
Metadatenverwaltung

Seminar Dateisysteme SS07

Fikret Kaplan

Ruprecht-Karls-Universität Heidelberg
Institut für Informatik

Inhalt

- Metadaten
 - Allgemein
 - Metadaten-Konsistenz
- Metadatenverwaltung
 - Strategien zur Konsistenzsicherung
- Zusammenfassung

Metadaten

Allgemein

- „Daten über die Daten“
- beschreiben Eigenschaften von Dateien
- Inode
 - Besitzer
 - Typ
 - Grösse
 - Zeitstempel (Erzeugung, letzte Modifikation)
 - Zugriffsrechte
 - Datenblockreferenzen
 - ...
- Metadaten = Verwaltungsinformationen

Oft zu lesen: Metadaten = Inodes,

Hier: Metadaten = Inodes, Superblock, Directory

- Superblock enthält Verwaltungsinformationen des Dateisystems
 - Grösse des Dateisystems
 - Anzahl/ Liste der freien Datenblöcke
 - Zeiger auf ersten freien Datenblock in Liste
 - Grösse der Inode-Liste
 - Anzahl/ Liste der freien Inodes
 - Zeiger auf nächsten freien Inode in Liste
 - Sperr-Felder für freie Datenblöcke/ Inodes
 - Feld, ob Superblock verändert wurde

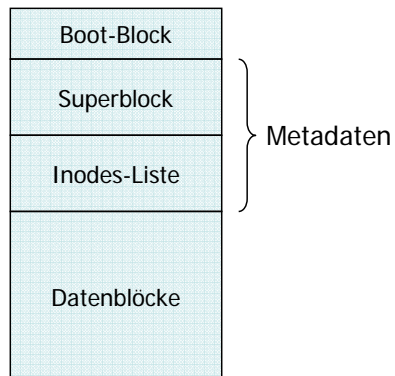
- Directories enthalten Dateinamen und Inode-Nummern

- Nummer der Inode
- Grösse
- Typ (Datei, Verzeichnis,...)
- Länge des Dateinamen

Inode#	Grösse	Typ	Länge	Dateiname		
--------	--------	-----	-------	-----------	--	--

Ein Directory kann beliebig (!) viele Einträge enthalten.

Grobe Struktur des UNIX-Dateisystems



Metadaten einer Datei beinhalten:

- Inode
- Verzeichniseintrag
- Block-Bitmap-Eintrag
- Inode-Bitmap-Eintrag

- Dateisystem-interne Datenstruktur
- Stellen sicher, dass Daten korrekt organisiert und zugreifbar sind
- Informationen über Inodes, freien Speicher, etc.
- Beschreiben Dateien, nicht die Daten in den Dateien



Spiegeln die Struktur, d.h. Verzeichnisbaum, Dateien, belegter/ freier Speicher

Metadaten		Allgemein							
File System	owner	POSIX file permissions	creation ts	last acces ts	last change ts	last archive ts	access control lists	security/ MAC labels	extended attributes
ext2	■	■	■	■	■	■	■	■	■
ext3	■	■	■	■	■	■	■	■	■
ext4	■	■	■	■	■	■	■	■	■
ReiserFS	■	■	■	■	■	■	■	■	■
Reiser4	■	■	■	■	■	■	■	■	■
XFS	■	■	■	■	■	■	■	■	■
JFS	■	■	■	■	■	■	■	■	■
NTFS	■	■	■	■	■	■	■	?	■

Quelle: Wikipedia

Dateisysteme – Metadatenverwaltung 8/25

POSIX permissions: -rwxrw-r--

last change = last metadata change

ACL: Werte-Paar, (guest, read) file_x (Gruppe guest hat read-Zugriff auf Datei file_x)

MAC (mandatory access control): Authorisierung & Sicherheit (z.B. auch bei Zugriffen durch Prozesse)

extended attributes: Autor, Checksum, etc.

Metadaten

Konsistenz

Änderungen (=Operationen) am Dateisystem...


- manipulieren die Metadaten
- sind nicht atomar
- sind asynchron

Systemabsturz während Änderung führt zu inkonsistenten Metadaten

asynchron: zuerst Änderungen im Speicher, später auf Festplatte

Beispiel: Umbenennen einer Datei

1. Verzeichniseintrag mit Verweis auf Inode löschen
2. Neuer Eintrag mit Verweis auf selben Inode
3. Systemabsturz zwischen 1. und 2.
4. Inode-Verweis fehlt (oder doppelt vorhanden)

 inkonsistentes Dateisystem

- Metadaten enthalten Pointer und Informationen zu den Sektoren auf der Festplatte und zu den Dateien
- Um das Dateisystem konsistent zu halten, muss die Konsistenz der Metadaten gewahrt werden

Metadatenverwaltung Strategien

Strategien zur Sicherung der Metadaten-Konsistenz

- Logging
- Soft Updates
- Snapshotting

- Was wird geloggt?
 - neue Datei (**create**)
 - neues Verzeichnis (**mkdir**)
 - neuer Link (**link**)
 - Datei entfernen (**unlink**)
 - symbolischer Link (**symlink**)
 - umbenennen (**rename**)
 - ...
- Log = atomare Transaktion

geloggt wird ins Journal.

z.B. in ext3 auch Möglichkeit neben den Metadaten die Datenänderungen ebenfalls zu loggen.

Log muss atomar sein, da sonst inkonsistente Metadaten geschrieben werden könnten.

- Ablauf
 1. Änderung (z.B. Anlegen neuer Datei)
 2. Log ins Journal (Beginn Transaktion)
 3. Neue Metadaten im Dateisystem
 4. **commit** (Ende Transaktion)
- Erst mit **commit** sind die neuen Metadaten für das Dateisystem gültig

Beispiel Transaktion bei Dateierstellung

```
TxBegin(dip->i_ipmnt, &tid, 0);
tblk = &TxBlock[tid];
tblk->xflag |= COMMIT_CREATE;
tblk->ip = ip;
/* Work is done to create file */
rc = txCommit(tid, 2, &iplist[0], 0);
TxEnd(tid);
```

TxBegin kennzeichnet den Anfang, TxEnd das Ende des atomaren Blocks.

Metadatenverwaltung

Logging

- früher (z.B. ext2):

Nach Systemabsturz *alle* Metadaten prüfen (**fsck**)
→ lange Wiederherstellungszeit

- mit Logging (z.B. ext3, reiserfs):

Metadatenänderungen werden geloggt

Nur inkonsistente Metadaten prüfen
→ kurze Wiederherstellungszeit

Logging (Journaling) sichert die Konsistenz des Dateisystems, nicht unbedingt die Konsistenz der Daten.

Metadatenverwaltung Soft Updates

- Metadaten-Updates werden im Speicher gehalten und sortiert auf die Platte geschrieben
- Reihenfolge wird durch die Operation bestimmt
- Soft Updates pflegt somit die Abhängigkeiten (z.B. Datenblockverweise, Verzeichniseinträge)
- Metadaten-Updates sind atomar, d.h. entweder ganz oder garnicht

Metadatenverwaltung Soft Updates

- Regeln
 1. Niemals einen Pointer auf eine Struktur zeigen lassen, bevor diese initialisiert wurde
 2. Niemals eine Ressource neu vergeben, bevor alle bisherigen Pointer auf sie entfernt wurden.
 3. Niemals einen alten Pointer auf eine Ressource löschen, bevor nicht ein neuer Pointer auf diese Ressource zeigt

Metadatenverwaltung Soft updates

- alle Operationen haben ihre festgelegte Reihenfolge der Suboperationen
- z.B. Erstellen einer Datei
 - Inode erstellen und speichern
 - in Inode-Bitmap eintragen
 - Verzeichniseintrag erstellen
- falls umgekehrt, bleibt der Verzeichniseintrag nach Systemabsturz unreferenziert

Für jede Operation wird eine Reihenfolge der Suboperationen festgelegt. Diese Folgen
sehen bei den unterschiedlichen Operationen auch jeweils anders aus.

Metadatenverwaltung

Logging vs. Soft Updates

Logging	Soft Updates
zuerst Log, dann commit doppeltes Schreiben einfache Implementierung	Update-Folge beachten schneller komplexer Code

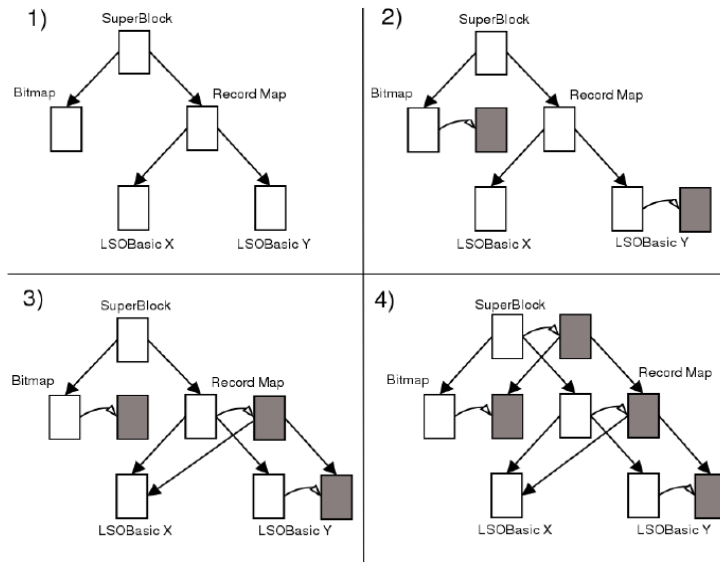
- beide Methoden garantieren Metadaten-Konsistenz
- Soft Updates schneller als Logging, falls nicht viele Dateien erstellt und gelöscht werden (**rollback()**)

Metadatenverwaltung Snapshotting

- hält Metadaten auf der Platte konsistent
- kopiert veränderte Version der Metadaten (ähnlich Subversioning)
- neue Version wird gültig durch Superblock-Update

Eine neue Version der Metadaten wird Generation genannt. Eine Generation beinhaltet die kopierten, veränderten Versionen des Superblocks, der Inode-Map und der geänderten Inodes.

Metadatenverwaltung Snapshotting



Quelle: Meta-data Snapshotting: A Simple Mechanism for File System Consistency (Livio B Soares, Department of Computer Science, Universidade Sao Paulo, Brazil)

Metadatenverwaltung Snapshotting

- Frage: Wann eine neue Generation erzeugen?
 - verzögert:
 - gut für die Performance, da weniger Speicherverbrauch
 - dafür aber nach Systemabsturz neuste Änderungen nicht verfügbar
 - Kriterien:
 - Menge an „dirty meta-data“
 - vergangene Zeit seit letztem Superblock-Update
 - **sync ()** - schreibt aktuelle und erzeugt neue Generation

sync() ist ein Systemaufruf. Aufruf wird auch durch die anderen zwei Kriterien bestimmt

Metadatenverwaltung Snapshotting

- Probleme:
 - Superblock-Update ist nicht atomar.
Deshalb zwei Superblocks mit Versionsnummern in den Sektoren (Redundanz)

Zusammenfassung

- Metadaten (Inodes, Superblock) beschreiben das Dateisystem
- um Datenverluste zu vermeiden muss die Konsistenz der Metadaten gesichert werden
- um lange Dateisystemüberprüfungen zu vermeiden, sind verschiedene Mechanismen im Einsatz (hauptsächlich Logging)

Quellen

- Journaling Filesysteme unter Linux, <http://www.fh-wedel.de/~si/seminare/ws01/Ausarbeitung/3.journalfs/glossar.html>
- Das I-Node System, <http://www.linux-praxis.de/linux1/filesystem3.html>
- JFS, <http://jfs.sourceforge.net/>
- Metadata-Snapshotting, www.ele.uri.edu/tcca/camera_ready/Livio_snapshot-final.pdf
- Soft Updates, www.usenix.org/events/usenix99/full_papers/mckusick/mckusick.pdf
- Wikipedia:
 - http://en.wikipedia.org/wiki/Comparison_of_file_systems
 - <http://de.wikipedia.org/wiki/Superblock>
 - <http://de.wikipedia.org/wiki/Inode>