Dateisystem Benchmark: Btrfs & F2FS

Sven Fiergolla & Tobias Dahlem
Betriebssysteme WiSe 2019/2020
Universität Trier

3. März 2020

1 Flashspeicher

Flashspeicher werden heutzutage in vielen Bereichen verwendet, wie Computern, Smartphones und Servern. Neben ihren Vorteilen, wie die schnellere Geschwindigkeit, gibt es aber auch einige Einschränkungen, wie das Bedürfnis die Zellen zu löschen bevor sie beschrieben werden oder dass die einzelnen Blöcke nur eine beschränkte Lebensdauer besitzen. In dem Bereich der Dateisysteme gibt es verschiedene Vertreter, welche die Eigenschaften der Flashspeicher berücksichtigen. Zu den durch die Bauart bestimmten Charakteristika zählen das Adress Mapping, die Garbage Collection und das Wear Leveling.

Im Rahmen des Adress-Mappings erfolgt die zuordnung der logischen Adressen der Speicherblöcken zu den physischen. Durch die Besonderheit des Flashspeichers, wie zum Beispiel dem Bedürfnis eine Zelle zu löschen bevor sie geschrieben werden kann, kann dieses Mapping häufig geändert werden. Die Besonderheit der Garbage Collection ist, dass je nach Granularität in den löschbaren Einheit der Speicherblöcke mehrere Dateien gespeichert sein können. Nach der Änderung eines Teils der Daten aus einem Block, können also bestimmte Teile veraltet sein. Im Rahmen der Garbage Collection liegt also der Fokus darauf, veraltete Daten effizient auszusortieren ohne die I/O-Performance zu verschlechtern, um nicht unnötig Speicherplatz zu verbrauchen. Im Rahmen des Wear-Levelings wird versucht den Verschleiß aller Speicherzellen auf einem gleichmäßigem Stand zu halten, bedingt durch die beschränkte Haltbarkeit dieser Zellen.

Die meisten Dateisysteme werden dabei nicht direkt auf Flashspeicher zugeschnitten und führen diese Aufgaben nicht selbst aus, sondern verwenden dafür als Schnittstelle den sogenannten File Translation Layer (FTL) der die zuvor erwähnten Aufgaben umsetzt und dem Dateisystem eine definierte Schnittstelle zur Verfügung stellt ohne den Zugriff auf die ,,raw" Speicherzellen zu gewähren. Im folgenden Stellen wir zwei aktuelle Dateisysteme vor und Evaluieren ihre Performance in einem Benchmark-Test. Zu den Dateisystemen zählt das Flash-Dateisystem F2FS und dem Copy-On-Write Dateisystem Btrfs.

2 Btrfs

TODO

3 F2FS

F2FS ist ein Dateisystem speziell nur für Flashspeicher und wurde von Samsung 2012 veröffentlicht. Entwickelt wurde das Dateisystem mit dem Ziel der Optimierung von Performance und Lebenszeit von Flashspeichern, indem die speziellen Charakteristika berücksichtigt wurden. Dabei wurde das System für Linux als Open-Source Projekt entwickelt und ist seit dem Kernel 3.8 verwendbar, klassische Anwendungsbereiche stellen SD-Karten, embedded multimedia cards (eMMC) und klassische SSDs dar. Unter dem Android-System findet es zum Beispiel Verwendung in Smartphons von Huawei und dem Galaxy Note 10 von Samsung. Im Vergleich zu anderen Flash-Dateisystemen, kann F2FS allerdings nicht direkt auf den ,,raw" Speicherzellen arbeiten, sondern nutzt dafür den FTL des Speichers. Im Gesamten verfügt F2FS über viele Möglichkeiten das System an die benötigt Einsatzumgebung anzupassen. Grundlegend wird dabei das UNIX-Prinzip der Inodes und Datenblöcke zur Speicherung verwendet, zusätzlich wird dabei noch der Ansatz eines Log-structured File Systems (LFS), im Sinne eines append-only loggings, verfolgt.

Im Folgenden werden einige Besonderheiten des F2FS Dateisystems aufgezeigt, welche das System charakterisieren, dazu zählen das On-Disk-Layout, das Multi-Head Logging, die Indexstruktur und die Garbage Collection.

Das On-Disk-Layout orientiert sich von der Blockgröße an der Größe von 4KB. Im Gesamten werden aller Speicherzellen in Blöcke dieser Größe unterteilt. Die weiteren übergeordneten Einheiten sind Segmente, Sections und Zonen. Im Gesamten ist das Volumen in 2 Bereiche aufgeteilt, die Metadaten und der Hauptbereich. Zu den Metadaten zählt der Superblock mit allgemeinen Informationen über das Dateisystem. Die Checkpoints des im Rahmen des LFS werden in einem einzelnen Bereich im Prinzip einer Schattentechnik gespeichert. Die Segment-Information-Table wird für jedes Segment gespeichert, welche beinhalteten Blöcke noch gültig sind oder ob er im Rahmen der Garbage Collection gereinigt werden könnte. In der Segment Summary Area werden sogenannte Informationen für alle Blöcke eines Segments gesichert, sie beinhalten Informationen wie zu welcher Datei jeder Block gehört. Die Node-Address-Table ist eine Block-Adressen-Tabelle, welche für die physischen Adressen jedes Speicherblocks die logischen Adresse kennt. Durch diese Tabelle wird das sogenannte wandernde Baum Problem, welches beim Ändern von Daten entsteht, bekämpft. Verweise in Blöcken zeigen in F2FS daher nicht auf die physischen Blöcke, sondern speichern eine logische Adresse und erfragen die physische über die NAT. So müssen bei der Adressänderung eines Blocks nicht mehr alle Verweise angepasst werden, was einen hohen Schreibaufwand bedeutet, sondern nur die Zuordnung innerhalb der NAT. Zusätzlich wird der Metadaten Teil nicht im Sinne eines LFS verwaltet, sondern In-Place. Um den Schreibaufwand bei diesen Random-Writes auf bestimmten Speicherzellen nicht zu überlasten, werden häufig geschriebene Daten wie die NAT in einen flüchtigen Speicher gehalten und nur zum setzten eines Checkpoints auf die Speicherzellen geschrieben.

Im Hauptspeicher sind die iNodes und Datenblöcke abgelegt, dabei wird hier im Sinne eines LFS vorgegangen. Allerdings liegen diese in verschiedenen Bereichen, denn F2FS verwendet den Ansatz eines Multi-Head Loggings und hat sechs parallele Logsegmente, deren parallel Verwendung durch die klassische Flash-Architektur möglich ist. Die Idee dabei ist, die Daten, iNodes und Datenblöcke, in Klassen hot/warm/cold zu klassifizieren, in Bezug auf ihre Update-Frequenz. Dadurch soll ein häufiges neuschreiben von validen Teilsegmenten vermieden werden, wenn nur Teil des Segments veraltet sind. Ebenfalls wird dadurch die Anzahl der veralteten Daten reduziert und die Garbage Collection vereinfacht.

Die Garbage Collection (GC) wird auf zwei Arten durchgeführt, On-Demand und im Hintergrund. On-Demand wird dabei verwendet, wenn nicht genügend Speicherplatz zum schreiben verfügbar ist. In diesem Sinne wird dann, mit einem Greedy Algorithmus, auf Basis der wenigsten gültigen Blöcken ein Opfersegment ausgesucht. Im Hintergrund wird die GC mit einen Kosten-Effizienten Algorithmus bei einer geringen Auslastung des Dateisystems automatisch

ausgeführt. Dieser wählt Opfersegmente auf Grund ihres Alters und der Anzahl der gültigen Blöcke aus. Die Blöcke, welche noch gültig sind, werden dann wie bei einem normalen Schreiben wieder in die aktiven Segmente eingereiht und im Anschluss geschrieben.

4 Benchmark

Literatur