btrfs & F2FS vs. ext4

Sven Fiergolla & Tobias Dahlem

3. März 2020

Introduction - Flash memory

btrfs

F2FS

benchmarks

Besonderheiten des Flashspeichers

- Adressen Mapping
 - Zuweisung von logischen zu physischen Adressen
 - ► Häufige Verwendung wegen Charakteristika von Flash-Speichern
- ► Garbage Collection
 - Alte Daten/Seiten werden als ungültig markiert (Allokierter Speicherplatz)
 - Hoher Aufwand durch kopieren von Blöcken
- Wear Leveling
 - Begrenzte Haltbarkeit von Flash-Zellen (Löschen, Schreiben)
 - Gleichmäßige Abnutzung der Zellen
- ▶ Verwendung eines Flash Translation Layer (FTL) im Controller

btrfs

btrfs - Struktur

btrfs - CoW

btrfs - RAID

F2FS

- ► Flash-Dateisystem von Samsung (veröffentlicht 2012)
- ► Entwickelt nur für Flash-Speicher (SD-Karte, SSDs, eMMC-Karten)
- ▶ Ziel: Optimierung der Performance und Lebenszeit von Flash-Speichern
- ▶ Entwickelt als Open-Source Projekt
- Verfolgt den Ansatz eine Log-structured File System (append-only logging)
- ► Arbeitet nicht auf ,,raw" Flash-Zellen (Benötigt einen FTL)
- ▶ Viele Möglichkeiten zur Anpassung des Systems
- ▶ Verwendung von iNodes und Datenblöcken (Ahnlich zu UNIX)
- Verfügbar ab Linux Kernel 3.8
- ▶ Verwendung in Huawei (2016), Galaxy Note 10, Google Nexus

F2FS

- ► Flash-Dateisystem von Samsung (veröffentlicht 2012)
- Entwickelt nur für Flash-Speicher (SD-Karte, SSDs, eMMC-Karten)
- ▶ Ziel: Optimierung der Performance und Lebenszeit von Flash-Speichern
- ► Entwickelt als Open-Source Projekt
- Verfolgt den Ansatz eine Log-structured File System (append-only logging)
- ► Arbeitet nicht auf "raw" Flash-Zellen (Benötigt einen FTL)
- ▶ Viele Möglichkeiten zur Anpassung des Systems
- ▶ Verwendung von iNodes und Datenblöcken (Ähnlich zu UNIX)
- Verfügbar ab Linux Kernel 3.8
- ▶ Verwendung in Huawei (2016), Galaxy Note 10, Google Nexus

F2FS

- ► Flash-Dateisystem von Samsung (veröffentlicht 2012)
- Entwickelt nur für Flash-Speicher (SD-Karte, SSDs, eMMC-Karten)
- ▶ Ziel: Optimierung der Performance und Lebenszeit von Flash-Speichern
- ► Entwickelt als Open-Source Projekt
- Verfolgt den Ansatz eine Log-structured File System (append-only logging)
- ► Arbeitet nicht auf "raw" Flash-Zellen (Benötigt einen FTL)
- ▶ Viele Möglichkeiten zur Anpassung des Systems
- ▶ Verwendung von iNodes und Datenblöcken (Ähnlich zu UNIX)
- Verfügbar ab Linux Kernel 3.8
- ▶ Verwendung in Huawei (2016), Galaxy Note 10, Google Nexus

- Orientierung an FTL-Einheiten um Kosten zu Vermeiden
- ▶ Einheiten: Blöcke, Segmente, Sektionen, Zonen
- ► Metadaten
 - Random Writes: Vorhalten in Arbeitsspeicher (Bei Checkpoints schreiben)
- Haupt-Speicherbereich:
 - Aufgeteilt in Standardmäßig 4KB Blocks (Jeder Block ist Node- oder Data-Block)
 - Node- und Data-Blocks liegen in verschiedenen Segmenten

- Orientierung an FTL-Einheiten um Kosten zu Vermeiden
- ► Einheiten: Blöcke, Segmente, Sektionen, Zonen
- Metadaten:
 - Random Writes: Vorhalten in Arbeitsspeicher (Bei Checkpoints schreiben)
- Haupt-Speicherbereich:
 - Aufgeteilt in Standardmäßig 4KB Blocks (Jeder Block ist Node- oder Data-Block)
 - Node- und Data-Blocks liegen in verschiedenen Segmenten

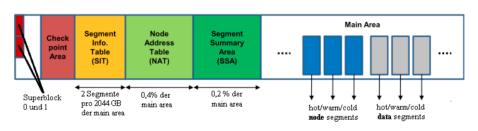


Figure: on-disk Layout F2FS

- ▶ Orientierung an FTL-Einheiten um Kosten zu Vermeiden
- ► Einheiten: Blöcke, Segmente, Sektionen, Zonen
- Metadaten:
 - Random Writes: Vorhalten in Arbeitsspeicher (Bei Checkpoints schreiben)
- Haupt-Speicherbereich:
 - Aufgeteilt in Standardmäßig 4KB Blocks (Jeder Block ist Node- oder Data-Block)
 - ▶ Node- und Data-Blocks liegen in verschiedenen Segmenten

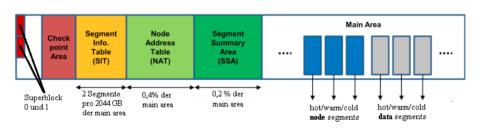


Figure: on-disk Layout F2FS

- ▶ Orientierung an FTL-Einheiten um Kosten zu Vermeiden
- ► Einheiten: Blöcke, Segmente, Sektionen, Zonen
- Metadaten:
 - Random Writes: Vorhalten in Arbeitsspeicher (Bei Checkpoints schreiben)
- Haupt-Speicherbereich:
 - Aufgeteilt in Standardmäßig 4KB Blocks (Jeder Block ist Node- oder Data-Block)
 - ▶ Node- und Data-Blocks liegen in verschiedenen Segmenten

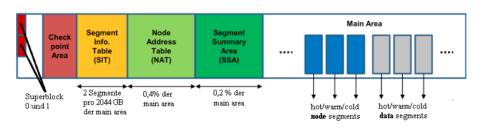


Figure: on-disk Layout F2FS

F2FS - Besonderheiten I

- Multi-Head Logging
 - Mehrere aktive Logsegmente parallel (Standard 6)
 - Parallele Verwendung durch Architektur möglich (multi-Streaming Interface)
 - Unterscheidung der Daten in hot/warm/cold Schema (Update Frequenz)
- Kosten-Effiziente Index Struktur
 - Verwendung einer neuartigen Indexes: note adress table (NAT)
 - Zur Vermeidung des "wandering tree" Problems
 - Nur Update des direct Node Block und NAT
 - ▶ Reduktion der Updates um Schreiboperationen zu sparen
- Adaptive logging
 - Append-only Logging : Standardmäßig (random writes werden sequentiell)
 - ► Threaded Logging : Verwendung bei hoher Auslastung (random writes)

F2FS - Besonderheiten I

- Multi-Head Logging
 - Mehrere aktive Logsegmente parallel (Standard 6)
 - ► Parallele Verwendung durch Architektur möglich (multi-Streaming Interface)
 - Unterscheidung der Daten in hot/warm/cold Schema (Update Frequenz)
- Kosten-Effiziente Index Struktur
 - Verwendung einer neuartigen Indexes: note adress table (NAT)
 - ► Zur Vermeidung des ,,wandering tree" Problems
 - Nur Update des direct Node Block und NAT
 - ▶ Reduktion der Updates um Schreiboperationen zu sparen
- Adaptive logging
 - Append-only Logging : Standardmäßig (random writes werden sequentiell)
 - ▶ Threaded Logging : Verwendung bei hoher Auslastung (random writes)

F2FS - Besonderheiten I

- ► Multi-Head Logging
 - Mehrere aktive Logsegmente parallel (Standard 6)
 - Parallele Verwendung durch Architektur möglich (multi-Streaming Interface)
 - Unterscheidung der Daten in hot/warm/cold Schema (Update Frequenz)
- Kosten-Effiziente Index Struktur
 - Verwendung einer neuartigen Indexes: note adress table (NAT)
 - ► Zur Vermeidung des ,,wandering tree" Problems
 - Nur Update des direct Node Block und NAT
 - ▶ Reduktion der Updates um Schreiboperationen zu sparen
- Adaptive logging
 - ► Append-only Logging : Standardmäßig (random writes werden sequentiell)
 - ► Threaded Logging : Verwendung bei hoher Auslastung (random writes)

F2FS - Besonderheiten II

- ► Garbage Collection
 - ▶ On-Demand: Wenn nicht genügend Speicherplatz verfügbar ist
 - ▶ Greedy: Auswahl des Opfersegments mit wenigsten gültigen Blöcken
 - ▶ Background: Bei geringer Auslastung des Systems von Kernel ausgeführt
 - ▶ Kosten-Effizient: Auswahl durch Segment-Alter und Anzahl gültiger Blöcke
- RAID
 - ► In Progress ...

F2FS - Besonderheiten II

- ► Garbage Collection
 - ▶ On-Demand: Wenn nicht genügend Speicherplatz verfügbar ist
 - ▶ Greedy: Auswahl des Opfersegments mit wenigsten gültigen Blöcken
 - ▶ Background: Bei geringer Auslastung des Systems von Kernel ausgeführt
 - ▶ Kosten-Effizient: Auswahl durch Segment-Alter und Anzahl gültiger Blöcke
- ► RAID
 - ▶ In Progress ...

F2FS - Bewertung

Vorteile

- Optimierung der Zusammenarbeit von FTL und Dateisystem
- Vermeidung des Wandering Tree Problems
- Anpassung des Dateisystems an System-Status
- ► Hohe Anzahl an Parametern um Dateisystem anzupassen

Nachteile

F2FS - Bewertung

Vorteile

- Optimierung der Zusammenarbeit von FTL und Dateisystem
- Vermeidung des Wandering Tree Problems
- Anpassung des Dateisystems an System-Status
- Hohe Anzahl an Parametern um Dateisystem anzupassen

Nachteile

- Nur für Flash-Speicher (mit einem FTL)
- FTL Qualität wichtiges Kriterium
- Initialer hoher belegter Speicherplatz durch Metadaten
- ► Hohe CPU-Belastung beim Schreiben von Dateien

benchmarks

benchmarks - Results

Table: Benchmark