten und Metadaten

- Metadaten sind „Daten über Daten“
  - Dateigröße
  - Datei-/Verzeichnisnamen
  - Zeitstempel, Berechtigungen, . . .
- Metadaten und Daten werden verschieden behandelt
  - Inodes für Metadaten
  - „normale“ Blöcke für Daten
- Metadaten und Daten nur im Team sinnvoll
  - siehe Suche im `lost+found`
  - siehe Dateien gefüllt mit Nullbytes

Daten und Metadaten

- Metadaten sind „Daten über Daten“
  - Dateigröße
  - Datei-/Verzeichnisnamen
  - Zeitstempel, Berechtigungen, . . .
- Metadaten und Daten werden verschieden behandelt
  - Inodes für Metadaten
  - „normale“ Blöcke für Daten
- Metadaten und Daten nur im Team sinnvoll
  - siehe Suche im `lost+found`
  - siehe Dateien gefüllt mit Nullbytes

Die leidige Integrität / Konsistenz

- bestimmte Änderungen im Dateisystem erfordern viele Operationen
- Änderungen verlaufen nicht zeitgleich
- Absturz / Stromausfall / Reset
  - Dateisystem im undefinierten Zustand
  - File System Check (*fsck*) erforderlich
- Integrität $\neq$ Konsistenz
  - Konsistenz = definierter Zustand des Dateisystems
  - Integrität = überprüfte Daten und Metadaten
- Prüfung von Daten und Metadaten bei großen FS zeitaufwendig
  - $\approx$ 2h für $\approx$ 4 TB Ext4 mit Millionen von Dateien
  - `xfs_repair` kann Minuten/Stunden benötigen
  - Mag wer `fsck.vfat` mit 8+ TB Daten versuchen?

Die leidige Integrität / Konsistenz

- bestimmte Änderungen im Dateisystem erfordern viele Operationen
- Änderungen verlaufen nicht zeitgleich
- Absturz / Stromausfall / Reset
  - Dateisystem im undefinierten Zustand
  - File System Check (*fsck*) erforderlich
- Integrität $\neq$ Konsistenz
  - Konsistenz = definierter Zustand des Dateisystems
  - Integrität = überprüfte Daten und Metadaten
- Prüfung von Daten und Metadaten bei großen FS zeitaufwendig
  - $\approx$ 2h für $\approx$ 4 TB Ext4 mit Millionen von Dateien
  - `xfs_repair` kann Minuten/Stunden benötigen
  - Mag wer `fsck.vfat` mit 8+ TB Daten versuchen?

Die leidige Integrität / Konsistenz

- bestimmte Änderungen im Dateisystem erfordern viele Operationen
- Änderungen verlaufen nicht zeitgleich
- Absturz / Stromausfall / Reset
  - Dateisystem im undefinierten Zustand
  - File System Check (*fsck*) erforderlich
- Integrität $\neq$ Konsistenz
  - Konsistenz = definierter Zustand des Dateisystems
  - Integrität = überprüfte Daten und Metadaten
- Prüfung von Daten und Metadaten bei großen FS zeitaufwendig
  - $\approx$ 2h für $\approx$ 4 TB Ext4 mit Millionen von Dateien
  - `xfs_repair` kann Minuten/Stunden benötigen
  - Mag wer `fsck.vfat` mit 8+ TB Daten versuchen?

Die leidige Integrität / Konsistenz

- bestimmte Änderungen im Dateisystem erfordern viele Operationen
- Änderungen verlaufen nicht zeitgleich
- Absturz / Stromausfall / Reset
  - Dateisystem im undefinierten Zustand
  - File System Check (*fsck*) erforderlich
- Integrität $\neq$ Konsistenz
  - Konsistenz = definierter Zustand des Dateisystems
  - Integrität = überprüfte Daten und Metadaten
- Prüfung von Daten und Metadaten bei großen FS zeitaufwendig
  - $\approx$ 2h für $\approx$ 4 TB Ext4 mit Millionen von Dateien
  - `xfs_repair` kann Minuten/Stunden benötigen
  - Mag wer `fsck.vfat` mit 8+ TB Daten versuchen?

Die leidige Integrität / Konsistenz

- bestimmte Änderungen im Dateisystem erfordern viele Operationen
- Änderungen verlaufen nicht zeitgleich
- Absturz / Stromausfall / Reset
  - Dateisystem im undefinierten Zustand
  - File System Check (*fsck*) erforderlich
- Integrität $\neq$ Konsistenz
  - Konsistenz = definierter Zustand des Dateisystems
  - Integrität = überprüfte Daten und Metadaten
- Prüfung von Daten und Metadaten bei großen FS zeitaufwendig
  - $\approx$ 2h für $\approx$ 4 TB Ext4 mit Millionen von Dateien
  - `xfs_repair` kann Minuten/Stunden benötigen
  - Mag wer `fsck.vfat` mit 8+ TB Daten versuchen?

Strategien für Integrität / Konsistenz

- Soft Updates (*BSD)
  - asynchrones Schreiben von Metadaten
  - Schreiboperation hält DS immer konsistent
  - Garbage Collection nach Absturz (`fsck` und Suche nach *orphaned inodes*)
- Journal
  - alle Änderungen werden im Journal vorab geschrieben
  - Änderungen werden nach Ausführung im Journal markiert
  - nach Absturz sind Änderungen entweder
    - korrekt ausgeführt oder
    - nicht durchgeführt (*not replayed*)
- Copy On Write (COW)
  - Änderungen werden nur an Kopien von Daten vorgenommen

```
1 string x(''Hello'');
2 string y = x;
3 y += '', World!'';
```

  - COW erzeugt daher stetig neue Versionen
  - nach Absturz wird letzte konsistente Version benutzt

Strategien für Integrität / Konsistenz

- Soft Updates (*BSD)
  - asynchrones Schreiben von Metadaten
  - Schreiboperation hält DS immer konsistent
  - Garbage Collection nach Absturz (`fsck` und Suche nach *orphaned inodes*)
- Journal
  - alle Änderungen werden im Journal vorab geschrieben
  - Änderungen werden nach Ausführung im Journal markiert
  - nach Absturz sind Änderungen entweder
    - korrekt ausgeführt oder
    - nicht durchgeführt (*not replayed*)
- Copy On Write (COW)
  - Änderungen werden nur an Kopien von Daten vorgenommen

```
1 string x(''Hello'');
2 string y = x;
3 y += '', World!'';
```

  - COW erzeugt daher stetig neue Versionen
  - nach Absturz wird letzte konsistente Version benutzt

Strategien für Integrität / Konsistenz

- Soft Updates (*BSD)
  - asynchrones Schreiben von Metadaten
  - Schreiboperation hält DS immer konsistent
  - Garbage Collection nach Absturz (`fsck` und Suche nach *orphaned inodes*)
- Journal
  - alle Änderungen werden im Journal vorab geschrieben
  - Änderungen werden nach Ausführung im Journal markiert
  - nach Absturz sind Änderungen entweder
    - korrekt ausgeführt oder
    - nicht durchgeführt (*not replayed*)
- Copy On Write (COW)
  - Änderungen werden nur an Kopien von Daten vorgenommen

```
1 string x(''Hello'');
2 string y = x;
3 y += '', World!'';
```

  - COW erzeugt daher stetig neue Versionen
  - nach Absturz wird letzte konsistente Version benutzt

Strategien für Integrität / Konsistenz

- Soft Updates (*BSD)
  - asynchrones Schreiben von Metadaten
  - Schreiboperation hält DS immer konsistent
  - Garbage Collection nach Absturz (`fsck` und Suche nach *orphaned inodes*)
- Journal
  - alle Änderungen werden im Journal vorab geschrieben
  - Änderungen werden nach Ausführung im Journal markiert
  - nach Absturz sind Änderungen entweder
    - korrekt ausgeführt oder
    - nicht durchgeführt (*not replayed*)
- Copy On Write (COW)
  - Änderungen werden nur an Kopien von Daten vorgenommen
    1. `string x(''Hello'');`
    2. `string y = x;`
    3. `y += '', World!'';`
  - COW erzeugt daher stetig neue Versionen
  - nach Absturz wird letzte konsistente Version benutzt

Ext2/Ext3

- Ext2 ist Weiterentwicklung von Ext
  - Einführung Januar 1994
  - Ziel: Ablösung von Minix & Ext, Schaffung eines „Linux-FS“
  - kein Journal, Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GiB und 2 TiB
  - maximale Größe zwischen 2 und 32 TiB
- Ext3 ist Weiterentwicklung von Ext2
  - Einführung November 2001
  - Journal, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GB und 2 TiB
  - maximale Größe zwischen 2 und 16 TiB

Ext2/Ext3

- Ext2 ist Weiterentwicklung von Ext
  - Einführung Januar 1994
  - Ziel: Ablösung von Minix & Ext, Schaffung eines „Linux-FS“
  - kein Journal, Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GiB und 2 TiB
  - maximale Größe zwischen 2 und 32 TiB
- Ext3 ist Weiterentwicklung von Ext2
  - Einführung November 2001
  - Journal, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GB und 2 TiB
  - maximale Größe zwischen 2 und 16 TiB

Ext2/Ext3

- Ext2 ist Weiterentwicklung von Ext
  - Einführung Januar 1994
  - Ziel: Ablösung von Minix & Ext, Schaffung eines „Linux-FS“
  - kein Journal, Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GiB und 2 TiB
  - maximale Größe zwischen 2 und 32 TiB
- Ext3 ist Weiterentwicklung von Ext2
  - Einführung November 2001
  - Journal, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - maximale Dateigröße zwischen 16 GB und 2 TiB
  - maximale Größe zwischen 2 und 16 TiB

Ext4

- Ext4 ist Weiterentwicklung von Ext3
  - Einführung Juni 2008 (Entwicklung ab 2006)
  - Journal mit Checksummen, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - Extents, Pre-Allocation, Delayed Allocation, 32.000+ Unterverzeichnisse
  - schnelleres `mkfs.ext4`, Multiblock-Allokator
  - maximale Dateigröße 16 TiB
  - maximale Größe 1 EiB (1.000.000 TiB)
- Ext4 ist eine *temporäre* Maßnahme
  - Ext4 beseitigt Designschwächen von Ext3
  - Ext4 hat Features von anderen Dateisystemen übernommen
  - Ext4 wird bereitgestellt bis besseres Dateisystem einsatzbereit ist

Ext4

- Ext4 ist Weiterentwicklung von Ext3
  - Einführung Juni 2008 (Entwicklung ab 2006)
  - Journal mit Checksummen, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - Extents, Pre-Allocation, Delayed Allocation, 32.000+ Unterverzeichnisse
  - schnelleres `mkfs.ext4`, Multiblock-Allokator
  - maximale Dateigröße 16 TiB
  - maximale Größe 1 EiB (1.000.000 TiB)
- Ext4 ist eine *temporäre* Maßnahme
  - Ext4 beseitigt Designschwächen von Ext3
  - Ext4 hat Features von anderen Dateisystemen übernommen
  - Ext4 wird bereitgestellt bis besseres Dateisystem einsatzbereit ist

Ext4

- Ext4 ist Weiterentwicklung von Ext3
  - Einführung Juni 2008 (Entwicklung ab 2006)
  - Journal mit Checksummen, Index für Verzeichnisse (H-Tree, B-Tree)
  - Bitmap & Tabelle für Blöcke/Metadaten
  - Extents, Pre-Allocation, Delayed Allocation, 32.000+ Unterverzeichnisse
  - schnelleres `mkfs.ext4`, Multiblock-Allokator
  - maximale Dateigröße 16 TiB
  - maximale Größe 1 EiB (1.000.000 TiB)
- Ext4 ist eine *temporäre* Maßnahme
  - Ext4 beseitigt Designschwächen von Ext3
  - Ext4 hat Features von anderen Dateisystemen übernommen
  - Ext4 wird bereitgestellt bis besseres Dateisystem einsatzbereit ist

XFS

- Entwicklung 1993, Einführung 1994 (IRIX) & 2001/2002 (Linux)
- maximale Dateigröße 8 EiB (8.000.000 TiB) - 1 Byte
- maximale Größe 16 EiB (16.000.000 TiB)
- Allocation Groups (interne „Unter-Dateisysteme“)
- Extents, Pre-Allocation, Delayed Allocation
- Direct I/O, garantierte I/O Raten
- Snapshots („Einfrieren“ von Dateisystemen)
- Online Defragmentierung und Vergrößerung
- Backup/Restore Werkzeuge speziell für XFS
- kein `fsck.xfs` (!), aber `xfs_check` & `xfs_repair`
- viele Verbesserungen im 3.x Linuxkern („metadata bottleneck“, Block-/Journal-Sortierung, ...)

XFS

- Entwicklung 1993, Einführung 1994 (IRIX) & 2001/2002 (Linux)
- maximale Dateigröße 8 EiB (8.000.000 TiB) - 1 Byte
- maximale Größe 16 EiB (16.000.000 TiB)
- Allocation Groups (interne „Unter-Dateisysteme“)
- Extents, Pre-Allocation, Delayed Allocation
- Direct I/O, garantierte I/O Raten
- Snapshots („Einfrieren“ von Dateisystemen)
- Online Defragmentierung und Vergrößerung
- Backup/Restore Werkzeuge speziell für XFS
- kein `fsck.xfs` (!), aber `xfs_check` & `xfs_repair`
- viele Verbesserungen im 3.x Linuxkern („metadata bottleneck“, Block-/Journal-Sortierung, ...)

XFS

- Entwicklung 1993, Einführung 1994 (IRIX) & 2001/2002 (Linux)
- maximale Dateigröße 8 EiB (8.000.000 TiB) - 1 Byte
- maximale Größe 16 EiB (16.000.000 TiB)
- Allocation Groups (interne „Unter-Dateisysteme“)
- Extents, Pre-Allocation, Delayed Allocation
- Direct I/O, garantierte I/O Raten
- Snapshots („Einfrieren“ von Dateisystemen)
- Online Defragmentierung und Vergrößerung
- Backup/Restore Werkzeuge speziell für XFS
- kein `fsck.xfs` (!), aber `xfs_check` & `xfs_repair`
- viele Verbesserungen im 3.x Linuxkern („metadata bottleneck“, Block-/Journal-Sortierung, ...)

XFS

- Entwicklung 1993, Einführung 1994 (IRIX) & 2001/2002 (Linux)
- maximale Dateigröße 8 EiB (8.000.000 TiB) - 1 Byte
- maximale Größe 16 EiB (16.000.000 TiB)
- Allocation Groups (interne „Unter-Dateisysteme“)
- Extents, Pre-Allocation, Delayed Allocation
- Direct I/O, garantierte I/O Raten
- Snapshots („Einfrieren“ von Dateisystemen)
- Online Defragmentierung und Vergrößerung
- Backup/Restore Werkzeuge speziell für XFS
- kein `fsck.xfs` (!), aber `xfs_check` & `xfs_repair`
- viele Verbesserungen im 3.x Linuxkern („metadata bottleneck“, Block-/Journal-Sortierung, ...)

XFS - Anmerkungen

- XFS unter Linux® verwendet Code von IRIX®
  - Linux®-Kompatibilitätsschicht
  - größeres Kernelmodul
- XFS versucht wenig Overhead zu produzieren
  - möglichst wenig zwischen Applikation und Blockgerät
  - XFS verwaltet keine Blockgeräte
- XFS ist wartungsarm
  - `xfs_check` und `xfs_repair` selten notwendig
  - XFS schaltet FS bei Hardwarefehlern ab (wie Ext4)
- XFS löscht Dateien mit Nullbytes!
  - falsch: XFS führt Journal *nur für Metadaten*
  - Daten können *sparse* sein bis sie geschrieben sind
  - Konsistenz wichtig, alle Journalling DS haben dieses Ziel!

XFS - Anmerkungen

- XFS unter Linux® verwendet Code von IRIX®
  - Linux®-Kompatibilitätsschicht
  - größeres Kernelmodul
- XFS versucht wenig Overhead zu produzieren
  - möglichst wenig zwischen Applikation und Blockgerät
  - XFS verwaltet keine Blockgeräte
- XFS ist wartungsarm
  - `xfs_check` und `xfs_repair` selten notwendig
  - XFS schaltet FS bei Hardwarefehlern ab (wie Ext4)
- XFS löscht Dateien mit Nullbytes!
  - falsch: XFS führt Journal *nur für Metadaten*
  - Daten können *sparse* sein bis sie geschrieben sind
  - Konsistenz wichtig, alle Journalling DS haben dieses Ziel!

XFS - Anmerkungen

- XFS unter Linux® verwendet Code von IRIX®
  - Linux®-Kompatibilitätsschicht
  - größeres Kernelmodul
- XFS versucht wenig Overhead zu produzieren
  - möglichst wenig zwischen Applikation und Blockgerät
  - XFS verwaltet keine Blockgeräte
- XFS ist wartungsarm
  - `xfs_check` und `xfs_repair` selten notwendig
  - XFS schaltet FS bei Hardwarefehlern ab (wie Ext4)
- XFS löscht Dateien mit Nullbytes!
  - falsch: XFS führt Journal *nur für Metadaten*
  - Daten können *sparse* sein bis sie geschrieben sind
  - Konsistenz wichtig, alle Journalling DS haben dieses Ziel!

XFS - Anmerkungen

- XFS unter Linux® verwendet Code von IRIX®
  - Linux®-Kompatibilitätsschicht
  - größeres Kernelmodul
- XFS versucht wenig Overhead zu produzieren
  - möglichst wenig zwischen Applikation und Blockgerät
  - XFS verwaltet keine Blockgeräte
- XFS ist wartungsarm
  - `xfs_check` und `xfs_repair` selten notwendig
  - XFS schaltet FS bei Hardwarefehlern ab (wie Ext4)
- XFS löscht Dateien mit Nullbytes!
  - falsch: XFS führt Journal *nur für Metadaten*
  - Daten können *sparse* sein bis sie geschrieben sind
  - Konsistenz wichtig, alle Journalling DS haben dieses Ziel!

XFS - Anmerkungen

- XFS unter Linux® verwendet Code von IRIX®
  - Linux®-Kompatibilitätsschicht
  - größeres Kernelmodul
- XFS versucht wenig Overhead zu produzieren
  - möglichst wenig zwischen Applikation und Blockgerät
  - XFS verwaltet keine Blockgeräte
- XFS ist wartungsarm
  - `xfs_check` und `xfs_repair` selten notwendig
  - XFS schaltet FS bei Hardwarefehlern ab (wie Ext4)
- XFS löscht Dateien mit Nullbytes!
  - falsch: XFS führt Journal *nur für Metadaten*
  - Daten können *sparse* sein bis sie geschrieben sind
  - Konsistenz wichtig, alle Journalling DS haben dieses Ziel!

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs - „ButterFS“

- Entwicklung 2007, „Einführung“ 2012
- inspiriert von Ext4, ZFS und Reiser4; basiert auf B-Trees
- maximale Dateigröße & Größe 16 EiB (16.000.000 TiB)
- Copy On Write (COW), *kein* Journal
- Extents, Pre-Allocation, Delayed Allocation
- Online Defragmentierung, Vergrößerung und Verkleinerung
- Checksummen (CRC-32C) für Daten *und* Metadaten
- Snapshots und Subvolumes („Unter-Dateisysteme“, spart LVM)
- Multi-Device Support & Objekt RAID0/RAID1/RAID10
- Transparente Kompression (gzip/LZO)
- Offline Konvertierung von Ext3/Ext4 in Btrfs (und zurück)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

Btrfs Komponenten

- B-Trees zur internen Organisation
- Chunks zur Verteilung von Speicherplatz
- Extents zum Allokieren von Dateispeicherplatz
- Checksummen für Daten und Metadaten
- Referenzen auf Kopien
  - COW referenziert Objekte
  - Snapshots arbeiten mit Referenzen
- API für direkte Abfragen an die B-Trees (vorgesehen)

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

B-Trees in Btrfs

- Tree Root B-Tree
  Verzeichnis aller B-Tree Wurzeln
- Chunk B-Tree
  Verzeichnis aller Chunks mit Zugehörigkeit
- Extent B-Tree
  Verzeichnis aller Extents und Referenzen
- Checksum B-Tree
  Verzeichnis aller Checksummen
- Dateisystem B-Tree
  Verzeichnis der eigentlichen Dateisysteminformationen, der Dateien und der Verzeichnisse

Blockgeräte

- Btrfs kann mit allen Blockgeräten kombiniert werden
  - Hardware RAID
  - `sd`, `hd`, `vd`, ...
  - MD Geräte (Software RAID*n*)
  - DM / DM Crypt (LVM & LUKS)
  - Loopback
  - AoE, NBD, RADOS, ...
- Btrfs kann selbst Blockgeräte „verwalten“
  - RAID0/RAID1/RAID10 implementiert (auf Objektebene)
  - RAID5/RAID6 geplant
  - Btrfs kann Defekte selbst durch Kopien reparieren

Blockgeräte

- Btrfs kann mit allen Blockgeräten kombiniert werden
  - Hardware RAID
  - `sd`, `hd`, `vd`, ...
  - MD Geräte (Software RAID*n*)
  - DM / DM Crypt (LVM & LUKS)
  - Loopback
  - AoE, NBD, RADOS, ...
- Btrfs kann selbst Blockgeräte „verwalten“
  - RAID0/RAID1/RAID10 implementiert (auf Objektebene)
  - RAID5/RAID6 geplant
  - Btrfs kann Defekte selbst durch Kopien reparieren

Blockgeräte

- Btrfs kann mit allen Blockgeräten kombiniert werden
  - Hardware RAID
  - `sd`, `hd`, `vd`, ...
  - MD Geräte (Software RAID*n*)
  - DM / DM Crypt (LVM & LUKS)
  - Loopback
  - AoE, NBD, RADOS, ...
- Btrfs kann selbst Blockgeräte „verwalten“
  - RAID0/RAID1/RAID10 implementiert (auf Objektebene)
  - RAID5/RAID6 geplant
  - Btrfs kann Defekte selbst durch Kopien reparieren

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10)
  (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10)
  (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10)
  (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10) (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10)
  (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing

- Problem: *silent corruption / bit rot*
  - Verfall von Speichermedien
  - Ansammlung von „bad blocks“
- Btrfs führt Checksummen über alle Daten/Metadaten
- Scrubbing vergleicht Objekte on-disk mit Checksummen
  - impliziert Lesen aller Objekte
  - Defekte können lokalisiert werden
- Btrfs repariert Schäden durch Kopien (bei RAID1/RAID10)
  (bei RAID0 weiß Btrfs nur was kaputt ist)
- Scrubbing verläuft (noch) nicht automatisch

Scrubbing (2)

- Btrfs anweisen alle Daten/Metadaten zu prüfen:
- `btrfs scrub start /srv`
- `btrfs scrub status /srv`

```
scrub status for 228efd5f-be62-4970-93c1-d219bc7d32bd
 scrub started at ... 00:47:10 2012 and finished after 1752 seconds
 total bytes scrubbed: 81.99GB with 0 errors
```

- Scrub erkennt Fehler, muß aber alle Daten lesen!
- Scrubbing kann daher zeitintensiv sein

Scrubbing (2)

- Btrfs anweisen alle Daten/Metadaten zu prüfen:
- `btrfs scrub start /srv`
- `btrfs scrub status /srv`

```
scrub status for 228efd5f-be62-4970-93c1-d219bc7d32bd
 scrub started at ... 00:47:10 2012 and finished after 1752 seconds
 total bytes scrubbed: 81.99GB with 0 errors
```

- Scrub erkennt Fehler, muß aber alle Daten lesen!
- Scrubbing kann daher zeitintensiv sein

Scrubbing (2)

- Btrfs anweisen alle Daten/Metadaten zu prüfen:
- `btrfs scrub start /srv`
- `btrfs scrub status /srv`

```
scrub status for 228efd5f-be62-4970-93c1-d219bc7d32bd
 scrub started at ... 00:47:10 2012 and finished after 1752 seconds
 total bytes scrubbed: 81.99GB with 0 errors
```

- Scrub erkennt Fehler, muß aber alle Daten lesen!
- Scrubbing kann daher zeitintensiv sein

Scrubbing (2)

- Btrfs anweisen alle Daten/Metadaten zu prüfen:
- `btrfs scrub start /srv`
- `btrfs scrub status /srv`

```
scrub status for 228efd5f-be62-4970-93c1-d219bc7d32bd
 scrub started at ... 00:47:10 2012 and finished after 1752 seconds
 total bytes scrubbed: 81.99GB with 0 errors
```

- Scrub erkennt Fehler, muß aber alle Daten lesen!
- Scrubbing kann daher zeitintensiv sein

Scrubbing (2)

- Btrfs anweisen alle Daten/Metadaten zu prüfen:
- `btrfs scrub start /srv`
- `btrfs scrub status /srv`

```
scrub status for 228efd5f-be62-4970-93c1-d219bc7d32bd
 scrub started at ... 00:47:10 2012 and finished after 1752 seconds
 total bytes scrubbed: 81.99GB with 0 errors
```

- Scrub erkennt Fehler, muß aber alle Daten lesen!
- Scrubbing kann daher zeitintensiv sein

Btrfs in Aktion - mkfs

- „einfaches“ Btrfs auf einem Datenträger und Vergrößerung
  - `mkfs.btrfs /dev/sdc1`
  - `mount -t btrfs -o autodefrag /dev/sdc1 /srv`
  - `btrfs device add /dev/sdd1 /srv`
  - `btrfs filesystem balance /srv`
- Btrfs mit RAID1 für Metadaten und Daten
  - `mkfs.btrfs -m raid1 -d raid1 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache /dev/sdc1 /srv`
- Btrfs mit RAID1 für Metadaten und RAID0 für Daten
  - `mkfs.btrfs -m raid1 -d raid0 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache,autodefrag /dev/sdc1 /srv`

Btrfs in Aktion - mkfs

- „einfaches“ Btrfs auf einem Datenträger und Vergrößerung
  - `mkfs.btrfs /dev/sdc1`
  - `mount -t btrfs -o autodefrag /dev/sdc1 /srv`
  - `btrfs device add /dev/sdd1 /srv`
  - `btrfs filesystem balance /srv`
- Btrfs mit RAID1 für Metadaten und Daten
  - `mkfs.btrfs -m raid1 -d raid1 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache /dev/sdc1 /srv`
- Btrfs mit RAID1 für Metadaten und RAID0 für Daten
  - `mkfs.btrfs -m raid1 -d raid0 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache,autodefrag /dev/sdc1 /srv`

Btrfs in Aktion - mkfs

- „einfaches“ Btrfs auf einem Datenträger und Vergrößerung
  - `mkfs.btrfs /dev/sdc1`
  - `mount -t btrfs -o autodefrag /dev/sdc1 /srv`
  - `btrfs device add /dev/sdd1 /srv`
  - `btrfs filesystem balance /srv`
- Btrfs mit RAID1 für Metadaten und Daten
  - `mkfs.btrfs -m raid1 -d raid1 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache /dev/sdc1 /srv`
- Btrfs mit RAID1 für Metadaten und RAID0 für Daten
  - `mkfs.btrfs -m raid1 -d raid0 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache,autodefrag /dev/sdc1 /srv`

Btrfs in Aktion - mkfs

- „einfaches“ Btrfs auf einem Datenträger und Vergrößerung
  - `mkfs.btrfs /dev/sdc1`
  - `mount -t btrfs -o autodefrag /dev/sdc1 /srv`
  - `btrfs device add /dev/sdd1 /srv`
  - `btrfs filesystem balance /srv`
- Btrfs mit RAID1 für Metadaten und Daten
  - `mkfs.btrfs -m raid1 -d raid1 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache /dev/sdc1 /srv`
- Btrfs mit RAID1 für Metadaten und RAID0 für Daten
  - `mkfs.btrfs -m raid1 -d raid0 /dev/sdc1 /dev/sdd1`
  - `mount -t btrfs -o space_cache,inode_cache,autodefrag /dev/sdc1 /srv`

Btrfs Multi-Disk - Balancing

- Btrfs' RAID-Level sind objektorientiert
  - Btrfs legt Kopien von Datenstrukturen an
  - keine 1 : 1 Blockkopie wie bei MD
  - kein sinnloses Kopieren von leeren Blöcken (wie bei LVM)
- Btrfs verteilt Kopien selbstständig
- `btrfs filesystem balance` kann Kopien bei
  - Wechseln der RAID Level oder
  - Ändern von Blockgeräten

  neu verteilen.

Btrfs Multi-Disk - Balancing

- Btrfs' RAID-Level sind objektorientiert
  - Btrfs legt Kopien von Datenstrukturen an
  - keine 1 : 1 Blockkopie wie bei MD
  - kein sinnloses Kopieren von leeren Blöcken (wie bei LVM)
- Btrfs verteilt Kopien selbstständig
- `btrfs filesystem balance` kann Kopien bei
  - Wechseln der RAID Level oder
  - Ändern von Blockgeräten

  neu verteilen.

Btrfs Multi-Disk - Balancing

- Btrfs' RAID-Level sind objektorientiert
  - Btrfs legt Kopien von Datenstrukturen an
  - keine 1 : 1 Blockkopie wie bei MD
  - kein sinnloses Kopieren von leeren Blöcken (wie bei LVM)
- Btrfs verteilt Kopien selbstständig
- `btrfs filesystem balance` kann Kopien bei
  - Wechseln der RAID Level oder
  - Ändern von Blockgeräten

  neu verteilen.

Btrfs Multi-Disk - Balancing

- Btrfs' RAID-Level sind objektorientiert
  - Btrfs legt Kopien von Datenstrukturen an
  - keine 1 : 1 Blockkopie wie bei MD
  - kein sinnloses Kopieren von leeren Blöcken (wie bei LVM)
- Btrfs verteilt Kopien selbstständig
- `btrfs filesystem balance` kann Kopien bei
  - Wechseln der RAID Level oder
  - Ändern von Blockgeräten

  neu verteilen.

Btrfs Show & df

- `df -h` sagt: `/dev/sda6 813G 191G 622G 24% /home`
- `btrfs filesystem df /home` sagt:

```
Data, RAID1: total=96.00GB, used=94.80GB
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=20.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=1.00GB, used=383.64MB
Metadata: total=8.00MB, used=0.00
```

- `df` lügt, `btrfs filesystem df` gibt genauere Auskunft (`btrfs filesystem show` zeigt Disks)
- Snapshots, Extents und COW erschweren Auskunft nach freiem Speicher

Btrfs Show & df

- `df -h` sagt: `/dev/sda6 813G 191G 622G 24% /home`
- `btrfs filesystem df /home` sagt:

```
Data, RAID1: total=96.00GB, used=94.80GB
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=20.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=1.00GB, used=383.64MB
Metadata: total=8.00MB, used=0.00
```

- `df` lügt, `btrfs filesystem df` gibt genauere Auskunft (`btrfs filesystem show` zeigt Disks)
- Snapshots, Extents und COW erschweren Auskunft nach freiem Speicher

Btrfs Show & df

- `df -h` sagt: `/dev/sda6 813G 191G 622G 24% /home`
- `btrfs filesystem df /home` sagt:

```
Data, RAID1: total=96.00GB, used=94.80GB
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=20.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=1.00GB, used=383.64MB
Metadata: total=8.00MB, used=0.00
```

- `df` lügt, `btrfs filesystem df` gibt genauere Auskunft (`btrfs filesystem show` zeigt Disks)
- Snapshots, Extents und COW erschweren Auskunft nach freiem Speicher

Btrfs Show & df

- `df -h` sagt: `/dev/sda6 813G 191G 622G 24% /home`
- `btrfs filesystem df /home` sagt:

```
Data, RAID1: total=96.00GB, used=94.80GB
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=20.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=1.00GB, used=383.64MB
Metadata: total=8.00MB, used=0.00
```

- `df` lügt, `btrfs filesystem df` gibt genauere Auskunft (`btrfs filesystem show` zeigt Disks)
- Snapshots, Extents und COW erschweren Auskunft nach freiem Speicher

Btrfs Dynamic Inode Allocation

- Btrfs trennt Metadaten und Daten
  - Metadaten werden in 256 MiB Stücken belegt
  - Daten werden in 1 GiB Stücken belegt
  - Bereiche mischen sich nicht
- Problem bei kleinen Datenträgern
  - Problem: Datenträger voll, obwohl Blöcke frei sind
  - Lösung: Metadaten und Daten mischen
  - `mkfs.btrfs --mixed /dev/sdb1`
- Problem bei Multi-Disk Btrfs
  - RAID1 mit verschieden großen Disks
  - Lösung 1: zusätzliche Disks & Rebalance verwenden
  - Lösung 2: Mixed Mode

Btrfs Dynamic Inode Allocation

- Btrfs trennt Metadaten und Daten
  - Metadaten werden in 256 MiB Stücken belegt
  - Daten werden in 1 GiB Stücken belegt
  - Bereiche mischen sich nicht
- Problem bei kleinen Datenträgern
  - Problem: Datenträger voll, obwohl Blöcke frei sind
  - Lösung: Metadaten und Daten mischen
  - `mkfs.btrfs --mixed /dev/sdb1`
- Problem bei Multi-Disk Btrfs
  - RAID1 mit verschieden großen Disks
  - Lösung 1: zusätzliche Disks & Rebalance verwenden
  - Lösung 2: Mixed Mode

Btrfs Dynamic Inode Allocation

- Btrfs trennt Metadaten und Daten
  - Metadaten werden in 256 MiB Stücken belegt
  - Daten werden in 1 GiB Stücken belegt
  - Bereiche mischen sich nicht
- Problem bei kleinen Datenträgern
  - Problem: Datenträger voll, obwohl Blöcke frei sind
  - Lösung: Metadaten und Daten mischen
  - `mkfs.btrfs --mixed /dev/sdb1`
- Problem bei Multi-Disk Btrfs
  - RAID1 mit verschieden großen Disks
  - Lösung 1: zusätzliche Disks & Rebalance verwenden
  - Lösung 2: Mixed Mode

Btrfs Dynamic Inode Allocation

- Btrfs trennt Metadaten und Daten
  - Metadaten werden in 256 MiB Stücken belegt
  - Daten werden in 1 GiB Stücken belegt
  - Bereiche mischen sich nicht
- Problem bei kleinen Datenträgern
  - Problem: Datenträger voll, obwohl Blöcke frei sind
  - Lösung: Metadaten und Daten mischen
  - `mkfs.btrfs --mixed /dev/sdb1`
- Problem bei Multi-Disk Btrfs
  - RAID1 mit verschieden großen Disks
  - Lösung 1: zusätzliche Disks & Rebalance verwenden
  - Lösung 2: Mixed Mode

Fehlermeldungen Btrfs

```
btrfs: i/o error at logical 42648674304 on dev /dev/sdc3,
 sector 25651856, root 5, inode 313545, offset 220033024,
 length 4096, links 1 (path: squid/spool/10/D5/0010D575)
btrfs: i/o error at logical 313913802752 on dev /dev/sdc3,
 sector 210295872, root 5, inode 3048173, offset 98304,
 length 4096, links 1
 (path: mfs/bah/2D/chunk_000000000003CB2D_00000001.mfs)
btrfs: i/o error at logical 313913806848 on dev /dev/sdc3,
 sector 210295880, root 5, inode 3048173, offset 102400,
 length 4096, links 1
 (path: mfs/bah/2D/chunk_000000000003CB2D_00000001.mfs)
```

Fehlermeldungen Btrfs

```
btrfs: i/o error at logical 42648674304 on dev /dev/sdc3,
 sector 25651856, root 5, inode 313545, offset 220033024,
 length 4096, links 1 (path: squid/spool/10/D5/0010D575)
btrfs: i/o error at logical 313913802752 on dev /dev/sdc3,
 sector 210295872, root 5, inode 3048173, offset 98304,
 length 4096, links 1
 (path: mfs/bah/2D/chunk_000000000003CB2D_00000001.mfs)
btrfs: i/o error at logical 313913806848 on dev /dev/sdc3,
 sector 210295880, root 5, inode 3048173, offset 102400,
 length 4096, links 1
 (path: mfs/bah/2D/chunk_000000000003CB2D_00000001.mfs)
```

Ist Btrfs reif für Produktion?

- Status *experimental* im Linux Source Code
- Oracle® und Novell setzen Btrfs ab 2012 produktiv ein
- `btrfsck` ist bald fertig (markiert mit „donteveruse“)
  - `btrfsck` repariert noch nichts
  - COW und Transaktionen reichen meist aus
  - Mount Option `recovery` (ab Linux 3.2)
  - `restore` Tool „evakuiert“ Daten aus Btrfs
  - `btrfsck` hat schon ausgeholfen
  - **Backups, Backups, Backups!**
- Btrfs muss noch getestet werden
- Btrfs läßt sich schon einsetzen, Abhängig von den Workloads . . .

Ist Btrfs reif für Produktion?

- Status *experimental* im Linux Source Code
- Oracle® und Novell setzen Btrfs ab 2012 produktiv ein
- `btrfsck` ist bald fertig (markiert mit „donteveruse“)
  - `btrfsck` repariert noch nichts
  - COW und Transaktionen reichen meist aus
  - Mount Option `recovery` (ab Linux 3.2)
  - `restore` Tool „evakuiert“ Daten aus Btrfs
  - `btrfsck` hat schon ausgeholfen
  - **Backups, Backups, Backups!**
- Btrfs muss noch getestet werden
- Btrfs läßt sich schon einsetzen, Abhängig von den Workloads . . .

Ist Btrfs reif für Produktion?

- Status *experimental* im Linux Source Code
- Oracle® und Novell setzen Btrfs ab 2012 produktiv ein
- `btrfsck` ist bald fertig (markiert mit „donteveruse“)
  - `btrfsck` repariert noch nichts
  - COW und Transaktionen reichen meist aus
  - Mount Option `recovery` (ab Linux 3.2)
  - `restore` Tool „evakuiert“ Daten aus Btrfs
  - `btrfsck` hat schon ausgeholfen
  - **Backups, Backups, Backups!**
- Btrfs muss noch getestet werden
- Btrfs läßt sich schon einsetzen, Abhängig von den Workloads . . .

Ist Btrfs reif für Produktion?

- Status *experimental* im Linux Source Code
- Oracle® und Novell setzen Btrfs ab 2012 produktiv ein
- `btrfsck` ist bald fertig (markiert mit „donteveruse“)
  - `btrfsck` repariert noch nichts
  - COW und Transaktionen reichen meist aus
  - Mount Option `recovery` (ab Linux 3.2)
  - `restore` Tool „evakuiert“ Daten aus Btrfs
  - `btrfsck` hat schon ausgeholfen
  - **Backups, Backups, Backups!**
- Btrfs muss noch getestet werden
- Btrfs läßt sich schon einsetzen, Abhängig von den Workloads . . .

Ist Btrfs reif für Produktion?

- Status *experimental* im Linux Source Code
- Oracle® und Novell setzen Btrfs ab 2012 produktiv ein
- `btrfsck` ist bald fertig (markiert mit „donteveruse“)
  - `btrfsck` repariert noch nichts
  - COW und Transaktionen reichen meist aus
  - Mount Option `recovery` (ab Linux 3.2)
  - `restore` Tool „evakuiert“ Daten aus Btrfs
  - `btrfsck` hat schon ausgeholfen
  - **Backups, Backups, Backups!**
- Btrfs muss noch getestet werden
- Btrfs läßt sich schon einsetzen, Abhängig von den Workloads . . .

Die leidige Religion - Workloads

- Welche Aufgabe(n) soll das Dateisystem lösen?
  - große Dateien (Archive, Videos), ...
  - große Dateien mit Löchern (virtuelle Maschinen), ...
  - viele kleine Dateien, ...
  - viele Metadaten, ...
  - von allem etwas, ...
- Welche Anforderungen an die Performance gibt es?
- In welche Umgebung wird das Dateisystem eingesetzt?
  - Hardware, Virtualisierung, ...
  - Software (Linuxkern), Betriebssystem, ...
  - I/O Subsystem, ...
- Wer wird das Dateisystem administrieren?
  - „FS Tools muß man ja nur im Notfall beherrschen.“
  - Alle Dateisysteme benötigen Wartung!

Die leidige Religion - Workloads

- Welche Aufgabe(n) soll das Dateisystem lösen?
  - große Dateien (Archive, Videos), ...
  - große Dateien mit Löchern (virtuelle Maschinen), ...
  - viele kleine Dateien, ...
  - viele Metadaten, ...
  - von allem etwas, ...
- Welche Anforderungen an die Performance gibt es?
- In welche Umgebung wird das Dateisystem eingesetzt?
  - Hardware, Virtualisierung, ...
  - Software (Linuxkern), Betriebssystem, ...
  - I/O Subsystem, ...
- Wer wird das Dateisystem administrieren?
  - „FS Tools muß man ja nur im Notfall beherrschen.“
  - Alle Dateisysteme benötigen Wartung!

Die leidige Religion - Workloads

- Welche Aufgabe(n) soll das Dateisystem lösen?
  - große Dateien (Archive, Videos), ...
  - große Dateien mit Löchern (virtuelle Maschinen), ...
  - viele kleine Dateien, ...
  - viele Metadaten, ...
  - von allem etwas, ...
- Welche Anforderungen an die Performance gibt es?
- In welche Umgebung wird das Dateisystem eingesetzt?
  - Hardware, Virtualisierung, ...
  - Software (Linuxkern), Betriebssystem, ...
  - I/O Subsystem, ...
- Wer wird das Dateisystem administrieren?
  - „FS Tools muß man ja nur im Notfall beherrschen.“
  - Alle Dateisysteme benötigen Wartung!

Die leidige Religion - Workloads

- Welche Aufgabe(n) soll das Dateisystem lösen?
  - große Dateien (Archive, Videos), ...
  - große Dateien mit Löchern (virtuelle Maschinen), ...
  - viele kleine Dateien, ...
  - viele Metadaten, ...
  - von allem etwas, ...
- Welche Anforderungen an die Performance gibt es?
- In welche Umgebung wird das Dateisystem eingesetzt?
  - Hardware, Virtualisierung, ...
  - Software (Linuxkern), Betriebssystem, ...
  - I/O Subsystem, ...
- Wer wird das Dateisystem administrieren?
  - „FS Tools muß man ja nur im Notfall beherrschen.“
  - Alle Dateisysteme benötigen Wartung!

Die leidige Religion - Workloads

- Welche Aufgabe(n) soll das Dateisystem lösen?
  - große Dateien (Archive, Videos), ...
  - große Dateien mit Löchern (virtuelle Maschinen), ...
  - viele kleine Dateien, ...
  - viele Metadaten, ...
  - von allem etwas, ...
- Welche Anforderungen an die Performance gibt es?
- In welche Umgebung wird das Dateisystem eingesetzt?
  - Hardware, Virtualisierung, ...
  - Software (Linuxkern), Betriebssystem, ...
  - I/O Subsystem, ...
- Wer wird das Dateisystem administrieren?
  - „FS Tools muß man ja nur im Notfall beherrschen.“
  - Alle Dateisysteme benötigen Wartung!

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

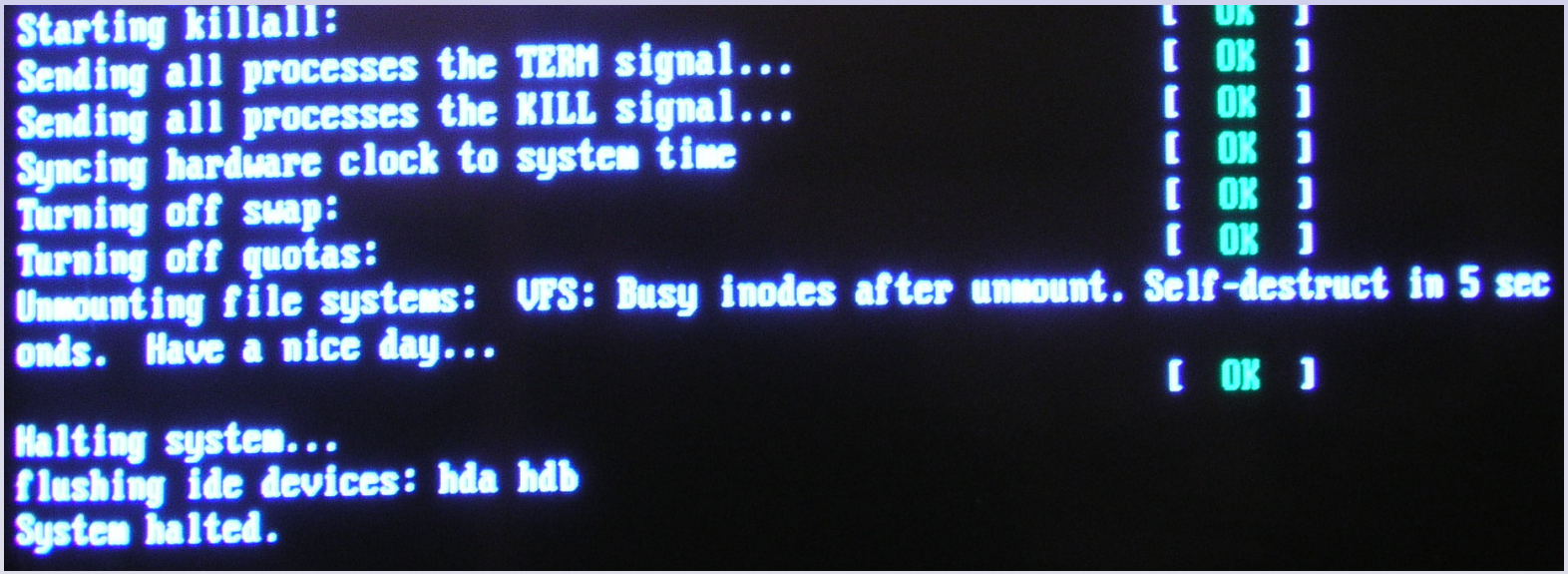
Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Kurze subjektive Rezepte

- gewohnte Umgebung $\longrightarrow$ gewohntes Dateisystem
- ungewohnte Umgebung $\longrightarrow$ Trainieren! Simulieren! Lernen!
- kleine Datenträger, USB Sticks $\longrightarrow$ Ext*n* ohne Journal, Btrfs
- viele oder große Daten $\longrightarrow$ XFS
- viele kleine und große Dateien $\longrightarrow$ XFS, Ext4, Btrfs
- VM Images $\longrightarrow$ XFS oder Ext4
  - Qemu VM mit COW Image auf COW Btrfs $\longrightarrow \approx$ 1-4 MB/s
  - Qemu VM mit Raw Image auf COW Btrfs $\longrightarrow \approx$ 50-65 MB/s
- Datenbanken $\longrightarrow$ XFS oder Ext4
- Snapshots, flexible Multi-Volumes, Ceph $\longrightarrow$ Btrfs
- SSDs $\longrightarrow$ XFS, Ext4 oder Btrfs

Noch Fragen?


Starting killall:
Sending all processes the TERM signal...
Sending all processes the KILL signal...
Syncing hardware clock to system time
Turning off swap:
Turning off quotas:
Unmounting file systems: VFS: Busy inodes after unmount. Self-destruct in 5 seconds. Have a nice day...
Halting system...
flushing ide devices: hda hdb
System halted.
[OK]

Quellen

- A short history of btrfs
- Btrfs Wiki (**nicht** auf `kernel.org`!)
- Btrfs To Go Production-Ready In Oracle Linux
- Ext4 (and Ext2/Ext3) Wiki
- JFS
- Oracle's Chris Mason Talks Up Btrfs Features
- XFS Wiki
- XFS Developer Takes Shots At Btrfs, EXT4

Über dieses Dokument

- Autor: René Pfeiffer
- Erstellt mit LaTeX und LaTeX Beamer Class
- Dokumentensammlung unter
  http://web.luchs.at/information/docs.php