Journaling-Dateisysteme

1. **Theoretische Grundlagen**

   Was bedeutet Journaling?

   Wieso ist Journaling nützlich/nötig?

   Welche Möglichkeiten gibt es?

   Was sind die Nachteile?

2. **Implementierungsbeispiele**

   ext3 & ReiserFS

3. **Abschluss & Fragen** :))

Was bedeutet Journaling?

Protokollierung

- Schreiboperationen werden protokolliert
- jeweils ein zusammen-hängender Vorgang
- Auswertung nur bei Unterbrechung nötig

Log-Structured

- Alles steht im Protokoll
- Protokoll dehnt sich linear im Speicher aus
- Inodes werden indiziert
- geschrieben wird nur an einer Stelle

Wieso ist Journaling nützlich/nötig?

Inkonsistenzen

- bei Strom- oder Hardwareausfällen
- zu Schreibendes im Zwischenspeicher
- mögliche Folgen:
  - verlorene Inode-Referenzen
  - falsche Meta-Informationen
  - zerstörte Inhalte

Klassische Lösung

- automatische oder routinierte Überprüfung beim Einhängen
- sehr zeitaufwändig
- Überprüft werden:
  - alle Referenzen und -Zähler
  - Markierung der belegten Blöcke
  - doppelt belegte Blöcke
- Reperatur nicht vollständig

Lösung mit Journal

- zur Reperatur muss nur das Protokoll gelesen werden
- ein konsistenter Zustand lässt sich immer herstellen
- teilweise werden auch die neuen Daten gerettet

Atomares Dateisystem

Transaktionsmodell:

- eine zusammenhängende Operation wird unterbrochen
- Nach der Wiederherstellung ist die Operation entweder komplett oder gar nicht durchgeführt

Welche Möglichkeiten gibt es?

Logging der Meta-Daten

- nur Dateioperationen werden protokolliert, nicht die Daten
- Organisationsstruktur bleibt konsistent
- Inhalte gehen verloren

Beispielprotokoll

```
 Inode 777:   intent-to-commit
Block 3111:   data update (changes)
Block 3506:   data update (changes)
Block 3790:   data update (changes)
Block 3791:   data update (changes)
 Inode 777:   update data block list to
              3110, 3111, 3506,
              3790, 3791
 Inode 777:   access time 23:42
              19-JAN-2003
 Inode 777:   modification time 23:42
              19-JAN-2003
 Inode 777:   committed
```

Vollständiges Logging

- “Data Journaling”
- auch die Daten selbst werden protokolliert
- schlägt ein Überschreibe-vorgang fehl, ist der alte Inhalt nicht zerstört

→ atomares Dateisystem

Log-Structured

- gleicher Effekt wie Data Journaling
- Daten werden immer anhängend geschrieben
- zur Wiederherstellung wird vom letzten “Checkpoint” aus gearbeitet

Checkpoint-Regions

- zwei fixe Checkpoint-Regions werden abwechselnd geschrieben
- sie enthalten:
  - Zeiger auf letztes Segment
  - globale Daten (Inode-Map)
  - Zeitpukt der Erstellung (zuletzt)
- CP-Regions werden nur ab und an geschrieben
- ab dem letzten CP kann das Log weiter nachvollzogen werden

Was sind die Nachteile?

Protokollierungsaufwand

- das Protokoll muss sehr häufig geschrieben werden
- der Schreibkopf muss zum Protokoll und zurück springen

- Protokollieren stark optimiert - erheblich geringere Kosten als für die Schreiboperation selbst

Doppeltes Schreiben

- Datenblöcke müssen erst ins Log geschrieben werden, dann an ihren Bestimmungsort
- somit können Daten atomar überschrieben werden

➡ Der Schreibaufwand ist quasi verdoppelt

Fragmentierung bei Log-Structured

- doppeltes Schreiben entfällt
- dafür verwaiste Speicher-bereiche innerhalb des Logs
- Log ist in Segmente unterteilt, die jeweils aufgeräumt werden können
- Mehrere lückenhafte Segmente werden zusammengefasst

Implementierung: ext3

ext3 – ext2 mit Journal

- abwärtskompatibel zu ext2
- folgende Journal-Arten:
  - **data=writeback**
  - **data=ordered**
  - **data=journal**
- wegen der Beliebtheit von ext2 etabliert

ext3 – Journal-Arten

- **data=writeback**

  entspricht einfachem Metadaten-Journaling

- **data=ordered**

  Metadaten-Journaling mit synchronem Schreiben

- **data=journal**

  volles Data Journaling

Implementierung: ReiserFS

ReiserFS (3 & 4)

- konservatives Journaling bei Reiser3, Metadaten oder Full
- dagegen Reiser4 atomares Dateisystem
- Data Journaling ohne doppeltes Schreiben:
  Journal wird im Speicher bewegt

Reiser4: Vor-/Nachteile

- Data Journaling erhöht Schreibdauer im Schnitt nur um 20%
- viele performanceträchtige Techniken

Nachteil:

- erzwungene Fragmentierung

Abschluss

ext3/ReiserFS verwenden

- Migration von ext2 auf ext3 nur ein Befehl
- Reiser3 im aktuellen 2.4er und 2.6er Kernel
- Reiser4 noch in der Testphase

Fragen™?