Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bericht

Features von modernen Dateisystemen

vorgelegt von

Lennart Lundelius

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Proseminar Speicher- und Dateisysteme

Studiengang: Wirtschaftsinformatik
Matrikelnummer: 6931344

Betreuer: Eugen Betke

Hamburg, 31.03.2019

Inhaltsverzeichnis

1 Einleitung

Um den heutigen Schutzzielen eines konsistenten und fehlerfreien Dateisystems nachzukommen, bedarf es mehrerer Mechanismen die im Einklang miteinander funktionieren müssen.
Zusätzlich gestaltet sich die Speicherverwaltung unflexibel und es fehlt an Funktionen die den Komfort der Administration und Verwaltung steigern.

Im folgenden Bericht wird dazu ein Überblick über die Funktionsweise von aktuellen Dateisystemen geschaffen. Dazu werden die Grundfunktionen herausgearbeitet und deren Grenzen erläutert.

Im zweiten Teil der Arbeit werden bestimmte Features von moderneren Dateisystemen erläutert. Dabei wird die Funktionalität erläutert und die wesentlichen Unterschiede zu aktuellen Dateisystemen genannt.

Im letzten Teil wird ein Ausblicke auf zukünftige Dateisysteme gegeben und wie sie sich von den modernen absetzen.

Im Fazit werden die wichtigsten Punkte zusammengefasst und abschließend aufgeführt.

2 Moderne Dateisysteme

2.1 Allgemein

Ein Dateisystem organisiert und strukturiert für das Betriebssystem die Dateien und Verzeichnisse und deren Metadaten.

Die Bezeichnung von Dateien wird in zwei Bereiche unterteilt, den Namen und die Namenserweiterung. Bei der Datei *SouthAfrica.jpg* wäre *SouthAfrica* der Name und die Erweiterung *jpg*. *JPG* steht dabei für ein Bildformat. Getrennt wird der Name von der Erweiterung mit einem Punkt.

Die Verzeichnisstruktur beginnt im Wurzelverzeichnis und baut sich je nach Betriebssystem in verschiedenen Baumstrukturen (z.B. Hierarchisch) auf. Das Wurzelverzeichnis kann pro Ebene weitere Unterverzeichnisse oder Dateien besitzen. [1]

2.2 Layout

Rein technisch betrachtet dient das Dateisystem damit als Schnittstelle zwischen Betriebssystem und den Partitionen.
Folgend abgebildet ist ein Layout eines Speichermediums. Der *Master Boot Record* (MBR) lädt beim Starten des Computers die erste aktiv markierte Partition aus der *Partitionstabelle*. Der abgebildete Aufbau von dem Dateisystem kann zu anderen Dateisystemen variieren.[2]

- Der *Boot block* führt das installierte Betriebssystem aus.
- Der *Superblock* enthält die Schlüsselparameter.
- Der *Free space mgmt* ist als Bitmap für die freie Speicherverwaltung zuständig.
- Die *I-nodes* (Index-Nodes) enthalten Informationen über alle Dateien.
- Der *Root dir* gibt das Wurzelverzeichnis an.
- Die *Files and directories* enthalten alle Dateien und Unterverzeichnisse.

[1] Quelle: [Tan16a]
[2] Quelle: [Tan16b]

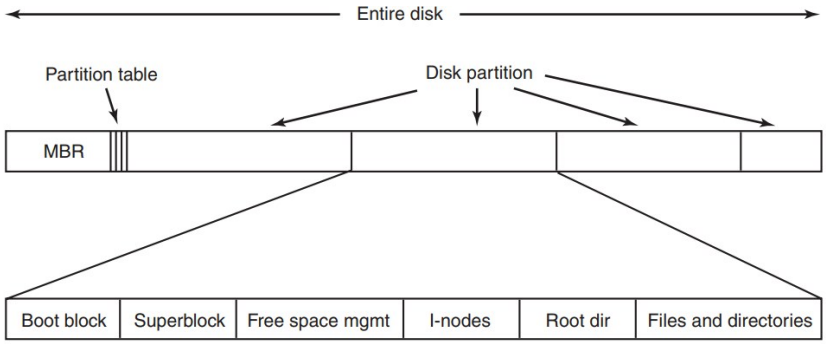

Abbildung 2.1: [Tan16c]

2.3 Sicherung

Es gibt zwei grundlegende Konzepte der Sicherung.

1. Physische Sicherung

   Bei der physischen Sicherung wird das ganze Speichermedium 1-zu-1 gesichert. Unnötigerweise werden dabei auch alle freien und defekte Blöcke übernommen.

2. Logische Sicherung

   Hier besteht zusätzlich die Möglichkeit Dateien und Verzeichnisse auszuklammern oder nach der ersten Vollsicherung eine inkrementelle Sicherung durchzuführen. Dabei wird nur das Delta, also alle geänderten Dateien und Verzeichnisse zur vorherigen Sicherung übernommen.

Dateisysteme setzen dazu auf standardisierte Sicherungsalgorithmen die in vier Phasen eine Sicherung durchführen. Dazu indiziert er als erstes eine Bitmap mit den Index-Nodes Nummern ab dem Wurzelverzeichnis und durchläuft folgende Phasen über die Bitmap.[3]

1. Rekursive Markierung aller modifizierten Dateien und Verzeichnisse.
2. Demarkieren der Verzeichnisse, die keine modifizierten Dateien oder Verzeichnisse unter sich haben.
3. Sicherung aller markierten Verzeichnisse.
4. Sicherung aller markierten Dateien.

[3]Quelle: [Tan16d]

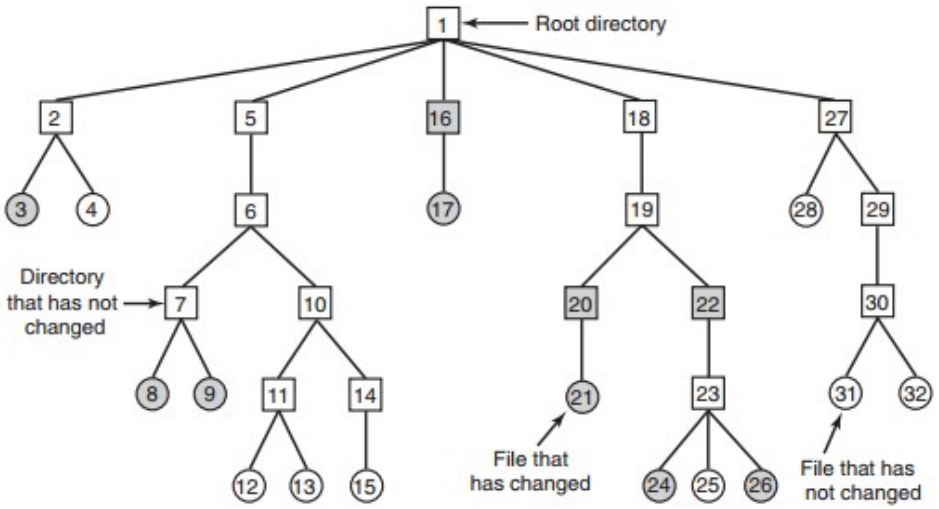

Abbildung 2.2: [Tan16e]

2.4 Konsistenz

Die Konsistenz der Daten muss immer gewährleistet sein. Dafür gibt es Hilfsprogramme und Konsistenzprüfungen auf Datei und Blockebene.

- fsck (file system consistency check)
- sfc (system file checker)

Bei der Konsistenzprüfung werden zwei Tabellen mit einander verglichen. Die erste Tabelle enthält je einen Eintrag pro belegten Block und die zweite je einen Eintrag pro freien Block.[4]

2.5 Optimierung

Es gibt vier wesentliche Optimierungsmöglichkeiten der Performance.[5]

1. Block-Cache (Puffer)
   Hierbei wird überprüft, ob der zu lesende Block schon im Cache vorhanden ist. Wenn es nicht der Fall ist, dann wird er in den Cache geladen und dann gelesen.

2. Vorausschauendes lesen von Blöcken
   Da Blöcke oft sequentiell gelesen werden, wird bei einem Zugriff auf Block k angekommen, dass Block k+1, also der nachfolgenden Block, auch gelesen wird. Bei dieser Annahme wird er schon vor dem eigentlichen Zugriff in den Cache geladen.

[4]Quelle: [Tan16f]
[5]Quelle: [Tan16g]

3. Plattenarmbewegung reduzieren
   Für dieses Verfahren werden Blöcke, auf die möglicherweise nacheinander zugegriffen wird, auf den selben Zylinder der Festplatte platziert.

4. Defragmentierung[6]
   Die Defragmentierung ist das Konzept der Neuordnung. Durch löschen und fragmentieren von Blöcken entstehen Lücken, die durch eine Defragmentierung geschlossen werden.

[6]Sollte nicht bei einer SSD angewendet werden.

3 Moderne Features

3.1 Copy-on-Write

Im Allgemeinen werden kopierte Daten, aus platzsparenden Gründen nicht real kopiert, sondern nur auf das Original verlinkt. So entstehen statt zwei identischer Dateien, nur eine Datei und eine Verlinkung darauf. Erst bei Änderungen die vom Original abweichen, wird die Verlinkung aufgehoben und die Datei neu geschrieben. Dabei wird der alte Datensatz überschrieben.
Im Gegensatz dazu wird beim Copy-on-Write nie das Original überschrieben. Deshalb werden neue Datensätze immer nach dem letzten erstellten Datensatz des Speichermediums geschrieben und die Metadaten angepasst.
Dadurch sind alle vorherigen Zustände von Daten noch vorhanden, wiederherstellbar und können durch Snapshots für einen bestimmten Zeitpunkt erfasst und eingefroren werden.[1]

3.2 Verfügbarkeit

Mit Snapshots können effiziente Backups erstellt werden. Da das Verfahren jederzeit on-the-fly, also im laufenden Betrieb durchgeführt werden kann. Snapshots können aufeinander aufbauen und müssen nur das alte Snapshot mit den neuen Daten zusammenfassen. Das führt zur einer schnellen Durchführung und Speicherersparnis, da nie alle Daten neu erfasst werden müssen.
Um sicherzustellen das keine doppelte Daten gespeichert werden, wird jede Datei mit einer Hash-Prüfsumme versehen. Diese Prüfsumme wird für jede Datei einmalig berechnet und vergeben. Alle Prüfsummen werden in einer Tabelle im Puffer gehalten, damit jederzeit auf sie schnell zugegriffen werden kann. Das Verfahren beugt der Deduplikation vor.[2]

3.3 Storage Pools

In einem *Storage Pool* werden alle zur Verfügung stehenden Festplatten in einem einzigen Verbund (*Pool*) zusammengeführt.

[1] Quelle: [Orab]
[2] Quelle: [Orab]

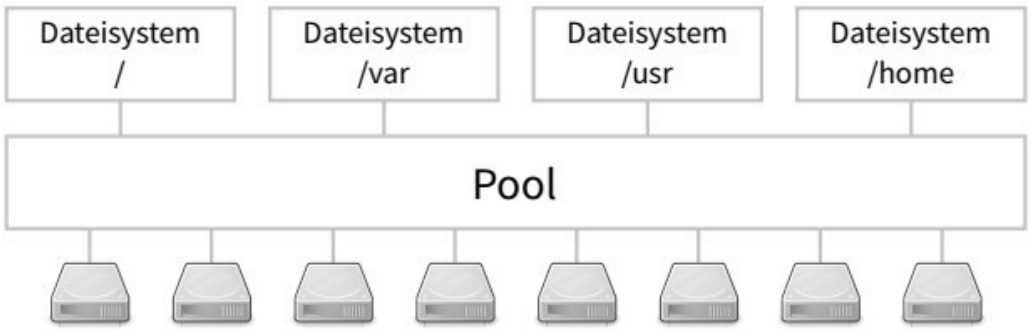

Abbildung 3.1: [Kuh16]

Geht man in der oben abgebildeten Grafik davon aus, dass jede Festplatte 10 Tebibyte bereitstellt, dann steht dem Pool 80 Tebibyte (8 Stk. * 10 TiB) zur Verfügung. Dieser Pool kann danach in beliebig viele weitere virtuelle Geräte unterteilt und bereitgestellt werden. In diesem Beispiel mit vier virtuellen Geräten (root, /var, /usr, /home). Durch die Nutzung eines *Storage Pools* hat man Zugriff auf die gesamte Speicherkapazität der Festplatten. Zusätzlich lässt sich die Kapazität der einzelnen virtuellen Geräte dynamisch anpassen.

3.4 RAID-Funktionalität

Die virtuellen Geräte unterstützen dabei einen *raidz3* Verbund bzw. *RAID Triple Parity*. Dabei handelt sich es um ein RAID5 mit zusätzlicher Verteilung der Paritätsinformationen auf allen Geräten. Es können also bis zu drei Geräte ausfallen, ohne das Daten verloren gehen.

Das eingebettete RAID-Subsystem ermöglicht eine Differenzierung von belegten und freien Blöcken. Dadurch können im Schadensfall zielsicher nur die belegten Blöcke wiederhergestellt werden. [3]

Beim *Striping* werden die Daten dynamischen auf die im Pool zusammengeführten Geräte verteilt. Dabei wird der Datenträger mit dem größten freien Speicherbereich als stärkstes Priorisiert. Dadurch werden neue eingebundene Datenträger, vor den schon vorhandenen Datenträger, bevorzugt beschrieben.

3.5 Datenkapazität

Durch immer größer werdende Dateien und deren Anzahl, sind moderne Dateisysteme auf größere Kapazitäten ausgelegt.

[3] Quelle: [Orad]

Ein neues 128-Bit-Dateisystem unterstützt folgende Kapazitäten:

- 256 Zebibyte große Dateisysteme
- 16 Exbibyte große Dateien
- 2^{48} Anzahl von Dateien pro Verzeichnis

3.6 Scrubbing

Scrubbing ist ein Verfahren um die Integrität der Daten zu gewährleisten. Dabei wird für jeden Block in einem *Pool* eine einmalige *checksum* (Prüfsumme) berechnet und gespeichert. Sobald ein Block danach gelesen oder geändert wird, wird die Prüfsumme neu berechnet und mit der Ursprünglichen verglichen.
Sollte es dabei zu keiner Übereinstimmung kommen, dann wird der Fehler protokolliert und wenn möglich repariert. Dazu können aus Snapshot die Blöcke wiederhergestellt werden.[4]

3.7 Kompression und Verschlüsselung

Trotz der Möglichkeit immer größere Dateien zu speichern, bieten neue Dateisystem zusätzlich verschiedene Kompressionsalgorithmen (z.B. LZ4, LZO) an. Dabei setzen sie auf eine Echtzeitkomprimierung, die für alle Daten gleich ist und trotzdem einen transparenten Zugriff ermöglichen. Dadurch wird der Speicherverbrauch reduziert und Investitionen für Neuanschaffung gespart.

Zusätzlich können Daten auf Dateiebene mit dem *Advanced Encryption Standard* verschlüsselt werden. Dabei wird eine Schlüssellänge von bis zu 256 Bit unterstützt. Der AES Sicherheitsschlüssel wird mit einem *wrapping key* vom User verschlüsselt. Damit hat der User keinen direkten Zugriff auf den AES Sicherheitsschlüssel, kann diesen aber trotzdem verwalten.[5]

[4] Quelle: [Orac]
[5] Quelle: [Oraa]

4 Zukünftige Dateisysteme

4.1 Searchable VFS

Da es eine breite an unterschiedlichen Dateisystemen gibt und immer mehr Daten auf eine einfache Art bereitgestellt werden müssen, bietet das *Searchable VFS* eine Möglichkeit unterschiedliche Dateisysteme zu verbinden und auch leicht zu durchsuchen.

Ein *Searchable VFS* ist ein modifiziertes *Virtual File System*. Ein virtuelles Dateisystem ist eine weitere Abstraktionsschicht zwischen dem Kernel und dem richtigen Dateisystem. Dadurch entsteht eine direkte Schnittstelle zwischen den Applikationen und einem neuen Dateisystem.

Dadurch können Applikationen von unterschiedlichen lokalen Dateisystemen, auf ein gemeinsames virtuelles Dateisystemen zugreifen. Zusätzlich bieten diese Dateisysteme eine Suchfunktion. Dafür werden folgende Informationen von allen Dateien und Verzeichnisse in Tabellen vorgehalten:[1]

- Attribute pro Datei auf derselben Partition
- Indexierung der Datei auf separater Partition

4.2 Cluster Dateisysteme

Gerade bei Applikationen die über ein Netzwerk bereitgestellt werden und im klassischen Client-Server Modell laufen, kommt es gerne zu Engpässen der Performance. Dabei hilft ein Cluster Dateisystem. Es verbindet die Rechenkapazität aller Server zu einem Cluster. Die Clients greifen dabei auf ein einziges Dateisystem zu, was aber durch mehrere Server bereitgestellt wird.

Durch die einfache Skalierbarkeit der Servern und Speichermedien im Cluster, kann die beste Performance erzielt und eine ausreichende Kapazität gewährleistet werden.[2]

[1] Quelle: [SY06]
[2] Quelle: [Fra]

5 Fazit

Zusammenfassend wurden vier verschiedene Dateisystemen betrachtet:

- Resilient File System von Microsoft
- Apple File System von Apple
- B-tree File System von Oracle Corporation
- Zettabyte File System von Oracle Corporation

Die folgende Auswertung zeigt welche Features durch welches Dateisystem unterstützt werden.

	ReFS	APFS	Btrfs	ZFS
Copy-on-Write	✓	✓	✓	✓
Deduplizierung			✓	✓
Snapshots	✓	✓	✓	✓
Storage Pools			✓	✓
RAID	✓		✓	✓
Dynamisches Striping	✓		✓	✓
Scrubbing	✓		✓	✓
Kompression		✓	✓	✓
Verschlüsselung		✓		✓

Durch stetig größer werdende und vielfältigere Daten auf unseren Servern und Clients, sind Kompressions- und Deduplikationsfeatures unabdingbar geworden. Sie helfen dabei Speicherplatz zu sparen und wieder Herr über die Masse an Daten zu werden.

Für die Einhaltung der Schutzziele wie Vertraulichkeit, Integrität und Verfügbarkeit, bieten moderne Dateisystem eine standardisierte Verschlüsselung, Scrubbing-Funktionen zur Fehlerbehebung und Copy-on-Write Funktionalitäten, für eine einfache aber konstante Erstellung von Snapshots.

Literaturverzeichnis

[bW] brtfs Wiki. B-tree file system. Website. https://btrfs.wiki.kernel.org.

[Fra] ThinkParQ Frauenhofer. Beegfs. Website. https://www.beegfs.io/content/documentation/#documentation.

[Kuh16] Michael Kuhn. *Hochleistungs-Ein-/Ausgabe*, chapter Moderne Dateisysteme, page 18. Wissenschaftliches Rechnen, Fachbereich Informatik, Universität Hamburg, 2016.

[Oraa] Oracle. Encryption. Website. https://docs.oracle.com/cd/E53394_01/html/E54801/gkkih.html.

[Orab] Oracle. Oracle solaris zfs administration guide. Website. https://docs.oracle.com/cd/E19253-01/819-5461/zfsover-2/.

[Orac] Oracle. Scrubbing. Website. https://docs.oracle.com/cd/E19253-01/819-5461/gbbxi/index.html.

[Orad] Oracle. System administration commands. Website. https://docs.oracle.com/cd/E18752_01/html/816-5166/zpool-1m.html#REFMAN1Mzpool-1m.

[Sol] Oracle Solaris. Zettabyte file system. Website. https://docs.oracle.com.

[SY06] Lee H. Song Y., Choi Y. *Searchable Virtual File System: Toward an Intelligent Ubiquitous Storage*, page 2. Springer, Berlin, Heidelberg, 2006.

[Tan16a] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 265. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16b] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 282. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16c] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 351. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16d] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 310. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16e] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 383. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16f] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 312. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Tan16g] Andrew S. Tanenbaum. *Moderne Betriebssysteme*, chapter No Silver Bullet - Essence and Accident in Software Engineering, page 314. PEARSON EDUCATION DEUTSCHLAND GMBH, Lilienthalstr. 2, 4. edition, 2016.

[Wika] Wikipedia. Apple file system. Website. `https://en.wikipedia.org/wiki/Apple_File_System`.

[Wikb] Wikipedia. Resilient file system. Website. `https://en.wikipedia.org/wiki/ReFS`.

[ZFS] Open ZFS. Open zfs. Website. `http://open-zfs.org/wiki/Features`.

Anhänge

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Wirtschaftsinformatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Unterschrift

Veröffentlichung

Ich bin damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik eingestellt wird.

Ort, Datum	Unterschrift