Dateisysteme für Flash-Speicher

Proseminar Speicher- und Dateisysteme
Sommersemester 2012

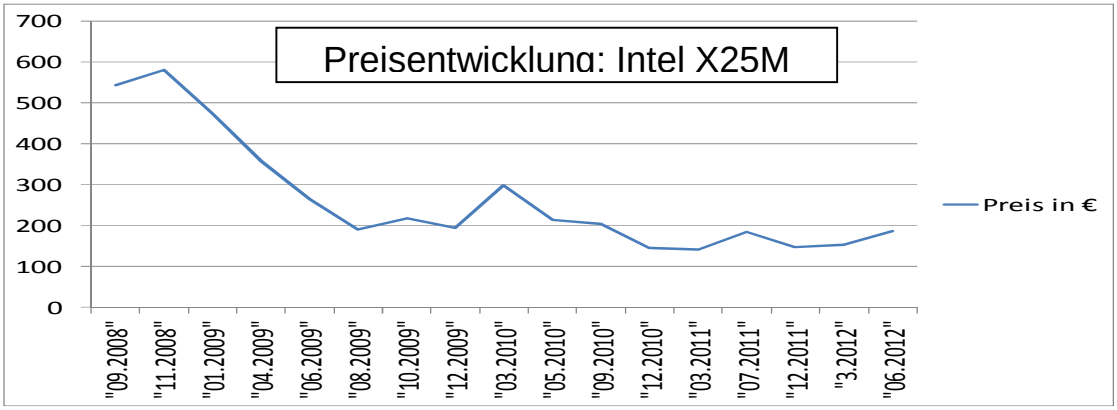
Betreuer: Michael Kuhn, Michaela Zimmer

Verfasser: Jannik Schröder

Inhaltsverzeichnis

1.Einleitung

Heutzutage sind die Kosten für Speicherplatz auf dem Niveau, dass man sich für den Privatgebrauch ausreichend Speicherplatz zulegen kann (0,05 Cent/GB)[1]; zudem sind sie so gefächert, dass man sich als Privatnutzer auch genügend Speicherplatz in Form von HDDs zulegen kann. Die HDD hat jedoch den Nachteil, dass sie relativ langsame Zugriffszeiten bietet, Probleme mit Erschütterungen und Stößen hat und relativ groß ist. Diese Nachteile fallen besonders im mobilen Bereich auf, da man dort großen Wert auf schnelle Startzeiten, geringe Größe und Robustheit legt.
In diesem Bereich wird daher vor allem Flash-Speicher verwendet; aber auch im privaten Bereich wird Flash-Speicher vor allem auf Grund der fallenden Preise (siehe Grafik: Preisentwicklung Intel X25M 80GB[2]) interessant. Dennoch befindet sich Flash-Speicher bisher in einem Stadium, in dem man noch nicht das Maximum an Leistung herausgeholt hat. Dieses liegt zum einen auf der Betriebssystems-/ Softwareseite, zum anderen auf der technischen Seite.
Daher werde ich mich in dieser Ausarbeitung zum Thema "Dateisysteme für Flash-Speicher" damit beschäftigen, wie versucht wird, dieses Problem auf der Betriebssystems-/ Softwareseite zu lösen. Zum besseren Verständnis werde ich mit der Geschichte des Flash-Speichers und den technischen Grundlagen von Flash-Speichern beginnen und später zu der Entwicklung verschiedener Dateisysteme übergehen, welche die Besonderheiten von Flash-Speicher auszunutzen. Den Abschluss wird ein Fazit bilden.



[1] vgl.: http://www.alternate.de/html/product/Seagate/ST2000DM001_2_TB/965390/?
[2] eigene Grafik erstellt aus www.gh.de

2.Allgemeines über Flash-Speicher

2.1 Geschichte des Flash-Speichers

Abb: Von Links nach Rechts: SD-Karte, CF-Karte, xD-Karte, Memory Stick, Usb-Stick [1]

Die Entwicklung des Flash-Speichers ist eng verbunden mit der Entwicklung von Digitalkameras, da es dort besonders auf schnelle Datenübertragung und Robustheit des Speichermediums ankommt.
1975 entstand die erste richtige Digitalkamera, welche als Speichermedium eine Digitalkassette nutze, welche jedoch relativ groß war.[2] Aus diesem Grund wurde die Forderung nach einem alternativen Speichermedium laut. Im Jahr 1984 entstand dann die Bezeichnung Flash-Speicher, welche im Zielmarkt Notebooks mit 2,5- und 1,8-Zoll-Festplatten anvisierte.[3]
Mit 600$ für 32 Gigabyte war es jedoch immer noch zu teuer, um die HDD verdrängen zu können. Die preisliche Lage[4] verbesserte sich jedoch weiter, sodass seit Anfang 2012 die Preislage so ist, dass die SSD als Festplatte für Betriebssysteme oder sogar als einziger Speicher in z.B. Ultrabooks[5] eingesetzt wird.
Flash-Speicher wird heute vor allem in Speicherkarten, SSDs, Hybridfestplatten, eingebetteten Systemen und für Firmware benutzt.
Laut einer Anekdote soll der Name "Flash-Speicher" im Entwicklungslabor von Toshiba entstanden. Shoji Ariizumi, ein Mitarbeiter, soll der Löschvorgang des Speichers an einen Kamerablitz erinnert haben, daher soll er den Namen Flash vorgeschlagen haben.[6]

---

[1] Bilder von www.amazon.de
[2] siehe: http://de.wikipedia.org/wiki/Digitalkamera
[3] siehe: http://de.wikipedia.org/wiki/Solid-State-Drive
[4] siehe Grafik aus der Einleitung
[5] Ultrabook: eingetragenes Warenzeichen von Intel für besonders kleine und leichte Notebooks mit Intel-Prozessoren. siehe dazu auch: http://de.wikipedia.org/wiki/Ultrabook
[6] siehe: http://de.wikipedia.org/wiki/Flash-Speicher

Die erste SSD wurde im Jahr 1985 in einem IBM PC verbaut, damals war die SSD Technik jedoch noch so teuer, dass sie nur für das Militär interessant war. Die erste Flashspeicherkarte war eine CompactFlash-Karte, hatte vier Megabyte und wurde im Jahr 1994 von SanDisk vorgestellt. In Form von Speicherkarten und Usb-Sticks wurde Flash-Speicher in den nächsten Jahren populär.
Die Entwicklung von Flash-Speicher als Festplatten für private Anwender begann im Jahr 1996, als M-Systems eine SSD vorstellte. Preislich war diese jedoch so teuer, dass sie dem Militär und wenigen preissensitiven Märkten vorbehalten war.[1] Bis 2006 blieben die SSDs auf diesem preislichen Niveau, bis Samsung im März 2006 ein Modell fertigte, das mit einem Achtel des Preises auf den Heimnutzer abzielte.

2.2 Funktionsweise

Flash-Speicher gehören zur Gruppe der Electrically Erasable Programmable Read-Only Memorys (EEPROM, wörtlich übersetzt: elektrisch löschbarer programmierbarer Nur-Lese-Speicher); dieses sind elektronische Speicherbausteine, auf denen Daten nichtflüchtig gespeichert werden können. EEPROMs haben in der Computertechnik die EPROMs(Erasable Programmable Read-Only Memorys) abgelöst, bei denen Daten nur Mittels UV-Licht gelöscht werden konnten. Bei EEPROMs ist dieses nun elektrisch (Electrically) möglich. Flash-EEPROMs unterscheiden sich zu gewöhnlichen EEPROM darin, dass die kleineste adressierbare Speichereinheit nicht einzeln löschbar ist.[2]

Abb: Foto eines Flash-EEPROM-ICs (links) und eines EPROM-ICs (rechts).[3]

[1] siehe: http://de.wikipedia.org/wiki/Solid-State-Drive
[2] siehe: http://de.wikipedia.org/wiki/Electrically_Erasable_Programmable_Read-Only_Memory
[3] siehe: http://upload.wikimedia.org/wikipedia/commons/6/65/E-eprom.jpg

Die Speicherung in einer Speicherzelle findet bei Flash-EEPROMs in Form von elektrischer Ladung auf einem speziellen Transistor statt.
Dieser Transistor ist bei Flash-Speicher ein Metall-Oxid-Halbleiter-Feldeffekttransistor(MISFET).[1]

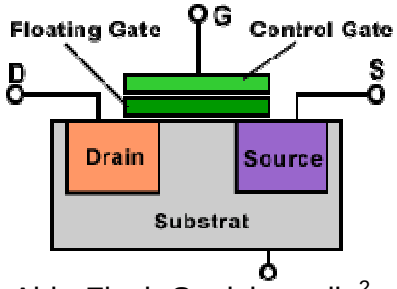


Abb: Flash-Speicherzelle[2]

Diese Speicherzelle ist dem gewöhnlichen Feldeffekttransistor ähnlich, bei dem die von Source zu Drain fließende Strommenge durch ein Control Gate reguliert wird. Bei Flash-Speicherzellen kommt jedoch noch ein Floating Gate hinzu, welches durch eine Oxid-Schicht isoliert ist und auf dem Elektronen gespeichert werden können. Durch den Tunneleffekt können Elektronen auf den Floating Gate "tunneln", dabei wird eine hohe positive Spannung an das Control Gate angelegt. Das Entladen funktioniert durch Anlegen einer negativen Spannung. Allerdings wird beim Löschen die Oxid-Schicht beschädigt, was dafür sorgt, dass ein Flash-Speicherchip ausfällt, wenn die Oxid-Schicht zu dünn wird und die Elektronen auf Grund dessen nicht mehr im Floating-Gate gehalten werden können. [3]
Wird nun eine Spannung angelegt, fließt abhängig von der Ladung des Floating Gates eine Spannung zwischen Source und Drain.
Diese Spannung wird mittels eines Cell-Sensors gemessen und muss danach noch als 0 oder 1 interpretiert werden.

2.3 Architekturen

Man unterscheidet bei Flash-Speicher die 2 verschiedenen Grundarchitekturen NAND und NOR. Bei der NAND-Architektur sind die Speicherzellen in einer Reihenschaltung angeordnet und daher nur blockweise les- und löschbar. Gegenüber NOR- haben NAND-Chips jedoch nur 2/5 des Flächenbedarfs, was auch dafür sorgt, dass

[1] siehe: http://de.wikipedia.org/wiki/Electrically_Erasable_Programmable_Read-Only_Memory
[2] siehe: http://www.elektronik-kompendium.de/sites/com/bilder/09040611.gif
[3] siehe: http://www.elektronik-kompendium.de/sites/com/0312261.htm

NAND-Chips mehr Speicherplatz als NOR-Chips bieten können. Da die Fertigung zudem einfacher ist, sind NAND-Chips kostengünstiger. Allerdings bestehen bei der Chipfertigung schon beschädigte Speicherblöcke, deren Position durch den Hersteller als defekt markiert werden und später von der Treibersoftware berücksichtigt werden müssen. NAND-Chips finden vor allem Anwendung bei Speicherkarten und SSDs.[1]
NOR-Speicherblöcke sind in Parallelschaltung angeordnet und bieten daher wahlfreien Lesezugriff. Löschen ist hier allerdings, ebenso wie bei NAND-Chips, nur blockweise möglich. Die Fertigung ist komplizierter und die NOR Architektur erfordert mehr Platz; daher sind NOR-Chips teurer als NAND-Chips. Allerdings sind NOR-Chips bei der Fertigung fehlerfrei und halten mehr Schreibzugriffe aus. Anwendung finden NOR-Chips dort, wo es auf schnelle Lese-Geschwindigkeiten und Code-Ausführung ankommt, als Beispiele sind hier Microcontroller und Firmwarespeicher zu nennen.[2]

Abb: Speicherkarten[3] und SSD [4]

[1] siehe: http://de.wikipedia.org/wiki/NAND-Flash
[2] siehe: http://de.wikipedia.org/wiki/NOR-Flash
[3] siehe: http://www.slipperybrick.com/2010/04/samsung-in-production-with-new-20nm-nand-flash/
[4] siehe: http://ecx.images-amazon.com/images/I/41WpsF6EmIL._SL500_AA300_.jpg

Fig. 1 Comparison of NOR and NAND Flash

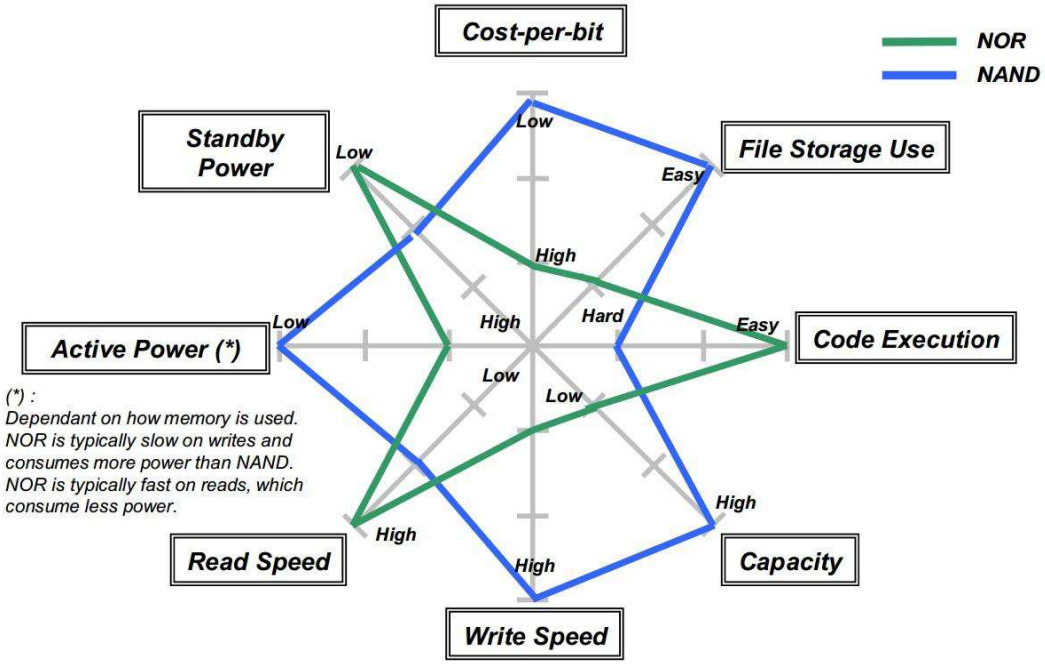


Abb: Vergleich von NAND- und NOR-Flash[1]

Weiter unterscheidet man bei Flash-Speicherzellen noch zwischen MLC(Multi-Level Cell) und SLC (Single-Level Cell). Dabei liegen die Unterschiede von MLC und SLC in der Anzahl der zu unterscheidenden Zustände. Bei MLC werden mehrere Zustände unterschieden, was jedoch auf Kosten der Geschwindigkeit geht.

[1] siehe: http://umcs.maine.edu/~cmeadow/courses/cos335/Toshiba%20NAND_vs_NOR_Flash_Memory_Technology_Overviewt.pdf

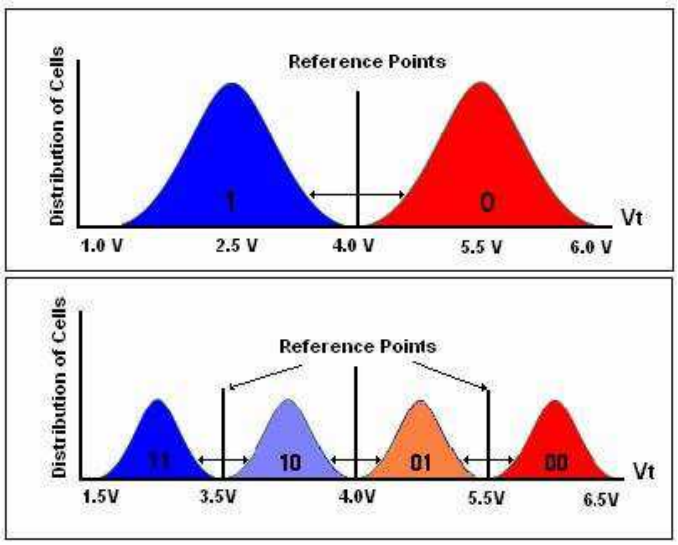


Abb.: Unterscheidung der Zuständen bei ; oben:SLC unten: MLC[1]

Die folgende Tabelle vergleicht MLC und SLC:

MLC	SLC
Hohe Speicherkapazität	Geringe Speicherkapazität
Niedrige Kosten	Hohe Kosten
Hoher Aufwand	Geringer Aufwand
Hohe Fehleranfälligkeit	Niedrige Fehleranfälligkeit
	Schneller
	Höhere Haltbarkeit

[1] siehe: http://www.oempcworld.com/support/SLC_vs_MLC.htm

3.Dateisysteme

3.1 Anforderungen

Ein Dateisystem für Flash-Speicher muss zum einen auf die unterschiedlichen Chiparten (NAND/NOR und MLC/SLC) abgestimmt sein und weiterhin dafür sorgen, dass die Besonderheiten der Flash-Technik, wie nur blockweises Löschen oder die Abnutzung der Oxidschicht, beachtet werden.
Daher sollte ein Dateisystem für Flash-Speicher Wear-Leveling, also das gleichmäßige Beschreiben aller Speicherblöcke, Bad Block Management, also das Überwachen und Markieren von beschädigten Speicherblöcken, und eine Garbage Collection, also das Verschieben von Daten um löschbare Speicherblöcke zu bilden, bieten.

3.2 Flash Translation Layer

Der Flash Translation Layer (FTL) ist dafür da, ein Block-System auf einen Flash-Speicher zu emulieren. Früher wurde dabei einfach eins zu eins übertragen, was allerdings dafür sorgte, dass bei Veränderung an einem emulierten Block der Flash-Speicherblock komplett gelesen wird und die Veränderung inklusive der alten Daten neu geschrieben wird. Allerdings war dieses sehr ineffektiv für die Haltbarkeit des Flash-Speichers. Daher ist man dazu übergegangen, den FTL als Journaling Dateisystem designed, hierbei kennt der FTL die Lage der einzelnen Dateien und ein Löschen der alten Information findet erst bei kompletter Neuspeicherung der Datei statt.
Der FTL ist heute meist im Controller von SSDs untergebracht und verfügt über alle in den Anforderungen formulierten Spezifikationen.

3.3 True FFS (True Flash File System)

Abb: Logo True FFS[1]

Das 1992 von M-Systems entwickelte True Flash File System ist für SSDs ohne Firmware konzipiert und daher eher ein FTL als ein richtiges Dateisystem. Es besitzt ein Bad Block Management, Wear Levelling und Fehlerkorrektur. Der Hersteller wirbt außerdem damit, dass True FFS 5Mio. Schreib- Lesezugriffe, ohne Ausfall der SSD, möglich macht.

3.4 Extreme FFS

Ende 2008 gab SanDisk bekannt, ein Dateisystem für Flashspeicher zu entwickeln, welches die Schreibgeschwindigkeit um den Faktor 100 erhöhen sollte. Außerdem sollte es sowohl MLC als auch SLC NAND SSDs unterstützen. Allerdings gab es seit längerer Zeit keine Neuigkeiten zu diesem Dateisystem, sodass Ungewiss ist, wie weit die Entwicklung derzeit ist.

3.5 JFFS(Journaling Flash File System)

Das 1999 von Axis entwickelte Journaling Flash File System nennt sich zwar "Journaling" ist aber eigentlich ein Log-Strukturiertes Dateisystem für Linux.
Bei Log-Strukturierten Dateisystemen ist die grundlegende Idee, dass die gesamte Platte als Log strukturiert wird und Schreibaufträge zuerst im Arbeitsspeicher gepuffert werden, um dann in Form eines kompletten Segments auf die Platte geschrieben zu werden.
Bei JFFS werden Blöcke sequentiell geschrieben und mit einer Metadatennotiz verknüpft (I-Node), welche auch die Versionsnummer hält. Gelöschte Daten werden "deleted" markiert und später von der Garbage Collection so sortiert, das die ersten Blöcke des Flash-Speichers frei geräumt werden. Diese ist schon so

[1] siehe: http://www.spezial.com/newsletter/artikel.php?AID=53&TID=514

optimiert, dass versucht wird, aus kleinen große Blöcke zu erstellen. Die Garbage Collection hat jedoch das Problem, dass immer genug Speicher zur Verfügung stehen muss, um die Daten verschieben zu können.
Ein weiteres großes Problem von JFFS ist die Mountzeit, denn hierbei muss das gesamte Medium gelesen und mit I-Nodes markiert werden. Weiter unterstützte JFFS noch keine Kompression von Daten und auch Hardlinks, also das zwei Dateinamen auf die gleiche Datei zeigen ist nicht möglich.[1]

3.6 JFFS2

JFFS2 wurde auf Grund des großen Erfolges von JFFS entwickelt.
Als Neuerung gibt es nun verschiedene Arten von I-Nodes: eines, das wie die alten I-Nodes funktioniert, ein I-Node zur Verlinkung von Dateinamen und ein I-Node, welches freie Blöcke markiert.
Dateisystemblöcke müssen nun genau so groß sein wie ein Speicherblock des Mediums, was ein Vorteil bei der Garbage Collection ist. Allerdings hat JFFS2 auch ein Problem bei der Garbage Collection, denn durch die neue Funktion der Kompression ist es möglich, dass hierbei Deadlocks entstehen, wenn ein stark komprimierter Block durch eine Änderung nicht mehr zu komprimieren ist und dadurch wesentlich größer als der Speicherblock wird. Dieses könnte zu einem Deadlock führen, was vor allem für kritische Systeme, wie Krankenhäusern, nicht akzeptabel ist. Außerdem ist die Mountzeit noch immer sehr hoch, da weiterhin das gesamte Medium gescannt wird.[2]

3.7 Log FS

Log FS ist ein noch in der Entwicklung befindliches Dateisystem. Wie der Name vermuten lässt, könnte Log FS ein log-basiertes Dateisystem sein, was aber zu Beginn der Entwicklung nicht so war, denn der Name war als Anlehnung an JFFS gedacht, welches sich auch "Journaling" nannte, allerdings kein Journaling Dateisystem war.

[1] siehe: http://en.wikipedia.org/wiki/JFFS
[2] siehe: http://en.wikipedia.org/wiki/JFFS2

Im aktuellen Stand ist man jedoch dazu gekommen, bei Log FS Journaling und Log zu kombinieren. Dabei wird die interne Struktur über einen Dateibaum dargestellt. Änderungen werden dabei nach dem "wandernden Baum" Prinzip vorgenommen.

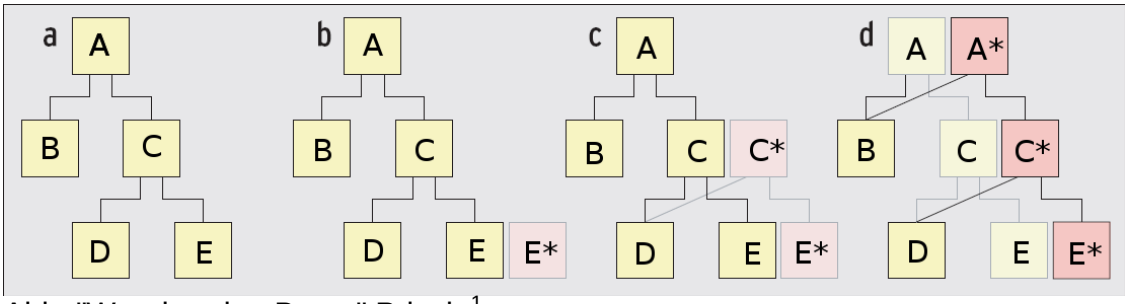


Abb: "Wandernden Baum" Prinzip[1]

Wie in der Abbildung zu sehen, soll Block E geändert werden, dabei wird im ersten Schritt Block E neu geschrieben und überprüft, welche weiteren Blöcke von C abhängen. Anschließend wird C neu geschrieben und überprüft, welche Blöcke von A abhängen; dann wird A neu geschrieben und die Verknüpfung von D zu C und von B zu A vorgenommen. Durch die Verknüpfung als letzten Arbeitsschritt ist gesichert, dass bei einem Stromausfall nicht neue und alte Datei verloren gehen.
Da die neuen Blöcke immer ans Ende des Logs geschrieben werden, entsteht dabei schon gleich ein Wear Leveling.

3.8 Flex FS

Flex FS ist ein noch in der Entwicklung befindliches Dateisystem, welches für MLC NAND Flash entwickelt wird und ausnutzt, dass MLC-Speicherzellen wie SLC-Speicherzellen genutzt werden können und dabei die gleichen Vorteile haben wie normale SLC-Speicherzellen. Dabei wird beim Mounten das gesamte Medium in MLC- und SLC-Blöcke aufgeteilt. Dadurch sinkt zwar die Kapazität des Mediums, allerdings soll durch die Kombination von MLC- und SLC-Technik die Eingabe- und Lesegeschwindigkeit stark erhöht werden. Alle neuen Dateien werden erstmal in den SLC-Bereich geschrieben und später als hot (stark genutzt) und cold (wenig

[1] siehe: http://www.fh-wedel.de/~si/seminare/ws08/Ausarbeitung/03.Flash/images/tree.png

genutzt) markiert. In ungenutzter Zeit aktiviert sich dann ein Background Migrator, welcher die als cold markierten Dateien in den MLC Bereich verschiebt. Als hot markierte Daten verbleiben im SLC bereich. Im MLC Bereich wird dann auch die Garbage Collection vorgenommen.
Flex FS hat den Nachteil, dass es keinen Energiesparmodus geben kann, da ungenutzte Zeit vom Backgroundmigrator genutzt wird.[1]

4.Fazit

Flash-Speicher und deren Dateisysteme bieten große Möglichkeiten, da sie es ermöglichen können, große Datenmengen schnell zu speichern und abzurufen.
Die Entwicklung von Dateisystemen für Flash-Speicher wird jedoch durch die Besonderheiten der Flash-Architekturen beeinflusst. Dabei sollte besonders auf eine hohe Haltbarkeit bei hohen Geschwindigkeiten geschaut werden. Wenn man sich die Entwicklung von Dateisystemen wie Flex FS anschaut, muss man auch sagen, dass Geschwindigkeitssteigerungen durch clevere Dateisysteme durchaus möglich sind.
Fraglich ist jedoch in wie weit es noch nötig ist, neue Dateisysteme zu entwickeln, da FTL in den Controllern der SSDs und bereits die meisten Funktionen der speziell für Flash-Speicher entwickelten Dateisysteme übernehmen und z.B. Windows weitere Funktionen bietet um ein Dateisystem wie NTFS für SSDs zu nutzen.

[1] siehe: http://static.usenix.org/event/usenix09/tech/full_papers/lee/lee.pdf

Quellenverzeichnis

1. http://sourceware.org/jffs2/jffs2-html/node1.html#SECTION00012000000000000000
2. http://www.rz.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaaabgqim
3. http://www.ibm.com/developerworks/linux/library/l-flash-filesystems/
4. http://www.itwissen.info/definition/lexikon/Flash-Speicher-flash-memory.html
5. http://www.youtube.com/watch?v=LlX69Mpmqko
6. http://en.wikipedia.org/wiki/Flash_file_system
7. http://en.wikipedia.org/wiki/Flash_memory
8. http://static.usenix.org/event/usenix09/tech/full_papers/lee/lee.pdf
9. http://de.wikipedia.org/wiki/Solid_State_Drive
10. http://www.flash-speicher.at/
11. http://www.alternate.de/html/product/Seagate/ST2000DM001_2_TB/965390/?
12 eigene Grafik erstellt aus www.gh.de
13. www.amazon.de
14. http://de.wikipedia.org/wiki/Digitalkamera
15. http://de.wikipedia.org/wiki/Electrically_Erasable_Programmable_Read-Only_Memory
16. http://www.elektronik-kompendium.de/sites/com/0312261.htm
17. http://de.wikipedia.org/wiki/NAND-Flash
18. http://en.wikipedia.org/wiki/JFFS
19. http://en.wikipedia.org/wiki/JFFS2